property, when the determining indicates a match between the prefix of the CSS property and the established value; and

sending the modified CSS to the requesting client computing device.

12. The apparatus as set forth in claim 11, wherein the processor is further configured to execute programmed instructions stored in the memory further comprising:

determining whether there is a transformation function for the CSS property based on the identified type of browser and the name of the CSS property, when the determining indicates a match between the prefix of the CSS property and the established value; and

removing the CSS property, when it is determined that there is not a transformation function for the CSS property.

13. The apparatus as set forth in claim 11, wherein the CSS property further comprises at least one value following the name of the CSS property and the transformation function is further configured to replace the prefix or the value of the CSS property.

14. The apparatus as set forth in claim 11, wherein the processor is further configured to execute programmed instructions stored in the memory further comprising:

determining whether the at least one CSS includes one or more other CSS properties; and

repeating the determining whether the prefix of the one or more other CSS properties matches the established value and the applying the transformation function for the CSS property for each of the one or more other CSS properties prior to sending the modified CSS to the requesting client computing device.

15. The apparatus as set forth in claim 11, wherein the CSS property is a CSS rule or a CSS extension function.

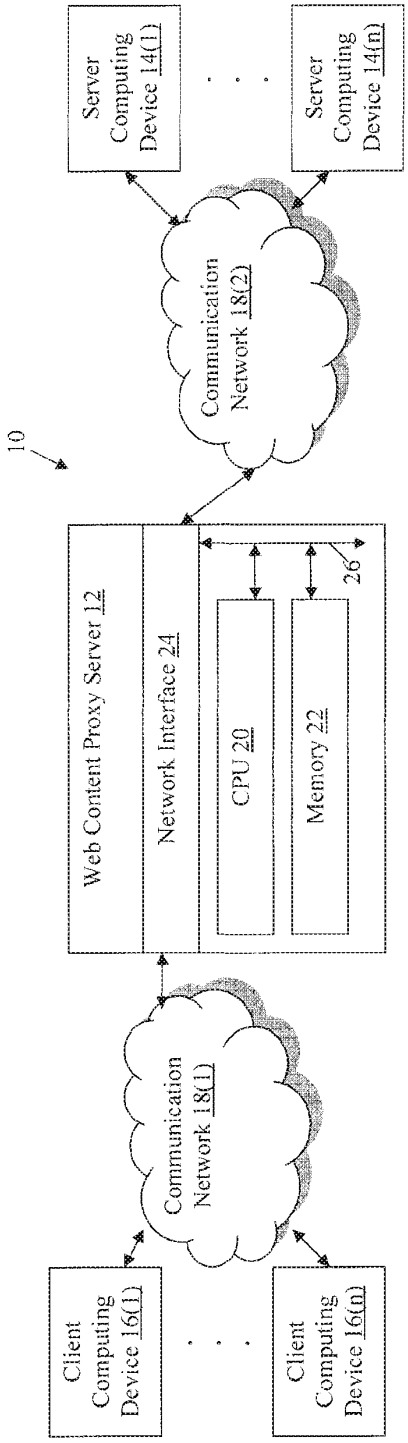

Server Computing Device 14(1)
Server Computing Device 14(n)
Communication Network 18(2)
10
Web Content Proxy Server 12
Network Interface 24
CPU 20
Memory 22
26
Communication Network 18(1)
Client Computing Device 16(1)
Client Computing Device 16(n)

FIG. 1

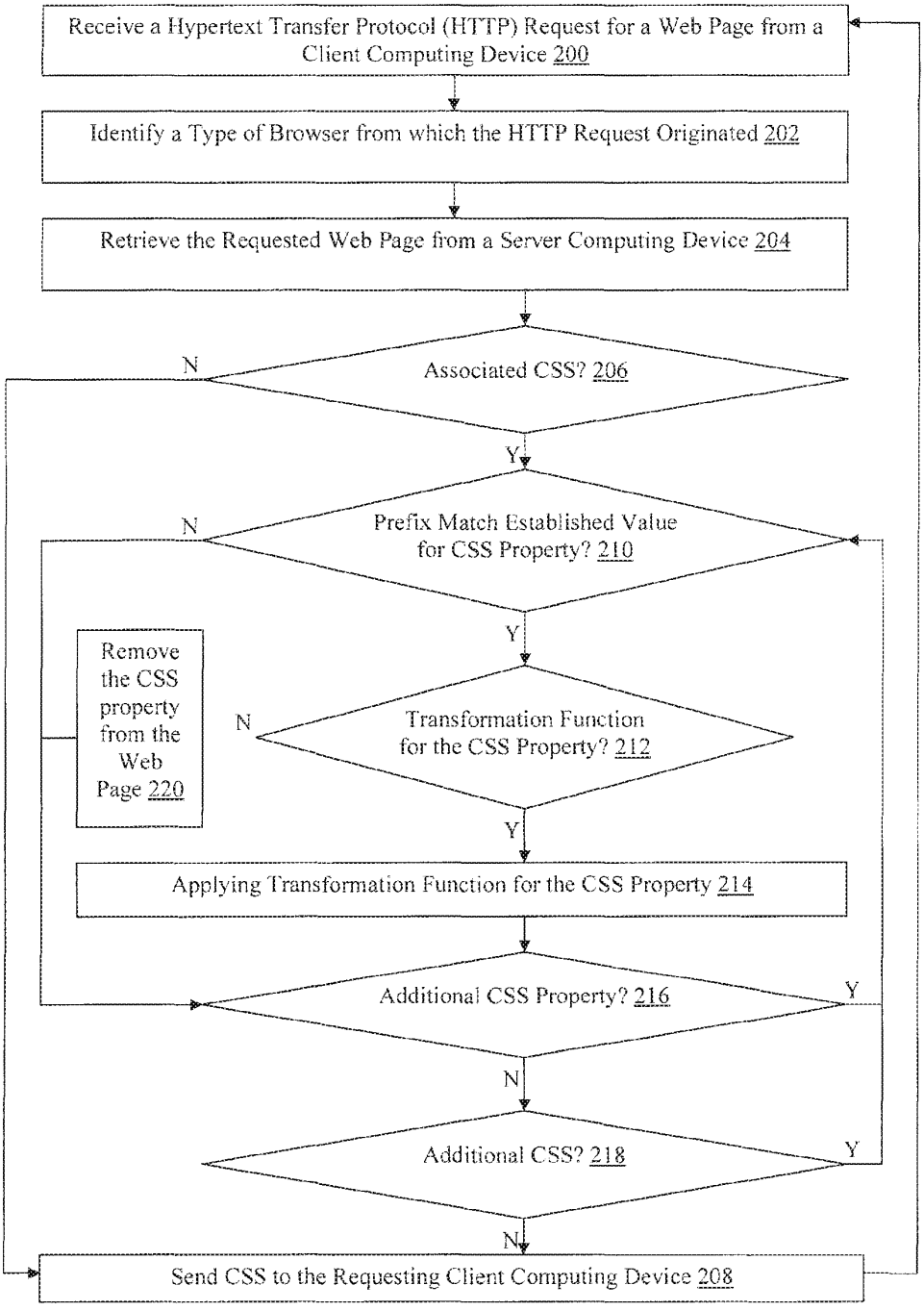


FIG. 2

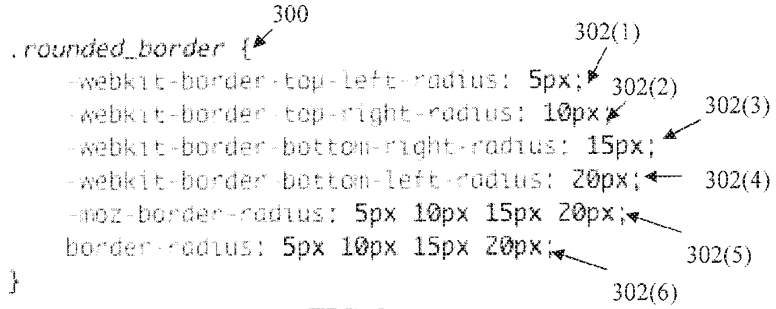


FIG. 3

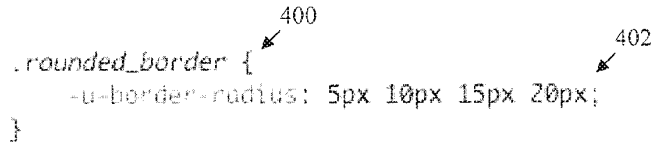


FIG. 4

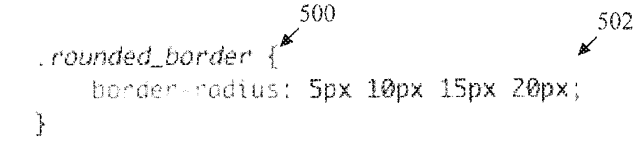


FIG. 5

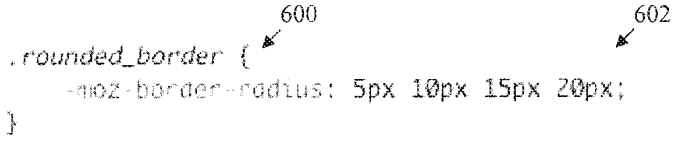


FIG. 6

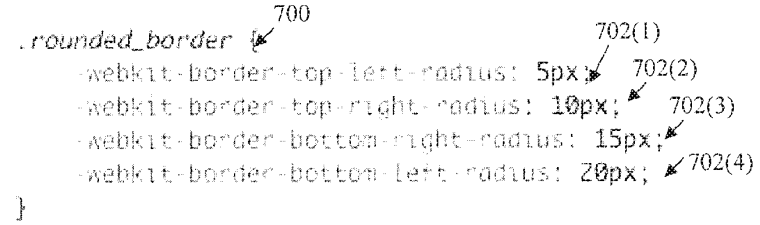


FIG. 7

Server Computing Device 14(1)
Server Computing Device 14(n)
Communication Network 18(2)
10
Web Content Proxy Server 12
Network Interface 24
CPU 20
Memory 22
26
Communication Network 18(1)
Client Computing Device 16(1)
Client Computing Device 16(n)

Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2845279 A1 2014/09/13

(21) 2 845 279

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(22) Date de dépôt/Filing Date: 2014/03/07

(41) Mise à la disp. pub./Open to Public Insp.: 2014/09/13

(30) Priorité/Priority: 2013/03/13 (US13/801,821)

(51) Cl.Int./Int.Cl. *H04L 12/16* (2006.01), *G06F 17/00* (2006.01), *G06F 17/20* (2006.01), *G06F 5/00* (2006.01), *G06F 9/44* (2006.01), *H04W 4/00* (2009.01)

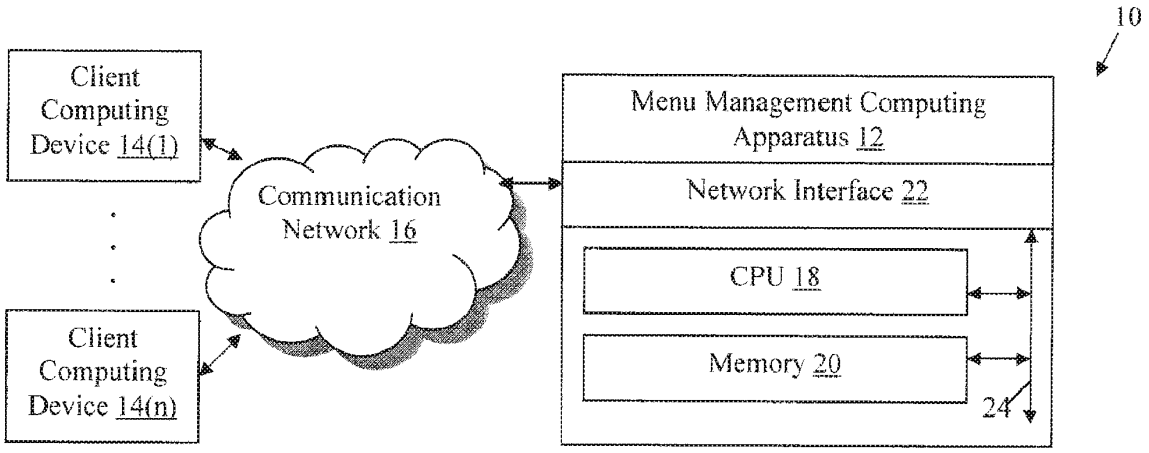
(71) Demandeur/Applicant:
USABLENET INC., US

(72) Inventeurs/Inventors:
SCODA, ENRICO, IT;
BRONDANI, MARCO, IT

(74) Agent: PARLEE MCLAWS LLP

(54) Titre : PROCEDES POUR COMPRIMER DES MENUS DE PAGES WEB ET DISPOSITIFS DE CEUX-CI

(54) Title: METHODS FOR COMPRESSING WEB PAGE MENUS AND DEVICES THEREOF



(57) Abrégé/Abstract:
A method, non-transitory computer readable medium, and apparatus that obtains an original menu associated with a web page requested by a client computing device. The original menu comprises at least a plurality of URLs, each comprising one or more fragments, and a plurality of titles, each comprising one or more words. A unique index for one occurrence of each of at least a subset of the fragments and the words is generated. At least one dictionary comprising the generated indices associated with a corresponding one of the fragments or the words is generated. A modified menu is generated by replacing each occurrence of each of the at least a subset of the one or more fragments and the one or more words of the original menu with a corresponding one of the unique indices. The modified menu and the at least one dictionary are sent to the client computing device.

ABSTRACT

A method, non-transitory computer readable medium, and apparatus that obtains an original menu associated with a web page requested by a client computing device. The original menu comprises at least a plurality of URLs, each comprising one or more fragments, and a plurality of titles, each comprising one or more words. A unique index for one occurrence of each of at least a subset of the fragments and the words is generated. At least one dictionary comprising the generated indices associated with a corresponding one of the fragments or the words is generated. A modified menu is generated by replacing each occurrence of each of the at least a subset of the one or more fragments and the one or more words of the original menu with a corresponding one of the unique indices. The modified menu and the at least one dictionary are sent to the client computing device.

METHODS FOR COMPRESSING WEB PAGE MENUS AND DEVICES THEREOF

FIELD

[0001] This technology generally relates to methods, non-transitory computer readable medium, and apparatuses for compressing source code defining web page menus and, more particularly, for reducing the amount of time required to send, and the amount of memory required to store, source code defining web page menus.

BACKGROUND

[0002] Many web sites, including particularly retail web sites, have a large number of web pages organized based of categories and subcategories of content. In order to present a user with information regarding the organization of the web site, these web pages often include a hierarchical menu. The hierarchical menu allows a user to browse the categories and subcategories without leaving the current web page.

[0003] While a hierarchical menu enhances the user experience, including a hierarchical menu increases the size of the web pages. The increased size is due to the information, including category and subcategory titles and hyperlink addresses, required to present the hierarchical menu. Due to the increased size, web pages with menus require an increased amount of time to retrieve from a server.

[0004] To reduce the time required to retrieve each web page, some web site hosts store the menu source code in an external file separate from the web pages. The external file is downloaded one time at a client device and is referenced by the web pages when the menu is rendered by a web browser of the client device. While the web pages might be retrieved relatively quickly, the initial download of this external file can require a significant amount of time. Additionally, this external file stored at the client device can require a significant amount of memory. In many client devices, particularly mobile devices, such as smart phones, memory is a limited resource and utilizing a significant amount of memory to store this external file defining a menu is not desirable.

SUMMARY

[0005] A method for compressing menus includes obtaining, with a menu management computing apparatus, an original menu associated with a web page requested by a client computing device. The original menu comprises at least a plurality of uniform resource locators (URLs), each comprising one or more fragments, and a plurality of titles, each comprising one or more words. A unique index for one occurrence of each of at least a subset of the one or more fragments and the one or more words is generated with the menu management computing apparatus. At least one dictionary comprising the generated indices associated with a corresponding one of the subset of the one or more fragments or the one or more words is generated with the menu management computing apparatus. A modified menu is generated by the menu management computing apparatus by replacing each occurrence of each of the at least a subset of the one or more fragments and the one or more words of the original menu with a corresponding one of the unique indices. The modified menu and the at least one dictionary are sent with the menu management computing apparatus to the client computing device.

[0006] A non-transitory computer readable medium having stored thereon instructions for compressing menus includes machine executable code which when executed by a processor, causes the processor to perform steps including obtaining an original menu associated with a web page requested by a client computing device. The original menu comprises at least a plurality of URLs, each comprising one or more fragments, and a plurality of titles, each comprising one or more words. A unique index for one occurrence of each of at least a subset of the one or more fragments and the one or more words is generated. At least one dictionary comprising the generated indices associated with a corresponding one of the subset of the one or more fragments or the one or more words is generated. A modified menu is generated by replacing each occurrence of each of the at least a subset of the one or more fragments and the one or more words of the original menu with a corresponding one of the unique indices. The modified menu and the at least one dictionary are sent to the client computing device.

[0007] A menu management computing apparatus includes a memory and a processor coupled to the memory. The processor is configured to execute programmed instructions stored in the memory including obtaining an original menu associated with a web page requested by a client computing device. The original menu comprises at least a plurality of URLs, each comprising one or more fragments, and a plurality of titles, each comprising one or more words. A unique index for one occurrence of each of at least a subset of the one or more fragments and the one or more words is generated. At least one dictionary comprising the generated indices associated with a corresponding one of the subset of the one or more fragments or the one or more words is generated. A modified menu is generated by replacing each occurrence of each of the at least a subset of the one or more fragments and the one or more words of the original menu with a corresponding one of the unique indices. The modified menu and the at least one dictionary are sent to the client computing device.

[0008] This technology provides a number of advantages including methods, non-transitory computer readable medium, and apparatuses that reduce the amount of resources required to send and store web page menus. With this technology, a web page menu is parsed at a menu management computing apparatus and modified to replace URL fragments and words in category and subcategory titles with unique indices. A dictionary is generated to store each of the fragments and words and corresponding indices. The modified original menu and the dictionary, which together require less time to send and less memory to store than the original menu, are then sent to a client computing device that requested the web page having the associated original menu.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] FIG. 1 is a block diagram of a network environment which incorporates an exemplary menu management computing apparatus;

[0010] FIG. 2 is a flowchart of an exemplary method for compressing a web page menu;

[0011] FIG. 3 is an exemplary original web page menu; and

[0012] FIG. 4 is the exemplary original web page menu of FIG. 3 modified according to the exemplary method of FIG. 2.

DETAILED DESCRIPTION

[0013] An exemplary network environment 10 is illustrated in FIG. 1 as including an exemplary menu management computing apparatus 12. In this example, the menu management computing apparatus 12 is coupled to a plurality of client computing devices 14(1)-14(n) by communication network 16, although other types and numbers of devices, components, and elements in other topologies could be used. This technology provides a number of advantages including methods, non-transitory computer readable medium, and apparatuses that compress web page menus and reduce the amount of resources required to send and store web page menus.

[0014] Referring more specifically to FIG. 1, the menu management computing apparatus 12 includes at least one processor or central processing unit (CPU) 18, a memory 20, and a network interface 22, which are coupled together by a bus 24 or other link, although other numbers and types of components, parts, devices, systems, and elements in other configurations and locations can also be used. Generally, the menu management computing apparatus 12 processes requests for web pages and other web content received from the client computing devices 14(1)-14(n) via the communication network 16 according to the HTTP-based protocol, for example, although the menu management computing apparatus 12 can also provide other numbers and types of functions. The processor 18 in the menu management computing apparatus 12 may execute a program of stored instructions one or more aspects of the present invention, as described and illustrated by way of the embodiments herein, although the processor 18 could execute other numbers and types of programmed instructions.

[0015] The memory 20 in the menu management computing apparatus 12 stores these programmed instructions for one or more aspects of the present invention as described and illustrated herein, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a RAM

or a ROM in the system or a floppy disk, hard disk, CD ROM, DVD ROM, or other non-transitory computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to the processor 18, can be used for the memory 20 in the menu management computing apparatus 12.

[0016] The network interface 22 in the menu management computing apparatus 12 is used to operatively couple and communicate between the menu management computing apparatus 12 and the client devices 14(1)-14(n) via the communication network 16, although other types and numbers of networks with other types and numbers of connections and configurations can also be used. The menu management computing apparatus 12 may be a server computing device, such as any version of Microsoft® IIS server or Apache® server, although other types of servers may be used, or a web content proxy server, for example.

[0017] The communication network 16 can include one or more networks, such as one or more local area networks (LANs) and/or wide area networks (WANs) such as the Internet. By way of example only, the communication network 16 can use TCP/IP over Ethernet and industry-standard protocols, including Hypertext transfer protocol (HTTP), secure HTTP (HTTPS), wireless application protocol (WAP), and/or SOAP, although other types and numbers of communication networks having their own communications protocols, can also be used.

[0018] The client computing devices 14(1)-14(n) enable a user to request, receive, and interact with applications, web services, and content hosted by the menu management computing apparatus 12 via the communication network 16, although one or more of the client computing devices 14(1)-14(n) could access content and utilize other types and numbers of applications from other sources and could provide a wide variety of other functions for the user. By way of example only, the client computing devices 14(1)-14(n) can be mobile computing devices, smart phones, personal digital assistants, or computers.

[0019] In this example, each of the client computing devices 14(1)-14(n) includes at least one processor or a CPU, a memory, a network interface, a user input device, and a display, which are coupled together by a bus or other link, although one or more of client

computing devices 14(1)-14(n) can include other numbers and types of components, parts, devices, systems, and elements in other configurations. The processor in each of the client computing devices 14(1)-14(n) can execute a program of instructions stored in the memory of each of the client computing devices 14(1)-14(n) for one or more aspects of the present invention as described and illustrated herein, although the processor could execute other numbers and types of programmed instructions.

[0020] The memory in each of the client computing devices 14(1)-14(n) stores these programmed instructions for one or more aspects of the present invention, as described and illustrated herein, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a RAM or a ROM in the system or a floppy disk, hard disk, CD ROM, or other non-transitory computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to processor can be used for the memory in each of the client computing devices 14(1)-14(n). Each of the client computing devices 14(1)-14(n) can be configured to access web services and content through a web browser stored in the memory.

[0021] The network interface in each of the client computing devices 14(1)-14(n) is used to operatively couple and communicate between each of the client computing devices 14(1)-14(n) and the menu management computing apparatus 12 via the communication network 16, although other types and numbers of communication networks with other types and numbers of connections and configurations can be used.

[0022] The user input device in each of the client computing devices 14(1)-14(n) can be used to input selections, such as a request for a particular web page, although the user input device could be used to input other types of requests and data and interact with other elements. The user input device in each of the client computing devices 14(1)-14(n) can include a keypad, touch screen, and/or vocal input processing system, although other types and numbers of user input devices can also be used.

[0023] The display in each of the client computing devices 14(1)-14(n) can be used to show data and information to the user, such as a requested web page by way of example only. The display in each of the client computing devices 14(1)-14(n) can be an LCD, LED, or OLED display, for example, although other types and numbers of displays could be used depending on the particular type of client computing device 14(1)-14(n).

[0024] Although embodiments of the menu management computing apparatus 12 and client computing devices 14(1)-14(n) are described and illustrated herein, each of the menu management computing apparatus 12, and client computing devices 14(1)-14(n) can be implemented on any suitable computer apparatus or computing device. It is to be understood that the apparatuses and devices of the embodiments described herein are for exemplary purposes, as many variations of the specific hardware and software used to implement the embodiments are possible, as will be appreciated by those skilled in the relevant art(s). Furthermore, each of the devices of the embodiments may be conveniently implemented using one or more general purpose computers, microprocessors, digital signal processors, and micro-controllers, programmed according to the teachings of the embodiments, as described and illustrated herein, and as will be appreciated by those ordinary skill in the art.

[0025] In addition, two or more computing apparatuses or devices can be substituted for any one of the devices in any embodiment described herein. Accordingly, principles and advantages of distributed processing, such as redundancy and replication also can be implemented, as desired, to increase the robustness and performance of the devices of the embodiments. The embodiments may also be implemented on computer apparatuses or devices that extend across any suitable network using any suitable interface mechanisms and communications technologies, including by way of example only telecommunications in any suitable form (e.g., voice and modem), wireless communications media, wireless communications networks, cellular communications networks, G3 communications networks, Public Switched Telephone Network (PSTNs), Packet Data Networks (PDNs), the Internet, intranets, and combinations thereof.

[0026] The examples may also be embodied as a non-transitory computer readable medium having instructions stored thereon for one or more aspects of the present technology as described and illustrated by way of the examples herein, as described herein, which when executed by a processor, cause the processor to carry out the steps necessary to implement the methods of the examples, as described and illustrated herein.

[0027] An exemplary method for compressing an original menu will now be described with reference to FIGS. 1-4. Referring more specifically to FIG. 2, in step 200 the menu management computing apparatus 12 receives a request for a web page, such as a hypertext transfer protocol (HTTP) request, from one of the client computing devices 14(1)-14(n). In step 202, the menu management computing apparatus 12 obtains the requested web page from one of the menu management computing apparatus 12.

[0028] In step 204, the menu management computing apparatus 12 determines whether the web page has an associated original menu. The associated original menu can be any menu, although hierarchical menus with categories and subcategories may have repeated content allowing for relatively significant compression, as described and illustrated in more detail later. In some examples, the original menu is included in the source code for the web page and, in other examples, the web page source code references a separate external file, such as a JavaScript file, which includes the original menu.

[0029] In examples in which the original menu is defined in a separate file, the initially-requested web page source code may cause a web browser of the requesting one of the client computing device 14(1)-14(n) to submit a request for the file. The menu management computing apparatus 12 can parse the web page or file obtained from the one of the menu management computing apparatus 12 to identify any key words or patterns representing a menu (e.g. alternating plan text titles and URL hyperlinks), for example, although other methods of identifying an original menu associated with the requested web page can also be used.

[0030] If the menu management computing apparatus 12 determines in step 204 that the requested web page does not have an associated original menu, then the No branch is

taken to step 200 and another request is received from one of the client computing devices 14(1)-14(n). If the menu management computing apparatus 12 determines in step 204 that the requested web page does have an associated original menu, then the Yes branch is taken to step 206.

[0031] Optionally, in step 206, the menu management computing apparatus 12 determines whether the requesting one of the client computing devices 14(1)-14(n) is a device having relatively limited storage capabilities, referred to herein as a mobile computing device. The menu management computing apparatus 12 can determine whether the requesting one of the client computing devices 14(1)-14(n) is a mobile computing device based on a user agent header included in one or more of the packets associated with the HTTP request received in step 200, for example, although other methods of identifying mobile computing devices can also be used.

[0032] If the menu management computing apparatus 12 determines that the requesting one of the client computing devices 14(1)-14(n) is not a mobile computing device, then the No branch is taken to step 200 and another request is received from one of the client computing devices 14(1)-14(n). Accordingly, in some examples, steps 208-220 are only performed for mobile computing devices which have relatively limited storage capabilities and can therefore benefit most from the compression method described and illustrated herein.

[0033] If the menu management computing apparatus 12 determines in step 206 that the requesting one of the client computing devices 14(1)-14(n) is a mobile computing device, then the Yes branch is taken to step 208. In step 208, the menu management computing apparatus 12 parses the original menu and generates a unique index for at least a subset of the fragments and/or words of the original menu. The original menu can include a plurality of titles, each including one or more words, and a plurality of URLs, each including one or more fragments. Exemplary URL fragments include an entire URL, a prefix, a path, one or more words of a path, a query, a key/value pair of a query, a key of a query, a value of a query, or combinations thereof, for example, although other URL fragments can also be used.

[0034] Referring to FIG. 3, an exemplary original menu 300 associated with a web page is illustrated. In this example, the original menu 300 defines a hierarchical menu with categories having titles of "Shoes," "Accessories," and "Dresses." The "Shoes" and "Dresses" categories each have two sub-categories with titles of "Men Shoes" and "Women Shoes" and "Long Dresses" and "Ceremonies," respectively. In this example, each of the "Accessories" category and the four subcategories has an associated URL of a web page having content associated with the respective category or subcategory title.

[0035] Referring back to FIG. 2, in step 208, only one index is generated for multiple occurrences of the same fragment and/or word in the original menu 300. In this example, only one index is generated for the title word "Shoes" in the source code 300 even though the word appears in more than one title. Additionally, only one index is generated for the key/value pair fragment "sort=price" even though the fragment appears in all of the URLs. The menu management computing apparatus 12 can be configured to parse the original menu 300 to identify title words and URL fragments based on an established configuration in order to generate the indices.

[0036] The menu management computing apparatus 12 parses some or all of the titles of the original menu 300 and generates an index for some or all words or combination of words in each of the titles. Words can be any sequence of characters and numbers separated by a space (e.g. "Men" and "Shoes"). In one exemplary configuration, the menu management computing apparatus 12 generates an index for every encountered word in a title. Words included in titles may be repeated as the title in a lower category in a hierarchical menu is likely to share word(s) with the title in a higher category, while adding word(s) to be more specific.

[0037] The menu management computing apparatus 12 also parses some or all of the URLs of the original menu 300 and generates an index for some or all fragments or combination of fragments in each of the URLs. In one exemplary configuration, the menu management computing apparatus 12 only generates an index upon encountering a URL fragment more than once or a predetermined number of times, although indices can also be

generated by the menu management computing apparatus 12 upon initially encountering a fragment.

[0038] In the exemplary original menu 300 illustrated in FIG. 3, in this example, the menu management computing apparatus 12 will only generate a unique index for the multiple occurrences of the "sort=price" key/value pair fragment upon encountering it a second time in the URL associated with the "Women Shoes" subcategory. The menu management computing apparatus 12 can parse the original menu 300 and maintain a list of previously encountered fragments and/or words which is used to determine whether an associated index should be, or has previously been, generated.

[0039] In another example, the menu management computing apparatus 12 can be configured to parse each URL to identify the prefix (e.g. "http://www.acme.com/catalog"), path (e.g. "Men_Shoes/products"), and/or query fragments (e.g. "plp_i=2354&sort=price") that are likely to be repeated in other URLs of the source code 300. Although one example is illustrated and described here, other methods and configurations can be used to parse the URLs and titles of the original menu 300.

[0040] Optionally, the generated indices include one or more digits in base 62 format with each digit being an ASCII lower case letter, upper case letter, or number from 0-9. In one example, the menu management computing apparatus 12 uses two base 62 digits which allows the menu management computing apparatus 12 to represent 620 or 3844 numbers, although other numbers of digits and bases can also be used.

[0041] In step 210, the menu management computing apparatus 12 generates at least one dictionary including the generated indices associated with a corresponding one of the fragments or words or combinations of fragments or words. In this example, the menu management computing apparatus 12 generates a words dictionary for storing the generated unique indices for occurrences of a word, URL, path, or word of a path, and a parameters dictionary for storing the generated unique indices for occurrences of a query, key/value pair of a query, key of a query, or value of a query for example. Other numbers of dictionaries storing other fragments and/or words can also be used.

[0042] An exemplary words dictionary generated in step 210 for the source code 300 is illustrated in Table 1.

Table 1

Index	Fragment/Word
00	products
01	02/ 00
02	Accessories
03	Shoes
04	Men
05	Women
06	price
07	Dresses
08	Long
09	Ceremonies

[0043] An exemplary parameters dictionary for the source code 300 is illustrated in Table 2.

Table 2

Index	Fragment/Word
00	sort=price
01	plp_i
02	sort

[0044] Optionally, in step 212, the menu management computing apparatus 12 compresses at least a portion of one or more of the dictionaries. Also optionally, the menu management computing apparatus 12 can determine whether the requesting one of the client computing devices 14(1)-14(n) is a mobile computing device, as described and illustrated earlier in step 206, and only compress at least a portion of one or more of the dictionaries when it is determined that the requesting one of the client computing devices 14(1)-14(n) is a mobile computing device. Accordingly, in some examples, step 212 is only performed for mobile computing devices which have relatively limited storage capabilities and can therefore benefit most from a second level of compression as described and illustrated herein.

[0045] In the exemplary original menu 300 illustrated in FIG. 3, in one example, the words dictionary includes an entry of "01:Accessories/Products" which establishes an association of the base 62 number "01" with the "Accessories/Products" path fragment. In this example, the menu management computing apparatus 12 can compress the dictionary by replacing "Accessories/Products" in the dictionary entry associated with the "01" index with indices generated in step 208 for portions of the fragment (e.g. "Accessories" and "Products"). In this example, "Accessories" is associated with the 02 index and "Products" is associated with the 00 index in the dictionary. Accordingly, the menu management computing apparatus 12 can replace the "01:Accessories/Products" entry using the indices generated in step 208 (e.g. "00:02/ 00").

[0046] Alternatively, in examples in which indices were not generated for portions of a fragment in step 208, the menu management computing apparatus 12 can generate indices in step 212, store the generated indices and associations in the dictionary, and replace the fragment as described and illustrated earlier. In some examples, the menu management computing apparatus 12 can compress the dictionary for fragments and/or words having a threshold number of characters or a threshold number of portions shared with other fragments and/or words in the dictionary. By establishing a threshold, the increased time required to compress the dictionary can be balanced with the reduced size of the compressed dictionary. Other methods of compressing the dictionary can also be used.

[0047] In step 214, the menu management computing apparatus 12 generates a modified menu by replacing occurrence(s) of the fragments and/or words in the original menu 300 with a corresponding one of the indices generated in step 208. In one example, the modifications to the original menu 300 occur during an initial parsing of the original menu 300 and while the dictionary is being generated by the menu management computing apparatus 12. In another example, the menu management computing apparatus 12 parses the original menu 300 a second time, subsequent to generating the dictionary, and uses the dictionary to determine which fragments and/or words are to be replaced.

[0048] Optionally, the fragments and/or words in the original menu 300 are replaced by the longest (e.g. largest number of successive characters) corresponding fragment and/or word in the dictionary. For example, the path fragment (e.g. "Men_Shoes/products") of a URL associated with a subcategory is repeated several times in the original menu 300. Based on the configuration of the menu management computing apparatus 12, the path fragment may be included in the dictionary associated with an index. In this example, occurrences of the path of the URL are replaced by the corresponding index even when indices were generated, in step 208, and associated in the dictionary, in step 210, with one or more words of the path (e.g. "Men," "Shoes," or "products"). In this example, the dictionary entry for the path fragment can optionally be compressed using the entries of the words of the path, as described and illustrated earlier with reference to step 212, and path fragment in the original menu 300 will be replaced with a reduced number of indices.

[0049] Referring to FIG. 4, an exemplary modified menu 400 generated in step 214 is illustrated. In this example, the modified menu 400 includes "0403" in place of the "Men Shoes" subcategory title included in the original menu 300. As indicated in the exemplary words dictionary shown in Table 1, the base 62 index "04" was generated by the menu management computing apparatus 12, in step 208, and associated with the word "Men" in the dictionary, in step 210. Similarly, the base 62 index "03" was generated by the menu management computing apparatus 12, in step 208, and associated with the word "Shoes" in the dictionary, in step 210. Other indices and dictionary entries can also be used.

[0050] The source code 400 further includes " 0403/ 00?01&2354&00" in place of the "http://www.acme.com/catalog/Men_Shoes/products?plp_i=2354&sort=price" URL associated with the "Men Shoes" subcategory. The "Men_Shoes" portion of the path in the URL associated with the "Men Shoes" subcategory is replaced with "0403" which are the indices associated with the "Men" and "Shoes" words, respectively, in the words dictionary. Additionally, the "products" portion of the path and "plp_i" key fragment in the URL associated with the "Men Shoes" subcategory are replaced with "00" and "01," respectively, which are the indices associated with the "products" portion of the path and the "plp_i" key fragment in the words and parameters dictionaries, respectively. Finally, the "sort=price" key/value pair fragment in the URL associated with the "Men Shoes" subcategory is replaced with "00," which is the index associated with the "sort=price" key/value pair fragment in the parameters dictionary. The other titles and URLs in the source code 300 are similarly modified as illustrated in the modified menu 400.

[0051] In step 216, the menu management computing apparatus 12 optionally applies one or more rules to remove and/or insert spaces and/or special characters in the modified menu 400, for example, so that the titles and URLs can be properly interpreted and decompressed at the requesting one of the client computing device 14(1)-14(n). In one example in which all words in titles are replaced by a corresponding two digit base 62 index generated in step 208, the spaces or other standard separator in the titles can be removed. In this example, every two digits in a title represents a word of the title so the separators do not need to be included and can be removed to reduce the size of the modified menu 300. In the

modified menu 400, the space between indices that replaced the words in the "Men Shoes," "Women Shoes," and "Long Dresses" titles is removed.

[0052] In examples in which some fragments of a URL (e.g. words of a path) are not be replaced by an index, the menu management computing apparatus 12 can insert a space before and/or after any set of two or more sequential indices in the URL. By inserting a space, which is not a valid character inside a URL, any valid characters and/or fragments not replaced can be differentiated from any indices that replaced fragments included in the original menu 300. Other invalid characters can also be used to differentiate indices that replaced fragments in the original menu 300.

[0053] Additionally, in examples in which all key/value pair fragments are converted, a "&" or other special character separating sequential key/value pairs can be removed by the menu management computing apparatus 12. Instead, in these examples, the "&" character can be inserted by the menu management computing apparatus 12 before and/or after any value fragment of a query fragment for any of the URLs. By inserting the "&" character(s), the value can be differentiated from any indices that replaced fragments included in the original menu 300.

[0054] In the modified menu 400, the value fragment "2354" of the URL associated with the "Men Shoes" subcategory is preceded and followed by a "&" character, which differentiates the "2354" value fragment from the "01" and "00" indices that precede and follow, respectively, the "2354" value fragment. Other special characters can also be used to differentiate indices that replaced fragments in the original menu 300.

[0055] Because all of the URLs have the same prefix fragment of "http://www.acme.com/catalog" in the original menu 300, the prefix fragment is removed and replaced by a space in the modified menu 400. The prefix fragment could also be replaced by an index value or a special character, for example. Additionally, the "_" special character can be removed from any of the URLs since two indices (e.g. "0403") representing two words of a path fragment are always separated by the "_" character as a space is an

invalid character in a URL and the words would be represented by only one index if there were no "_" character between them.

[0056] Other characters (e.g. "/" and "?") are not removed from the URLs in this example since they are valid characters and are not digits in the base 62 format used for the indices. Other rules for modifying titles and/or URL fragments with respect to spaces and special characters can also be used.

[0057] Referring back to FIG. 2, in step 218, the menu management computing apparatus 12 sends the modified menu 400 and the dictionary to the requesting one of the client computing devices 14(1)-14(n). In step 220, the menu management computing apparatus 12 sends decompression source code to the requesting one of the client computing devices 14(1)-14(n). The decompression source code, when executed by the requesting one of the client computing devices 14(1)-14(n), converts the modified menu 400 into the original menu 300 using the dictionaries generated in step 210, and sent to the requesting one of the client computing devices 14(1)-14(n) in step 218, and based on the rules applied in step 216.

[0058] Accordingly, the decompression source code, sent by the menu management computing apparatus 12 with the dictionaries and the modified menu 400, is configured to parse the modified menu 400, replace encountered indices with corresponding URL fragments and/or title words in the dictionaries, and reverse the results of the rules applied in step 216. For example, the decompression source code can be configured to inset a space following each word in a title that is converted. In another example, a "_" special character in a path fragment of a URL is inserted by the menu management computing apparatus 12 following any each word of the path that is converted.

[0059] The decompression source code can be configured to reverse the result of applying one or more other rules and/or perform other functions in order to convert the modified menu 400 into the original menu 300. Additionally, the decompression source code can be stored in the memory 20 of the menu management computing apparatus 12 and sent to

a requesting one of the client computing devices 14(1)-14(n) following, or along with, the modified menu 400 and the dictionaries.

[0060] Accordingly, as illustrated and described herein this technology provides a number of advantages including methods, non-transitory computer readable medium, and apparatuses that reduce the storage requirements for, and time required to retrieve, web page menus. With this technology, a web page menu is compressed using at least one dictionary to store indices corresponding to word(s) in menu titles and/or fragment(s) in URLs of the menu. The compressed menu is sent to a requesting client computing device along with the dictionaries and decompression source code configured to convert the modified menu into the original menu. Advantageously, the modified menu, dictionaries, and decompression source code can be sent in less time and are smaller in size than the original menu.

[0061] Having thus described the basic concept of the invention, it will be rather apparent to those skilled in the art that the foregoing detailed disclosure is intended to be presented by way of example only, and is not limiting. Various alterations, improvements, and modifications will occur and are intended to those skilled in the art, though not expressly stated herein. These alterations, improvements, and modifications are intended to be suggested hereby, and are within the spirit and scope of the invention. Additionally, the recited order of processing elements or sequences, or the use of numbers, letters, or other designations therefore, is not intended to limit the claimed processes to any order except as may be specified in the claims. Accordingly, the invention is limited only by the following claims and equivalents thereto.

CLAIMS

What is claimed is:

1. A method for compressing menus, the method comprising:

obtaining, with a menu management computing apparatus, an original menu associated with a web page requested by a client computing device, the original menu comprising at least a plurality of uniform resource locators (URLs), each comprising one or more fragments, and a plurality of titles, each comprising one or more words;

generating, with the menu management computing apparatus, a unique index for one occurrence of each of at least a subset of the one or more fragments and the one or more words;

generating, with the menu management computing apparatus, at least one dictionary comprising the generated indices associated with a corresponding one of the subset of the one or more fragments or the one or more words;

generating, with the menu management computing apparatus, a modified menu by replacing each occurrence of each of the at least a subset of the one or more fragments and the one or more words of the original menu with a corresponding one of the unique indices; and

sending, with the menu management computing apparatus, the modified menu and the at least one dictionary to the client computing device.

2. The method of claim 1, wherein the one or more fragments comprise one or more of a URL, prefix, path, word of a path, query, key/value pair of a query, key of a query, or value of a query.

3. The method of claim 1, wherein the at least one dictionary comprises a words dictionary for storing the generated unique indices for occurrences of a word, URL, prefix, path, or word of a path and a parameters dictionary for storing the generated unique indices for occurrences of a query, key/value pair of a query, key of a query, or value of a query.

4. The method of claim 1, further comprising:

compressing, with the menu management computing apparatus, at least a portion of the at least one dictionary; and

sending, with the menu management computing apparatus, to the client computing device, source code configured to convert the modified source code into the original source code using the at least one dictionary.

5. The method of claim 1, wherein the subset of the one or more fragments includes only one or more of the fragments occurring more than once in the plurality of URLs.

6. The method of claim 1, wherein the unique indices comprise one or more digits in base 62 format.

7. The method of claim 1, wherein the modifying further comprises:

removing any separator between words for any of the plurality of titles comprising two or more words;

inserting a space before or after any two or more sequential indices for any of the URLs; and

inserting a special character before or after any value of a query for any of the URLs.

8. The method of claim 1, further comprising:

determining, with the menu management computing apparatus, whether the client computing device is a mobile computing device; and

only performing the generating and sending steps when it is determined that the client computing device is a mobile computing device.

9. A non-transitory computer readable medium having stored thereon instructions for compressing menus comprising machine executable code which when executed by a processor, causes the processor to perform steps comprising:

obtaining an original menu associated with a web page requested by a client computing device, the original menu comprising at least a plurality of uniform resource locators (URLs), each comprising one or more fragments, and a plurality of titles, each comprising one or more words;

generating a unique index for one occurrence of each of at least a subset of the one or more fragments and the one or more words;

generating at least one dictionary comprising the generated indices associated with a corresponding one of the subset of the one or more fragments or the one or more words;

generating a modified menu by replacing each occurrence of each of the at least a subset of the one or more fragments and the one or more words of the original menu with a corresponding one of the unique indices; and

sending the modified menu and the at least one dictionary to the client computing device.

10. The medium of claim 9, wherein the one or more fragments comprise one or more of a URL, prefix, path, word of a path, query, key/value pair of a query, key of a query, or value of a query.

11. The medium of claim 9, wherein the at least one dictionary comprises a words dictionary for storing the generated unique indices for occurrences of a word, URL, prefix, path, or word of a path and a parameters dictionary for storing the generated unique indices for occurrences of a query, key/value pair of a query, key of a query, or value of a query.

12. The medium of claim 9, further comprising machine executable code which, when executed by the processor, causes the processor to perform steps further comprising:

compressing at least a portion of the at least one dictionary; and

sending to the client computing device source code configured to convert the modified source code into the original source code using the at least one dictionary.

13. The medium of claim 9, wherein the subset of the one or more fragments includes only one or more of the fragments occurring more than once in the plurality of URLs.

14. The medium of claim 9, wherein the unique indices comprise one or more digits in base 62 format.

15. The medium of claim 9, wherein the modifying further comprises:

removing any separator between words for any of the plurality of titles comprising two or more words;

inserting a space before or after any two or more sequential indices for any of the URLs; and

inserting a special character before or after any value of a query for any of the URLs.

16. The medium of claim 9, further comprising machine executable code which, when executed by the processor, causes the processor to perform steps further comprising:

determining whether the client computing device is a mobile computing device; and

only performing the generating and sending steps when it is determined that the client computing device is a mobile computing device.

17. A menu management computing apparatus, comprising:

a memory; and

a processor coupled to the memory and configured to execute programmed instructions stored in the memory, comprising:

obtaining an original menu associated with a web page requested by a client computing device, the original menu comprising at least a plurality of uniform resource locators (URLs), each comprising one or more fragments, and a plurality of titles, each comprising one or more words;

generating a unique index for one occurrence of each of at least a subset of the one or more fragments and the one or more words;

generating at least one dictionary comprising the generated indices associated with a corresponding one of the subset of the one or more fragments or the one or more words;

generating a modified menu by replacing each occurrence of each of the at least a subset of the one or more fragments and the one or more words of the original menu with a corresponding one of the unique indices; and

sending the modified menu and the at least one dictionary to the client computing device.

18. The apparatus of claim 17, wherein the one or more fragments comprise one or more of a URL, prefix, path, word of a path, query, key/value pair of a query, key of a query, or value of a query.

19. The apparatus of claim 17 wherein the at least one dictionary comprises a words dictionary for storing the generated unique indices for occurrences of a word, URL, prefix, path, or word of a path and a parameters dictionary for storing the generated unique indices for occurrences of a query, key/value pair of a query, key of a query, or value of a query.

20. The apparatus of claim 17, wherein the processor is further configured to execute programmed instructions stored in the memory further comprising:

compressing at least a portion of the at least one dictionary; and

sending to the client computing device source code configured to convert the modified source code into the original source code using the at least one dictionary.

21. The apparatus of claim 17, wherein the subset of the one or more fragments includes only one or more of the fragments occurring more than once in the plurality of URLs.

22. The apparatus of claim 17 wherein the unique indices comprise one or more digits in base 62 format.

23. The apparatus of claim 17, wherein the modifying further comprises:

removing any separator between words for any of the plurality of titles comprising two or more words;

inserting a space before or after any two or more sequential indices for any of the URLs; and

inserting a special character before or after any value of a query for any of the URLs.

24. The apparatus of claim 17, wherein the processor is further configured to execute programmed instructions stored in the memory further comprising:

determining whether the client computing device is a mobile computing device; and

only performing the generating and sending steps when it is determined that the client computing device is a mobile computing device.

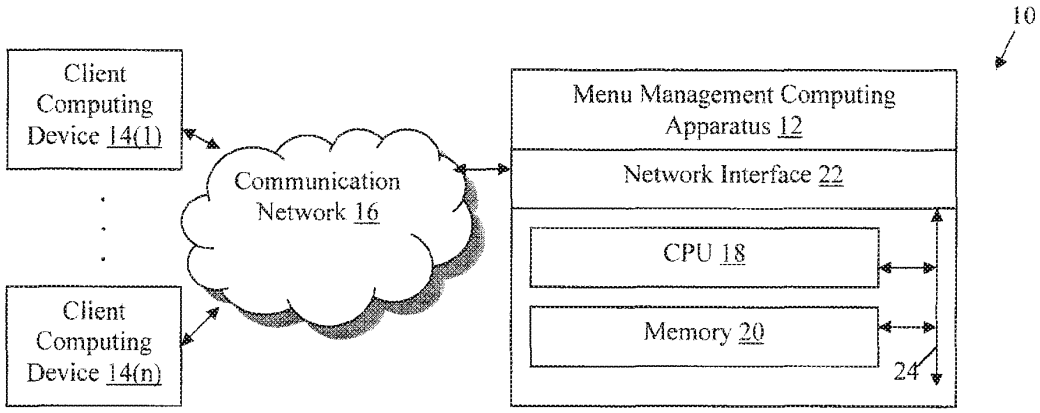


FIG. 1

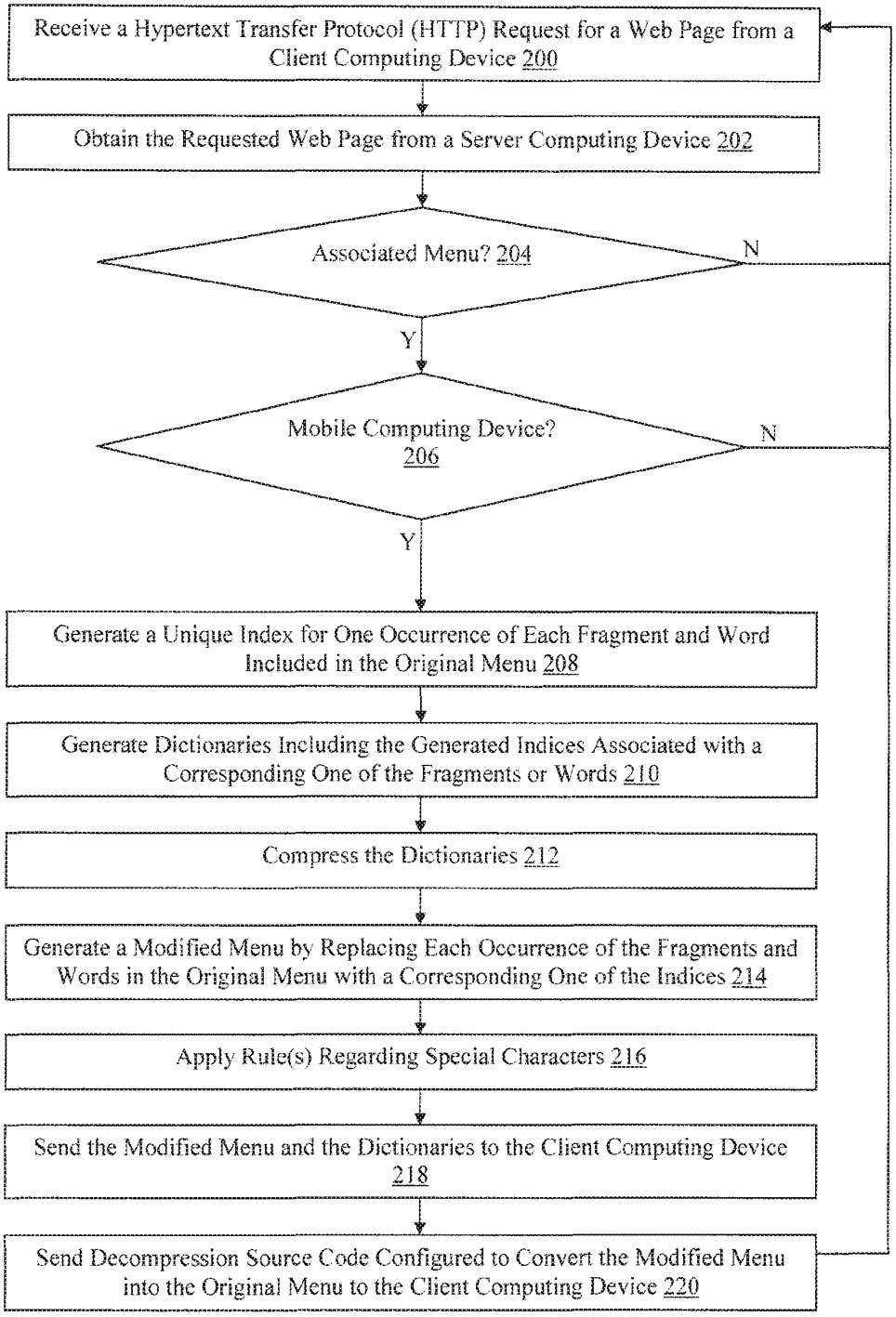


FIG. 2

2/4

```
[
    "Shoes",                                    300
    [                                           ↙
        "Men Shoes",
        "http://www.acme.com/catalog/Men_Shoes/products?plp_i=2354&sort=price",
        "Women Shoes",
        "http://www.acme.com/catalog/Women_Shoes/products?plp_i=2358&sort=price"
    ],
    "Accessories",
    "http://www.acme.com/catalog/Accessories/products?plp_i=1348&sort=price",
    "Dresses",
    [
        "Long Dresses",
        "http://www.acme.com/catalog/Accessories/products?plp_i=4242&sort=price",
        "Ceremonies",
        "http://www.acme.com/catalog/Ceremonies/products?plp_i=7718&sort=price"
    ]
]
```

FIG. 3

```
{
    "longWords":3,                                          400
    "longParams":1,
    "ch":"_",
    "baseUrl":"http://www.acme.com/catalog/",
    "params":["02& 06&","plp_i","sort"],
    "wl":2,
    "words":["products"," 02/ 00","Accessories","Shoes","Men",
             "Women","price","Dresses","Long","Ceremonies"],
    "tree":[
        "03",
        [
            "0403",
            " 0403/ 00?01&2354&00",
            "0503",
            " 0503/ 00?01&2358&00"
        ],
        "02",
        " 02/ 00?01&1348&00",
        "07",
        [
            "0807",
            " 02/ 00?01&4242&00",
            "09",
            " 09/ 00?01&7718&00"
        ]
    ]
}
```

FIG. 4

10
Menu Management Computing Apparatus 12
Network Interface 22
CPU 18
Memory 20
24
Communication Network 16
Client Computing Device 14(1)
Client Computing Device 14(n)

Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2848175 A1 2014/10/03

(21) 2 848 175

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(22) Date de dépôt/Filing Date: 2014/04/02

(41) Mise à la disp. pub./Open to Public Insp.: 2014/10/03

(30) Priorité/Priority: 2013/04/03 (US13/856,183)

(51) Cl.Int./Int.Cl. *H04L 29/06* (2006.01), *H04L 12/16* (2006.01)

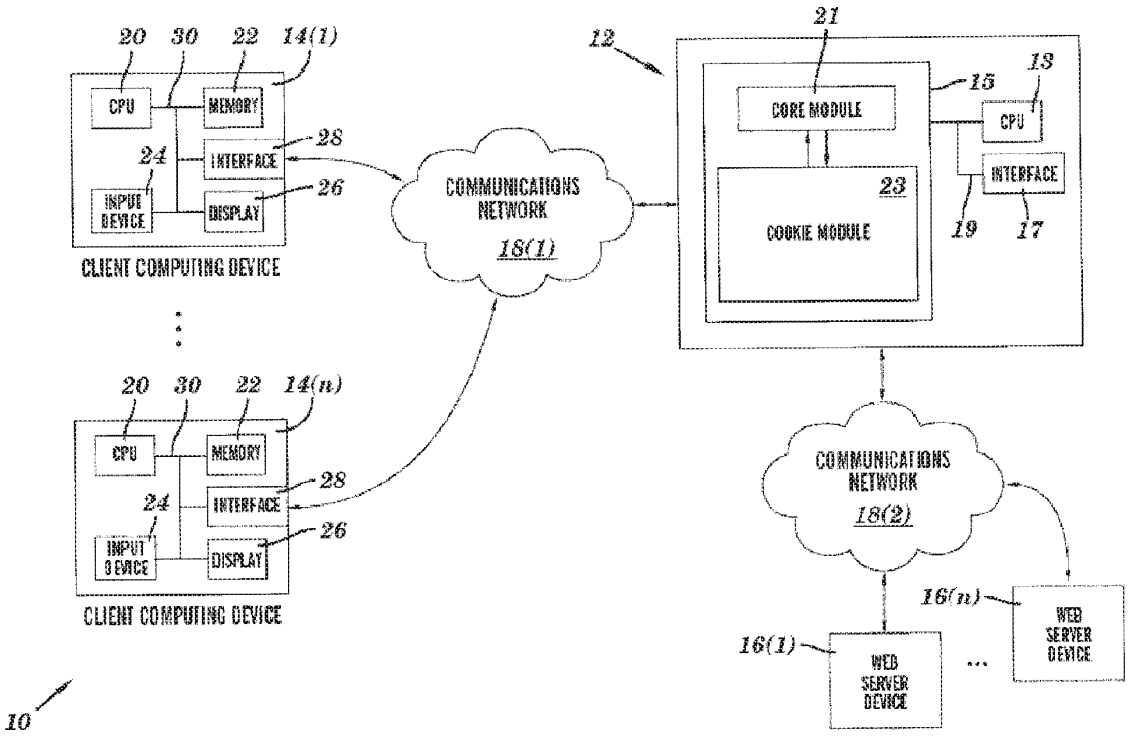
(71) Demandeur/Applicant:
USABLENET INC., US

(72) Inventeur/Inventor:
SCODA, ENRICO, IT

(74) Agent: PARLEE MCLAWS LLP

(54) Titre : PROCEDE POUR OPTIMISER UN SERVEUR MANDATAIRE DE CONTENU WEB ET DISPOSITIFS CORRESPONDANTS

(54) Title: METHODS FOR OPTIMIZING A WEB CONTENT PROXY SERVER AND DEVICES THEREOF



(57) Abrégé/Abstract:
A method, non-transitory computer readable medium, and apparatus that includes obtaining content with an original server cookie comprising a name and a value in response to a client request. Whether the value includes one or more of an established set of characters is determined. A new value is generated based on the value of the original server cookie and a URL encoding of the one or more of the established set of characters and any percent characters included in the value of the original server cookie prefixed by a first indicator character, when it is determined that the value includes one or more of the established set of characters. A web optimized client cookie comprising the new value and the name of the original server cookie concatenated with a domain attribute and path attribute associated with the content is generated. The web optimized client cookie is provided to the client.

ABSTRACT

A method, non-transitory computer readable medium, and apparatus that includes obtaining content with an original server cookie comprising a name and a value in response to a client request. Whether the value includes one or more of an established set of characters is determined. A new value is generated based on the value of the original server cookie and a URL encoding of the one or more of the established set of characters and any percent characters included in the value of the original server cookie prefixed by a first indicator character, when it is determined that the value includes one or more of the established set of characters. A web optimized client cookie comprising the new value and the name of the original server cookie concatenated with a domain attribute and path attribute associated with the content is generated. The web optimized client cookie is provided to the client.

METHODS FOR OPTIMIZING A WEB CONTENT PROXY SERVER AND DEVICES THEREOF

[0001] This application is a continuation-in-part of U.S. Patent Application Serial No. 13/685,346, filed on November 26, 2012, which is a continuation of U.S. Patent Application Serial No. 12/660,637, filed on March 2, 2010, now U.S. Patent No. 8,321,502, each of which is hereby incorporated by reference in its entirety.

FIELD

[0002] This invention generally relates to proxy servers and, more particularly, methods for optimizing web content proxy servers and apparatuses thereof.

BACKGROUND

[0003] A web content proxy server optimizes web pages obtained from remote web servers for client devices with special requirements, such as mobile phones, PDAs, and smartphones. Every time a client device requests a web page, the web content proxy server downloads the original page from a remote web server, applies some customized rules to extract relevant content, and adapts it to fit the needs of the requesting client device. By way of example, the web content proxy server may remove JavaScript, linearize content, and adapt the original page to a smaller screen layout for the requesting client device.

[0004] In computing, a cookie, such as a tracking cookie, browser cookie, and HTTP cookie, is a small piece of text stored by a web browser on the client device. A cookie includes one or more name-value pairs containing data, such as user preferences, shopping cart contents, the identifier for a server-based session, or other data used by websites.

[0005] Web content proxy servers need to save cookies to enable the client devices to interact with the original website at the remote web servers in the correct

way. Accordingly, web content proxy servers store these cookies in an internal memory and associate them with the corresponding session from each client device so that when the same client device sends a request for a new page, the web content proxy server will load the matching cookies and send them to the remote web server to get the page to process. Unfortunately, storing the cookies for these client devices causes problems with scalability, security, and privacy of the web content proxy servers.

SUMMARY

[0006] A method for optimizing a web content proxy server includes obtaining at a web content proxy server content with an original server cookie comprising at least a name and a value from a content server in response to a request from a client device for the content. A determination is made whether the value of the original server cookie includes one or more of an established set of characters at the web content proxy server. A new value is generated at the web content proxy server based on the value of the original server cookie and a uniform resource locator (URL) encoding of at least the one or more of the established set of characters and any percent characters included in the value of the original server cookie prefixed by a first indicator character, when the determination indicates the value of the original server cookie includes one or more of the established set of characters. A web optimized client cookie including the new value and a new name with at least the name of the original server cookie concatenated with at least a portion of a domain attribute and a path attribute associated with the obtained content is generated at the web content proxy server. At least the web optimized client cookie is provided by the web content proxy server to the requesting client device.

[0007] A non-transitory computer readable medium having stored thereon instructions for optimizing a web content proxy server comprising machine executable code which when executed by a processor, causes the processor to perform steps including obtaining content with an original server cookie comprising at least a name and a value from a content server in response to a request from a client

device for the content. A determination is made whether the value of the original server cookie includes one or more of an established set of characters. A new value is generated based on the value of the original server cookie and a uniform resource locator (URL) encoding of at least the one or more of the established set of characters and any percent characters included in the value of the original server cookie prefixed by a first indicator character, when the determination indicates the value of the original server cookie includes one or more of the established set of characters. A web optimized client cookie including the new value and a new name with at least the name of the original server cookie concatenated with at least a portion of a domain attribute and a path attribute associated with the obtained content is generated. At least the web optimized client cookie is provided to the requesting client device.

[0008] A web content proxy server apparatus, comprising a memory and a processor coupled to the memory and configured to execute programmed instructions stored in the memory including obtaining content with an original server cookie comprising at least a name and a value from a content server in response to a request from a client device for the content. A determination is made whether the value of the original server cookie includes one or more of an established set of characters. A new value is generated based on the value of the original server cookie and a uniform resource locator (URL) encoding of at least the one or more of the established set of characters and any percent characters included in the value of the original server cookie prefixed by a first indicator character, when the determination indicates the value of the original server cookie includes one or more of the established set of characters. A web optimized client cookie including the new value and a new name with at least the name of the original server cookie concatenated with at least a portion of a domain attribute and a path attribute associated with the obtained content is generated. At least the web optimized client cookie is provided to the requesting client device.

[0009] This technology provides a number of advantages including providing a method, non-transitory computer readable medium and an apparatus that optimizes

implementation of a web content proxy server for interactions involving cookies between client devices and remote web servers. With this technology, original server cookies are transformed by the web content proxy server to web optimized client cookies which are transmitted to the client devices requesting the web pages for storage and use with subsequent requests.

[0010] This technology provides greater scalability because the web optimized client cookies are stored in the web browser at the client device, not in memory at the web content proxy server. As a result, the web content proxy server does not face any issues with respect to memory storage capacity due to the number of sessions with cookies for client devices. The web content proxy server can use the same memory whether there are 100 or 1,000,000 or more client devices engaged in sessions with the remote web servers through the web content proxy server.

[0011] Additionally, this technology provides greater security and privacy because the web content proxy server does not contain a centralized database of original server cookies which contain session information from client devices browsing pages of web sites. Instead, these original server cookies are translated into web optimized client cookies which are then dispersed out among the client devices. As a result, the web content proxy server does not have any stored cookies from interactions between client devices and remote web servers that could be used to steal identity or other confidential information of these client devices.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] FIG. 1 is a block diagram of an exemplary environment with an optimized web content proxy server;

[0013] FIG. 2A is an example of a HTTP request for a web page from a remote web server;

[0014] FIG. 2B is an example of a HTTP response with an original server cookie from a remote web server to a HTTP request;

[0015] FIG. 2C is an example of a HTTP response containing the web optimized client cookie generated from the original server cookie received shown in FIG. 2B;

[0016] FIG. 2D is an example of another HTTP request with the web optimized client cookie shown in FIG. 2C for a web page from a remote web server.

[0017] FIG. 2E is an example of the another HTTP request with the web optimized client cookie shown in FIG. 2D translated into the original server cookie for transmission to the remote web server with the another get request;

[0018] FIG. 3 is a flow chart of an example of a method for generating a web optimized client cookie from an original server cookie to optimize implementation of a web content proxy server; and

[0019] FIG. 4 is a flow chart of an example of a method for transforming a web optimized client cookie back to an original server cookie to optimize implementation of a web content proxy server.

DETAILED DESCRIPTION

[0020] An exemplary environment 10 in which a web content proxy server 12 is optimized is illustrated in FIG. 1. The exemplary environment 10 includes a web content proxy server or apparatus 12, client devices 14(1)-14(n), web server devices 16(1)-16(n), and communication networks 18(1)-18(2), although other numbers and types of systems, devices, and/or elements in other configurations and environments with other communication network topologies can be used. This technology provides a number of advantages including providing a method, computer readable medium and an apparatus that optimizes implementation of a web content proxy server for interactions involving cookies between client devices and remote web servers.

[0021] Referring more specifically to FIG. 1, the web content proxy server 12 optimizes the handling of original server cookies from the web server devices 16(1)-

16(n) for requesting client devices 14(1)-14(n) and the handling of web optimized client cookies, although the web content proxy server 12 can provide other numbers and types of functions. Although one web content proxy server 12 is shown, other numbers and types of web content proxy devices and systems can be used.

[0022] The web content proxy server 12 includes a central processing unit (CPU) or processor 13, a memory 15, and an interface system 17 which are coupled together by a bus 19 or other link, although other numbers and types of components, parts, devices, systems, and elements in other configurations and locations can be used. The processor 13 in the web content proxy server 12 executes a program of stored instructions one or more aspects of the present invention as described and illustrated by way of the embodiments herein, although the processor could execute other numbers and types of programmed instructions.

[0023] The memory 15 in the web content proxy server 12 stores these programmed instructions for one or more aspects of the present invention as described and illustrated herein, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, DVD ROM, or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to the processor 13, can be used for the memory 15 in the web content proxy server 12. In these embodiments, the memory 15 includes a core module 21 and a cookie module 23 which store programmed instructions for one or more aspects of the present invention as described and illustrated herein, although the memory can comprise other types and numbers of systems, devices, and elements in other configurations which store other data. The cookie module 23 includes programmed instructions and/or logic configured to translate an original server cookie into a web optimized client cookie and to extract the original server cookie when a web optimized client cookie is

received, although the cookie module 23 can have other types and numbers of functions as described and illustrated herein.

[0024] The interface system 17 in the web content proxy server 12 is used to operatively couple and communicate between the web content proxy server 12 and the client devices 14(1)-14(n) and the web server devices 16(1)-16(n) via the communication networks 18(1) and 18(2), although other types and numbers of communication networks with other types and numbers of connections and configurations can be used. By way of example only, the communication networks 18(1) and 18(2) can use TCP/IP over Ethernet and industry-standard protocols, including HTTP, HTTPS, WAP, and SOAP, although other types and numbers of communication networks, such as a direct connection, a local area network, a wide area network, modems and phone lines, e-mail, and wireless and hardwire communication technology, each having their own communications protocols, can be used.

[0025] Each of the client devices 14(1)-14(n) enables a user to request, get and interact with web pages from one or more web sites hosted by the web server devices 16(1)-16(n) through the web content proxy server 12 via one or more communication networks, although one or more of the client devices 14(1)-14(n) could access content and utilize other types and numbers of applications from other sources and could provide a wide variety of other functions for the user. Although multiple client devices 14(1)-14(n) are shown, other numbers and types of user computing systems could be used. In this example, the client devices 14(1)-14(n) comprise mobile devices with Internet access that permit a website form page or other retrieved data to be displayed, although each of the client devices 14(1)-14(n). By way of example only, one or more of the client devices 14(1)-14(n) can comprise smart phones, personal digital assistants, or computers.

[0026] Each of client devices 14(1)-14(n) in this example is a computing device that includes a central processing unit (CPU) or processor 20, a memory 22, user input device 24, a display 26, and an interface system 28, and which are coupled

together by a bus 30 or other link, although one or more of client devices 14(1)-14(n) can include other numbers and types of components, parts, devices, systems, and elements in other configurations. The processor 20 in each of client devices 14(1)-14(n) executes a program of stored instructions for one or more aspects of the present invention as described and illustrated herein, although the processor could execute other numbers and types of programmed instructions.

[0027] The memory 22 in each of the client devices 14(1)-14(n) stores these programmed instructions for one or more aspects of the present invention as described and illustrated herein as well as the web optimized client cookies, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to processor 20 can be used for the memory 22 in each of the client devices 14(1)-14(n.

[0028] The user input device 24 in each of the client devices 14(1)-14(n) is used to input selections, such as requests for a particular website form page or to enter data in fields of a form page, although the user input device could be used to input other types of data and interact with other elements. The user input device can include keypads, touch screens, and/or vocal input processing systems although other types and numbers of user input devices can be used.

[0029] The display 26 in each of the client devices 14(1)-14(n) is used to show data and information to the user, such as website page by way of example only. The display in each of the client devices 14(1)-14(n) is a phone screen display, although other types and numbers of displays could be used depending on the particular type of client device.

[0030] The interface system 28 in each of the client devices 14(1)-14(n) is used to operatively couple and communicate between the client devices 14(1)-14(n) and the web content proxy server 12 and web server devices 16(1)-16(n) over the communication networks 18(1) and 18(2), although other types and numbers of communication networks with other types and numbers of connections and configurations can be used.

[0031] The web server devices 16(1)-16(n) provide one or more pages from one or more web sites for use by one or more of the client devices 14(1)-14(n) via the web content proxy server 12, although the web server devices 16(1)-16(n) can provide other numbers and types of applications and/or content and can have provide other numbers and types of functions. Although web server devices 16(1)-16(n) are shown for ease of illustration and discussion, other numbers and types of web server systems and devices can be used.

[0032] Each of the web server devices 16(1)-16(n) include a central processing unit (CPU) or processor, a memory, and an interface system which are coupled together by a bus or other link, although each of the web server devices 16(1)-16(n) could have other numbers and types of components, parts, devices, systems, and elements in other configurations and locations can be used. The processor in each of the web server devices 16(1)-16(n) executes a program of stored instructions one or more aspects of the present invention as described and illustrated by way of the embodiments herein, although the processor could execute other numbers and types of programmed instructions.

[0033] The memory in each of the web server devices 16(1)-16(n) stores these programmed instructions for one or more aspects of the present invention as described and illustrated by way of the embodiments, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, DVD ROM, or other computer readable medium which is read from and/or written to

by a magnetic, optical, or other reading and/or writing system that is coupled to the processor, can be used for the memory in each of the web server devices 16(1)-16(n).

[0034] The interface system in each of the web server devices 16(1)-16(n) is used to operatively couple and communicate between the web server devices 16(1)-16(n) and the web content proxy server 12 and the client devices 14(1)-14(n) via communication networks 18(1) and 18(2), although other types and numbers of communication networks with other types and numbers of connections and configurations can be used.

[0035] Although embodiments of the web content proxy server 12, the client devices 14(1)-14(n), and the web server devices 16(1)-16(n), are described and illustrated herein, each of the client devices 14(1)-14(n), the web content proxy server 12, and the web server devices 16(1)-16(n), can be implemented on any suitable computer system or computing device. It is to be understood that the devices and systems of the embodiments described herein are for exemplary purposes, as many variations of the specific hardware and software used to implement the embodiments are possible, as will be appreciated by those skilled in the relevant art(s).

[0036] Furthermore, each of the systems of the embodiments may be conveniently implemented using one or more general purpose computer systems, microprocessors, digital signal processors, and micro-controllers, programmed according to the teachings of the embodiments, as described and illustrated herein, and as will be appreciated by those ordinary skill in the art.

[0037] In addition, two or more computing systems or devices can be substituted for any one of the systems in any embodiment of the embodiments. Accordingly, principles and advantages of distributed processing, such as redundancy and replication also can be implemented, as desired, to increase the robustness and performance of the devices and systems of the embodiments. The embodiments may also be implemented on computer system or systems that extend across any suitable network using any suitable interface mechanisms and communications technologies,

including by way of example only telecommunications in any suitable form (e.g., voice and modem), wireless communications media, wireless communications networks, cellular communications networks, G3 communications networks, Public Switched Telephone Network (PSTNs), Packet Data Networks (PDNs), the Internet, intranets, and combinations thereof.

[0038] The embodiments may also be embodied as a computer readable medium having instructions stored thereon for one or more aspects of the present invention as described and illustrated by way of the embodiments herein, as described herein, which when executed by a processor, cause the processor to carry out the steps necessary to implement the methods of the embodiments, as described and illustrated herein.

[0039] An exemplary method for generating a web optimized client cookie from an original server cookie to optimize implementation of the web content proxy server 12 will now be described with reference to FIGS. 1-2C and 3. In step 50, in this example one of the client devices 14(1)-14(n) via a web browser requests a page A.html at the website, "www.example.com" as shown in one example in FIG. 2A. This request is transmitted to the web content proxy server 12 which processes and transmits the request to the one of the web servers 16(1)-16(n) hosting the website "www.example.com."

[0040] The hosting one of the web servers 16(1)-16(n) provides a response in this example for the requested page A.html which also contains an original server cookie "SESSION" to the web content proxy server 12 as shown in FIG. 2B. In this example, SESSION has a value equal to "1234", the domain attribute is equal to ".example.com" and the path attribute is equal to "/". This response uses the HTTP header Field "Set-Cookie". The cookie is a string formed by the pair "name=value", followed by optional attributes, like those in this example indicating the server domain attribute and path attribute accepting this cookie. Although one illustrative example is described herein, this technology can be used with specifications for all cookies.

[0041] Next, in step 52 the web content proxy server 12 determines whether the original server cookie includes the domain attribute for the requested web page provided by the hosting one of the web servers 16(1)-16(n). If in step 52 the web content proxy server 12 determines the original server cookie does not include the domain attribute, then the No branch is taken to step 54. In step 54, the web content proxy server 12 extracts the domain attribute from the requested web page provided by the hosting one of the web servers 16(1)-16(n). If in step 52 the web content proxy server 12 determines the original server cookie does include the domain attribute, then the Yes branch is taken to step 56.

[0042] In step 56, the web content proxy server 12 determines whether the original server cookie includes the path attribute for the requested web page provided by the hosting one of the web servers 16(1)-16(n). If in step 56 the web content proxy server 12 determines the original server cookie does not include the path attribute, then the No branch is taken to step 58. In step 58, the web content proxy server 12 extracts the path attribute from the requested web page provided by the hosting one of the web servers 16(1)-16(n). If in step 56 the web content proxy server 12 determines the original server cookie does include the path attribute, then the Yes branch is taken to step 60.

[0043] In step 60, the web content proxy server 12 determines whether the domain attribute or the path attribute contains any characters in an established set or a percent, a first, or a second character. In this example, the established set of characters includes a control character, a double quote character, a space character, a comma character, a semicolon character, a backslash character, or one or more reserved characters identified in Request for Comment No. 6265 ("RFC6265"). In this example, the first character is a plus sign character and the second character is an equal sign character, although other characters can also be used in the established set or as the first or second character.

[0044] If the web content proxy server 12 determines that the domain attribute or the path attribute associated with the content contains any characters in the

established set or a percent, a first, or a second character, then the Yes branch is taken to step 62. In step 62, the web content proxy server 12 generates a new domain attribute and/or a new path attribute by performing a uniform resource locator (URL) encoding of any characters in the established set and any percent, first, or second characters in the domain attribute and/or path attribute. Since the domain attribute and path attribute in this example are incorporated into a name of a web optimized client cookie, as described and illustrated in more detail below, the domain attribute and path attribute must be converted to conform to cookie name specifications set forth in RFC6265.

[0045] Additionally, the plus sign character is used to separate the name of the original server cookie from a domain attribute and a path attribute in a new name for the web optimized client cookie, as described and illustrated in detail below. The equal sign character is used to separate the new name from a new value in the web optimized client cookie, also as described and illustrated in detail below. Other characters can also be used as separators and corresponding first and second characters. Since, in this example, some character(s) of the domain attribute and/or the path attribute may be URL encoded, instances of the percent character in the domain attribute and path attribute must also be URL encoded to distinguish instances of the percent character from a percent character introduced by a URL encoding of another character in the domain attribute or path attribute.

[0046] Subsequent to the encoding in step 62 or, if the web content proxy server 12 determines, in step 60, that the domain attribute and path attributes associated with the content do not contain any characters in the established set or any percent, first, or second characters, and the No branch is taken, the web content proxy server 12 generates a name for a new web optimized client cookie in step 64. The web content proxy server 12 generates the name for the new web optimized client cookie by concatenating the original name and the original domain attribute and/or the path attribute, or the new domain attribute and/or the new path attribute generated

in step 62, each separated by the plus character, although other manners for generating the new name can be used.

[0047] The resulting new name is "universal resource locator encoded" to keep conformance to the cookie specification. Additionally, the resulting new name is unique even if different domain attributes contain cookies with the same name. This new name contains all the information necessary for the web content proxy server 12 to extract the original server cookie later, as described in greater detail by reference to FIG. 4.

[0048] In step 66, the web content proxy server 12 determines whether the value of the original server cookie includes one or more of an established set of characters. Optionally, the established set of characters is the same set of characters used in step 60, although a different set of characters can also be used. If the web content proxy server 12 determines that the value of the original server cookie includes one or more of the established set of characters, then the Yes branch is taken to step 68.

[0049] In step 68, the web content proxy server generates a new value based on the value of the original server cookie and a URL encoding of any characters included in the established set of characters, as well as any percent characters. Since the new value must comply with RFC 6265 but it is not guaranteed that original server cookie value is compliant, the original sever cookie value must be converted to comply with cookie-octet specifications

[0050] Additionally, instances of the percent character in the value must also be URL encoded to distinguish the instances with a percent character introduced by a URL encoding of another character in the value. In this example, the generated new value is further prefixed by a first indicator character such as "e" in this example, although other characters can be used for the first indicator character.

[0051] Referring back to step 66, if the web content proxy server 12 determines that the value of the original server cookie does not include any of the established set of characters, then the No branch is taken to step 70. In step 70, the web content proxy server 12 optionally generates a new value based on the value of the original server cookie prefixed by a second indicator character such as "n" in this example, although other characters can be used for the second indicator character. In other examples, the first indicator character is used to determine whether the value has been encoded, as described and illustrated in greater detail with reference to FIG. 4, and a second indicator character is not used.

[0052] In step 72, the web content proxy server 12 forms a new web optimized client cookie having the new name generated in step 64 and new value generated in step 68 or step 70. In this example, the domain attribute in the web optimized client cookie is not specified, and the path attribute is associated with a value "/". Other values can be used, such as one for the path attribute that corresponds to a prefix associated with this optimization method (by way of example only "/mt/").

[0053] By way of example only, when the web content proxy server 12 receives a response with the original server cookie as shown in FIG. 2B, the web content proxy server 12 generates a web optimized client cookie as shown in FIG. 2C. More specifically, the original server cookie: SESSION=1234; domain attribute=.example.com; and path attribute=/ is transformed by the web content proxy server 12 to a web optimized client cookie: SESSION+.example.com+/=n1234; path attribute=/mt/. Accordingly, in this illustrative example the new web optimized client cookie name represents the concatenation of the original server cookie name, original domain attribute, and original path attribute each separated by the plus sign character. Additionally, in this example, the new web optimized client cookie value represents the original server cookie value prefixed by an indicator character indicating whether the original server cookie value is encoded.

[0054] In this example, the original domain attribute and original path attribute are used because none of the characters of the original domain attribute and original path attribute were included in the established set of characters or matched the first or second characters (plus sign and equal sign, respectively, in this example). Other orders and manners for forming the name of the web optimized client cookie can also be used. In this example, the value of the web optimized client cookie is prefixed with an "n" character indicating that the value of the original server cookie did not include any characters in the established set of characters and, therefore, was not URL encoded. Additionally, in this example, the new path attribute corresponds to a prefix "/mt/" associated with this optimization method.

[0055] Next, in step 74 the web content proxy server 12 copies the remaining attributes in the original server cookie, such as an expiration date for the original server cookie by way of example, in the web optimized client cookie, although other amounts of the remaining attributes could be copied and other information also could be appended.

[0056] In step 76, the original server cookie which has been translated into the web optimized client cookie is now provided to the core module 21 in the web content proxy server 12. The core module 21 includes programmed instructions and/or logic to manage the transmission of the web optimized client cookie and the content from the web content proxy server 12 to the requesting one of the client devices 14(1)-14(n). The web browser at the requesting one of the client devices 14(1)-14(n) receives and saves the web optimized client cookie in the memory 22 at the requesting one of the client devices 14(1)-14(n).

[0057] Accordingly, in this illustrative example, the web optimized client cookie shown in FIG. 2C is stored in the memory 22 at the requesting one of the client devices 14(1)-14(n) and is not stored by the web content proxy server 12. While conformance with cookie name specifications could be maintained by URL encoding all characters, or all special characters, of the domain attribute, path attribute, and value of the original server cookie, in this example, only a limited

number of characters are URL encoded, thereby reducing the size of the web optimized client cookie. By reducing the size of the web optimized client cookie, less space in the memory 22 of the requesting one of the client devices 14(1)-14(n) is utilized and the web optimized client cookie can be sent to the requesting one of the client devices 14(1)-14(n) in less time.

[0058] Referring now to FIGS. 1, 2D-2E, and 4, an exemplary method for translating a web optimized client cookie back to an original server cookie to optimize the implementation of the web content proxy server 12 will now be described. In step 100, in this example one of the client devices 14(1)-14(n) via a web browser submits another request to the web content proxy server 12 for page B.html at the website, "www.example.com" as shown in one example in FIG. 2D. This request includes a web optimized client cookie, which in this example comprises a name/value pair: SESSION+.example.com+/=n1234.

[0059] In step 102, the web content proxy server 12 extracts the original server cookie name and the encoded domain attribute and path attributes from the name of the web optimized client cookie. In this illustrative example, the original server cookie name and the domain attribute and path attributes are extracted by the web content proxy server from the name: SESSION+.example.com+/. The domain attribute can be extracted based on the characters following the first plus sign character separating the domain attribute from the original server cookie name and the path attribute can be extracted based on the characters following the second plus sign character separating the domain attribute from the path attribute. Additionally, the value can be extracted based on the characters of the web optimized client cookie following the equal sign character . The web content proxy server 12 further performs a URL decoding of the domain attribute and path attributes. In this example, no characters of the domain attribute and path attribute were encoded in step 62 and the extracted domain attribute and path attribute represent the original domain attribute and path attributes as included in the original server cookie.

[0060] In step 104, the web content proxy server 12 determines whether the extracted domain attribute and path attribute identify a web optimized client cookie that is a match to a universal resource locator for the requested web page. If in step 104 the web content proxy server 12 determines the extracted domain attribute and path attributes identify a web optimized client cookie that is not a match, then the No branch is taken to step 106. In step 106, the web content proxy server 12 submits the request to the hosting one of the web servers 16(1)-16(n) hosting the request page without an original server cookie. In this illustrative example, the requested page is "B.html." If in step 104 the web content proxy server 12 determines the extracted domain attribute and path attributes identify a web optimized client cookie that is a match, then the Yes branch is taken to step 108.

[0061] In step 108, the web content proxy server 12 determines whether the web optimized client cookie included in the request received in step 100 includes a value, prefixed with the first indicator character or the second indicator character. If the web content proxy server 12 determines that the value of the web optimized client cookie is prefixed by the first indicator character, then the Yes branch is taken to step 110.

[0062] In step 110, the web content proxy server 12 generates a decoded value by performing a URL decoding of any characters following the first indicator character in the value of the web optimized client cookie extracted in step 102. Referring back to step 108, if the web content proxy server 12 determines that the value of the web optimized client cookie is prefixed by the second character, then the No branch is taken to step 112. In examples in which the second indicator is not used, and optional step 70 is not performed, the No branch is taken by web content proxy server 12 when the web content proxy sever 12 determines the extracted value is not prefixed by the first indicator character.

[0063] In step 112, the web content proxy server 12 generates a decoded value including the characters following the second indicator character in the value of the web optimized client cookie extracted in step 102. In examples in which the

second indicator character is not used, the value of the web optimized client cookie extracted in step 102 is used as the decoded value. Since, in the example illustrated in FIG. 2D, the value n1234 is prefixed with an "n", the No branch is taken from step 108 and the decoded value generated in step 112 is 1234. Accordingly, one of the indicator characters is inserted by the web content proxy server 12 to indicate to the web content proxy server 12 whether the value included in a subsequent request including a web optimized client has been URL encoded, as described and illustrated earlier with reference to step 68.

[0064] In step 114, the web content proxy server 12 creates a new cookie by associating the name extracted from the web optimized client cookie in step 102 with the value for the original server cookie decoded in step 110 or 112. The extracted name and decoded value comprise the original server cookie which is appended to the HTTP cookie header fields of the request to be sent to the one of the web servers 16(1)-16(n) hosting the requested web page. In this illustrative example, the extracted name SESSION is associated with the value 1234.

[0065] In step 116, the web content proxy server 12 submits the request with the reconstituted original server cookie to the one of the web servers 16(1)-16(n) hosting the requested page. In this illustrative example, the request with the reconstituted original server cookie as shown in FIG. 2E is transmitted to the one of the web servers 16(1)-16(n) hosting the requested web page B.html.

[0066] Accordingly, as illustrated and described herein this technology provides a number of advantages including providing a method, computer readable medium and an apparatus that optimizes implementation of a web content proxy server for interactions involving cookies between client devices and remote web servers. With this technology, the web content proxy server is much more scalable because of the reduced memory storage demands and the web content proxy server poses a much lower security and privacy risk to information provided by the client devices 14(1)-14(n). Additionally, the size of the cookies, and associated time required to send cookies to client devices and storage requirements on the client

devices, is advantageously reduced while conformance with cookie name specifications is maintained.

[0067] Having thus described the basic concept of the invention, it will be rather apparent to those skilled in the art that the foregoing detailed disclosure is intended to be presented by way of example only, and is not limiting. Various alterations, improvements, and modifications will occur and are intended to those skilled in the art, though not expressly stated herein. These alterations, improvements, and modifications are intended to be suggested hereby, and are within the spirit and scope of the invention. Additionally, the recited order of processing elements or sequences, or the use of numbers, letters, or other designations therefore, is not intended to limit the claimed processes to any order except as may be specified in the claims. Accordingly, the invention is limited only by the following claims and equivalents thereto.

CLAIMS

What is claimed is:

1. A method for optimizing a web content proxy server, the method comprising:

obtaining at a web content proxy server content with an original server cookie comprising at least a name and a value from a content server in response to a request from a client device for the content;

determining at the web content proxy server whether the value of the original server cookie includes one or more of an established set of characters;

generating with the web content proxy server a new value based on the value of the original server cookie and a uniform resource locator (URL) encoding of at least the one or more of the established set of characters and any percent characters included in the value of the original server cookie prefixed by a first indicator character, when the determining indicates the value of the original server cookie includes one or more of the established set of characters;

generating at the web content proxy server a web optimized client cookie comprising the new value and a new name comprising at least the name of the original server cookie concatenated with at least a portion of a domain attribute and a path attribute associated with the obtained content; and

providing with the web content proxy server at least the web optimized client cookie to the requesting client device.

2. The method as set forth in claim 1, further comprising generating with the web content proxy server a new value comprising the value of the original server cookie prefixed by a second indicator character, when it is determined that the value of the original server cookie does not include one or more of the established set of characters.

3. The method as set forth in claim 1, further comprising:

generating with the web content proxy server a new domain attribute or a new path attribute based on the domain attribute or the path attribute associated with the content and a URL encoding of at least any percent character, first character, or second character of the domain attribute or the path attribute associated with the content; and

wherein the domain attribute or the path attribute of the web optimized client cookie comprises the new domain attribute or the new path attribute and the new path attribute and the new value are separated by the second character.

4. The method as set forth in claim 3, further comprising:

determining at the web content proxy server whether the original server cookie includes a domain attribute or a path attribute;

generating with the web content proxy server the new domain attribute or the new path attribute based on the domain attribute or the path attribute included in the original server cookie, when it is determined that the original server cookie includes the domain attribute or the path attribute; and

generating with the web content proxy server the new domain attribute or the new path attribute based on a domain attribute or a path attribute of a network address of the obtained content, when it is determined that the original server cookie does not include the domain attribute or the path attribute.

5. The method as set forth in claim 3, wherein the first character is a plus character and the second character is an equal character and the established set of characters further comprises one or more of a control character, a double quote character, a space character, a comma character, a semicolon character, a backslash character, the plus character, the equal character, or one or more reserved characters identified in Request for Comment No. 6265.

6. The method of claim 1, further comprising:

processing with the web content proxy server a subsequent request from the client device by reconstituting the original server cookie using a web optimized client cookie included in the subsequent request, the reconstituting comprising:

determining whether the web optimized client cookie included in the subsequent request includes a value prefixed with the first indicator character; and

generating a URL decoding of a plurality of characters following the first indicator character included in the web optimized client cookie included in the subsequent request, when it is determined that the web optimized client cookie included in the subsequent request includes a value prefixed with the first indicator character.

7. The method as set forth in claim 6, wherein the processing further comprises:

determining whether a domain attribute and a path attribute of the reconstituted original server cookie correspond with a network address of the subsequent request; and

providing the subsequent request with the reconstituted original server cookie to the content server, when it is determined that the domain attribute and the path attribute of the reconstituted original server cookie correspond with the network address of the subsequent request.

8. The method as set forth in claim 1, wherein the providing further comprises providing the obtained content and the web optimized client cookie to the client device without storing the original server cookie or the web optimized client cookie.

9. The method as set forth in claim 1, wherein the generating the web optimized client cookie further comprises appending one or more original attributes of the original server cookie to the web optimized client cookie.

10. A non-transitory computer readable medium having stored thereon instructions for optimizing a web content proxy server comprising machine executable code which when executed by a processor, causes the processor to perform steps comprising:

obtaining content with an original server cookie comprising at least a name and a value from a content server in response to a request from a client device for the content;

determining whether the value of the original server cookie includes one or more of an established set of characters;

generating a new value based on the value of the original server cookie and a uniform resource locator (URL) encoding of at least the one or more of the established set of characters and any percent characters included in the value of the original server cookie prefixed by a first indicator character, when the determining indicates the value of the original server cookie includes one or more of the established set of characters;

generating a web optimized client cookie comprising the new value and a new name comprising at least the name of the original server cookie concatenated with at least a portion of a domain attribute and a path attribute associated with the obtained content; and

providing at least the web optimized client cookie to the requesting client device.

11. The medium as set forth in claim 10, wherein the medium further has stored thereon instructions comprising machine executable code which when executed by at least one processor, causes the processor to perform steps further comprising generating a new value comprising the value of the original server cookie

prefixed by a second indicator character, when it is determined that the value of the original server cookie does not include one or more of the established set of characters.

12. The medium as set forth in claim 10, wherein the medium further has stored thereon instructions comprising machine executable code which when executed by at least one processor, causes the processor to perform steps further comprising:

generating with the proxy server a new domain attribute or a new path attribute based on the domain attribute or the path attribute associated with the content and a URL encoding of at least any percent character, first character, or second character of the domain attribute or the path attribute associated with the content; and

wherein the domain attribute or the path attribute of the web optimized client cookie comprises the new domain attribute or the new path attribute and the new path attribute and the new value are separated by the second character.

13. The medium as set forth in claim 12, wherein the medium further has stored thereon instructions comprising machine executable code which when executed by at least one processor, causes the processor to perform steps further comprising:

determining whether the original server cookie includes a domain attribute or a path attribute;

generating the new domain attribute or the new path attribute based on the domain attribute or the path attribute included in the original server cookie, when it is determined that the original server cookie includes the domain attribute or the path attribute; and

generating the new domain attribute or the new path attribute based on a domain attribute or a path attribute of a network address of the obtained

content, when it is determined that the original server cookie does not include the domain attribute or the path attribute.

14. The medium as set forth in claim 12, wherein the first character is a plus character and the second character is an equal character and the established set of characters further comprises one or more of a control character, a double quote character, a space character, a comma character, a semicolon character, a backslash character, the plus character, the equal character, or one or more reserved characters identified in Request for Comment No. 6265.

15. The medium as set forth in claim 10, wherein the medium further has stored thereon instructions comprising machine executable code which when executed by at least one processor, causes the processor to perform steps further comprising:

processing a subsequent request from the client device by reconstituting the original server cookie using a web optimized client cookie included in the subsequent request, the reconstituting comprising:

determining whether the web optimized client cookie included in the subsequent request includes a value prefixed with the first indicator character; and

generating a URL decoding of a plurality of characters following the first indicator character included in the web optimized client cookie included in the subsequent request, when it is determined that the web optimized client cookie included in the subsequent request includes a value prefixed with the first indicator character.

16. The medium as set forth in claim 15, wherein the processing further comprises:

determining whether a domain attribute and a path attribute of the reconstituted original server cookie correspond with a network address of the subsequent request; and

providing the subsequent request with the reconstituted original server cookie to the content server, when it is determined that the domain attribute and the path attribute of the reconstituted original server cookie correspond with the network address of the subsequent request.

17. The medium as set forth in claim 10, wherein the providing further comprises providing the obtained content and the web optimized client cookie to the client device without storing the original server cookie or the web optimized client cookie.

18. The medium as set forth in claim 10, wherein the generating the web optimized client cookie further comprises appending one or more original attributes of the original server cookie to the web optimized client cookie.

19. A web content proxy server apparatus, comprising:

a memory; and

a processor coupled to the memory and configured to execute programmed instructions stored in the memory, comprising:

obtaining content with an original server cookie comprising at least a name and a value from a content server in response to a request from a client device for the content;

determining whether the value of the original server cookie includes one or more of an established set of characters;

generating a new value based on the value of the original server cookie and a uniform resource locator (URL) encoding of at least the one or more of the established set of characters and any percent characters included in the value of the original server cookie prefixed by a first indicator character, when the

determining indicates the value of the original server cookie includes one or more of the established set of characters;

generating a web optimized client cookie comprising the new value and a new name comprising at least the name of the original server cookie concatenated with at least a portion of a domain attribute and a path attribute associated with the obtained content; and

providing at least the web optimized client cookie to the requesting client device.

20. The apparatus as set forth in claim 19, wherein the processor is further configured to execute programmed instructions stored in the memory further comprising generating a new value comprising the value of the original server cookie prefixed by a second indicator character, when it is determined that the value of the original server cookie does not include one or more of the established set of characters;

21. The apparatus as set forth in claim 19, wherein the processor is further configured to execute programmed instructions stored in the memory further comprising:

generating with the proxy server a new domain attribute or a new path attribute based on the domain attribute or the path attribute associated with the content and a URL encoding of at least any percent character, first character, or second character of the domain attribute or the path attribute associated with the content; and

wherein the domain attribute or the path attribute of the web optimized client cookie comprises the new domain attribute or the new path attribute and the new path attribute and the new value are separated by the second character.

22. The apparatus as set forth in claim 21, wherein the processor is further configured to execute programmed instructions stored in the memory further comprising:

determining whether the original server cookie includes a domain attribute or a path attribute;

generating the new domain attribute or the new path attribute based on the domain attribute or the path attribute included in the original server cookie, when it is determined that the original server cookie includes the domain attribute or the path attribute; and

generating the new domain attribute or the new path attribute based on a domain attribute or a path attribute of a network address of the obtained content, when it is determined that the original server cookie does not include the domain attribute or the path attribute.

23. The apparatus as set forth in claim 21, wherein the first character is a plus character and the second character is an equal character and the established set of characters further comprises one or more of a control character, a double quote character, a space character, a comma character, a semicolon character, a backslash character, the plus character, the equal character, or one or more reserved characters identified in Request for Comment No. 6265.

24. The apparatus as set forth in claim 19, wherein the processor is further configured to execute programmed instructions stored in the memory further comprising:

processing a subsequent request from the client device by reconstituting the original server cookie using a web optimized client cookie included in the subsequent request, the reconstituting comprising:

determining whether the web optimized client cookie included in the subsequent request includes a value prefixed with the first indicator character; and

generating a URL decoding of a plurality of characters following the first indicator character included in the web optimized client cookie included in the subsequent request, when it is determined that the web optimized client cookie included in the subsequent request includes a value prefixed with the first indicator character.

25. The apparatus as set forth in claim 24, wherein the processing further comprises:

determining whether a domain attribute and a path attribute of the reconstituted original server cookie correspond with a network address of the subsequent request; and

providing the subsequent request with the reconstituted original server cookie to the content server, when it is determined that the domain attribute and the path attribute of the reconstituted original server cookie correspond with the network address of the subsequent request.

26. The apparatus as set forth in claim 19, wherein the providing further comprises providing the obtained content and the web optimized client cookie to the client device without storing the original server cookie or the web optimized client cookie.

27. The apparatus as set forth in claim 19, wherein the generating the web optimized client cookie further comprises appending one or more original attributes of the original server cookie to the web optimized client cookie.

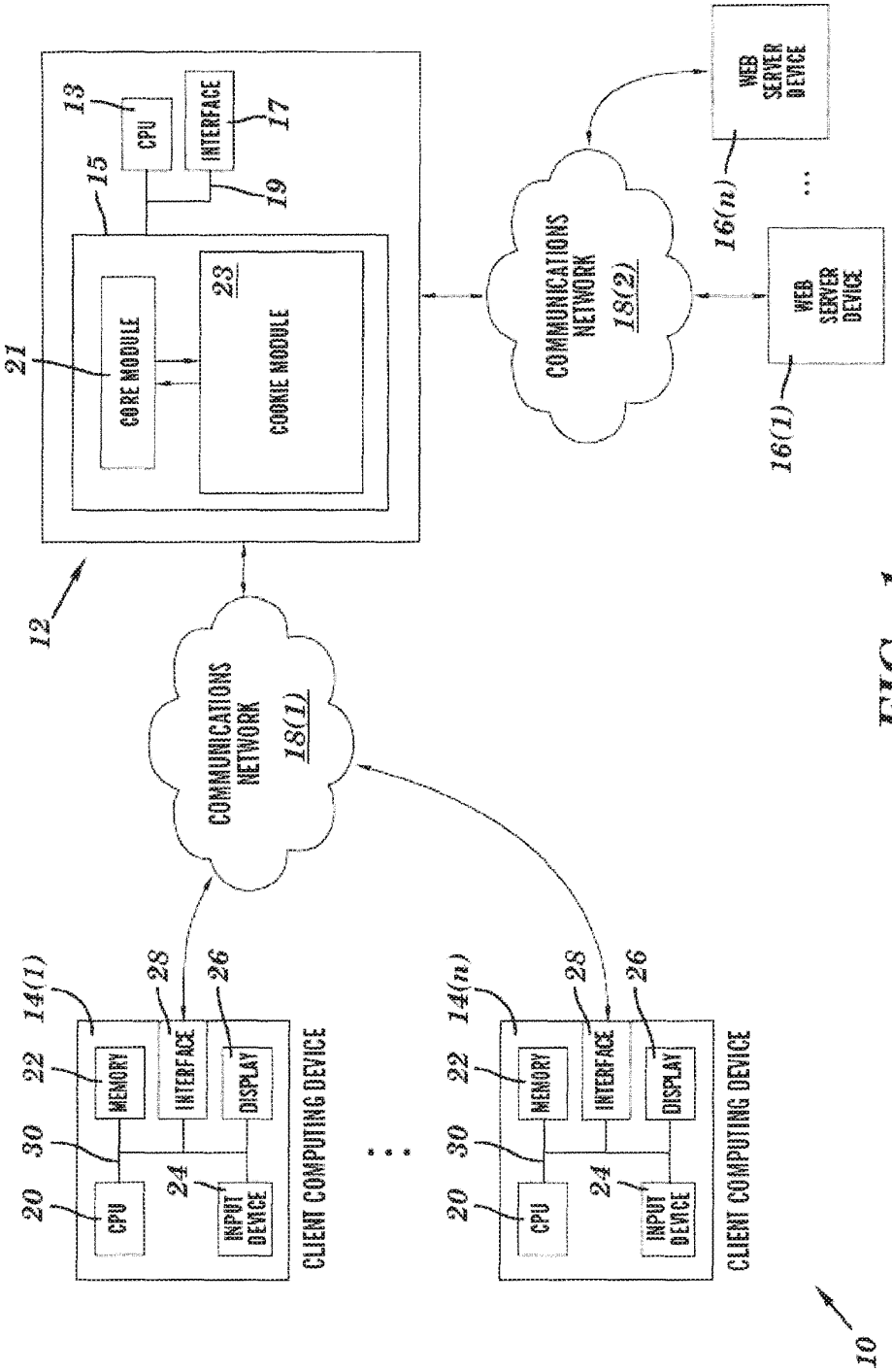

10
12
13
CPU
INTERFACE
17
19
15
21
CORE MODULE
23
COOKIE MODULE
COMMUNICATIONS NETWORK 18(1)
COMMUNICATIONS NETWORK 18(2)
WEB SERVER DEVICE
16(n)
16(1)
14(1)
14(n)
CLIENT COMPUTING DEVICE
MEMORY
INTERFACE
DISPLAY
CPU
INPUT DEVICE
20
22
24
26
28
30

FIG. 1

```
GET /A.html HTTP/1.1
Host: www.example.com
Accept: */*
User-Agent: my-mobile-browser 1.0
```

FIG. 2A

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Content-Length: 5300
Set-Cookie: SESSION=1234; domain=.example.com; path=/
```

FIG. 2B

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Content-Length: 5300
Set-Cookie: SESSION+.example.com+/=n1234; path=mt/
```

FIG. 2C

```
GET /mt/www.example.com/B.html HTTP/1.1
User-Agent: my-mobile-browser 1.0
Host: m.proxy.com
Accept: */*
Cookie: SESSION+.example.com+/=n1234
```

FIG. 2D

```
GET /B.html HTTP/1.1
User-Agent: my-mobile-browser 1.0
Host: www.example.com
Accept: */*
Cookie: SESSION=1234
```

FIG. 2E

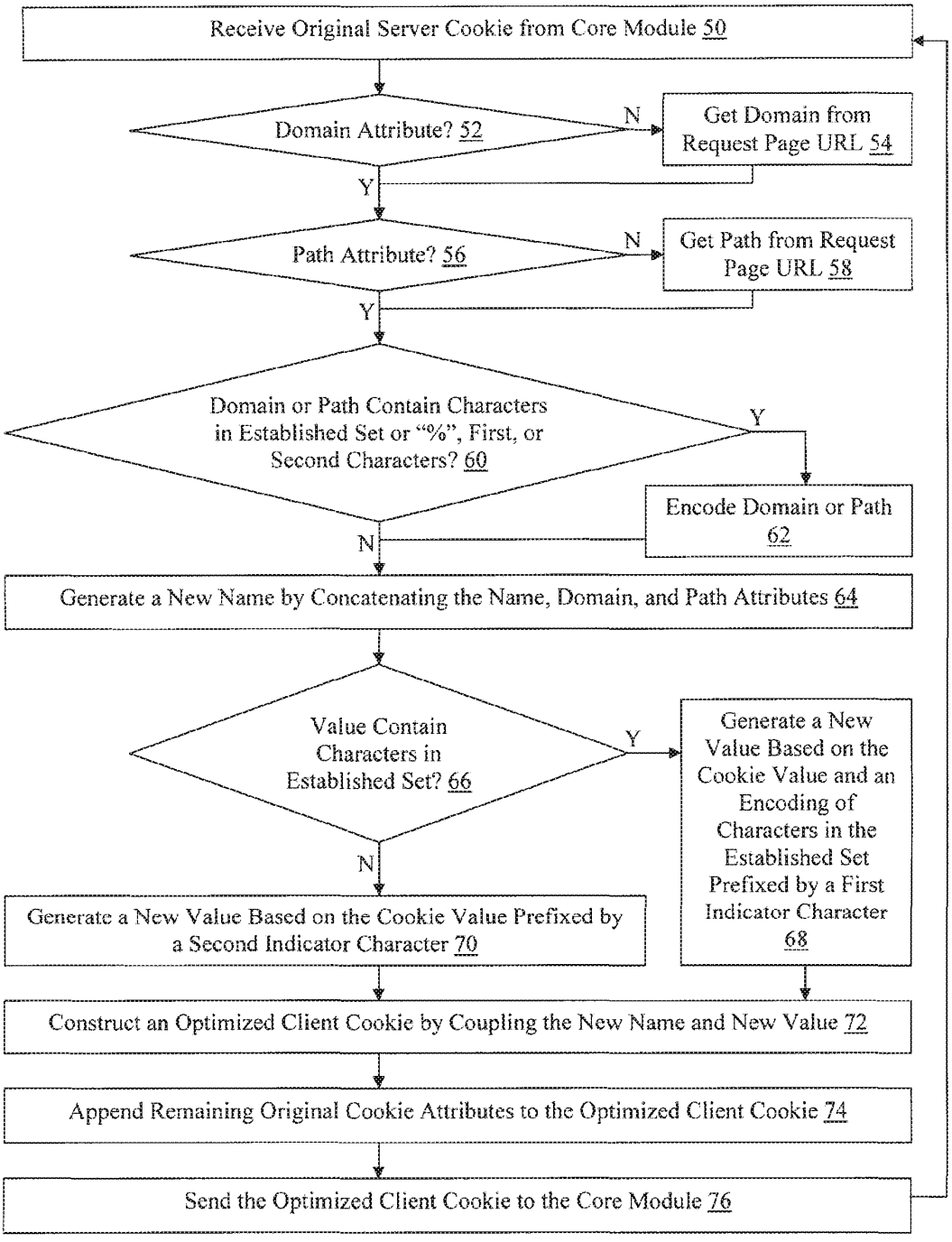


FIG. 3

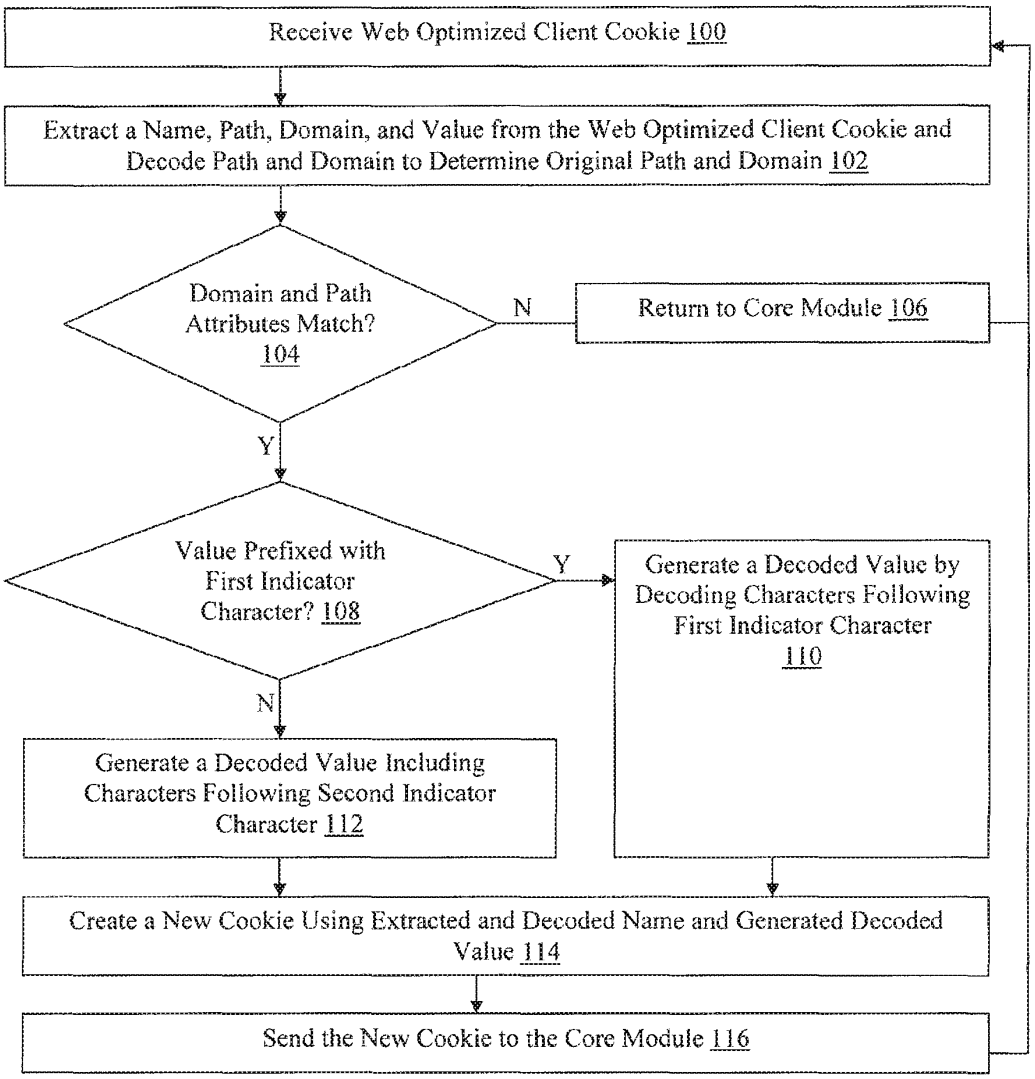


FIG. 4

10
CLIENT COMPUTING DEVICE
14(1)
20
CPU
30
22
MEMORY
24
INPUT DEVICE
28
INTERFACE
26
DISPLAY
CLIENT COMPUTING DEVICE
14(n)
20
CPU
30
22
MEMORY
24
INPUT DEVICE
28
INTERFACE
26
DISPLAY
COMMUNICATIONS NETWORK
18(1)
12
21
CORE MODULE
23
COOKIE MODULE
15
CPU
13
INTERFACE
17
19
COMMUNICATIONS NETWORK
18(2)
WEB SERVER DEVICE
16(1)
WEB SERVER DEVICE
16(n)

Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2855420 A1 2015/01/09

(21) 2 855 420

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(22) Date de dépôt/Filing Date: 2014/07/03

(41) Mise à la disp. pub./Open to Public Insp.: 2015/01/09

(30) Priorité/Priority: 2013/07/09 (US13/937,596)

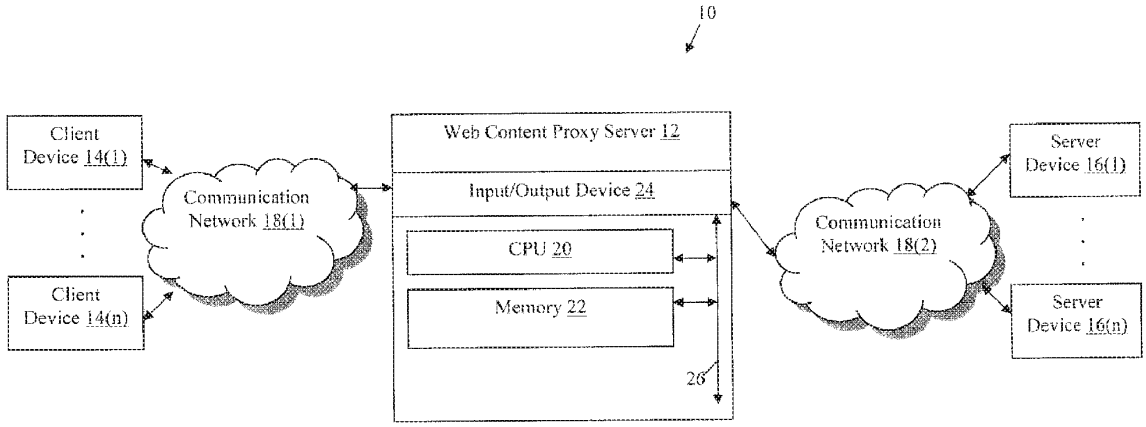
(51) Cl.Int./Int.Cl. *H04L 12/16* (2006.01), *G06F 17/00* (2006.01)

(71) Demandeur/Applicant:
USABLENET INC., US

(72) Inventeur/Inventor:
SCODA, ENRICO, IT

(74) Agent: PARLEE MCLAWS LLP

(54) Titre : PROCEDES POUR GROUPER DES IMAGES ET DISPOSITIFS DE CEUX-CI
(54) Title: METHODS FOR BUNDLING IMAGES AND DEVICES THEREOF



(57) Abrégé/Abstract:
A method, non-transitory computer readable medium, and web server device that obtains a web page comprising a plurality of image elements each including a source attribute having a value identifying an image. Each of the image elements is modified to insert a data attribute having a value of the respective source attribute value and to replace the source attribute value with a data URI. A reference to a executable file is inserted into the web page. The web page is sent to the client device and a request from the client device for the executable file is received. The executable file is sent to the client device and is configured when executed to replace the source attribute value of each of the image elements with a data URI of an image identified by the respective data attribute value.

ABSTRACT

A method, non-transitory computer readable medium, and web server device that obtains a web page comprising a plurality of image elements each including a source attribute having a value identifying an image. Each of the image elements is modified to insert a data attribute having a value of the respective source attribute value and to replace the source attribute value with a data URI. A reference to a executable file is inserted into the web page. The web page is sent to the client device and a request from the client device for the executable file is received. The executable file is sent to the client device and is configured when executed to replace the source attribute value of each of the image elements with a data URI of an image identified by the respective data attribute value.

METHODS FOR BUNDLING IMAGES AND DEVICES THEREOF

FIELD

[0001] This technology generally relates to methods and devices for optimizing transmission of web page images and, more particularly, methods for bundling images and devices thereof.

BACKGROUND

[0002] Many web sites are increasingly sophisticated and provide rich multimedia experiences for users. Often, the multimedia content of a web page includes a significant number of images which are sent across a communication network. For example, product catalog and social network web pages are generally image-intensive, although many other types of web pages also have significant graphical content. The images of a web page are generally each retrieved through a hypertext transfer protocol (HTTP) request sent from a client device while rendering the web page. However, many networks have high latency and require a significant amount of time to transmit each request and server response including one of the requested images. Latency is a particularly significant issue with respect to networks generally utilized by mobile devices.

[0003] In order to reduce the number of HTTP requests sent by a client device for the images of a web page, web servers can parse web pages requested by client devices to identify referenced images, retrieve the images, generate an encoding of each of the images, and modify the web pages to include the encoding of the images in-line prior to sending the web pages to the requesting client devices. The embedded encoding can be a base 64 representation of each image and can be included in the web page according to a data uniform resource identifier (URI) scheme, for example. By including the referenced images in-line, web pages can be rendered by web browsers of client devices without requiring an HTTP request and response for each of the images.

[0004] While including images in-line can reduce the time required to render a web page, particularly in high latency communication networks, there are several drawbacks to this approach. For example, the images are not cached separately from the web page that includes the corresponding in-line encoding. Accordingly, every time a change is made to the web page, and a cached version of the web page becomes invalid, the images must again be encoded and embedded in-line. Another drawback is that multiple copies of the encoded version of images that are referenced more than once in a web page are generated, while externally-referenced images are downloaded only once irrespective of the number of references to the images in the web page.

SUMMARY

[0005] A method for bundling images includes obtaining, with a web server, a web page requested by a client device, the web page comprising a plurality of image elements each including a source attribute having a value identifying an image. Each of the plurality of image elements is modified, with the web server, to insert a data attribute having a value of the respective source attribute value and to replace the source attribute value with a data uniform resource indicator (URI). With the web server, a reference to an executable file is inserted into the requested web page, the requested web page is sent to the client device, and then a request from the client device for the executable file is received. The executable file is sent with the web server to the client device in response to the request and is configured when executed to replace the source attribute value of each of the plurality of image elements with a data URI of an image identified by the respective data attribute value.

[0006] A non-transitory computer readable medium having stored thereon instructions for bundling images comprising machine executable code which when executed by a processor, causes the processor to perform steps including obtaining a web page requested by a client device, the web page comprising a plurality of image elements each including a source attribute having a value identifying an image. Each of the plurality of image elements is modified to insert a data attribute having a value of the

respective source attribute value and to replace the source attribute value with a data URI. A reference to a executable file is inserted into the web page, the web page is sent to the client device, and a request from the client device for the executable file is received. The executable file is sent to the client device in response to the request and is configured when executed to replace the source attribute value of each of the plurality of image elements with a data URI of an image identified by the respective data attribute value.

[0007] A web server device includes a processor coupled to a memory and configured to execute programmed instructions stored in the memory including obtaining a web page requested by a client device, the web page comprising a plurality of image elements each including a source attribute having a value identifying an image. Each of the plurality of image elements is modified to insert a data attribute having a value of the respective source attribute value and to replace the source attribute value with a data URI. A reference to a executable file is inserted into the web page, the web page is sent to the client device, and a request from the client device for the executable file is received. The executable file is sent to the client device in response to the request and is configured when executed to replace the source attribute value of each of the plurality of image elements with a data URI of an image identified by the respective data attribute value.

[0008] This technology provides a number of advantages including methods, non-transitory computer readable medium, and a web server device that mitigates the negative affects of high network latency with respect to retrieval of web page images, while also reducing storage overhead. With this technology, web page images are encoded, bundled, and included in-line in web pages by a retrieved executable file. Advantageously, only one copy of each of the image encodings is stored irrespective of the number of references to the images in the web page. Additionally, the image encodings are stored separately from the associated web pages. Therefore, changes made to a web page that render a cached version invalid do not necessitate a subsequent downloading and encoding of the images of the web page.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] FIG. 1 is a block diagram of an environment with an exemplary web content proxy server;

[0010] FIG. 2 is a flow chart of an exemplary method for bundling web page images;

[0011] FIG. 3 is exemplary hypertext markup language (HTML) code fragment defining a portion of a web page that references multiple images;

[0012] FIG. 4 is an exemplary version of the HTML code fragment of FIG. 3 with image elements modified to include a data uniform resource identifier (URI) of a default image and a respective data attribute; and

[0013] FIG. 5 is exemplary JavaScript code fragment configured to insert, when executed by a client device, a data URI of the images referenced by the HTML code fragment of FIG. 3.

DETAILED DESCRIPTION

[0014] An exemplary environment 10 with a web content proxy server 12 coupled to client devices 14(1)-14(n) and server devices 16(1)-16(n) by communication networks 18(1)-18(2) is illustrated in FIG. 1. Other numbers and types of systems, devices, and/or elements in other configurations and environments with other communication network topologies can also be used. This technology provides a number of advantages including providing methods, non-transitory computer readable medium, and devices for bundling and in-lining web page images to reduce the number of communications required to render the images on a client device while also reducing storage overhead.

[0015] Referring more specifically to FIG. 1, the web content proxy server 12 includes a central processing unit (CPU) 20 or processor, a memory 22, and an

input/output device 24, which are coupled together by a bus 26 or other link, although other numbers and types of components, parts, devices, systems, and elements in other configurations and locations can be used. The CPU 20 in the web content proxy server 12 executes a program of stored instructions one or more aspects of the present invention as described and illustrated by way of the embodiments herein, although the CPU 20 could execute other numbers and types of programmed instructions.

[0016] The memory 22 in the web content proxy server 16 stores these programmed instructions for one or more aspects of the present invention as described and illustrated herein, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, DVD ROM, or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to the CPU 20, can be used for the memory 22 in the web content proxy server 12.

[0017] The input/output device 24 in the web content proxy server 12 is used to operatively couple and communicate between the web content proxy server 12, client devices 14(1)-14(n), and server devices 16(1)-16(n) via the communication networks 18(1)-18(2). One or more of the communication networks 18(1)-18(2) can include one or more networks, such as one or more local area networks (LANs) and/or wide area networks (WANs). By way of example only, the communication networks can use TCP/IP over Ethernet and industry-standard protocols, including hypertext transfer protocol (HTTP), secure HTTP (HTTPS), wireless application protocol (WAP), and/or SOAP, although other types and numbers of communication networks, such as a direct connection, modems and phone lines, e-mail, and wireless and hardwire communication technology, each having their own communications protocols, can be used.

[0018] The client devices 14(1)-12(n) enable a user to request, receive, and interact with applications, web services, and content hosted by the server devices 16(1)-

16(n) through the web content proxy server 12 via the communication network 18(1), although one or more of the client devices 14(1)-14(n) could access content and utilize other types and numbers of applications from other sources and could provide a wide variety of other functions for the user. In some examples, the client devices 14(1)-14(n) comprise mobile computing devices with Internet access that enable web pages and other content stored by the server devices 16(1)-16(n) to be retrieved and rendered. By way of example only, the client devices 14(1)-14(n) can be smart phones, personal digital assistants, or computers.

[0019] Each of the client devices 14(1)-14(n) includes a CPU, a memory, an input device, a display device, and an input/output device, which are coupled together by a bus or other link, although one or more of client devices 14(1)-14(n) can include other numbers and types of components, parts, devices, systems, and elements in other configurations. The CPU in the client devices 14(1)-14(n) can execute a program of instructions stored in the memory of the client device 14(1)-14(n) for one or more aspects of the present invention as described and illustrated herein, although the CPU could execute other numbers and types of programmed instructions.

[0020] The input device in each of the client devices 14(1)-14(n) can be used to input selections, such as a request for a particular web page, although the input device could be used to input other types of requests and data and interact with other elements. The input device can include keypads, touch screens, and/or vocal input processing systems, although other types and numbers of input devices can be used.

[0021] The display device in each of the client devices 14(1)-14(n) can be used to show data and information to the user, such as web pages retrieved from the server devices 16(1)-16(n) by way of example only. The display device in each of the client devices 14(1)-14(n) can be a mobile phone screen display, although other types and numbers of displays could be used depending on the particular type of client device.

[0022] The input/output device in each of the client devices 14(1)-14(n) can be used to operatively couple and communicate between the client devices 14(1)-14(n), the

web content proxy server 12, and the server devices 16(1)-16(n) over the communication networks 18(1)-18(2).

[0023] Each of the server devices 16(1)-16(n) provides content including web pages for use by one or more of the client devices 14(1)-14(n) via the web content proxy server 12, although the server devices 16(1)-16(n) can provide other numbers and types of content and perform other functions. Each of the server devices 16(1)-16(n) can include a CPU, a memory, and an input/output device, which are coupled together by a bus or other link, although each of the server devices 16(1)-16(n) could have other numbers and types of components, parts, devices, systems, and elements in other configurations and locations.

[0024] The CPU in each of the server devices 16(1)-16(n) executes a program of instructions stored in the memory of the server devices 16(1)-16(n) for one or more aspects of the present invention, as described and illustrated by way of the embodiments herein, although the CPU could execute other numbers and types of programmed instructions.

[0025] The input/output device in each of the server devices 16(1)-16(n) is used to operatively couple and communicate between the server devices 16(1)-16(n), the web content proxy server 12, and the client devices 14(1)-14(n) via the communication networks 18(1)-18(2).

[0026] Although embodiments web content proxy server 12, the client devices 14(1)-14(n), and the server devices 16(1)-16(n) are described and illustrated herein, each of the web content proxy server 12, the client devices 14(1)-14(n), and the server devices 16(1)-16(n) can be implemented on any suitable computer apparatus or computing device. It is to be understood that the apparatuses and devices of the embodiments described herein are for exemplary purposes, as many variations of the specific hardware and software used to implement the embodiments are possible, as will be appreciated by those skilled in the relevant art(s).

[0027] Furthermore, each of the devices of the embodiments may be conveniently implemented using one or more general purpose computers, microprocessors, digital signal processors, and micro-controllers, programmed according to the teachings of the embodiments, as described and illustrated herein, and as will be appreciated by those ordinary skill in the art.

[0028] In addition, two or more computing apparatuses or devices can be substituted for any one of the devices in any embodiment described herein. Accordingly, principles and advantages of distributed processing, such as redundancy and replication also can be implemented, as desired, to increase the robustness and performance of the devices of the embodiments. The embodiments may also be implemented on computer apparatuses or devices that extend across any suitable network using any suitable interface mechanisms and communications technologies, including by way of example only telecommunications in any suitable form (e.g., voice and modem), wireless communications media, wireless communications networks, cellular communications networks, G3 communications networks, Public Switched Telephone Network (PSTNs), Packet Data Networks (PDNs), the Internet, intranets, and combinations thereof.

[0029] The embodiments may also be embodied as one or more non-transitory computer readable medium having instructions stored thereon for one or more aspects of the present invention as described and illustrated by way of the embodiments herein, as described herein, which when executed by a CPU, cause the CPU to carry out the steps necessary to implement the methods of the embodiments, as described and illustrated herein.

[0030] An exemplary method for bundling images will now be described with reference to FIGS. 1-5. In this example, in step 200, the web content proxy server 12 receives a request for a web page from one of the client devices 14(1)-14(n). While in this example, steps 200-224 illustrated in FIG. 2 are performed by the web content proxy server 12, one or more of steps 200-224 could be performed by one of the server devices 16(1)-16(n) or any other web server in communication with the one of the client devices

14(1)-14(n) through one or more of the communication networks 18(1) and 18(2). Accordingly, in this example, the request can be a hypertext transfer protocol (HTTP) request for a web page stored by one of the server devices 16(1)-16(n).

[0031] Upon receipt of the request from one of the client devices 14(1)-14(n), the web content proxy server 12 obtains the requested web page. In this example, the web content proxy server 12 obtains the requested web page by retrieving the requested web page from the one of the server devices 16(1)-16(n) on behalf of the requesting one of the client devices 14(1)-14(n). In other examples, the requested web page can be obtained by retrieving the requested web page from local memory, such as the memory 22, or by generating the requested web page, for example.

[0032] In this example, the requested web page is a hypertext markup language (HTML) document, a fragment 300 of which is illustrated in FIG. 3. The fragment 300 of the HTML document references multiple images including "fb.png" and "twitter.png". The "fb.png" and "twitter.png" images are stored in the "icons" directory, as indicated in the source attribute values of the image elements 302 and 304.

[0033] In step 202, the web content proxy server 12 determines whether there is an image element with a source attribute value identifying a directory with a version file. In the fragment 300, the directory identified by the source attribute value of each of the image elements 302 and 304 is the "icons" directory. Accordingly, the web content proxy server 12 determines whether the icons directory in this example includes a version file, which can be a text file with a default file name, for example, although other file formats can also be used. The version file is created and stored in the icons directory by a developer of the web page and is used to indicate that any images in the icons directory should be bundled when provided to a client device. The version file can also be used for other functions, as described and illustrated later.

[0034] In some examples, not all image elements of the requested web page are bundled and have a source attribute value identifying a directory with a version file. For example, directories in which only one image is stored will require the same number of

communications to retrieve the image as required to retrieve an image bundle, as described and illustrated in more detail later. In other examples, image elements may reference images that are not bundled for any number of other reasons according to developer preference and, in yet other examples, all images of a web page can be bundled.

[0035] Accordingly, if the web content proxy server 12 determines in step 202 that there is an image element with a source attribute value identifying a directory with a version file, then the Yes branch is taken to step 204. In step 204, the web content proxy server modifies the image element to include a data attribute and to replace the source attribute value with a data uniform resource identifier (URI).

[0036] In this example, the inserted data attribute has a value that is equivalent to the source attribute value of the image element. Referring to FIG. 4, an exemplary fragment 400 is illustrated. In the fragment 400, the image element 302 is modified to include a data attribute "data-bundle='icons/fb.png'", among other modifications described and illustrated later, resulting in image element 402. Accordingly, image element 402 includes a data attribute having a value equivalent to the source attribute value "icons/fb.png" of the image element 302. Other methods of preserving the source attribute value of the image element can also be used.

[0037] Additionally, in step 204, the web content proxy server 12 replaces the source attribute value of the image element with a data URI. The data URI is an encoded version of an image that will be rendered in place of the image identified by the source attribute value of the image element 302. The data URI can be a base 64 encoding, although any other encoding can also be used. The data URI is effectively a placeholder, as described and illustrated in more detail later.

[0038] In this example, the source attribute value of the image element 302 is replaced with a data URI of a spacer graphic interchange format (GIF) transparent image in the element 402, although a data URI of any other default image can also be used. By using a spacer GIF image that is transparent, the user of the requesting one of the client

devices 14(1)-14(n) will not see the image when the web page is rendered. Additionally, the overhead of including the data URI of the spacer GIF image is minimal.

[0039] In step 206, the web content proxy server 12 determines whether a script reference associated with the directory identified by the value of the data attribute included in the image element in step 204 has been previously inserted into the web page. A script reference will not have been previously inserted for each directory first encountered by the web content proxy server 12 in a source attribute value of an image element.

[0040] In this example, the directory identified by the data attribute of element 402, as inserted into the image element 302, is the "icons" directory. Since the "icons" directory is encountered for the first time in a first iteration in this example, the script reference will not have been previously included. If the web content proxy server 12 determines that a script reference associated with the directory identified by the value of the data attribute included in the image element in step 204 has not been previously included, then the No branch is taken to step 208.

[0041] In step 208, the web content proxy server 12 inserts a reference to a JavaScript file based on the version file included in the directory identified by the data attribute value of the image element 402. The reference to the JavaScript file can be a script element having a source attribute with a value identifying the JavaScript file or a jQuery function call, for example, although other script references can also be used. The version file includes content which is used as a portion of the file name included in the script reference. The content of the version file is inserted by a developer of the web page when the version file is stored in the directory, and is used to indicate whether a new JavaScript image bundle code should be generated, as described and illustrated in more detail later.

[0042] In this example, the fragment 400 includes a script reference as script element 406 having a source attribute value of "icons/v1.js". Accordingly, the icons directory includes a version file having content "v1" which, along with the JavaScript file

extension ".js" is used to form the "v1.js" name of the JavaScript file in the source attribute value of the script element 406. Other content of the version file and other methods of naming the JavaScript file can also be used.

[0043] Upon inserting the reference to the JavaScript file in step 208, the web content proxy server 12 proceeds to step 202. In step 202, the web content proxy server 12 determines whether there is another image element with a source attribute value identifying a directory with a version file, as described and illustrated earlier. In this example, the next image element 304 of the fragment 300 also has a source attribute value identifying a directory with a version file, as the identified "icons" directory was determined to have a version file, in this example, in the previous iteration of step 202.

[0044] Accordingly, in this second iteration, the Yes branch is taken from step 202 and the image element 304 is modified in step 204 by the web content proxy server 12, as described and illustrated earlier, resulting in image element 404 of fragment 400. In this example, the data URI that replaced the source attribute value of image element 304 is the same data URI used in the prior iteration of step 206 to replace the source attribute value of image element 302, although different data URIs can also be used.

[0045] Upon replacing the source attribute value of the image element 304, the web content proxy server 12 proceeds to step 206. In step 206, the web content proxy server 12 determines whether a script reference associated with the directory identified by the value of the data attribute included in the image element in this second iteration of step 204 has been previously inserted into the web page. In this iteration, the web content proxy server 12 will determine that script element 406 was previously inserted and has a source attribute value identifying a directory ("icons") matching the data attribute value of image element 404. Accordingly, in the second iteration in this example, the Yes branch is taken by the web content proxy server 12 from step 206 to step 202.

[0046] In step 202, the web content proxy server 12 determines whether there is another image element with a source attribute value identifying a directory with a version file, as described and illustrated earlier. In the fragment 300, the only other image

element 306 does not include a source attribute value identifying a directory with a version file. Instead, the source attribute value of the image element 306 is a uniform resource locator (URL) that does not identify any directory. As there are no other image elements with a source attribute value identifying a directory with a version file, the No branch is taken to step 210.

[0047] In step 210, the web content proxy server 12 sends the web page to the requesting one of the client devices 14(1)-14(n). In other examples, the web content proxy server 12 can mark the directory, or otherwise maintain a list of directories in step 208, and include each of the script references at substantially the same time prior to sending the web page to the requesting one of the client devices 14(1)-14(n) in step 210. In these examples, the web content proxy server 12 can determine in step 206 whether the data attribute of the image element identifies a directory that has been previously marked or included in the list of directories for which a script reference is to be inserted.

[0048] In step 212, the web content proxy server 12 receives a request for a JavaScript file identified by one of the script references inserted in step 208 from the requesting one of the client devices 14(1)-14(n). The requested is sent by the one of the client devices 14(1)-14(n) upon encountering the script reference while rendering the web page.

[0049] In step 214, the web content proxy server 12 extracts a directory and file name from a path included in the request. The request can be an HTTP request including a URL path identifying the JavaScript file. In this example, the JavaScript file will be identified in the URL path of the HTTP request as "icons/v1.js" based on the source attribute value of the script reference 406 and, accordingly, the web content proxy server 12 will extract "icons" as the directory and "v1.js" as the file name from the URL path.

[0050] In step 216, the web content proxy server 12 determines whether the requested JavaScript file was previously generated. The web content proxy server 12 can determine whether the requested JavaScript file was previously generated based on whether a JavaScript file with the extracted file name is stored in the extracted directory.

If the web content proxy server 12 determines that the requested JavaScript file was not previously generated, then the No branch is taken to step 218.

[0051] In step 218, the web content proxy server 12 generates a data URI map for each image in the extracted directory. Each of the data URIs can be generated by generating an encoding of each of the images, such as a base 64 encoding, for example, although other encodings can also be used. Additionally, each of the data URIs is mapped to a directory and file name corresponding to the images encoded by the data URI. Accordingly, irrespective of the number of references to each of the images in the web page, only one encoding of each of the images is generated.

[0052] In step 220, the web content proxy server 12 generates the requested JavaScript file based on the data URIs generated in step 218. The generated JavaScript file includes JavaScript code that, when executed by the one of the client devices 14(1)-14(n), is configured to replace the source attribute value of each of the image elements with one of the data URIs. The source attribute values of the image elements are replaced based on a match of the value of the data attribute of the image elements inserted in step 204 with the directory and file name mapped to one of the data URIs. Optionally, the JavaScript code is also configured to remove the data attribute from each of the image elements.

[0053] Referring to FIG. 5, an exemplary JavaScript code fragment 500 of a JavaScript file is illustrated. In this example, the fragment 500 includes a data URI 502 mapped to the "icon/fb.png" directory and file name and a data URI 504 mapped to the "icons/twitter.png" directory and file name. The fragment 500 is configured to retrieve the data attribute of all of the image elements of the web page. For each image element with a data attribute, the fragment 500 is configured to determine whether the value of the data attribute matches a directory and file name mapped to one of the data URIs. When it is determined that the data attribute value matches a directory and file name mapped to one of the data URIs, the fragment 500 is configured to replace the source attribute value of with the one data URI.

[0054] In step 222, the web content proxy server 12 stores the generated JavaScript file in the directory extracted in step 214. The JavaScript file is named according to the file name extracted in step 214, which corresponds with the contents of the version file stored in the extracted directory, as described and illustrated earlier. In this example, the JavaScript file is stored in the "icons" directory with a file name of "v1.js".

[0055] In step 224, the web content proxy server 12 sends the JavaScript file to the one of the client devices 14(1)-14(n) in response to the request received in step 212. Accordingly, in this example the images are sent as a bundle based on the encodings included in the JavaScript file. Accordingly, only one request and response will be required for the one of the client devices 14(1)-14(n) to retrieve all of the images stored in a respective local directory associated with the requested web page. Subsequent to sending the JavaScript file to the requesting one of the client devices 14(1)-14(n) in step 224, the web content proxy server 12 then proceeds to step 212 to receive another request for a JavaScript file from the same or a different one of the client devices 14(1)-14(n).

[0056] Upon receipt of the JavaScript file, the requesting one of the client devices 14(1)-14(n) will execute the JavaScript file resulting in the replacement of the source attribute value for one or more of the image elements of the associated web page with a data URI, as described and illustrated earlier. Subsequent to replacing the source attribute value(s), the one of the client devices 14(1)-14(n) will render the image encoded in and defined by the data URI that replaced the source attribute value for each of the image elements. Accordingly, in this example, source attribute values of the image elements 402 and 404, previously including the data URI of the default spacer GIF image, will be replaced with the data URIs 502 and 504 of the "fb.png" and "twitter.png" images, respectively.

[0057] Referring back to step 216, if the web content proxy server 12 determines that the requested JavaScript file has been previously generated, then the Yes branch is taken to step 224 and the requested JavaScript file is sent to the requesting one of the

client devices 14(1)-14(n). Whenever a change is made to the web page by a developer of the web page (e.g., addition or removal of an image), the developer can cause a new JavaScript file to be generated when requested by one of the client devices 14(1)-14(n) by changing the contents of the version file in the directory of the added or removed images (e.g., from "v1" to "v2").

[0058] By changing the contents of the version file, the reference to the JavaScript file inserted in step 208 will be to a JavaScript file with a file name not matching the file name of a previously generated JavaScript file in the directory (e.g., "v2.js"). Accordingly, the web content proxy server 12 will determine, in step 216, that the requested JavaScript file was not previously generated and will proceed to take the No branch in order to generate a new JavaScript file, which will be configured to insert the appropriate source attribute values of the image elements, as described and illustrated earlier.

[0059] Accordingly, as illustrated and described herein, this technology provides a number of advantages including providing methods, non-transitory computer readable medium, and a web content proxy server that facilitates image bundling and reduce the number of requests and responses required to retrieve web page images. Thereby, the amount of time required to render web pages is significantly reduced, particularly in high latency networks.

[0060] With this technology, data URIs including encodings of web page images are provided in a bundle through a JavaScript file that is configured to modify the web page to include the data URIs in respective image elements. Advantageously, only one encoding of each image is generated irrespective of the number of references to an image. Additionally, the encodings are stored separately from the associated web page in a JavaScript file in local directories associated with the images. Accordingly, the encodings in a directory advantageously do not have to be regenerated in response to web page changes that do not involve the images stored in the directory.

[0061] Having thus described the basic concept of the invention, it will be rather apparent to those skilled in the art that the foregoing detailed disclosure is intended to be presented by way of example only, and is not limiting. Various alterations, improvements, and modifications will occur and are intended to those skilled in the art, though not expressly stated herein. These alterations, improvements, and modifications are intended to be suggested hereby, and are within the spirit and scope of the invention. Additionally, the recited order of processing elements or sequences, or the use of numbers, letters, or other designations therefore, is not intended to limit the claimed processes to any order except as may be specified in the claims. Accordingly, the invention is limited only by the following claims and equivalents thereto.

CLAIMS

What is claimed is:

1. A method for bundling images, the method comprising:

obtaining, with a web server, a web page requested by a client device, the web page comprising a plurality of image elements each including a source attribute having a value identifying an image;

modifying, with the web server, each of the plurality of image elements to insert a data attribute having a value of the respective source attribute value and to replace the source attribute value with a data uniform resource indicator (URI);

inserting, with the web server, a reference to an executable file into the requested web page, sending the requested web page to the client device, and receiving a request from the client device for the executable file; and

sending, with the web server, the executable file to the client device in response to the request, the executable file configured when executed to replace the source attribute value of each of the plurality of image elements with a data URI of an image identified by the respective data attribute value.

2. The method of claim 1, wherein the source attribute value of each of the plurality of image elements is replaced with a data URI of a spacer graphic interchange format (GIF) transparent image.

3. The method of claim 1, wherein the source attribute of one or more of the image elements has a value identifying a different directory than one or more other of the plurality of image elements and the reference to the executable file comprises a reference to a different executable file for each directory.

4. The method of claim 1, wherein the reference identifies the executable file based on a directory identified by the value of the source attribute

included in the plurality of image elements and contents of a version file in the directory, the contents including at least a portion of a file name of the executable file.

5. The method of claim 1, wherein the reference to the executable file is a script element or a jQuery function call, the script element comprising a source attribute with a value identifying the executable file.

6. The method of claim 1, wherein the sending the executable file further comprises:

extracting at least a directory and a file name from a path included in the request received from the client device for the executable file;

determining whether the executable file is stored in the directory based on the file name; and

generating and storing the executable file, when it is determined that the executable file is not stored in the directory, the generating the executable file further comprising generating a data URI for each image in the directory.

7. The method of claim 6, wherein the generating the executable file further comprises generating a data URI for a cascading style sheet (CSS) background image stored in the directory and the executable file is further configured to, when executed by the client device, insert a style element into the web page, the style element including the data URI of the CSS background image.

8. A non-transitory computer readable medium having stored thereon instructions for bundling images comprising machine executable code which when executed by a processor, causes the processor to perform steps comprising:

obtaining a web page requested by a client device, the web page comprising a plurality of image elements each including a source attribute having a value identifying an image;

modifying each of the plurality of image elements to insert a data attribute having a value of the respective source attribute value and to replace the source attribute value with a data uniform resource indicator (URI);

inserting a reference to a executable file into the requested web page, sending the requested web page to the client device, and receiving a request from the client device for the executable file; and

sending the executable file to the client device in response to the request, the executable file configured when executed to replace the source attribute value of each of the plurality of image elements with a data URI of an image identified by the respective data attribute value.

9. The medium of claim 8, wherein the source attribute value of each of the plurality of image elements is replaced with a data URI of a spacer graphic interchange format (GIF) transparent image.

10. The medium of claim 8, wherein the source attribute of one or more of the image elements has a value identifying a different directory than one or more other of the plurality of image elements and the reference to the executable file comprises a reference to a different executable file for each directory.

11. The medium of claim 8, wherein the reference identifies the executable file based on a directory identified by the value of the source attribute included in the plurality of image elements and contents of a version file in the directory, the contents including at least a portion of a file name of the executable file.

12. The medium of claim 8, wherein the reference to the executable file is a script element or a jQuery function call, the script element comprising a source attribute with a value identifying the executable file.

13. The medium of claim 8, wherein the sending the executable file further comprises:

extracting at least a directory and a file name from a path included in the request received from the client device for the executable file;

determining whether the executable file is stored in the directory based on the file name; and

generating and storing the executable file, when it is determined that the executable file is not stored in the directory.

14. The medium of claim 13, wherein the generating the executable file further comprises generating a data URI for a cascading style sheet (CSS) background image stored in the directory and the executable file is further configured to, when executed by the client device, insert a style element into the web page, the style element including the data URI of the CSS background image.

15. A web server device, comprising:

a processor coupled to a memory and configured to execute programmed instructions stored in the memory comprising:

obtaining a web page requested by a client device, the web page comprising a plurality of image elements each including a source attribute having a value identifying an image;

modifying each of the plurality of image elements to insert a data attribute having a value of the respective source attribute value and to replace the source attribute value with a data uniform resource indicator (URI);

inserting a reference to a executable file into the requested web page, sending the requested web page to the client device, and receiving a request from the client device for the executable file; and

sending the executable file to the client device in response to the request, the executable file configured when executed to replace the source

attribute value of each of the plurality of image elements with a data URI of an image identified by the respective data attribute value.

16. The device of claim 15, wherein the source attribute value of each of the plurality of image elements is replaced with a data URI of a spacer graphic interchange format (GIF) transparent image.

17. The device of claim 15, wherein the source attribute of one or more of the image elements has a value identifying a different directory than one or more other of the plurality of image elements and the reference to the executable file comprises a reference to a different executable file for each directory.

18. The device of claim 15, wherein the reference identifies the executable file based on a directory identified by the value of the source attribute included in the plurality of image elements and contents of a version file in the directory, the contents including at least a portion of a file name of the executable file.

19. The device of claim 15, wherein the reference to the executable file is a script element or a jQuery function call, the script element comprising a source attribute with a value identifying the executable file.

20. The device of claim 15, wherein the sending the executable file further comprises:

extracting at least a directory and a file name from a path included in the request received from the client device for the executable file;

determining whether the executable file is stored in the directory based on the file name; and

generating and storing the executable file, when it is determined that the executable file is not stored in the directory.

21. The device of claim 20, wherein the generating the executable file further comprises generating a data URI for a cascading style sheet (CSS) background image stored in the directory and the executable file is further configured to, when executed by the client device, insert a style element into the web page, the style element including the data URI of the CSS background image.

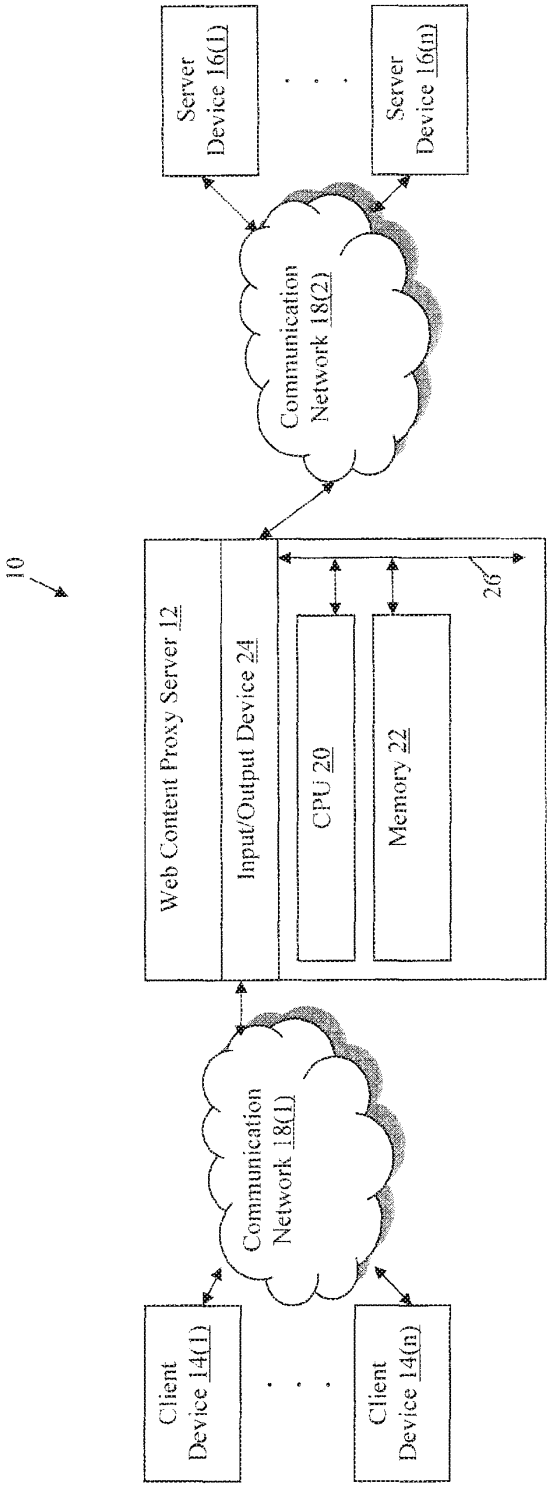

10
Web Content Proxy Server 12
Input/Output Device 24
CPU 20
Memory 22
26
Server Device 16(1)
Server Device 16(n)
Communication Network 18(2)
Communication Network 18(1)
Client Device 14(1)
Client Device 14(n)

FIG. 1

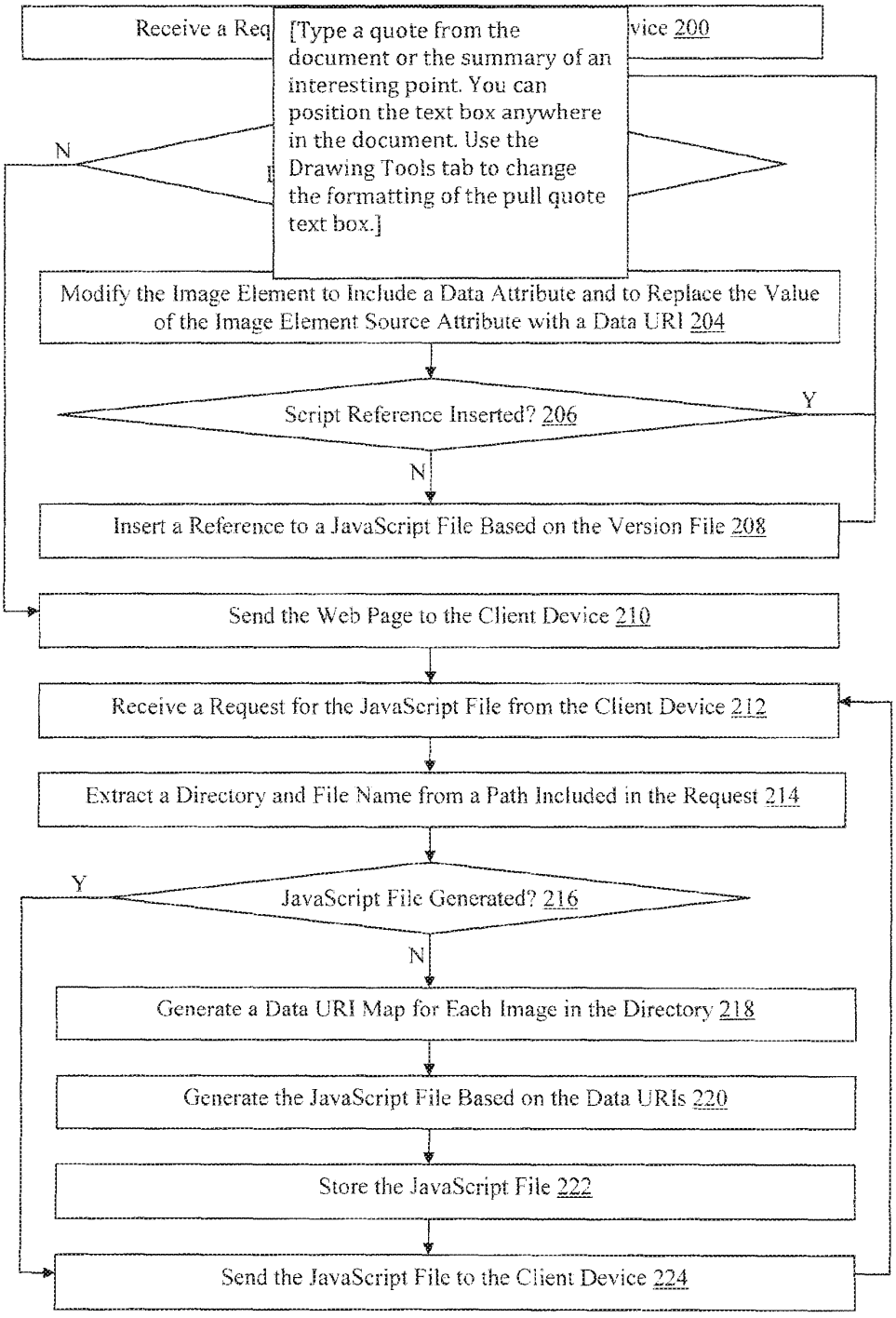

Receive a Req
vice 200
[Type a quote from the document or the summary of an interesting point. You can position the text box anywhere in the document. Use the Drawing Tools tab to change the formatting of the pull quote text box.]
N
Modify the Image Element to Include a Data Attribute and to Replace the Value of the Image Element Source Attribute with a Data URI 204
Script Reference Inserted? 206
Y
N
Insert a Reference to a JavaScript File Based on the Version File 208
Send the Web Page to the Client Device 210
Receive a Request for the JavaScript File from the Client Device 212
Extract a Directory and File Name from a Path Included in the Request 214
Y
JavaScript File Generated? 216
N
Generate a Data URI Map for Each Image in the Directory 218
Generate the JavaScript File Based on the Data URIs 220
Store the JavaScript File 222
Send the JavaScript File to the Client Device 224

FIG. 2

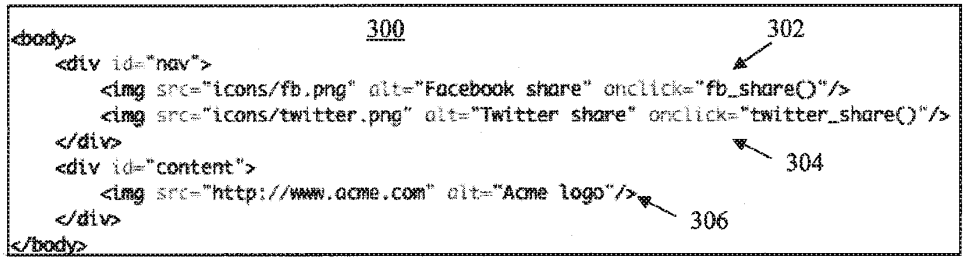


FIG. 3

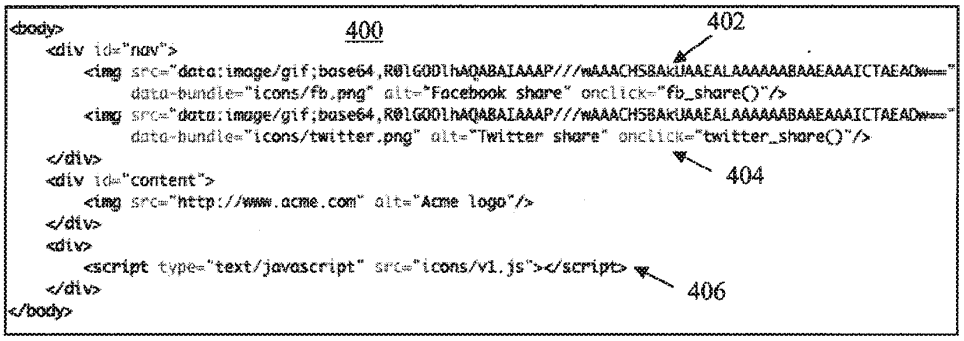


FIG. 4

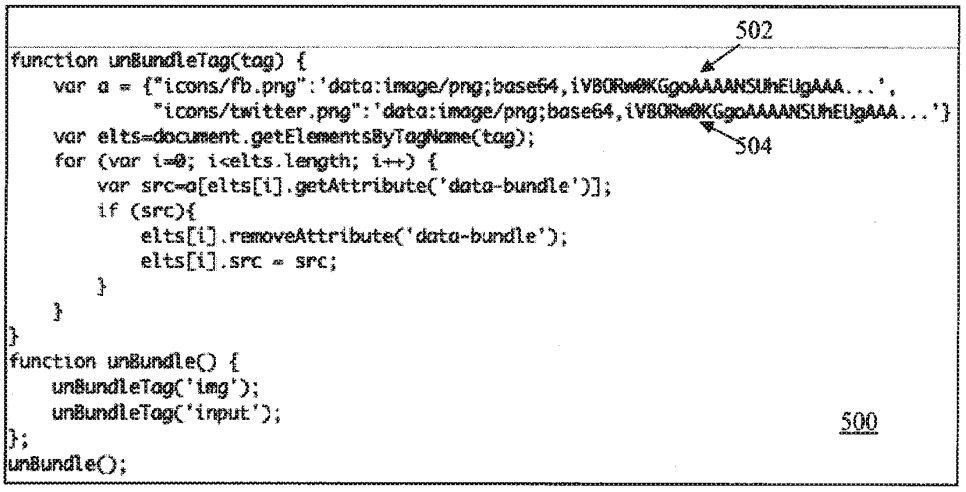


FIG. 5

10
Client Device 14(1)
Client Device 14(n)
Communication Network 18(1)
Web Content Proxy Server 12
Input/Output Device 24
CPU 20
Memory 22
26
Communication Network 18(2)
Server Device 16(1)
Server Device 16(n)

Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2858590 A1 2015/02/28

(21) 2 858 590

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(22) Date de dépôt/Filing Date: 2014/08/07

(41) Mise à la disp. pub./Open to Public Insp.: 2015/02/28

(30) Priorité/Priority: 2013/08/28 (US14/012,051)

(51) Cl.Int./Int.Cl. *H04L 12/16* (2006.01), *G06F 9/44* (2006.01), *G06F 9/46* (2006.01)

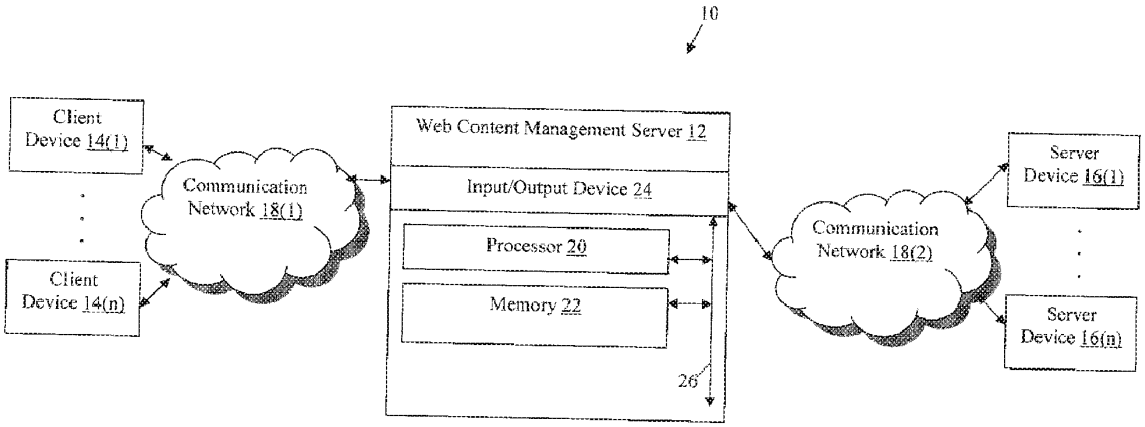
(71) Demandeur/Applicant:
USABLENET INC., US

(72) Inventeur/Inventor:
SCODA, ENRICO, IT

(74) Agent: PARLEE MCLAWS LLP

(54) Titre : PROCEDES DE TRAITEMENT DES DEMANDES DE SERVICE WEB AU MOYEN DE SERVICES WEB AGILES PARALLELES ET LEURS DISPOSITIFS

(54) Title: METHODS FOR SERVICING WEB SERVICE REQUESTS USING PARALLEL AGILE WEB SERVICES AND DEVICES THEREOF



(57) Abrégé/Abstract:
A method, non-transitory computer readable medium, and web content management server device that sends each of a plurality of jobs requiring execution in order to service a received web service request to one of a plurality of slave web services configured to execute the plurality of jobs in parallel. A response from each of the plurality of slave web services is received. A web service response is generated based on the received responses. The generated web service response is provided in response to the received web service request.

ABSTRACT

A method, non-transitory computer readable medium, and web content management server device that sends each of a plurality of jobs requiring execution in order to service a received web service request to one of a plurality of slave web services configured to execute the plurality of jobs in parallel. A response from each of the plurality of slave web services is received. A web service response is generated based on the received responses. The generated web service response is provided in response to the received web service request.

METHODS FOR SERVICING WEB SERVICE REQUESTS USING PARALLEL AGILE WEB SERVICES AND DEVICES THEREOF

FIELD

[0001] This technology generally relates to methods and devices for optimizing delivery of web content and, more particularly, to methods for servicing web service requests using parallel agile web services and devices thereof.

BACKGROUND

[0002] Web services provide a standardized way of integrating web-based applications traditionally using eXtensible Markup Language (XML), SOAP, Web Services Description Language (WSDL), and/or Universal Description Discovery, for example, and Integration (UDDI) standards over an Internet Protocol (IP) backbone. XML can be used to tag data used by a web service, SOAP can be used to transfer the data, WSDL can be used for describing the web services available, and UDDI can be used for listing the available web services. Web services allow different applications located at different sources to communicate with each other efficiently and without custom coding which can require a significant amount of resources. Additionally, because communications can be in XML, web services are not tied to any operating system or programming language. Recently, JavaScript Object Notation (JSON) has emerged as a standard object-based messaging format alternative to XML that can be used with the Hypertext Transfer Protocol (HTTP) to generate web services for web applications.

[0003] Unlike traditional client/server models, web services do not provide an end user with a graphical user interface (GUI). Instead, web services share data and processes through an application interface across a network. These application interfaces are invoked and used to interpret any resulting data. Web services are increasingly popular since it is relatively easy to integrate them into applications to extend the features offered to end users. However, web service requests are also increasingly complex. For example, many web services require content from various sources to be mashed-up in

order to provide a response. Accordingly, the processing of web service requests often requires significant resources, as well as a significant amount of time.

SUMMARY

[0004] A method for servicing web service requests using parallel agile web services includes sending, with a web content management server, each of a plurality of jobs requiring execution in order to service a received web service request to one of a plurality of slave web services configured to execute the plurality of jobs in parallel. A response from each of the plurality of slave web services is received with the web content management server. A web service response is generated, with the web content management server, based on the received responses. The generated web service response is provided, with the web content management server, in response to the received web service request

[0005] A non-transitory computer readable medium having stored thereon instructions for servicing web service requests using parallel agile web services comprising machine executable code which when executed by a processor, causes the processor to perform steps including sending each of a plurality of jobs requiring execution in order to service a received web service request to one of a plurality of slave web services configured to execute the plurality of jobs in parallel. A response from each of the plurality of slave web services is received. A web service response is generated based on the received responses. The generated web service response is provided in response to the received web service request.

[0006] A web content management server device includes a processor coupled to a memory and configured to execute programmed instructions stored in the memory including sending each of a plurality of jobs requiring execution in order to service a received web service request to one of a plurality of slave web services configured to execute the plurality of jobs in parallel. A response from each of the plurality of slave web services is received. A web service response is generated based on the received

responses. The generated web service response is provided in response to the received web service request.

[0007] This technology provides a number of advantages including methods, non-transitory computer readable medium, and devices that facilitate more efficient and effective processing of web service requests using parallel slave web services to retrieve content required to generate a web service response. The parallel slave web services advantageously run in an emulated JavaScript environment which is capable of executing other web service functionality implemented using JavaScript. Accordingly, with this technology, web service requests can be processed by an increased number of devices, including mobile devices and tablets with special requirements, and in relatively less time.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] FIG. 1 is a block diagram of an environment with an exemplary web content management server;

[0009] FIG. 2 is a flow chart of an exemplary method for processing web service requests with a master web service using parallel agile web services;

[0010] FIG. 3 is a flow chart of an exemplary method for processing with parallel agile web service jobs received from a master web service;

[0011] FIG. 4 is an exemplary code fragment of an exemplary master web service configured to perform the exemplary method of FIG. 2;

[0012] FIG. 5 is an exemplary code fragment of an exemplary slave web service configured to perform the exemplary method of FIG. 3; and

[0013] FIG. 6 is an exemplary configuration file defining the role of an exemplary slave web service.

DETAILED DESCRIPTION

[0014] An exemplary network environment 10 with a web content management server 12 coupled to client devices 14(1)-14(n) and server devices 16(1)-16(n) is illustrated in FIG. 1. In this example, the web content management server 12, client devices 14(1)-14(n), and server devices 16(1)-16(n) are coupled together by communication networks 18(1)-18(2), although other numbers and types of systems, devices, and/or elements in other configurations or network topologies can also be used. This technology provides a number of advantages including methods, non-transitory computer readable medium, and devices that facilitate more efficient and effective processing of web service requests using parallel slave web services to retrieve content required to generate a web service response.

[0015] The web content management server 12 is coupled to the client devices 14(1)-14(n) by the communication network 18(1) which can include one or more local area network(s) (LANs) and/or wide area network(s) (WANs). In this example, the web content management server 12 is further coupled to the server devices 16(1)-16(n) by the communication network 18(2), which may also include one or more LANs and/or WANs. Other network devices configured to generate, send, and receive network communications and coupled together via other topologies can also be used. While not shown, the network environment 10 also may include additional network components, such as routers, switches and other devices, which are well known to those of ordinary skill in the art and thus will not be described here.

[0016] The web content management server 12 may perform any number of functions including optimizing content retrieved from the server devices 16(1)-16(n) for display on the client devices 14(1)-14(n), for example. In this example the web content management server 12 includes a processor 20, a memory 22, and an input/output system 24, which are coupled together by a bus 26 or other link, although other numbers and types of components, parts, devices, systems, and elements in other configurations and locations can be used.

[0017] The processor 20 in the web content management server 12 executes a program of stored instructions one or more aspects of the present invention, as described and illustrated by way of the embodiments herein, although the processor 20 could execute other numbers and types of programmed instructions. The processor 20 of the web content management server 12 may comprise one or more central processing units or general purpose processors with one or more processing cores, for example.

[0018] The memory 22 in the web content management server 12 stores these programmed instructions for one or more aspects of the present invention, as described and illustrated herein, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, DVD ROM, or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to the processor 20, can be used for the memory 22 in the web content management server 12.

[0019] The input/output system 24 in the web content management server 12 is used to operatively couple and communicate between the web content management server 12, client devices 14(1)-14(n), and server devices 16(1)-16(n), which are all coupled together via the communication networks 18(1)-18(2), although other types and numbers of communication networks or systems with other types and numbers of connections and configurations to other devices and elements can also be used. By way of example only, the communication networks 18(1)-18(2) can use TCP/IP over Ethernet and industry-standard protocols, including hypertext transfer protocol (HTTP), secure HTTP (HTTPS), wireless application protocol (WAP), and/or SOAP, although other types and numbers of communication networks, such as a direct connection, modems and phone lines, e-mail, and wireless and hardwire communication technology, each having their own communications protocols, can be used.

[0020] The client devices 14(1)-12(n) enable a user to request, receive, and interact with applications, web services, and content hosted by the server devices 16(1)-16(n) through the web content management server 12 and using the communication network 18(1), although one or more of the client devices 14(1)-14(n) could access content and utilize other types and numbers of applications from other sources and could provide a wide variety of other functions for the user. In some examples, the client devices 14(1)-14(n) comprise mobile devices with Internet access that enable web pages and other content stored by the server devices 16(1)-16(n) to be retrieved and rendered. By way of example only, the client devices 14(1)-14(n) can be smart phones, personal digital assistants, tablets, or computers.

[0021] Each of the client devices 14(1)-14(n) includes a processor, a memory, an input device, a display device, and an input/output system, which are coupled together by a bus or other link, although one or more of client devices 14(1)-14(n) can include other numbers and types of components, parts, devices, systems, and elements in other configurations. The processor in each of the client devices 14(1)-14(n) can execute a program of instructions stored in the memory the client device 14(1)-14(n) for one or more aspects of the present invention, as described and illustrated herein, although the processor could execute other numbers and types of programmed instructions.

[0022] The input device in each of the client devices 14(1)-14(n) can be used to input selections, such as a request for a particular web page or other content stored by one or more of the server devices 16(1)-16(n), although the input device could be used to input other types of requests and data and interact with other elements. The input device can include keypads, touch screens, and/or vocal input processing systems, although other types and numbers of input devices can also be used.

[0023] The display device in each of the client devices 14(1)-14(n) can be used to show data and information to the user, such as web pages and other content retrieved from the server devices 16(1)-16(n) by way of example only. The display device in each of the client devices 14(1)-14(n) can be a mobile phone screen display, although other

types and numbers of displays could be used depending on the particular type of client device.

[0024] The input/output system in each of the client devices 14(1)-14(n) can be used to operatively couple and communicate between the client devices 14(1)-14(n), the web content management server 12, and the server devices 16(1)-16(n) over the communication networks 18(1)-18(2).

[0025] Each of the server devices 16(1)-16(n) provides content including web pages and web applications for use by one or more of the client devices 14(1)-14(n) via the web content management server 12, although the server devices 16(1)-16(n) can provide other numbers and types of content and perform other functions. Each of the server devices 16(1)-16(n) can include a processor, a memory, and an input/output system, which are coupled together by a bus or other link, although each of the server devices 16(1)-16(n) can have other numbers and types of components, parts, devices, systems, and elements in other configurations.

[0026] The processor in each of the server devices 16(1)-16(n) executes a program of instructions stored in the memory of the server devices 16(1)-16(n) for one or more aspects of the present invention, as described and illustrated by way of the embodiments herein, although the processor could execute other numbers and types of programmed instructions.

[0027] The input/output system in each of the server devices 16(1)-16(n) is used to operatively couple and communicate between the server devices 16(1)-16(n), the web content management server 12, and the client devices 14(1)-14(n) via the communication networks 18(1)-18(2).

[0028] Although embodiments web content management server 12, the client devices 14(1)-14(n), and the server devices 16(1)-16(n) are described and illustrated herein, each of the web content management server 12, the client devices 14(1)-14(n), and the server devices 16(1)-16(n) can be implemented on any suitable computer

apparatus or computing device. It is to be understood that the apparatuses and devices of the embodiments described herein are for exemplary purposes, as many variations of the specific hardware and software used to implement the embodiments are possible, as will be appreciated by those skilled in the relevant art(s).

[0029] Furthermore, each of the devices of the embodiments may be conveniently implemented using one or more general purpose computers, microprocessors, digital signal processors, and micro-controllers, programmed according to the teachings of the embodiments, as described and illustrated herein, and as will be appreciated by those ordinary skill in the art.

[0030] In addition, two or more computing apparatuses or devices can be substituted for any one of the devices in any embodiment described herein. Accordingly, principles and advantages of distributed processing, such as redundancy and replication also can be implemented, as desired, to increase the robustness and performance of the devices of the embodiments. The embodiments may also be implemented on computer apparatuses or devices that extend across any suitable network using any suitable interface mechanisms and communications technologies, including by way of example only telecommunications in any suitable form (e.g., voice and modem), wireless communications media, wireless communications networks, cellular communications networks, G3 communications networks, Public Switched Telephone Network (PSTNs), Packet Data Networks (PDNs), the Internet, intranets, and combinations thereof.

[0031] The embodiments may also be embodied as one or more non-transitory computer readable medium having instructions stored thereon for one or more aspects of the present invention as described and illustrated by way of the embodiments herein, as described herein, which when executed by a processor, cause the processor to carry out the steps necessary to implement the methods of the embodiments, as described and illustrated herein.

[0032] An exemplary method for servicing web service requests using parallel agile web services will now be described with reference to FIGS. 1-6. Referring more

specifically to FIG. 2, a flow chart of an exemplary method for processing web service requests with a master web service using parallel agile web services is illustrated. In this example, in step 200, the web content management server 12 receives a web service request from one of the client devices 14(1)-14(n). The web service request can be for social media posts, blog summaries, or any other content stored by one or more of the server devices 16(1)-16(n), for example. In the example described and illustrated herein, the received web service request is for the last five social media (e.g., Facebook®) posts for each of twenty users of a social media service associated with a user (e.g., Facebook® friends) of the requesting one of the client devices 14(1)-14(n).

[0033] In step 202, the web content management server 12 determines whether the request received in step 200 requires execution of a plurality of jobs in order to service the request. In the example described herein, the web content management server 12 may determine that the request requires execution of twenty jobs, one for each of the users for which the last five posts have been requested. In this example, the social media service will have to be called twenty times in order to retrieve the last five posts for each of the twenty users identified in the received web service request.

[0034] In other examples, the received web service request may be for the last ten posts for only one user of the social media service, in which case the web content management server 12 may determine in step 202 that a plurality of jobs are not required in order to service the request. The web content management server 12 can be configured by an administrator, for example, to determine that a received web service request requires execution of a plurality of jobs based on any criteria or type of analysis of the web service request.

[0035] Additionally, an administrator of the web content management server 12 can define a threshold number of jobs requiring execution in order to satisfy the condition in step 202. Alternatively, the condition in step 202 can be satisfied whenever the web content management server 12 determines more than one job must be executed in order to service the received web service request. If the web content management server 12

determines in step 202 that the web service request received in step 200 does not require execution of a plurality of jobs, then the web content management server 12 proceeds to step 204.

[0036] In step 204, the web content management server 12 retrieves content from one or more of the server devices 16(1)-16(n) and generates and sends a web service response to the requesting one of the client devices 14(1)-14(n). Accordingly, if the web content management server 12 determines a plurality of jobs are not required in order to service the received web service request, then the web service request is processed based on one or more sequential requests for content sent by the web content management server 12 to the server devices 16(1)-16(n) on behalf of the requesting one of the client devices 14(1)-14(n). Subsequent to servicing the web service request, or in parallel, the web content management server 12 may retrieve another web service request from one of the client devices 14(1)-14(n) in step 200.

[0037] Referring back to step 202, if the web content management server determines that the web service request received in step 200 does require execution of a plurality of jobs, then the web content management server 12 proceeds to step 206. In step 206, the web content management server 12 executes a master web service in an emulated JavaScript environment, although other types of environments could be used. The emulated JavaScript environment can be generated, for example, as described and illustrated in U.S. Patent Application No. 12/802,670, entitled "Methods for Utilizing a JavaScript Emulator in a Web content management server and Devices Thereof," which is incorporated by reference herein in its entirety, although other methods of generating an emulated JavaScript environment can also be used.

[0038] In step 208, the master web service executed by the web content management server 12 inserts the plurality of jobs requiring execution in order to service the web service request into a queue. The queue can be maintained in the memory 22 for example, although the queue can also be stored elsewhere. In the example described

earlier, the master web service inserts the twenty jobs that require a call to the social media service for the last five social media posts for one of the twenty specified users.

[0039] In step 210, the master web service executed by the web content management server 12 determines whether the queue is empty. In a first iteration, the web content management server 12 will determine in step 210 that the queue is not empty. However, in subsequent iterations, all of the jobs may have been sent by the master web service to threads configured to request execution of each of the jobs by one of a plurality of slave web services resulting in an empty queue. If the master web service executed by the web content management server 12 determines that the queue is not empty, then the No branch is taken to step 212.

[0040] In step 212, the master web service executed by the web content management server 12 determines whether a thread is available to receive one of the jobs stored in the queue. In a first iteration, the web content management server 12 will determine in step 212 that a thread is available. However, in subsequent iterations, slave web services may be executing jobs sent by each of the threads and a thread may not be therefore be available.

[0041] The total number of slave web services can be established by an administrator of the web content management server 12, for example. The number of slave web services can be based on a number of threads or processes the system administrator wants to allow the web content management server 12 to execute concurrently, although the number of slave web services can be based on any other criteria. Additionally, web services can be designated as master or slave based on a configuration established by an administrator of the web content management server 12, as described and illustrated in more detail later with reference to FIG. 6. Accordingly, if the master web service executed by the web content management server 12 determines that a slave web service is available, then the Yes branch is taken to step 214.

[0042] In step 214, the master web service executed by the web content management server 12 generates a thread and sends one of the jobs from the queue to the

thread along with an index. The thread is configured to request execution of the one of the jobs by one of the available slave web services. The index will be returned to the master web service context and used to reorder the content received from the slave web services, as described and illustrated in more detail later with reference to step 222. Upon sending one of the jobs from the queue, the master web service again determines whether the queue is empty in step 210.

[0043] Accordingly, in this example, steps 210-214 are performed for each of the jobs and the master web service proceeds to send a job from the queue to a generated thread configured to request execution of the job from one of the slave web services until the master web service determines that the queue is empty in step 210 or that no more threads are available in step 212. For example, assuming that none of the slave web services have processed a job as requested by one of the threads, in this example, the master web service will determine in an eleventh iteration of step 212 that a thread is not available.

[0044] If the master web service determines that a thread is not available, then the No branch is taken from step 212 back to step 212 and the master web service effectively waits until it determines a thread has finished processing a response sent by an associated slave web service, and is available to receive another job, prior to sending one of the jobs from the queue to the generated thread in step 214. Subsequent to generating a thread and sending another one of the jobs and an index to the generated thread in step 214, the master web service proceeds back to step 210, as described and illustrated earlier. Upon the master web service determining, in step 210, that the queue is empty, the Yes branch is taken to step 216.

[0045] In step 216, one of the threads generated by the master web service executed by the web content management server 12 receives a response from one of the slave web services. In this example, the response is a JavaScript Object Notation (JSON) response, although the response can be any type of response. The response includes content obtained from one or more of the server devices 16(1)-16(n) that satisfies the job

the slave web service was requested to execute by the thread, as described and illustrated in more detail later with reference to FIG. 3.

[0046] In step 218, the one of the threads generated by the master web service executed by the web content management server 12 converts the received response into a JavaScript object and passes the JavaScript object and an index to a callback function executed in the master web service context, although the response and/or index can be converted and/or passed in other manners. The index passed in step 218 is the same index sent by the master web service to the one of the threads in step 214. In this example, the thread generated by the master web service converts the received response by using the JSON.parse() function, although other methods of converting the response received from the slave web service can be used.

[0047] In step 220, the master web service executed by the web content management server 12 determines whether there are more responses required to satisfy the web service request received in step 200. The master web service can determine whether there are more responses based on whether it has received a response from the threads corresponding to each of the indices sent along with one of the jobs in step 214. Alternatively, one or more of steps 210-220 can be performed by the master web service within the context of a function call such that control will only return to the master web service context when all of the slave web services have returned a response which is received by the master web service in step 216.

[0048] If the master web service determines in step 220 that more responses are required to satisfy the web service requested received in step 200, then the Yes branch is taken to step 216. Steps 216-220 are repeated until the master web service determines in step 220 that no more responses from the slave web services are required. In this particular example, steps 216-220 will be repeated twenty times or one time for each of the jobs sent to the threads in step 214. If the master web service determines in step 220 that no more responses are required, then the No branch is taken to step 222.

[0049] In this example, steps 216-220 proceed in parallel with steps 210-214 such that a response may be received by the master web service from one of the threads prior to the queue being empty and all of the jobs being sent to one of the threads. Since the slave web services process the jobs in parallel, the responses from the slave web services can be received from the threads by the master web service in any order, generally depending on the time required for each slave web service to complete its job and retrieve the associated content from one or more of the server devices 16(1)-16(n).

[0050] In step 222, the master web service executed by the web content management server 12 generates a web service response based on the responses received in step 216, as converted in step 218, and the indices. As described and illustrated earlier, the threads can optionally pass the converted responses to a callback function in the master web service context in step 218. The callback function can be configured to store the JavaScript objects in a common location, optionally as associated with a corresponding one of the indices, extract one or more properties of the JavaScript objects, and/or perform any number of other functions.

[0051] Accordingly, upon control returning to the master web service context, the master web service can generate a response to the web service request received in step 200 by retrieving the JavaScript objects from the common location, for example. Once retrieved, the master web service can order the JavaScript objects, and/or content included therein, based on the index associated with each of the JavaScript objects, to formulate the web service response.

[0052] In some examples, a plurality of different callback functions configured to execute different functionality can be provided in the master web service context and used by a thread depending on the job sent to the thread by the master web service. For example, the master web service may send a first subset of the threads jobs associated with a first social media service and a second subset of the threads jobs associated with a second social media service. In this example, the master web service may provide, and the threads may pass JavaScript objects and indices to, different callback functions

depending on the job received from the master web service in step 214, and the master web service can thereby mash up converted responses from the various slave web services to generate a web service response.

[0053] In step 224, the master web service executed by the web content management server 12 sends the generated web service response to the requesting one of the client device 14(1)-14(n). Upon sending the web service response with the master web service, or during any of steps 202-224, the web content management server 12 can receive another web service request from one of the client devices 14(1)-14(n) in step 200.

[0054] Referring more specifically to FIG. 3, a flow chart of an exemplary method for processing with parallel agile web service jobs received from a master web service is illustrated. In this example, in FIG. 300, one of the slave web services executed by the web content management server 12 receives one of the jobs, such as from a thread generated by the master web service, as described and illustrated earlier with reference to step 214 of FIG. 2.

[0055] In step 302, the slave web service executed by the web content management server 12 retrieves content satisfying the job received in step 300 from one or more of the server devices 16(1)-16(n). Accordingly, in this example, the slave web service retrieves the last five posts for one of the users of the social media service specified in the received job. The retrieved content can be HTML web pages, XML documents, RESTful web services, for example, or any other type of content.

[0056] In step 304, the slave web service executed by the web content management server 12 generates a response including the retrieved content. In this example, the response is a JSON response, which can be generated using the JSON.stringify() function, for example, although other methods of generating a JSON response and other types of responses can also be used.

[0057] In step 306, the slave web service executed by the web content management server 12 sends the response to the thread from which the job was received. After processing the response received from the slave web service, the thread from which the job was received becomes available to receive another job, as described and illustrated earlier with reference to step 212 of FIG. 2.

[0058] In this example, steps 300-306 are performed in parallel for each of the slave web services receiving one of the jobs from a thread generated by the master web service in step 214 of FIG. 2. By breaking the web service request into a plurality of jobs, and processing the jobs in parallel with a plurality of slave web services, the time required for the web content management server 12 to generate a response to the web service request can be reduced.

[0059] Referring more specifically to FIG. 4, an exemplary code fragment 400 of an exemplary master web service configured to perform the exemplary method of FIG. 2 is illustrated. In this example, the master web service uses "counter" and "generic" callback functions, as described and illustrated earlier with reference to step 218 of FIG. 2, depending on whether a response is successfully or unsuccessfully, respectively, received by a thread from one of the slave web services. The "counter" callback function takes the index of a job and a JavaScript object, generated by a thread from a response received from slave web service, as parameters.

[0060] The total number of threads is established as "10" in this example which corresponds to the number of slave web services currently processing or available to process a job at any time. Additionally, the executeBatch() function can be used in this example to perform steps 216-220. Upon completion of the executeBatch() function, control returns to the master web service context to generate the web service response, as described and illustrated earlier with reference to step 222 of FIG. 2. Accordingly, the master web service will automatically determine whether there are more responses in step 220 of FIG. 2 based on whether the executeBatch() function has completed its execution.

[0061] Referring more specifically to FIG. 5, an exemplary code fragment 500 of an exemplary slave web service configured to perform the exemplary method of FIG. 3 is illustrated. Referring more specifically to FIG. 6, an exemplary configuration file 600 defining the role of the exemplary slave web service 500 is illustrated. In this example, the slave web service 500 called catalogue-p-plp is assigned the role of "slave." The configuration file 600 can be established by an administrator of the web content management server 12 and can be stored in the memory 22, for example, as described and illustrated earlier. Additionally, a corresponding configuration file can be established, or the same configuration file can be used, to define the role of a plurality of master and/or slave web services executable by the web content management sever 12.

[0062] With this technology, web service requests, such as requests requiring a large amount of content and/or content from a number of disparate sources, for example, can be divided into a plurality of jobs which are processed in parallel by slave agile web services to reduce the overall amount of time required to generate a response to the web service request. Accordingly, portions of content required to satisfy a web service request can be retrieved in parallel and mashed-up to generate a web service response. Advantageously, the process can utilize JavaScript executed in an emulated environment on behalf of client devices, including mobile computing or other devices with special requirements that may otherwise be unable to process such content.

[0063] Having thus described the basic concept of the invention, it will be rather apparent to those skilled in the art that the foregoing detailed disclosure is intended to be presented by way of example only, and is not limiting. Various alterations, improvements, and modifications will occur and are intended to those skilled in the art, though not expressly stated herein. These alterations, improvements, and modifications are intended to be suggested hereby, and are within the spirit and scope of the invention. Additionally, the recited order of processing elements or sequences, or the use of numbers, letters, or other designations therefore, is not intended to limit the claimed processes to any order except as may be specified in the claims. Accordingly, the invention is limited only by the following claims and equivalents thereto.

CLAIMS

What is claimed is:

1. A method for servicing web service requests using parallel agile web services, the method comprising:

sending, with a web content management server, each of a plurality of jobs requiring execution in order to service a received web service request to one of a plurality of slave web services configured to execute the plurality of jobs in parallel;

receiving, with the web content management server, a response from each of the plurality of slave web services;

generating, with the web content management server, a web service response based on the received responses; and

providing, with the web content management server, the generated web service response in response to the received web service request.

2. The method of claim 1 wherein the sending further comprises sending each of the jobs and an index to one of a plurality of threads generated by a master web service executed in an emulated JavaScript environment, each of the threads configured to request execution of one of the jobs by one of the slave web services.

3. The method of claim 2 wherein each of the slave web services is further configured to retrieve content from a server device, convert the retrieved content into a JavaScript Object Notation (JSON) response, and return the JSON response.

4. The method of claim 3, wherein the generating further comprises:

converting each of the returned JSON responses into a JavaScript object;

passing each of the JavaScript objects and the index to a callback function, wherein the index corresponds to an order of a respective one of the jobs; and

organizing the JavaScript objects based on the received indices.

5. The method of claim 4, wherein the passing further comprises passing each of the JavaScript objects and the index to one of a plurality of callback functions based on the job sent to each of the threads.

6. The method of claim 1, further comprising:

determining, with the web content management server, when the received web service request requires execution of a plurality of jobs; and

performing the executing, sending, receiving, generating, and providing steps when the determining indicates the received web service request requires execution of a plurality of jobs.

7. The method of claim 6, further comprising:

determining, with the web content management server, when the plurality of jobs exceeds a maximum number of slave web services executable in parallel; and

performing, with the web content management server, when the determining indicates the plurality of jobs exceeds a maximum number of slave web services executable in parallel, steps comprising:

inserting, with the web content management server, the plurality of jobs into a queue;

sending, with the web content management server, a number of jobs corresponding to the maximum number of slave web services executable in parallel to the plurality of slave web services from the queue; and

sending, with the web content management server, one of the plurality of jobs remaining in the queue upon receiving each response from one of the plurality of slave web services and until the queue is empty.

8. A non-transitory computer readable medium having stored thereon instructions for servicing web service requests using parallel agile web services

comprising machine executable code which when executed by a processor, causes the processor to perform steps comprising:

sending each of a plurality of jobs requiring execution in order to service a received web service request to one of a plurality of slave web services configured to execute the plurality of jobs in parallel;

receiving a response from each of the plurality of slave web services;

generating a web service response based on the received responses; and

providing the generated web service response in response to the received web service request.

9. The medium of claim 8, wherein the sending further comprises sending each of the jobs and an index to one of a plurality of threads generated by a master web service executed in an emulated JavaScript environment, each of the threads configured to request execution of one of the jobs by one of the slave web services.

10. The medium of claim 9, wherein each of the slave web services is further configured to retrieve content from a server device, convert the retrieved content into a JavaScript Object Notation (JSON) response, and return the JSON response.

11. The medium of claim 10, wherein the generating further comprises:

converting each of the returned JSON responses into a JavaScript object;

passing each of the JavaScript objects and the index to a callback function, wherein the index corresponds to an order of a respective one of the jobs; and

organizing the JavaScript objects based on the received indices.

12. The medium of claim 11, wherein the passing further comprises passing each of the JavaScript objects and the index to one of a plurality of callback functions based on the job sent to each of the threads.

13. The medium of claim 8, further having stored thereon instructions comprising machine executable code which when executed by the processor, causes the processor to perform steps further comprising:

determining when the received web service request requires execution of a plurality of jobs; and

performing the executing, sending, receiving, generating, and providing steps when the determining indicates the received web service request requires execution of a plurality of jobs.

14. The medium of claim 13, further having stored thereon instructions comprising machine executable code which when executed by the processor, causes the processor to perform steps further comprising:

determining when the plurality of jobs exceeds a maximum number of slave web services executable in parallel; and

performing when the determining indicates the plurality of jobs exceeds a maximum number of slave web services executable in parallel, steps comprising:

inserting the plurality of jobs into a queue;

sending a number of jobs corresponding to the maximum number of slave web services executable in parallel to the plurality of slave web services from the queue; and

sending one of the plurality of jobs remaining in the queue upon receiving each response from one of the plurality of slave web services and until the queue is empty.

15. A web content management server device, comprising:

a processor coupled to a memory and configured to execute programmed instructions stored in the memory comprising:

sending each of a plurality of jobs requiring execution in order to service a received web service request to one of a plurality of slave web services configured to execute the plurality of jobs in parallel;

receiving a response from each of the plurality of slave web services;

generating a web service response based on the received responses; and

providing the generated web service response in response to the received web service request.

16. The device of claim 15, the sending further comprises sending each of the jobs and an index to one of a plurality of threads generated by a master web service executed in an emulated JavaScript environment, each of the threads configured to request execution of one of the jobs by one of the slave web services.

17. The device of claim 16, wherein each of the slave web services is further configured to retrieve content from a server device, convert the retrieved content into a JavaScript Object Notation (JSON) response, and return the JSON response.

18. The device of claim 17, wherein the generating further comprises:

converting each of the returned JSON responses into a JavaScript object;

passing each of the JavaScript objects and the index to a callback function, wherein the index corresponds to an order of a respective one of the jobs; and

organizing the JavaScript objects based on the received indices.

19. The device of claim 18, wherein the passing further comprises passing each of the JavaScript objects and the index to one of a plurality of callback functions based on the job sent to each of the threads.

20. The device of claim 15, wherein the processor is further configured to execute programmed instructions stored in the memory further comprising:

determining when the received web service request requires execution of a plurality of jobs; and

performing the executing, sending, receiving, generating, and providing steps when the determining indicates the received web service request requires execution of a plurality of jobs.

21. The device of claim 20, wherein the processor is further configured to execute programmed instructions stored in the memory further comprising:

determining when the plurality of jobs exceeds a maximum number of slave web services executable in parallel; and

performing when the determining indicates the plurality of jobs exceeds a maximum number of slave web services executable in parallel, steps comprising:

inserting the plurality of jobs into a queue;

sending a number of jobs corresponding to the maximum number of slave web services executable in parallel to the plurality of slave web services from the queue; and

sending one of the plurality of jobs remaining in the queue upon receiving each response from one of the plurality of slave web services and until the queue is empty.

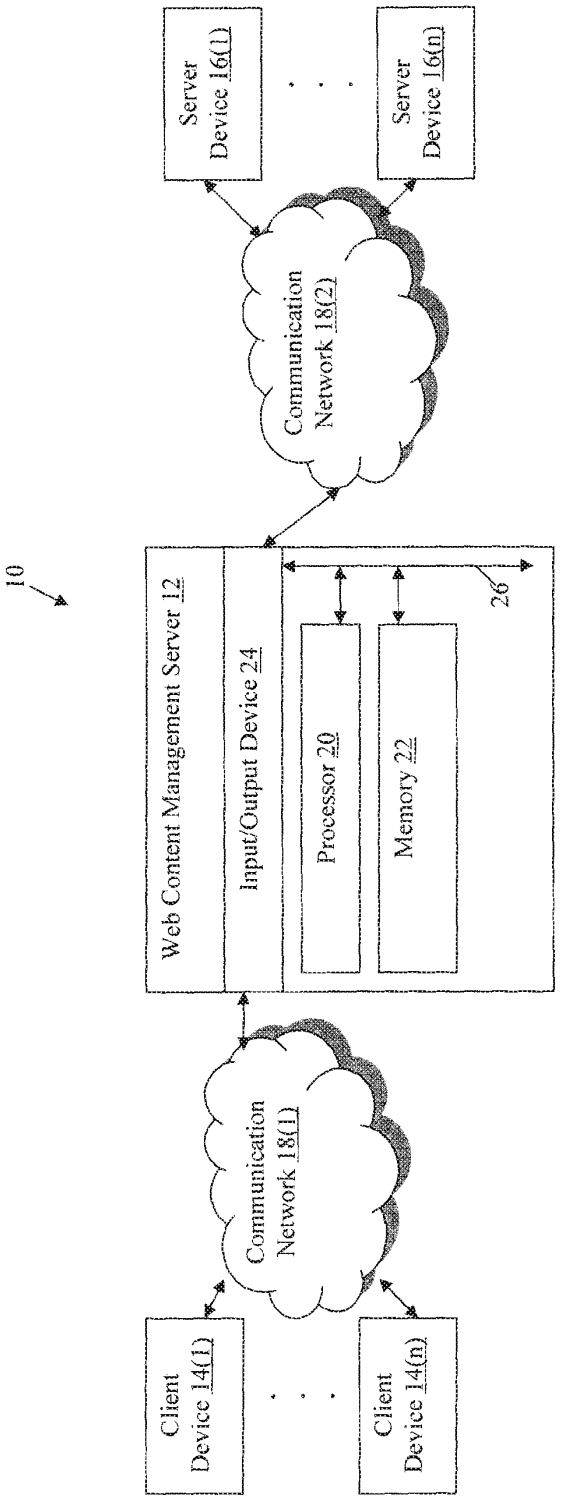

10
Web Content Management Server 12
Input/Output Device 24
Processor 20
Memory 22
26
Client Device 14(1)
Client Device 14(n)
Communication Network 18(1)
Communication Network 18(2)
Server Device 16(1)
Server Device 16(n)

FIG. 1

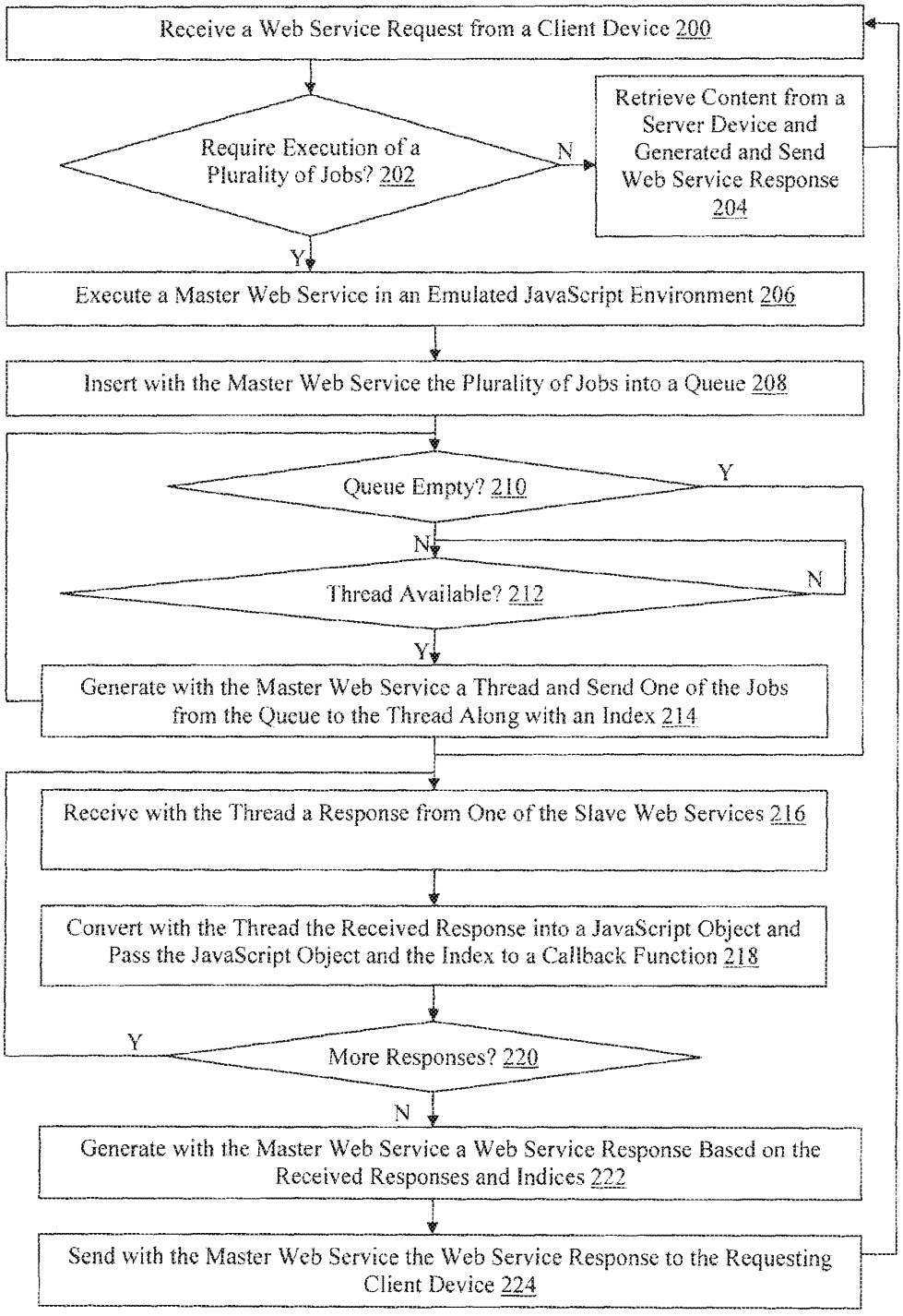

Receive a Web Service Request from a Client Device 200
Require Execution of a Plurality of Jobs? 202
N
Retrieve Content from a Server Device and Generated and Send Web Service Response 204
Y
Execute a Master Web Service in an Emulated JavaScript Environment 206
Insert with the Master Web Service the Plurality of Jobs into a Queue 208
Queue Empty? 210
Y
N
Thread Available? 212
N
Y
Generate with the Master Web Service a Thread and Send One of the Jobs from the Queue to the Thread Along with an Index 214
Receive with the Thread a Response from One of the Slave Web Services 216
Convert with the Thread the Received Response into a JavaScript Object and Pass the JavaScript Object and the Index to a Callback Function 218
Y
More Responses? 220
N
Generate with the Master Web Service a Web Service Response Based on the Received Responses and Indices 222
Send with the Master Web Service the Web Service Response to the Requesting Client Device 224

FIG. 2

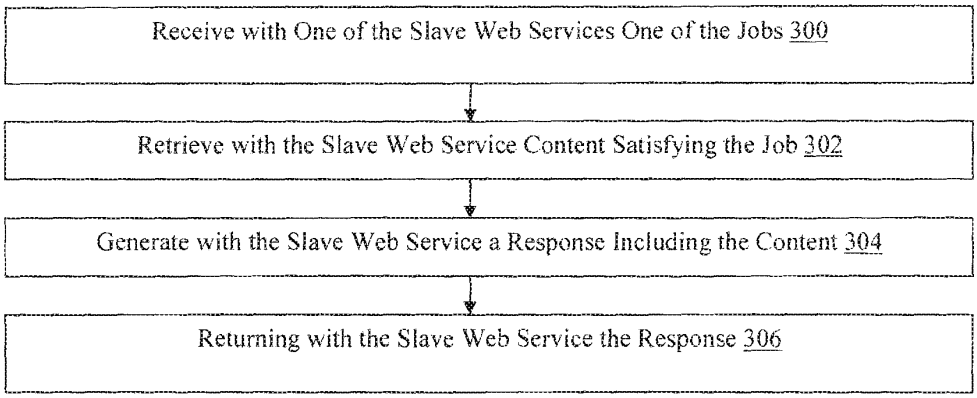


FIG. 3

```
var batch = [];
for (var i=0; i<pages; i++) {
    result.push(null);
    var o = {
        'service': 'catalogue-p-plp',                    400
        'params': {'url': pages[i]},
        'cache': '1d',
        'retryTimeouts': 5,
        'success': 'counter',
        'failure': 'generic'
    }
    batch.push(o);
}

ws.executeBatch({
    'threads': 10,
    'batch': batch,
    'success': {
        'counter': function(index, obj) {
            result[index] = obj;
        }
    },
    'failure': {
        'generic': function(index, status) {
            ws.response.error("Error on page " + index + " " + status);
        }
    }
});
```

FIG. 4

```
function service (ws) {
    jse.settings.httpTimeout = 30;
    var url = ws.request.params['url'];
    if (url) {
        ws.load (url, function (ws, xhr) {
            ws.response = JSON.stringify({
                'title': $('title')
            });
        }, function (ws, xhr) {
            ws.response.error(xhr.status, xhr.statusText);
        }) ;
    }
}
```

500

FIG. 5

```
<access_control>
    <global-rule protocols="https">
    </global-rule>
    <rule for="catalogue-p-plp" role="slave"/>
</access_control>
```

600

FIG. 6

10
Client Device 14(1)
Client Device 14(n)
Communication Network 18(1)
Web Content Management Server 12
Input/Output Device 24
Processor 20
Memory 22
26
Communication Network 18(2)
Server Device 16(1)
Server Device 16(n)

Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2861602 A1 2013/08/01

(21) 2 861 602

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(86) Date de dépôt PCT/PCT Filing Date: 2013/01/09
(87) Date publication PCT/PCT Publication Date: 2013/08/01
(85) Entrée phase nationale/National Entry: 2014/07/16
(86) N° demande PCT/PCT Application No.: US 2013/020726
(87) N° publication PCT/PCT Publication No.: 2013/112285
(30) Priorité/Priority: 2012/01/27 (US13/360,357)

(51) Cl.Int./Int.Cl. *G06F 17/21* (2006.01), *G06F 15/16* (2006.01)

(71) Demandeur/Applicant:
USABLENET INC., US

(72) Inventeur/Inventor:
SCODA, ENRICO, IT

(74) Agent: PARLEE MCLAWS LLP

(54) Titre : PROCEDES POUR TRANSFORMER DES DEMANDES DE CONTENU WEB ET DISPOSITIFS ASSOCIES
(54) Title: METHODS FOR TRANSFORMING REQUESTS FOR WEB CONTENT AND DEVICES THEREOF

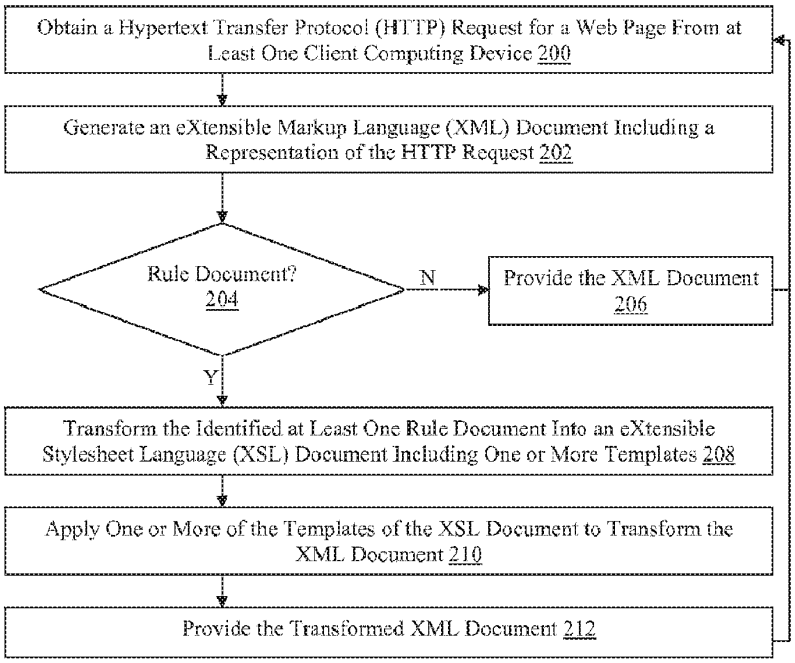


FIG. 2

(57) Abrégé/Abstract:
A method, computer readable medium and apparatus for transforming a request for web content includes obtaining at a web content optimization computing apparatus a hypertext transfer protocol (HTTP) request for a web page from at least one client

(21) 2 861 602

(13) A1

(57) **Abrégé(suite)/Abstract(continued):**
computing device. An eXtensible Markup Language (XML) document including a representation of the HTTP request is generated with the web content optimization computing apparatus. At least one rule document associated with the HTTP request is identified with the web content optimization computing apparatus. The identified at least one rule document is transformed with the web content optimization computing apparatus into an eXtensible Stylesheet Language (XSL) document including one or more templates. One or more of the templates of the XSL document are applied with the web content optimization computing apparatus to transform the XML document. The transformed XML document is provided by the web content optimization computing apparatus.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
1 August 2013 (01.08.2013)

WIPO | PCT

(10) International Publication Number
WO 2013/112285 A1

(51) International Patent Classification:
G06F 17/21 (2006.01) *G06F 15/16* (2006.01)

(21) International Application Number:
PCT/US2013/020726

(22) International Filing Date:
9 January 2013 (09.01.2013)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
13/360,357 27 January 2012 (27.01.2012) US

(71) Applicant: **USABLENET INC.** [US/US]; 28 W. 23rd St., 6th Floor, New York, NY 10010 (US).

(72) Inventor; and
(71) Applicant : **SCODA, Enrico** [IT/IT]; Via Cividina 416/3, I-33035 Martignacco (UD) (IT).

(74) Agents: **GALLO, Nicholas** et al.; LeClairRyan, A Professional Corporation, 70 Linden Oaks, Suite 210, Rochester, NY 14625 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

---- *with international search report (Art. 21(3))*

(54) Title: METHODS FOR TRANSFORMING REQUESTS FOR WEB CONTENT AND DEVICES THEREOF

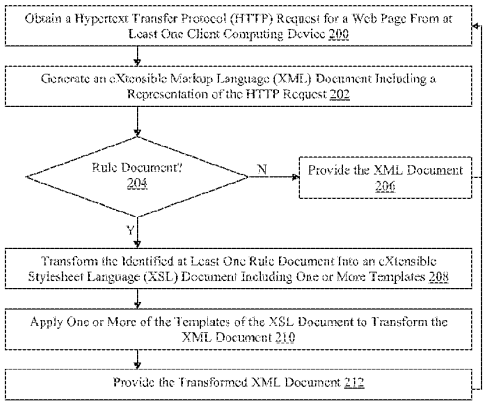


FIG. 2

(57) Abstract: A method, computer readable medium and apparatus for transforming a request for web content includes obtaining at a web content optimization computing apparatus a hypertext transfer protocol (HTTP) request for a web page from at least one client computing device. An eXtensible Markup Language (XML) document including a representation of the HTTP request is generated with the web content optimization computing apparatus. At least one rule document associated with the HTTP request is identified with the web content optimization computing apparatus. The identified at least one rule document is transformed with the web content optimization computing apparatus into an eXtensible Stylesheet Language (XSL) document including one or more templates. One or more of the templates of the XSL document are applied with the web content optimization computing apparatus to transform the XML document. The transformed XML document is provided by the web content optimization computing apparatus.

WO 2013/112285 A1

METHODS FOR TRANSFORMING REQUESTS FOR WEB CONTENT AND DEVICES THEREOF

[0001] This application claims the benefit of U.S. Patent Application Serial No. 13/360,357, filed January 27, 2012, which is hereby incorporated by reference in its entirety.

FIELD

[0002] This technology generally relates to methods and devices for transforming requests for web content and, more particularly, for extending the extensible stylesheet language (XSL) in order to manage hypertext transfer protocol (HTTP) requests for web pages.

BACKGROUND

[0003] The introduction of extensible markup language (XML) and the extensible stylesheet language (XSL) specifications has provided an easy way to transform documents between various formats. This functionality has been included in web development frameworks, giving them the ability to automatically transform an XML document into a document with a different format, such as hypertext markup language (HTML) or extensible hypertext markup language (XHTML), integrating the original data with graphic layout and user interface components. The XSL specifications are based on special constructs called templates that match a single element or a set of similar elements and rewrite them and their content based on instructions defined in the template.

[0004] Unfortunately, there is currently no effective way to define actions to take, based on attributes of a hypertext transfer protocol (HTTP) request, or otherwise to alter or control the flow of a web application and/or transaction processing the HTTP request. Attributes of the HTTP request can include HTTP header values and/or names of query or post parameters that correspond with web page user interface functionality, such as a submit or purchase button, for example. While XSL has been utilized to transform the format of web content provided in response to an HTTP request resulting from engagement with such

user interface functionality, there is no effective method or device that is capable of manipulating an HTTP request based on the engaged functionality, on the type of request, or on any other attribute of the request, in order to communicate with a web application or otherwise affect the flow and/or processing of the request.

SUMMARY

[0005] A method for transforming a request for web content includes obtaining at a web content optimization computing apparatus a hypertext transfer protocol (HTTP) request for a web page from at least one client computing device. An extensible markup language (XML) document including a representation of the HTTP request is generated with the web content optimization computing apparatus. At least one rule document associated with the HTTP request is identified with the web content optimization computing apparatus. The identified at least one rule document is transformed with the web content optimization computing apparatus into an extensible stylesheet language (XSL) document including one or more templates. One or more of the templates of the XSL document are applied with the web content optimization computing apparatus to transform the XML document. The transformed XML document is provided by the web content optimization computing apparatus.

[0006] A computer readable medium having stored thereon instructions for transforming a request for web content comprising machine executable code which when executed by at least one processor, causes the processor to perform steps including obtaining an HTTP request for a web page from at least one client computing device. An XML document including a representation of the HTTP request is generated. At least one rule document associated with the HTTP request is identified. The identified at least one rule document is transformed into an XSL document including one or more templates. One or more of the templates of the XSL document are applied to transform the XML document. The transformed XML document is provided.

[0007] A web content optimization computing apparatus for transforming a request for web content includes one or more processors and a memory coupled

to the one or more processors which are configured to execute programmed instructions stored in the memory including obtaining an HTTP request for a web page from at least one client computing device. An XML document including a representation of the HTTP request is generated. At least one rule document associated with the HTTP request is identified. The identified at least one rule document is transformed into an XSL document including one or more templates. One or more of the templates of the XSL document are applied to transform the XML document. The transformed XML document is provided.

[0008] This technology provides a number of advantages including providing a method, a computer readable medium, and an apparatus that transforms requests for web content by utilizing XSL to manipulate an HTTP request for the content. More specifically, examples of this technology generate an XML document representing an HTTP request and apply one or more rules to the XML document, the rules being predefined and represented in one or more templates of an XSL document. With this technology, the applied rules can manipulate one or more HTTP request headers or request parameters and/or the actions responsive to the HTTP request and/or flow of the application and/or web transaction configured to process the HTTP request.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] FIG. 1 is a block diagram of an exemplary environment with a web content optimization computing apparatus configured to transform a request for web content;

[0010] FIG. 2 is a flow chart of an exemplary method transforming a request for web content;

FIG. 3 is an exemplary hypertext transfer protocol (HTTP) request;

[0011] FIG. 4 is an exemplary extensible markup language (XML) document including a representation of the HTTP request of FIG. 3;

[0012] FIG. 5 is an exemplary rule document;

[0013] FIG. 6 is an exemplary extensible stylesheet language (XSL) document including a plurality of exemplary templates and resulting from an exemplary transformation of the rule document of FIG. 5; and

[0014] FIG. 7 is an exemplary XML document transformed according to an exemplary application of the templates of the XSL document of FIG. 6.

DETAILED DESCRIPTION

[0015] An exemplary environment 10 with a web content optimization computing apparatus 12 configured to transform requests for web content is illustrated in FIG. 1, although this technology can be implemented on other types of devices, such as one of the web server devices 16(1)-16(n), or any other server computing apparatus configured to receive and process hypertext transfer protocol (HTTP) requests, by way of example only. The exemplary environment 10 includes the web content optimization computing apparatus 12, client devices 14(1)-14(n), the web server devices 16(1)-16(n), and communication networks 18(1)-18(2), although other numbers and types of systems, devices, and/or elements in other configurations and environments with other communication network topologies can be used. This technology provides a number of advantages including providing a method, computer readable medium and an apparatus that transforms HTTP requests for web content, such as a hypertext markup language (HTML) web page, for example, in order to manipulate one or more actions taken based on the HTTP request headers and/or parameters.

[0016] Referring more specifically to FIG. 1, the web content optimization computing apparatus 12 includes a central processing unit (CPU) or processor 13, a memory 15, and an interface system 17 which are coupled together by a bus 19 or other link, although other numbers and types of components, parts, devices, systems, and elements in other configurations and locations can be used. The processor 13 in the web content optimization computing apparatus 12 executes a program of stored instructions for one or more aspects of the present invention as described and illustrated by way of the embodiments herein, although the processor could execute other numbers and types of programmed instructions.

[0017] The memory 15 in the web content optimization computing apparatus 12 stores these programmed instructions for one or more aspects of the present invention as described and illustrated herein, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, DVD ROM, or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to the processor 13, can be used for the memory 15 in the web content optimization computing apparatus 12.

[0018] The interface system 17 in the web content optimization computing apparatus 12 is used to operatively couple and communicate between the web content optimization computing apparatus 12 and the client devices 14(1)-14(n) and the web server devices 16(1)-16(n) via the communication networks 18(1) and 18(2), although other types and numbers of communication networks with other types and numbers of connections and configurations can be used. By way of example only, the communication networks 18(1) and 18(2) can use TCP/IP over Ethernet and industry-standard protocols, including HTTP, HTTPS, WAP, and SOAP, although other types and numbers of communication networks, such as a direct connection, a local area network, a wide area network, modems and phone lines, e-mail, and wireless and hardwire communication technology, each having their own communications protocols, can be used.

[0019] Each of the client devices 14(1)-14(n) enables a user to request, receive, and interact with web pages from one or more web sites hosted by the web server devices 16(1)-16(n) through the web content optimization computing apparatus 12 via one or more communication networks 18(1), although one or more of the client devices 14(1)-14(n) could access content and utilize other types and numbers of applications from other sources and could provide a wide variety of other functions for the user. Although multiple client devices 14(1)-14(n) are shown, other numbers and types of user computing systems could be used. In one example, the client devices 14(1)-14(n) comprise smart phones, personal digital assistants, computers, or mobile devices with Internet access that permit a website

form page or other retrieved web content to be displayed on the client devices 14(1)-14(n).

[0020] Each of the client devices 14(1)-14(n) in this example is a computing device that includes a central processing unit (CPU) or processor 20, a memory 22, user input device 24, a display 26, and an interface system 28, which are coupled together by a bus 30 or other link, although one or more of the client devices 14(1)-14(n) can include other numbers and types of components, parts, devices, systems, and elements in other configurations. The processor 20 in each of the client devices 14(1)-14(n) executes a program of stored instructions for one or more aspects of the present invention as described and illustrated herein, although the processor could execute other numbers and types of programmed instructions.

[0021] The memory 22 in each of the client devices 14(1)-14(n) stores these programmed instructions for one or more aspects of the present invention as described and illustrated herein, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to processor 20 can be used for the memory 22 in each of the client devices 14(1)-14(n).

[0022] The user input device 24 in each of the client devices 14(1)-14(n) is used to input selections, such as requests for a particular website form page or to enter data in fields of a form page, although the user input device could be used to input other types of data and interact with other elements. The user input device can include keypads, touch screens, and/or vocal input processing systems, although other types and numbers of user input devices can be used.

[0023] The display 26 in each of the client devices 14(1)-14(n) is used to show data and information to the user, such as website page by way of example

only. The display in each of the client devices 14(1)-14(n) can be a mobile phone screen display, although other types and numbers of displays could be used depending on the particular type of client device 14(1)-14(n).

[0024] The interface system 28 in each of the client devices 14(1)-14(n) is used to operatively couple and communicate between the client devices 14(1)-14(n), the web content optimization computing apparatus 12, and the web server devices 16(1)-16(n) over the communication networks 18(1) and 18(2), although other types and numbers of communication networks with other types and numbers of connections and configurations can be used.

[0025] The web server devices 16(1)-16(n) provide web content such as one or more pages from one or more web sites for use by one or more of the client devices 14(1)-14(n) via the web content optimization computing apparatus 12, although the web server devices 16(1)-16(n) can provide other numbers and types of applications and/or content and can provide other numbers and types of functions. Although the web server devices 16(1)-16(n) are shown for ease of illustration and discussion, other numbers and types of web server systems and devices can be used.

[0026] Each of the web server devices 16(1)-16(n) include a central processing unit (CPU) or processor, a memory, and an interface system which are coupled together by a bus or other link, although each of the web server devices 16(1)-16(n) could have other numbers and types of components, parts, devices, systems, and elements in other configurations and locations. The processor in each of the web server devices 16(1)-16(n) executes a program of stored instructions one or more aspects of the present invention as described and illustrated by way of the embodiments herein, although the processor could execute other numbers and types of programmed instructions.

[0027] The memory in each of the web server devices 16(1)-16(n) stores these programmed instructions for one or more aspects of the present invention as described and illustrated by way of the embodiments described and illustrated herein, although some or all of the programmed instructions could be stored

and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, DVD ROM, or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to the processor, can be used for the memory in each of the web server devices 16(1)-16(n).

[0028] The interface system in each of the web server devices 16(1)-16(n) is used to operatively couple and communicate between the web server devices 16(1)-16(n), the web content optimization computing apparatus 12, and the client devices 14(1)-14(n) via the communication networks 18(1) and 18(2), although other types and numbers of communication networks with other types and numbers of connections and configurations can be used.

[0029] Although embodiments of the web content optimization computing apparatus 12, the client devices 14(1)-14(n), and the web server devices 16(1)-16(n), are described and illustrated herein, each of the client devices 14(1)-14(n), the web content optimization computing apparatus 12, and the web server devices 16(1)-16(n), can be implemented on any suitable computer system or computing device. It is to be understood that the devices and systems of the embodiments described herein are for exemplary purposes, as many variations of the specific hardware and software used to implement the embodiments are possible, as will be appreciated by those skilled in the relevant art(s).

[0030] Furthermore, each of the systems of the embodiments may be conveniently implemented using one or more general purpose computer systems, microprocessors, digital signal processors, and micro-controllers, programmed according to the teachings of the embodiments, as described and illustrated herein, and as will be appreciated by those ordinary skill in the art.

[0031] In addition, two or more computing systems or devices can be substituted for any one of the systems in any of the embodiments. Accordingly, principles and advantages of distributed processing, such as redundancy and replication also can be implemented, as desired, to increase the robustness and

performance of the devices and systems of the embodiments. The embodiments may also be implemented on computer system or systems that extend across any suitable network using any suitable interface mechanisms and communications technologies, including by way of example only telecommunications in any suitable form (e.g., voice and modem), wireless communications media, wireless communications networks, cellular communications networks, G3 communications networks, Public Switched Telephone Network (PSTNs), Packet Data Networks (PDNs), the Internet, intranets, and combinations thereof.

[0032] The embodiments may also be embodied as a non-transitory computer readable medium having instructions stored thereon for one or more aspects of the present invention as described and illustrated by way of the embodiments herein, as described herein, which when executed by a processor, cause the processor to carry out the steps necessary to implement the methods of the embodiments, as described and illustrated herein.

[0033] An exemplary method for transforming an HTTP request for web content with the web content optimization computing apparatus 12 will now be described with reference to FIGS. 2-7, although this technology can be executed by other types of devices, such as by one of the web server devices 16(1)-16(n) and without a web content optimization computing apparatus 12 or other proxy server, for example. Referring more specifically to FIG. 2, in step 200 the web content optimization computing apparatus 12 obtains an HTTP request for web content, such as an HTML web page, from at least one of the client devices 14(1)-14(n). The requested web content can be stored on one or more of the web server devices 16(1)-16(n), for example.

[0034] Referring to FIG. 3, an exemplary HTTP request 300 for the "http://processor.com/app/www.acme.com/sample/?a=1&b=2" uniform resource located (URL) is shown as communicated by one of the client devices 14(1)-14(n) using a web browser having a "my_browser" associated user agent identification. In this example, the "app" portion of the URL refers to the "app" application operating on the "processor.com" device, which can be the web content optimization computing apparatus 12. The "app" application is configured to

download the content referred to by the "www.acme.com/sample/?a=1&b=2" portion of the URL and optionally perform some operation on the content, based on the headers and/or query and/or post parameters of the HTTP request 300, such as transforming the content into a desired format, such as HTML or XHTML and/or a desired graphical format optimal for the browser identified by the user agent, for example. With this technology, as described in detail below, the HTTP request can be manipulated such that the "app" application downloads web content referred to by a different URL than the original URL, performs a different operation on the content than indicated by the headers and/or query and/or post parameter values of the HTTP request 300, and/or performs an action different than, or in addition to, the transformation of the content into a different format, by way of example only.

[0035] In order to manipulate the HTTP request 300, an XML document 400 including a representation of the HTTP request is generated, at step 202, by the web content optimization computing apparatus 12. The XML document 400 can include a plurality of elements, optionally nested by a request root element, the values of which are based on the contents of the HTTP request 300. In this example, the value of the URL element is the URL of the content to be downloaded, without any query parameters, the value of the currentserver element is the name of the device running the "app" application, such as the web content optimization computing apparatus 12, that will process the HTTP request 300 to at least retrieve the requested content from one of the web server devices 16, the value of the remoteip element is the IP address of the requesting client device 14(1)-14(n), the value of the headers element is a param element list, the name attribute of each element of which corresponds to each header of the HTTP request 300 such as the host, user-agent, and accept headers and the value of each element of which is included in the HTTP request 300, the value of the original-ua element is the user agent header of the HTTP request 300, and the value of the query element is a param element list, the name attribute of each element of which corresponds to each query parameter of the URL associated with the HTTP request 300 and the value of each element of which is included in the URL associated with the HTTP request 300.

[0036] While empty in the exemplary XML document 400 shown in FIG. 4, a plurality of other elements can be provided in the XML document 400, and can contain value(s) based on the HTTP request 300, including a post element having a value of a param element list, the name attribute of each element of which corresponds to each post parameter of the HTTP request 300 and the value of each element of which is set forth in the HTTP request 300, a cookies element having a value of a param element list, the name attribute of each element of which corresponds to a cookie name as included in the HTTP request 300 and the value of each element of which corresponds to a cookie value as included in the HTTP request 300, and an imode element having a string value indicating whether the requesting client device 14(1)-14(n) supports i-mode services.

[0037] In other examples, the XML document 400 can contain a plurality of other elements representing actions that can be performed by the "app" application including a redirect element having a value of a URL of a location to which the HTTP request 300 is to be redirected, an auto_redirect element having a value of a string indicating the "app" application should implement an automatic redirect algorithm, such as that described in U.S. Patent Application Nos. 12/927,169 and 13/135,707, each of which is hereby incorporated by reference in its entirety, an encoding element having a string value indicating the character set to be used to read the web content requested by the HTTP request 300, a content-type element having a string value indicating the content or mime type to be used to read the web content requested by the HTTP request 300, a popup element having a value to be sent to the requesting client device 14(1)-14(n) to be displayed on the requesting client device 14(1)-14(n) instead of the requested web content, an error element having a string value including an error message to be displayed on the requesting client device 14(1)-14(n) when one or more attributes and/or values of the attributes of the HTTP request 300 are invalid, for example, and a ua element having a string value indicating a user agent to be used by the "app" application, prior to retrieving the requested web content from the web server device 16(1)-16(n), instead of the user agent indicated in the HTTP request 300.

[0038] In step 204, the web content optimization computing apparatus 12 determines whether a rule document exists for the HTTP request 300. The web content optimization computing apparatus 12 can identify at least one rule document 500 associated with the HTTP request 300 based on a match of at least a portion of the URL included in the HTTP request 300 or the value of any of the headers included in the HTTP request 300, for example. Accordingly, a plurality of rule documents can be stored in the memory 15 of the web content optimization computing apparatus 12 as associated, such as in a table, with one or more attributes of an HTTP request. In one example, a rule document is provided for a plurality of URLs and the web content optimization computing apparatus 12 is configured to identify the rule document applicable to the current HTTP request based on a match in the table of the URL included in the current HTTP request. In another example, a rule document is stored for one or more user agents and the web content optimization computing apparatus 12 is configured to identify the rule document applicable to the current HTTP request based on a match in the table of the value of the user agent indicated in the HTTP request or in the ua element of the XML document representing the HTTP request, for example. If no rule document is identified at step 204, the web content optimization computing apparatus 12 provides the XML document 400 at step 206, such that the requested web content can be retrieved from the web server device 16(1)-16(n) and communicated to the requesting client device 14(1)-14(n).

[0039] If a rule document 500 is identified for the HTTP request at step 204, the web content optimization computing apparatus 12 transforms, at step 208, the identified rule document 500 into an eXtensible Stylesheet Language (XSL) document 600 including one or more templates. An exemplary rule document 500 is shown in FIG. 5 as including two rule elements, each including a "for" condition satisfied based upon a match of a query or post parameter name, for example, and each also including one or more commands nested by an execute element.

[0040] In this example, the rule elements of the rule document 500 are established based on a match of a name of the query parameters included in a URL associated with the HTTP request 300, although in other examples one or

more rules can be applied based on a match of the name of a post parameter, or based on any other attribute of the HTTP request 300. In one example, the name of one of a query or a post parameter included in a URL associated with the HTTP request 300 corresponds to the name of user interface functionality that, when engaged by a user of one of the client computing devices 14(1)-14(n), results in generating the HTTP request 300 that is then communicated to the web content optimization computing apparatus 12.

[0041] In the example shown in FIG. 5, the parameter names included in the conditional expression are "a" and "c" and the commands in the first rule element are set-encoding and set-query-param and the command in the second rule is set-query-param. While the "c" query parameter is not included in the HTTP request 300, the "a" parameter name can indicate the name of user interface functionality such as a button that, when engaged by a user of one of the client computing devices 14(1)-14(n), causes the communication of the HTTP request to be sent to the web content optimization computing apparatus 12. In other examples, the name of the query parameter and associated user interface functionality can be "submit" or "purchase", for example, or any other string value corresponding to a button, image, link, or any other web functionality initiating an HTTP request 300. Accordingly, a rule element, such as the first rule element in the exemplary rule document 500 of FIG. 5, can be included and can correspond to the name of the user interface functionality. The set-encoding command of the first rule element has a string value indicating the character set to be applied to the requested web content by the "app" application. The set-query-param command has a name attribute of a query parameter of the HTTP request 300 to be replaced by the value of the string indicated in the element.

[0042] Accordingly, in this example, whenever a user engages the "a" button on a web page, an HTTP request 300 for content is initiated whereby the content is stored on a web server device 16(1)-16(n) located in Japan, for example, and requires Japanese encoding for proper manipulation and/or display. Therefore, a rule element is recited in a rule document 500 conditional upon a match of the "a" query parameter name included in the HTTP request 300 and

including a set-encoding command with a "Shift_JIS" value indicating a Japanese character set.

[0043] In other examples, commands are set forth in one or more rule elements of the rule document 500 to manipulate one or more parameters of the HTTP request 300, including a remove-post-param command configured to remove a specified post parameter from the HTTP request 300, a remove-query param command configured to remove a specified query parameter from the HTTP request 300, a set-all-post-params command configured to replace all post parameters of the HTTP request 300 with a specified post parameter value, a set-all-query-params command configured to replace all query parameters of the HTTP request 300 with a specified query parameter value, and a set-post-param configured to set a specified post parameter of the HTTP request 300 with a specified value.

[0044] In other examples, commands are set forth in one or more rule elements of the rule document 500 to change one or more parameters of the HTTP request 300 and/or one or more parameters of an HTTP response, including a set-content-type command configured to set the content type of the requested web content including in the HTTP request 300 or an HTTP response, a set-cookie command configured to set a new cookie in the HTTP request 300 or an HTTP response, a set-header command configured to add a header to the HTTP request 300 or an HTTP response, and a set-user-agent command configured to change or set the user agent included in the HTTP request 300 or an HTTP response.

[0045] In yet other examples, commands are set forth in one or more rule elements of the rule document 500 to change the flow of a web transaction and/or an action performed by the identified "app" application including a set-auto-redirect command configured to enable an automatic redirect algorithm, as identified above, a set-error command configured to set an error message to be displayed on the requesting client device 14(1)-14(n), a set-popup command configured to display a virtual page on the requesting client device 14(1)-14(n) instead of the web content requested by the HTTP request 300, a set-redirect command configured to generate an HTTP response status code 302 to be

communicated to the requesting client device 14(1)-14(n), and a set-url command configured to change the URL of the requested web content included in the HTTP request 300.

[0046] Accordingly, in this example, the web content optimization computing apparatus 12 identifies the rule document 500, at step 204, of FIG. 5, based on an attribute of the HTTP request 300, as represented by the XML document 400, generated at step 202, and, in order to apply the rule(s) to the HTTP request 300, transforms the rule document 500, at step 208, into an XSL document 600 including one or more templates, an example of which is shown in FIG. 6. The XSL document 600 includes at least one template corresponding to each rule element included in the rule document 500 and, optionally, each command included in each rule element.

[0047] In the exemplary rule document 500 of FIG. 5, a first rule element is associated with a query or post parameter value of "a" and a second rule is associated with a query or post parameter value of "c". The first rule includes a first command to set the encoding of the character set of the requested web content to "Shift_JIS", a Japanese encoding, and a second command to set the value of the query parameter having a value of "b" in the initial HTTP request to the value of "JP." The second rule includes a command to set the value of the query parameter having a value of "e" in the HTTP request 300 to the value of "TRANSLATE". While the HTTP request 300 does not include a query parameter having a value of "e", as shown in FIG. 3, in one example all rule elements are included as part of the transformation of the rule document 500 into the XSL document 600, and have a corresponding template in the XSL document 600, irrespective of whether the rule element(s) are applicable to the current HTTP request 300.

[0048] In step 210, the web content optimization computing apparatus 12 applies one or more of the templates of the XSL document 600 in order to transform the XML document 400 into the transformed XML document 700, an example of which is shown in FIG. 7. The exemplary XSL document 600, resulting from a transformation of the rule document 500, includes a first template

with a match expression for the "a" query parameter and configured to set a value for the encoding element to "Shift_JIS" when applied to the XML document 400, as shown in the resulting transformed XML document 700. The XSL document 600 further includes a second template with a match expression for the "a" query parameter and configured to change the value of the "b" query parameter from "2" to "JP" when applied to the XML document 400, as shown in the resulting transformed XML document 700. The XSL document 600 further includes a third template with a match expression for a "c" query parameter which is not satisfied by the current HTTP request 300. The rule elements included in the rule document 500 are exemplary only and the rule elements can include any number and type of commands as identified above, or any other command. Additionally, the associated templates of the XSL document 600, resulting from a transformation of the rule elements of the rule document 500, can change and/or set values for any element of the HTTP request 300.

[0049] In step 212, the web content optimization computing apparatus 12 provides the transformed XML document 700, such as to the "app" application identified by the URL associated with the HTTP request 300 and configured to process and service the HTTP request 300 according to the transformed XML document 700, including by retrieving the requested web content from the web server device 16(1)-16(n) and communicating it to the requesting client computing device 14(1)-14(n).

[0050] Accordingly, in this example, the "app" application interprets the HTTP request represented by the transformed XML document 700 to retrieve the requested content as identified by the "http://www.acme.com/sample" value of the url element and encodes the requested content with the Shift_JIS character set as indicated by the value of the encoding element, thereby overriding any other encoding that may have been applied by the "app" application. Accordingly, in this example, the HTTP request 300 is manipulated, based on predefined rules, to ensure the proper display of the requested web content on the client device 14(1)-14(n).

[0051] Accordingly, as illustrated and described herein this technology provides a number of advantages including providing a method, a computer readable medium, and an apparatus that transforms requests for web content by utilizing XSL to manipulate an HTTP request for the content. More specifically, examples of this technology identify a rule document based on one or more attributes of an HTTP request, transform the rule document into an XSL document, and apply the XSL document to an XML representation of the HTTP request. With this technology, one or more HTTP request headers, one or more HTTP request parameters and/or parameter values, and/or one or more actions performed in response to an HTTP request can be manipulated based on one or more predefined rules, thereby enabling control over the flow of a web application and/or transaction.

[0052] Having thus described the basic concept of the invention, it will be rather apparent to those skilled in the art that the foregoing detailed disclosure is intended to be presented by way of example only, and is not limiting. Various alterations, improvements, and modifications will occur and are intended to those skilled in the art, though not expressly stated herein. These alterations, improvements, and modifications are intended to be suggested hereby, and are within the spirit and scope of the invention. Additionally, the recited order of processing elements or sequences, or the use of numbers, letters, or other designations therefore, is not intended to limit the claimed processes to any order except as may be specified in the claims. Accordingly, the invention is limited only by the following claims and equivalents thereto.

CLAIMS

What is claimed is:

1. A method for transforming a request for web content, comprising:

obtaining at a web content optimization computing apparatus a hypertext transfer protocol (HTTP) request for a web page from at least one client computing device;

generating with the web content optimization computing apparatus an eXtensible Markup Language (XML) document including a representation of the HTTP request;

identifying with the web content optimization computing apparatus at least one rule document associated with the HTTP request;

transforming with the web content optimization computing apparatus the identified at least one rule document into an eXtensible Stylesheet Language (XSL) document including one or more templates;

applying with the web content optimization computing apparatus one or more of the templates of the XSL document to transform the XML document; and

providing by the web content optimization computing apparatus the transformed XML document.

2. The method as set forth in claim 1 wherein the providing further comprises providing with the web content optimization computing apparatus the XML document when there is not at least one rule document associated with the HTTP request.

3. The method as set forth in claim 1 wherein the XML document includes a plurality of elements selected from uniform resource locator (URL), redirect, auto_redirect, encoding, content-type, popup, error, currentserver, remoteip, headers, original-ua, ua, query, post, cookies, or imode.

4. The method as set forth in claim 1 wherein the at least one rule document includes at least one rule and wherein each rule includes at least one command selected from remove-post-param, remove-query-param, set-all-post-params, set-all-query-params, set-auto-redirect, set-content-type, set-cookie, set-encoding, set-error, set-header, set-popup, set-post-param, set-query-param, set-redirect, set-url, or set-user-agent.

5. The method as set forth in claim 1 wherein the rule document includes at least one rule including a conditional statement satisfied based on a match of a name of one of a query parameter or a post parameter included in a URL associated with the HTTP request wherein the name corresponds to a name of user interface functionality configured to initiate an HTTP request when engaged.

6. The method as set forth in claim 1 wherein the providing further comprises:

communicating with the web content optimization computing apparatus the transformed XML document to an application identified by a URL associated with the HTTP request; and

servicing with the web content optimization computing apparatus the HTTP request based at least in part on the transformed XML document.

7. A non-transitory computer readable medium having stored thereon instructions for transforming requests for web content comprising machine executable code which when executed by at least one processor, causes the processor to perform steps comprising:

obtaining a hypertext transfer protocol (HTTP) request for a web page from at least one client computing device;

generating an eXtensible Markup Language (XML) document including a representation of the HTTP request;

identifying at least one rule document associated with the HTTP request;

transforming the identified at least one rule document into an eXtensible Stylesheet Language (XSL) document including one or more templates;

applying one or more of the templates of the XSL document to transform the XML document; and

providing the transformed XML document.

8. The medium as set forth in claim 7 wherein the providing further comprises providing the XML document when there is not at least one rule document associated with the HTTP request.

9. The medium as set forth in claim 7 wherein the XML document includes a plurality of elements selected from uniform resource locator (URL), redirect, auto_redirect, encoding, content-type, popup, error, currentserver, remoteip, headers, original-ua, ua, query, post, cookies, or imode.

10. The medium as set forth in claim 7 wherein the at least one rule document includes at least one rule and wherein each rule includes at least one command selected from remove-post-param, remove-query-param, set-all-post-params, set-all-query-params, set-auto-redirect, set-content-type, set-cookie, set-encoding, set-error, set-header, set-popup, set-post-param, set-query-param, set-redirect, set-url, or set-user-agent.

11. The medium as set forth in claim 7 wherein the rule document includes at least one rule including a conditional statement satisfied based on a match of a name of one of a query parameter or a post parameter included in a URL associated with the HTTP request wherein the name corresponds to a name of user interface functionality configured to initiate an HTTP request when engaged.

12. The medium as set forth in claim 7 wherein the providing further comprises:

communicating the transformed XML document to an application identified by a URL associated with the HTTP request; and

servicing the HTTP request based at least in part on the transformed XML document.

13. A web content optimization computing apparatus for transforming a request for web content, comprising:

one or more processors;

a memory coupled to the one or more processors which are configured to execute programmed instructions stored in the memory comprising:

obtaining a hypertext transfer protocol (HTTP) request for a web page from at least one client computing device;

generating an eXtensible Markup Language (XML) document including a representation of the HTTP request;

identifying at least one rule document associated with the HTTP request;

transforming the identified at least one rule document into an eXtensible Stylesheet Language (XSL) document including one or more templates;

applying one or more of the templates of the XSL document to transform the XML document; and

providing the transformed XML document.

14. The apparatus as set forth in claim 13 wherein the providing further comprises providing the XML document when there is not at least one rule document associated with the HTTP request.

15. The apparatus as set forth in claim 13 wherein the XML document includes a plurality of elements selected from uniform resource locator (URL), redirect, auto_redirect, encoding, content-type, popup, error, currentserver, remoteip, headers, original-ua, ua, query, post, cookies, or imode.

16. The apparatus as set forth in claim 13 wherein the at least one rule document includes at least one rule and wherein each rule includes at least one command selected from remove-post-param, remove-query-param, set-all-post-params, set-all-query-params, set-auto-redirect, set-content-type, set-cookie, set-encoding, set-error, set-header, set-popup, set-post-param, set-query-param, set-redirect, set-url, or set-user-agent.

17. The apparatus as set forth in claim 13 wherein the rule document includes at least one rule including a conditional statement satisfied based on a match of a name of one of a query parameter or a post parameter included in a URL associated with the HTTP request wherein the name corresponds to a name of user interface functionality configured to initiate an HTTP request when engaged.

18. The apparatus as set forth in claim 13 wherein the providing further comprises:

communicating the transformed XML document to an application identified by a URL associated with the HTTP request; and

servicing the HTTP request based at least in part on the transformed XML document.

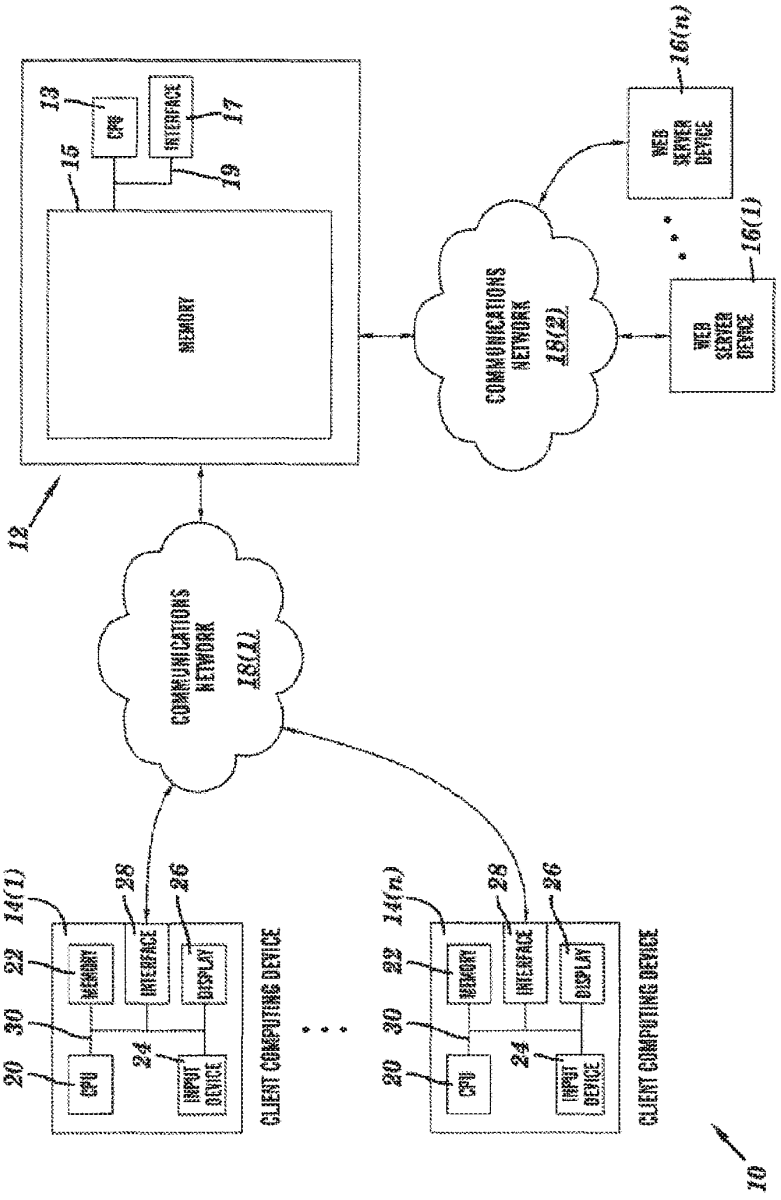

12
13
CPU
INTERFACE
17
15
19
MEMORY
COMMUNICATIONS NETWORK 18(2)
COMMUNICATIONS NETWORK 18(1)
16(n)
WEB SERVER DEVICE
16(1)
WEB SERVER DEVICE
14(1)
28
26
22
MEMORY
INTERFACE
DISPLAY
30
20
CPU
24
INPUT DEVICE
CLIENT COMPUTING DEVICE
14(n)
28
26
22
MEMORY
INTERFACE
DISPLAY
30
20
CPU
24
INPUT DEVICE
CLIENT COMPUTING DEVICE
10

FIG. 1

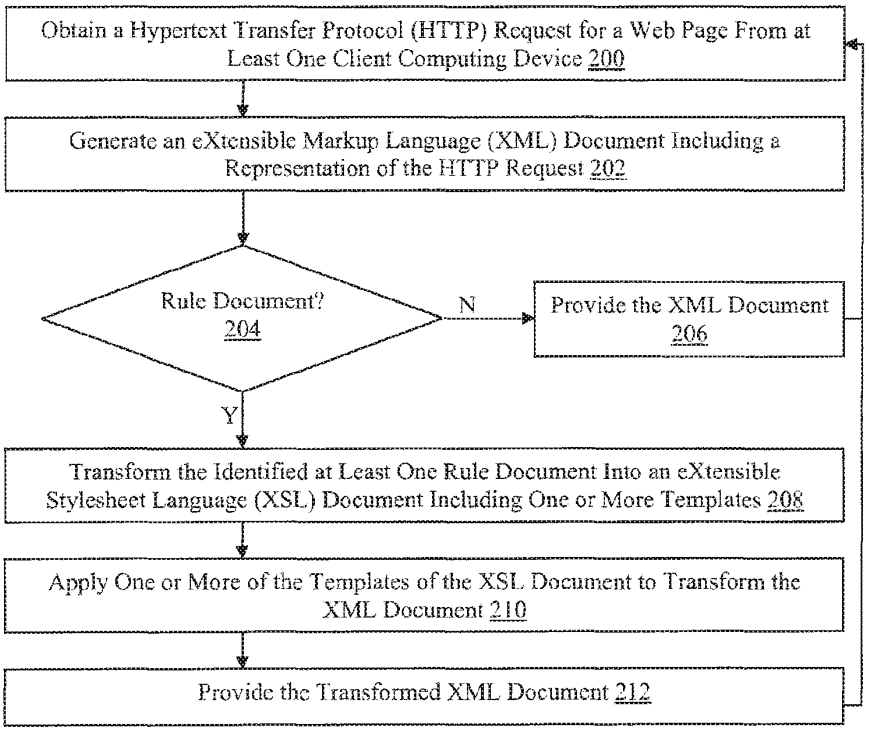

Obtain a Hypertext Transfer Protocol (HTTP) Request for a Web Page From at Least One Client Computing Device 200
Generate an eXtensible Markup Language (XML) Document Including a Representation of the HTTP Request 202
Rule Document? 204
N
Provide the XML Document 206
Y
Transform the Identified at Least One Rule Document Into an eXtensible Stylesheet Language (XSL) Document Including One or More Templates 208
Apply One or More of the Templates of the XSL Document to Transform the XML Document 210
Provide the Transformed XML Document 212

FIG. 2


* About to connect() to processor.com port 80 (#0)
*   Trying 63.70.164.32... connected
* Connected to processor.com (63.70.164.32) port 80 (#0)
> GET /app/www.acme.com/sample/?a=1&b=2 HTTP/1.1
> User-Agent: my_browser
> Host: processor.com
> Accept: */*
>
300

FIG. 3

```
<?xml version="1.0" encoding="utf-8"?>
<request>
    <url>http://www.acme.com/sample/</url>
    <redirect/>
    <auto_redirect/>
    <encoding/>
    <content-type/>
    <popup/>
    <error/>
    <currentserver>processor.com</currentserver>
    <remoteip>192.168.10.x</remoteip>
    <headers>
        <param name="host">processor.com</param>
        <param name="user-agent">my_browser</param>
        <param name="accept">*/*</param>
    </headers>
    <original-ua>my_browser</original-ua>
    <ua>my_browser</ua>
    <query>
        <param name="a">1</param>
        <param name="b">2</param>
    </query>
    <post></post>
    <cookies></cookies>
    <imode/>
</request>
```

400

FIG. 4

```
<request-rules>                                          500
    <rule for="a">
        <execute>
            <set-encoding>Shift_JIS</set-encoding>
            <set-query-param name="b">JP</set-query-param>
        </execute>
    </rule>

    <rule for="c">
        <execute>
            <set-query-param name="e">TRANSLATE</set-query-param>
        </execute>
    </rule>

</request-rules>
```

FIG. 5

600

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:request="http://request-rules.org" version="1.0">

    <xsl:template match="encoding[request:query-param('a')!=''
                        or request:post-param('a')!='']">
        <xsl:copy>
            Shift_JIS
        </xsl:copy>
    </xsl:template>

    <xsl:template match="query[request:query-param('a')!=''
                               or request:post-param('a')!='']">
        <xsl:copy>
            <xsl:copy-of select="param[@name!='b']"/>
            <param name="b">JP</param>
        </xsl:copy>
    </xsl:template>

    <xsl:template match="query[request:query-param('c')!=''
                        or request:post-param('c')!='']">
        <xsl:copy>
            <xsl:apply-templates/>
            <param name="e">TRANSLATE</param>
        </xsl:copy>
    </xsl:template>

    <xsl:template match="*">
        <xsl:copy>
            <xsl:copy-of select="@*"/>
            <xsl:apply-templates/>
        </xsl:copy>
    </xsl:template>

</xsl:stylesheet>
```

FIG. 6

```
<?xml version="1.0" encoding="utf-8"?>
<request>
    <url>http://www.acme.com/sample/</url>
    <redirect/>
    <auto_redirect/>
    <encoding>Shift_JIS</encoding>
    <content-type/>
    <popup/>
    <error/>
    <currentserver>processor.com</currentserver>
    <remoteip>192.168.10.x</remoteip>
    <headers>
        <param name="host">processor.com</param>
        <param name="user-agent">my_browser</param>
        <param name="accept">*/*</param>
    </headers>
    <original-ua>my_browser</original-ua>
    <ua>my_browser</ua>
    <query>
        <param name="a">1</param>
        <param name="b">JP</param>
    </query>
    <post></post>
    <cookies></cookies>
    <imode/>
</request>
```

700

FIG. 7

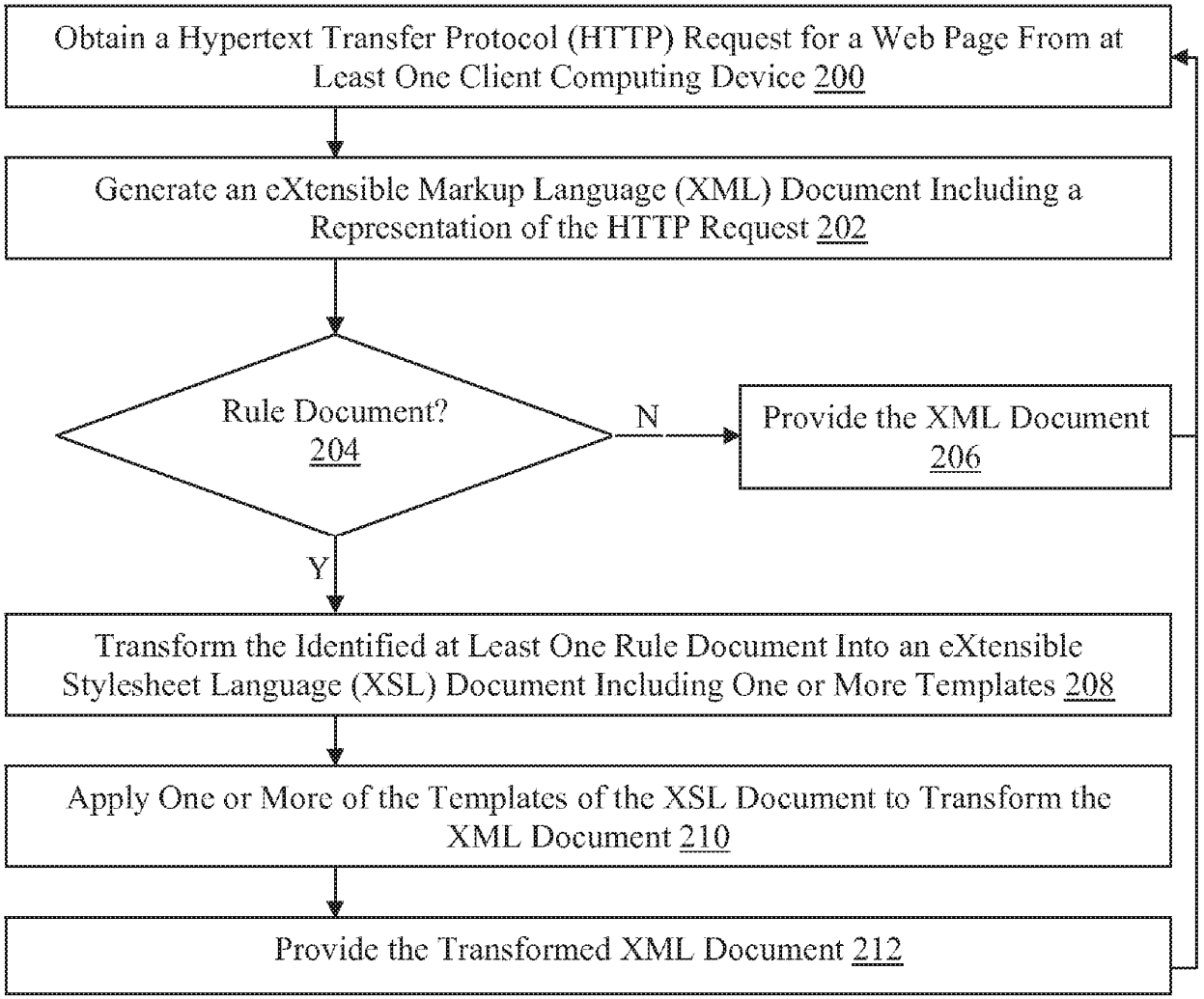


FIG. 2

Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2868162 A1 2015/05/04

(21) 2 868 162

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(22) Date de dépôt/Filing Date: 2014/10/23

(41) Mise à la disp. pub./Open to Public Insp.: 2015/05/04

(30) Priorité/Priority: 2013/11/04 (US14/070,764)

(51) Cl.Int./Int.Cl. *G06F 9/44* (2006.01), *G06F 9/45* (2006.01), *H04L 12/16* (2006.01)

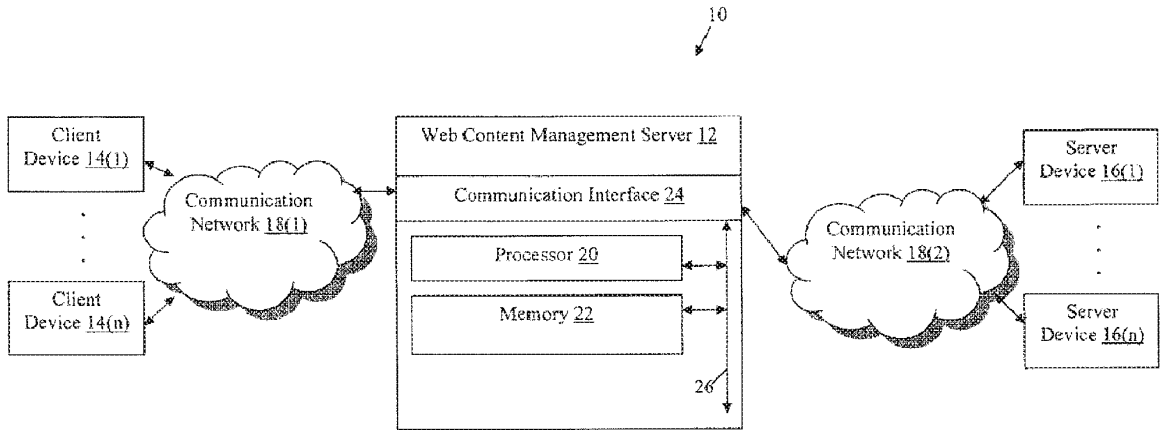
(71) Demandeur/Applicant:
USABLENET INC., US

(72) Inventeur/Inventor:
SCODA, ENRICO, IT

(74) Agent: PARLEE MCLAWS LLP

(54) Titre : PROCEDES POUR ETENDRE UNE INTERFACE DE PROGRAMMATION D'APPLICATION DE SELECTEUR ET DISPOSITIFS DE CEUX-CI

(54) Title: METHODS FOR EXTENDING A SELECTOR APPLICATION PROGRAMMING INTERFACE AND DEVICES THEREOF



(57) Abrégé/Abstract:
A method, non-transitory computer readable medium, and web content management server device that receives a cascading style sheet (CSS) selector expression comprising a plurality of expression components. Whether an equivalent XML path language (XPath) expression can be generated for each of the expression components is determined. Each of the expression components for which an equivalent XPath expression cannot be generated is transformed into a transformed expression comprising at least one XPath expression and an extension function.

ABSTRACT

A method, non-transitory computer readable medium, and web content management server device that receives a cascading style sheet (CSS) selector expression comprising a plurality of expression components. Whether an equivalent XML path language (XPath) expression can be generated for each of the expression components is determined. Each of the expression components for which an equivalent XPath expression cannot be generated is transformed into a transformed expression comprising at least one XPath expression and an extension function.

METHODS FOR EXTENDING A SELECTOR APPLICATION PROGRAMMING INTERFACE AND DEVICES THEREOF

FIELD

[0001] This technology generally relates to methods and devices for optimizing delivery of web content and, more particularly, to methods for extending a selectors application programming interface (API) and devices thereof.

BACKGROUND

[0002] A web content management server is a server that optimizes web pages and web page interactions for client devices with special requirements, such as smartphones. By way of example, a client device can send an HTTP request for a web page which is retrieved from a server device by the web content management server. Next, the web content management server can optimize the content of the web page by applying transformation rules tailored to the requesting client device. This optimization process can include extracting content relevant to the requesting client device and adapting the extracted content to fit the specifications of the requesting client device. In order to adapt the content, the web content management server can perform transformations including JavaScript removal, content linearization, and small screen adaptation, for example.

[0003] Although this process works well to optimize content for display at the requesting client device, the web page may not function properly when the original content heavily depends on JavaScript technology that cannot be implemented on some client devices. For example, if the requested web page includes JavaScript code responsible for populating form fields, validating form submissions, retrieving data from external resources (e.g., based on AJAX technology), and even generating components that may change the structure of the web page, the page at the client device will not be able to properly function.

[0004] Accordingly, JavaScript included in web pages can be removed, stored by the web content management server, and replaced by a hidden field with an identifier, for example. The identifier is subsequently sent by the client device in an HTTP request in

response to a web page interaction requiring execution of the JavaScript, such as a request for validation of login credentials. Based on the identifier, the web content management server can retrieve the JavaScript, execute the JavaScript in an emulated JavaScript environment on behalf of the client device, and return a result to the client device.

[0005] In some implementations, the emulated JavaScript environment can support application programming interfaces (APIs) such as those defined by hypertext markup language (HTML), events, cascading style sheet (CSS), range, traversal, and views models, for example. Some of the supported APIs can allow web page developers to embed JavaScript defining functionality for accessing a document object model (DOM) associated with a web page in order to perform an operation. For example, the selector API provides CSS selector expressions or queries configured to be applied to a DOM to obtain a result set of elements of the DOM.

[0006] However, CSS selector expressions can be difficult to implement, can be complex, and can require specific code or libraries to be implemented. While some CSS selector expressions can be translated into XML path language (XPath) expressions or queries, which are relatively efficient to execute, many CSS selector expressions are not translatable.

SUMMARY

[0007] A method for extending a selector application programming interface (API) includes receiving, with a web content management server, a cascading style sheet (CSS) selector expression comprising a plurality of expression components. Whether an equivalent XML path language (XPath) expression can be generated for each of the expression components is determined with the web content management server. Each of the expression components for which an equivalent XPath expression cannot be generated is transformed with the web content management server into a transformed expression comprising at least one XPath expression and an extension function.

[0008] A non-transitory computer readable medium having stored thereon instructions for extending the selector API comprising machine executable code which when executed by a processor, causes the processor to perform steps including receiving a cascading style sheet (CSS) selector expression comprising a plurality of expression components. Whether an equivalent XML path language (XPath) expression can be generated for each of the expression components is determined. Each of the expression components for which an equivalent XPath expression cannot be generated is transformed into a transformed expression comprising at least one XPath expression and an extension function.

[0009] A web content management server device includes a processor coupled to a memory and configured to execute programmed instructions stored in the memory including receiving a cascading style sheet (CSS) selector expression comprising a plurality of expression components. Whether an equivalent XML path language (XPath) expression can be generated for each of the expression components is determined. Each of the expression components for which an equivalent XPath expression cannot be generated is transformed into a transformed expression comprising at least one XPath expression and an extension function.

[0010] This technology provides a number of advantages including methods, non-transitory computer readable medium, and devices that facilitate more efficient processing of CSS selector expressions using XPath. With this technology, CSS selector expressions are fully translatable into XPath expressions which can be evaluated in an emulated JavaScript environment by a relatively efficient XPath processor without requiring the specific code otherwise necessary to execute the CSS selector expressions.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] FIG. 1 is a block diagram of a network environment with an exemplary web content management server;

[0012] FIG. 2 is a flow chart of an exemplary method for extending the selector API to an emulated JavaScript environment;

[0013] FIG. 3 is a flow chart of an exemplary method performed by an extension function for evaluating the :first-of-type, :last-of-type, :nth-of-type, and :nth-last-of-type pseudo classes;

[0014] FIG. 4 is a flow chart of an exemplary method performed by an extension function for evaluating the :last-of-type, :only-of-type, and :nth-last-of-type pseudo classes;

[0015] FIG. 5 is a flow chart of an exemplary method performed by an extension function for evaluating the :lang pseudo class;

[0016] FIG. 6 is a flow chart of an exemplary method performed by an extension function for evaluating the :not pseudo class; and

[0017] FIG. 7 is a flow chart of an exemplary method performed by an extension function for evaluating the :selected jQuery pseudo class.

DETAILED DESCRIPTION

[0018] An exemplary network environment 10 with a web content management server 12 coupled to client devices 14(1)-14(n) and server devices 16(1)-16(n) is illustrated in FIG. 1. In this example, the web content management server 12, client devices 14(1)-14(n), and server devices 16(1)-16(n) are coupled together by communication networks 18(1)-18(2), although other numbers and types of systems, devices, and/or elements in other configurations or network topologies can also be used. This technology provides a number of advantages including methods, non-transitory computer readable medium, and devices that facilitate more efficient processing of CSS selector expressions in an emulated JavaScript environment.

[0019] The web content management server 12 is coupled to the client devices 14(1)-14(n) by the communication network 18(1), which can include one or more local area network(s) (LANs) and/or wide area network(s) (WANs). In this example, the web content

management server 12 is further coupled to the server devices 16(1)-16(n) by the communication network 18(2), which may also include one or more LANs and/or WANs. Other network devices configured to generate, send, and receive network communications and coupled together via other topologies can also be used. While not shown, the network environment 10 also may include additional network components, such as routers, switches and other devices, which are well known to those of ordinary skill in the art and thus will not be described here.

[0020] The web content management server 12 may perform any number of functions including optimizing content retrieved from the server devices 16(1)-16(n) for display on the client devices 14(1)-14(n), for example. In this example the web content management server 12 includes a processor 20, a memory 22, and a communication interface 24, which are coupled together by a bus 26 or other link, although other numbers and types of components, parts, devices, systems, and elements in other configurations and locations can be used.

[0021] The processor 20 in the web content management server 12 executes a program of stored instructions one or more aspects of the present invention, as described and illustrated by way of the embodiments herein, although the processor 20 could execute other numbers and types of programmed instructions. The processor 20 of the web content management server 12 may comprise one or more central processing units or general purpose processors with one or more processing cores, for example.

[0022] The memory 22 in the web content management server 12 stores these programmed instructions for one or more aspects of the present invention, as described and illustrated herein, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, DVD ROM, or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to the processor 20, can be used for the memory 22 in the web content management server 12.

[0023] The communication interface 24 in the web content management server 12 is used to operatively couple and communicate between the web content management server 12, client devices 14(1)-14(n), and server devices 16(1)-16(n), which are all coupled together via the communication networks 18(1)-18(2), although other types and numbers of communication networks or systems with other types and numbers of connections and configurations to other devices and elements can also be used. By way of example only, the communication networks 18(1)-18(2) can use TCP/IP over Ethernet and industry-standard protocols, including hypertext transfer protocol (HTTP), secure HTTP (HTTPS), wireless application protocol (WAP), and/or SOAP, although other types and numbers of communication networks, such as a direct connection, modems and phone lines, e-mail, and wireless and hardwire communication technology, each having their own communications protocols, can be used.

[0024] The client devices 14(1)-12(n) enable a user to request, receive, and interact with applications, web services, and content hosted by the server devices 16(1)-16(n) through the web content management server 12 and using the communication network 18(1), although one or more of the client devices 14(1)-14(n) could access content and utilize other types and numbers of applications from other sources and could provide a wide variety of other functions for the user. In some examples, the client devices 14(1)-14(n) comprise mobile devices with Internet access that enable web pages and other content stored by the server devices 16(1)-16(n) to be retrieved and rendered. By way of example only, the client devices 14(1)-14(n) can be smart phones, personal digital assistants, tablets, or computers.

[0025] Each of the client devices 14(1)-14(n) includes a processor, a memory, an input device, a display device, and a communication interface, which are coupled together by a bus or other link, although one or more of client devices 14(1)-14(n) can include other numbers and types of components, parts, devices, systems, and elements in other configurations. The processor in each of the client devices 14(1)-14(n) can execute a program of instructions stored in the memory the client device 14(1)-14(n) for one or more aspects of the present invention, as described and illustrated herein, although the processor could execute other numbers and types of programmed instructions.

[0026] The input device in each of the client devices 14(1)-14(n) can be used to input selections, such as a request for a particular web page or other content stored by one or more of the server devices 16(1)-16(n), although the input device could be used to input other types of requests and data and interact with other elements. The input device can include keypads, touch screens, and/or vocal input processing systems, although other types and numbers of input devices can also be used.

[0027] The display device in each of the client devices 14(1)-14(n) can be used to show data and information to the user, such as web pages and other content retrieved from the server devices 16(1)-16(n) by way of example only. The display device in each of the client devices 14(1)-14(n) can be a mobile phone screen display, although other types and numbers of displays could be used depending on the particular type of client device.

[0028] The communication interface in each of the client devices 14(1)-14(n) can be used to operatively couple and communicate between the client devices 14(1)-14(n), web content management server 12, and server devices 16(1)-16(n) over the communication networks 18(1)-18(2).

[0029] Each of the server devices 16(1)-16(n) provides content including web pages and web applications for use by one or more of the client devices 14(1)-14(n) via the web content management server 12, although the server devices 16(1)-16(n) can provide other numbers and types of content and perform other functions. Each of the server devices 16(1)-16(n) can include a processor, a memory, and a communication interface, which are coupled together by a bus or other link, although each of the server devices 16(1)-16(n) can have other numbers and types of components, parts, devices, systems, and elements in other configurations.

[0030] The processor in each of the server devices 16(1)-16(n) executes a program of instructions stored in the memory of the server devices 16(1)-16(n) for one or more aspects of the present invention, as described and illustrated by way of the embodiments herein, although the processor could execute other numbers and types of programmed instructions.

[0031] The communication interface in each of the server devices 16(1)-16(n) is used to operatively couple and communicate between the server devices 16(1)-16(n), the web content management server 12, and the client devices 14(1)-14(n) via the communication networks 18(1)-18(2).

[0032] Although embodiments web content management server 12, the client devices 14(1)-14(n), and the server devices 16(1)-16(n) are described and illustrated herein, each of the web content management server 12, the client devices 14(1)-14(n), and the server devices 16(1)-16(n) can be implemented on any suitable computer apparatus or computing device. It is to be understood that the apparatuses and devices of the embodiments described herein are for exemplary purposes, as many variations of the specific hardware and software used to implement the embodiments are possible, as will be appreciated by those skilled in the relevant art(s).

[0033] Furthermore, each of the devices of the embodiments may be conveniently implemented using one or more general purpose computers, microprocessors, digital signal processors, and micro-controllers, programmed according to the teachings of the embodiments, as described and illustrated herein, and as will be appreciated by those ordinary skill in the art.

[0034] In addition, two or more computing apparatuses or devices can be substituted for any one of the devices in any embodiment described herein. Accordingly, principles and advantages of distributed processing, such as redundancy and replication also can be implemented, as desired, to increase the robustness and performance of the devices of the embodiments. The embodiments may also be implemented on computer apparatuses or devices that extend across any suitable network using any suitable interface mechanisms and communications technologies, including by way of example only telecommunications in any suitable form (e.g., voice and modem), wireless communications media, wireless communications networks, cellular communications networks, G3 communications networks, Public Switched Telephone Network (PSTNs), Packet Data Networks (PDNs), the Internet, intranets, and combinations thereof.

[0035] The embodiments may also be embodied as one or more non-transitory computer readable medium having instructions stored thereon for one or more aspects of the present invention as described and illustrated by way of the embodiments herein, as described herein, which when executed by a processor, cause the processor to carry out the steps necessary to implement the methods of the embodiments, as described and illustrated herein.

[0036] An exemplary method for extending a selector application programming interface (API) will now be described with reference to FIGS. 1-7. Referring more specifically to FIG. 2, a flow chart of an exemplary method for extending the selector API to an emulated JavaScript environment is illustrated. In this example, in step 200, the web content management server 12 receives an HTTP request for a web page from one of the client devices 14(1)-14(n) and retrieves the requested web page from one of the server devices 16(1)-16(n).

[0037] In step 202, the web content management server 12 processes the retrieved web page to generate an XML document object model (DOM) and to remove one or more JavaScript code portions. In this example, the web content management server 12 stores the DOM and each of the removed JavaScript portions in the memory 22. Each of the removed JavaScript portions can be stored as associated with a respective identifier.

[0038] Additionally, the web page can be modified by the web content management server 12 to insert hidden fields in place of the removed JavaScript. Each of the hidden fields include at least one of the identifiers and are inserted so as to cause the requesting one of the client devices 14(1)-14(n) to send one or more of the identifiers in a subsequent HTTP request. Optionally, the web page can be further processed to optimize the web page for a display on the requesting one of the client devices 14(1)-14(n) and/or to improve the transmission and/or load time of the web page, for example, although any other type of processing can also be performed on the web page.

[0039] In step 204, the web content management server 12 sends the processed web page to the requesting one of the client devices 14(1)-14(n). Once received, the requesting one of the client devices 14(1)-14(n) can load the web page in a web browser for example.

[0040] In step 206, the web content management server 12 receives an HTTP request which includes the identifier from the requesting one of the client devices 14(1)-14(n) in response to a user interaction with the web page. For example, the user may submit login credentials using a submit button which prompts the requesting one of the client devices 14(1)-14(n) to generate and send an HTTP request including one of the identifiers previously included in a tag associated with the submit button in step 202. In response to receiving the HTTP request, the web content management server 12 retrieves the JavaScript code corresponding to the identifier included in the request from the memory 22.

[0041] In step 208, the web content management server 12 executes the retrieved JavaScript code on behalf of the requesting one of the client devices 14(1)-14(n) in an emulated JavaScript environment. The emulated JavaScript environment essentially emulates the web browser environment of the requesting one of the client devices 14(1)-14(n). Methods and devices for utilizing an emulated JavaScript environment are described in U.S. Patent Application No. 12/802,670 entitled "Methods for utilizing a javascript emulator in a web content proxy server and devices thereof," which is incorporated herein by reference in its entirety.

[0042] During execution of the JavaScript code, the web content management server 12 determines, in step 210, whether a cascading stylesheet (CSS) selector expression is encountered. CSS selector expressions include a querySelector() or querySelectorAll() function call in this example and a plurality of expression components, although the CSS selector expressions could include other function calls and other content. Each of the querySelector() and querySelectorAll() functions are defined in the selector API and receive at least one CSS selector as a parameter. If the web content management server 12 determines that a CSS selector expression has been encountered, then the Yes branch is taken to step 212.

[0043] While steps 200-212 are illustrated for exemplary purposes, other methods of encountering or receiving a CSS selector expression can be used. Additionally, other types of environments other than an emulated JavaScript environment can be used. For example, the methods and devices described and illustrated herein are operable with any browser emulator executed on a web content management server in which the DOM for each retrieved web page is implemented using an XML DOM implementation paired with an XPath processor.

[0044] In step 212, the web content management server 12 extracts an expression component from the CSS selector expression, and more particularly, from the CSS selector(s) included as parameters in the CSS selector expression. One exemplary expression component can be "div.red" (e.g., as part of a CSS selector expression "querySelectorAll('div.red')"). This expression component would require selecting all the "div" elements of the DOM generated in step 202 having a class attribute containing "red". Other expression components can also be extracted in step 212.

[0045] In step 214, the web content management server 12 determines whether an equivalent XML path language (XPath) expression can be generated for the extracted expression component. The determination can be made based on a stored mapping of expression component formats or types to equivalent XPath expressions, for example, although other methods of determining whether the expression component has an equivalent XPath expression can also be used. In this example, the "div.red" expression component is equivalent to the "//div[contains(concat(' ', @class, ' '), 'red')]" XPath expression. Accordingly, if the web content management server 12 determines that an equivalent XPath expression can be generated for the expression component, then the Yes branch is taken to step 216.

[0046] In step 216, the web content management server 12 generates the equivalent XPath expression from the expression component. Optionally, the web content management server 12 also stores the equivalent XPath expression at least temporarily in the memory 22.

[0047] In step 218, the web content management server 12 determines whether there are more expression components that can be extracted from the CSS selector expression. If the web content management server 12 determines there are more expression components that can be extracted from the CSS selector expression, then the Yes branch is taken back to step 212 and another expression component is extracted, as described and illustrated earlier.

[0048] The web content management server 12 then determines in step 214 whether an equivalent XPath expression can be generated for the extracted expression component, also as described and illustrated earlier. In this iteration, the exemplary extracted expression component is ".red:nth-of-type(odd)". This exemplary expression component would require selecting all the elements of the DOM generated in step 202 having a class attribute containing "red" and being in an odd position in the set of all children of the element's parents having the same name as the element. In this example, the :nth-of-type pseudo class does not have an equivalent XPath expression. Accordingly, in this iteration, the web content management server 12 determines in step 214 that an equivalent XPath expression cannot be generated for the extracted expression component and the No branch is taken to step 220.

[0049] In step 220, the web content management server 12 transforms the expression component into a transformed expression including at least one XPath expression and at least one extension function. Optionally, the web content management server 12 also stores the transformed expression at least temporarily in the memory 22. The extension functions in this example are custom functions written in Java, optionally defined in a file stored in the memory 22, and associated with an XPath processor of the web content management server 12. The extension functions can be mapped to one or more psuedo classes in the memory 22 and selected for a particular transformation based on the mapping.

[0050] Optionally, the extension functions are prefixed by a namespace which facilitates locating of the extension functions by the XPath processor. Also optionally, the transformed expression can include a mathematical operation (e.g., in order to implement an "odd" parameter of the psuedo class of the expression component). An exemplary set of

extension functions is described and illustrated in more detail later with reference to FIGS. 3-7.

[0051] In this example, the expression component can be transformed into a transformed expression including "//*[contains(concat(' ', @class, ' '), ' red ')][expr:css-type-position(.) mod 2 = 1]". Accordingly, in this example, the XPath expression of the transformed expression includes "//*[contains(concat(' ', @class, ' '), ' red ')]". The extension function of the transformed expression in this example is the "css-type-position()" extension function, although other extension functions with different names can also be used.

[0052] The extension function in this example takes in an indication of an element of the DOM as a parameter (or "." which indicates all elements as used in this example), although other parameters can also be used. In this example, "expr:" is used as a namespace prefix, although any namespace prefix can be used. Additionally, a modulus operation and comparison is performed on the result of the execution of the extension function in this example, although other mathematical or logical operations can also be used.

[0053] In step 218, the web content management server 12 determines whether there are any more expression components in the CSS selector expression, as described and illustrated earlier. If the web content management server 12 determines there are no more expression components in this iteration, then the No branch is taken to step 222.

[0054] In step 222, the web content management server 12 generates a combined expression including at least all of the transformed expressions and generated equivalent XPath expressions optionally stored in the memory 22 in steps 216 and step 220. In this example, the expressions can be merged together using the XPath operators corresponding to the CSS descendants operators, for example, although other operators and methods of combining the expressions can also be used.

[0055] In step 224, the web content management server 12 applies the combined expression to the DOM, generated in step 202 and corresponding to the web page, to generate a result. The result can be a set of elements of the DOM that satisfy or correspond

to the CSS selector expression encountered in step 210, although other types and numbers of results are also possible. Subsequent to applying the combined in expression to the DOM or, referring back to step 210, if the web content management server 12 determines during execution of the JavaScript code in step 208 that a CSS selector expression has not been encountered and the No branch is taken, the web content management server proceeds to step 226.

[0056] In step 226, the web content management server 12 determines whether execution of the retrieved JavaScript has completed. If the web content management server 12 proceeds to step 226 subsequent to applying the combined expression to the DOM in step 224, it is likely that execution of the JavaScript is not complete. Accordingly, in these examples, the No branch is taken back to step 208 and the web content management server 12 continues executing the JavaScript retrieved in step 206. Additionally, the result generated in step 224 can be used by the JavaScript in these examples such as to manipulate, filter, perform an operation on, or reduce the DOM elements indicated therein. During the continued execution of the JavaScript code, the web content management server 12 again proceeds to step 210 in a subsequent iteration, as described and illustrated earlier.

[0057] However, if the No branch is taken from step 210, execution of the JavaScript is likely complete and no CSS selector expressions were encountered, or no additional CSS selector expression subsequent to a prior iteration of steps 214-224 was encountered. Accordingly, in these examples, the Yes branch is taken from step 226 to step 228. In step 228, the web content management server 12 sends a response to the HTTP request received in step 206 to the requesting one of the client devices 14(1)-14(n). The contents of the HTTP response can be based on the execution of the JavaScript code, for example, including any CSS selector expressions embedded therein.

[0058] Referring to FIGS. 3-7, exemplary methods performed by an exemplary set of extension functions when executed during the application of the combined expression to the DOM in step 224 is illustrated. The extension functions can be executed by an XPath processor, which is optionally part of an extensible stylesheet language (XSL) processor, of

the web content management server 12, for example, although other methods of executing the extension functions can also be used.

[0059] Referring more specifically to FIG. 3, a flow chart of an exemplary method performed by an extension function for evaluating the :first-of-type, :last-of-type, :nth-of-type, and :nth-last-of-type pseudo classes is illustrated. In step 300, the web content management server 12 receives an indication of a DOM element, such as passed through a parameter of the extension function. In this example, the extension function can be the "css-type-position(element)" extension function identified earlier for evaluating the :nth-of-type pseudo class of the ".red:nth-of-type(odd)" expression component of the example described and illustrated earlier, although any other name can be used for the extension function.

[0060] In step 302, the web content management server 12 instantiates a variable T having a value of the tag name of the DOM element received in step 300. In step 304, the web content management server 12 instantiates a variable C having a value of 1. Other variables and variable names can also be used.

[0061] In step 306, the web content management server 12 determines whether there are any preceding siblings in the DOM of the DOM element received in step 300. If the web content management server 12 determines there are any preceding siblings in the DOM, then the Yes branch is taken to step 308. In step 308, the web content management server 12 determines whether one of the preceding siblings is an element and has a tag name equal to T. If the web content management server 12 determines that the preceding sibling is an element and has a tag name equal to T, then the Yes branch is taken to step 310. In step 310, the web content management server 12 increments the value of C. If the web content management server 12 determines that the preceding sibling is not an element or does not have a tag name equal to T, then the No branch is taken back to step 306.

[0062] In a subsequent iteration of step 306 in this example, the web content management server 12 again determines whether there are any additional preceding siblings of the element that have not yet been considered. If the web content management server 12 determines there is not any additional preceding siblings, then the No branch is taken to step

312. In step 312, the web content management server 12 returns the value of C. The returned value can be used to continue evaluating the combined expression applied to the DOM, for example. Accordingly, the extension function in this example returns a number of preceding sibling elements of the element of the DOM having the same tag name as the element plus one.

[0063] Referring more specifically to FIG. 4, a flow chart of an exemplary method performed by an extension function for evaluating the :last-of-type, :only-of-type, and :nth-last-of-type pseudo classes is illustrated. In step 400, the web content management server 12 receives an indication of a DOM element, such as passed through a parameter of the extension function. In this example, the extension function can be a "css-type-total(element)" extension function, although any other name can be used for the extension function.

[0064] In step 402, the web content management server 12 instantiates a variable T having a value of the tag name of the DOM element received in step 400. In step 404, the web content management server 12 instantiates a variable C having a value of 0. Other variables and variable names can also be used.

[0065] In step 406, the web content management server 12 determines whether there are any children in the DOM of the parent of the DOM element received in step 400 or, alternatively, any siblings of the DOM element. If the web content management server 12 determines there are children of the parent of the DOM element, then the Yes branch is taken to step 408. In step 408, the web content management server 12 determines whether one of the children is an element and has a tag name equal to T. If the web content management server 12 determines that the child is an element and has a tag name equal to T, then the Yes branch is taken to step 410. In step 410, the web content management server 12 increments the value of C. If the web content management server 12 determines that the child is not an element or does not have a tag name equal to T, then the No branch is taken back to step 406.

[0066] In a subsequent iteration of step 406 in this example, the web content management server 12 again determines whether there are any additional children of the

parent of the element that have not yet been considered. If the web content management server 12 determines there is not any additional children, then the No branch is taken to step 312. In step 312, the web content management server 12 returns the value of C. The returned value can be used to continue evaluating the combined expression applied to the DOM, for example. Accordingly, the extension function in this example returns a number of child elements of the parent of the element of the DOM having the tag name one of the element.

[0067] Referring more specifically to FIG. 5, a flow chart of an exemplary method performed by an extension function for evaluating the :lang pseudo class is illustrated. In step 500, the web content management server 12 receives an indication of a DOM node, such as passed through a parameter of the extension function. In this example, the extension function can be a "lang(node)" extension function, although any other name can be used for the extension function.

[0068] In step 502, the web content management server 12 instantiates a variable N having a value indicating the node received in step 500. In step 504, the web content management server 12 instantiates a variable L having a value of 'en' or any other default indication of a language. Other variables and variable names can also be used.

[0069] In step 506, the web content management server 12 determines whether N indicates a node that is an element. If the web content management server 12 determines N indicates a node that is an element, then the Yes branch is taken to step 508. In step 508, the web content management server 12 determines whether the node indicated by N has a language attribute. If the web content management server 12 determines that the node indicated by N does not have a language attribute, then the No branch is taken to step 510.

[0070] In step 510, the web content management server 12 sets the value of N to indicate the parent of the node indicated by the current value of N. The web content management server 12 then proceeds back to step 506 and again determines whether N indicates a node that is an element. Referring back to step 508, if the web content

management server 12 determines that the node indicated by N does have a language attribute, then the Yes branch is taken to step 512.

[0071] In step 512, the value of L is assigned the value of the language attribute of the node indicated by N. In step 514, the web content management server 12 returns the value of L. Referring back to step 506, if the web content management server 12 determines in a first or subsequent iteration that N indicates a node that is not an element, then the No branch is taken to step 514 and the default value of L is returned. The returned value can be used to continue evaluating the combined expression applied to the DOM, for example. Accordingly, the extension function in this example returns a value of a first identified language attribute based on a traversal of the element of the DOM and any ancestors of the element of the DOM.

[0072] Referring more specifically to FIG. 6, a flow chart of an exemplary method performed by an extension function for evaluating the :not pseudo class is illustrated. In step 600, the web content management server 12 receives an indication of a DOM element and a string including a CSS selector expression, such as passed through a parameter of the extension function. In this example, the extension function can be a "filter-out(element, css-selector)" extension function, although any other name can be used for the extension function.

[0073] In step 602, the web content management server 12 applies the received CSS selector expression to the DOM to generate a result, such as described and illustrated earlier with reference to steps 210-224, for example, although other methods of evaluating the received CSS selector expression can also be used. In this example, the CSS selector expression includes a CSS selector function querySelectorAll(), although other functions can also be included in the CSS selector expression of the string received in step 600.

[0074] In step 604, the web content management server 12 determines whether there are any nodes in the result generated by the application of the CSS selector expression in step 602. If the web content management server 12 determines there are node(s) in the result, then the Yes branch is taken to step 606. In step 606, the web content management server 12

determines whether one of the node(s) in the result is equal to the DOM element received in step 600. If the web content management server 12 determines that the one of the node(s) in the result is equal to the DOM element, then the Yes branch is taken to step 608. In step 608, the web content management server 12 returns a value of false.

[0075] Referring back to step 606, if the web content management server 12 determines that the one of the node(s) in the result is not equal to the DOM element, then the No branch is taken back to step 604. If the web content management server 12 determines in step 604 that the no nodes in the result of the evaluation of the CSS selector expression, or no additional nodes not previously considered, then the No branch is taken to step 610. In step 610, the web content management server 12 returns a value of true. Accordingly, the extension function in this example returns a true value when the element of the DOM received in step 600 is not included in a result obtained by applying the CSS selector expression received in step 600 and a false value when the element of the DOM is included in the result.

[0076] Referring more specifically to FIG. 7, a flow chart of an exemplary method performed by an extension function for evaluating the :selected jQuery pseudo class is illustrated. In step 700, the web content management server 12 receives an indication of a DOM option element, such as passed through a parameter of the extension function. In this example, the extension function can be an "option-selected(option)" extension function, although any other name can be used for the extension function. The indication of the option element can represent an option in a dropdown list in an HTML form, for example, although the option can also refer to other types of elements.

[0077] In step 701, the web content management server 12 determines whether the option is explicitly selected. If the web content management server 12 determines that the option is explicitly selected, then the Yes branch is taken to step 704. In step 704, the web content management server returns a value of true. Referring back to step 702, if the web content management server 12 determines that the option is not explicitly selected, then the No branch is taken to step 706.

[0078] In step 706, the web content management server 12 obtains a select element from the DOM that contains the option element corresponding to the indication received in step 700. In step 708, the web content management server 12 determines whether the select element is a single selection element with size attribute equal to 1. If the web content management server 12 determines that the select element is a single selection element with size attribute equal to 1 then the Yes branch is taken to step 210.

[0079] In step 710, the web content management server 12 determines whether the option element corresponding to the indication received in step 700 is the first option element descendant of the select element. If the web content management server 12 determines that the option element is the first option element descendant of the select element, then the Yes branch is taken to step 712.

[0080] In step 712, the web content management server 12 determines whether all of the other option elements descendant from the select element are not selected. If the web content management server 12 determines that all of the other option elements descendant from the select element are not selected then the Yes branch is taken to step 714. In step 714, the web content management server 12 returns a value of true.

[0081] Referring back to steps 708, 710, and 712, if the web content management server determines that the select element is not a single selection element or does not have a size attribute equal to 1, the option element corresponding to the indication received in step 700 is not the first option element descendant of the select element, or any of the other option elements descendant from the select element are selected, respectively, then one of the respective No branches is taken to step 716. In step 716, the web content management server 12 returns a value of false. Accordingly, the extension function in this example returns a true value if the option element of the DOM is selected and a false value if the option element of the DOM is not selected. In this example, the option is considered selected if it is explicitly selected or if it is the first option of a select element that has no explicitly selected options and is not a multiple selections select element.

[0082] With this technology, web page scripts including JavaScript code can be emulated by a web content management server on behalf of client devices that may not otherwise be able to take advantage of such functionality. Advantageously, any CSS selector expressions embedded in the JavaScript can be translated to XPath expressions based on an equivalent XPath expression or, in the case of CSS selector expressions without equivalent XPath expressions, by using an extension function. Thereby, CSS selector expressions can be fully translatable to XPath expressions and implemented using a relatively efficient XPath processor without the significant additional code and associated complexity required to implement the CSS selector expressions of the selector API.

[0083] Having thus described the basic concept of the invention, it will be rather apparent to those skilled in the art that the foregoing detailed disclosure is intended to be presented by way of example only, and is not limiting. Various alterations, improvements, and modifications will occur and are intended to those skilled in the art, though not expressly stated herein. These alterations, improvements, and modifications are intended to be suggested hereby, and are within the spirit and scope of the invention. Additionally, the recited order of processing elements or sequences, or the use of numbers, letters, or other designations therefore, is not intended to limit the claimed processes to any order except as may be specified in the claims. Accordingly, the invention is limited only by the following claims and equivalents thereto.

CLAIMS

What is claimed is:

1. A method for extending a selector application programming interface (API), the method comprising:

receiving, with a web content management server, a cascading style sheet (CSS) selector expressions comprising a plurality of expression components;

determining, with the web content management server, whether an equivalent XML path language (XPath) expression can be generated for each of the expression components; and

transforming, with the web content management server, each of the expression components for which an equivalent XPath expression cannot be generated into a transformed expression comprising at least one XPath expression and an extension function.

2. The method of claim 1 further comprising generating, with the web content management server, the equivalent XPath expression for each of the plurality of expression components for which the equivalent XPath expression can be generated.

3. The method of claim 2 further comprising:

generating, with the web content management server, a combined expression comprising each of the transformed expressions and each of the equivalent XPath expressions; and

applying, with the web content management server, the combined expression to a document object model (DOM) to generate a result.

4. The method of claim 3 wherein the extension function comprises JavaScript code and the applying further comprises executing the extension function in an emulated JavaScript environment.

5. The method of claim 3 wherein:

at least one of the expression components for which an equivalent XPath expression cannot be generated comprises a pseudo class comprising :first-of-type, :last-of-type, :nth-of-type, or :nth-last-of-type; and

the extension function of the transformed expression for the at least one of the expression components is configured to receive an indication of an element of the DOM, the indication comprising a tag name of the element of the DOM, and return a number of one or more preceding sibling elements of the element of the DOM having the tag name.

6. The method of claim 3 wherein:

at least one of the expression components for which an equivalent XPath expression cannot be generated comprises a pseudo class comprising :last-of-type, :only-of-type, or :nth-last-of-type; and

the extension function of the transformed expression for the at least one of the expression components is configured to receive an indication of an element of the DOM, the indication comprising a tag name of the element of the DOM and return a number of child elements of the parent of the element of the DOM having the tag name plus one.

7. The method of claim 3 wherein:

at least one of the expression components for which an equivalent XPath expression cannot be generated comprises a :lang pseudo class; and

the extension function of the transformed expression for the at least one of the expression components is configured to receive an indication of an element of the DOM and return a value of a first identified language attribute based on a traversal of the element of the DOM and any ancestors of the element of the DOM.

8. The method of claim 3 wherein:

at least one of the expression components for which an equivalent XPath expression cannot be generated comprises a :not pseudo class; and

the extension function of the transformed expression for the at least one of the expression components is configured to receive an indication of an element of the DOM and a second CSS selector expression and return a true value when the element of the DOM is not included in a second result obtained by applying the second CSS selector expression and a false value when the element of the DOM is included in the second result.

9. The method of claim 3 wherein:

at least one of the expression components for which an equivalent XPath expression cannot be generated comprises a :selected jQuery pseudo class; and

the extension function of the transformed expression for the at least one of the expression components is configured to receive an option element of the DOM and return a true value if the option element of the DOM is selected and a false value if the option element of the DOM is not selected.

10. A non-transitory computer readable medium having stored thereon instructions for extending a selector application programming interface (API) comprising machine executable code which when executed by a processor, causes the processor to perform steps comprising:

receiving a cascading style sheet (CSS) selector expressions comprising a plurality of expression components;

determining whether an equivalent XML path language (XPath) expression can be generated for each of the expression components; and

transforming each of the expression components for which an equivalent XPath expression cannot be generated into a transformed expression comprising at least one XPath expression and an extension function.

11. The medium of claim 10 further having stored thereon instructions comprising machine executable code which when executed by the processor, causes the processor to perform steps further comprising generating the equivalent XPath expression for

each of the plurality of expression components for which the equivalent XPath expression can be generated.

12. The medium of claim 11 further having stored thereon instructions comprising machine executable code which when executed by the processor, causes the processor to perform steps further comprising:

generating a combined expression comprising each of the transformed expressions and each of the equivalent XPath expressions; and

applying the combined expression to a document object model (DOM) to generate a result.

13. The medium of claim 12 wherein the extension function comprises JavaScript code and the applying further comprises executing the extension function in an emulated JavaScript environment.

14. The medium of claim 12 wherein:

at least one of the expression components for which an equivalent XPath expression cannot be generated comprises a pseudo class selected from :first-of-type, :last-of-type, :nth-of-type, or :nth-last-of-type; and

the extension function of the transformed expression for the at least one of the expression components is configured to receive an indication of an element of the DOM, the indication comprising a tag name of the element of the DOM, and return a number of preceding sibling elements of the element of the DOM having the tag name.

15. The medium of claim 12 wherein:

at least one of the expression components for which an equivalent XPath expression cannot be generated comprises a pseudo class selected from :last-of-type, :only-of-type, or :nth-last-of-type; and

the extension function of the transformed expression for the at least one of the expression components is configured to receive an indication of an element of the

DOM, the indication comprising a tag name of the element of the DOM and return a number of child elements of the parent of the element of the DOM having the tag name plus one.

16. The medium of claim 12 wherein:

at least one of the expression components for which an equivalent XPath expression cannot be generated comprises a :lang pseudo class; and

the extension function of the transformed expression for the at least one of the expression components is configured to receive an indication of an element of the DOM and return a value of a first identified language attribute based on a traversal of the element of the DOM and any ancestors of the element of the DOM.

17. The medium of claim 12 wherein:

at least one of the expression components for which an equivalent XPath expression cannot be generated comprises a :not pseudo class; and

the extension function of the transformed expression for the at least one of the expression components is configured to receive an indication of an element of the DOM and a second CSS selector expression and return a true value when the element of the DOM is not included in a second result obtained by applying the second CSS selector expression and a false value when the element of the DOM is included in the second result.

18. The medium of claim 12 wherein:

at least one of the expression components for which an equivalent XPath expression cannot be generated comprises a :selected jQuery pseudo class; and

the extension function of the transformed expression for the at least one of the expression components is configured to receive an option element of the DOM and return a true value if the option element of the DOM is selected and a false value if the option element of the DOM is not selected.

19. A management server device, comprising:

a processor coupled to a memory and configured to execute programmed instructions stored in the memory comprising:

receiving a cascading style sheet (CSS) selector expressions comprising a plurality of expression components;

determining whether an equivalent XML path language (XPath) expression can be generated for each of the expression components; and

transforming each of the expression components for which an equivalent XPath expression cannot be generated into a transformed expression comprising at least one XPath expression and an extension function.

20. The device of claim 19 wherein the processor is further configured to execute programmed instructions stored in the memory further comprising generating the equivalent XPath expression for each of the plurality of expression components for which the equivalent XPath expression can be generated.

21. The device of claim 20 wherein the processor is further configured to execute programmed instructions stored in the memory further comprising:

generating a combined expression comprising each of the transformed expressions and each of the equivalent XPath expressions; and

applying the combined expression to a document object model (DOM) to generate a result.

22. The device of claim 21 wherein the extension function comprises JavaScript code and the applying further comprises executing the extension function in an emulated JavaScript environment.

23. The device of claim 21 wherein:

at least one of the expression components for which an equivalent XPath expression cannot be generated comprises a pseudo class selected from :first-of-type, :last-of-type, :nth-of-type, or :nth-last-of-type; and

the extension function of the transformed expression for the at least one of the expression components is configured to receive an indication of an element of the DOM, the indication comprising a tag name of the element of the DOM, and return a number of preceding sibling elements of the element of the DOM having the tag name.

24. The device of claim 21 wherein:

at least one of the expression components for which an equivalent XPath expression cannot be generated comprises a pseudo class selected from :last-of-type, :only-of-type, or :nth-last-of-type; and

the extension function of the transformed expression for the at least one of the expression components is configured to receive an indication of an element of the DOM, the indication comprising a tag name of the element of the DOM and return a number of child elements of the parent of the element of the DOM having the tag name plus one.

25. The device of claim 21 wherein:

at least one of the expression components for which an equivalent XPath expression cannot be generated comprises a :lang pseudo class; and

the extension function of the transformed expression for the at least one of the expression components is configured to receive an indication of an element of the DOM and return a value of a first identified language attribute based on a traversal of the element of the DOM and any ancestors of the element of the DOM.

26. The device of claim 21 wherein:

at least one of the expression components for which an equivalent XPath expression cannot be generated comprises a :not pseudo class; and

the extension function of the transformed expression for the at least one of the expression components is configured to receive an indication of an element of the DOM and a second CSS selector expression and return a true value when the element of the DOM is not included in a second result obtained by applying the second CSS selector expression and a false value when the element of the DOM is included in the second result.

27. The device of claim 21 wherein:

at least one of the expression components for which an equivalent XPath expression cannot be generated comprises a :selected jQuery pseudo class; and

the extension function of the transformed expression for the at least one of the expression components is configured to receive an option element of the DOM and return a true value if the option element of the DOM is selected and a false value if the option element of the DOM is not selected.

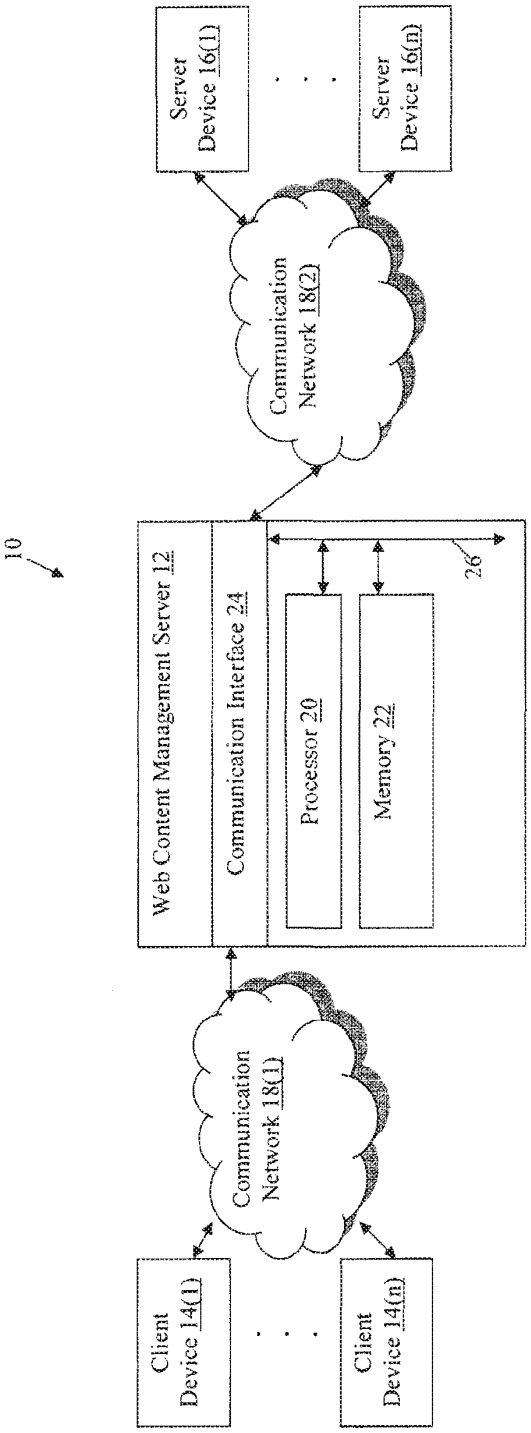

10
Web Content Management Server 12
Communication Interface 24
Processor 20
Memory 22
26
Communication Network 18(1)
Communication Network 18(2)
Client Device 14(1)
Client Device 14(n)
Server Device 16(1)
Server Device 16(n)

FIG. 1

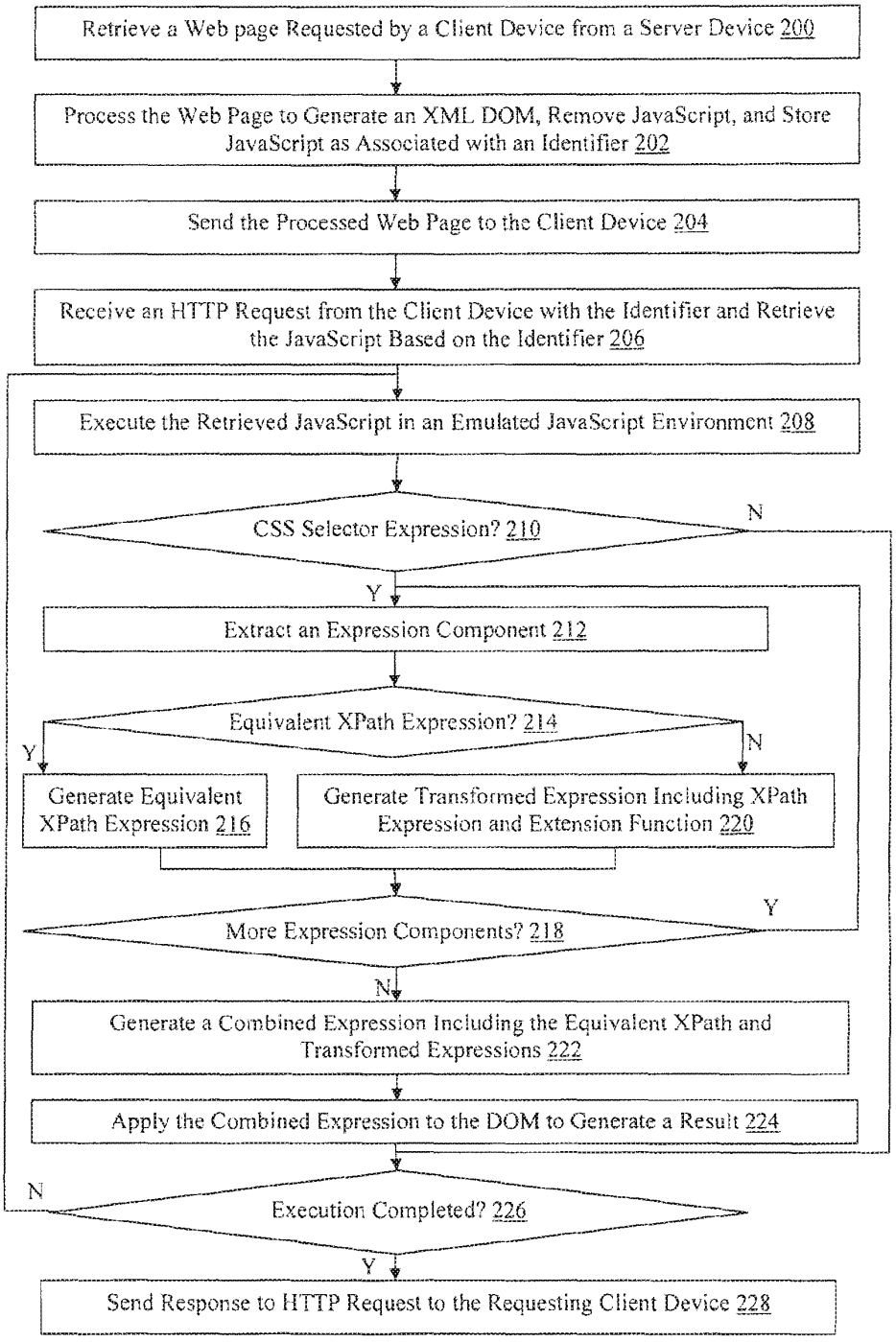

Retrieve a Web page Requested by a Client Device from a Server Device 200
Process the Web Page to Generate an XML DOM, Remove JavaScript, and Store JavaScript as Associated with an Identifier 202
Send the Processed Web Page to the Client Device 204
Receive an HTTP Request from the Client Device with the Identifier and Retrieve the JavaScript Based on the Identifier 206
Execute the Retrieved JavaScript in an Emulated JavaScript Environment 208
CSS Selector Expression? 210
N
Y
Extract an Expression Component 212
Equivalent XPath Expression? 214
Y
N
Generate Equivalent XPath Expression 216
Generate Transformed Expression Including XPath Expression and Extension Function 220
More Expression Components? 218
Y
N
Generate a Combined Expression Including the Equivalent XPath and Transformed Expressions 222
Apply the Combined Expression to the DOM to Generate a Result 224
N
Execution Completed? 226
Y
Send Response to HTTP Request to the Requesting Client Device 228

FIG. 2

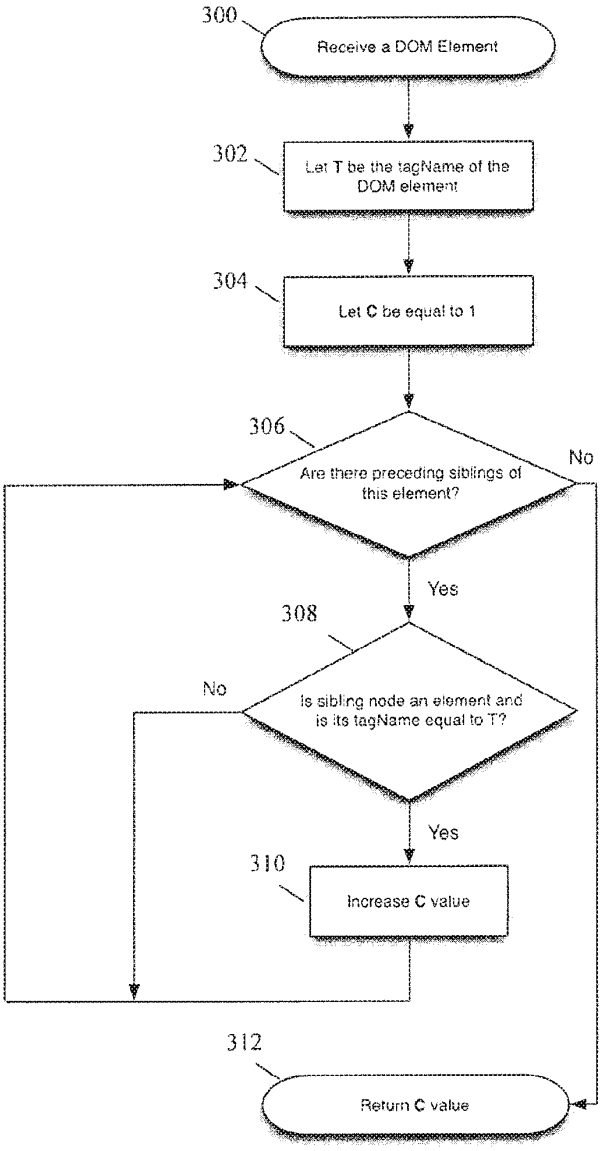

300
Receive a DOM Element
302
Let T be the tagName of the DOM element
304
Let C be equal to 1
306
Are there preceding siblings of this element?
No
Yes
308
Is sibling node an element and is its tagName equal to T?
No
Yes
310
Increase C value
312
Return C value

FIG. 3

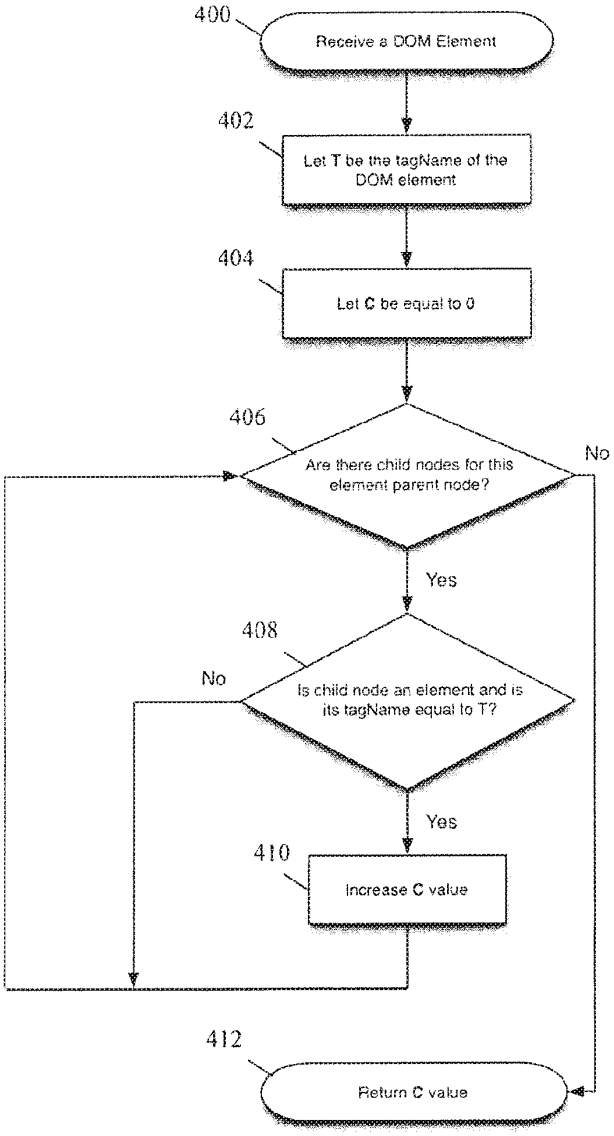

400
Receive a DOM Element
402
Let T be the tagName of the DOM element
404
Let C be equal to 0
406
Are there child nodes for this element parent node?
No
Yes
408
Is child node an element and is its tagName equal to T?
No
Yes
410
Increase C value
412
Return C value

FIG. 4

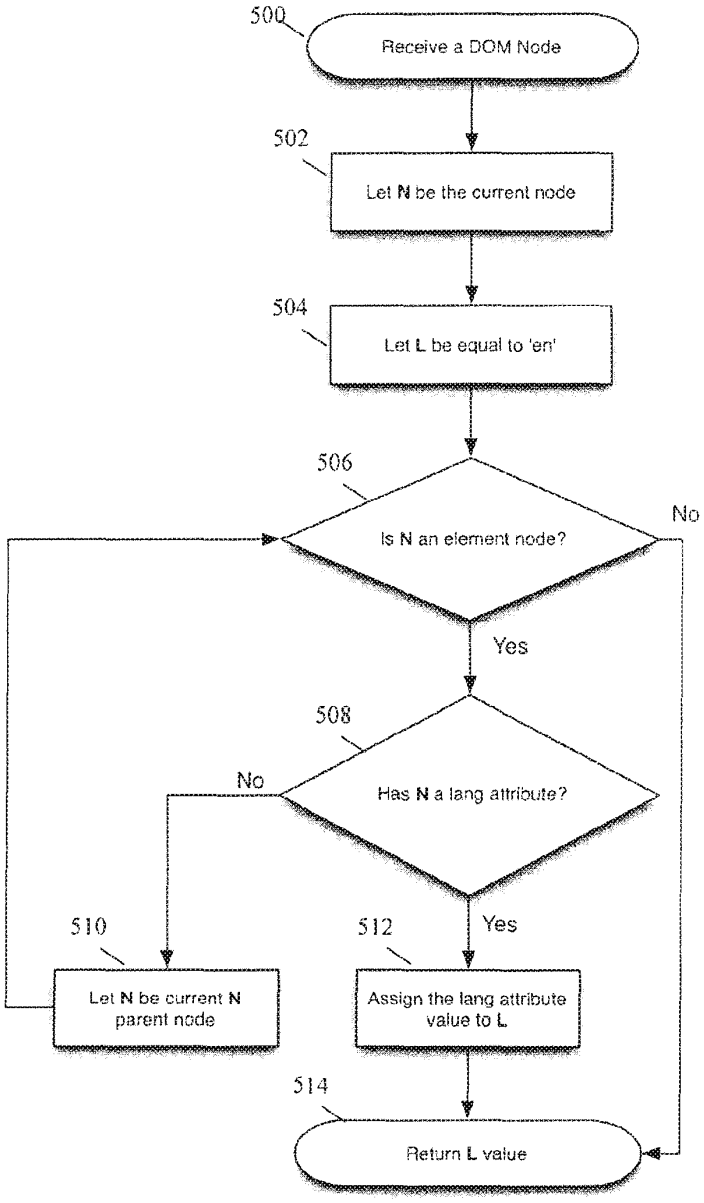

500
Receive a DOM Node
502
Let N be the current node
504
Let L be equal to 'en'
506
Is N an element node?
No
Yes
508
Has N a lang attribute?
No
Yes
510
Let N be current N parent node
512
Assign the lang attribute value to L
514
Return L value

FIG. 5

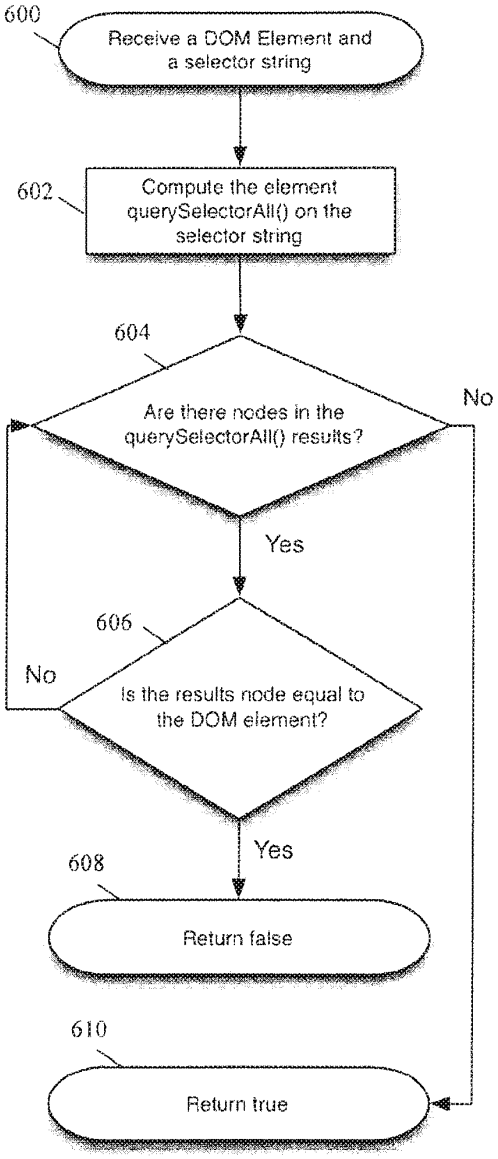

600
Receive a DOM Element and a selector string
602
Compute the element querySelectorAll() on the selector string
604
Are there nodes in the querySelectorAll() results?
No
Yes
606
No
Is the results node equal to the DOM element?
Yes
608
Return false
610
Return true

FIG. 6

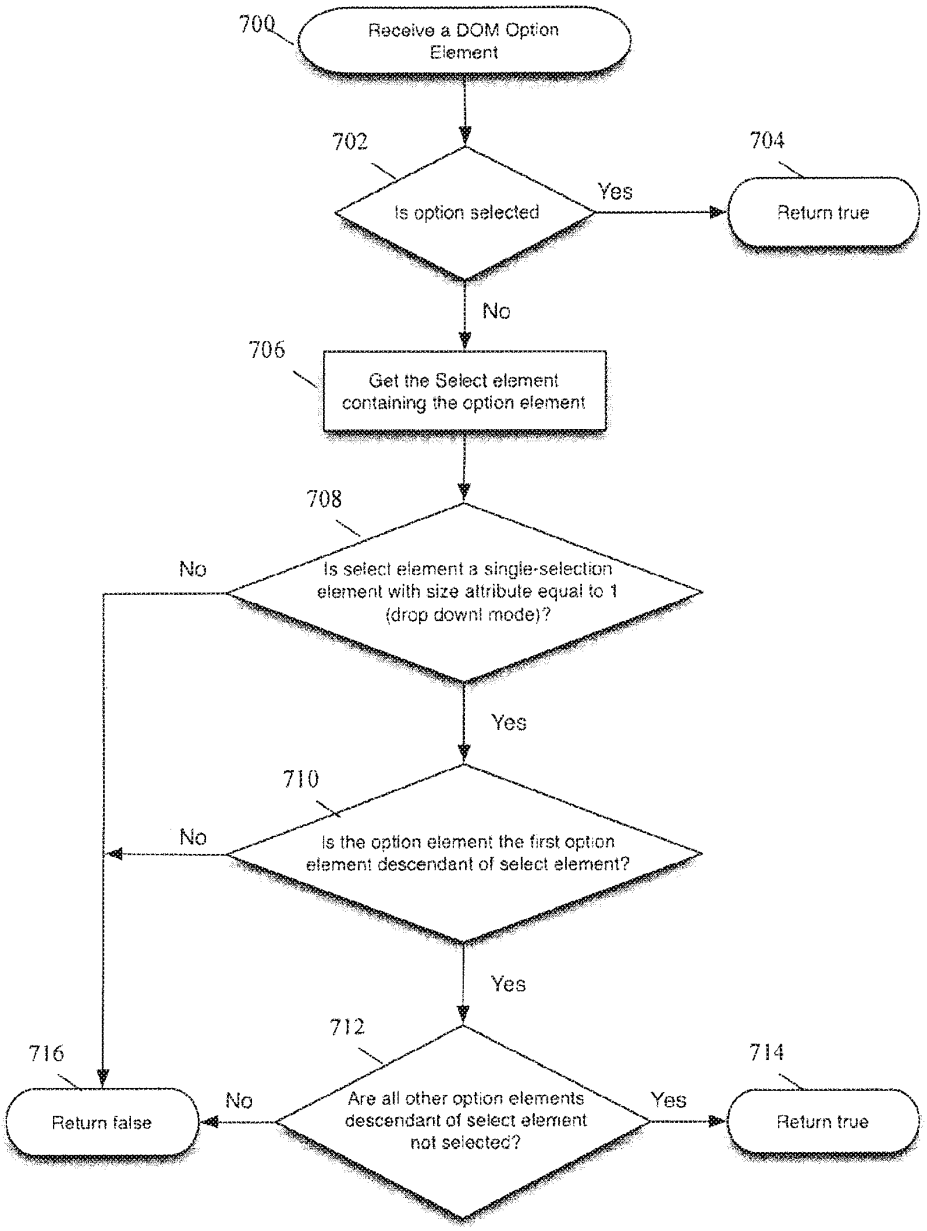

700
Receive a DOM Option Element
702
Is option selected
Yes
704
Return true
No
706
Get the Select element containing the option element
708
Is select element a single-selection element with size attribute equal to 1 (drop down mode)?
No
Yes
710
Is the option element the first option element descendant of select element?
No
Yes
712
Are all other option elements descendant of select element not selected?
No
Yes
714
Return true
716
Return false

FIG. 7

10
Server Device 16(1)
Server Device 16(n)
Communication Network 18(2)
Web Content Management Server 12
Communication Interface 24
Processor 20
Memory 22
26
Communication Network 18(1)
Client Device 14(1)
Client Device 14(n)

Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2944659 A1 2017/04/15

(21) 2 944 659

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(22) Date de dépôt/Filing Date: 2016/10/05

(41) Mise à la disp. pub./Open to Public Insp.: 2017/04/15

(30) Priorité/Priority: 2015/10/15 (US14/883,886)

(51) Cl.Int./Int.Cl. *H04L 12/16* (2006.01)

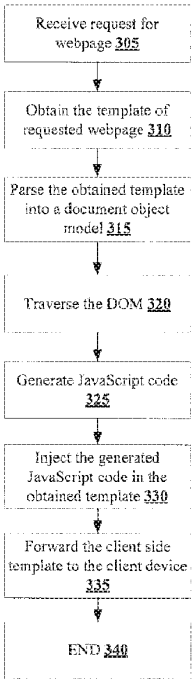
(71) Demandeur/Applicant:
USABLENET INC., US

(72) Inventeurs/Inventors:
SCODA, ENRICO, IT;
PITTINO, LUCA, IT;
PEZZANO, SIMONE, IT

(74) Agent: PARLEE MCLAWS LLP

(54) Titre : METHODES DE TRANSFORMATION D'UN GABARIT COTE SERVEUR EN UN GABARIT COTE CLIENT ET DISPOSITIFS ASSOCIES

(54) Title: METHODS FOR TRANSFORMING A SERVER SIDE TEMPLATE INTO A CLIENT SIDE TEMPLATE AND DEVICES THEREOF



(57) Abrégé/Abstract:
A method, non-transitory computer readable medium, and web content server that assists with transforming server side template to client side template includes obtaining a server side template comprising a plurality of Hypertext Markup Language (HTML) and

CA 2944659 A1 2017/04/15
(21) 2 944 659
(13) A1

(57) Abrégé(suite)/Abstract(continued):
logical instructions responsive to a request for webpage received from a client device. The obtained server side template is parsed to generate a document object model. The parsed document object model is traversed to identify the plurality of HTML and logical instructions. The obtained server side template is transformed to a client side template by replacing each of the identified plurality of HTML and logical instructions during the traversing with an equivalent JavaScript code.

ABSTRACT

A method, non-transitory computer readable medium, and web content server that assists with transforming server side template to client side template includes obtaining a server side template comprising a plurality of Hypertext Markup Language (HTML) and logical instructions responsive to a request for webpage received from a client device. The obtained server side template is parsed to generate a document object model. The parsed document object model is traversed to identify the plurality of HTML and logical instructions. The obtained server side template is transformed to a client side template by replacing each of the identified plurality of HTML and logical instructions during the traversing with an equivalent JavaScript code.

METHODS FOR TRANSFORMING A SERVER SIDE TEMPLATE INTO A CLIENT SIDE TEMPLATE AND DEVICES THEREOF

FIELD

[0001] This technology generally relates to methods and devices for optimizing web content and, more particularly, methods for transforming a server side template into a client side template and devices thereof.

BACKGROUND

[0002] Many web sites are increasingly sophisticated and provide rich user experiences. Accordingly, in web development, the user or client side dynamic code is often crafted for each specific case to provide rich user experiences. In prior technologies, a dynamic Hypertext Markup Language (HTML) based template (server side template) is used to describe the websites to provide rich user experience. The dynamic HTML based template includes references to variables, express conditions, description of iterations and other types of HTML instructions. Accordingly, with the growth of client side web technologies, a need of having client side templates which would generate HTML directly on the browser grew as well.

[0003] Most commonly, the existing products working as client side templates are made of HTML extensions and are using JavaScript as logical language. However, as previously illustrated, the server side templates are dynamic HTML based templates which do not share most peculiarities with the current client side implementations. Therefore with prior technologies, software developers were required to know multiple programming and markup languages in order to develop and transform the server side template to client side templates. As noted earlier, with many websites having a sophisticated layout to provide rich user experience, the manual transformation of both server side templates and client side templates of one into another became a tedious task.

Additionally, due to large amounts of human intervention (web developers), the methods used in the prior technologies also become inefficient and error prone.

SUMMARY

[0004] A method for transforming server side template to a client side template includes obtaining, by a web content server, a server side template comprising a plurality of Hypertext Markup Language (HTML) and logical instructions responsive to a request for webpage received from a client device. The obtained server side template is parsed by the web content server to generate a document object model. The parsed document object model is traversed by the web content server to identify the plurality of HTML and logical instructions. The obtained server side template is transformed by the web content server to a client side template by replacing each of the identified plurality of HTML instructions and logical during the traversing with an equivalent JavaScript code.

[0005] A non-transitory computer readable medium having stored thereon instructions transforming server side template to a client side template comprising machine executable code which when executed by a processor, causes the processor to perform steps including obtaining a server side template comprising a plurality of Hypertext Markup Language (HTML) and logical instructions responsive to a request for webpage received from a client device. The obtained server side template is parsed to generate a document object model. The parsed document object model is traversed to identify the plurality of HTML and logical instructions. The obtained server side template is transformed to a client side template by replacing each of the identified plurality of HTML and logical instructions during the traversing with an equivalent JavaScript code.

[0006] A web content server includes a processor coupled to a memory and configured to execute programmed instructions stored in the memory including obtaining a server side template comprising a plurality of Hypertext Markup Language (HTML) and logical instructions responsive to a request for webpage received from a client

device. The obtained server side template is parsed to generate a document object model. The parsed document object model is traversed to identify the plurality of HTML and logical instructions. The obtained server side template is transformed to a client side template by replacing each of the identified plurality of HTML and logical instructions during the traversing with an equivalent JavaScript code.

[0007] This technology provides a number of advantages including methods, non-transitory computer readable medium, and a web content server for transforming templates designed for the server side execution to templates that are suitable to be executed on the client side. Additionally, this technology allows developers to use one language for both developing server side pages and client side content. Further with this technology, the server side templates can be transformed to client side templates that allow the creation of complex web pages where the displayed data can change without the page being reloaded. Even further, with this technology by directly transforming the server side template to a client side template, an autonomous JavaScript based template is created for the client side without the need of an intermediate template language to be reinterpreted on the browser of the client device.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] FIG. 1 is a block diagram of an environment with an example of a web content server that effectively transforms a server side template into a client side template;

[0009] FIG. 2 is a block diagram of the example of the web content server shown in FIG. 1;and

[0010] FIG. 3 is a flow chart of an example of a method for transforming server side template into a client side template;

[0011] FIGS. 4-5 are exemplary server side templates; and

[0012] FIGS. 6-7 are exemplary client side templates that are transformed from the server side templates.

DETAILED DESCRIPTION

[0013] An exemplary environment 10 with a web content server 14 coupled to plurality of client computing devices 12(1)-12(n) and plurality of server devices 16(1)-16(n) by communication network 30 is illustrated in FIG. 1. Other numbers and types of systems, devices, and/or elements in other configurations and environments with other communication network topologies can also be used. This technology provides a number of advantages including providing methods, non-transitory computer readable medium, and devices for transforming templates designed for the server side execution to templates that are suitable to be executed on the client side.

[0014] Referring more specifically to FIG. 1, the web content server 14 includes a central processing unit (CPU) 18 or processor, a memory 20, and a communication interface 24, which are coupled together by a bus 26 or other link, although other numbers and types of components, parts, devices, systems, and elements in other configurations and locations can be used.

[0015] The processor 18 of the web content server 14 may execute one or more programmed instructions stored in the memory 20 for transforming templates designed for the server side execution to templates that are suitable to be executed on the client side as illustrated and described in the examples herein, although other types and numbers of functions and/or other operation can be performed. The processor 18 of the web content server 14 may include one or more central processing units ("CPUs") or general purpose processors with one or more processing cores, such as AMD® processor(s), although other types of processor(s) could be used (e.g., Intel®).

[0016] The memory 20 of the web content server 14 stores the programmed instructions and other data for one or more aspects of the present technology as described and illustrated herein, although some or all of the programmed instructions could be

stored and executed elsewhere. A variety of different types of memory storage devices, such as a non-volatile memory, random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, DVD ROM, or other computer readable medium which is read from and written to by a magnetic, optical, or other reading and writing system that is coupled to the processor 18, can be used for the memory 20.

[0017] The communication interface 24 in the web content server 14 is used to operatively couple and communicate between the web content server 14, plurality of client computing devices 12(1)-12(n), and plurality of server devices 16(1)-16(n) via the communication network 30. One or more of the communication network 30 can include one or more networks, such as one or more local area networks (LANs) and/or wide area networks (WANs). By way of example only, the communication networks 30 can use TCP/IP over Ethernet and industry-standard protocols, including hypertext transfer protocol (HTTP), secure HTTP (HTTPS), wireless application protocol (WAP), and/or SOAP, although other types and numbers of communication networks, such as a direct connection, modems and phone lines, e-mail, and wireless and hardwire communication technology, each having their own communications protocols, can be used.

[0018] The plurality of client devices 12(1)-12(n) enable a user to request, receive, and interact with applications, web services, and content hosted by the plurality of server devices 16(1)-16(n) through the web content server 14 via the communication network 30, although one or more of the plurality of client computing devices 12(1)-12(n) could access content and utilize other types and numbers of applications from other sources and could provide a wide variety of other functions for the user. In some examples, the plurality of client computing devices 12(1)-12(n) comprise mobile computing devices with Internet access that enable web pages and other content stored by the plurality of server devices 16(1)-16(n) to be retrieved and rendered. By way of example only, the plurality of client computing devices 12(1)-12(n) can be smart phones, personal digital assistants, or computers.

[0019] Each of the plurality of client computing devices 12(1)-12(n) includes a CPU, a memory, an input device, a display device, and an input/output device, which are coupled together by a bus or other link, although one or more of plurality of client computing devices 12(1)-12(n) can include other numbers and types of components, parts, devices, systems, and elements in other configurations. The CPU in the plurality of client computing devices 12(1)-12(n) can execute a program of instructions stored in the memory of the plurality of client computing devices 12(1)-12(n) for one or more aspects of the present invention as described and illustrated herein, although the CPU could execute other numbers and types of programmed instructions.

[0020] The input device in each of the plurality of client computing devices 12(1)-12(n) can be used to input selections, such as a request for a particular web page, although the input device could be used to input other types of requests and data and interact with other elements. The input device can include keypads, touch screens, and/or vocal input processing systems, although other types and numbers of input devices can be used.

[0021] The display device in each of the plurality of client computing devices 12(1)-12(n) can be used to show data and information to the user, such as web pages retrieved from the plurality of server devices 16(1)-16(n) by way of example only. The display device in each of the plurality of client computing devices 12(1)-12(n) can be a mobile phone screen display, although other types and numbers of displays could be used depending on the particular type of client device.

[0022] The input/output device in each of the plurality of client computing devices 12(1)-12(n) can be used to operatively couple and communicate between the plurality of client computing devices 12(1)-12(n), the web content server 14, and the plurality of server devices 16(1)-16(n) over the communication network 30.

[0023] Each of the plurality of server devices 16(1)-16(n) provides content including web pages for use by one or more of the plurality of client computing devices

12(1)-12(n) via the web content server 14, although the plurality of server devices 16(1)-16(n) can provide other numbers and types of content and perform other functions. Each of the plurality of server devices 16(1)-16(n) can include a CPU, a memory, and an input/output device, which are coupled together by a bus or other link, although each of the plurality of server devices 16(1)-16(n) could have other numbers and types of components, parts, devices, systems, and elements in other configurations and locations.

[0024] The CPU in each of the plurality of server devices 16(1)-16(n) executes a program of instructions stored in the memory of the plurality of server devices 16(1)-16(n) for one or more aspects of the present invention, as described and illustrated by way of the embodiments herein, although the CPU could execute other numbers and types of programmed instructions.

[0025] The input/output device in each of the plurality of server devices 16(1)-16(n) is used to operatively couple and communicate between the plurality of server devices 16(1)-16(n), the web content server 14, and the plurality of client computing devices 12(1)-12(n) via the communication network 30.

[0026] Although embodiments web content server 14, the plurality of client computing devices 12(1)-12(n), and the plurality of server devices 16(1)-16(n) are described and illustrated herein, each of the web content server 14, the plurality of client computing devices 12(1)-12(n), and the plurality of server devices 16(1)-16(n) can be implemented on any suitable computer apparatus or computing device. It is to be understood that the apparatuses and devices of the embodiments described herein are for exemplary purposes, as many variations of the specific hardware and software used to implement the embodiments are possible, as will be appreciated by those skilled in the relevant art(s).

[0027] Furthermore, each of the devices of the embodiments may be conveniently implemented using one or more general purpose computers, microprocessors, digital signal processors, and micro-controllers, programmed according to the teachings of the

embodiments, as described and illustrated herein, and as will be appreciated by those ordinary skill in the art.

[0028] In addition, two or more computing apparatuses or devices can be substituted for any one of the devices in any embodiment described herein. Accordingly, principles and advantages of distributed processing, such as redundancy and replication also can be implemented, as desired, to increase the robustness and performance of the devices of the embodiments. The embodiments may also be implemented on computer apparatuses or devices that extend across any suitable network using any suitable interface mechanisms and communications technologies, including by way of example only telecommunications in any suitable form (e.g., voice and modem), wireless communications media, wireless communications networks, cellular communications networks, G3 communications networks, Public Switched Telephone Network (PSTNs), Packet Data Networks (PDNs), the Internet, intranets, and combinations thereof.

[0029] The embodiments may also be embodied as one or more non-transitory computer readable medium having instructions stored thereon for one or more aspects of the present invention as described and illustrated by way of the embodiments herein, as described herein, which when executed by a processor, cause the processor to carry out the steps necessary to implement the methods of the embodiments, as described and illustrated herein.

[0030] An exemplary method for transforming templates designed for the server side execution to templates that are suitable to be executed on the client side will now be described with reference to FIGS. 1-7.

[0031] The exemplary method begins at step 305 where the web content server 14 receives a request for a web page from one of the plurality of client computing devices 12(1)-12(n). In this example, the requested web page is a hypertext markup language (HTML) or extensible markup language template of web page stored by one of the plurality of server devices 16(1)-16(n).

[0032] Upon receipt of the request from one of the plurality of client computing devices 12(1)-12(n), in step 310 the web content server 14 obtains the template of the requested web page. In this example, the web content server 14 obtains the requested web page by retrieving the template of requested web page from the one of the plurality of server devices 16(1)-16(n) on behalf of the requesting one of the plurality of client computing devices 12(1)-12(n). By way of example only, the template of the requested webpage that is obtained by the web content server 14 is illustrated in FIG. 4. Additionally, another example of the server side template obtained by the web content server 14 is illustrated in FIG. 5. As illustrated in FIGS. 4-5, the template (server side template) of the requested webpage obtained by the web content server 14 includes full support of HTML tags and support of special tags that are able to perform some special operations, such as iteration over a collection of items; evaluation of logical conditions; set of variables; and print of variables, although the server side template can include other types and/or amounts of information.

[0033] In step 315, the web content server 14 parses the obtained server side template and generates a document object model (DOM) based on the parsed content.

[0034] In step 320, the web content server 14 traverses through each line in the DOM to identify: the types of HTML instructions, such as static HTML instructions; HTML tags including any special tags, condition tag, iteration tag; variable print instructions; and variable set instructions, although the web content server 14 can identify other types and/or amounts of instructions or other information. By way of example only, line 402 in FIG. 4 illustrates the condition tag and line 502 of FIG. 5 illustrates the iteration tag that is identified by the web content server 14.

[0035] In step 325, the web content server 14 generates JavaScript code representing the exact same procedure illustrated in the HTML instructions in the DOM as the client side template for each identified type of HTML instruction. By way of example only, the web content server 14 generates the JavaScript code 602 illustrated in FIG. 6 for the condition tag in the HTML instruction illustrated in line 402 of FIG. 4. In

this example, when the web content server 14 encounters a condition tag while traversing the DOM, it generates a JavaScript fragment that illustrates the condition and injects the inner content inside the condition tag as illustrated in line 602 of FIG. 6. Additionally, the web content server 14 generates a JavaScript code illustrated in FIG. 7 for the iteration tag illustrated in line 502 of FIG. 5. In this example, when the web content server 14 encounters an iteration tag while traversing the DOM, it invokes a specifically crafted utility function passing the whole iteration tag as a parameter. The function will take care of interpreting the iteration code and transform it into a JavaScript iterator that will also evaluate all the inner content as illustrated in FIG. 7, by way of example only. The dynamic HTML based template extends the basic HTML instructions by adding logical commands such as conditions, iterations and transformations. The result of the server side interpretation of the HTML template is an HTML page that differs in the content by contextual factors.

[0036] In step 330, upon transforming each identified type of HTML instruction to an equivalent JavaScript code, the web content server 14 transforms the server side template to a client side template by injecting the generated JavaScript code in the obtained server side template, although the web content server 14 can inject other types and/or amounts of information. In this example, the resulting client side template includes a number of utility JavaScript functions that can receive parameters. By generating the client side template using this technique, this example of the technology transforms a server side template into an autonomous JavaScript based client side template, without the need of an intermediate template language to be reinterpreted on the browser of the requesting one of the plurality of client computing devices 12(1)-12(n).

[0037] In step 335, the web content server 14 forwards the client side template to the requesting one of the plurality of client computing devices 12(1)-12(n) responsive to the received request for the webpage. Upon receipt of the client side template, the web browser in the requesting one of the plurality of client computing devices 12(1)-12(n) may execute the client side template resulting in the rendering of the requested webpage. While the web browser of the requesting one of the plurality of client computing devices

12(1)-12(n) executes the client side template, at a certain stage of the flow in the execution of the client side template will trigger the JavaScript code that will retrieve data in JavaScript Object Notion (JSON) format from one of the plurality of server devices 16(1)-16(n) hosting the requested webpage, although the data can be obtained in other formats. This example of the method ends in step 340.

[0038] Accordingly, as illustrated and described by way of the examples herein this technology effectively transforms templates designed for the server side execution to templates that are suitable to be executed on the client side. Additionally, this technology allows developers to use one language for both developing server side pages and client side content. Further with this technology, the server side templates can be transformed to client side templates that allow the creation of complex web pages where the displayed data can change without the page being reloaded. Even further, with this technology by directly transforming the server side template to a client side template, an autonomous JavaScript based template is created for the client side without the need of an intermediate template language to be reinterpreted on the browser of the client device.

[0039] Having thus described the basic concept of the invention, it will be rather apparent to those skilled in the art that the foregoing detailed disclosure is intended to be presented by way of example only, and is not limiting. Various alterations, improvements, and modifications will occur and are intended to those skilled in the art, though not expressly stated herein. These alterations, improvements, and modifications are intended to be suggested hereby, and are within the spirit and scope of the invention. Additionally, the recited order of processing elements or sequences, or the use of numbers, letters, or other designations therefore, is not intended to limit the claimed processes to any order except as may be specified in the claims. Accordingly, the invention is limited only by the following claims and equivalents thereto.

CLAIMS

What is claimed is:

1. A method for transforming server side template to a client side template, the method comprising:

obtaining, by a web content server, a server side template comprising a plurality of Hypertext Markup Language (HTML) and logical instructions responsive to a request for webpage received from a client device;

parsing, by the web content server, the obtained server side template to generate a document object model;

traversing, by the web content server, the generated document object model to identify the plurality of HTML and logical instructions; and

transforming, by the web content server, the obtained server side template to a client side template by replacing each of the identified plurality of HTML and logical instructions during the traversing with an equivalent JavaScript code.

2. The method as set forth in claim 1 wherein traversing further comprises identifying, by the web content server, one or more instruction types in the identified plurality of HTML and logical instructions, wherein the one or more instruction types comprises a condition tag, an iteration tag, a variable print instruction and a variable set instruction.

3. The method as set forth in claim 2 further comprising generating, by the web content server, the equivalent JavaScript code configured to express a condition and inject an inner content of the condition inside a branch of the condition when the identified one or more instruction types is the condition tag.

4. The method as set forth in claim 2 further comprising generating, by the web content server, the equivalent JavaScript code configured to invoke a

specifically crafted utility function to pass the iteration tag as a parameter when the identified one or more instruction types is the iteration tag.

5. The method as set forth in claim 2 further comprising transforming, by the web content server, the variable print instruction into a JavaScript print instruction when the identified one or more instruction types is the variable print instruction.

6. The method as set forth in claim 2 further comprising transforming, by the web content server, the variable set instruction into a JavaScript variable set instruction when the identified one or more instruction types is the variable set instruction.

7. A non-transitory computer readable medium having stored thereon instructions for transforming server side template to a client side template comprising machine executable code which when executed by a processor, causes the processor to perform steps comprising:

obtaining a server side template comprising a plurality of Hypertext Markup Language (HTML) and logical instructions responsive to a request for webpage received from a client device;

parsing the obtained server side template to generate a document object model;

traversing the generated document object model to identify the plurality of HTML and logical instructions; and

transforming the obtained server side template to a client side template by replacing each of the identified plurality of HTML and logical instructions during the traversing with an equivalent JavaScript code.

8. The medium as set forth in claim 7 wherein traversing further comprises identifying one or more instruction types in the identified plurality of HTML

and logical instructions, wherein the one or more instruction types comprises a condition tag, an iteration tag, a variable print instruction and a variable set instruction.

9. The medium as set forth in claim 8 further comprising generating the equivalent JavaScript code configured to express a condition and inject an inner content of the condition inside a branch of the condition when the identified one or more instruction types is the condition tag.

10. The medium as set forth in claim 8 further comprising generating the equivalent JavaScript code configured to invoke a specifically crafted utility function to pass the iteration tag as a parameter when the identified one or more instruction types is the iteration tag.

11. The medium as set forth in claim 8 further comprising transforming the variable print instruction into a JavaScript print instruction when the identified one or more instruction types is the variable print instruction.

12. The medium as set forth in claim 8 further comprising transforming the variable set instruction into a JavaScript variable set instruction when the identified one or more instruction types is the variable set instruction.

13. A web content server, comprising:

a processor;

a memory, wherein the memory coupled to the processor which are configured to execute programmed instructions stored in the memory comprising:

obtaining a server side template comprising a plurality of Hypertext Markup Language (HTML) and logical instructions responsive to a request for webpage received from a client device;

parsing the obtained server side template to generate a document object model;

traversing the generated document object model to identify the plurality of HTML and logical instructions; and

transforming the obtained server side template to a client side template by replacing each of the identified plurality of HTML and logical instructions during the traversing with an equivalent JavaScript code.

14. The device as set forth in claim 13 wherein the processor is further configured to execute programmed instructions stored in the memory for the traversing further comprises identifying one or more instruction types in the identified plurality of HTML and logical instructions, wherein the one or more instruction types comprises a condition tag, an iteration tag, a variable print instruction and a variable set instruction.

15. The device as set forth in claim 14 wherein the processor is further configured to execute programmed instructions stored in the memory further comprising generating the equivalent JavaScript code configured to express a condition and inject an inner content of the condition inside a branch of the condition when the identified one or more instruction types is the condition tag.

16. The device as set forth in claim 14 wherein the processor is further configured to execute programmed instructions stored in the memory further comprising generating the equivalent JavaScript code configured to invoke a specifically crafted utility function to pass the iteration tag as a parameter when the identified one or more instruction types is the iteration tag.

17. The device as set forth in claim 14 wherein the processor is further configured to execute programmed instructions stored in the memory further comprising transforming the variable print instruction into a JavaScript print instruction when the identified one or more instruction types is the variable print instruction.

18. The device as set forth in claim 14 wherein the processor is further configured to execute programmed instructions stored in the memory further comprising transforming the variable set instruction into a JavaScript variable set instruction when the identified one or more instruction types is the variable set instruction.

5

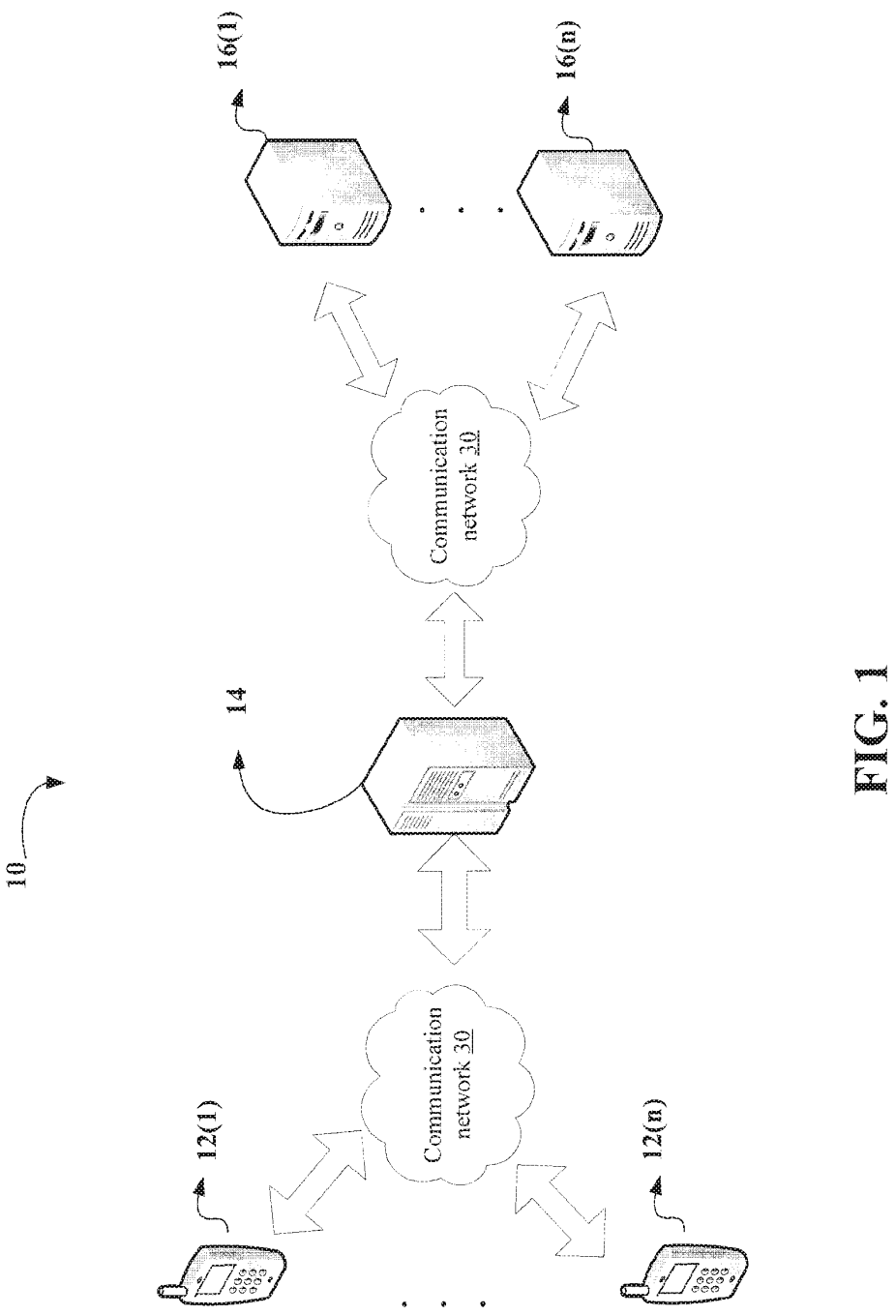

16(1)
16(n)
Communication network 30
14
10
Communication network 30
12(1)
12(n)

FIG. 1

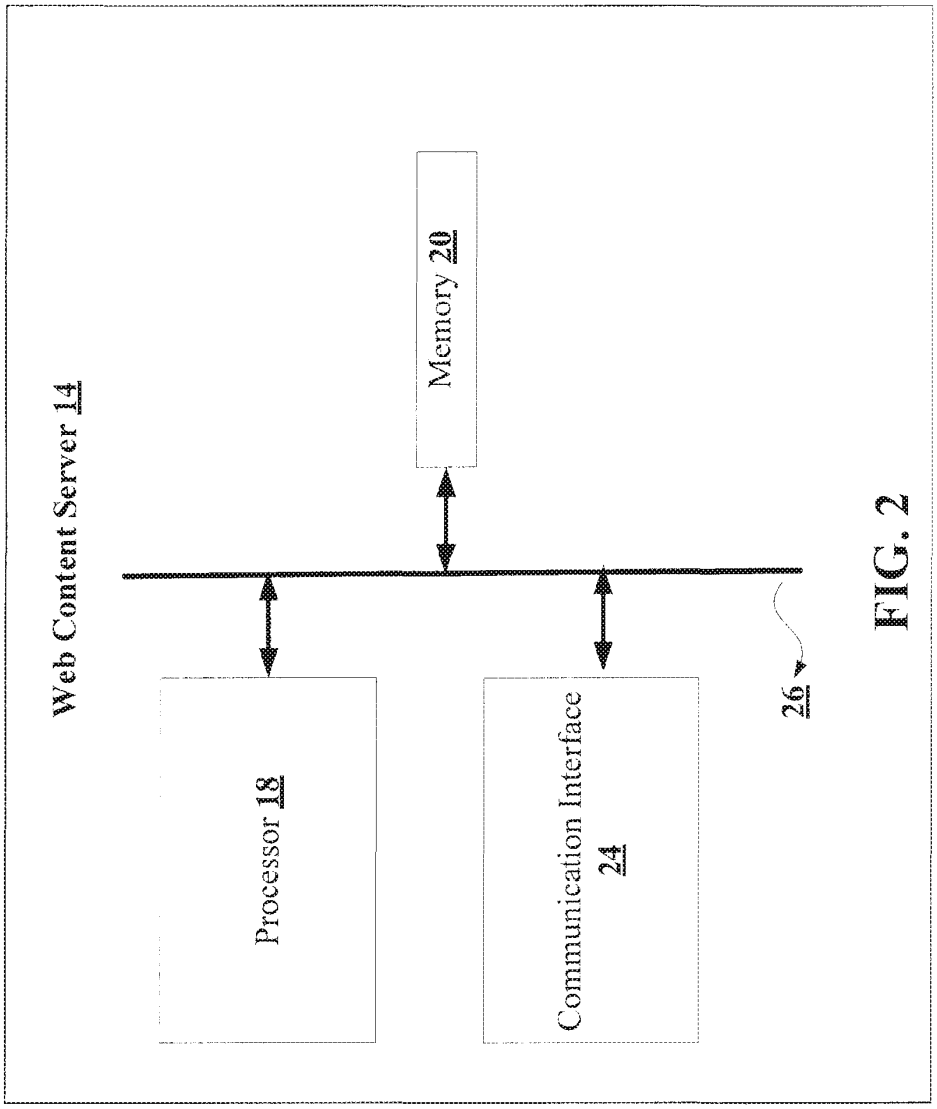

Web Content Server 14
Memory 20
Processor 18
Communication Interface 24
26

FIG. 2

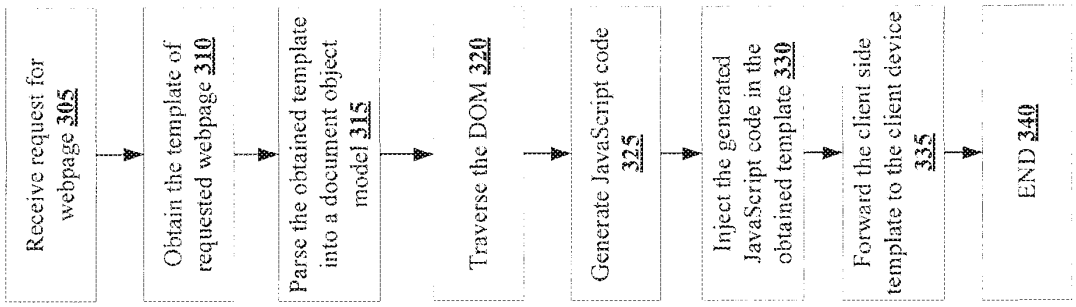

Receive request for webpage 305
Obtain the template of requested webpage 310
Parse the obtained template into a document object model 315
Traverse the DOM 320
Generate JavaScript code 325
Inject the generated JavaScript code in the obtained template 330
Forward the client side template to the client device 335
END 340

FIG. 3

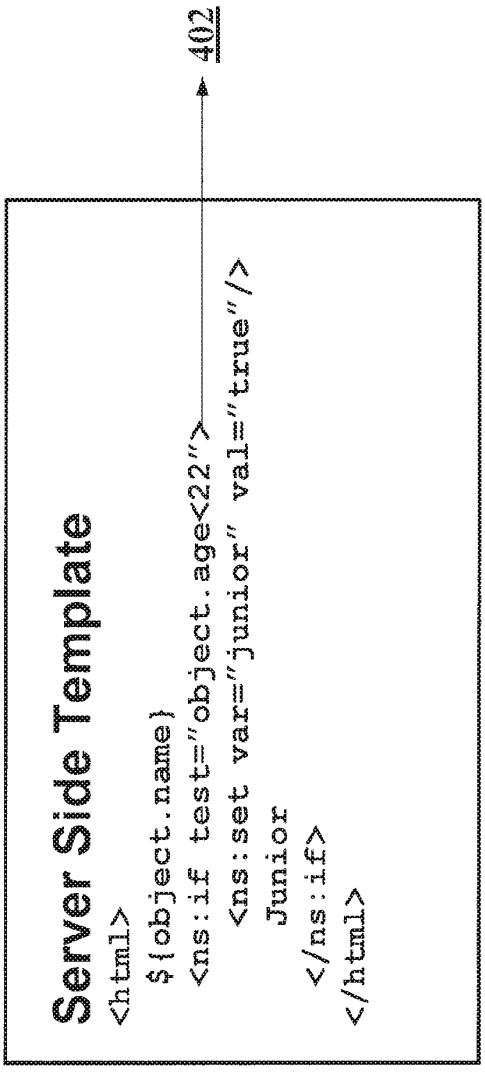
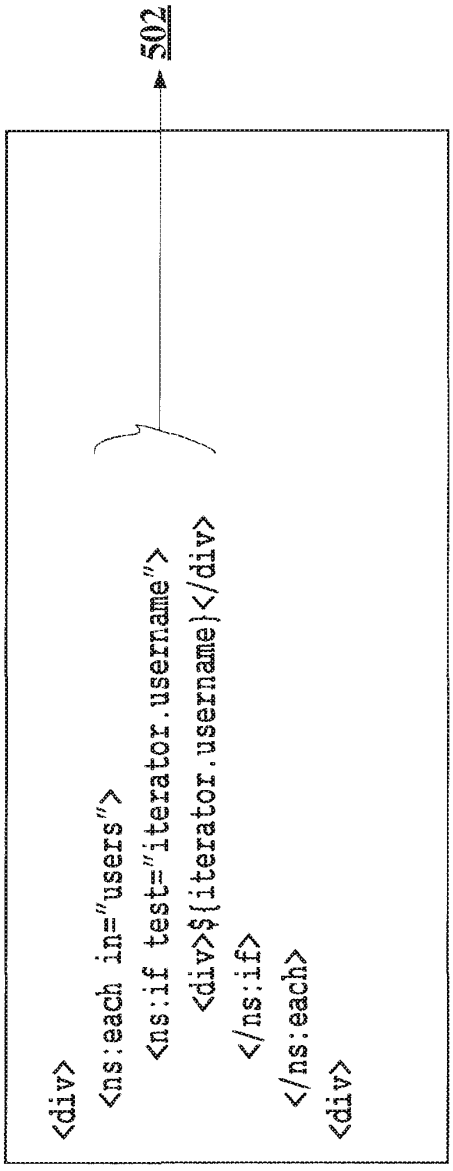

Server Side Template
<html>
${object.name}
<ns:if test="object.age<22">
<ns:set var="junior" val="true"/>
Junior
</ns:if>
</html>
402

FIG. 4


502

<ns:each in="users">
<ns:if test="iterator.username">
${iterator.username}
</ns:if>
</ns:each>

FIG. 5

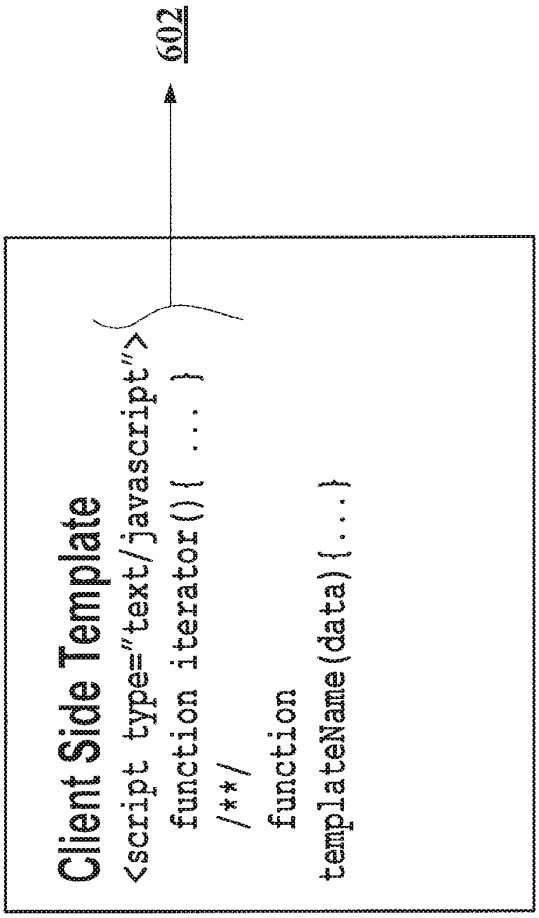

602
Client Side Template
<script type="text/javascript">
function iterator(){ ... }
/**/
function
templateName(data){...}

FIG. 6

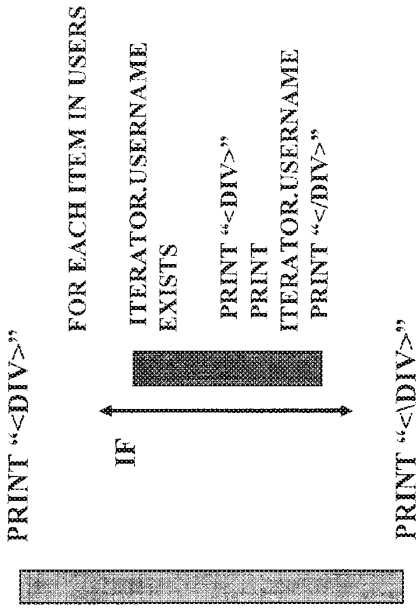
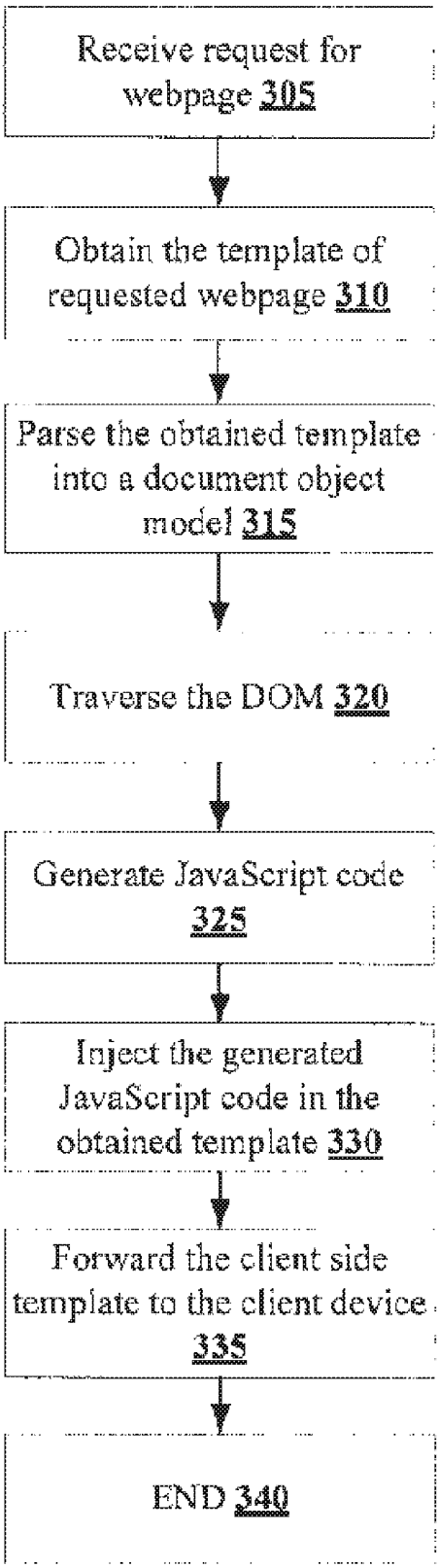

PRINT "<DIV>"
FOR EACH ITEM IN USERS
IF
ITERATOR.USERNAME
EXISTS
PRINT "<DIV>"
PRINT
ITERATOR.USERNAME
PRINT "</DIV>"
PRINT "</DIV>"

FIG. 7


Receive request for webpage 305
Obtain the template of requested webpage 310
Parse the obtained template into a document object model 315
Traverse the DOM 320
Generate JavaScript code 325
Inject the generated JavaScript code in the obtained template 330
Forward the client side template to the client device 335
END 340

Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2947402 A1 2015/11/12

(21) 2 947 402

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(86) Date de dépôt PCT/PCT Filing Date: 2015/04/02
(87) Date publication PCT/PCT Publication Date: 2015/11/12
(85) Entrée phase nationale/National Entry: 2016/10/28
(86) N° demande PCT/PCT Application No.: US 2015/024172
(87) N° publication PCT/PCT Publication No.: 2015/171228
(30) Priorité/Priority: 2014/05/05 (US61/988,639)

(51) Cl.Int./Int.Cl. *G06F 9/445* (2006.01), *G06F 3/0484* (2013.01)

(71) Demandeur/Applicant:
USABLENET INC., US

(72) Inventeur/Inventor:
SCODA, ENRICO, IT

(74) Agent: PARLEE MCLAWS LLP

(54) Titre : PROCEDES POUR FACILITER UNE INTERFACE A DISTANCE ET DISPOSITIFS ASSOCIES
(54) Title: METHODS FOR FACILITATING A REMOTE INTERFACE AND DEVICES THEREOF

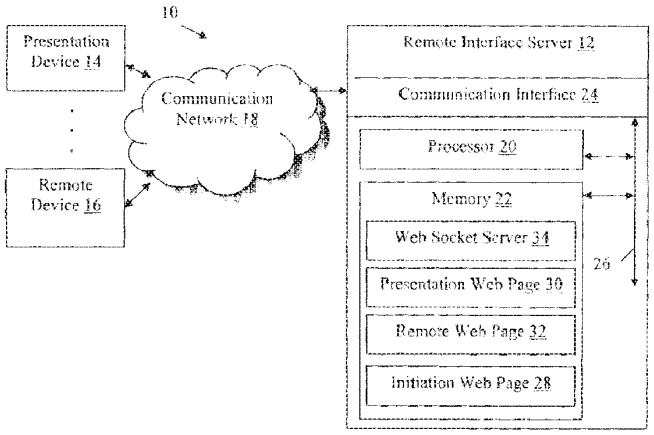


FIG. 1

(57) Abrégé/Abstract:
A method, non-transitory computer readable medium, remote interface server computing device, and system that provides a presentation web page to a presentation device and a remote web page to a remote device. The remote web page is configured to, when executed by the remote device, register the remote device as associated with the presentation device and render a swipe panel on a display of the remote device. A first message is received from the remote device in response to an interaction with the swipe panel. A second message is sent to the presentation device in response to receiving the first message. The second message, when executed by the presentation device, is configured to cause the presentation device to perform an action on the presentation web page corresponding to the user interaction with the swipe panel.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
12 November 2015 (12.11.2015)

WIPO | PCT

(10) International Publication Number
WO 2015/171228 A1

(51) International Patent Classification:
G06F 9/445 (2006.01)

(21) International Application Number:
PCT/US2015/024172

(22) International Filing Date:
2 April 2015 (02.04.2015)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/988,639 5 May 2014 (05.05.2014) US

(71) Applicant: **USABLENET INC.** [US/US]; 142 W. 57th Street, 7th Floor, New York, NY 10019 (US).

(72) Inventor: **SCODA, Enrico**; Via Cividina 416/3, 33035 Martignacco Ud (IT).

(74) Agents: **GALLO, Nicholas, J.** et al.; LeClairRyan, 70 Linden Oaks, Suite 210, Rochester, NY 14625 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

---- *with international search report (Art. 21(3))*

(54) Title: METHODS FOR FACILITATING A REMOTE INTERFACE AND DEVICES THEREOF

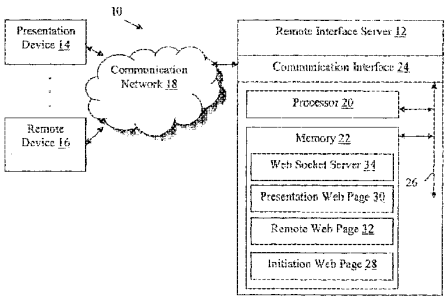


FIG. 1

(57) Abstract: A method, non-transitory computer readable medium, remote interface server computing device, and system that provides a presentation web page to a presentation device and a remote web page to a remote device. The remote web page is configured to, when executed by the remote device, register the remote device as associated with the presentation device and render a swipe panel on a display of the remote device. A first message is received from the remote device in response to an interaction with the swipe panel. A second message is sent to the presentation device in response to receiving the first message. The second message, when executed by the presentation device, is configured to cause the presentation device to perform an action on the presentation web page corresponding to the user interaction with the swipe panel.

WO 2015/171228 A1

METHODS FOR FACILITATING A REMOTE INTERFACE AND DEVICES THEREOF

[0001] This application claims the benefit of U.S. Provisional Patent Application Serial No. 61/988,639 filed on May 5, 2014, which is hereby incorporated by reference in its entirety.

FIELD

[0002] This technology generally relates to kiosk and other presentation devices, and more particularly to methods and devices for facilitating a remote interface for interacting with such presentation devices.

BACKGROUND

[0003] Presentation devices, such as kiosks and other devices with relatively large screen sizes, are often available for interaction in commercial and other settings. Presentation devices can display product information associated with a catalog of available products for a retailer, for example, advertising information, or any other information directed to consumers or other members of the public.

[0004] The method of interaction with presentation devices is often through a multi-touch screen. However, such presentation devices are generally complex and have relatively high associated cost due to the multi-touch screens and required processing power. Additionally, presentation devices with relatively large screen sizes are currently unable to effectively present, and/or allow users to input, private information (e.g. personally identifiable information or credit card numbers) in a discreet manner.

SUMMARY

[0005] A method for facilitating a remote interface includes providing, by a remote interface server computing device, a presentation web page to a presentation device and a remote web page to a remote device. The remote web

page is configured to, when executed by the remote device, register the remote device as associated with the presentation device and render a swipe panel on a display of the remote device. A first message is received, by the remote interface server computing device, from the remote device in response to an interaction with the swipe panel. A second message is sent, by the remote interface server computing device, to the presentation device in response to receiving the first message. The second message, when executed by the presentation device, is configured to cause the presentation device to perform an action on the presentation web page corresponding to the user interaction with the swipe panel.

[0006] A non-transitory computer readable medium having stored thereon instructions for facilitating a remote interface comprising executable code which when executed by a processor, causes the processor to perform steps including providing a presentation web page to a presentation device and a remote web page to a remote device. The remote web page is configured to, when executed by the remote device, register the remote device as associated with the presentation device and render a swipe panel on a display of the remote device. A first message is received from the remote device in response to an interaction with the swipe panel. A second message is sent to the presentation device in response to receiving the first message. The second message, when executed by the presentation device, is configured to cause the presentation device to perform an action on the presentation web page corresponding to the user interaction with the swipe panel.

[0007] A remote interface server computing device including a processor and a memory coupled to the processor which is configured to be capable of executing programmed instructions comprising and stored in the memory to provide a presentation web page to a presentation device and a remote web page to a remote device. The remote web page is configured to, when executed by the remote device, register the remote device as associated with the presentation device and render a swipe panel on a display of the remote device. A first message is received from the remote device in response to an interaction with the swipe panel. A second message is sent to the presentation device in response to receiving the first message. The second message, when executed by the

presentation device, is configured to cause the presentation device to perform an action on the presentation web page corresponding to the user interaction with the swipe panel.

[0008] A system for facilitating a remote interface includes a remote interface server computing device including a first processor and a first memory coupled to the first processor. The first process is configured to be capable of executing programmed instructions comprising and stored in the first memory to provide a presentation web page to a presentation device and a remote web page to a remote device. The remote web page configured to, when executed by the remote device, register the remote device as associated with the presentation device and render at least a swipe panel on a display of the remote device. The system further includes a web socket server computing device including a second processor and a second memory coupled to the second processor. The second processor is configured to be capable of executing programmed instructions comprising and stored in the second memory to receive a first message from the remote device in response to a user interaction with the swipe panel. A second message is sent to the presentation device in response to receiving the first message. The second message, when executed by the presentation device, is configured to cause the presentation device to perform an action on the presentation web page corresponding to the user interaction with the swipe panel.

[0009] This technology provides a number of advantages including providing methods, non-transitory computer readable media, devices, and systems that facilitate remote interfaces for presentation devices. With this technology, presentation device (e.g., a kiosk) can be seamlessly controlled by a remote device (e.g., mobile phones) using messages exchanged based on the web socket protocol. By leveraging remote devices, this technology allows presentation devices to be less complex and less costly. Additionally, private information can be advantageously submitted without displaying the information in a visible format on the display of a presentation device

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] FIG. 1 is a block diagram of a network environment with an exemplary remote interface server coupled to a presentation device and a remote device;

[0011] FIG. 2 is a block diagram of another network environment with an exemplary remote interface server coupled to a presentation device, a remote device, and a web socket server;

[0012] FIG. 3 is a flowchart of an exemplary method for facilitating a remote interface;

[0013] FIG. 4 is an exemplary initiation web page;

[0014] FIG. 5 is an exemplary presentation web page;

[0015] FIG. 6 is an exemplary remote web page;

[0016] FIG. 7 is an exemplary presentation web page;

[0017] FIG. 8 is an exemplary remote web page modified according to a horizontal swipe gesture with a swipe panel;

[0018] FIG. 9 is an exemplary presentation web page with input fields;

[0019] FIG. 10 is an exemplary remote web page with input fields;

[0020] FIG. 11 is an exemplary remote web page subsequent to user interaction with an edit button;

[0021] FIG. 12 is an exemplary remote web page with a virtual keyboard;

[0022] FIG. 13 is an exemplary presentation web page with input fields subsequent to a user editing content;

[0023] FIG. 14 is an exemplary presentation web page subsequent to user interaction with a play video button of a remote web page; and

[0024] FIG. 15 is an exemplary remote web page subsequent to user interaction with a play video button of the remote web page.

DETAILED DESCRIPTION

[0025] An exemplary network environment 10 with a remote interface server 12 coupled to a presentation device 14 and a remote device 16 is illustrated in FIG. 1. In this example, the remote interface server 12, presentation device 14, and remote device 16 are coupled together by at least one communication network 18, although other numbers and types of systems, devices, and/or elements in other configurations or network topologies can also be used. This technology provides a number of advantages including methods, non-transitory computer readable media, devices, and systems that facilitate a remote interfaces to effectively replicate, on the presentation device 14, user interactions with a web page rendered on the remote device 16 while maintaining information privacy.

[0026] The remote interface server 12 (also referred to herein as a remote interface server computing device) in this particular example is coupled to the presentation device 14 and the remote device 16 by the communication network 18 which can include one or more local area network(s) (LANs) and/or wide area network(s) (WANs). Other network devices configured to generate, send, and receive network communications and coupled together via other topologies can also be used. While not shown, the network environment 10 also may include additional network components, such as routers, switches and other devices, which are well known to those of ordinary skill in the art and thus will not be described here.

[0027] The remote interface server 12 may perform any number of functions including hosting and providing web content and facilitating communications between the presentation device 14 and the remote device 16 according to the web socket protocol, for example. In this example, the remote interface server 12 includes a processor 20, a memory 22, and a communication interface 24, which are coupled together by a bus 24 or other communication link,

although other numbers and types of components, parts, devices, systems, and elements in other configurations and locations can be used.

[0028] The processor 20 in the remote interface server 12 executes a program of stored instructions for one or more aspects of this technology, as described and illustrated by way of the embodiments herein, although the processor 20 could execute other numbers and types of programmed instructions. The processor 20 of the remote interface server 12 may include one or more central processing units or general purpose processors with one or more processing cores, for example.

[0029] The memory 24 in the remote interface server 12 stores these programmed instructions for one or more aspects of this technology, as described and illustrated herein, although some or all of the programmed instructions could be stored and/or executed elsewhere. Optionally, the memory 24 in this example stores a plurality of web pages including at least one initiation web page 28, presentation web page 30, and remote web page 32, as described and illustrated in more detail later. A variety of different types of memory storage devices, such as a random access memory (RAM), read only memory (ROM), hard disk drive(s), flash, solid state drive(s), or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to the processor, can be used for the memory 22 in the remote interface server 12.

[0030] In this particular example, the memory 24 also includes a web socket server 34. The web socket server 34 in this example is a software module that includes programmed instructions that, when executed by the processor, generate a web socket server configured to facilitate communications between the presentation device 14 and the remote device 16 according to the web socket protocol, as described and illustrated in more detail later.

[0031] The communication interface 24 in the remote interface server 12 is used to operatively couple and communicate between the remote interface server 12, the presentation device 14, and the remote device 16, which are all

coupled together via the communication network 18, although other types and numbers of communication networks or systems with other types and numbers of connections and configurations to other devices and elements can also be used. By way of example only, the communication network 18 can use TCP/IP over Ethernet and industry-standard protocols, including hypertext transfer protocol (HTTP), and/or secure HTTP (HTTPS), although other types and numbers of communication networks, such as a direct connection, modems and phone lines, e-mail, and wireless and hardwire communication technology, each having their own communications protocols, can be used.

[0032] The presentation device 14 and the remote device 16 in this particular example enable a user to request, receive, and interact with applications, web services, and content hosted by the remote interface server 12 using the communication network 18, although one or more of the presentation device 14 or remote device 16 could access content and utilize other types and numbers of applications from other sources and could provide a wide variety of other functions for the user.

[0033] Each of the presentation device 14 and remote device 16 in this example includes a processor, a memory, an input device, a display device, and a communication interface, which are coupled together by a bus or other communication link, although one or more of presentation device 14 or remote device 16 can include other numbers and types of components, parts, devices, systems, and elements in other configurations. The processor in each of the presentation device 14 and remote device 16 can execute a program of instructions stored in the memory the client device for one or more aspects of this technology, as described and illustrated herein, although the processor could execute other numbers and types of programmed instructions.

[0034] The input device in each of the presentation device 14 and remote device 16 can be used to input selections, such as a request for a particular web page or other content stored by the remote interface server 12 or another web content server, although the input device could be used to input other types of requests and data and interact with other elements. The input device can include

keypads, touch screens, and/or vocal input processing systems, although other types and numbers of input devices can also be used.

[0035] The display device in each of the presentation device 14 and remote device 16 can be used to show data and information to a user, such as web pages and other content retrieved from the remote interface server 12 or another web content server by way of example only. The display device in the presentation device 14 can be a television screen and the display device in the remote device 16 can be a mobile phone screen, for example, although other types and numbers of display devices could be used depending on the particular type of presentation device 14 and remote device 16. The communication interface in each of the presentation device 14 and remote device 16 can be used to operatively couple and communicate between the presentation device 14, remote device 16, and remote interface server 12 over the communication network 18.

[0036] By way of example only, the presentation device 14 can be relatively less mobile than the remote device 16 and can include a television, kiosk, or other device with a relatively large display as compared to that of the remote device 16, although other types of presentation devices can also be used. Accordingly, in some examples, the remote device 16 is relatively more mobile than the presentation device 14 and can be a smartphone, personal digital assistant, tablet, netbook, notebook, or other device with a relatively small display as compared to that of the presentation device 14, although other types of remote devices can also be used.

[0037] Referring more specifically to FIG. 2 another exemplary network environment 36 with a remote interface server 12 coupled to a presentation device 14, a remote device 16, and a web socket server 38 is illustrated. The remote interface server 12, presentation device 14, remote device 16, and communication network 18 in this example are the same as described and illustrated earlier with reference to FIG. 1 except that the remote interface server 12 does not includes the web socket server 24. Instead, in this particular example, the web socket server 38 is provided as a separate web socket server computing device in the environment 36 that is also configured to communicate with the presentation

device 14 and the remote device 16 via the communication network 18. Other network topologies and numbers of remote interface servers and/or web socket servers can also be provided in network environment 10 or 36.

[0038] The web socket server 38 in the particular example illustrated in FIG. 2 includes a processor 40, a memory 42, and a communication interface 44, which are coupled together by a bus 46 or other communication link, although other numbers and types of components, parts, devices, systems, and elements in other configurations and locations can be used. The processor 40 in the web socket server 38 executes a program of stored instructions one or more aspects of this technology, as described and illustrated by way of the embodiments herein, although the processor 40 could execute other numbers and types of programmed instructions. The processor 40 of the web socket server 38 may include one or more central processing units or general purpose processors with one or more processing cores, for example.

[0039] The memory 42 in the web socket server 38 stores these programmed instructions for one or more aspects of this technology, as described and illustrated herein, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM), read only memory (ROM), hard disk drive(s), flash, solid state drive(s), or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to the processor 40, can be used for the memory 42 in the web socket server 38.

[0040] The communication interface 44 in the web socket server 38 is used to operatively couple and communicate between the web socket server 38, the presentation device 14, and the remote device 16, which are all coupled together via the communication network 18, although other types and numbers of communication networks or systems with other types and numbers of connections and configurations to other devices and elements can also be used. By way of example only, the communication network 18 can use TCP/IP over Ethernet and industry-standard protocols, including hypertext transfer protocol (HTTP) and the

web socket protocol, although other types and numbers of communication networks, such as a direct connection, modems and phone lines, e-mail, and wireless and hardwire communication technology, each having their own communications protocols, can also be used.

[0041] The embodiments of the remote interface server 12, web socket server 38, presentation device 14, and remote device 16 are described and illustrated herein for exemplary purposes and many variations of the specific hardware and software used to implement the embodiments are possible, as will be appreciated by those skilled in the relevant art(s). Furthermore, each of the devices of the embodiments may be conveniently implemented using one or more general purpose computers, microprocessors, digital signal processors, and micro-controllers, programmed according to the teachings of the embodiments, as described and illustrated herein, and as will be appreciated by those ordinary skill in the art.

[0042] In addition, two or more computing apparatuses or devices can be substituted for any one of the devices in any embodiment described herein. Accordingly, principles and advantages of distributed processing, such as redundancy and replication also can be implemented, as desired, to increase the robustness and performance of the devices of the embodiments. The embodiments may also be implemented on computer apparatuses or devices that extend across any suitable network using any suitable interface mechanisms and communications technologies, including by way of example only telecommunications in any suitable form (e.g., voice and modem), wireless communications media, wireless communications networks, cellular communications networks, G3 communications networks, Public Switched Telephone Network (PSTNs), Packet Data Networks (PDNs), the Internet, intranets, and combinations thereof.

[0043] The embodiments may also be embodied as one or more non-transitory computer readable media having instructions stored thereon for one or more aspects of this technology as described and illustrated by way of the embodiments herein, as described herein, which when executed by a processor,

cause the processor to carry out the steps necessary to implement the methods of the embodiments, as described and illustrated herein.

[0044] An exemplary method for facilitating a remote interface will now be described with reference to FIGS. 1-15. Referring more specifically to FIG. 3, in step 300 in this example, the remote interface server 12 sends the initiation web page 28 to the presentation device 14 and registers the presentation device 14 with the web socket server 34 or 38. The remote interface server 12 can send the initiation web page 28 in response to a request for the initiation web page 28 received from the presentation device 14. In one example, the presentation device 14 is a smart television executing a web browser which facilitates the retrieval of the initiation web page 28 at the request of a user, although other types of presentation devices and other methods of providing the initiation web page 28 can also be used. Upon receipt of the initiation web page 28, the presentation device 14 executes JavaScript code included with the initiation web page 28, which is configured to communicate with the remote interface server 12 to register the presentation device 12 by establishing a connection between the presentation device and the web socket server 34 or 38.

[0045] In step 302, the remote interface server 12 sends a remote web page 32 to the remote device 16, registers the remote device 16 with the web socket server 34 or 38 as associated with the presentation device 14, and sends a presentation web page 30 to the presentation device 14. The remote web page 32 and presentation web page 30 can be sent by the remote interface server 12 in response to a request from the remote device 16 initiated based on an interaction by the remote device 16 with at least a portion of the initiation web page 28 rendered on the display of the presentation device 14.

[0046] Referring more specifically to FIG. 4, an exemplary initiation web page 28 is illustrated. In this example, the initiation web page 28 includes a portion with an interactive mode interface 400, which is a three dimensional bar code in this example, although other types of interactive mode interfaces and portions of the initiation web page 28 can also be used.

[0047] Accordingly, a user of the remote device 16 in this example can scan the interactive mode interface 400 which encodes at least a Uniform Resource Locator (URL) and causes a web browser executed by the remote device 16 to request the remote web page 32 located at the URL from the remote interface server 12, which sends the remote web page 32 to the remote device 16 in response. In this example, the remote web page 32 is configured to, when executed by the web browser of the remote device 16, register the remote device 16 with the web socket server 34 or 38 as associated with the presentation device 14.

[0048] Accordingly, the remote web page 32 can include JavaScript code executed by the remote device 16 that facilitates communication by the remote device 16 with the web socket server 34 or 38 to establish a connection between the remote device 16 and the web socket server 34 or 38, as well as an association with the presentation device 14. Optionally, the interactive mode interface 400 of the initiation web page 28 can further encode an identifier of the presentation device 14 which can be used to facilitate the association of the presentation device 14 and the remote device 16 with the web socket server 34 or 38. Other methods of initiating the association of the presentation device 14 and the remote device 16 with the web socket server 34 or 38 can also be used.

[0049] In response to receipt of the request from the remote device 16 for the remote web page 32, or in response to a subsequent communication to the remote interface server 12 by the remote device 16 executing the JavaScript code of the remote web page 32, the remote interface server 12 also sends the presentation web page 30 to the presentation device 14. The remote web page 32 and presentation web page 30 can be different versions of a same web page such that the remote web page 32 includes at least a portion of the content of the presentation web page 30, although other types of presentation and remote web pages can also be used.

[0050] Referring more specifically to FIG. 5, an exemplary presentation web page 30 is illustrated and referring more specifically to FIG. 6, an exemplary remote web page 32 is illustrated. In this example, the presentation web page 30

and remote web page 32 are different versions of the same web page as the presentation web page includes multiple panels that can be manipulated, as described and illustrated in more detail later, whereas the remote web page 32 is a mobile version of the web page which includes content of only one of the panels included in the presentation web page 30. The remote web page 32 is also configured to, when executed by the remote device 16, render a swipe panel 600, and optionally one or more buttons, on the display of the remote device 16. In this example, the swipe panel 600 includes the content of the panel corresponding to one of the panels of the presentation web page 30, although the swipe panel 600 can be located elsewhere in the web page and/or display of the remote device 16.

[0051] Referring back to FIG. 3, in step 304 the web socket server 34 or 38 receives a message from the remote device 16 in response to a user interaction with the remote web page 32. The message can comply with the web socket protocol and can be received by the web socket server 34 or 38 using the connection established with the remote device 16. Accordingly, the JavaScript code of the remote web page 32 executed by the remote device 16 can determine when a user has interacted with the remote web page 32 and send a message to the web socket server 34 or 38 corresponding to the interaction in response. The message can include information regarding the type of interaction and any other contextual information, for example.

[0052] In step 306, the web socket server 34 or 38 determines whether the user interaction corresponding to the message received in step 304 is a swipe panel interaction, and optionally whether the interaction was a horizontal or vertical swipe gesture, for example. If the web socket server 34 or 38 determines that the user interaction is a swipe panel interaction, then the Yes branch is taken to step 308. In step 308, the web socket server 34 or 38 sends a message to the presentation device 14 to cause the presentation device 14 to perform an action on the presentation web page 30 corresponding to the swipe panel interaction.

[0053] Referring more specifically to FIG. 7, a presentation web page 30 subsequent to performing an action corresponding to a horizontal swipe gesture with the swipe panel 600 of the remote web page 32 is illustrated. In this example,

the panel with a video illustrated in the foreground in FIG. 5 has been rotated to the left, such as by three dimensional rotation, for example, resulting in the modified presentation web page 30 illustrated in FIG. 7 in which a new panel has been rotated to the foreground.

[0054] Referring more specifically to FIG. 8, a remote web page 32 modified according to a horizontal swipe gesture with the swipe panel 600 of the remote web page 32 is illustrated. In this example, the panel with the video illustrated in the swipe panel 600 in FIG. 6 has been moved off screen, such as by two dimensional slide animation, for example, resulting in the modified remote web page 32 illustrated in FIG. 8 in which a new panel with different content has replaced the previous panel in the swipe panel 600.

[0055] In other examples, the user interaction can be a vertical swipe gesture and the action can be a vertical scroll. For example, a user can perform a vertical swipe gesture on the swipe panel 600 of the remote web page illustrated in FIG. 8 resulting in a vertical scroll action on the presentation web page 30 illustrated in FIG. 7. Other exemplary gestures and interactions with the swipe panel 600 and corresponding actions, as well as animations and rotations can also be used. Accordingly, in this example, a user of the remote device 16 can contemporaneously control the display of the presentation device 14, and in particular the presentation web page 30, without physically interacting with the presentation device 14 and using only the interface provided on the remote device 16 through the remote web page 32.

[0056] Referring back to FIG. 3, if the web socket server 34 or 38 determines that the user interaction is not a swipe panel interaction in step 306, then the No branch is taken to step 310. In step 310, the web socket server 34 or 38 determines whether the user interaction corresponding to the message received in step 304 is a save button interaction. If the web socket server 34 or 38 determines that the user interaction is a save button interaction, then the Yes branch is taken to step 312.

[0057] Optionally, the buttons of the remote web page 32 as rendered on the display of the remote device 16 can change based on functionality present in the remote web page 32 and/or presentation web page 30. For example, referring back to FIG. 6, the remote web page 32 includes play button 602 corresponding to the video content rendered in the swipe panel 600. Referring more specifically to FIG. 9, a presentation web page 30 with input fields 900 is illustrated and referring more specifically to FIG. 10, a remote web page 32 with the input fields 1000 is illustrated. In this example, the buttons are modified by the JavaScript code of the remote web page 32 to include an edit button 1002 corresponding to the content of the input fields 1000 of the remote web page 32.

[0058] Referring more specifically to FIG. 11, the remote web page 32 of FIG. 10 is illustrated subsequent to user interaction with the edit button 1002. Upon user interaction with the edit button 1002, the remote web page 32 is configured to render a save button 1100 in place of the edit button 1002 as well as editable input fields 1000 corresponding to the input fields 900 of the presentation web page 30 of FIG. 9. In this example, private information such as a credit card number is optionally obfuscated in the presentation web page 30 since the presentation web page 30 is rendered on a presentation device 14 which may have a relatively large display and/or may be visible to the environment or other members of the public. However, the editable input fields 1000 rendered by the remote web page 32 in response to the user interaction with the edit button 1002 are rendered without the obfuscation to allow user editing.

[0059] Referring more specifically to FIG. 12, optionally, the remote device 16 is configured to display a virtual keyboard 1200 upon user selection of one of the editable input fields 1000 allowing the user to edit the information. In this example, the user has edited the name, credit card number, and CVV fields. Upon entering the new information, the user can select the save button 1100 as illustrated in FIG. 11. In response to user selection of the save button 1100, the message received by the web socket server 34 or 38 in step 304 of FIG. 3 is sent by the remote device 16 and includes at least any information updated by the user.

[0060] In step 312, the web socket server 34 or 38 sends a message to the presentation device 14 including information included in the message received from the remote device 16 in step 304. Referring more specifically to FIG. 13, the presentation web page 30 with input fields 900 subsequent to user editing of the content is illustrated. In this example, any private information continues to be rendered in an obfuscated manner in the presentation web page 30 rendered on the display of the presentation device 14.

[0061] Referring back to FIG. 3, if the web socket server 34 or 38 determines in step 310 that the user interaction is not a save button interaction, then the No branch is taken to step 314. In step 314, the web socket server 34 or 38 determines whether the user interaction corresponding to the message received in step 304 is a video button interaction. If the web socket server 34 or 38 determines that the user interaction is a video button interaction, then the Yes branch is taken to step 316.

[0062] In step 316, the web socket server 34 or 38 sends a message to the presentation device 14 corresponding to the video button interacted with by the user of the remote device 16. Referring back to FIG. 6, user interaction with the play button 602, for example, can cause a message to be sent to the web socket server 34 or 38 which, in step 316 of FIG. 3, sends a message to the presentation device 16 to initiate the video of the presentation web page 30 in response. Referring more specifically to FIG. 14, the presentation web page 30 subsequent to user interaction with the play button 602 of the remote web page 32 is illustrated.

[0063] In FIG. 15, the remote web page 32 subsequent to user interaction with the play button 602 of the remote web page 32 is illustrated. Optionally, in this example, the remote web page 32 is configured to convert the swipe panel 600 to indicate that the video is playing and to render a pause button 1500 in place of the play button 602, although the remote web page 32 can be configured to provide other functionality in response to the user interaction with the play button 602. In another example, user interaction with the pause button 1500 of the remote web page 32 can be determined in step 316, which can cause a message to

be sent to the presentation device 14 to pause the video of the presentation web page 30 in response. In yet other examples, a stop button can be rendered on the remote web page 32 and any other type of button can also be used.

[0064] Referring back to FIG. 3, in step 318, the web socket server 34 or 38 optionally determines whether a message is received from the presentation device 14 in response to the message sent to the presentation device 14 in step 316 in examples in which the video button interaction is a user interaction with a play button 602. Optionally, one or more callbacks can be received by the web socket server 34 or 38 from the presentation device 14 after any of the messages sent in the after any of steps 308, 312, or 316. However, in this example, the message received from the presentation device 14 in step 318 optionally includes video information (e.g., elapsed time) and/or a URL. The URL can correspond with content displayed in the video. For example, if the video is of a model on a runway at a fashion show, the URL can point to content including information regarding an article of the clothing worn by the model including associated cost and purchase information.

[0065] Accordingly, if the web socket server 34 or 38 determines that a message is received from the presentation device 14 in step 318, then the Yes branch is taken to step 320. In step 320, the web socket server 34 or 38 sends a message to the remote device 16 in response to the message received from the presentation device 14 in step 316. The message sent by the web socket server 34 or 38 in step 320 can include the video information and/or the URL included in the message received from the presentation device 14 in step 318. In response, the remote web page 32 can be configured to render the video information and/or content located at the URL on the display of the remote device 16, such as on the swipe panel 600 for example.

[0066] Referring back to step 318, if the web socket server determines a message is not received from the presentation device 14, then the No branch is taken back to step 304 and the web socket server 34 or 38 receives another message from the remote device 16 in response to a subsequent user interaction with the remote web page 32. Referring back to step 314, if the web socket server

34 or 38 determines that the user interaction is not a video button interaction, then the No branch is taken to step 322. In step 322, the web socket server 34 or 38 sends a message to the presentation device 14, if necessary, to reproduce activity associated with the remote web page 32.

[0067] Accordingly, while the swipe panel, save button, and video button user interactions have been described and illustrated earlier by way of example only, other interactions with the remote web page 32 are possible. In response to the user interactions, the remote web page 32 is configured to send a message to the web socket server 34 or 38, if necessary, which is configured to identify the associated presentation device 14 and send a corresponding message to the presentation device 14 to modify the presentation web page 30 accordingly. Thereby, a user of the remote device 16 interacting with the remote web page 32 can effectively control the presentation web page 30 rendered on the display of the presentation device 14.

[0068] Accordingly, with this technology, presentation devices can be seamlessly controlled by remote devices using messages exchanged based on the web socket protocol. As the remote devices are used to facilitate an interface, using specially programmed web pages and without any dedicated hardware or software, the presentation devices can be less complex and less costly and do not require multi-touch displays or any other physical interfaces. Additionally, private information can be advantageously submitted, such as with respect to facilitating product purchases, without displaying the information in a visible format on the display of the presentation device, which is visible publicly in many environments.

[0069] Having thus described the basic concept of the invention, it will be rather apparent to those skilled in the art that the foregoing detailed disclosure is intended to be presented by way of example only, and is not limiting. Various alterations, improvements, and modifications will occur and are intended to those skilled in the art, though not expressly stated herein. These alterations, improvements, and modifications are intended to be suggested hereby, and are within the spirit and scope of the invention. Additionally, the recited order of

processing elements or sequences, or the use of numbers, letters, or other designations therefore, is not intended to limit the claimed processes to any order except as may be specified in the claims. Accordingly, the invention is limited only by the following claims and equivalents thereto.

5

CLAIMS

What is claimed is:

1. A method for facilitating a remote interface, the method comprising:

providing, by a remote interface server computing device, a presentation web page to a presentation device and a remote web page to a remote device, the remote web page configured to, when executed by the remote device, register the remote device as associated with the presentation device and render at least a swipe panel on a display of the remote device;

receiving, by the remote interface server computing device, a first message from the remote device in response to a received indication of a user interaction with the swipe panel; and

sending, by the remote interface server computing device, a second message to the presentation device in response to receiving the first message, wherein the second message, when executed by the presentation device, is configured to cause the presentation device to perform an action on the presentation web page corresponding to the user interaction with the swipe panel.

2. The method of claim 1, further comprising providing, by the remote interface server computing device, an initiation web page to the presentation device, the initiation web page configured to register the presentation device and to establish a first connection with a web socket server, wherein:

the remote web page is further configured to register the remote device as associated with the presentation device and to establish a second connection with the web socket server;

the first and second messages comply with a web socket protocol; and

the presentation and remote web pages are provided to the presentation and remote devices, respectively, in response to a received indication of an interaction by the remote device with at least a portion of the initiation web page.

3. The method of claim 1, wherein the received indication of the user interaction further comprises a received indication of at least one of:

a horizontal swipe gesture resulting in a first panel transition on the remote device and the second message, when executed by the presentation device, is configured to cause the presentation device to execute a second panel transition; or

a vertical swipe gesture and the second message, when executed by the presentation device, is configured to cause the presentation device to execute a vertical scroll.

4. The method of claim 1, wherein:

the remote web page is further configured to, when executed by the remote device, render an edit button on the display of the remote device;

the presentation web page further comprises a first input field with obfuscated sensitive information; and

the remote web page is further configured to, when executed by the remote device and in response to receiving a user selection of the edit button, render an editable second input field with the sensitive information visible on the display of the remote device.

5. The method of claim 1, wherein the remote web page is further configured to, when executed by the remote device, render a play button on the display of the remote device, the presentation web page and the remote web page comprise a video, the remote web page is further configured to, when executed by the remote device and in response to receiving a user selection of the play button, modify the swipe panel to display information retrieved using a uniform resource locator (URL), and the method further comprises:

receiving, by the remote interface server computing device, a third message from the remote device in response to a user selection of the play button;

sending, by the remote interface server computing device, a fourth message to the presentation device in response to the third message, the

fourth message indicating to the presentation device that the video has been initiated;

receiving, by the remote interface server computing device, a fifth message from the presentation device, the fifth message including the URL; and

sending, by the remote interface server computing device, a sixth message to the remote device in response to the fifth message, the sixth message including the URL.

6. A remote interface server computing device, comprising a processor and a memory coupled to the processor which is configured to be capable of executing programmed instructions comprising and stored in the memory to:

provide a presentation web page to a presentation device and a remote web page to a remote device, the remote web page configured to, when executed by the remote device, register the remote device as associated with the presentation device and render at least a swipe panel on a display of the remote device;

receive a first message from the remote device in response to a received indication of a user interaction with the swipe panel; and

send a second message to the presentation device in response to receiving the first message, wherein the second message, when executed by the presentation device, is configured to cause the presentation device to perform an action on the presentation web page corresponding to the user interaction with the swipe panel.

7. The remote interface server computing device of claim 6, wherein the processor is further configured to be capable of executing at least one additional programmed instruction comprising and stored in the memory to provide an initiation web page to the presentation device, the initiation web page configured to register the presentation device and to establish a first connection with a web socket server, wherein:

the remote web page is further configured to register the remote device as associated with the presentation device and to establish a second connection with the web socket server;

the first and second messages comply with a web socket protocol; and

the presentation and remote web pages are provided to the presentation and remote devices, respectively, in response to a received indication of an interaction by the remote device with at least a portion of the initiation web page.

8. The remote interface server computing device of claim 6, wherein the received indication of the user interaction further comprises a received indication of at least one of:

a horizontal swipe gesture resulting in a first panel transition on the remote device and the second message, when executed by the presentation device, is configured to cause the presentation device to execute a second panel transition; or

a vertical swipe gesture and the second message, when executed by the presentation device, is configured to cause the presentation device to execute a vertical scroll.

9. The remote interface server computing device of claim 6, wherein:

the remote web page is further configured to, when executed by the remote device, render an edit button on the display of the remote device;

the presentation web page further comprises a first input field with obfuscated sensitive information; and

the remote web page is further configured to, when executed by the remote device and in response to receiving a user selection of the edit button, render an editable second input field with the sensitive information visible on the display of the remote device.

10. The remote interface server computing device of claim 6, wherein the remote web page is further configured to, when executed by the remote device, render a play button on the display of the remote device, the presentation web page and the remote web page comprise a video, the remote web page is further configured to, when executed by the remote device and in response to receiving a user selection of the play button, modify the swipe panel to display information retrieved using a uniform resource locator (URL), and the processor is further configured to be capable of executing at least one additional programmed instruction comprising and stored in the memory to:

receive a third message from the remote device in response to a user selection of the play button;

send a fourth message to the presentation device in response to the third message, the fourth message indicating to the presentation device that the video has been initiated;

receive a fifth message from the presentation device, the fifth message including the URL; and

send a sixth message to the remote device in response to the fifth message, the sixth message including the URL.

11. A non-transitory computer readable medium having stored thereon instructions for facilitating a remote interface comprising executable code which when executed by a processor, causes the processor to perform steps comprising:

providing a presentation web page to a presentation device and a remote web page to a remote device, the remote web page configured to, when executed by the remote device, register the remote device as associated with the presentation device and render at least a swipe panel on a display of the remote device;

receiving a first message from the remote device in response to a received indication of a user interaction with the swipe panel; and

sending a second message to the presentation device in response to receiving the first message, wherein the second message, when executed by the presentation device, is configured to cause the presentation device

to perform an action on the presentation web page corresponding to the user interaction with the swipe panel.

12. The non-transitory computer readable medium of claim 11, wherein the executable code when executed by the processor further causes the processor to perform at least one additional step comprising providing an initiation web page to the presentation device, the initiation web page configured to register the presentation device and to establish a first connection with a web socket server, wherein:

the remote web page is further configured to register the remote device as associated with the presentation device and to establish a second connection with the web socket server;

the first and second messages comply with a web socket protocol; and

the presentation and remote web pages are provided to the presentation and remote devices, respectively, in response to a received indication of an interaction by the remote device with at least a portion of the initiation web page.

13. The non-transitory computer readable medium of claim 11, wherein the received indication of the user interaction further comprises a received indication of at least one of:

a horizontal swipe gesture resulting in a first panel transition on the remote device and the second message, when executed by the presentation device, is configured to cause the presentation device to execute a second panel transition; or

a vertical swipe gesture and the second message, when executed by the presentation device, is configured to cause the presentation device to execute a vertical scroll.

14. The non-transitory computer readable medium of claim 11, wherein:

the remote web page is further configured to, when executed by the remote device, render an edit button on the display of the remote device;

the presentation web page further comprises a first input field with obfuscated sensitive information; and

the remote web page is further configured to, when executed by the remote device and in response to receiving a user selection of the edit button, render an editable second input field with the sensitive information visible on the display of the remote device.

15. The non-transitory computer readable medium of claim 11, wherein the remote web page is further configured to, when executed by the remote device, render a play button on the display of the remote device, the presentation web page and the remote web page comprise a video, the remote web page is further configured to, when executed by the remote device and in response to receiving a user selection of the play button, modify the swipe panel to display information retrieved using a uniform resource locator (URL), and the executable code when executed by the processor further causes the processor to perform at least one additional step comprising:

receiving a third message from the remote device in response to a user selection of the play button;

sending a fourth message to the presentation device in response to the third message, the fourth message indicating to the presentation device that the video has been initiated;

receiving a fifth message from the presentation device, the fifth message including the URL; and

sending a sixth message to the remote device in response to the fifth message, the sixth message including the URL.

16. A system for facilitating a remote interface, the system comprising:

a remote interface server computing device comprising a first processor and a first memory coupled to the first processor which is

configured to be capable of executing programmed instructions comprising and stored in the first memory to:

provide a presentation web page to a presentation device and a remote web page to a remote device, the remote web page configured to, when executed by the remote device, register the remote device as associated with the presentation device and render at least a swipe panel on a display of the remote device; and

a web socket server computing device comprising a second processor and a second memory coupled to the second processor which is configured to be capable of executing programmed instructions comprising and stored in the second memory to:

receive a first message from the remote device in response to a user interaction with the swipe panel; and

send a second message to the presentation device in response to receiving the first message, wherein the second message, when executed by the presentation device, is configured to cause the presentation device to perform an action on the presentation web page corresponding to the user interaction with the swipe panel.

17. The system of claim 16, wherein the first processor is further configured to be capable of executing at least one additional programmed instruction comprising and stored in the first memory to provide an initiation web page to the presentation device, the initiation web page configured to register the presentation device and to establish a first connection with the web socket server computing device, wherein:

the remote web page is further configured to register the remote device as associated with the presentation device and to establish a second connection with the web socket server computing device;

the first and second messages comply with a web socket protocol; and

the presentation and remote web pages are provided to the presentation and remote devices, respectively, in response to a received indication

of an interaction by the remote device with at least a portion of the initiation web page.

18. The system of claim 16, wherein the received indication of the user interaction further comprises a received indication of at least one of:

a horizontal swipe gesture resulting in a first panel transition on the remote device and the second message, when executed by the presentation device, is configured to cause the presentation device to execute a second panel transition; or

a vertical swipe gesture and the second message, when executed by the presentation device, is configured to cause the presentation device to execute a vertical scroll.

19. The system of claim 16, wherein:

the remote web page is further configured to, when executed by the remote device, render an edit button on the display of the remote device;

the presentation web page further comprises a first input field with obfuscated sensitive information; and

the remote web page is further configured to, when executed by the remote device and in response to receiving a user selection of the edit button, render an editable second input field with the sensitive information visible on the display of the remote device.

20. The system of claim 16, wherein the remote web page is further configured to, when executed by the remote device, render a play button on the display of the remote device, the presentation web page and the remote web page comprise a video, the remote web page is further configured to, when executed by the remote device and in response to receiving a user selection of the play button, modify the swipe panel to display information retrieved using a uniform resource locator (URL), and the second processor is further configured to be capable of executing at least one additional programmed instruction comprising and stored in the second memory to:

receive a third message from the remote device in response to a user selection of the play button;

send a fourth message to the presentation device in response to the third message, the fourth message indicating to the presentation device that the video has been initiated;

receive a fifth message from the presentation device, the fifth message including the URL; and

send a sixth message to the remote device in response to the fifth message, the sixth message including the URL.

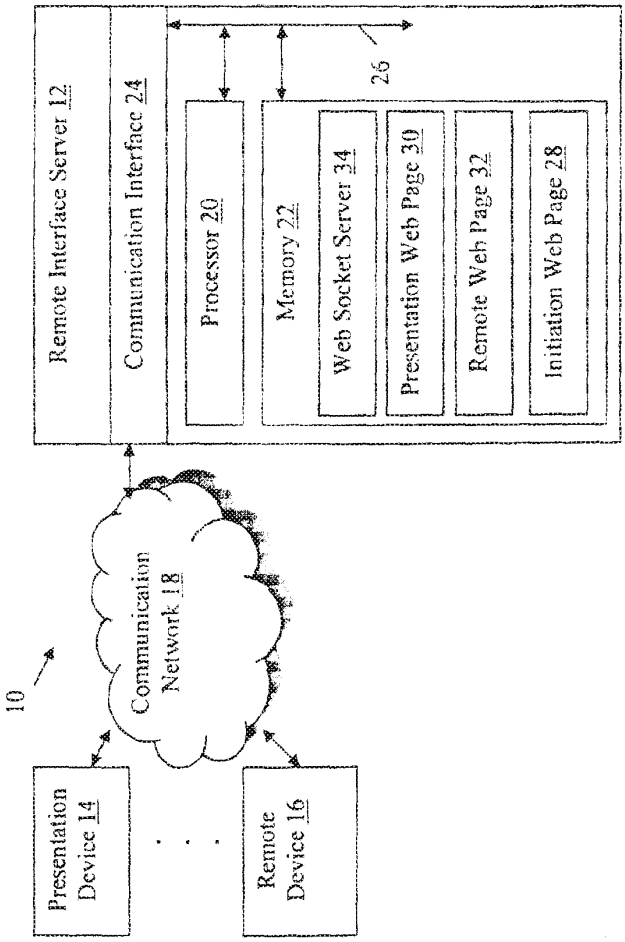

Remote Interface Server 12
Communication Interface 24
Processor 20
Memory 22
Web Socket Server 34
Presentation Web Page 30
Remote Web Page 32
Initiation Web Page 28
26
Communication Network 18
10
Presentation Device 14
Remote Device 16

FIG. 1

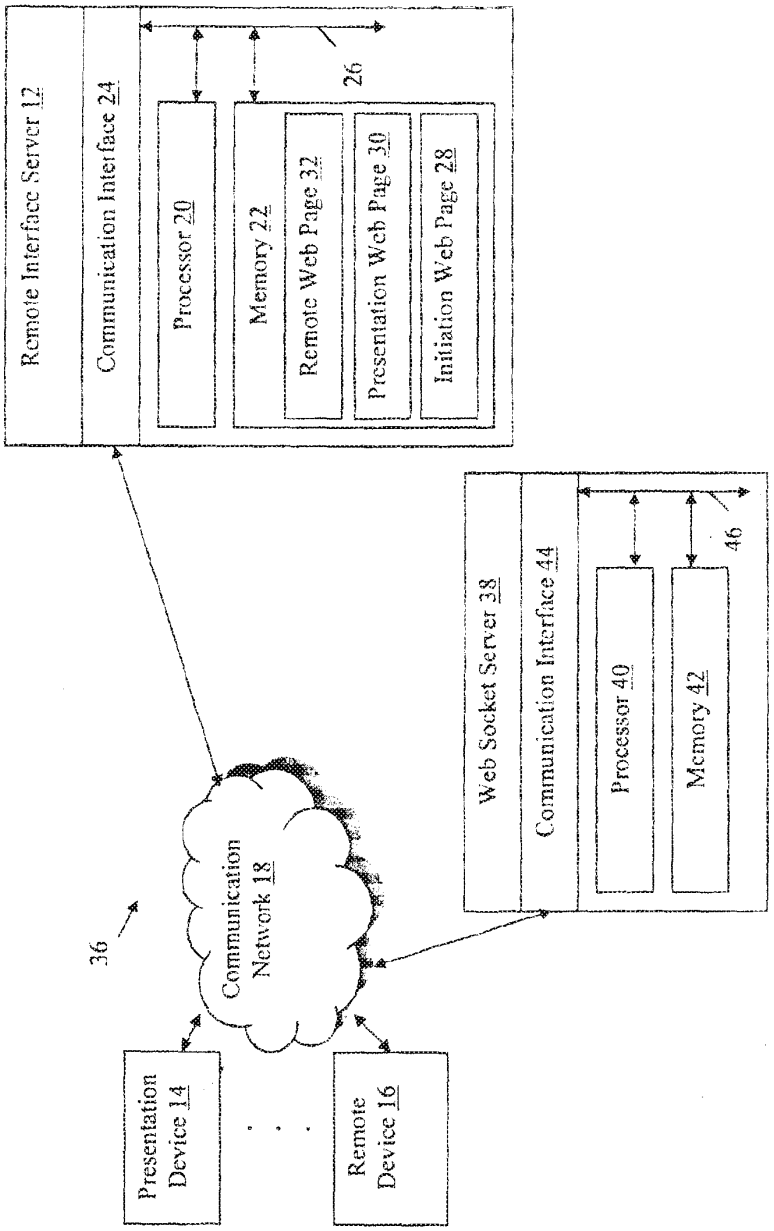

Remote Interface Server 12
Communication Interface 24
Processor 20
Memory 22
Remote Web Page 32
Presentation Web Page 30
Initiation Web Page 28
26
Web Socket Server 38
Communication Interface 44
Processor 40
Memory 42
46
Communication Network 18
36
Presentation Device 14
Remote Device 16

FIG. 2

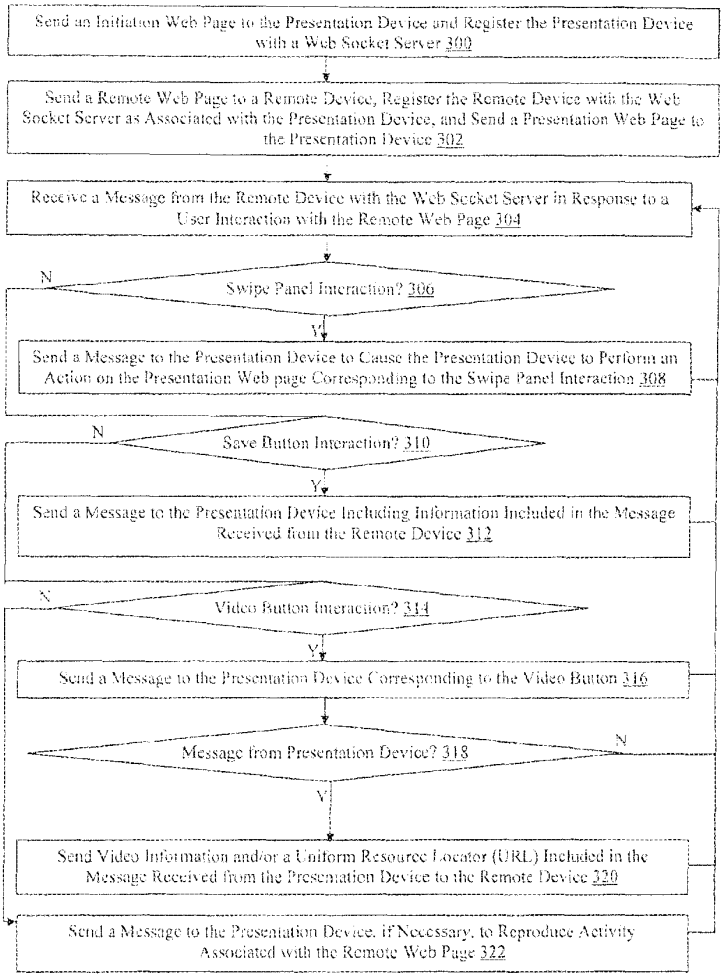

Send an Initiation Web Page to the Presentation Device and Register the Presentation Device with a Web Socket Server 300
Send a Remote Web Page to a Remote Device, Register the Remote Device with the Web Socket Server as Associated with the Presentation Device, and Send a Presentation Web Page to the Presentation Device 302
Receive a Message from the Remote Device with the Web Socket Server in Response to a User Interaction with the Remote Web Page 304
Swipe Panel Interaction? 306
N
Y
Send a Message to the Presentation Device to Cause the Presentation Device to Perform an Action on the Presentation Web page Corresponding to the Swipe Panel Interaction 308
Save Button Interaction? 310
N
Y
Send a Message to the Presentation Device Including Information Included in the Message Received from the Remote Device 312
Video Button Interaction? 314
N
Y
Send a Message to the Presentation Device Corresponding to the Video Button 316
Message from Presentation Device? 318
N
Y
Send Video Information and/or a Uniform Resource Locator (URL) Included in the Message Received from the Presentation Device to the Remote Device 320
Send a Message to the Presentation Device, if Necessary, to Reproduce Activity Associated with the Remote Web Page 322

FIG. 3

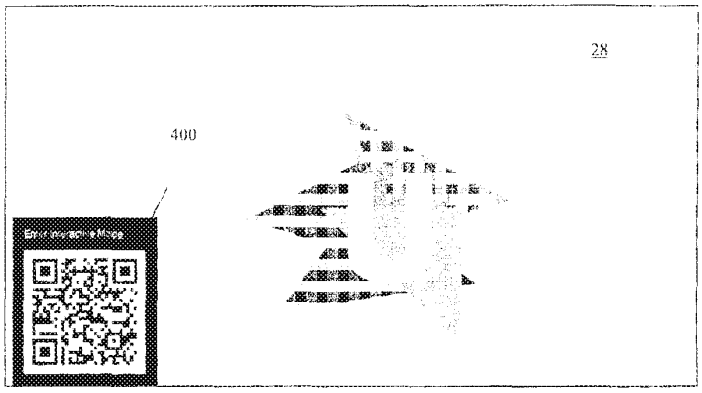


FIG. 4

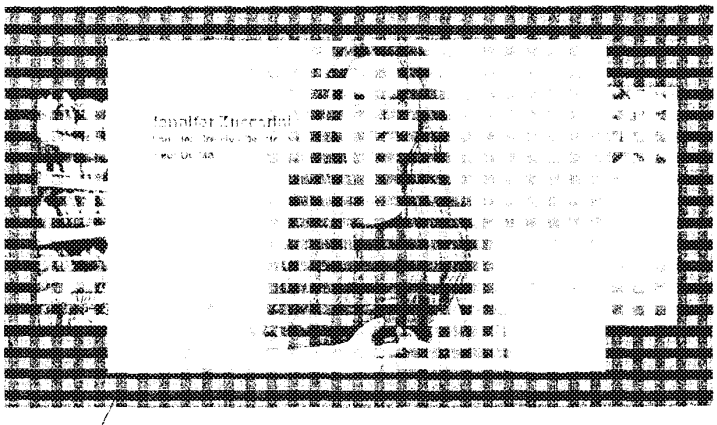


FIG. 5

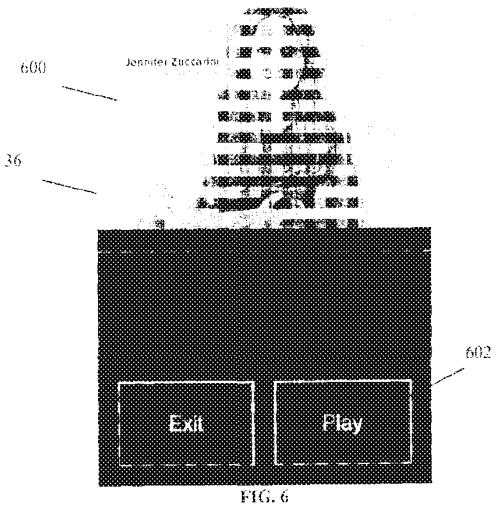


FIG. 6

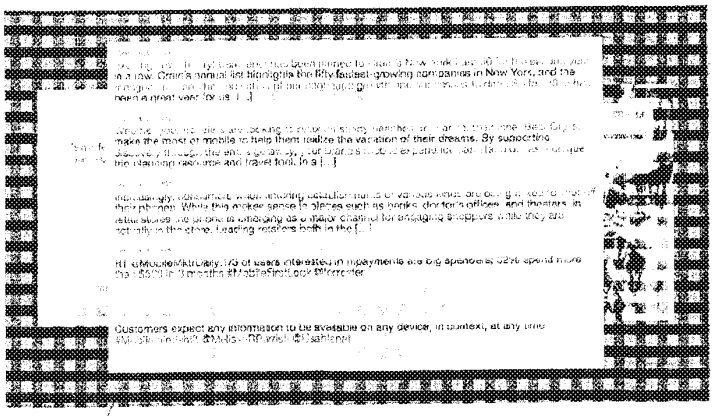


FIG. 7

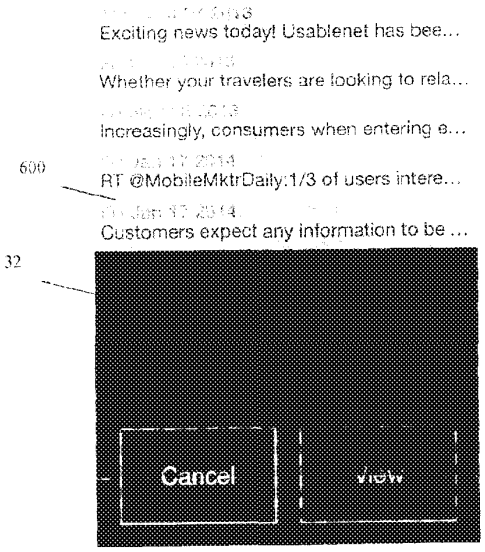

Exciting news today! Usablenet has bee...
Whether your travelers are looking to rela...
Increasingly, consumers when entering e...
RT @MobileMktrDaily:1/3 of users intere...
Customers expect any information to be ...
600
32
Cancel

FIG. 8

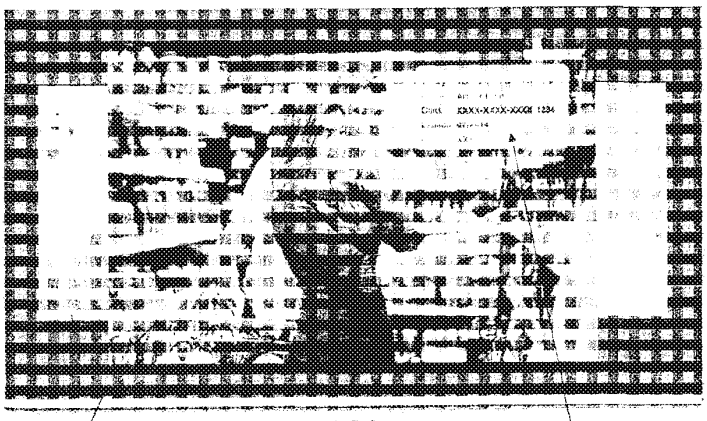


FIG. 9

30 900

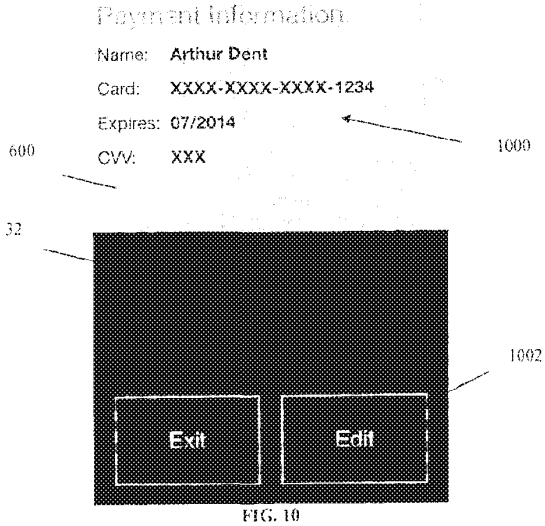


FIG. 10

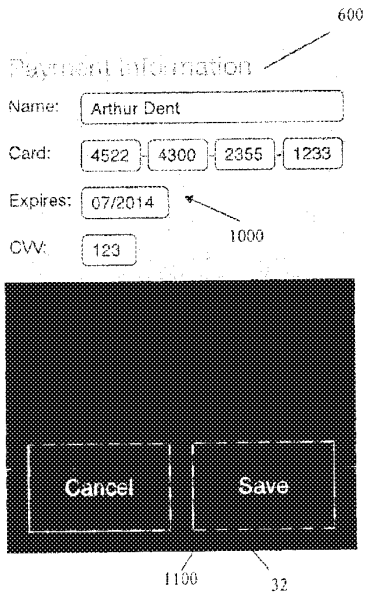

600
Payment Information
Name:
Arthur Dent
Card:
4522
4300
2355
1233
Expires:
07/2014
1000
CVV:
123
Cancel
Save
1100
32

FIG. 11

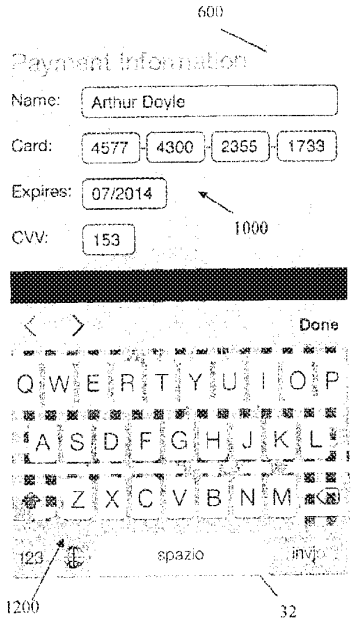

600
Payment Information
Name:
Arthur Doyle
Card:
4577
4300
2355
1733
Expires:
07/2014
1000
CVV:
153
Done
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
123
spazio
invio
1200
32

FIG. 12

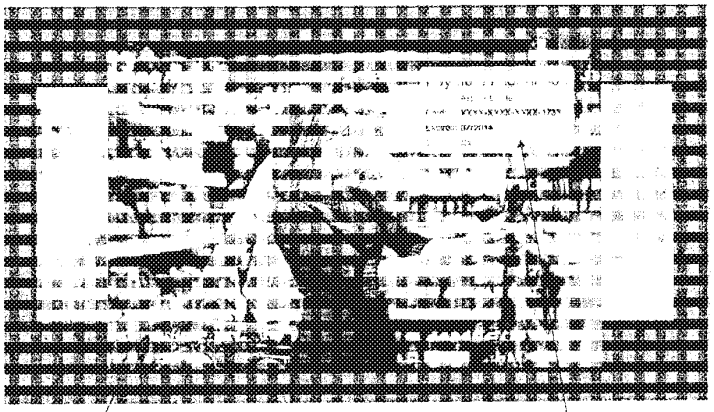


FIG. 13

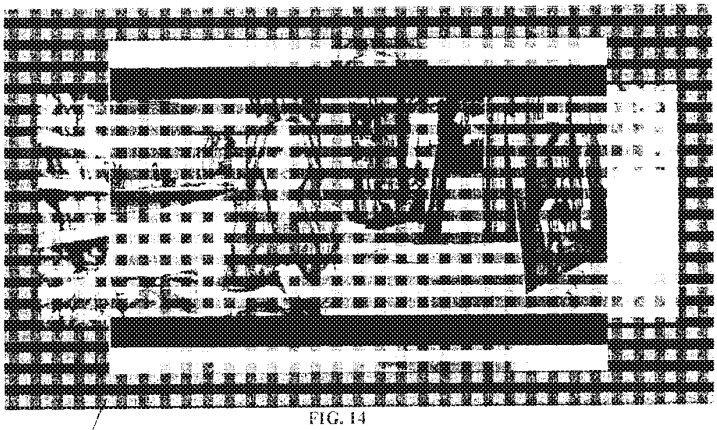


FIG. 14

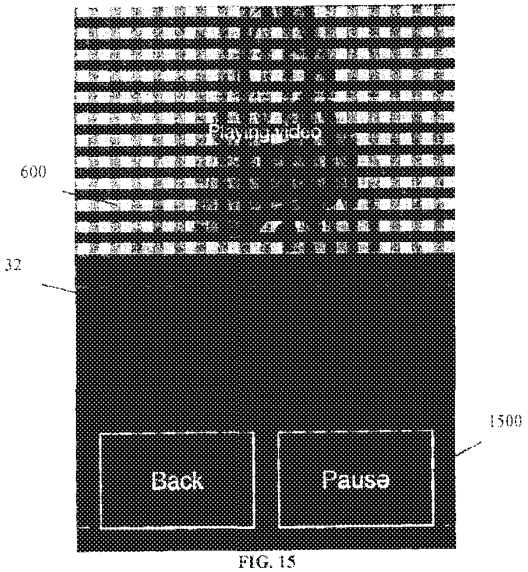


FIG. 15

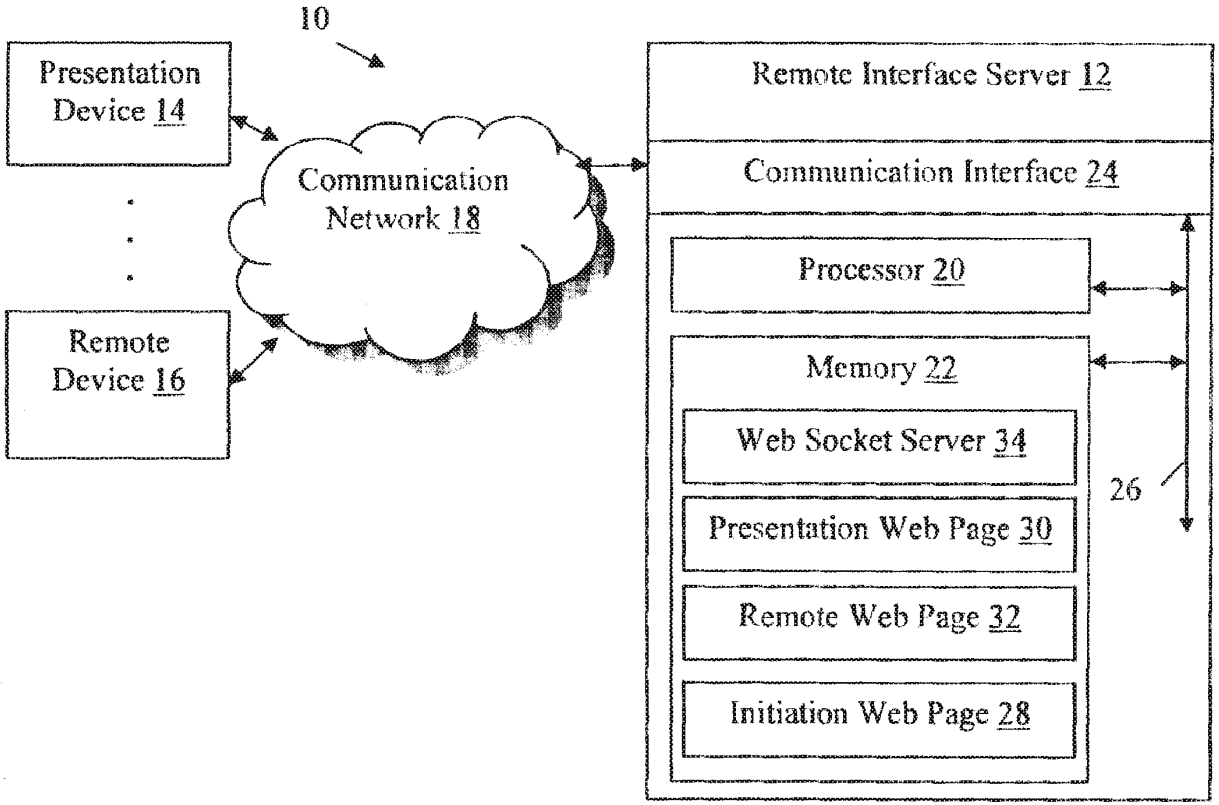


FIG. 1

Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2951659 A1 2017/06/22

(21) 2 951 659

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(22) Date de dépôt/Filing Date: 2016/12/13

(41) Mise à la disp. pub./Open to Public Insp.: 2017/06/22

(30) Priorité/Priority: 2015/12/22 (US14/977,995)

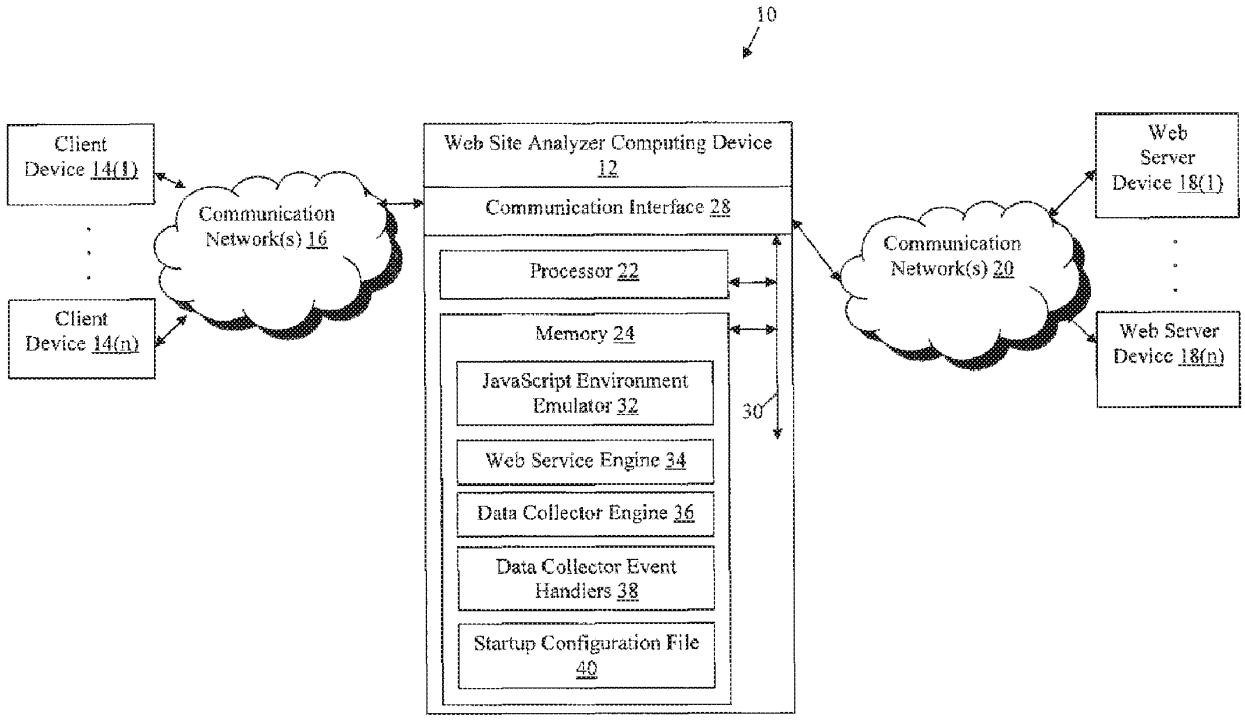
(51) Cl.Int./Int.Cl. *H04L 12/16* (2006.01),
H04L 12/26 (2006.01)

(71) Demandeur/Applicant:
USABLENET INC., US

(72) Inventeurs/Inventors:
SCODA, ENRICO, IT;
BRONDANI, MARCO, IT

(74) Agent: PARLEE MCLAWS LLP

(54) Titre : METHODE D'ANALYSE DE SITES WEB AU MOYEN DE SERVICES WEB ET DISPOSITIFS CONNEXES
(54) Title: METHODS FOR ANALYZING WEB SITES USING WEB SERVICES AND DEVICES THEREOF



(57) Abrégé/Abstract:
Methods, non-transitory computer readable media, and devices that determine when a job extracted from a stack is a pending job. When the determining indicates that the job is a pending job, a web service indicated in the job is executed. Another job is extracted from a web service response, and the another job is inserted into the stack. The web service is configured to obtain a web page to be analyzed based on the URL, execute the web page in an emulated JavaScript environment, and return the web

CA 2951659 A1 2017/06/22
(21) 2 951 659
(13) A1

(57) Abrégé(suite)/Abstract(continued):
service response. When the determining indicates that the job is not a pending job, a data collector event handler indicated in the job is executed. The data collector event handler is configured to update an output resource based on content of the analyzed web page included in the job.

ABSTRACT

Methods, non-transitory computer readable media, and devices that determine when a job extracted from a stack is a pending job. When the determining indicates that the job is a pending job, a web service indicated in the job is executed. Another job is extracted from a web service response, and the another job is inserted into the stack. The web service is configured to obtain a web page to be analyzed based on the URL, execute the web page in an emulated JavaScript environment, and return the web service response. When the determining indicates that the job is not a pending job, a data collector event handler indicated in the job is executed. The data collector event handler is configured to update an output resource based on content of the analyzed web page included in the job.

METHODS FOR ANALYZING WEB SITES USING WEB SERVICES AND DEVICES THEREOF

BACKGROUND

[0001] Web site analyzers, which are often referred to as indexers, spiders, bots, or crawlers, for example, navigate web sites and collect information regarding their structure or content. These analyzers have many uses including identifying security threats in a web site, evaluating web pages associated with a web site for implementation quality, and producing a list or sitemap of web pages of a web site that should be indexed by search engines, for example. In some instances, analyzers can be used to generate client-facing content, such as a list of available products with current offers in a retailer web site or a static version of a retailer web site catalog that can be used as a catalog navigation menu by a mobile application associated with the retailer or web site, for example.

[0002] However, current web site analyzers are limited to inspecting static web page documents associated with web sites. As a result, current web site analyzers do not effectively execute some web pages, including those web pages that include client-side JavaScript code. More specifically, current web site analyzers are unable to extract information from web pages that is hidden inside the client-side JavaScript code. In one particular example, a catalog navigation menu may be generated, when a web page is executed client-side, by downloading and processing a JavaScript Object Notation (JSON) resource, which would not be accessible or executable by current web site analyzers. Therefore, current analyzers have limited functionality and visibility into certain web sites resulting in relatively inaccurate or incomplete results that have limited utility.

SUMMARY

[0003] A method for analyzing web sites using web services includes determining, by a web site analyzer computing device, when a job extracted from a stack is a pending job. When the determining indicates that the job is a pending job, a web service indicated in the job is executed, by the web site analyzer computing device, by passing a Uniform Resource Locator (URL) included in the job as a parameter to the web service. Another job is

extracted, by the web site analyzer computing device, from a web service response, and the another job is inserted, by the web site analyzer computing device, into the stack. The web service is configured to obtain a web page to be analyzed based on the URL, execute the web page in an emulated JavaScript environment, and return the web service response. When the determining indicates that the job is not a pending job, then a data collector event handler indicated in that job is executed, by the web site analyzer computing device, by passing that job as a parameter to the data collector event handler. The data collector event handler is configured to update an output resource based on content of the analyzed web page included in that job.

[0004] A non-transitory computer readable medium having stored thereon programmed instructions for analyzing web sites using web services and includes executable code that, when executed by at least one processor, causes the processor to perform steps including determining when a job extracted from a stack is a pending job. When the determining indicates that the job is a pending job, a web service indicated in the job is executed by passing a URL included in the job as a parameter to the web service. Another job is extracted from a web service response, and the another job is inserted into the stack. The web service is configured to obtain a web page to be analyzed based on the URL, execute the web page in an emulated JavaScript environment, and return the web service response. When the determining indicates that the job is not a pending job, a data collector event handler indicated in that job is executed by passing that job as a parameter to the data collector event handler. The data collector event handler is configured to update an output resource based on content of the analyzed web page included in that job.

[0005] A web site analyzer computing device includes one or more processors coupled to a memory and configured to execute programmed instructions including and stored in the memory to determine when a job extracted from a stack is a pending job. When the determining indicates that the job is a pending job, a web service indicated in the job is executed by passing a URL included in the job as a parameter to the web service. Another job is extracted from a web service response, and the another job is inserted into the stack. The web service is configured to obtain a web page to be analyzed based on the URL,

execute the web page in an emulated JavaScript environment, and return the web service response. When the determining indicates that the job is not a pending job, a data collector event handler indicated in that job is executed by passing that job as a parameter to the data collector event handler. The data collector event handler is configured to update an output resource based on content of the analyzed web page included in that job.

[0006] This technology provides a number of advantages including providing methods, non-transitory computer readable media, and web site analyzer computing devices that utilize web services and emulated JavaScript environments to more effectively analyze web pages of web sites. In particular, this technology extracts web page information, including information inside client-side JavaScript code, to facilitate a more thorough analysis of web sites. This technology also advantageously utilizes data collector event handlers that provide flexibility with respect to the type of provided output.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] FIG. 1 is a block diagram of an environment with an exemplary web site analyzer computing device;

[0008] FIG. 2 is a flow chart of an exemplary method for analyzing web sites using web services and the exemplary web site analyzer computing device of FIG. 1;

[0009] FIG. 3A is an exemplary stack including a portion of a pending job generated based on a startup configuration file;

[0010] FIG. 3B is a portion of an exemplary web service response generated by the web service indicated in the exemplary pending job in the exemplary stack of FIG. 3A and using the URL of the exemplary pending job in the exemplary stack of FIG. 3A;

[0011] FIG. 3C is an exemplary stack including a portion of each of a plurality of exemplary jobs resulting from processing of the exemplary web service response of FIG. 3B;

[0012] FIG. 4 is a plurality of exemplary data collector event handlers that process a subset of the plurality of exemplary jobs included in the exemplary stacks of FIGS. 3C, 6B, and 6B;

[0013] FIG. 5 is an exemplary output resource generated by the exemplary method for analyzing web sites using web services of FIG. 2.

[0014] FIG. 6A is a portion of an exemplary web service response generated by the web service indicated in a pending one of the exemplary jobs in the exemplary stack of FIG. 3C;

[0015] FIG. 6B is an exemplary stack including a portion of each of a plurality of a plurality of exemplary jobs resulting from processing the web service response of FIG. 6A;

[0016] FIG. 7A is a portion of an exemplary web service response generated by the web service indicated in a pending one of the exemplary jobs in the exemplary stack of FIG. 6B; and

[0017] FIG. 7B is an exemplary stack including a portion of each of a plurality of a plurality of exemplary jobs resulting from processing the web service response of FIG. 7A.

DETAILED DESCRIPTION

[0018] An exemplary environment 10 with a web site analyzer computing device 12 coupled to client devices 14(1)-14(n) via communication network(s) 16 and web server devices 18(1)-18(n) via communication networks 20 is illustrated in FIG. 1. Other numbers and types of systems, devices, and/or elements in other configurations and environments with other communication network topologies can also be used. This technology provides a number of advantages including providing methods, non-transitory computer readable media, and web site analyzer computing devices that more effectively analyze web sites using web services that execute web pages of the web sites in an emulated JavaScript environment in order to extract information from client-side JavaScript code of the web pages.

[0019] The web site analyzer computing device 12 in this particular example includes a processor 22, a memory 26, and a communication interface 28 which are coupled together by a bus 30 or other communication link, although other numbers and types of components, parts, devices, systems, and elements in other configurations and locations can be used. The processor 22 in the web site analyzer computing device 12 executes a program of stored instructions for one or more aspects of this technology as described and illustrated by way of the examples herein, although the processor 22 could execute other numbers and types of programmed instructions.

[0020] The memory 24 in the web site analyzer computing device 12 stores these programmed instructions for one or more aspects of the present invention as described and illustrated herein, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM), a read only memory (ROM), solid state drives, flash, or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to the processor 22, can be used for the memory 24 in the web site analyzer computing device 12.

[0021] In this particular example, the memory 24 includes a JavaScript environment emulator 32, a web service engine 34, a data collector engine 36, data collector event handlers 38, and a startup configuration file 40, although other types or numbers of modules or applications can be included in the memory 24 in other examples. By way of example only, the JavaScript environment emulator 32, web service engine 34, and/or data collector engine 36, can be implemented as executable modules of programmed instructions and/or configurable hardware logic for one or more of aspects of the technology described and illustrated herein, which are stored in the memory 24 and executed by the processor 22 in the web site analyzer computing device 12.

[0022] The JavaScript environment emulator 32 in this example can include programmed instructions and/or hardware logic configured to simulate a JavaScript environment for executing JavaScript code that may be included in web pages, as described and illustrated in more detail below. By way of example only, a JavaScript environment

emulator 20 is illustrated and described in U.S. Patent Application Serial No. 12/802,670 entitled, "Methods For Utilizing A JavaScript Emulator In A Web content proxy Server And Devices Thereof," which is incorporated herein by reference in its entirety.

[0023] The web service engine 34 in this example can include programmed instructions and/or hardware logic configured to execute web services. Web services provide a standardized way of integrating web-based applications using eXtensible Markup Language (XML) and/or REpresentational State Transfer (REST) (e.g., using Java Universal Description Discovery and Integration (jUDDI) and/or a Swagger™ framework for a description standard) over an Internet Protocol (IP) backbone.

[0024] In this particular example, the web service engine 34 executes web services that execute web pages obtained from the server devices 18(1)-18(n) in an emulated JavaScript environment using the JavaScript environment emulator 32, extract information from the web pages, and return web service responses in a preconfigured format. By executing the web pages in an emulated JavaScript environment, the web services are able to extract information from client-side JavaScript code, as described and illustrated in more detail later. The returned web service responses can include pending jobs, as well as start and end jobs that indicate data collector event handlers 38 and include information regarding the analyzed web pages that the data collector event handlers 38 use to generate an output resource, as described and illustrated in more detail later.

[0025] The data collector engine 34 in this example can include programmed instructions and/or hardware logic configured to process web service responses including the jobs included therein. Accordingly, the data collector engine 34 collects and executes pending jobs and calls the data collector event handlers 38 for start and end jobs, as described and illustrated in more detail later.

[0026] The data collector event handlers 38 in this example are JavaScript functions that are called by the data collector engine 34 and take in start and end jobs as parameters. Based on the start and end jobs passed as parameters, the data collector event handlers 38 updates an output resource, also as described and illustrated in more detail later.

[0027] The startup configuration file 40 includes an indication of an initial web service to be called as well as a URL of an initial web page of a web site to be analyzed. Optionally, the startup configuration file 40 further includes a maximum number of web pages to process or a function callback configured to validate the output resource, for example, although the startup configuration file 40 can also include other information.

[0028] One or more of the JavaScript environment emulator 32, web service engine 34, and/or data collector engine 36, can also have other types and numbers of functions as described and illustrated herein. Additionally, one or more of the JavaScript environment emulator 32, web service engine 34, data collector engine 36, data collector event handlers 38, or startup configuration file 40 can be stored at and/or implemented by a separate device coupled to the web site analyzer computing device 12 by one or more of the communication network(s) 16 and 20, such as one or more of the web server devices 18(1)-18(n).

[0029] The communication interface 28 in the web site analyzer computing device 12 is used to operatively couple and communicate between the web site analyzer computing device 12, the client devices 14(1)-14(n) and the server devices 18(1)-18(n) via the communication network(s) 16 and 20, although other types and numbers of communication networks with other types and numbers of connections and configurations can be used. Additionally, one or more of the communication network(s) 16 and 20 can include one or more local area networks (LANs) and/or wide area networks (WANs). By way of example only, the communication network(s) 16 and 20 can use TCP/IP over Ethernet and industry-standard protocols, including hypertext transfer protocol (HTTP), secure HTTP (HTTPS), wireless application protocol (WAP), and/or SOAP, although other types and numbers of communication networks each having their own communications protocols, can be used.

[0030] The client devices 14(1)-14(n) in this example enable a user to request, receive, and interact with applications, web services, and content hosted by the server devices 18(1)-18(n) through the web site analyzer computing device 12 via one or more communication network(s) 16, although one or more of the client devices 14(1)-14(n) could access content and utilize other types and numbers of applications from other sources and could provide a wide variety of other functions for the user. In some examples, the client

devices 14(1)-14(n) comprise mobile computing devices with Internet access that enable one or more web services to be accessed. By way of example only, the client devices 14(1)-14(n) can be smart phones, personal digital assistants, or computers.

[0031] Each of the client devices 14(1)-14(n) includes one or more processors, a memory, a user input device, a display device, and a communication interface, which are coupled together by a bus or other communication link, although one or more of client devices 14(1)-14(n) can include other numbers and types of components, parts, devices, systems, and elements in other configurations. The processor(s) in the client devices 14(1)-14(n) can execute a program of instructions stored in the memory of the client devices 14(1)-14(n) for one or more aspects of this technology as described and illustrated herein, although the processor(s) could execute other numbers and types of programmed instructions.

[0032] The user input device in the client devices 14(1)-14(n) can be used to input selections, such as a request for a particular web site, although the user input device could be used to input other types of requests and data and interact with other elements. The user input device can include keypads, touch screens, and/or vocal input processing systems although other types and numbers of user input devices can be used.

[0033] The display device the client devices 14(1)-14(n) can be used to output data and information to the user, such as a requested web page by way of example only. The display device in the client devices 14(1)-14(n) can be a phone screen display, although other types and numbers of display devices could be used depending on the particular type of client device. The communication interface in the client devices 14(1)-14(n) can be used to operatively couple and communicate between the client devices 14(1)-14(n), the web site analyzer computing device 12, and the server devices 18(1)-18(n) over the communication networks 16 and 20.

[0034] The server devices 18(1)-18(n) provide content including web pages for use by one or more of the client devices 14(1)-14(n) or to be analyzed by the web site analyzer computing device 12, although the server devices 18(1)-18(n) can provide other numbers and types of functions. Each of the server devices 14(1)-14(n) in this example includes one or

more processors, a memory, and a communication interface which are coupled together by a bus or other communication link, although each of the web server devices 18(1)-18(n) could have other numbers and types of components, parts, devices, systems, and elements in other configurations and locations.

[0035] The processor in each of the server devices 18(1)-18(n) executes a program of instructions stored in the memory of the server devices 18(1)-18(n) for one or more aspects of this technology, as described and illustrated by way of the examples herein, although the processor could execute other numbers and types of programmed instructions. The communication interface in each of the server devices 18(1)-18(n) is used to operatively couple and communicate between the server devices 18(1)-18(n), the web site analyzer computing device 12, and the client devices 14(1)-14(n) via communication networks 16 and 20.

[0036] Although the exemplary web site analyzer computing device 12, client devices 14(1)-14(n), and server devices 18(1)-18(n), are described and illustrated herein, each of the web site analyzer computing device 12, client devices 14(1)-14(n), and server devices 18(1)-18(n), can be implemented on any suitable computer apparatus or computing device. It is to be understood that the apparatuses and devices of the embodiments described herein are for exemplary purposes, as many variations of the specific hardware and software used to implement the embodiments are possible, as will be appreciated by those skilled in the relevant art(s).

[0037] The examples of this technology described and illustrated herein may also be implemented on computer apparatuses or devices that extend across any suitable network using any suitable interface mechanisms and communications technologies, including by way of example only telecommunications in any suitable form (e.g., voice and modem), wireless communications media, wireless communications networks, cellular communications networks, G3 communications networks, Public Switched Telephone Network (PSTNs), Packet Data Networks (PDNs), the Internet, intranets, and combinations thereof.

[0038] The examples of this technology described and illustrated herein may also be embodied as one or more non-transitory computer readable media having instructions stored thereon for one or more aspects of this technology, as described and illustrated by way of the embodiments herein, which when executed by a processor, cause the processor to carry out the steps necessary to implement the methods of the examples, as described and illustrated herein.

[0039] An exemplary method for analyzing web sites using web services will now be described with reference to FIGS 1-7. Referring more specifically to FIG. 2, in step 200 in this particular example, the web site analyzer computing device 12 obtains the startup configuration file 40 from the memory 24 or from another location or network device. The startup configuration file includes at least a Uniform Resource Locator (URL) associated with a web site to be analyzed, such as a URL of a home web page for the web site, and an indication of a web service that is the initial web service that will initiate the analysis of the web site, as described and illustrated in more detail later. Optionally, the startup configuration file 40 can include other configuration information, such as a maximum number of web pages to process or a function callback configured to validate an output resource corresponding to a result of the analysis, for example, although other configuration information can also be included in the startup configuration file 40.

[0040] In step 202 in this example, the web site analyzer computing device 12 inserts a pending job including an indication of the web service and the URL included in the startup configuration file into a last-in-first-out (LIFO) data structure, which is referred to herein as a stack. The LIFO structure facilitates a depth-first inspection of the web pages of the web site, as described and illustrated in more detail later. In this particular example, a job is a structured record that can have a type indicating that the job is a pending job, a start job, or an end job.

[0041] A pending job in this example includes at least an indication of a web service, an indication of an endpoint of the web service, and a URL of a web page to be analyzed, and optionally also includes a name or a canonical URL associated with the URL of the web page to be analyzed. A start job in this example includes at least an indication of one of the data

collector event handlers 38 and optionally also includes a name or the content of an analyzed web page obtained based on the execution of the web page in the emulated JavaScript environment, as described and illustrated in more detail later. In this example, an end job includes at least an indication of one of the data collector event handlers 38, and optionally also includes a name. Other types of information can also be included in one or more of the jobs and jobs having other types can also be used in other examples.

[0042] Referring to FIG. 3A, an exemplary stack 300 including a portion of a pending job generated based on the startup configuration file 40 is illustrated. In this example, the web site analyzer computing device 12 generates the job 302 to have a pending type, an indication of the "root" web service included in the startup configuration file 40, and the "http://acme.com" URL included in the startup configuration file 40. Subsequent to generating the job 302, the web site analyzer computing device 12 inserts the job 302 into the stack 300.

[0043] Referring back to FIG. 2, in step 204 in this example, the web site analyzer computing device 12 determines whether the stack 300 is empty. In the first iteration, the stack 300 will never be empty. Accordingly, if the web site analyzer computing device 12 determines that the stack is not empty, then the No branch is taken to step 206. In step 206 in this example, the web site analyzer computing device 12 extracts a job from the stock 300, which is pending job 302 in the first iteration in this example.

[0044] In step 208 in this example, the web site analyzer computing device 12 determines whether the extracted job is a pending job based on a type identified in the job. In the first iteration, the job 302 generated based on the information contained in the startup configuration file 40 will always be of a pending type. Other nomenclature can also be used in other examples to indicate that a job includes an indication of a web service and a URL. Accordingly, if the web site analyzer computing device 12 determines that the job 302 is a pending job, then the Yes branch is taken to step 210.

[0045] In step 210 in this example, the web site analyzer computing device 12 optionally determines whether the extracted job 302 is a duplicate job. In the first iteration in

this example, the job 302 will never be a duplicate job. However, in subsequent iterations, a canonical URL included in the job can be compared by the web site analyzer computing device 12 to a stored set of canonical URLs associated with previously analyzed web pages. The canonical URL can be included in the job by a web service that generate a web service response defining the job, as described and illustrated in more detail later. If the web site analyzer computing device 12 determines that the canonical URL included in the job matches one of the stored set of canonical URLs, then the web site analyzer computing device 12 will determine that the job is a duplicate job and take the Yes branch from step 210 back to step 204 without performing steps 212 and 214 for the job.

[0046] However, if the web site analyzer computing device 12 determines that the canonical URL included in the job does not match one of the stored set of canonical URLs, then the web site analyzer computing device 12 will determine that the job is not a duplicate job. If the web site analyzer computing device 12 determines that the job is not a duplicate job, then the canonical URL included in the job can be added to the stored set of canonical URLs to facilitate subsequent identification of duplicate jobs. By configuring the web services to include canonical URLs in definitions of jobs in web service responses, and performing the comparison in step 210, the web site analyzer computing device 12 can avoid entering an infinite loop that could otherwise occur based on the topologies of some web site. Accordingly, if the web site analyze computing device 12 determines that the job 302 extracted in step 216 is not a duplicate job, then the No branch is taken from step 210 to step 212.

[0047] In step 212 in this example, the web site analyzer computing device 12 executes a web service indicated in the job 302, which is the "root" web service in job 302 in this example. The web service is configured to obtain the web page corresponding to the URL included in the job 302, execute the web page in an emulated JavaScript environment provided by the JavaScript Environment emulator 32, and return a web service response. By executing the web page in an emulated JavaScript environment, the web service is advantageously able to collect information from the web page that is only accessible by executing client-site JavaScript code. Accordingly, the web site analyzer computing device

12 calls the "root" web service in this example and passes the "http://acme.com" URL to the "root" web service as a parameter.

[0048] In step 214 in this example, the web site analyzer computing device 12 extracts one or more jobs from the web service response and insert the job(s) into the stack 300 between start and end jobs. Web service responses in this example include definitions of one or more job(s) and an indication of one of the data collector event handlers 38, and optionally also include a description the analyzed web page or a hash generated from the web page. The start and end jobs include an indication of one of the data collector event handlers 38 indicated in the web service response outside of the job definitions.

[0049] Referring to FIG. 3B, a portion of an exemplary web service response 304 generated by the "root" web service indicated in the pending job 302 in the stack 300 using the "http://acme.com" URL of the pending job 302 is illustrated. In this particular example, the "root" web service executes the web page corresponding to the "http://acme.com" URL and returns the web service response 304. The web service response identifies the "home" one of the data collector event handlers 38 (referred to in FIG. 3B as the "type"). The web service response 304 does not include a hash, but the optional hash can be used, in addition to or in place of the canonical URLs, to identify duplicated jobs based on web pages having corresponding content. The web service 304 also does not include any description of the web page (referring to in FIG. 3B as the "content"). However, the description can include content of the web page to be passed to one of the data collector event handlers 38, as described and illustrated in more detail later.

[0050] Additionally, the web service response 304 in this particular examples includes two job definitions, each of which identifies the same one of the data collector event handlers 38 (referred to as the "action" in FIG. 3B), which is the "plp" data collector event handler. Each of the jobs are defined to have a pending type and include a name, URL, and canonical URL, although other information can also be included in the job definitions in other examples. Optionally, the web services can be configured to define pending jobs based on specified criteria (e.g., type of URL associated with an identified the web page).

[0051] Referring to FIG. 3C, the stack 300 including a portion of each of the jobs resulting from processing of the exemplary web service response 304 of FIG. 3B is illustrated. In this particular example, the web site analyzer computing device 12 inserts a start job 306, pending jobs 308 and 310 extracted from the web service response 304, and an end job 312 into the stack 300. The pending jobs 308 and 310 include an indication of the "plp" one of the data collector event handlers 38 identified in the definition of each of the jobs 308 and 310 in the web service response 304. The web site analyzer computing device 12 inserts the pending jobs 308 and 310 into the stack between the start job 306 and the end job 312. Each of the start job 306 and the end job 312 includes an indication of the "home" one of the data collector event handlers 38 that is identified in the web service response 304.

[0052] Referring back to FIG. 2, subsequent to updating the stack 300 in step 214, the web site analyzer computing device 12 proceeds back to step 204 and again determines whether the stack 300 is empty. In this iteration in this example, the web site analyzer computing device 12 will again determine that the stack is not empty and will proceed to extract job 306 from the stack 300 in step 206. In step 208, the web site analyzer computing device 12 again determines whether the job is of a pending type. Since job 306 is a start job, the web site analyzer computing device 12 determines in this iteration that the extracted job is not a pending job and the No branch is taken to step 216.

[0053] In step 216 in this example, the web site analyzer computing device 12 executes one of the data collector event handlers 38 indicated in the job 306, which is the "home" one of the data collector event handlers 38 in this example. The data collector event handlers 38 are JavaScript functions that configured to update an output resource, although the data collector event handlers 38 can be written in other languages and can be configured to provide other functionality in other examples.

[0054] Referring to FIG. 4, a plurality of exemplary data collector event handlers 38(1)-38(6) that process jobs is illustrated. Accordingly, in this example, the job is a start job indicating the "home" data collector event handlers 38(1). Accordingly, the web site analyzer computing device 12 executes the data collector event handler 38(1) by passing the

job 306 in this iteration. Upon execution, the data collector event handler 38(1) updates an output resource.

[0055] Referring to FIG. 5 an exemplary output resource 500 is illustrated. In this particular example the output resource is an eXtensible Markup Language (XML) document, but the output resource can be a JavaScript Object Notation (JSON) file, one or more records configured to be stored in a database, an e-mail or other electronic communication, or any other type of resource in other examples.

[0056] Accordingly, the data collector event handler 38(1) in this example is configured in this example to update the output resource 500 in this example to include a "<catalog>" start tag. In other examples, one or more of the data collector event handlers 38, such as data collector event handlers 38(3) and 38(5), for example, are configured to update the output resource based on content of the analyzed web page included in the job, as described and illustrated in more detail later.

[0057] However, in this example, subsequent to executing the data collector event handler 38(1) indicated in the extracted job 306 in step 216, the web site analyzer computing device 12 again proceeds back to step 204 and again determines whether the stack 300 is empty. Since the stack is not empty subsequent to the extraction of job 306, the web site analyzer computing device 12 will again take the no branch to step 206 and extract the job 308 from the stack 300. In this iteration, the extracted job 308 is a pending job and, accordingly, the web site analyzer computing device 12 will take the Yes branch from step 208 to step 210. Additionally, since the job 308 is not a duplicate job, the web site analyzer computing device 12 will take the No branch from step 210 to step 212.

[0058] In step 212 in this iteration, the web site analyzer computing device 12 will execute the "plp" web service indicated in the job 308 by passing the job 308 as a parameter. The "plp" web service in this iteration is configured to obtain the web page corresponding to the "http://acme.com/c2141" URL included in the job 308, executes the web page in an emulated JavaScript environment, and returns a web service response, such as the web service response 600 illustrated in FIG. 6A, for example.

[0059] In this example, the web service response 600 identifies a "plp" one of the data collector event handlers 38, a description of the web page, and four jobs including two start and two end jobs. Referring back to FIG. 2, in step 214 in this iteration, the web site analyzer computing device extracts the four jobs from the web service response 600 and inserts the four jobs into the stack 300 between start and end jobs that include an indication of the "plp" one of the data collector event handlers 38.

[0060] Referring to FIG. 6B, the stack 300 subsequent to step 214 in this iteration is illustrated. Accordingly, the four jobs defined in the web service response 600 are jobs 602, 604, 606, and 608, and jobs 602, 604, 606, and 608 are inserted into the stack 300 between start job 610 and end job 612. Accordingly, jobs 602-612 will be processed before steps 310 and 312 resulting in a depth-first inspection of the web site due to the LIFO structure of the stack. Referring back to FIG. 2, subsequent to extracting the jobs 602, 604, 606, and 608 and inserting the jobs 602, 604, 606, 608, 610, and 612 into the stack 300, the web site analyzer computing device 12 proceeds back to step 204.

[0061] In this iteration, the web site analyzer computing device 12 will again determine that the stack 300 is not empty and take the No branch from step 204 to step 206. In step 206, the web site analyzer computing device 12 extracts job 610 from the stack. Job 610 is not a pending job and, accordingly, the website analyzer computing device 12 will take the No branch from step 208 to step 216. In step 216 in this iteration, the web site analyzer computing device 12 executes the data collector event handler 38(3) corresponding to the "plp" data collector event handler indicated in the job 610, as included based on the web service response 600, by passing the job 610 as a parameter.

[0062] As illustrated in FIG. 4, the data collector event handler 38(3) is configured to update the output resource to include the URL, name, and content from the job 610, as included in the definition of the job 610 in the web service response 600. Accordingly, the exemplary output resource 500 illustrated in FIG. 5 is updated to include the second through seventh lines of XML code, which correspond to the "http://acme.com/c2141/women_dresses" URL, "women dresses" name, and "Create the

perfect holiday wardrobe with 20% off swim and beachwear" description or content, and associated tags and XML code.

[0063] Referring back to FIG. 2, subsequent to executing the data collector event handler 38(3) indicated in the job 610, the web site analyzer computing device 12 again proceeds to step 204 and determines whether the stack is empty. Accordingly, the web site analyzer computing device 12 repeats steps 204, 206, 208, and 216, as described and illustrated earlier, for jobs 602, 604, 606, 608, and 612. Subsequent to processing jobs 602, 604, 606, 608, and 612, in another iteration, the web site analyzer computing device 12 will again determine in step 204 that the stack 300 is not empty and take the No branch to step 206.

[0064] In step 206 in this iteration, the web site analyzer computing device 12 extracts job 310 and performs steps 208 and 210 for job 310. In step 212 in this iteration, the "plp" web service is executing by passing the "http://acme.com/c1550" URL indicated in the job 310. The "plp" web service returns the web service response 700, as illustrated in FIG. 7A in this example, which includes four jobs including two start and two end jobs. Accordingly, in step 214 in this iteration, the web site analyzer computing device 12 inserts the four jobs 702, 704, 706, and 708 between start job 710 and end job 712, resulting in the stack 300 illustrated in FIG. 7B.

[0065] Accordingly, referring back to FIG. 2, the web site analyzer computing device 12 then proceeds to perform steps 204, 206, 208, and 216 for jobs 710, 702, 704, 706, 708, 712, and 312. Subsequent to executing the data collector event handler 38(2) indicated in the job 312 in step 216, the web site analyzer computing device 12 proceeds back to step 204. Since there are no pending jobs in the stack 300 illustrated in FIG. 7B, the web site analyzer computing device 12 will not execute any web services while processing jobs 710, 702, 704, 706, 708, 712, and 312, and will not therefore insert any new jobs into the stack 300.

[0066] Accordingly, in the next iteration subsequent to processing job 312, the web site analyzer computing device 12 will determine in step 204 that the stack 300 is empty and the Yes branch will be taken to step 218. In step 218, the web site analyzer computing

device 12 provides the output resource. The output resource 500 can be provided to a bot, a spider, or an indexer, for example, or any other type of application. Alternatively, as in the example describe and illustrated herein, the output resource 500 can be used to generate a mobile navigation menu, although different types of output resources can be generated and the output resource 500 can be used for different purposes or to facilitate different functionality in other examples.

[0067] Thus, as illustrated and described herein this technology provides a number of advantages including methods, non-transitory computer readable media, and web site analyzer computing devices that more effectively navigate web sites to collect information. With this technology, web services execute web pages in emulated JavaScript environments, which advantageously allows the web services to access and evaluate client-side JavaScript code and results in a more thorough and accurate inspection of the web pages.

[0068] Having thus described the basic concept of the invention, it will be rather apparent to those skilled in the art that the foregoing detailed disclosure is intended to be presented by way of example only, and is not limiting. Various alterations, improvements, and modifications will occur and are intended to those skilled in the art, though not expressly stated herein. These alterations, improvements, and modifications are intended to be suggested hereby, and are within the spirit and scope of the invention. Additionally, the recited order of processing elements or sequences, or the use of numbers, letters, or other designations therefore, is not intended to limit the claimed processes to any order except as may be specified in the claims. Accordingly, the invention is limited only by the following claims and equivalents thereto.

CLAIMS

What is claimed is:

1. A method for analyzing web sites using web services, the method comprising:

determining, by the web site analyzer computing device, when a job extracted from a stack is a pending job;

executing, by the web site analyzer computing device, a web service indicated in the job comprising passing a Uniform Resource Locator (URL) included in the job as a parameter to the web service, extracting another job from a web service response, and inserting the another job into the stack, when the determining indicates that the job is a pending job, wherein the web service is configured to obtain a web page to be analyzed based on the URL, execute the web page in an emulated JavaScript environment, and return the web service response; and

executing, by the web site analyzer computing device, a data collector event handler indicated in the job comprising passing the job as a parameter to the data collector event handler, when the determining indicates that the job is not a pending job, wherein the data collector event handler is configured to update an output resource based on content of the analyzed web page included in the job.

2. The method as set forth in claim 1, further comprising, when the determining indicates that the job is a pending job:

determining, by the web site analyzer computing device, when the job is a duplicate job based on a canonical URL included in the job;

storing, by the web site analyzer computing device, the canonical URL, when the determining indicates that the job is not a duplicate job; and

extracting, by the web site analyzer computing device, an additional job from the stack, without executing the web service indicated in the job, extracting the another job from the web service response, or inserting the another job into the stack, when the determining indicates that the job is a duplicate job.

3. The method as set forth in claim 1, further comprising, when the determining indicates that the job is a pending job, inserting, by the web site analyzer computing device, an end job into the stack before the another job and inserting a start job into the stack after the another job, wherein the start and end jobs identify another data collector event handler indicated in the web service response.

4. The method as set forth in claim 1, wherein the data collector event handler is a JavaScript function, the web service response comprises one or more of the another job, an indication of another data collector event handler, a description the web page, or a hash generated from the web page, and the job comprises a structured record and is:

a pending job comprising one or more of the indication of the web service, an indication of an endpoint of the web service, the URL of the web page to be analyzed, a name, or a canonical URL;

a start job comprising one or more of an indication of the data collector event handler, another name, or the content of the analyzed web page obtained based on the execution of the web page in the emulated JavaScript environment; or

an end job comprising one or more of the indication of the data collector event handler or the another name.

5. The method as set forth in claim 1, further comprising:

obtaining, by the web site analyzer computing device, a startup configuration file comprising one or more of an indication of the web service, the URL, a maximum number of web pages to process, or a function callback configured to validate the output resource;

generating, by the web site analyzer computing device, the job to have a pending type, an indication of the web service, and the URL; and

inserting, by the web site analyzer computing device, the job into the stack.

6. The method as set forth in claim 1, further comprising:

determining, by the web site analyzer computing device, when the stack is empty; and

providing, by the web site analyzer computing device, the output resource when the determining indicates that the stack is empty, wherein the output resource comprises an eXtensible Markup Language (XML) document, a JavaScript Object Notation (JSON) file, one or more records configured to be stored in a database, or an e-mail or other electronic communication.

7. A non-transitory computer readable medium having stored thereon programmed instructions for analyzing web sites using web services comprising executable code that, when executed by at least one processor, causes the processor to perform steps comprising:

determining when a job extracted from a stack is a pending job;

executing a web service indicated in the job comprising passing a Uniform Resource Locator (URL) included in the job as a parameter to the web service, extracting another job from a web service response, and inserting the another job into the stack, when the determining indicates that the job is a pending job, wherein the web service is configured to obtain a web page to be analyzed based on the URL, execute the web page in an emulated JavaScript environment, and return the web service response; and

executing a data collector event handler indicated in the job comprising passing the job as a parameter to the data collector event handler, when the determining indicates that the job is not a pending job, wherein the data collector event handler is configured to update an output resource based on content of the analyzed web page included in the job.

8. The non-transitory computer readable medium as set forth in claim 7, further having stored thereon one or more additional programmed instructions comprising executable code that, when executed by the processor further cause the processor to perform one or more additional steps comprising, when the determining indicates that the job is a

pending job:

determining when the job is a duplicate job based on a canonical URL included in the job;

storing the canonical URL, when the determining indicates that the job is not a duplicate job; and

extracting an additional job from the stack, without executing the web service indicated in the job, extracting the another job from the web service response, or inserting the another job into the stack, when the determining indicates that the job is a duplicate job.

9. The non-transitory computer readable medium as set forth in claim 7, further having stored thereon one or more additional programmed instructions comprising executable code that, when executed by the processor further cause the processor to perform one or more additional steps comprising, when the determining indicates that the job is a pending job, inserting an end job into the stack before the another job and inserting a start job into the stack after the another job, wherein the start and end jobs identify another data collector event handler indicated in the web service response.

10. The non-transitory computer readable medium as set forth in claim 7, wherein the data collector event handler is a JavaScript function, the web service response comprises one or more of the another job, an indication of another data collector event handler, a description the web page, or a hash generated from the web page, and the job comprises a structured record and is:

a pending job comprising one or more of the indication of the web service, an indication of an endpoint of the web service, the URL of the web page to be analyzed, a name, or a canonical URL;

a start job comprising one or more of an indication of the data collector event handler, another name, or the content of the analyzed web page obtained based on the execution of the web page in the emulated JavaScript environment; or

an end job comprising one or more of the indication of the data

collector event handler or the another name.

11. The non-transitory computer readable medium as set forth in claim 7, further having stored thereon one or more additional programmed instructions comprising executable code that, when executed by the processor further cause the processor to perform one or more additional steps comprising:

obtaining a startup configuration file comprising one or more of an indication of the web service, the URL, a maximum number of web pages to process, or a function callback configured to validate the output resource;

generating the job to have a pending type, an indication of the web service, and the URL; and

inserting the job into the stack.

12. The non-transitory computer readable medium as set forth in claim 7, further having stored thereon one or more additional programmed instructions comprising executable code that, when executed by the processor further cause the processor to perform one or more additional steps comprising:

determining when the stack is empty; and

providing the output resource when the determining indicates that the stack is empty, wherein the output resource comprises an eXtensible Markup Language (XML) document, a JavaScript Object Notation (JSON) file, one or more records configured to be stored in a database, or an e-mail or other electronic communication.

13. A web site analyzer computing device, comprising one or more processors coupled to a memory and configured to execute programmed instructions comprising and stored in the memory to:

determine when a job extracted from a stack is a pending job;

execute a web service indicated in the job and pass a Uniform Resource Locator (URL) included in the job as a parameter to the web service, extract another job from a web service response, and insert the another job into the stack, when the

determining indicates that the job is a pending job, wherein the web service is configured to obtain a web page to be analyzed based on the URL, execute the web page in an emulated JavaScript environment, and return the web service response; and

execute a data collector event handler indicated in the job and pass the job as a parameter to the data collector event handler, when the determining indicates that the job is not a pending job, wherein the data collector event handler is configured to update an output resource based on content of the analyzed web page included in the job.

14. The web site analyzer computing device as set forth in claim 13, wherein the processor are further configured to execute one or more additional programmed instructions comprising and stored in the memory to, when the determining indicates that the job is a pending job:

determine when the job is a duplicate job based on a canonical URL included in the job;

store the canonical URL, when the determining indicates that the job is not a duplicate job; and

extract an additional job from the stack, without executing the web service indicated in the job, extracting the another job from the web service response, or inserting the another job into the stack, when the determining indicates that the job is a duplicate job.

15. The web site analyzer computing device as set forth in claim 13, wherein the processor are further configured to execute one or more additional programmed instructions comprising and stored in the memory to, when the determining indicates that the job is a pending job, insert an end job into the stack before the another job and insert a start job into the stack after the another job, wherein the start and end jobs identify another data collector event handler indicated in the web service response.

16. The web site analyzer computing device as set forth in claim 13, wherein the data collector event handler is a JavaScript function, the web service response

comprises one or more of the another job, an indication of another data collector event handler, a description the web page, or a hash generated from the web page, and the job comprises a structured record and is:

a pending job comprising one or more of the indication of the web service, an indication of an endpoint of the web service, the URL of the web page to be analyzed, a name, or a canonical URL;

a start job comprising one or more of an indication of the data collector event handler, another name, or the content of the analyzed web page obtained based on the execution of the web page in the emulated JavaScript environment; or

an end job comprising one or more of the indication of the data collector event handler or the another name.

17. The web site analyzer computing device as set forth in claim 13, wherein the processor are further configured to execute one or more additional programmed instructions comprising and stored in the memory to:

obtain a startup configuration file comprising one or more of an indication of the web service, the URL, a maximum number of web pages to process, or a function callback configured to validate the output resource;

generate the job to have a pending type, an indication of the web service, and the URL; and

insert the job into the stack.

18. The web site analyzer computing device as set forth in claim 13, wherein the processor are further configured to execute one or more additional programmed instructions comprising and stored in the memory to:

determine when the stack is empty; and

provide the output resource when the determining indicates that the stack is empty, wherein the output resource comprises an eXtensible Markup Language (XML) document, a JavaScript Object Notation (JSON) file, one or more records configured to be stored in a database, or an e-mail or other electronic communication.

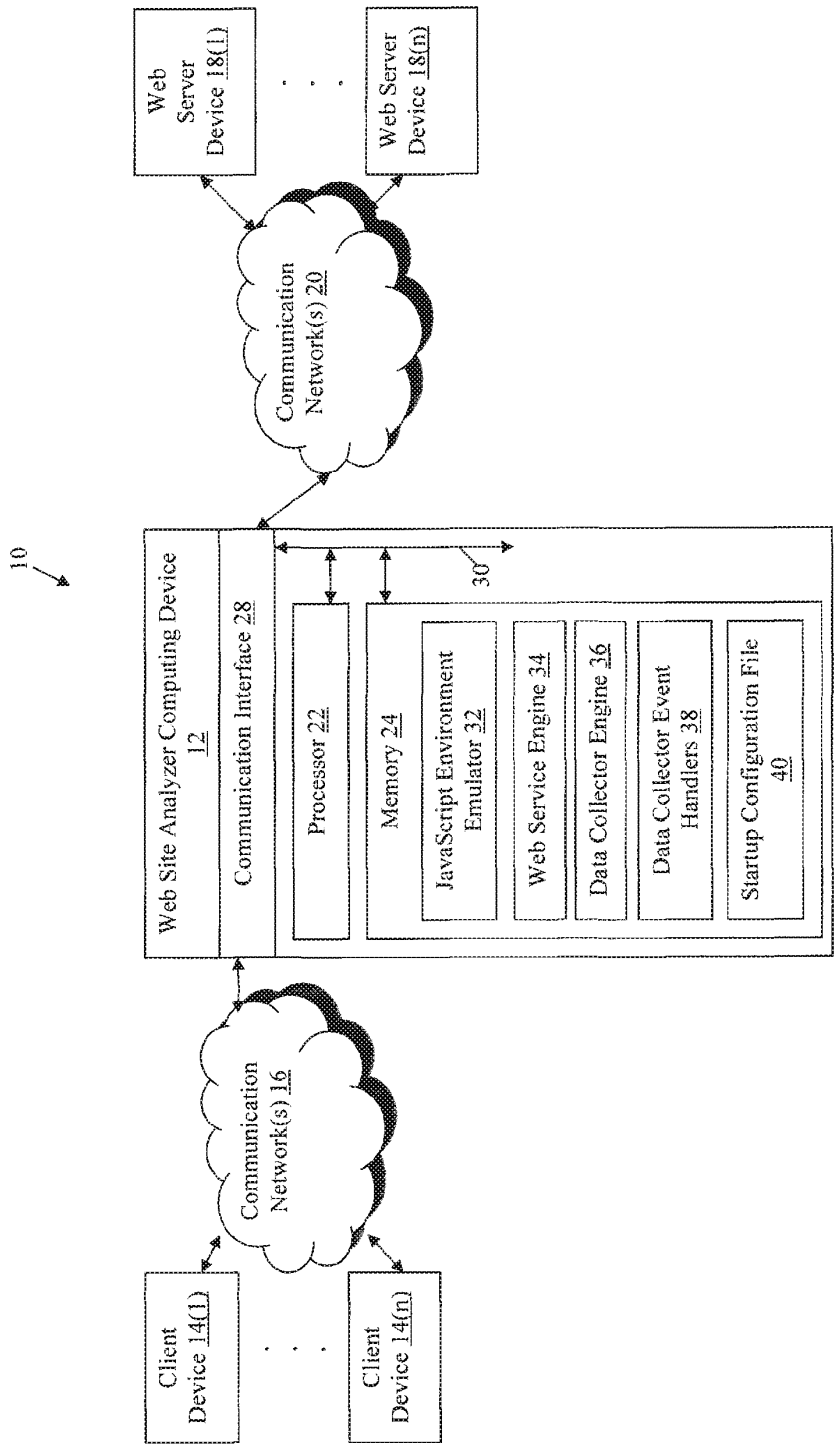

10
Web Site Analyzer Computing Device 12
Communication Interface 28
Processor 22
Memory 24
30
JavaScript Environment Emulator 32
Web Service Engine 34
Data Collector Engine 36
Data Collector Event Handlers 38
Startup Configuration File 40
Communication Network(s) 16
Client Device 14(1)
Client Device 14(n)
Communication Network(s) 20
Web Server Device 18(1)
Web Server Device 18(n)

FIG. 1

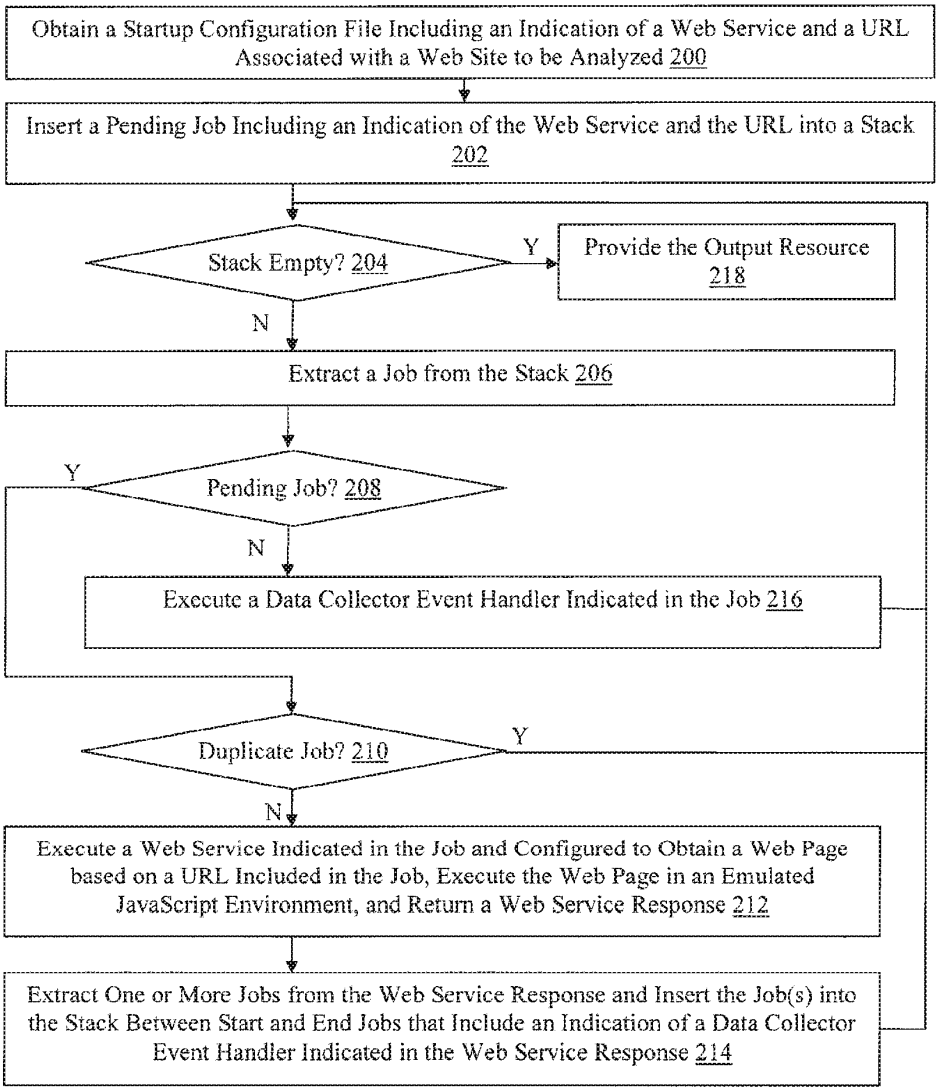

Obtain a Startup Configuration File Including an Indication of a Web Service and a URL Associated with a Web Site to be Analyzed 200
Insert a Pending Job Including an Indication of the Web Service and the URL into a Stack 202
Stack Empty? 204
Y
Provide the Output Resource 218
N
Extract a Job from the Stack 206
Pending Job? 208
Y
N
Execute a Data Collector Event Handler Indicated in the Job 216
Duplicate Job? 210
Y
N
Execute a Web Service Indicated in the Job and Configured to Obtain a Web Page based on a URL Included in the Job, Execute the Web Page in an Emulated JavaScript Environment, and Return a Web Service Response 212
Extract One or More Jobs from the Web Service Response and Insert the Job(s) into the Stack Between Start and End Jobs that Include an Indication of a Data Collector Event Handler Indicated in the Web Service Response 214

FIG. 2

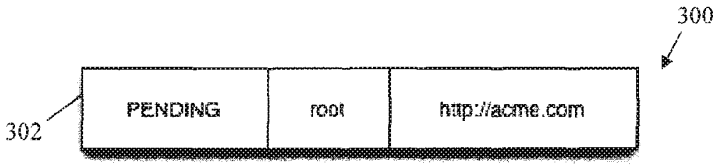


FIG. 3A


```
{
    "type": "home",
    "hash": "",
    "content": null,
    "jobs": [{
        "name": "women dresses",
        "type": "PENDING",
        "action": "plp",
        "url": "http://acme.com/c2141/women_dresses",
        "curl": "http://acme.com/c2141"
    }, {
        "name": "food & wine",
        "type": "PENDING",
        "action": "plp",
        "url": "http://acme.com/c1550/food_wine",
        "curl": "http://acme.com/c1550"
    }]
}
```


304

FIG. 3B

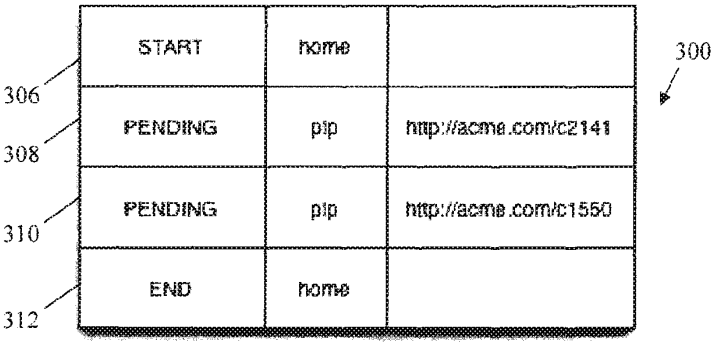


FIG. 3C


function home_start( ) {
out.write('<catalog>');
}
38(1)
function home_end( ) {
out.write('</catalog>');
}
38(2)
function plp_start( ) {
out.write('<plp url="' job.url '">');
out.write('<name>' job.name '</name>');
out.write('<description>' job.content '</description>');
}
38(3)
function plp_end( ) {
out.write('</catalog>');
}
38(4)
function pdp_start( ) {
out.write('<product>');
out.write('<name>' job.name '</name>');
(var i ; i job.content.sizes.length; i)
out.write('<size>' job.content.sizes[i] '</size>');
(job.content.bottles)
out.write('<bottles>' job.content.bottles '</bottles>');
out.write('<price>' job.content.price '</price>');
out.write('</product>');
}
38(5)
function pdp_end( ) {
}
38(6)

FIG. 4

500

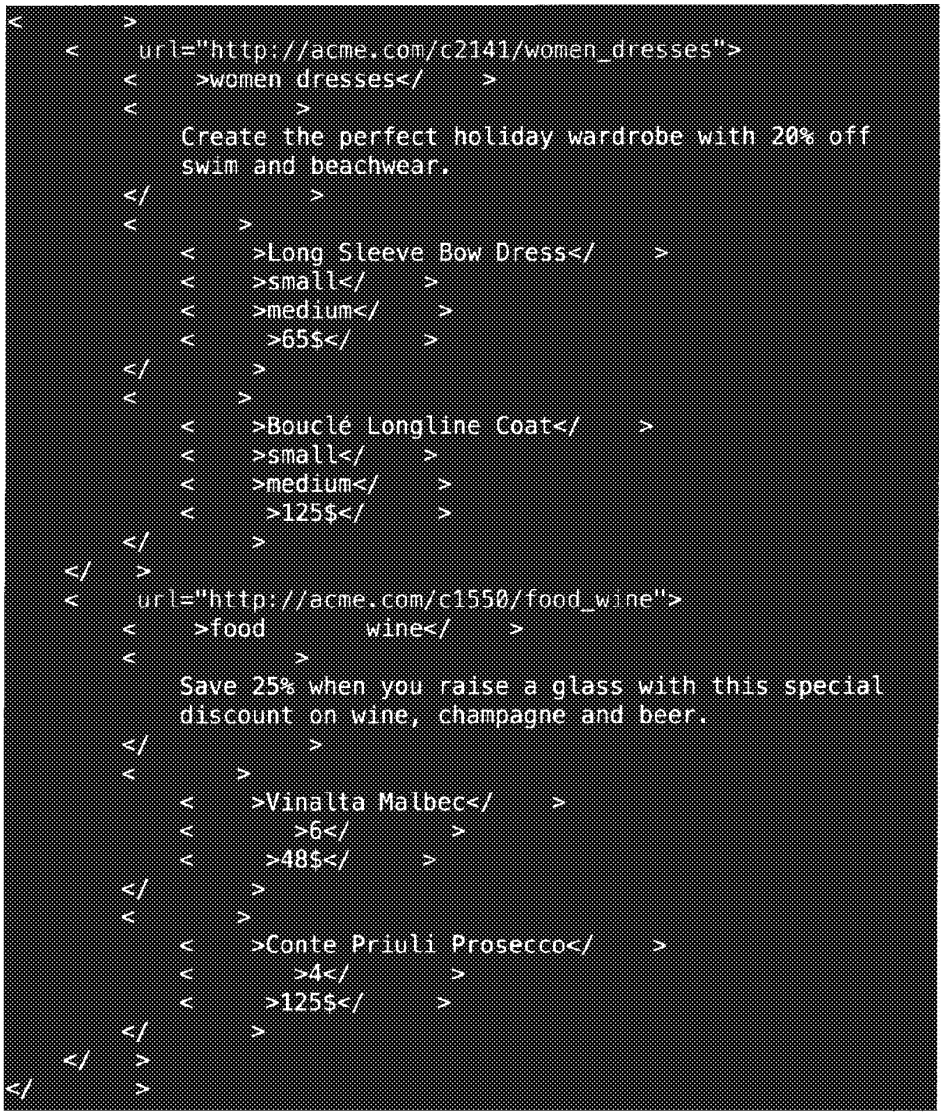

< >
< url="http://acme.com/c2141/women_dresses">
< >women dresses</ >
< >
Create the perfect holiday wardrobe with 20% off
swim and beachwear.
</ >
< >
< >Long Sleeve Bow Dress</ >
< >small</ >
< >medium</ >
< >65$</ >
</ >
< >
< >Bouclé Longline Coat</ >
< >small</ >
< >medium</ >
< >125$</ >
</ >
</ >
< url="http://acme.com/c1550/food_wine">
< >food wine</ >
< >
Save 25% when you raise a glass with this special
discount on wine, champagne and beer.
</ >
< >
< >Vinalta Malbec</ >
< >6</ >
< >48$</ >
</ >
< >
< >Conte Priuli Prosecco</ >
< >4</ >
< >125$</ >
</ >
</ >
</ >

Fig. 5

```
{
    "handler": "plp",
    "hash": "",
    "content": "Create the perfect holiday wardrobe with 20% off swim and beachwear.",
    "jobs": [{
        "name": "Long Sleeve Bow Dress",
        "type": "START",
        "handler": "pdp",
        "content": {
            "sizes": ["small", "medium"],
            "price": "65$"
        }
    }, {
        "name": "Long Sleeve Bow Dress",
        "type": "END",
        "handler": "pdp"
    }, {
        "name": "Bouclé Longline Coat",
        "type": "START",
        "handler": "pdp",
        "content": {
            "sizes": ["small", "medium"],
            "price": "125$"
        }
    }, {
        "name": "Bouclé Longline Coat",
        "type": "END",
        "handler": "pdp"
    }]
}
```


600

FIG. 6A

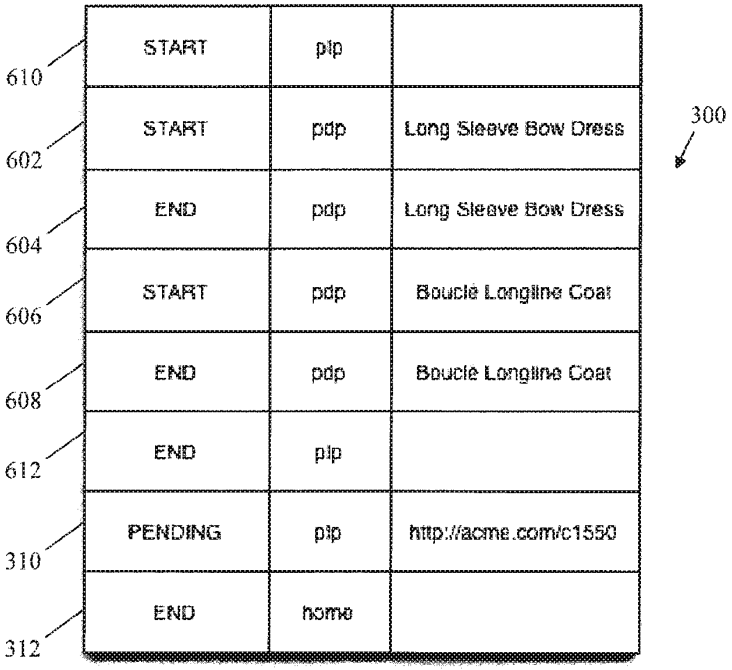


FIG. 6B

```
{
    "handler": "plp",
    "hash": "",
    "content": "Save 25% when you raise a glass with this special discount on wine, champagne and beer.",
    "jobs": [{
        "name": "Vinalta Malbec",
        "type": "START",
        "handler": "pdp",
        "content": {
            "price": "48$",
            "bottles": 6
        }
    }, {
        "name": "Vinalta Malbec",
        "type": "END",
        "handler": "pdp"
    }, {
        "name": "Conte Priuli Prosecco",
        "type": "START",
        "handler": "pdp",
        "content": {
            "price": "125$",
            "bottles": 4
        }
    }, {
        "name": "Conte Priuli Prosecco",
        "type": "END",
        "handler": "pdp"
    }]
}
```


700

FIG. 7A

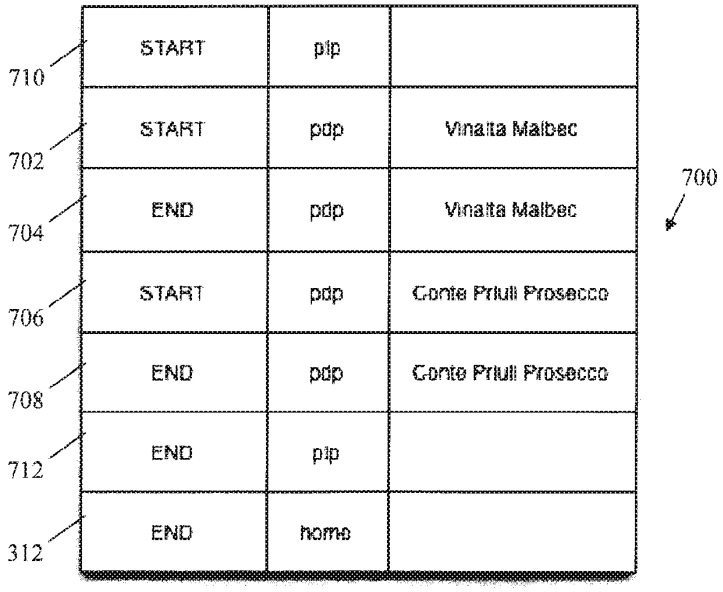


FIG. 7B

10
Client Device 14(1)
Client Device 14(n)
Communication Network(s) 16
Web Site Analyzer Computing Device 12
Communication Interface 28
Processor 22
Memory 24
30
JavaScript Environment Emulator 32
Web Service Engine 34
Data Collector Engine 36
Data Collector Event Handlers 38
Startup Configuration File 40
Communication Network(s) 20
Web Server Device 18(1)
Web Server Device 18(n)

Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2958235 A1 2016/03/03

(21) 2 958 235

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(86) Date de dépôt PCT/PCT Filing Date: 2015/06/19
(87) Date publication PCT/PCT Publication Date: 2016/03/03
(85) Entrée phase nationale/National Entry: 2017/02/15
(86) N° demande PCT/PCT Application No.: US 2015/036717
(87) N° publication PCT/PCT Publication No.: 2016/032602
(30) Priorité/Priority: 2014/08/25 (US14/467,210)

(51) Cl.Int./Int.Cl. *G06F 15/16* (2006.01)
(71) Demandeur/Applicant:
USABLENET INC., US

(72) Inventeur/Inventor:
SCODA, ENRICO, IT

(74) Agent: PARLEE MCLAWS LLP

(54) Titre : PROCEDES DE SYNCHRONISATION DE SESSIONS WEB ET DISPOSITIFS ASSOCIES
(54) Title: METHODS FOR SYNCHRONIZING WEB SESSIONS AND DEVICES THEREOF

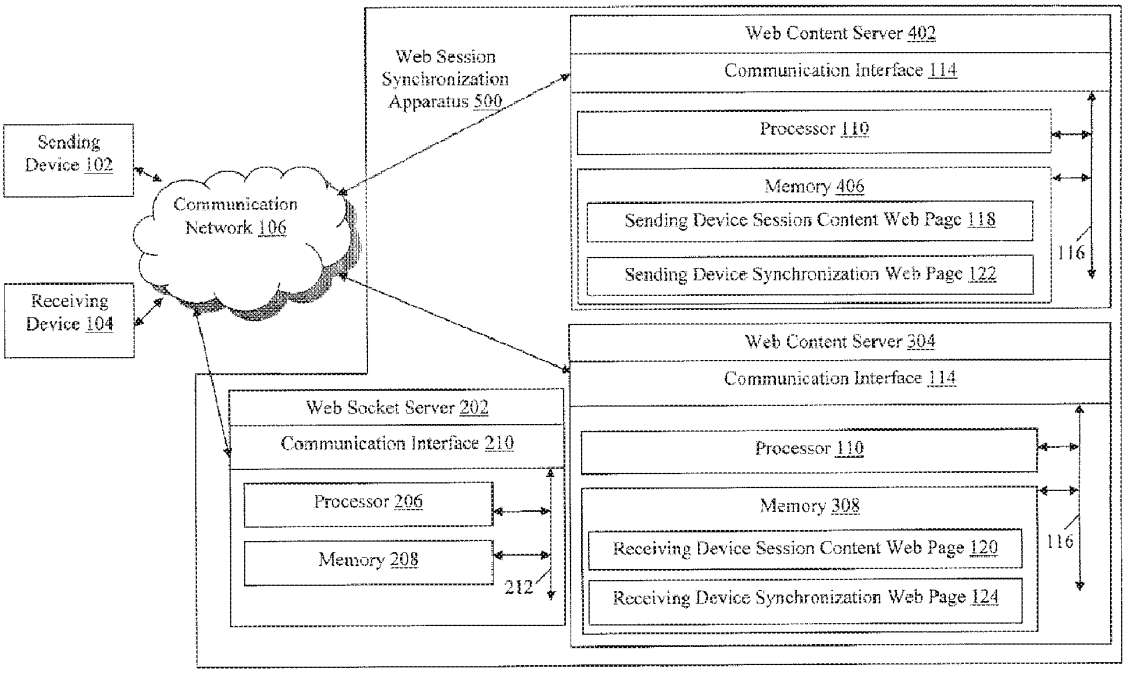


FIG. 5

(57) Abrégé/Abstract:
A first web socket connection is established with a sending device and a second web socket connection is established with a receiving device. The sending device is notified when the second web socket connection is established. One or more cookies

CA 2958235 A1 2016/03/03
(21) 2 958 235
(13) A1

(57) Abrégé(suite)/Abstract(continued):
including session information and a redirect uniform resource locator (URL) are received from the sending device in response to the notification and over the first web socket connection. The one or more cookies and the redirect URL are forwarded to the receiving device over the second web socket connection, wherein the redirect URL is associated with a web page that, when executed by the receiving device, is configured to comprise the session information.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
3 March 2016 (03.03.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/032602 A1

(51) International Patent Classification:
G06F 15/16 (2006.01)

(21) International Application Number:
PCT/US2015/036717

(22) International Filing Date:
19 June 2015 (19.06.2015)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
14/467,210 25 August 2014 (25.08.2014) US

(71) Applicant: **USABLENET INC.** [US/US]; 142 W. 57th Street, 7th Floor, New York, NY 10019 (US).

(72) Inventor: **SCODA, Enrico**; Via Cividina 416/3, I-33035 Martignacco Ud (IT).

(74) Agents: **GALLO, Nicholas, J.** et al.; LeClairRyan, A Professional Corporation, 70 Linden Oaks, Suite 210, Rochester, NY 04625 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

---- *with international search report (Art. 21(3))*

(54) Title: METHODS FOR SYNCHRONIZING WEB SESSIONS AND DEVICES THEREOF

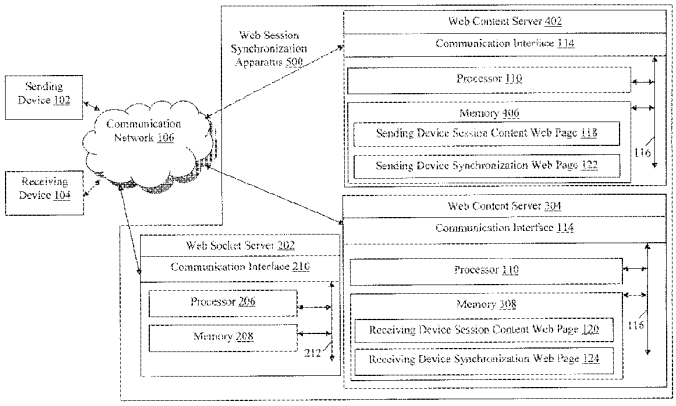


FIG. 5

(57) Abstract: A first web socket connection is established with a sending device and a second web socket connection is established with a receiving device. The sending device is notified when the second web socket connection is established. One or more cookies including session information and a redirect uniform resource locator (URL) are received from the sending device in response to the notification and over the first web socket connection. The one or more cookies and the redirect URL are forwarded to the receiving device over the second web socket connection, wherein the redirect URL is associated with a web page that, when executed by the receiving device, is configured to comprise the session information.

WO 2016/032602 A1

METHODS FOR SYNCHRONIZING WEB SESSIONS AND DEVICES THEREOF

FIELD

[0001] This technology generally relates to methods and devices for synchronizing web pages and, more particularly, to maintaining session data in web sessions synchronized between computing devices.

BACKGROUND

[0002] Many computing device users are increasingly using and switching between multiple computing devices such as mobile phones, tablets, and desktop computers. In order to improve the user experience, many devices are able to maintain continuity with respect to a web browsing experience. Accordingly, functionality, such as Handoff™ provided in recent operating systems on computing devices made available by Apple Inc. of Cupertino, California, has been developed to allow users to load a web page on one device and continue browsing the same web page by selecting an icon on another device. For example, a user may want to view a web page currently rendered on a mobile phone on a desktop computer instead, since the desktop computer may have a larger display. In another example, a user may want to switch from a tablet to a desktop computer to take advantage of the attached keyboard in order to input large amounts of data.

[0003] Currently, continuity is maintained by providing a receiving device with a uniform resource locator (URL) of the web page that the user wants to load on the receiving device. However, user web sessions are not currently transferable as associated session information is not maintained. Accordingly, the user experience is reduced for certain web pages associated with user sessions and having session information. For example, a user's web browsing experience cannot be effectively continued on a receiving device for a web page associated with a shopping cart and having session information such as the items the user has added to the shopping cart. Should the receiving device receive the URL for the shopping cart web page, the user session information will not be maintained and the user may have to repeat an operation, such as adding an item to the shopping cart, which is undesirable.

SUMMARY

[0004] A method for synchronizing web sessions includes receiving, with a web session synchronization apparatus, a request to establish a first web socket connection with a sending

device and establishing the first web socket connection in response to the request. A request to establish a second web socket connection is received, with the web session synchronization apparatus, from a receiving device and the second web socket connection is established with the receiving device in response to the request. The request to establish the second web socket connection also includes the synchronization identifier. The sending device is notified when the second web socket connection is established. One or more cookies including session information are received, with the web session synchronization apparatus, as well as a redirect uniform resource locator (URL) from the sending device in response to the notification and over the first web socket connection. The one or more cookies and the redirect URL are forwarded, with the web session synchronization apparatus, to the receiving device over the second web socket connection, wherein the redirect URL is associated with a web page that, when executed by the receiving device, is configured to comprise the session information.

[0005] A web session synchronization apparatus includes a processor coupled to a memory and configured to execute programmed instructions stored in the memory, including receiving a request to establish a first web socket connection with a sending device and establishing the first web socket connection in response to the request. A request to establish a second web socket connection is received from a receiving device and the second web socket connection is established with the receiving device in response to the request. The request to establish the second web socket connection also includes the synchronization identifier. The sending device is notified when the second web socket connection is established. One or more cookies including session information are received as well as a redirect uniform resource locator (URL) from the sending device in response to the notification and over the first web socket connection. The one or more cookies and the redirect URL are forwarded to the receiving device over the second web socket connection, wherein the redirect URL is associated with a web page that, when executed by the receiving device, is configured to comprise the session information.

[0006] A non-transitory computer readable medium having stored thereon instructions for synchronizing web sessions comprising machine executable code which when executed by a processor, causes the processor to perform steps including receiving a request to establish a first web socket connection with a sending device and establishing the first web socket connection in response to the request. A request to establish a second web socket connection is received from a receiving device and the second web socket connection is established with the receiving device in response to the request. The request to establish the second web socket connection also includes the synchronization identifier. The sending device is notified when the second web

socket connection is established. One or more cookies including session information are received as well as a redirect uniform resource locator (URL) from the sending device in response to the notification and over the first web socket connection. The one or more cookies and the redirect URL are forwarded to the receiving device over the second web socket connection, wherein the redirect URL is associated with a web page that, when executed by the receiving device, is configured to comprise the session information.

[0007] This technology provides a number of advantages including methods, non-transitory computer readable media, and devices that facilitate synchronization of web pages with session information between computing devices. This technology allows users to maintain continuity of a web browsing experience between computing devices. By maintaining session information, a user's web browsing experience across computing devices is improved since the user will not have to repeat operations performed within a web site and, instead, can continue a web browsing experience without loss of session information. Additionally, web socket connections are advantageously used to facilitate synchronization of a web session, without requiring that session information be stored server-side, thereby providing increased security.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] FIG. 1 is a block diagram of a network environment with an exemplary web session synchronization apparatus with sending and receiving device session content and synchronization web pages and a web socket server;

[0009] FIG. 2 is a block diagram of a network environment with an exemplary web session synchronization apparatus with a web socket server and a web content server with sending and receiving device session content and synchronization web pages;

[0010] FIG. 3 is a block diagram of a network environment with an exemplary web session synchronization apparatus with a web content server with a web socket server, sending device session content web page, and sending device synchronization web page and another web content server with a receiving device session content web page and receiving device synchronization web page;

[0011] FIG. 4 is a block diagram of a network environment with an exemplary web session synchronization apparatus with a web content server with a web socket server, receiving device session content web page, and receiving device synchronization web page and another

web content server with a sending device session content web page and sending device synchronization web page;

[0012] FIG. 5 is a block diagram of a network environment with an exemplary web session synchronization apparatus with a web socket server, a web content server with a receiving device session content web page, and receiving device synchronization web page, and another web content server with a sending device session content web page and sending device synchronization web page;

[0013] FIG. 6 is a flow chart of an exemplary method for initiating transfer of a web session by a sending device to a receiving device;

[0014] FIG. 7 is an exemplary sending device session content web page;

[0015] FIG. 8 is an exemplary sending device synchronization web page;

[0016] FIG. 9 is a flow chart of an exemplary method of sending session information to a receiving device using a connection between a sending device and a web socket server;

[0017] FIG. 10 is an exemplary receiving device synchronization web page;

[0018] FIG. 11 is a flow chart of an exemplary method of receiving by a receiving device session information using a connection with a web socket server;

[0019] FIG. 12 is an exemplary receiving device session content web page; and

[0020] FIG. 13 is a flow chart of an exemplary method of facilitating synchronization of a web session between sending and receiving devices by a web socket server.

DETAILED DESCRIPTION

[0021] An exemplary network environment with a web session synchronization apparatus 100 coupled to a sending device 102 and a receiving device 104 is illustrated in FIG. 1. In this example, the web session synchronization apparatus 100, sending device 102 and receiving device 104 are coupled together by at least one communication network 106, although other numbers and types of systems, devices, and/or elements in other configurations or network topologies can also be used. This technology provides a number of advantages including methods, non-transitory computer readable media, and devices that securely facilitate

synchronization of web pages associated with web sessions between computing devices while maintaining session information and thereby improving the user experience.

[0022] The web session synchronization apparatus 100 in this example includes a web content server 108 coupled to the sending device 102 and receiving device 104 by the communication network 106, which can include one or more local area network(s) (LANs) and/or wide area network(s) (WANs). Other network devices configured to generate, send, and receive network communications and coupled together via other topologies can also be used. While not shown, the network environment also may include additional network components, such as routers, switches and other devices, which are well known to those of ordinary skill in the art and thus will not be described herein.

[0023] The web session synchronization apparatus 100 may perform any number of functions including hosting and providing web pages and facilitating synchronization of the web pages, including any associated web session information, between the sending device 102 and the receiving device 104 using web socket connections. In this example, the web content server 108 includes a processor 110, a memory 112, and a communication interface 114, which are coupled together by a bus 116 or other link, although other numbers and types of components, parts, devices, systems, and elements in other configurations and locations can be used.

[0024] The processor 110 in the web content server 108 executes a program of stored instructions one or more aspects of the present invention, as described and illustrated by way of the embodiments herein, although the processor 110 could execute other numbers and types of programmed instructions. The processor 110 of the web content server 108 may include one or more central processing units or general purpose processors with one or more processing cores, for example.

[0025] The memory 112 in the web content server 108 stores these programmed instructions for one or more aspects of the present invention, as described and illustrated herein, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, DVD ROM, or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to the processor 110, can be used for the memory 112 in the web content server 108. The memory 112 in this example stores a plurality of web pages including at least one sending device session content web page 118, receiving

device session content web page 120, sending device synchronization web page 122, and receiving device synchronization web page 124 as well as a web socket server 126.

[0026] Optionally, the sending device session content web page 118 and receiving device session content web page 120 can be the same web page or a different version of the same web page. In the first case, the web server can deliver both desktop and mobile pages, for example, using the same URL (e.g., using responsive web design or adaptive web design). In the second case, the web server hosts two web sites: one for desktop web page versions and one for mobile web page versions, which can be located at different URLs) For example, if the sending device 102 and receiving device 104 are both desktop computers, the sending device session content web page 118 and receiving device session content web page 120 can be the same web page, such as a desktop version of a shopping cart web page for a web site. In another example in which the sending device 102 is a mobile phone and the receiving device 104 is a desktop computer, the sending device session content web page 118 can be a mobile version of the shopping cart web page for the web site and the receiving device session content web page 120 can be a desktop version of the shopping cart web page for the web site.

[0027] Additionally, the sending device synchronization web page 122 and the receiving device synchronization web page 124 can be the same web page. For example, the web content server 108 can inject JavaScript code into a same synchronization web page to provide certain functionality of the sending device synchronization web page 122 or the receiving device synchronization web page 124 depending on whether a cookie is present in the HTTP request for the synchronization web page, as described and illustrated in more detail later. In another example, a same synchronization web page can be stored in the memory 112 as including the certain functionality of both the sending device synchronization web page 122 and the receiving device synchronization web page 124 with the synchronization web page determining the behavior based on the presence of a cookie, also as described and illustrated in more detail later. Other permutations of the web pages 118, 120, 122, and 124 with other functionality can also be used and other web pages can be provided in the memory 112.

[0028] The memory 112 of the web content server 108 also includes a web socket server 126 in this example, although one or more of the web pages 118, 120, 122, and 124 and/or the web socket server 126 can be provided elsewhere in the network environment, such as described and illustrated later with reference to FIGS. 2-5, for example. The web socket server 126 in this example is a software module that includes programmed instructions that, when executed by the processor 110, generate a web socket server configured to facilitate communications over web

socket connections between the sending device 102 and the receiving device 104 according to the web socket protocol, as described and illustrated in more detail later.

[0029] The communication interface 114 in the web content server 108 is used to operatively couple and communicate between the w web content server 108, the sending device 102, and the receiving device 104, which are all coupled together via the communication network 106, although other types and numbers of communication networks or systems with other types and numbers of connections and configurations to other devices and elements can also be used. By way of example only, the communication network can use TCP/IP over Ethernet and industry-standard protocols, including hypertext transfer protocol (HTTP), and/or secure HTTP (HTTPS), although other types and numbers of communication networks, such as a direct connection, modems and phone lines, e-mail, and wireless and hardwire communication technology, each having their own communications protocols, can be used.

[0030] The sending device 102 and the receiving device 104 in this example enable a user to request, receive, interact with, and synchronize applications and web pages hosted by the web session synchronization apparatus 100 and using the communication network 106, although one or more of the sending device 102 or receiving device 104 could access content and utilize other types and numbers of applications from other sources and could provide a wide variety of other functions for a user. The sending device 102 and receiving device 104 can be the same type of computing device (e.g., mobile phone or desktop computer) or the sending device 102 and receiving device 104 can be different types of devices.

[0031] Each of the sending device 102 and receiving device 104 in this example includes a processor, a memory, an input device, a display device, and a communication interface, which are coupled together by a bus or other link, although one or more of sending device 102 or remote device 104 can include other numbers and types of components, parts, devices, systems, and elements in other configurations. The processor in each of the sending device and receiving device can execute a program of instructions stored in the memory for one or more aspects of the present invention, as described and illustrated herein, although the processor could execute other numbers and types of programmed instructions.

[0032] The input device in each of the sending device 102 and receiving device 104 can be used to input selections, such as a request for a particular web page or other content stored by the web session synchronization apparatus, although the input device could be used to input other types of requests and data and interact with other elements. The input device can include

keypads, touch screens, and/or vocal input processing systems, although other types and numbers of input devices can also be used.

[0033] The display device in each of the sending device 102 and receiving device 104 can be used to show data and information to a user, such as web pages and other content retrieved from the web session synchronization apparatus 100 by way of example only. The display device in one or more of the sending device 102 and receiving device 104 can be a television screen, a mobile phone screen display, a laptop screen, a tablet screen, or a monitor for example, although other types and numbers of displays could be used depending on the particular type of sending device 102 and receiving device 104. The communication interface in each of the sending device 102 and receiving device 104 can be used to operatively couple and communicate between the sending device 102 and receiving device 104 and the web session synchronization apparatus 100 over the communication network 106.

[0034] Referring to FIG. 2 another exemplary network environment with an exemplary web session synchronization apparatus 200, the sending device 102, the receiving device 104, and the communication network 106 is illustrated. In this example, the web session synchronization apparatus 200 includes a web socket server 202 and web content server 204 provided as separate devices. The web socket server 202 in this example includes a processor 206, a memory 208, and a communication interface 210 coupled together by a bus 212 or other link, although other numbers and types of components, parts, devices, systems, and elements in other configurations and locations can be used. The web socket server 202 performs the same functions as described and illustrated earlier with reference to the web socket server 124 but is a hardware device separate from any web content server.

[0035] The processor 206 in the web socket server 202 executes a program of stored instructions one or more aspects of the present invention, as described and illustrated by way of the embodiments herein, although the processor 206 could execute other numbers and types of programmed instructions. The processor 206 of the web socket server 202 may include one or more central processing units or general purpose processors with one or more processing cores, for example.

[0036] The memory 208 in the web socket server 202 stores these programmed instructions for one or more aspects of the present invention, as described and illustrated herein, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM)

or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, DVD ROM, or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to the processor, can be used for the memory 208 in the web socket server 202.

[0037] The communication interface 210 in the web socket server 202 is used to operatively couple and communicate between the web socket server 202, the sending device 102, and the receiving device 104, which are all coupled together via the communication network 106, although other types and numbers of communication networks or systems with other types and numbers of connections and configurations to other devices and elements can also be used.

[0038] The web content server 204 in this example includes a processor 110, a memory 214, and a communication interface 114 coupled together by a bus 116 or other link, although other numbers and types of components, parts, devices, systems, and elements in other configurations and locations can be used. The web content server 204 may perform any number of functions including hosting and providing web pages in response to requests received from the sending device 102 and receiving device 104. The web content server 204 is the same as the web content server 108 except that the memory 214 does not include web socket server 124 and instead only includes the sending device session content web page 116, receiving device session content web page 118, and sending device synchronization web page 120, and receiving device synchronization web page 122, although the sending device session content web page 116 and receiving device session content web page 118 could be the same session content web page and the sending device synchronization web page 120 and receiving device synchronization web page 122 could also be the same synchronization device web page, as described and illustrated earlier.

[0039] Referring to FIG. 3 another exemplary network environment with an exemplary web session synchronization apparatus 300, the sending device 102, the receiving device 104, and the communication network 106 is illustrated. In this example, the web session synchronization apparatus 300 includes a web content server 302 and another web content server 304. The web content servers 302 and 304 are the same as the web content servers 108 and 204 except that the memory 306 of the we content server 302 include the web socket server 124, sending device session content web page 118, and sending device synchronization web page 122 and the memory 308 of the separate web content server 304 includes the receiving device session content web page 120 and the receiving device synchronization web page 124. Accordingly, in this example, the sending device and the receiving device could be different types of devices

(e.g., a mobile phone and a desktop computer) and the sending device session content web page 118 and receiving device session content web page 120 are different versions of the same web page but hosted on different content servers.

[0040] Referring to FIG. 4 another exemplary network environment with an exemplary web session synchronization apparatus 400, the sending device 102, the receiving device 104, and the communication network 106 is illustrated. In this example, the web session synchronization apparatus 400 includes a web content server 402 and another web content server 404. The web content servers 402 and 404 are the same as the web content servers 108, 204, 302, and 304 except that the memory 406 of the web content server 302 include only the sending device session content web page 118 and sending device synchronization web page 122 and the memory 408 of the separate web content server 404 includes the web socket server 124, the receiving device session content web page 120, and the receiving device synchronization web page 124. Accordingly, in this example as in the example described and illustrated earlier with reference to FIG. 3, the sending device and the receiving device could be different types of devices.

[0041] Referring to FIG. 5 another exemplary network environment with an exemplary web session synchronization apparatus 400, the sending device 102, the receiving device 104, and the communication network 106 is illustrated. In this example, the web session synchronization apparatus 400 includes the web content server 402, the web content sever 304, and the web socket server 202, all as separate devices. Accordingly, in this example as in the example described and illustrated earlier with reference to FIG. 3, the sending device and the receiving device could be different types of devices. In other examples, the web session synchronization apparatus 100, 200, 300, 400, and/or 500 can include different permutations of the web content server 108, 204, 302, 304, 402, or 404 and/or web socket server 124 or 202 and each of the web content servers 108, 204, 302, 304, 402, and 404 can store different web pages and content.

[0042] Although embodiments of the web session synchronization apparatus 100, 200, 300, 400, and 500, sending device 102, and receiving device 104 are described and illustrated herein, each of the web session synchronization apparatus 100, 200, 300, 400, and 500, sending device 102, and receiving device 104 can be implemented on any suitable computer apparatus or computing device. It is to be understood that the apparatuses and devices of the embodiments described herein are for exemplary purposes, as many variations of the specific hardware and software used to implement the embodiments are possible, as will be appreciated by those skilled

in the relevant art(s). Furthermore, each of the devices of the embodiments may be conveniently implemented using one or more general purpose computers, microprocessors, digital signal processors, and micro-controllers, programmed according to the teachings of the embodiments, as described and illustrated herein, and as will be appreciated by those ordinary skill in the art.

[0043] In addition, two or more computing apparatuses or devices can be substituted for any one of the devices in any embodiment described herein. Accordingly, principles and advantages of distributed processing, such as redundancy and replication also can be implemented, as desired, to increase the robustness and performance of the devices of the embodiments. The embodiments may also be implemented on computer apparatuses or devices that extend across any suitable network using any suitable interface mechanisms and communications technologies, including by way of example only telecommunications in any suitable form (e.g., voice and modem), wireless communications media, wireless communications networks, cellular communications networks, G3 communications networks, Public Switched Telephone Network (PSTNs), Packet Data Networks (PDNs), the Internet, intranets, and combinations thereof.

[0044] The embodiments may also be embodied as one or more non-transitory computer readable medium having instructions stored thereon for one or more aspects of the present invention as described and illustrated by way of the embodiments herein, as described herein, which when executed by a processor, cause the processor to carry out the steps necessary to implement the methods of the embodiments, as described and illustrated herein.

[0045] An exemplary method for synchronizing web sessions will now be described with reference to FIGS. 1-13. For purposes of this example only, the network environment described and illustrated with reference to FIG. 5 will be used, although any of the network environments described and illustrated with reference to FIGS. 1-4 can also be used in other examples. Additionally, in this example, the sending device 102 is a mobile device and the receiving device 104 is a desktop computer, although, as described and illustrated earlier, the sending device 102 and receiving device 104 could also be any other types of devices as well as the same type of device.

[0046] Referring more specifically to FIG. 6, an exemplary method for initiating transfer of a web session by a sending device to a receiving device is illustrated. In step 600, the sending device 102 obtains and executes, in a web browser for example, the sending device session content web page 118, which is associated with a web session. The sending device session

content web page 118 can be a mobile shopping cart web page associated with a web session for the user and having session content, such as an item added to the shopping cart, for example. Upon obtaining and executing the sending device session content web page 118, the web browser of the sending device 102 sets one or more cookies including information corresponding to the session content, such as an indication of the item added to the shopping cart in this example.

[0047] Referring more specifically to FIG. 7, an exemplary sending device session content web page 118 is illustrated. In this example, the session content includes at least the daily dental dog treat item added to the shopping cart for the web site, although the session content can include other information (e.g., number of items, number of each item, price, other user information) and the web session can be associated with types of web pages other than a shopping cart web page for a web site.

[0048] In this example, the sending device session content web page 118 corresponds with a Uniform Resource Locator (URL), such as "https://m.acme.com/cart", although any other type of web page associated with a web session and located at any other URL can also be used. Accordingly, the web content server 402 in this example can be hosting mobile versions of various web pages, including the sending device session content web page 118, that are associated with various mobile sites.

[0049] Referring back to FIG. 6, in step 602, the sending device 102 receives a request to synchronize the web session with the receiving device 104. In this example, the sending device session content web page 118 includes a transfer cart to desktop button 702. Accordingly, the sending device session content web page 118 can receive the request to initiate synchronization of the web session upon user interaction with the transfer cart to desktop button 702, although other types of user inputs can be used and the request to synchronize the web session can be generated in other ways.

[0050] In step 604, the sending device 102 generates a synchronization identifier and, optionally, a cookie with a value of the synchronization identifier. Accordingly, interaction by the user with the transfer cart to desktop button 702 of the sending device session content web page 118 in this example can cause the web browser of the sending device 102 to execute JavaScript code of the sending device session content web page 118 to generate the synchronization identifier (e.g., "3823329234") and, optionally, set the cookie having the value of the synchronization identifier. The cookie with the value of the synchronization identifier is

optional and not required in this example since the sending device synchronization web page 122 and receiving device synchronization web page 124 are different web pages provided by different web content servers 304 and 402. However, in examples in which the sending device synchronization web page 122 and receiving device synchronization web page 124 are the same web page having both sending device and receiving device synchronization functionality, or having sending device or receiving device synchronization functionality injected by a web content server, the cookie with the value of the synchronization identifier is required, as described and illustrated in more detail later.

[0051] In step 606, the sending device 102 processes a redirect based on a URL of the sending device synchronization web page 122. Accordingly, the JavaScript code executed by the web browser of the sending device 102, upon user interaction with the transfer cart to desktop button 702 in this example, can be configured to initiate the redirect based on a URL of the sending device synchronization web page 122 that includes the synchronization identifier. In this example, the URL of the sending device synchronization web page 122 can be "https://m.acme.com/sync?id=3823329234", although any other URL that includes the synchronization identifier can also be used.

[0052] In other examples in which the cookie set in step 604 is required, the request for the sending device synchronization web page 122 sent to a web content server as part of processing the redirect includes the cookie with the synchronization identifier. Since, in these examples, the sending device synchronization web page 122 and receiving device synchronization web page 124 may be identified based on the same URL, the web content server can determine which synchronization web page to provide in response to the request based on whether the cookie with the synchronization identifier is included in the request. If the cookie with the synchronization identifier is present, then the web content server will respond with the sending device synchronization web page 122, or a synchronization web page injected with JavaScript code having the sending device synchronization functionality described and illustrated later with reference to FIG. 9. In other examples, the web content server will send the same synchronization web page in response to the request, but the JavaScript code will determine whether the sending or receiving device synchronization functionality is executed based on the cookie, as described and illustrated in more detail later.

[0053] Referring more specifically to FIG. 8, an exemplary sending device synchronization web page 122 is illustrated. In this example, the sending device synchronization web page 122 includes text indicating that the process of synchronizing the web session has been

initiated, although the sending device synchronization web page 122 can include any other text or content. The sending device synchronization web page 122 is obtained and executed by the web browser of the sending device 102 in step 606 in this example as part of processing the redirect.

[0054] Referring more specifically to FIG. 9, an exemplary method for sending session information to the receiving device 104 using a connection between the sending device 102 and the web socket server 202 is illustrated. The method described and illustrated with reference to FIG. 9 can be performed by the sending device 102 executing JavaScript code of the sending device synchronization web page 122. In examples in which the sending device synchronization web page 122 and receiving device synchronization web page 124 are the same synchronization web page, with both sending and receiving device synchronization functionality, the JavaScript code of the synchronization web page can be configured to determine whether the cookie with the synchronization identifier, as set in step 604 as described and illustrated earlier with reference to FIG. 6, was returned with the response that included the synchronization web page. If the cookie with the synchronization identifier was included in the response that included the synchronization web page, then the JavaScript code of the synchronization web page implements the sending device synchronization functionality described and illustrated with reference to FIG. 9.

[0055] Accordingly, in step 900 in this example, the sending device 102 executing the sending device synchronization web page 122 sends a request including the synchronization identifier to the web socket server 202 and establishes a web socket connection with the web socket server 202. The synchronization identifier can be obtained from the URL used to obtain the sending device synchronization web page 122 or from a cookie, such as the cookie optionally set as described and illustrated earlier with reference to step 604 of FIG. 6, for example. In other examples, the web socket server 202 can be a module of a web content server, such as described and illustrated earlier with reference to web content servers 302 or 404, or of a single web content server, such as described and illustrated earlier with reference to web content servers 108 or 204.

[0056] In step 902, the sending device 102 executing the sending device synchronization web page 122 determines whether a connection has been established with the receiving device 104. A connection is established with the receiving device 104 as described and illustrated in more detail later with reference to FIG. 11. However, upon the web socket server 202 establishing a connection with the receiving device 104, the web socket server 202 sends a

confirmation message to the sending device 102 using the connection established in step 900. Accordingly, in step 902, the sending device 102 determines whether the confirmation message confirming establishment of a connection between the web socket server 202 and the receiving device 104 has been received. If the sending device 102 determines that the connection with the receiving device 104 has not been established, then the No branch is taken back to step 902 and the sending device 102 effectively waits until the connection with the receiving device 104 is established.

[0057] However, if the sending device 102 determines in step 902 that the connection has been established with the receiving device 104, then the Yes branch is taken to step 904. In step 904, the sending device 102 executing the sending device synchronization web page 122 sends one or more cookies with session information and a redirect URL to the web socket server 202 over the connection established in step 900. The one or more cookies with the session information could have been set upon establishing the sending device session content web page 118, as described and illustrated in more detail earlier with reference to step 600 of FIG. 6. The redirect URL can be included in the JavaScript code of the sending device synchronization web page 122 and can be associated with the receiving device session content web page 120, which in this example is located at "https://www.acme.com/cart" since the receiving device 104 is a desktop computer. However, the sending device 102 may not be aware of the type of receiving device 104 and so the redirect URL can also be "https://m.acme.com/cart", which would cause the web browser of the receiving device 104 to redirect to "https://www.acme.com/cart" in this example.

[0058] In step 906, the sending device 102 executing the sending device synchronization web page 122 determines whether an end of communication message has been received from the web socket server 202 indicating that the one or more cookies with session information and redirect URL was successfully sent to the receiving device 104. If the sending device 102 determines that an end of communication message has not been received then the No branch is taken back to step 906 and the sending device 102 effectively waits for the end of communication message. In other examples, the sending device 102 can abort, resend the one or more cookies with session information and the redirect URL, or take another action.

[0059] However, if the sending device 102 determines in step 906 that an end of communication message has been received, then the Yes branch is taken to step 908. In step 908, the sending device 102 executing the sending device synchronization web page 122 generates a redirect to return to the sending device session content web page 118, which is

processed by the web browser of the sending device 102. Accordingly, the JavaScript code of the sending device synchronization web page 122 generates a redirect to the sending device session content web page 118 located at the "https://m.acme.com/cart" URL in this example, although other actions can also be performed by the sending device 102 in step 908. Accordingly, subsequent to processing the redirect in step 908, the web browser of the sending device 102 returns to the web page rendered prior to and at the time of the user interaction with the transfer cart to desktop button 702 that initiated the synchronization in this example.

[0060] Referring more specifically to FIG. 10, an exemplary receiving device synchronization web page 124 is illustrated. Subsequent to the redirect being processed by the web browser of the sending device 102, as described and illustrated earlier with reference to step 606 of FIG. 6, the user of the sending device 102 initiates the obtaining and executing of the receiving device synchronization web page 124 by the receiving device 104. The user causes the web browser of the receiving device 104 to obtain and execute the receiving device synchronization web page 124 by manually entering the URL of the sending device synchronization web page 122, which is rendered by the web browser of the sending device 102 after processing the redirect in step 606, into the web browser of the receiving device 104 or using a Handoff™ feature, for example, although other methods can also be used.

[0061] In this example, the web browser of the receiving device will attempt to obtain and execute a web page based on the "https://m.acme.com/sync?id=3823329234" URL of the sending device synchronization web page 122. Since the receiving device 104 is a desktop computer in this example, the web content server 402 will receive the request, recognize the user-agent header of the request as being associated with a desktop web browser, and generate a redirect to the desktop version of the receiving device synchronization web page 124 hosted by the web content server 304 in this example. In this example, the receiving device synchronization web page 124 includes text indicating that the process of synchronizing the web session has been initiated, although the receiving device synchronization web page 124 can include any other text or content.

[0062] Referring more specifically to FIG. 11, an exemplary method for receiving by the receiving device 104 session information using a connection with the web socket server 202 is illustrated. The method described and illustrated with reference to FIG. 11 can be performed by the receiving device 104 executing JavaScript of the receiving device synchronization web page 124, for example. In examples in which the cookie set in step 604 is required, as described and illustrated earlier with reference to FIG. 6, the request for the receiving device synchronization

web page 124 sent to a web content server, as part of obtaining the receiving device synchronization web page 124 by the receiving device 104, will not include the cookie with the synchronization identifier. As described and illustrated earlier, if the cookie with the synchronization identifier is not present, then a web content server will respond with the receiving device synchronization web page 124, or a synchronization web page injected with JavaScript code having the receiving device synchronization functionality described and illustrated with reference to FIG. 11.

[0063] In examples in which the sending device synchronization web page 122 and receiving device synchronization web page 124 are the same synchronization web page with sending and receiving device synchronization functionality, the JavaScript code of the synchronization web page can be configured to determine whether the cookie with the synchronization identifier, set as described and illustrated earlier with reference to step 604 of FIG. 6, was returned with the response that included the synchronization web page. If the cookie with the synchronization identifier was not included with the response that included the synchronization web page, then the synchronization web page can be configured to implement the receiving device synchronization functionality described and illustrated with reference to FIG. 11.

[0064] Accordingly, in step 1100 in this example, the receiving device 104 executing the receiving device synchronization web page 124 sends a request including the synchronization identifier to the web socket server 202 and establishes a session with the web socket server 202. The receiving device 104 can obtain the synchronization identifier from the URL input to the web browser of the receiving device 104 in order to obtain the receiving device synchronization web page 124, for example, although the synchronization identifier can be obtained in other ways.

[0065] In step 1102, the receiving device 104 executing the receiving device synchronization web page 124 determines whether a connection has been established with the sending device 102. A connection is established with the sending device 104 as described and illustrated in detail earlier with reference to FIG. 9. Upon the web socket server 202 establishing a connection with the sending device 102, and subsequent to the receiving device 104 establishing the connection with the web socket server 202, the web socket server 202 sends a confirmation message to the receiving device 104 using the connection established in step 1100. Accordingly, in step 1102, the receiving device 104 determines whether the message confirming establishment of a connection between the web socket server 202 and the sending device 102 has

been received. If the receiving device 104 determines that the connection with the sending device 102 has not been established, then the No branch is taken back to step 1102 and the receiving device 104 effectively waits until the connection with the sending device 102 is established.

[0066] However, if the receiving device 104 determines in step 1102 that the connection has been established with the sending device 102, then the Yes branch is taken to step 1104. In step 1104, the receiving device 104 executing the receiving device synchronization web page 124 receives the one or more cookies with session information and the redirect URL from the web socket server 202 over the connection established in step 1100. The one or more cookies with the session information and the redirect URL were sent by the sending device 102 to the web socket server 202 in this example as described and illustrated in more detail earlier with reference to step 904 of FIG. 9.

[0067] In step 1106, the receiving device 104 executing the receiving device synchronization web page 124, determines whether an end of communication message has been received from the web socket server 202 indicating the end of information to be received from the web socket server 202, as sent to the web socket server 202 by the sending device 102. If the receiving device 104 determines that an end of communication message has not been received then the No branch is taken back to step 1106 and the receiving device 104 effectively waits for the end of communication message.

[0068] However, if the receiving device 104 determines in step 1106 that an end of communication message has been received, then the Yes branch is taken to step 1108. In step 1108, the receiving device 104 executing the receiving device synchronization web page 124 manages the cookie data, including associated session information, received in step 1104. In this example, the web browser of the receiving device 104 sets one or more cookies including the session information according to the current domain and path.

[0069] In step 1110, the receiving device 104 executing the receiving device synchronization web page 124 generates a redirect based on the redirect URL received in step 1104. The generated redirect is processed by the web browser of the receiving device 104 which obtains and executes the receiving device session content web page 120 that is located at the redirect URL.

[0070] Referring more specifically to FIG. 12, an exemplary receiving device session content web page 120 is illustrated. The receiving device session content web page 120 is hosted by the web content server 304 and located at "https://www.acme.com/cart" in this example, which is the redirect URL sent by the sending device 102, as described and illustrated earlier with reference to step 904 of FIG. 9, and received by the receiving device 104, as described and illustrated earlier with reference to step 1104 of FIG. 11.

[0071] Subsequent to obtaining the receiving device session content web page 120, as described and illustrated earlier with reference to step 1108 of FIG. 11, the receiving device 104 can execute the receiving device session content web page 120 by rendering the receiving content web page 120 including the session information included in the one or more cookies received by the receiving device 104 as described and illustrated earlier with reference to step 1104 of FIG. 11. Accordingly, in this example, the web browser of the receiving device 104 can render a desktop version of the sending device session content web page 118 that includes the web session information, thereby advantageously maintaining continuity of the user's web browsing experience across devices.

[0072] Referring more specifically to FIG. 13, an exemplary method for facilitating synchronization of a web session between the sending device 102 and the receiving device 104 by the web socket server 202 of the web session synchronization apparatus 500 is illustrated. In step 1300, the web socket server 202 of the web session synchronization apparatus 500 receives a request to establish a connection from the sending device 102 and establishes a connection with the sending device 102 in response. The request is sent by the sending device 102 in this example as described and illustrated earlier with reference to step 900 of FIG. 9. Accordingly, the request includes the synchronization identifier which is optionally stored in a table in the memory 208 of the web socket server 202 allowing the web socket server 202 to identify active connections and associated synchronization identifiers.

[0073] In step 1302, the web socket server 202 receives a request to establish a connection from the receiving device 104 and establishes a connection with the receiving device 104 in response. The request is sent by the receiving device 104 in this example as described and illustrated earlier with reference to step 1100 of FIG. 11. Accordingly, the request includes the synchronization identifier.

[0074] In step 1304, the web socket server 202 of the web session synchronization apparatus 500 sends confirmation messages to the sending device 102 and the receiving device

104. Accordingly, the web socket server 202 can compare the synchronization identifier received from the receiving device 104 to synchronization identifiers of active connections and send a confirmation message confirming the establishment of a connection with the receiving device 104 to the sending device 102 associated with the active connection having a matching associated synchronization identifier. Subsequent to sending the confirmation message to the sending device 102, the web socket server 202 can send a confirmation message confirming the establishment of a connection with the sending device 102 to the receiving device 104. The confirmation messages can be received by the sending device 102 as described and illustrated earlier with reference to step 902 of FIG. 9 and by the receiving device 104 as described and illustrated earlier with reference to step 1102 of FIG. 11.

[0075] In step 1306, the web socket server 202 of the web session synchronization apparatus 500 receives one or more cookies with session information and a redirect URL from the sending device 102 over the established connection with the sending device 102 and forwards the one or more cookies and the redirect URL to the receiving device 104 over the established connection with the receiving device 104. The one or more cookies with session information and the redirect URL are sent by the sending device 102 in this example as described and illustrated earlier with reference to step 904 of FIG. 9. Additionally, the one or more cookies and the redirect URL are received by the receiving device 104 in this example as described and illustrated earlier with reference to step 1104 of FIG. 11.

[0076] In step 1308, the web socket server 202 of the web session synchronization apparatus 500 sends an end of communication message to each of the sending device 102 and the receiving device 104. The end of communication messages are received by the sending device 102 in this example as described and illustrated earlier with reference to step 906 of FIG. 9 by the receiving device 104 in this example as described and illustrated earlier with reference to step 1106 of FIG. 11. Optionally, the web socket server 202 closes the connections with each of the sending device 102 and the receiving device 104 following the sending of the end of communication messages.

[0077] With this technology, web sessions can be synchronized between devices such that web session information is maintained, thereby improving continuity of a user's web browsing experience. Accordingly, users can switch between devices and maintain a web session and web browsing experience without having to repeat interactions with a web site associated with the web session. Additionally, web socket connections to a web socket server are advantageously utilized to transfer the web session information between a sending and a

receiving device. Accordingly, security is increased as the session information is not stored server-side, such as on a web content server.

[0078] Having thus described the basic concept of the invention, it will be rather apparent to those skilled in the art that the foregoing detailed disclosure is intended to be presented by way of example only, and is not limiting. Various alterations, improvements, and modifications will occur and are intended to those skilled in the art, though not expressly stated herein. These alterations, improvements, and modifications are intended to be suggested hereby, and are within the spirit and scope of the invention. Additionally, the recited order of processing elements or sequences, or the use of numbers, letters, or other designations therefore, is not intended to limit the claimed processes to any order except as may be specified in the claims. Accordingly, the invention is limited only by the following claims and equivalents thereto.

CLAIMS

What is claimed is:

1. A method for synchronizing web sessions, the method comprising:

establishing, with a web session synchronization apparatus, in response to a request a first web socket connection with a sending device;

establishing, with the web session synchronization apparatus, in response to another request from a receiving device a second web socket connection;

notifying, with the web session synchronization apparatus, the sending device when the second web socket connection is established;

receiving, with the web session synchronization apparatus, over the first web socket connection one or more cookies comprising session information and a redirect uniform resource locator (URL) from the sending device in response to the notifying; and

forwarding, with the web session synchronization apparatus, the one or more cookies and the redirect URL to the receiving device over the second web socket connection, wherein the redirect URL is associated with a web page that, when executed by the receiving device, is configured to comprise the session information.

2. The method of claim 1, further comprising providing, with the web session synchronization apparatus, a sending device session content web page to the sending device, the session content web page configured to, when executed by the sending device, generate and send a request for a sending device synchronization web page, the request for the sending device synchronization web page including the synchronization identifier.

3. The method of claim 1, further comprising providing, with the web session synchronization apparatus, a sending device synchronization web page to the sending device, the sending device synchronization web page configured to, when executed by the sending device, send the request to establish the first web socket connection and the one or more cookies and redirect URL over the first web socket connection.

4. The method of claim 3, further comprising providing, with the web session synchronization apparatus, a receiving device synchronization web page to the receiving device, the receiving device synchronization web page configured to, when executed by the receiving device, generate the request to establish the second web socket connection, receive the

one or more cookies and the redirect URL, and redirect a web browser of the receiving device based on the redirect URL.

5. The method of claim 4, wherein the sending device synchronization web page and the receiving device web page are the same synchronization web page and the synchronization web page is configured to, when executed by the sending device or the receiving device, determine a behavior based on whether another cookie including the synchronization identifier is included in a request for the synchronization web page or a response including the synchronization web page.

6. The method of claim 1, wherein the receiving device session content web page is a same version of the sending device session content web page or a different version of the sending device session content web page adapted for a type of the receiving device and served from a different location than sending device session content web page.

7. A web session synchronization apparatus, comprising:

a processor coupled to a memory and configured to execute programmed instructions stored in the memory, comprising:

receiving a request to establish a first web socket connection with a sending device, the request to establish the first web socket connection, and establishing the first web socket connection in response to the request to establish the first web socket connection;

receiving a request to establish a second web socket connection from a receiving device, the request to establish the second web socket connection including the synchronization identifier, establishing the second web socket connection in response to the request to establish the second web socket connection, and notifying the sending device when the second web socket connection is established;

receiving one or more cookies including session information and a redirect uniform resource locator (URL) from the sending device in response to the notification and over the first web socket connection; and

forwarding the one or more cookies and the redirect URL to the receiving device over the second web socket connection, wherein the redirect URL is associated with a web page that, when executed by the receiving device, is configured to comprise the session information.

8. The apparatus of claim 7, wherein the processor is further configured to execute programmed instructions stored in the memory further comprising providing a sending device session content web page to the sending device, the session content web page configured to, when executed by the sending device, generate and send a request for a sending device synchronization web page, the request for the sending device synchronization web page including the synchronization identifier.

9. The apparatus of claim 7, wherein the processor is further configured to execute programmed instructions stored in the memory further comprising providing a sending device synchronization web page to the sending device, the sending device synchronization web page configured to, when executed by the sending device, send the request to establish the first web socket connection and the one or more cookies and redirect URL over the first web socket connection.

10. The apparatus of claim 9, wherein the processor is further configured to execute programmed instructions stored in the memory further comprising providing a receiving device synchronization web page to the receiving device, the receiving device synchronization web page configured to, when executed by the receiving device, generate the request to establish the second web socket connection, receive the one or more cookies and the redirect URL, and redirect a web browser of the receiving device based on the redirect URL.

11. The apparatus of claim 10, wherein the sending device synchronization web page and the receiving device web page are the same synchronization web page and the synchronization web page is configured to, when executed by the sending device or the receiving device, determine a behavior based on whether another cookie including the synchronization identifier is included in a request for the synchronization web page or a response including the synchronization web page.

12. The apparatus of claim 7, wherein the receiving device session content web page is a same version of the sending device session content web page or a different version of the sending device session content web page adapted for a type of the receiving device and served from a different location than sending device session content web page.

13. A non-transitory computer readable medium having stored thereon instructions for synchronizing web sessions comprising machine executable code which when executed by a processor, causes the processor to perform steps comprising:

receiving a request to establish a first web socket connection with a sending device, the request to establish the first web socket connection, and establishing the first web socket connection in response to the request to establish the first web socket connection;

receiving a request to establish a second web socket connection from a receiving device, the request to establish the second web socket connection including the synchronization identifier, establishing the second web socket connection in response to the request to establish the second web socket connection, and notifying the sending device when the second web socket connection is established;

receiving one or more cookies including session information and a redirect uniform resource locator (URL) from the sending device in response to the notification and over the first web socket connection; and

forwarding the one or more cookies and the redirect URL to the receiving device over the second web socket connection, wherein the redirect URL is associated with a web page that, when executed by the receiving device, is configured to comprise the session information.

14. The medium of claim 13, wherein the machine executable code when executed by the processor further causes the processor to perform steps further comprising providing a sending device session content web page to the sending device, the session content web page configured to, when executed by the sending device, generate and send a request for a sending device synchronization web page, the request for the sending device synchronization web page including the synchronization identifier.

15. The medium of claim 13, wherein the machine executable code when executed by the processor further causes the processor to perform steps further comprising providing a sending device synchronization web page to the sending device, the sending device synchronization web page configured to, when executed by the sending device, send the request to establish the first web socket connection and the one or more cookies and redirect URL over the first web socket connection.

16. The medium of claim 15, wherein the machine executable code when executed by the processor further causes the processor to perform steps further comprising providing a receiving device synchronization web page to the receiving device, the receiving device synchronization web page configured to, when executed by the receiving device, generate the request to establish the second web socket connection, receive the one or more cookies and the redirect URL, and redirect a web browser of the receiving device based on the redirect URL.

17. The medium of claim 16, wherein the sending device synchronization web page and the receiving device web page are the same synchronization web page and the synchronization web page is configured to, when executed by the sending device or the receiving device, determine a behavior based on whether a cookie including the synchronization identifier is included in a request for the synchronization web page or a response including the synchronization web page.

18. The medium of claim 13, wherein the receiving device session content web page is a same version of the sending device session content web page or a different version of the sending device session content web page adapted for a type of the receiving device and served from a different location than sending device session content web page.

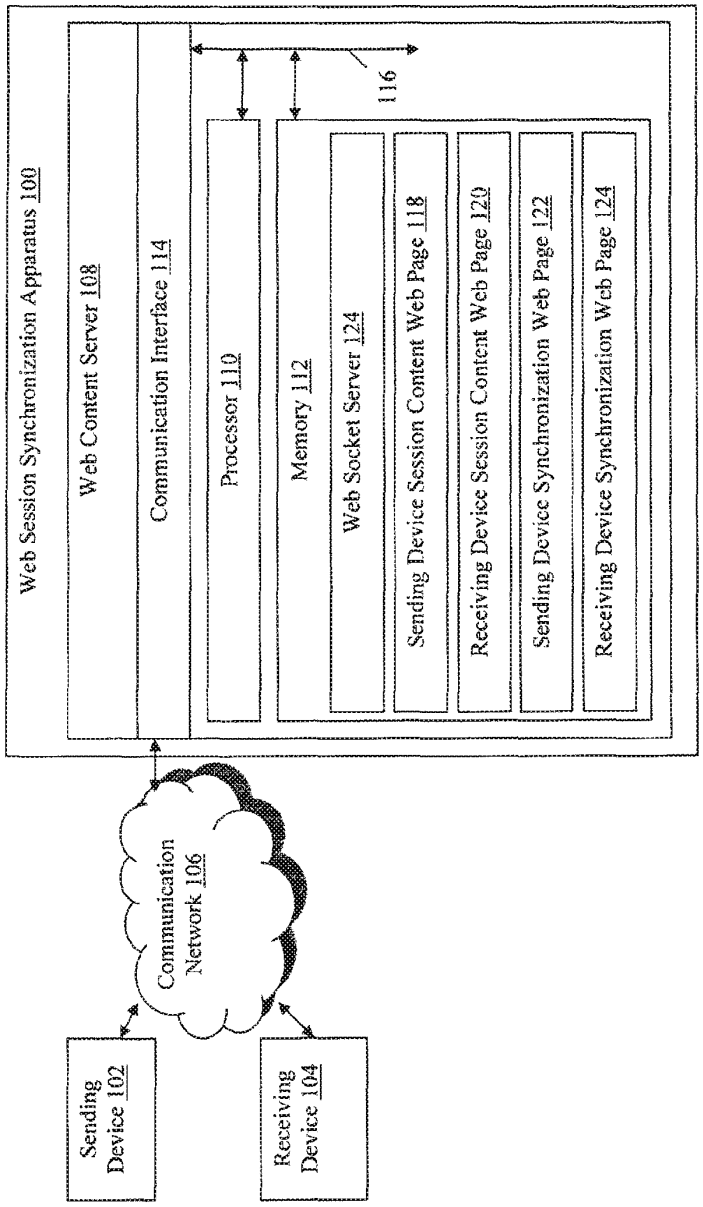

Web Session Synchronization Apparatus 100
Web Content Server 108
Communication Interface 114
Processor 110
Memory 112
116
Web Socket Server 124
Sending Device Session Content Web Page 118
Receiving Device Session Content Web Page 120
Sending Device Synchronization Web Page 122
Receiving Device Synchronization Web Page 124
Communication Network 106
Sending Device 102
Receiving Device 104

FIG. 1

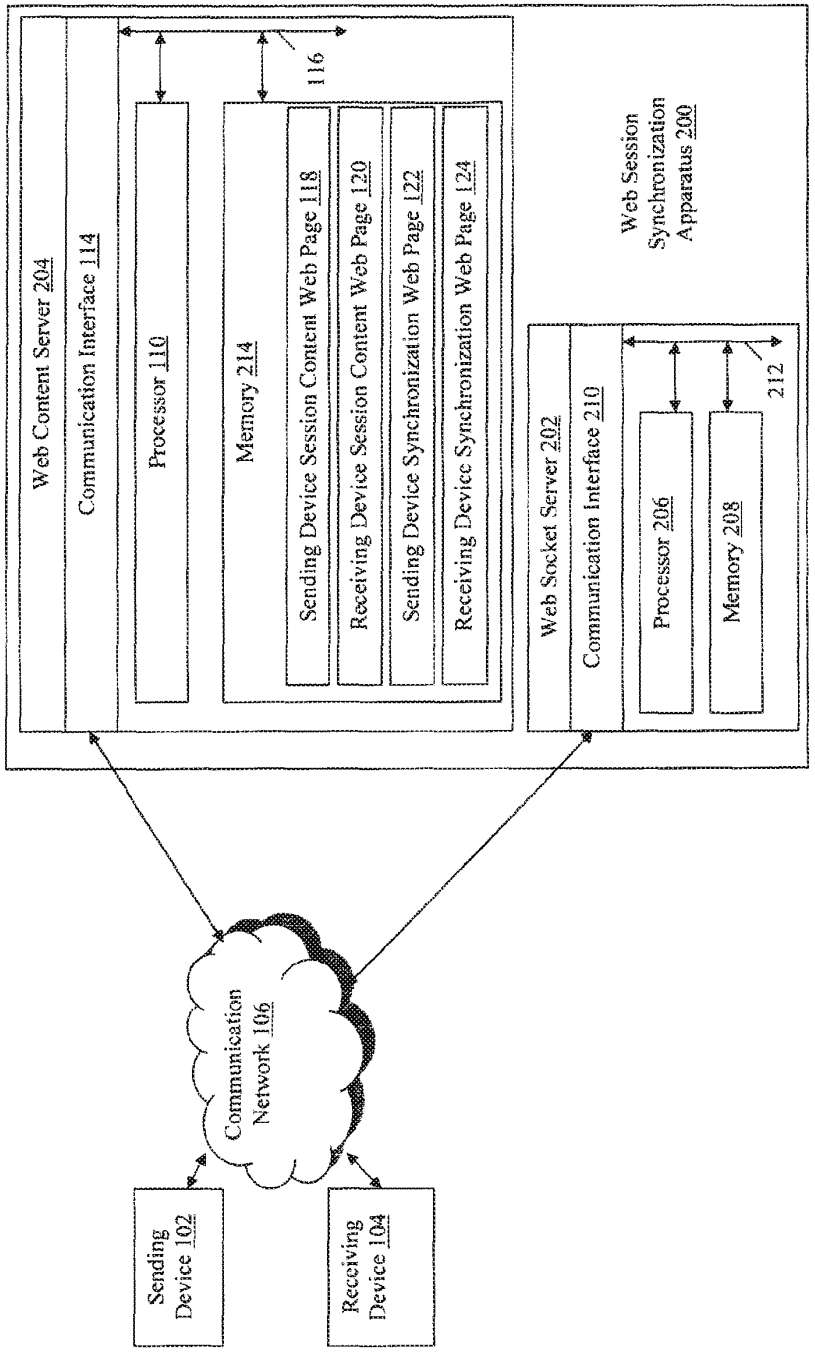

Web Content Server 204
Communication Interface 114
Processor 110
116
Memory 214
Sending Device Session Content Web Page 118
Receiving Device Session Content Web Page 120
Sending Device Synchronization Web Page 122
Receiving Device Synchronization Web Page 124
Web Session Synchronization Apparatus 200
Web Socket Server 202
Communication Interface 210
Processor 206
Memory 208
212
Communication Network 106
Sending Device 102
Receiving Device 104

FIG. 2

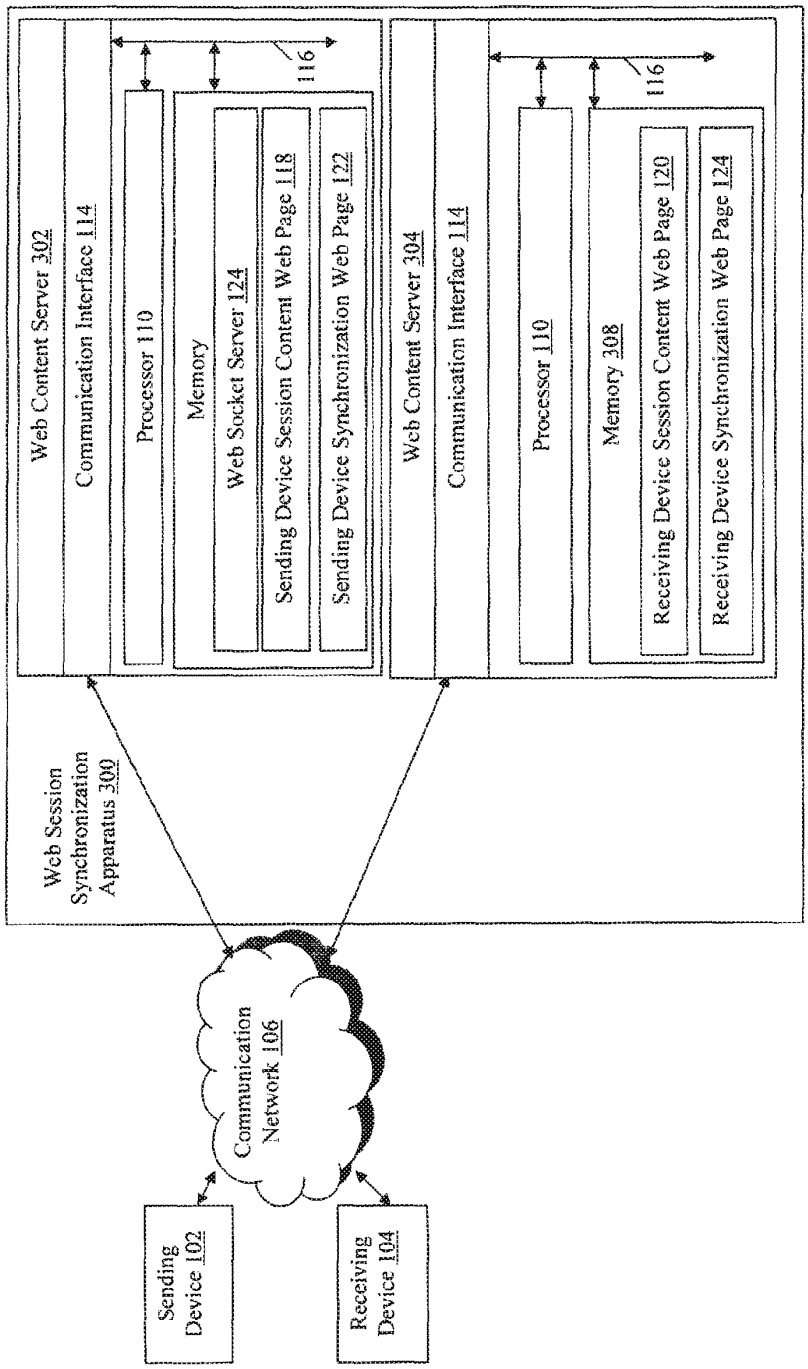

Web Session Synchronization Apparatus 300
Web Content Server 302
Communication Interface 114
Processor 110
Memory
Web Socket Server 124
Sending Device Session Content Web Page 118
Sending Device Synchronization Web Page 122
116
Web Content Server 304
Communication Interface 114
Processor 110
Memory 308
Receiving Device Session Content Web Page 120
Receiving Device Synchronization Web Page 124
116
Communication Network 106
Sending Device 102
Receiving Device 104

FIG. 3

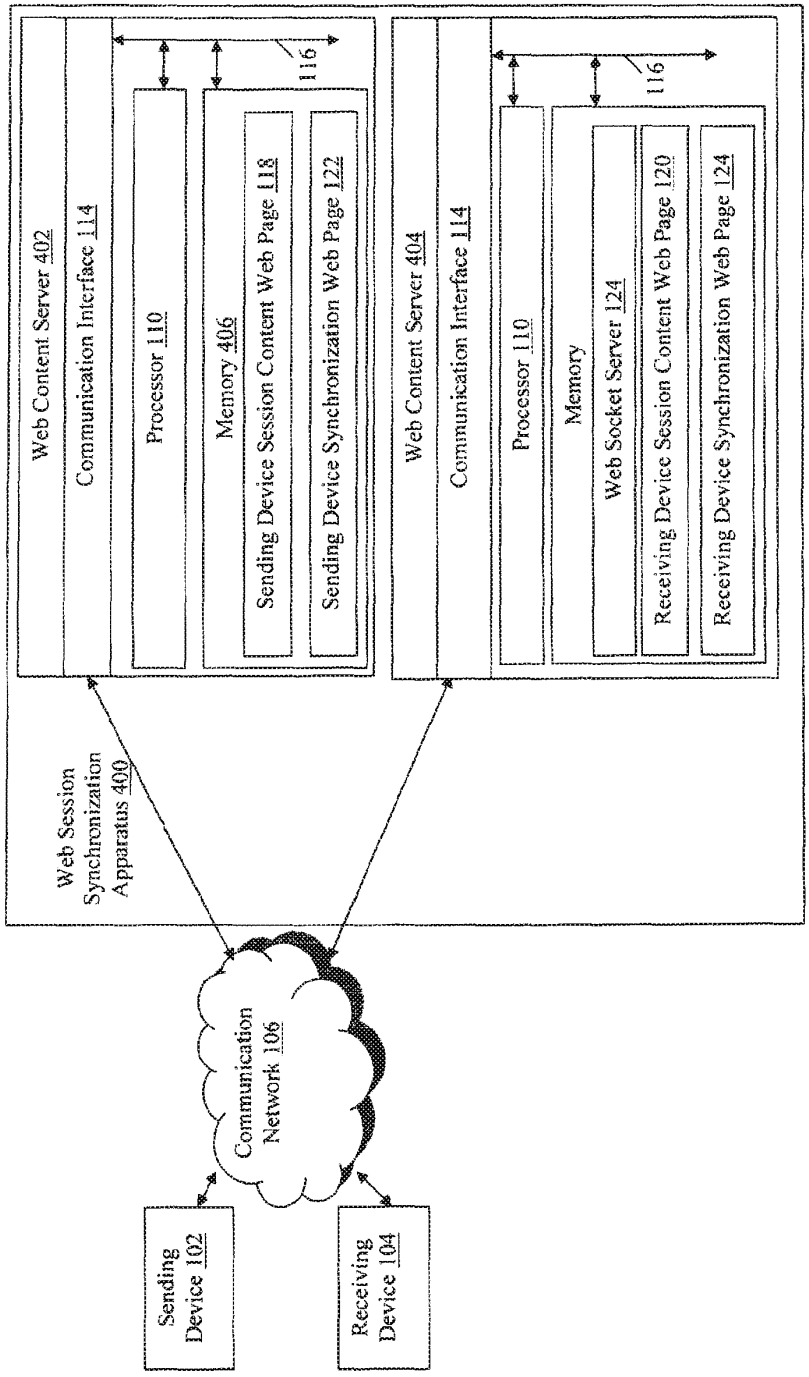

Web Session Synchronization Apparatus 400
Web Content Server 402
Communication Interface 114
Processor 110
Memory 406
Sending Device Session Content Web Page 118
Sending Device Synchronization Web Page 122
116
Web Content Server 404
Communication Interface 114
Processor 110
Memory
Web Socket Server 124
Receiving Device Session Content Web Page 120
Receiving Device Synchronization Web Page 124
116
Communication Network 106
Sending Device 102
Receiving Device 104

FIG. 4

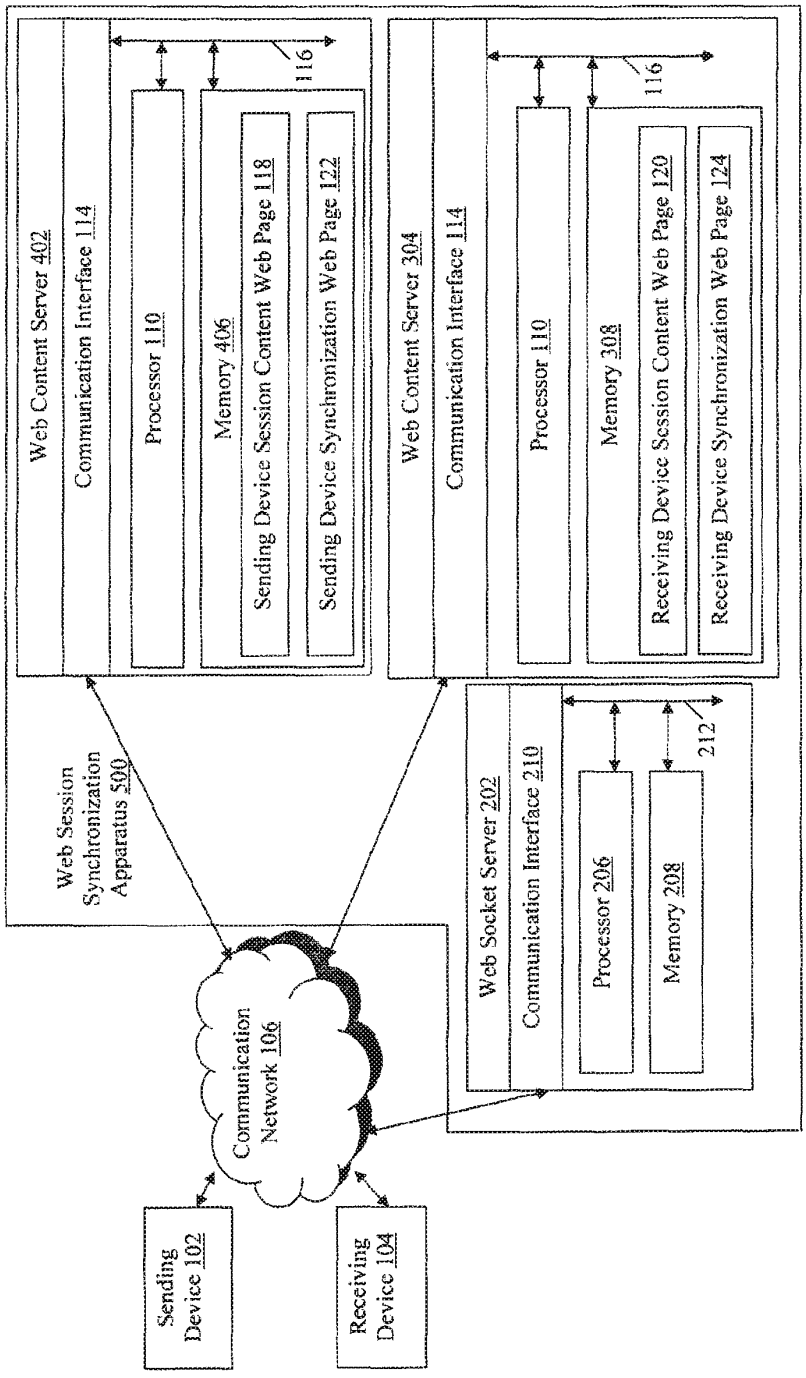

Web Content Server 402
Communication Interface 114
Processor 110
Memory 406
Sending Device Session Content Web Page 118
Sending Device Synchronization Web Page 122
116
Web Content Server 304
Communication Interface 114
Processor 110
Memory 308
Receiving Device Session Content Web Page 120
Receiving Device Synchronization Web Page 124
116
Web Session Synchronization Apparatus 500
Web Socket Server 202
Communication Interface 210
Processor 206
Memory 208
212
Communication Network 106
Sending Device 102
Receiving Device 104

FIG. 5

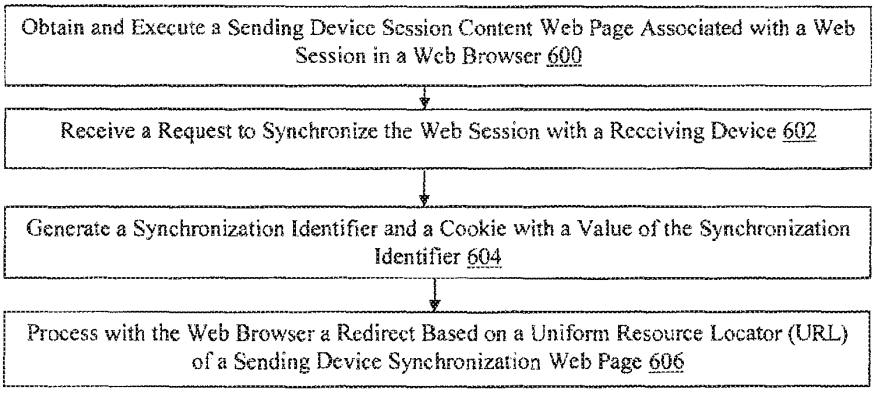

Obtain and Execute a Sending Device Session Content Web Page Associated with a Web Session in a Web Browser 600
Receive a Request to Synchronize the Web Session with a Receiving Device 602
Generate a Synchronization Identifier and a Cookie with a Value of the Synchronization Identifier 604
Process with the Web Browser a Redirect Based on a Uniform Resource Locator (URL) of a Sending Device Synchronization Web Page 606

FIG. 6

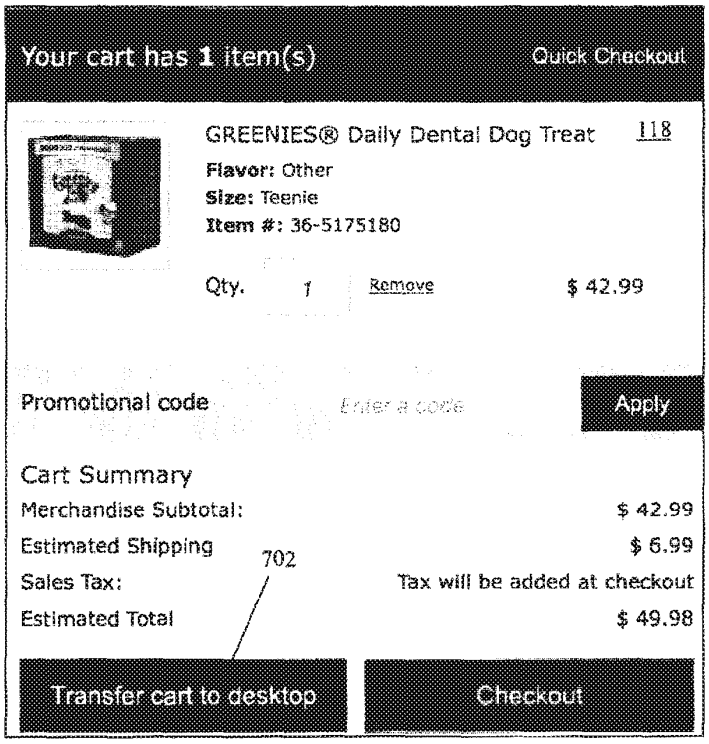

Your cart has 1 item(s)
Quick Checkout
GREENIES® Daily Dental Dog Treat
118
Flavor: Other
Size: Teenie
Item #: 36-5175180
Qty. 1
Remove
$ 42.99
Promotional code
Enter a code
Apply
Cart Summary
Merchandise Subtotal:
$ 42.99
Estimated Shipping
$ 6.99
Sales Tax:
Tax will be added at checkout
Estimated Total
$ 49.98
702
Transfer cart to desktop
Checkout

FIG. 7

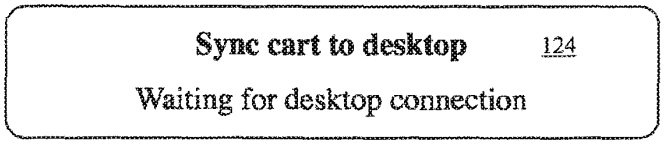


FIG. 8

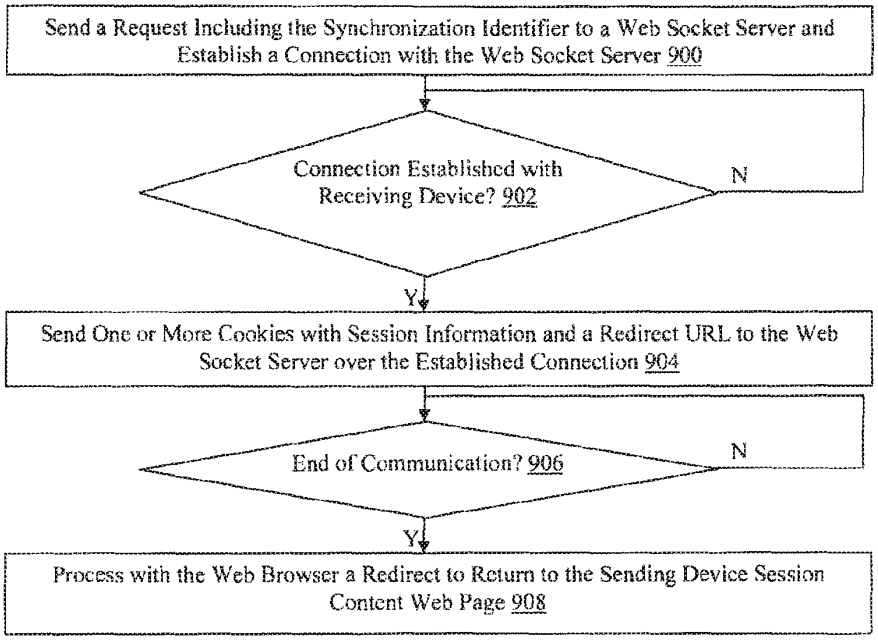


FIG. 9

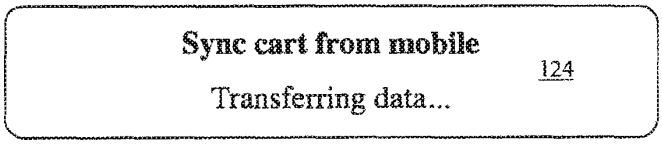


FIG. 10

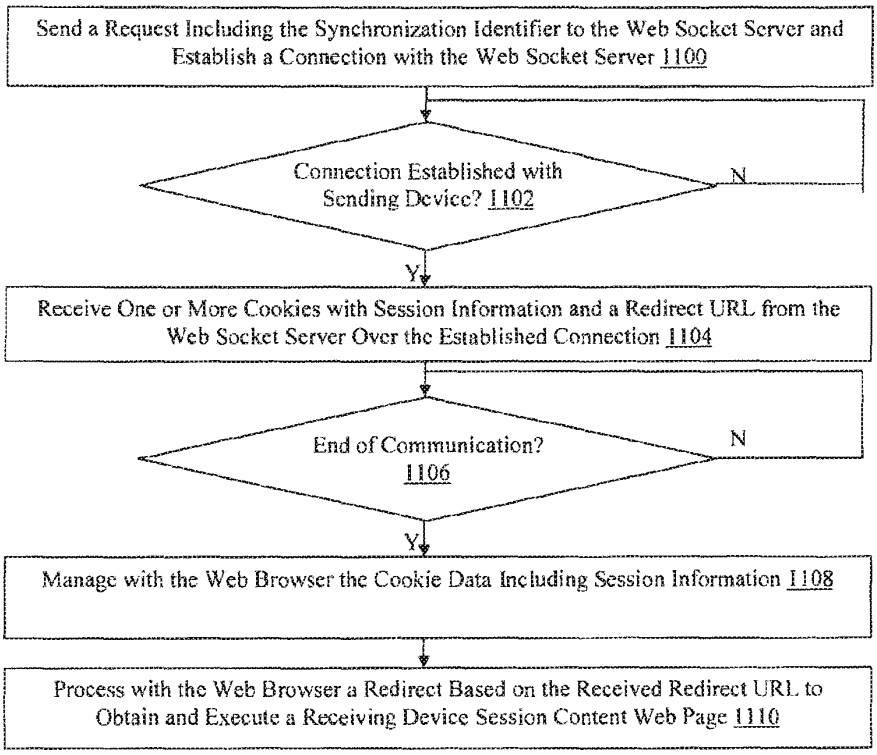

Send a Request Including the Synchronization Identifier to the Web Socket Server and Establish a Connection with the Web Socket Server 1100
Connection Established with Sending Device? 1102
N
Y
Receive One or More Cookies with Session Information and a Redirect URL from the Web Socket Server Over the Established Connection 1104
End of Communication? 1106
N
Y
Manage with the Web Browser the Cookie Data Including Session Information 1108
Process with the Web Browser a Redirect Based on the Received Redirect URL to Obtain and Execute a Receiving Device Session Content Web Page 1110

FIG. 11

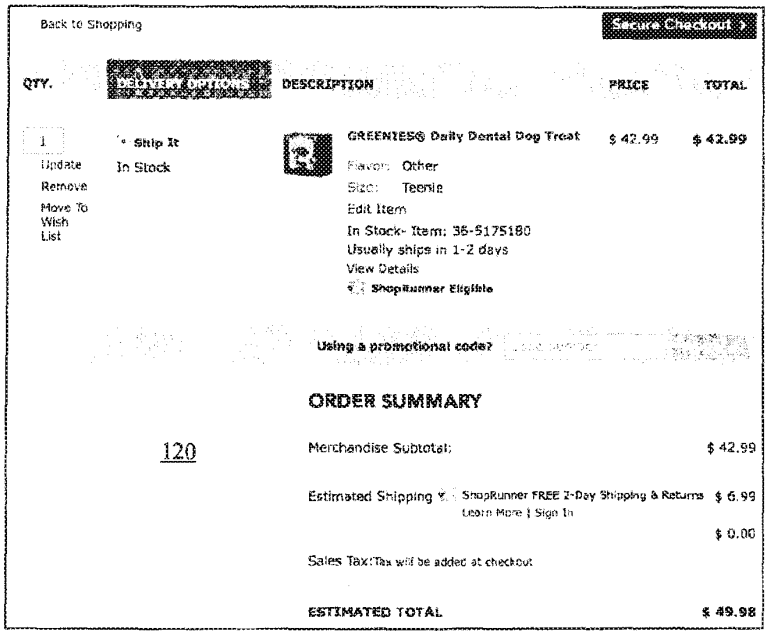

Back to Shopping
Secure Checkout
QTY.
DESCRIPTION
PRICE
TOTAL
1
Update
Remove
Move To Wish List
Ship It
In Stock
GREENIES® Daily Dental Dog Treat
$ 42.99
$ 42.99
Flavor: Other
Size: Teenie
Edit Item
In Stock- Item: 36-5175180
Usually ships in 1-2 days
View Details
ShopRunner Eligible
Using a promotional code?
ORDER SUMMARY
120
Merchandise Subtotal:
$ 42.99
Estimated Shipping
ShopRunner FREE 2-Day Shipping & Returns
Learn More | Sign In
$ 6.99
$ 0.00
Sales Tax: Tax will be added at checkout
ESTIMATED TOTAL
$ 49.98

FIG. 12

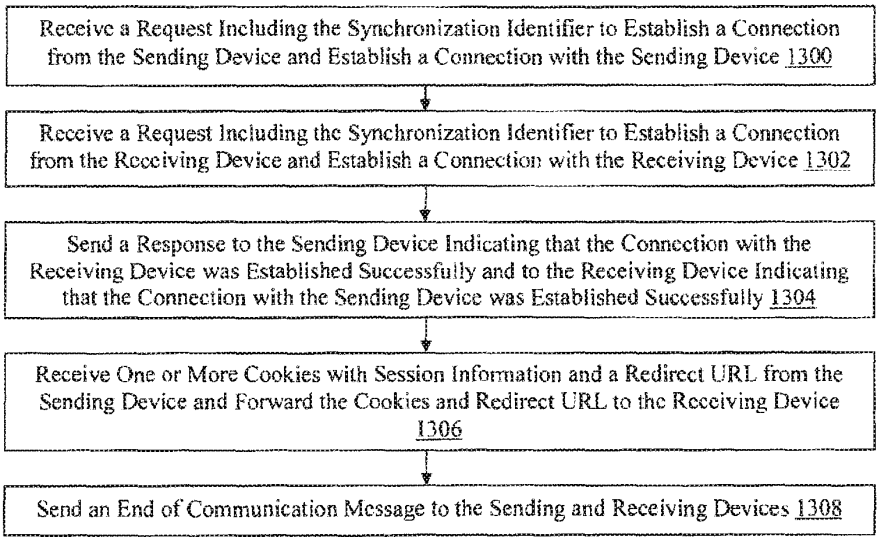

Receive a Request Including the Synchronization Identifier to Establish a Connection from the Sending Device and Establish a Connection with the Sending Device 1300
Receive a Request Including the Synchronization Identifier to Establish a Connection from the Receiving Device and Establish a Connection with the Receiving Device 1302
Send a Response to the Sending Device Indicating that the Connection with the Receiving Device was Established Successfully and to the Receiving Device Indicating that the Connection with the Sending Device was Established Successfully 1304
Receive One or More Cookies with Session Information and a Redirect URL from the Sending Device and Forward the Cookies and Redirect URL to the Receiving Device 1306
Send an End of Communication Message to the Sending and Receiving Devices 1308

FIG. 13

Web Session Synchronization Apparatus 500
Web Content Server 402
Communication Interface 114
Processor 110
Memory 406
Sending Device Session Content Web Page 118
Sending Device Synchronization Web Page 122
116
Web Content Server 304
Communication Interface 114
Processor 110
Memory 308
Receiving Device Session Content Web Page 120
Receiving Device Synchronization Web Page 124
116
Web Socket Server 202
Communication Interface 210
Processor 206
Memory 208
212
Communication Network 106
Sending Device 102
Receiving Device 104

Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2963393 A1 2016/04/14

(21) 2 963 393

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(86) Date de dépôt PCT/PCT Filing Date: 2015/06/22
(87) Date publication PCT/PCT Publication Date: 2016/04/14
(85) Entrée phase nationale/National Entry: 2017/03/31
(86) N° demande PCT/PCT Application No.: US 2015/036956
(87) N° publication PCT/PCT Publication No.: 2016/057092
(30) Priorité/Priority: 2014/10/08 (US14/509,235)

(51) Cl.Int./Int.Cl. *G06F 17/30* (2006.01), *G06Q 30/06* (2012.01)
(71) Demandeur/Applicant:
USABLENET INC., US

(72) Inventeur/Inventor:
SCODA, ENRICO, IT

(74) Agent: PARLEE MCLAWS LLP

(54) Titre : PROCEDES POUR FACILITER DES REFERENCES DANS UN CONTEXTE DE DIALOGUE EN LIGNE ET DISPOSITIFS ASSOCIES

(54) Title: METHODS FOR FACILITATING REFERENCES IN A CHAT CONTEXT AND DEVICES THEREOF

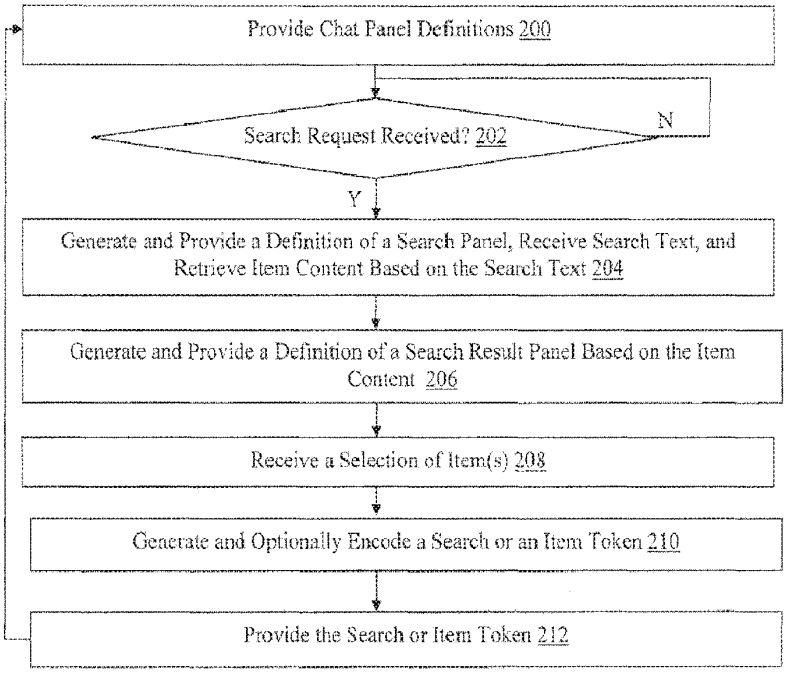


FIG. 2

(57) Abrégé/Abstract:
A method, non-transitory computer readable medium, and chat management server apparatus that receives a search request via a search panel provided in response to a user interaction with a chat panel. A token including a special character is generated based

(21) 2 963 393

(13) A1

(57) Abrégé(suite)/Abstract(continued):
on search text in the search request or a unique identifier for one of a plurality of items identified based on a search performed using the search text. The token is provided to a source of the search request for inclusion in the chat panel as a hyperlink. A preview panel request including the token is received in response to a user interaction with the hyperlink. Content for the items or for the one item is retrieved based on the special character included in the token. The content is provided to a source of the preview panel request.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
14 April 2016 (14.04.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/057092 A1

(51) International Patent Classification:
G06F 17/30 (2006.01)

(21) International Application Number:
PCT/US2015/036956

(22) International Filing Date:
22 June 2015 (22.06.2015)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
14/509,235 8 October 2014 (08.10.2014) US

(71) Applicant: **USABLENET INC.** [US/US]; 142 W. 57th Street, 7th Floor, New York, NY 10019 (US).

(72) Inventor: **SCODA, Enrico**; Via Cividina 416/3, Martignacco UD 33035 (IT).

(74) Agents: **GALLO, Nicholas, J.** et al.; LeClairRyan, A Professional Corporation, 70 Linden Oaks, Suite 210, Rochester, NY 14625 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

---- *with international search report (Art. 21(3))*

(54) Title: METHODS FOR FACILITATING REFERENCES IN A CHAT CONTEXT AND DEVICES THEREOF

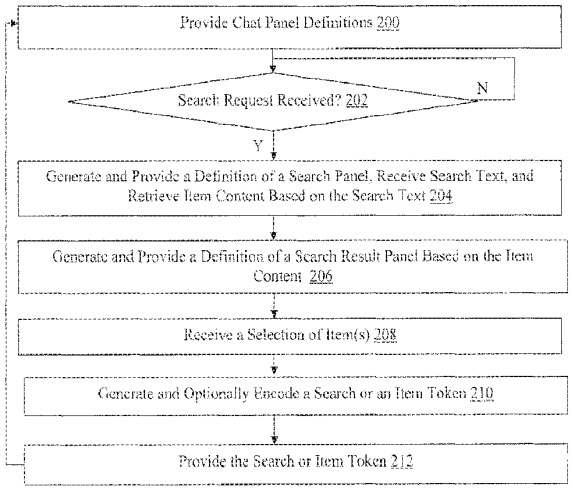


FIG. 2

(57) Abstract: A method, non-transitory computer readable medium, and chat management server apparatus that receives a search request via a search panel provided in response to a user interaction with a chat panel. A token including a special character is generated based on search text in the search request or a unique identifier for one of a plurality of items identified based on a search performed using the search text. The token is provided to a source of the search request for inclusion in the chat panel as a hyperlink. A preview panel request including the token is received in response to a user interaction with the hyperlink. Content for the items or for the one item is retrieved based on the special character included in the token. The content is provided to a source of the preview panel request.

WO 2016/057092 A1

METHODS FOR FACILITATING REFERENCES IN A CHAT CONTEXT AND DEVICES THEREOF

FIELD

[0001] This technology generally relates to web-based chat contexts and, more particularly, to methods, non-transitory computer readable media, and apparatuses that facilitate the inclusion of references in chat panels.

BACKGROUND

[0002] Increasingly, web developers are providing chat functionality in websites via chat contexts that include chat panels that are displayed to a user as embedded within a web page or via a pop-up window, for example. A chat panel can allow interaction and communication between the user and a representative of the website host. Accordingly, such functionality is particularly useful for websites providing user support, although many other types of websites also implement chat contexts. In chat contexts, the speed of an exchange is often critical to an effective experience for users and, accordingly, it is preferable that messages are simple and short.

[0003] As one example, in a commercial website context, sales representatives may use chat panels to communicate with potential customers to answer questions regarding products or services in order to facilitate and increase sales. In this example, a sales representative may want to refer a prospective customer to content hosted on other portions of the website, such as product catalog content relating to products that might satisfy desired criteria communicated by the prospective customer.

[0004] In order to refer the prospective customer to the content, the sales representative may copy and paste Uniform Resource Locators (URLs) as hyperlinks. However, URLs are often very long and inconvenient for use in a chat context. While available services can process a URL and replace the URL with a relatively short link, the process is cumbersome and would still result in numerous hyperlinks and a relatively long message when the content is located at a number of URLs (e.g., corresponding to a number of different products). Additionally, upon selecting any of the hyperlinks, the prospective customer may be taken to a different web page in a new tab or window, which is inconvenient and does not allow the customer to preview products prior to navigating to a different web page associated with one of the products. Accordingly, there is currently no way for the representative of a website host to identify items

and provide preview content associated with the items to the prospective customer in an efficient and effective manner in a chat context.

SUMMARY

[0005] A method for facilitating references in a chat context includes receiving by a chat management server apparatus a search request via a search panel provided in response to a user interaction with a chat panel. A token including a special character is generated by the chat management server apparatus based on search text in the search request or a unique identifier for one of a plurality of items identified based on a search performed using the search text. The token is provided by the chat management server apparatus to a source of the search request for inclusion in the chat panel as a hyperlink. A preview panel request including the token is received by the chat management server apparatus in response to a user interaction with the hyperlink. Content for the items or for the one item is retrieved by the chat management server apparatus based on the special character included in the token. The content is provided by the chat management server apparatus to a source of the preview panel request.

[0006] A non-transitory computer readable medium having stored thereon instructions for facilitating references in a chat context comprising executable code which when executed by a processor, causes the processor to perform steps including receiving a search request via a search panel provided in response to a user interaction with a chat panel. A token including a special character is generated based on search text in the search request or a unique identifier for one of a plurality of items identified based on a search performed using the search text. The token is provided to a source of the search request for inclusion in the chat panel as a hyperlink. A preview panel request including the token is received in response to a user interaction with the hyperlink. Content for the items or for the one item is retrieved based on the special character included in the token. The content is provided to a source of the preview panel request.

[0007] A chat management server apparatus including a processor and a memory coupled to the processor which is configured to be capable of executing programmed instructions comprising and stored in the memory to receive a search request via a search panel provided in response to a user interaction with a chat panel. A token including a special character is generated based on search text in the search request or a unique identifier for one of a plurality of items identified based on a search performed using the search text. The token is provided to a source of the search request for inclusion in the chat panel as a hyperlink. A preview panel request including the token is received in response to a user interaction with the hyperlink.

Content for the items or for the one item is retrieved based on the special character included in the token. The content is provided to a source of the preview panel request.

[0008] This technology provides a number of advantages including methods, non-transitory computer readable media, and apparatuses that more effectively facilitate references in a chat context. With this technology, website host representatives can identify content responsive to a search request from a user and communicate a reference to the content using a relatively short token. The token includes a special character and is introduced to a chat panel as a hyperlink. Upon selection of the hyperlink, the content is retrieved based on the token and included special character, and a preview panel is display to the user that includes the content.

[0009] Accordingly, using the tokens, the size of the communicated reference(s) can be reduced, particularly when multiple URLs would otherwise have been required to communicate references to content associated with multiple items responsive to a search request. Additionally, the user does not have to navigate away from the current web page to see the content. Moreover, the tokens can be reused, advantageously allowing the host representative to respond relatively quickly to certain search requests matching previously searched criteria.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] FIG. 1 is a block diagram of a network environment which incorporates an exemplary chat management server apparatus;

[0011] FIG. 2 is a flowchart of an exemplary method of generating an item preview panel based on a token;

[0012] FIG. 3 is an exemplary product web page with an exemplary chat panel link;

[0013] FIG. 4 is an exemplary host chat panel with a search request button;

[0014] FIG. 5 is an exemplary search panel for receiving search text;

[0015] FIG. 6 is an exemplary search result panel displaying content for a plurality of selectable items identified based on search text;

[0016] FIG. 7 is the exemplary chat panel of FIG. 4 with an item set link corresponding to a search token;

[0017] FIG. 8 is a flowchart of an exemplary method of generating a token for inclusion in a customer chat panel as a reference to item content;

[0018] FIG. 9 is an exemplary customer chat panel with the item set link and an exemplary multi-item preview panel with a navigational structure; and

[0019] FIG. 10 is the exemplary chat panel of FIG. 9 with an item link and an exemplary single item preview panel.

DETAILED DESCRIPTION

[0020] An exemplary network environment 10 is illustrated in FIG. 1 as including an exemplary chat management server apparatus 12. In this example, the chat management server apparatus 12 is coupled to a host representative device 14 by a local area network (LAN) 16 and a client device 18 by the LAN 16 and a wide area network (WAN) 20, although other types and numbers of devices, components, and elements in other topologies could be used. This technology provides a number of advantages including methods, non-transitory computer readable media, and apparatuses that more efficiently and effectively facilitate identifying and providing references to content matching user search criteria in a chat context.

[0021] In this example, the chat management server apparatus 12 includes a processor 22, a memory 24, and an interface device 26, which are coupled together by a bus 28 or other communication link, although other numbers and types of components, parts, devices, systems, and elements in other configurations and locations can be used. The processor 22 of the chat management server apparatus 12 may execute one or more stored programmed instructions for one or more aspects of this technology as described and illustrated by way of the embodiments herein, although the processor 22 could execute other numbers and types of programmed instructions.

[0022] The memory 24 of the chat management server apparatus 12 stores these programmed instructions for one or more aspects of this technology, as described and illustrated herein, although some or all of the programmed instructions could be stored and/or executed elsewhere. The memory 24 of the chat management server apparatus 12 may include one or more tangible storage media and/or devices, such as RAM, ROM, flash memory, hard disk drive(s), solid state memory, or any other memory storage types or devices, including combinations thereof, which are known to those of ordinary skill in the art.

[0023] In this example, the memory 24 of the chat management server apparatus 12 includes an item content catalog 30, a search web service 32, a preview web service 34, and an optional encoded token database 36, although the memory 24 can include other types and numbers of systems, devices, and elements in other configurations. Additionally, while the item content catalog 30, search web service 32, preview web service 34, and encoded token database 36 are illustrated in this example as being stored in the memory 24 of the chat management server apparatus 12, one or more of the item content catalog 30, search web service 32, preview web service 34, or encoded token database 36 could be stored elsewhere, including on another network device not shown in the network environment 10.

[0024] The item content catalog 30 in this example includes content for items, which can represent products or services, for example. The content can include a unique identifier for the item and information regarding the item including a description of the item, an item price, item options (e.g., colors), and/or any pictures or graphics associated with the item, for example, although any other type of content can also be stored in the item content catalog.

[0025] The search web service 32 in this example is configured to receive a request including search criteria and to identify matching item(s) in the item content catalog. The criteria can include search text, for example, when associated with a request to generate a search panel or a search token, for example, when associated with a request to generate a preview panel, as described and illustrated in more detail later. In response to the request, the search web service 32 retrieves at least a portion of the content (e.g., as used to present a preview of the item to a user) for each of the identified item(s) from the item content catalog 30 and generates and returns a HyperText Markup Language (HTML) fragment including the content, as described and illustrated in more detail later.

[0026] The preview web service 34 in this example is configured to receive a request to generate a preview panel that includes an item token. In response to the request, the preview web service 34 retrieves at least a portion of the content associated with an item identified based on the item token from the item content catalog 30 and generates and returns an HTML fragment including the content, as described and illustrated in more detail later.

[0027] The optional encoded token database 36 in this example stores encoded token values as associated with actual values. The actual values can be search text in the case of an encoded search token value or a unique item identifier in the case of an encoded item token value. As described and illustrated in more detail later, tokens can advantageously be encoded

with this technology to facilitate the inclusion of references in a chat panel that are even shorter than search text or item identifiers, for example.

[0028] The interface device 26 in the chat management server apparatus 12 is used to operatively couple and communicate between the chat management server apparatus 12 and the client device 18 via LAN 16 and WAN 20 and the host representative device 14 via the LAN 16, although other types and numbers of communication networks or systems with other types and numbers of connections and configurations to other devices and elements can also be used. The LAN 16 and WAN 20 can use TCP/IP over Ethernet and industry-standard protocols, including NFS, CIFS, SOAP, XML, LDAP, and SNMP, for example, although other types and numbers of communication networks can also be used.

[0029] The client device 18 in this example enables a user to request, receive and interact with services and content hosted by the chat management server apparatus 12 via the LAN 16 and WAN 20, although the client device 18 could access content and utilize other types and numbers of content or applications from other sources and could provide a wide variety of other functions for a user. By way of example only, the client device 16 can be a mobile computing device, smart phone, personal digital assistant, or computer, for example.

[0030] The client device 18 includes a processor 38, a memory 40, an interface device 42, an input device 43, and a display device 44 which are coupled together by a bus 45 or other communication link, although the client device 18 can include other numbers and types of components, parts, devices, systems, and elements in other configurations. The processor 38 in the client device 18 executes a program of stored instructions for one or more aspects of the present invention as described and illustrated herein, although the processor 38 could execute other numbers and types of programmed instructions.

[0031] The memory 40 in the client device 18 stores these programmed instructions for one or more aspects of the present invention as described and illustrated herein, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a RAM, ROM, hard disk drive(s), solid state storage device(s), and/or other storage device which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to processor 38 can be used for the memory 40 in the client device 18. In this example, the client device 18 is configured to access web services and web content through a web browser 46 stored in the memory 40. The web browser 46 in this example is configured to process programmed instructions (e.g., JavaScript

code) to render chat panels and preview panels, as well as provide other functionality, as described and illustrated in more detail later.

[0032] The interface device 42 in the client device 18 is used to operatively couple and communicate between the client device 18 and the chat management server apparatus 12 via the LAN 16 and the WAN 20, although other types and numbers of communication networks with other types and numbers of connections and configurations can be used.

[0033] The input device 43 in the client device 18 can be used to input selections, such as a request for a chat or preview panel, as well as messages to be exchanged with the user of the host representative device 14, although the input device 43 could be used to input other types of data and interact with other elements. The input device 43 can include keypads, touch screens, and/or vocal input processing systems, although other types and numbers of input devices can also be used.

[0034] The display device 44 in the client device 18 can be used to show data and information to a user, such as the requested chat or preview panel, although the display device 44 could be used to display other types of data and interact with other elements. The display device 44 can be television screen, a mobile phone screen display, a laptop screen, a tablet screen, or a monitor for example, although other types and numbers of displays could be used depending on the particular type of client device 18.

[0035] The host representative device 14 includes a processor 48, a memory 50, an interface device 52, an input device 53, and a display device 54, which are coupled together by a bus 55 or other communication link, although the host representative device 14 can include other numbers and types of components, parts, devices, systems, and elements in other configurations. The processor 48 in the host representative device 14 executes a program of stored instructions for one or more aspects of the present invention as described and illustrated herein, although the processor 48 could execute other numbers and types of programmed instructions.

[0036] The memory 50 in the host representative device 14 stores these programmed instructions for one or more aspects of the present invention as described and illustrated herein, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a RAM, ROM, hard disk drive(s), solid state storage device(s), and/or other storage device which is read from and/or

written to by a magnetic, optical, or other reading and/or writing system that is coupled to processor 38 can be used for the memory 40 in the host representative device 14.

[0037] In this example, the host representative device 14 is operated by a representative of a host of the website associated with the item content, although the host representative device 14 could be operated by other users, in order to engage in a chat with a user of the client device 18. Accordingly, the memory 50 in this example includes a web browser 56 through which the user of the host representative device can access web services and web content. The web browser 56 in this example is configured to process programmed instructions (e.g., JavaScript code) to render chat panels, search panels, and search result panels, as well as provide other functionality, as described and illustrated in more detail later.

[0038] The interface device 52 in the host representative device 14 is used to operatively couple and communicate between the host representative device 14 and the chat management server apparatus 12 via the LAN 16, although other types and numbers of communication networks with other types and numbers of connections and configurations can be used.

[0039] The input device 53 in the host representative device 18 can be used to input selections, such as a request for a search panel, as well as messages to be exchanged with the user of the client device 18, although the input device 53 could be used to input other types of data and interact with other elements. The input device 53 can include keypads, touch screens, and/or vocal input processing systems, although other types and numbers of input devices can also be used.

[0040] The display device 54 in the host representative device 18 can be used to show data and information to a user, such as the requested search panel, although the display device 54 could be used to display other types of data and interact with other elements. The display device 54 can be television screen, a mobile phone screen display, a laptop screen, a tablet screen, or a monitor for example, although other types and numbers of displays could be used depending on the particular type of host representative device 18.

[0041] Although embodiments of the chat management server apparatus 12, host representative device 14, and client device 18 are described and illustrated herein, each of these devices can be implemented on any suitable computer system or computing device. It is to be understood that the devices and systems of the embodiments described herein are for exemplary purposes, as many variations of the specific hardware and software used to implement the

embodiments are possible, as will be appreciated by those skilled in the relevant art(s). Furthermore, each of the systems of the embodiments may be conveniently implemented using one or more general purpose computer systems, microprocessors, digital signal processors, and micro-controllers, programmed according to the teachings of the embodiments, as described and illustrated herein, and as will be appreciated by those ordinary skill in the art.

[0042] In addition, two or more computing systems or devices can be substituted for any one of the devices in any embodiment. Accordingly, principles and advantages of distributed processing, such as redundancy and replication also can be implemented, as desired, to increase the robustness and performance of the devices and systems of the embodiments. The embodiments may also be implemented on computer system(s) that extend across any suitable network using any suitable interface mechanisms and communications technologies, including by way of example only telecommunications in any suitable form (e.g., voice and modem), wireless communications media, wireless communications networks, cellular communications networks, G3 communications networks, Public Switched Telephone Network (PSTNs), Packet Data Networks (PDNs), the Internet, intranets, and combinations thereof.

[0043] The examples may also be embodied as a non-transitory computer readable medium having instructions stored thereon for one or more aspects of the present technology as described and illustrated by way of the examples herein, as described herein, which when executed by a processor, cause the processor to carry out the steps necessary to implement the methods of the examples, as described and illustrated herein.

[0044] An exemplary method for facilitating references in a chat context will now be described with reference to FIGS. 1-10. Referring specifically to FIG. 2, an exemplary method of generating a token for inclusion in a chat panel as a reference to item content will now be described. In this example, in step 200, the chat management server apparatus 12 provides chat panel definitions in response to a request received from the client device 18. Referring more specifically to FIG. 3, an exemplary product web page 300 with an exemplary chat panel link 302 is illustrated. In this example, a user of the client device 18 use the input/display device 43 to select the chat panel link 302 thereby initiating a chat session and requesting a customer chat panel from the chat management server apparatus 12, although the chat session can be initiated in other manners.

[0045] In response, the chat management server apparatus 12 sends a definition of a customer chat panel to the client device 18 and a host chat panel to the host representative device

14. The definitions can define the chat panels using HTML, for example, which, when interpreted by the web browsers 46 and 56 cause the client device 18 and host representative device 14 to display the customer and host chat panels, respectively, in a pop-up window or an overlay, for example, although any other type of display for the chat panels can also be used. The chat panels facilitate communication of messages between the users of the client device 18 and host representative device 14.

[0046] Referring more specifically to FIG. 4, an exemplary host chat panel 400 is illustrated. In this example, a definition of the host chat panel 400 is provided by the chat management server 12 to the host representative device 14 in response to receiving an initiation of the chat session in step 200 from the client device 18. The definition sent by the chat management server 12 to the client device 18 in step 200 can define a customer chat panel similar to the host chat panel 400 but without the search request button 402, for example, although the chat panels can be similar or different in other ways and other methods of distributing and generating the chat panels can also be used. An exemplary customer chat panel will be described and illustrated in more detail later with reference to FIGS. 9-10.

[0047] In this example, the user of the client device 18 is a prospective customer of the host of the product web page 300 that is looking for assistance completing an outfit prior to purchasing, although the web page 300 can be any other type of web page and the user of the client device 14 can be any other type of user. Accordingly, the customer user of the client device 18 initiates the chat session in order to communicate with a representative of the host of the website, that is currently using the host representative device 14, to obtain the requested assistance in an efficient manner.

[0048] Referring back to FIG. 2, in step 202, the chat management server apparatus 12 determines when a search request has been received from the host representative device 14. The search request can be for content that a user of the host representative device 14 would like to refer the customer to in order to attempt to assist the customer and facilitate a purchase, for example. If the chat management server apparatus 12 determines that a search request has not been received, then the No branch is taken back to step 202 and the chat management server apparatus 12 effectively waits for a search request to be received.

[0049] Referring back to FIG. 4, the search request button 402 of the host chat panel 400 can be used to initiate a search request, although any other type of interface for initiating a search request can also be used. In this example, the customer user of the client device 18 is looking for

white shoes to pair with a specified dress. Accordingly, the host representative using the input/display device 53 of the host representative device 14 begins to type a message in an input field 404, having the text "No problem, here is a set of options for you:" in this example. Next, the host representative using the input/display device 53 of the host representative device 14 selects the search request button 402 in order to initiate a search for items responsive to the customer's request.

[0050] Referring back to step 202 of FIG. 2, upon selection of the search request button 402, or if the chat management server apparatus 12 otherwise determines that a search request has been received, then the Yes branch is taken to step 204. In step 204, the chat management server apparatus 12 generates, and provides to the host representative device 14, a definition of a search panel, receives search text, and retrieves item content based on the search text. The definition can define a search panel using HTML, for example, which, when interpreted by the web browser 56 can cause the host representative device 14 to display a search panel in a pop-up window or an overlay, for example, although the search panel can be displayed in other manners. The search panel is configured to receive search text, and optionally other search criteria, from the host representative using the host representative device 14.

[0051] Referring more specifically to FIG. 5, an exemplary search panel 500 for receiving search text via a text input box 502 is illustrated, although in other examples the search panel 500 can include inputs for other search criteria. In this example, the search text "faith court shoes white" is input by the host representative using the input/display device 53 of the host representative device 14. Upon selection of a search button 504 by the host representative, the search criteria are sent to the chat management server apparatus 12. In this example, the search criteria, including the search text, is sent to the search web service 32 of the chat management server apparatus 12, which is configured to process the criteria as described and illustrated in more detail later. Other methods of receiving search criteria can also be used.

[0052] In response to receiving the search text, the search web service 32 of the chat management server apparatus 12 identifies and retrieves item content responsive to the request in step 204. The item content can be retrieved from the item content catalog 30, which in this example includes content associated with a plurality of items for sale by the website host, although any other type of content associated with any other type of item can also be used. The content can include an item description, an item depiction, an item price, or any other information associated with each of the items. Optionally, the content includes at least

information suitable to provide the customer with a preview of the item to allow the customer to decide whether to learn more about the item in order to make a purchasing decision.

[0053] In step 206, the search web service 32 of the chat management server apparatus 12 generates and provides to the host representative device 14 a definition of a search result panel based on the item content retrieved in step 204. In this example, the item content can include a picture, a short description, and a price of various white faith court shoes identified based on the search text. Accordingly, the definition includes one or more HTML fragments for the identified item(s) that includes the content and is configured to generate a search result panel including the content when rendered by the web browser 56 of the host representative device 14.

[0054] Referring more specifically to FIG. 6, an exemplary search result panel 600 displaying content for a plurality of selectable items identified based on the search text is illustrated. In this example, the search result panel 600 includes item content 602(1) and 601(2) for two items ("white heeled court shoes" and "patent heeled court shoes") satisfying the search criteria received in step 204. Any number of items can be identified and included in the search result panel 600 and the search results can be displayed by the web browser 56 of the host representative device 14 in other manners.

[0055] Referring back to FIG. 2, in step 208, the chat management server 12 receives a selection from the host representative using the host representative device 14 of one or more of the item(s) for which content 602(1) and 601(2) was identified and retrieved in step 204, and provided to the host representative device 14 in step 206. The selected item(s) are those item(s) responsive to the customer's request and for which the host representative would like to specifically refer the customer to in order to assist the customer in making a purchasing decision in this example.

[0056] Referring back to FIG. 6, in this example, the content 602(1) and 602(2) for each item is associated with a select button 604(1) and 604(2), respectively. Additionally, the search result panel 600 includes a select all button 606. Upon selection of one of the buttons 604(1), 604(2), or 606 by the host representative using the input/display device 53 of the host representative device 14, an indication of the selection is sent to the chat management server apparatus 12. Accordingly, the definition of the search result panel 600 sent to the host representative device 14 in step 206 of FIG. 2 is configured to facilitate the selection of items, although other methods of facilitating the selection of item(s) can also be used.

[0057] In step 210, the chat management server apparatus 12 generates, and optionally encodes, a search or an item token. In this example, the tokens are prefixed by a first special character (e.g., "@") or a second special character (e.g., "#") according to whether the token is a search token or an item token, respectively, although the first or second special character can be included in the tokens in any location. The token is a search token if all of the items for which content is displayed in the search panel 600 are selected by the host representative using the host representative device 14 (e.g., by selecting the select all button 606). Additionally, the token is an item token if fewer than all of the items or which content is displayed in the search panel 600 are selected (e.g., using one or more of the select buttons 604(1) or 604(2)).

[0058] In this example, if the host representative using the host representative device 14 selects the select all button 606, an exemplary token including the first special character and the search text could be "@faith_court_shoes_white". By including the search text, the host representative can advantageously reuse this token in subsequent chat sessions as it will be relatively easy to remember. For example, the host representative can reuse the token with other prospective customers the host representative would like to refer to the same content, as described and illustrated in more detail later.

[0059] In another example, as described and illustrated in more detail later with reference to FIG. 10, if the host representative using the host representative device 14 selected only one of the items for which content was displayed on the search panel 600, an exemplary token including the second special character and a unique item identifier could be '#3611369". Accordingly, in this example, the "3611369" portion of the token corresponds to a unique identifier for the one selected item, as stored as associated with the content for the item in the content catalog 30.

[0060] However, in yet another example, the chat management server apparatus 12 can encode the token in order to reduce the size instead of merely using the search text or the unique item identifier for the portion of the token not including the special character. Accordingly, the chat management server apparatus 12 can decide to encode the token based on whether the number of characters in the search text or unique item identifier exceeds a threshold, for example, although the decision of whether to encode a token can be based on any other criteria. Additionally, the chat management server apparatus 12 can be configured to encode all or none of the tokens as a default setting.

[0061] If the chat management server apparatus 12 determines in step 210 that the search token in this example should be encoded, an exemplary encoded search token could be

"@42ad42", although any other encoding can be used. The chat management server apparatus 12 can replace the "faith_court_shoes_white" search text in the token by encoding the search text to generate an output of "42ad42". Any type of encoding function or formula can be used. Optionally, the output of the encoding can be limited to a certain number of characters in order to optimize the benefit of using an encoded token in place of a token including search text or an item identifier. Additionally, if the chat management server apparatus 12 determines in step 210 that the token should be encoded, then the chat management server apparatus 12 in this example stores at least the encoded portion of the search token as associated with the search criteria (e.g., the search text) in the encoded token database 36 so that the encoded token can subsequently be decoded, as described and illustrated in more detail later with reference to step 804 of FIG. 8.

[0062] Referring back to FIG. 2, in step 212, the chat management server apparatus 12 provides the search or item token to the host representative device 14. Referring more specifically to FIG. 7, the exemplary host chat panel 400 is illustrated with an item set link 702 corresponding to the search token "@42ad42", which is an encoded search token in this example. Accordingly, upon receipt by the host representative device 14 of the search token, the host chat panel 400 inserts the search token text into the input field 404.

[0063] Upon the host representative selecting the send button 700, using the input/display device 53 of the host representative device 14, the input text including the search token is sent to the chat management server 14, which routes the text to the customer chat panel currently rendered on the client device 18 using an established connection. Upon display of the input text in the host chat panel 400, as well as the customer chat panel currently rendered on the client device 18, the search token becomes the item set link 702.

[0064] Accordingly, the definition of the chat panel 400 sent to the host representative device 14 is configured to insert a token returned in step 212 into the input field 404 and render the token as a hyperlink (the item set link 702 in this example) used as described and illustrated in more detail with reference to FIG. 8. Optionally, at least the customer chat panel is configured, based on its definition, to render any text sent from a host representative in a chat session that includes the first or second special character as an item set link or an item link, respectively.

[0065] Referring more specifically to FIG. 8, an exemplary method of generating an item preview panel based on a token will now be described. In step 800 in this example, the chat management server apparatus 12 receives a request from the client device 18 for a preview panel.

The request includes a token and is sent in response to a selection by the customer, using the input/display device 43 of the client device 18, of a hyperlink including the token that was rendered in the customer chat panel. The hyperlink can be rendered in the customer chat panel subsequent to the host representative submitting a message including the token. The token can be manually entered by the host representative or provided by the chat management server apparatus 12, as described and illustrated earlier with reference to step 212.

[0066] In step 802, the chat management server apparatus 12 determines when the token included in the request for the preview panel received in step 800 is encoded. In order to determine whether the token is decoded, the chat management server apparatus 12 can compare the token to entries of the encoded token database 36 to determine where there is a match in this example, although other methods of determining whether the token is encoded can also be used. If the chat management server apparatus 12 determines that the token is encoded, then the Yes branch is taken to step 804.

[0067] In step 804, the chat management server apparatus 12 decodes the token. In order to decode the token in this example, the chat management server apparatus 12 retrieves the actual value of the portion of the token not including the special character from the matching entry of the encoded token database 36. The actual value could have been stored in the encoded token database 36 as described and illustrated in more detail earlier with reference to step 210 of FIG. 2. Accordingly, in this example, the chat management server 12 can obtain the "faith_court_shoes_white" actual value by decoding the "42ad42" encoded token value. Other methods of encoding or decoding the tokens, including using a reversible encoding function that does not require a database look-up, can also be used. Subsequent to decoding the token, or if the chat management server apparatus 12 determines in step 802 that the token is not encoded and the No branch is taken, the chat management server apparatus 12 proceeds to step 806.

[0068] In step 806, the chat management server apparatus 12 retrieves item content based on the token. In order to retrieve the item content, in this example, the chat management server apparatus 12 first determines whether the token is a search or an item token based on whether the token includes the first or second special character. If the chat management server apparatus 12 determines that the token is a search token, then the portion of the token not including the first special character, or the decoded actual value in examples in which step 804 is performed, is processed by the search web service 34. Accordingly, the search web service 32 searches the item content catalog 30, as described and illustrated earlier with reference to step 206 of FIG. 2, using the token value as the search text in order to identify and retrieve responsive item content.

[0069] However, if the chat management server apparatus 12 determines that the token is an item token, then the portion of the token not including the first special character, or the decoded actual value in examples in which step 804 is performed, is processed by the preview web service 34. Accordingly, the preview web service 34 searches the item content catalog 30 using the token value to identify and retrieve content for an item having a unique item identifier matching the token value.

[0070] In step 808, the chat management server apparatus 12 provides a preview panel definition, including HTML fragment(s) including the content retrieved in step 806, to the client device 18 in response to the request for the preview panel received in step 800. The preview panel definition is configured to, when rendered, cause the web browser 46 of the client device 18 to generate a preview panel that includes the item content. The preview panel can be generated a pop-up window, an overlay, or any other type of display that does not require navigation by the web browser 46 away from the customer chat panel. Additionally, the preview panel can be a multi-item preview panel with a navigation structure or a single item preview panel based on whether the token, received with the request for the preview panel in step 800, is a search token or an item token, respectively.

[0071] Optionally, at least a portion of the content displayed by the preview panel is, or another portion of the preview panel includes, a link that is selectable by the customer using the client device 18 in order to allow the customer to navigate to a different web page associated with the item that provides additional content. Also optionally, at least a portion of the content displayed by the preview panel is, or another portion of the preview panel includes, a link that is selectable by the customer using the client device 18 in order to allow the customer to navigate to an item purchase web page or add the item to a shopping cart, for example. Other types of links and other content can also be provided in the preview panel.

[0072] Referring more specifically to FIG. 9, an exemplary customer chat panel 900 with the item set link 702 corresponding to the encoded search token "@42ad42" and an exemplary multi-item preview panel 902 are illustrated. In this example, the multi-item preview panel 902 is generated, based on the definition provided in step 808, subsequent to the customer selecting the item set link 702 using the input/display device of the client device 18. The multi-item preview panel 902 includes the content 602(1) for one of the items that the host representative selected to be referred to the customer in this example.

[0073] Additionally, the multi-item preview panel 902 includes a navigation structure, which in this example includes a next button 904, which facilitates navigation between content associated with a plurality of items, including at least the content 602(2) of another of the items that the host representative selected to be referred to the customer in this example. Other types of navigational structures can also be used.

[0074] Referring more specifically to FIG. 10, the exemplary customer chat panel 900 of FIG. 9 with an item link 1000 and an exemplary single item preview panel 1002 are illustrated. In this example, the item link 1000 corresponds to an item token "#3611369" which includes a unique item identifier "3611369" for a single item selected by the host representative in an iteration of steps 202-212 of FIG. 2 performed prior to the example iteration described and illustrated in detail earlier. The item preview panel 1002 includes content 1004 for a navy colored shoe item that the customer in this example is not interested in. Instead, the customer indicated to the host representative a preference for a white shoe and the host representative submitted the request received in step 202 in the example iteration described and illustrated earlier in order to identify items responsive to the customer's preference.

[0075] Accordingly, with this technology, representatives of website hosts can more easily and effectively refer website users to preview content for items in a chat context. The references can be sent using tokens which are generally, or can be encoded to be, shorter than URLs associated with web pages corresponding to the items. Additionally, host representatives can refer users to preview content for item(s) by reusing tokens thereby facilitating relatively quick responsiveness. Moreover, items identified by a host representative can advantageously be displayed by a user in a preview panel without requiring the user to navigate away from the chat panel or the current web page and thereby significantly improving the functioning of the user's client device.

[0076] Having thus described the basic concept of the invention, it will be rather apparent to those skilled in the art that the foregoing detailed disclosure is intended to be presented by way of example only, and is not limiting. Various alterations, improvements, and modifications will occur and are intended to those skilled in the art, though not expressly stated herein. These alterations, improvements, and modifications are intended to be suggested hereby, and are within the spirit and scope of the invention. Additionally, the recited order of processing elements or sequences, or the use of numbers, letters, or other designations therefore, is not intended to limit the claimed processes to any order except as may be specified in the claims. Accordingly, the invention is limited only by the following claims and equivalents thereto.

CLAIMS

What is claimed is:

1. A method for facilitating references in a chat context, the method comprising:

receiving, by a chat management server apparatus, a search request via a search panel provided in response to a user interaction with a chat panel;

generating, by the chat management server apparatus, a token including a special character and generated based on search text in the search request or a unique identifier for one of a plurality of items identified based on a search performed using the search text;

providing, by the chat management server apparatus, the token to a source of the search request for inclusion in the chat panel as a hyperlink;

receiving, by the chat management server apparatus, a preview panel request in response to a user interaction with the hyperlink, the preview panel request including the token;

retrieving, by the chat management server apparatus, content for the items or for the one item based on the special character included in the token; and

providing, by the chat management server apparatus, the content to a source of the preview panel request.

2. The method as set forth in claim 1, further comprising, prior to generating the token:

retrieving, by the chat management server apparatus, the content for the items based on the search text;

providing, by the chat management server apparatus, the content for the items to the source of the search request for inclusion in a search result panel; and

receiving, by the chat management server apparatus, a selection of one or more of the items from the source of the search request.

3. The method as set forth in claim 2, wherein the selection is of all of the items and the generating further comprises generating a search token comprising a first special character and the request for the preview panel includes the search token.

4. The method as set forth in claim 2, wherein the selection is of one or more

of the items, the generating further comprises generating an item token for each of the one or more items, each item token comprises a second special character, and the request for the preview panel includes at least one of the item tokens.

5. The method as set forth in claim 3, wherein the search token includes at least one or more terms included in the search text.

6. The method as set forth in claim 1, wherein:

the generating further comprises encoding the search text or the unique identifier and storing the encoded search text or the encoded unique identifier in an encoded token database as associated with the corresponding search text or unique identifier, wherein the token is generated based on the encoded search text or the encoded unique identifier; and

the retrieving further comprises decoding the token comprising comparing the token to the encoded token database to retrieve the search text or unique identifier.

7. The method as set forth in claim 1, wherein the retrieving further comprises:

determining when the token included in the preview panel request is a search token based on a match of a specified character of the token with a first special character;

performing a search of an item content catalog using the search text, as determined based on the token included in the preview panel request , to retrieve the content for the items, when the token included in the preview panel request is determined to be the search token; and

retrieving content for the one item based on the unique identifier, as determined based on the token included in the preview panel request, when the token included in the preview panel request is not determined to be the search token.

8. A non-transitory computer readable medium having stored thereon instructions for facilitating references in a chat context comprising executable code which when executed by a processor, causes the processor to perform steps comprising:

receiving a search request via a search panel provided in response to a user interaction with a chat panel;

generating a token including a special character and generated based on search text in the search request or a unique identifier for one of a plurality of items identified

based on a search performed using the search text;

providing the token to a source of the search request for inclusion in the chat panel as a hyperlink;

receiving a preview panel request in response to a user interaction with the hyperlink, the preview panel request including the token;

retrieving content for the items or for the one item based on the special character included in the token; and

providing the content to a source of the preview panel request.

9. The non-transitory computer readable medium as set forth in claim 8, further having stored thereon instructions that when executed by the processor cause the processor to perform steps further comprising, prior to generating the search token or the item token:

retrieving the content for the items based on the search text;

providing the content for the items to the source of the search request for inclusion in a search result panel; and

receiving a selection of one or more of the items from the source of the search request.

10. The non-transitory computer readable medium as set forth in claim 9, wherein the selection is of all of the items and the generating further comprises generating a search token comprising a first special character and the request for the preview panel includes the search token.

11. The non-transitory computer readable medium as set forth in claim 9, wherein the selection is of one or more of the items, the generating further comprises generating an item token for each of the one or more items, each item token comprises a second special character, and the request for the preview panel includes at least one of the item tokens.

12. The non-transitory computer readable medium as set forth in claim 10, wherein the search token includes at least one or more terms included in the search text.

13. The non-transitory computer readable medium as set forth in claim 8, wherein:

the generating further comprises encoding the search text or the unique identifier and storing the encoded search text or the encoded unique identifier in an encoded token database as associated with the corresponding search text or unique identifier, wherein the token is generated based on the encoded search text or the encoded unique identifier; and

the retrieving further comprises decoding the token comprising comparing the token to the encoded token database to retrieve the search text or unique identifier.

14. The non-transitory computer readable medium as set forth in claim 8, wherein the retrieving further comprises:

determining when the token included in the preview panel request is a search token based on a match of a specified character of the token with a first special character;

performing a search of an item content catalog using the search text, as determined based on the token included in the preview panel request , to retrieve the content for the items, when the token included in the preview panel request is determined to be the search token; and

retrieving content for the one item based on the unique identifier, as determined based on the token included in the preview panel request, when the token included in the preview panel request is not determined to be the search token.

15. A chat management server apparatus, comprising a processor and a memory coupled to the processor which is configured to be capable of executing programmed instructions comprising and stored in the memory to:

receive a search request via a search panel provided in response to a user interaction with a chat panel;

generate a token including a special character and generated based on search text in the search request or a unique identifier for one of a plurality of items identified based on a search performed using the search text;

provide the token to a source of the search request for inclusion in the chat panel as a hyperlink;

receive a preview panel request in response to a user interaction with the hyperlink, the preview panel request including the token;

retrieve content for the items or for the one item based on the special character included in the token; and

provide the content to a source of the preview panel request.

16. The chat management server apparatus as set forth in claim 15, wherein the processor coupled to the memory is further configured to be capable of executing programmed instructions further comprising and stored in the memory to, prior to generating the search token or the item token:

retrieve the content for the items based on the search text;

provide the content for the items to the source of the search request for inclusion in a search result panel; and

receive a selection of one or more of the items from the source of the search request.

17. The chat management server apparatus as set forth in claim 16, wherein the selection is of all of the items, the processor coupled to the memory is further configured to be capable of executing at least one additional programmed instruction further comprising and stored in the memory to generate a search token comprising a first special character, and the request for the preview panel includes the search token.

18. The chat management server apparatus as set forth in claim 16, wherein the selection is of one or more of the items, the processor coupled to the memory is further configured to be capable of executing at least one additional programmed instruction further comprising and stored in the memory to generate an item token for each of the one or more items, each item token comprises a second special character, and the request for the preview panel includes at least one of the item tokens.

19. The chat management server apparatus as set forth in claim 17, wherein the search token includes at least one or more terms included in the search text.

20. The chat management server apparatus as set forth in claim 15, wherein the processor coupled to the memory is further configured to be capable of executing programmed instructions further comprising and stored in the memory to:

encode the search text or the unique identifier and storing the encoded search text or the encoded unique identifier in an encoded token database as associated with the corresponding search text or unique identifier, wherein the token is generated based on the encoded search text or the encoded unique identifier; and

decode the token comprising comparing the token to the encoded token database to retrieve the search text or unique identifier.

21. The chat management server apparatus as set forth in claim 15, wherein the processor coupled to the memory is further configured to be capable of executing programmed instructions further comprising and stored in the memory to:

determine when the token included in the preview panel request is a search token based on a match of a specified character of the token with a first special character;

perform a search of an item content catalog using the search text, as determined based on the token included in the preview panel request , to retrieve the content for the items, when the token included in the preview panel request is determined to be the search token; and

retrieve content for the one item based on the unique identifier, as determined based on the token included in the preview panel request, when the token included in the preview panel request is not determined to be the search token.

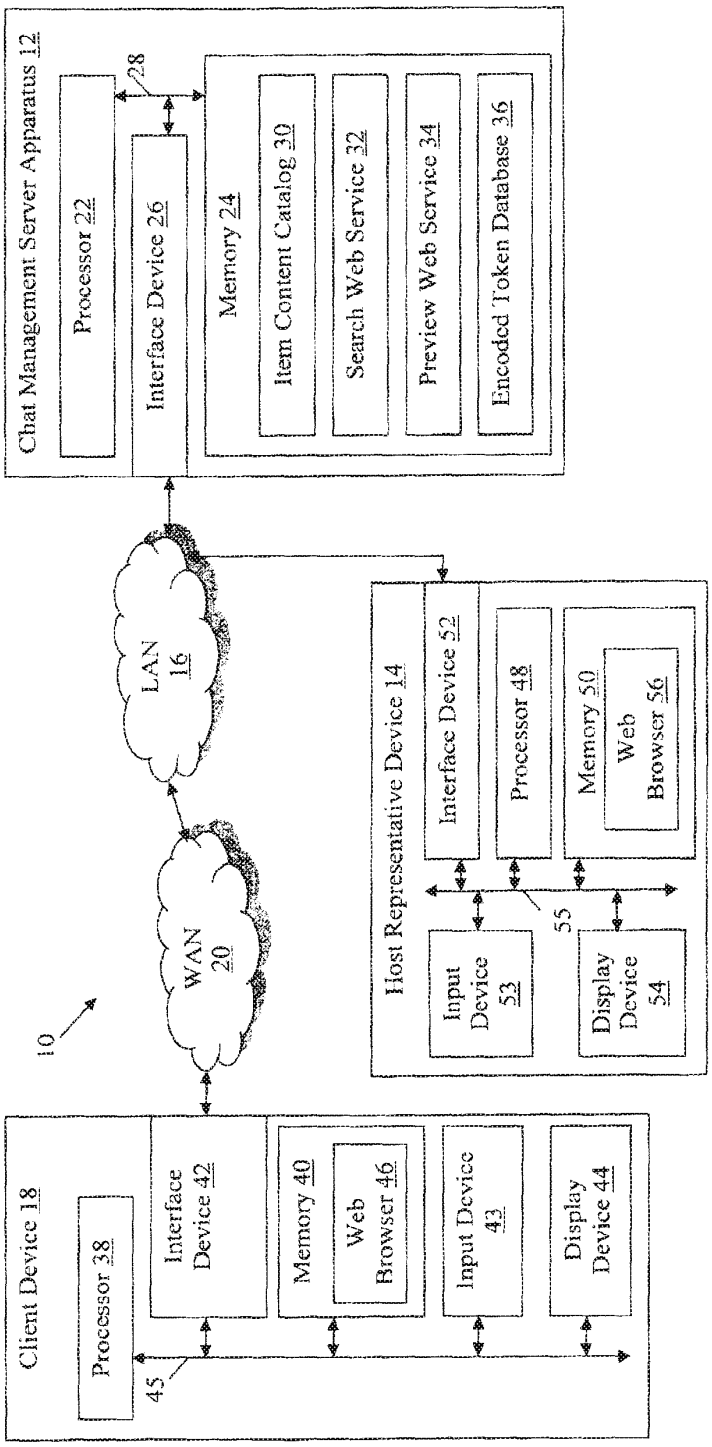


FIG. 1

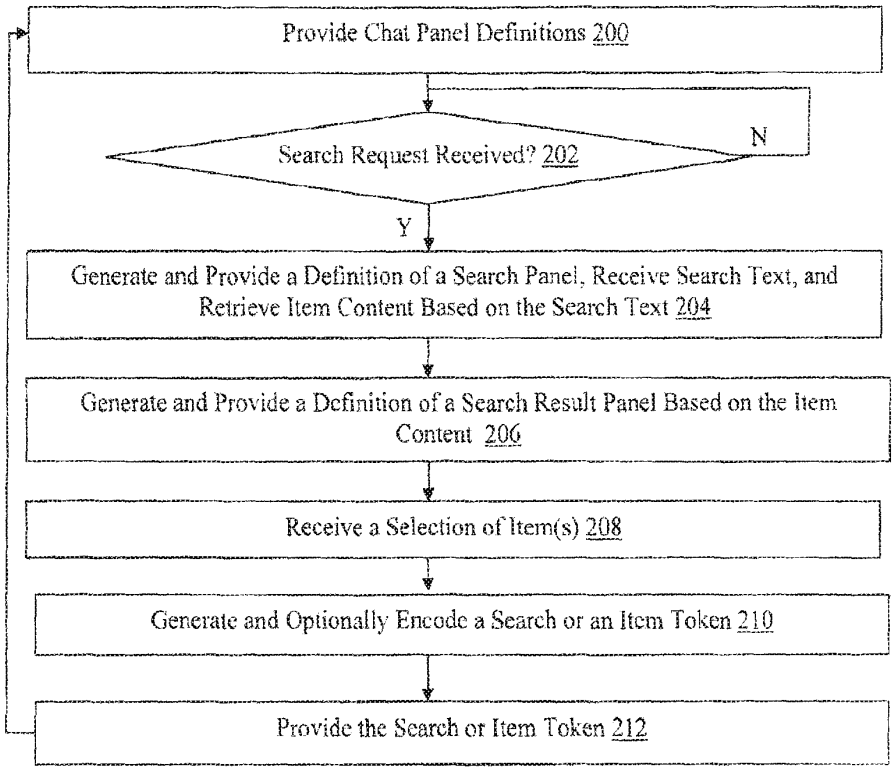


FIG. 2

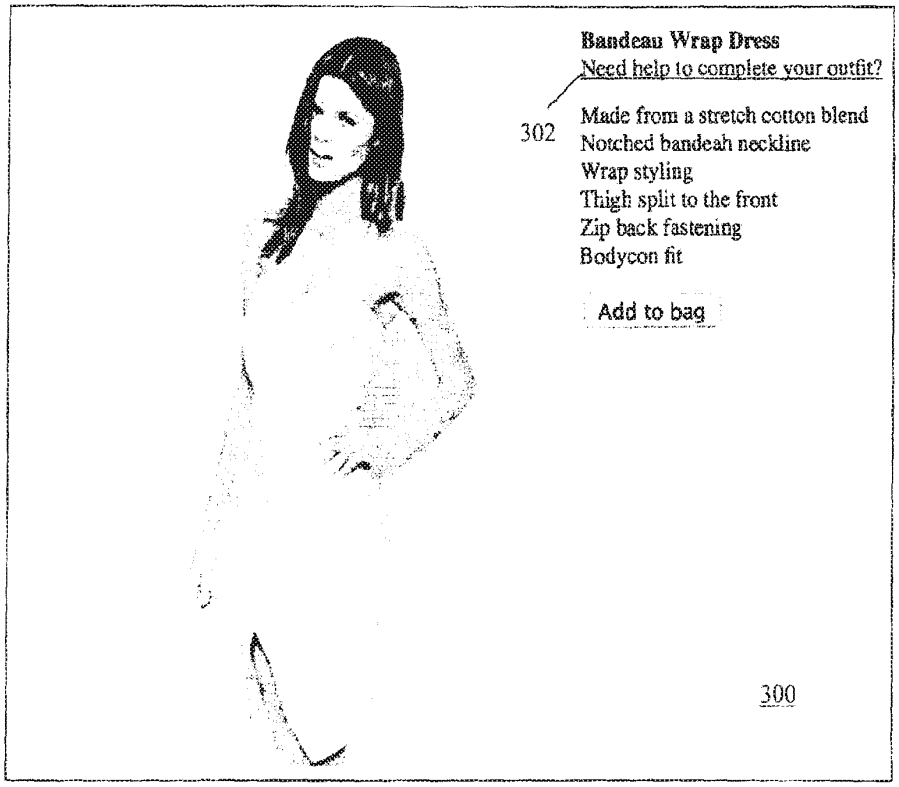


FIG. 3

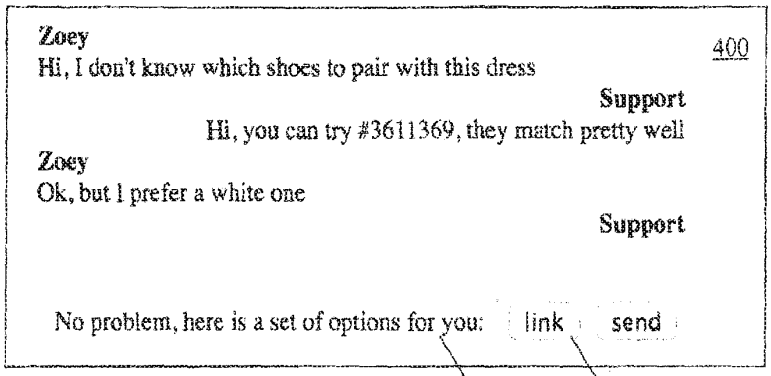


FIG. 4

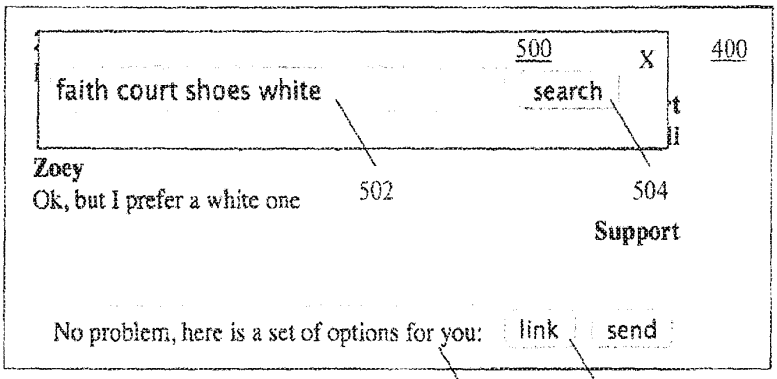


FIG. 5

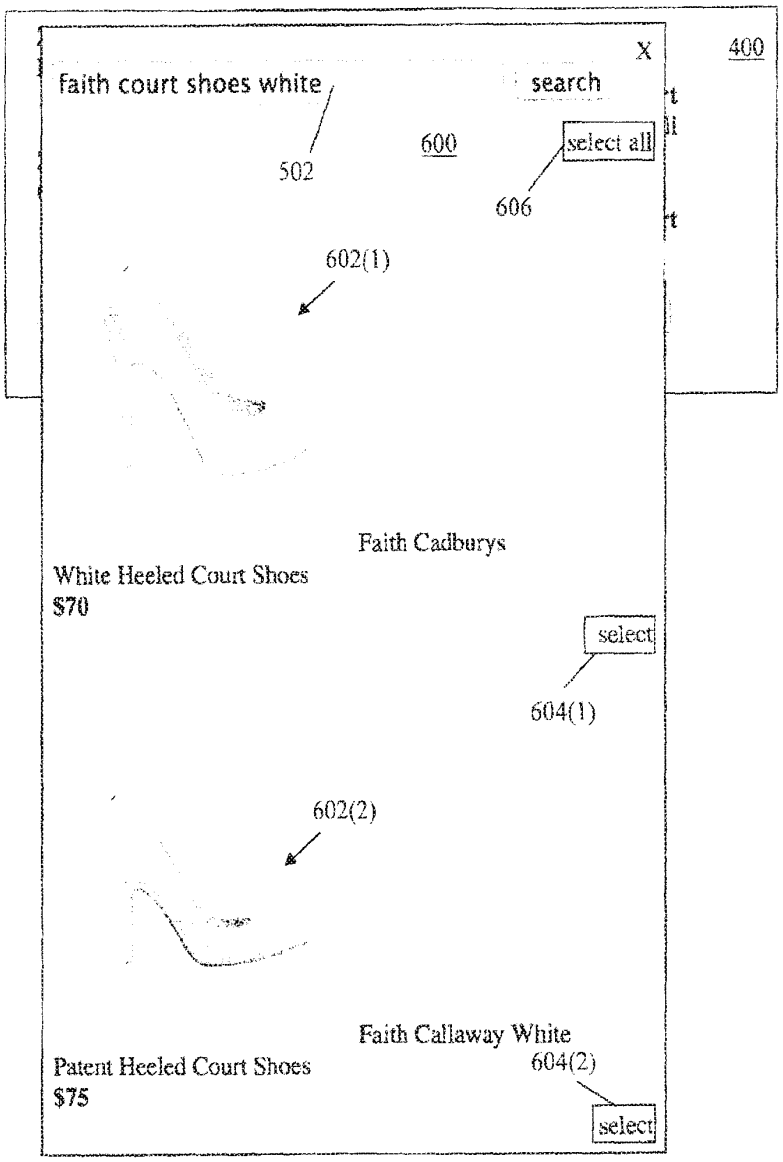


FIG. 6

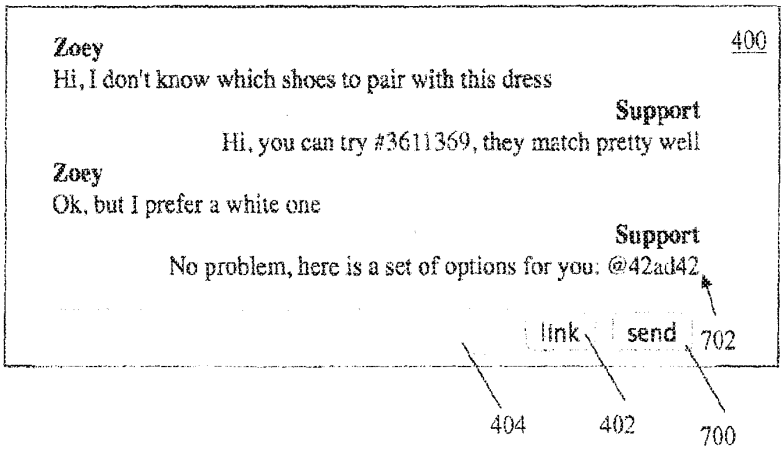


FIG. 7

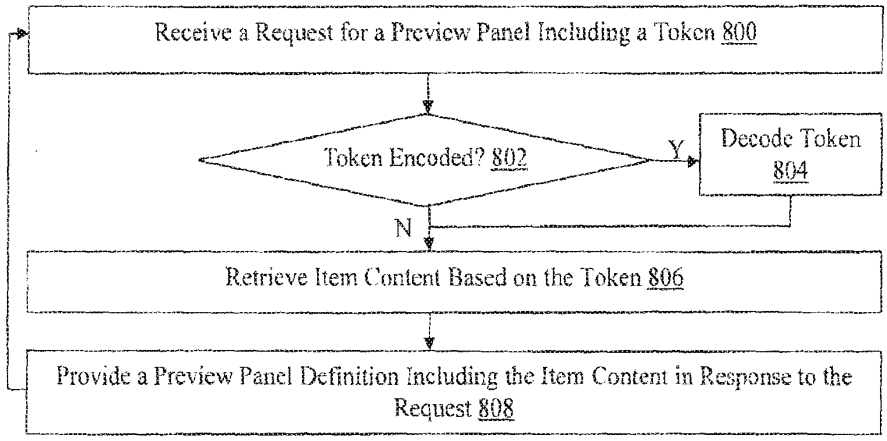


FIG. 8

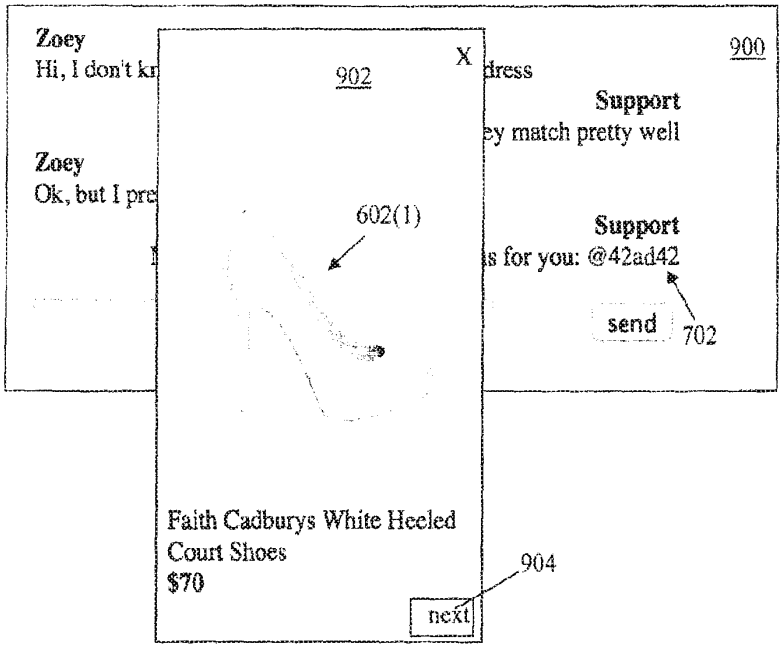


FIG. 9

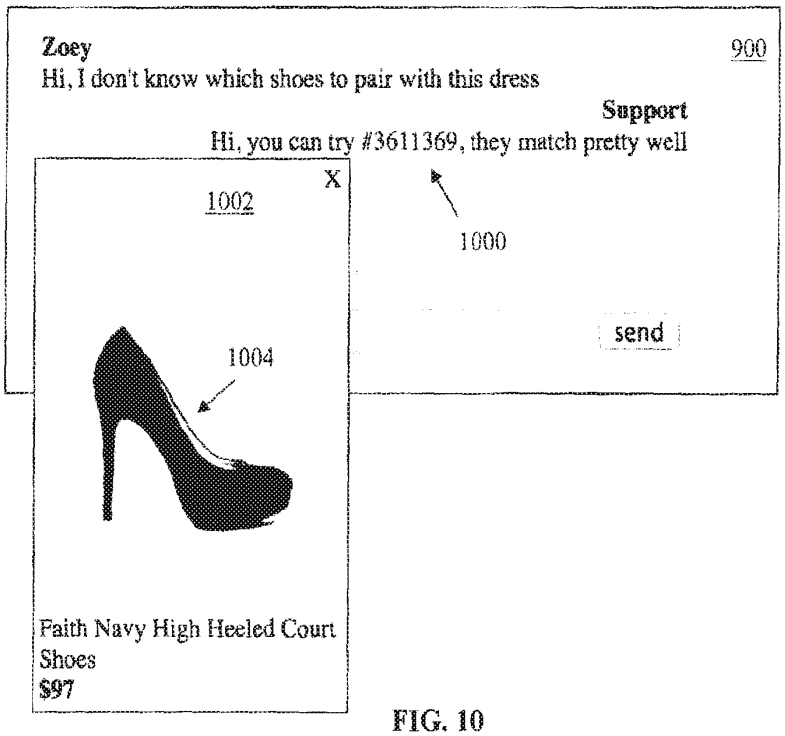


FIG. 10

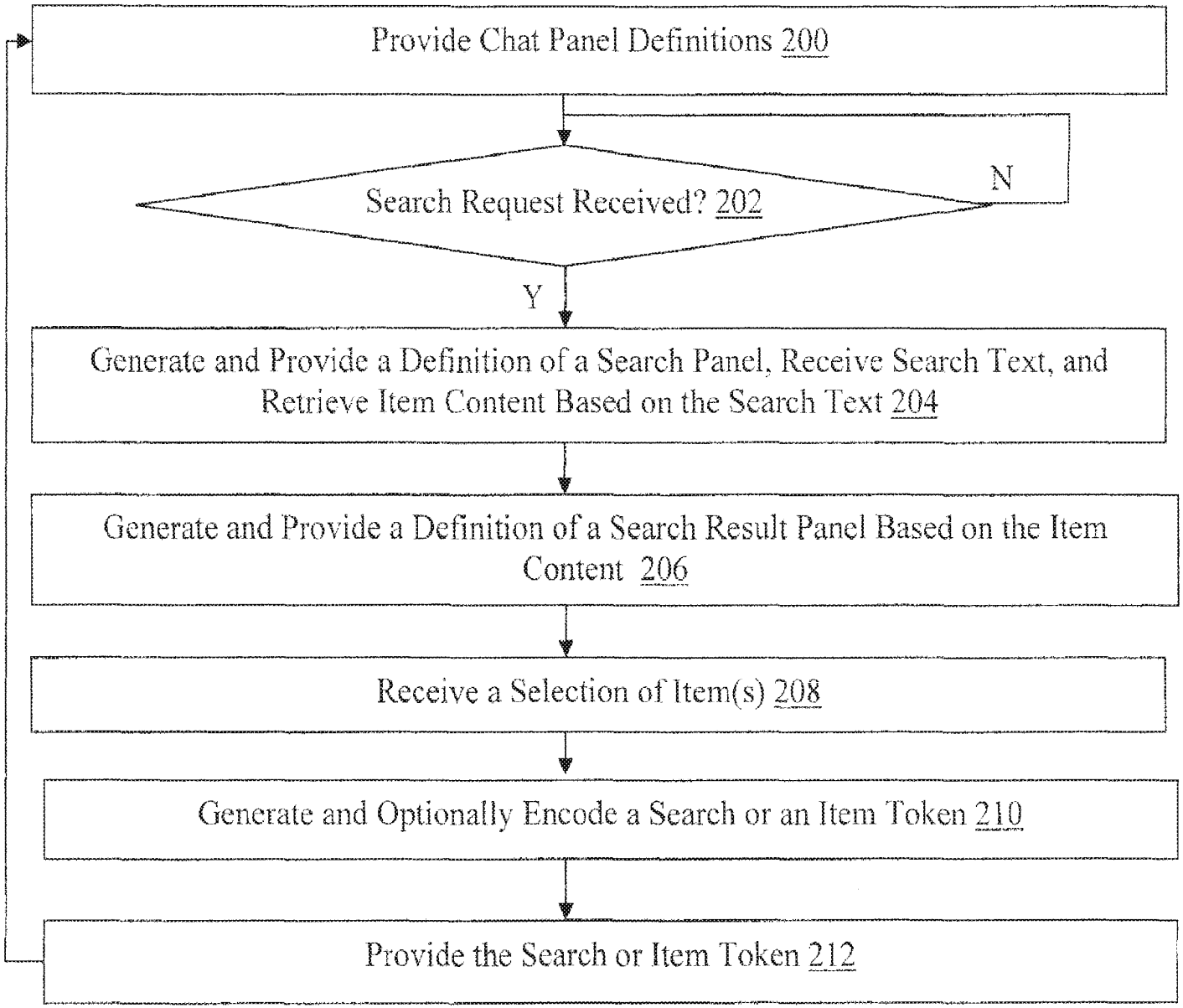


FIG. 2

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 2 363 995 B1

(12) EUROPEAN PATENT SPECIFICATION

(45) Date of publication and mention of the grant of the patent:
29.07.2015 Bulletin 2015/31

(51) Int Cl.:
H04L 29/06 (2006.01)

(21) Application number: **11156549.5**

(22) Date of filing: **02.03.2011**

(54) **Methods for optimizing a web content proxy server and devices thereof**

Verfahren zur Optimierung eines Proxy-Servers für Webinhalt und Vorrichtungen dafür

Procédé d'optimisation d'un serveur proxy à contenu Web et dispositifs associés

(84) Designated Contracting States:
AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO RS SE SI SK SM TR

(30) Priority: **02.03.2010 US 660637**

(43) Date of publication of application:
07.09.2011 Bulletin 2011/36

(73) Proprietor: **Usablenet Inc.**
New York, NY 10019 (US)

(72) Inventor: **Scoda, Enrico**
33035, Martignacco (IT)

(74) Representative: **De Ros, Alberto et al**
Notarbertolo & Gervasi S.p.A.
Corso di Porta Vittoria 9
20122 Milan (IT)

(56) References cited:
WO-A2-02/23375
US-A- 5 826 242
US-A1- 2001 054 020
US-A1- 2004 044 768

Printed by Jouve, 75001 PARIS (FR)

EP 2 363 995 B1

Description

FIELD OF THE INVENTION

[0001] This invention generally relates to proxy servers and, more particularly, methods for optimizing web content proxy servers and apparatuses thereof.

BACKGROUND

[0002] A web content optimization server is a proxy server that optimizes web pages obtained from remote web servers for client devices with special requirements, such as mobile phones, PDAs, and smartphones. Every time a client device requests a web page, the web content optimization server downloads the original page from a remote web server, applies some customized rules to extract relevant content, and adapts it to fit the needs of the requesting client device. By way of example, the web content optimization server may remove javascript, linearize content, and adapt the original page to a smaller screen layout for the requesting client device.

[0003] In computing, a cookie, such as a tracking cookie, browser cookie, and HTTP cookie, is a small piece of text stored by a web browser on the client device. A cookie includes one or more name-value pairs containing data, such as user preferences, shopping cart contents, the identifier for a server-based session, or other data used by websites.

[0004] Web content optimization servers need to save cookies to enable the client devices to interact with the original website at the remote web servers in the correct way. Accordingly, web content optimization servers store these cookies in an internal memory and associate them with the corresponding session from each client device so that when the same client device sends a request for a new page, the web content optimization server will load the matching cookies and send them to the remote web server to get the page to process. Unfortunately, storing the cookies for these client devices causes problems with scalability, security, and privacy of the web content optimization servers.

US 2004/044768 A1 discloses a reverse proxy performing cookie conversion between user terminals and web servers.

US 2001/054020 A1 discloses reformatting cookies received from a supplier by an intermediate server before sending them to the client.

WO 02/23375 A2 discloses converting data within the cookie to a format that can be stored within the final Markup Language Document that is returned to a client device.

US 5 826 242 A discloses interaction between clients and web servers using cookies.

SUMMARY OF THE INVENTION

[0005] The present invention defines a method according to claim 1 and a computer readable medium according to claim 6 and a web proxy apparatus according to claim 7. Further embodiments are set forth in the dependent claims 2-5 and 8-11.

[0006] This technology provides a number of advantages including providing a method, computer readable medium and an apparatus that optimizes implementation of a web content proxy server for interactions involving cookies between client devices and remote web servers. With this technology, original server cookies are transformed by the web content proxy server to web optimized client cookies which are transmitted to the client devices requesting the web pages for storage and use with subsequent requests.

[0007] This technology provides greater scalability because the web optimized client cookies are stored in the web browser at the client device, not in memory at the web content proxy server. As a result, the web content proxy server does not face any issues with respect to memory storage capacity due to the number of sessions with cookies for client devices. The web content proxy server can use the same memory whether there are 100 or 1,000,000 or more client devices engaged in sessions with the remote web servers through the web content proxy server.

[0008] Additionally, this technology provides greater security and privacy because the web content proxy server does not contain a centralized database of original server cookies which contain session information from client devices browsing pages of web sites. Instead, these original server cookies are translated into web optimized client cookies which are then dispersed out among the client devices. As a result, the web content proxy server does not have any stored cookies from interactions between client devices and remote web servers that could be used to steal identity or other confidential information of these client devices.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009]

FIG. 1 is a block diagram of an exemplary environment with an optimized web content proxy server;

FIG. 2A is an example of a HTTP request for a web page from a remote web server;

FIG. 2B is an example of a HTTP response with an original server cookie from a remote web server to a HTTP request;

FIG. 2C is an example of a HTTP response containing the web optimized client cookie generated from the original server cookie received shown in FIG. 2B;

FIG. 2D is an example of another HTTP request with the web optimized client cookie shown in FIG. 2C

for a web page from a remote web server.

FIG. 2E is an example of the another HTTP request with the web optimized client cookie shown in FIG. 2D translated into the original server cookie for transmission to the remote web server with the another get request;

FIG. 3 is a flow chart of an example of a method for generating a web optimized client cookie from an original server cookie to optimize implementation of a web content proxy server; and

FIG. 4 is a flow chart of an example of a method for transforming a web optimized client cookie back to an original server cookie to optimize implementation of a web content proxy server.

DETAILED DESCRIPTION

[0010] An exemplary environment 10 in which a web content proxy server 12 is optimized is illustrated in FIG. 1. The exemplary environment 10 includes a web content proxy server or apparatus 12, client devices 14(1)-14(n), web server devices 16(1)-16(n), and communication networks 18(1)-18(2), although other numbers and types of systems, devices, and/or elements in other configurations and environments with other communication network topologies can be used. This technology provides a number of advantages including providing a method, computer readable medium and an apparatus that optimizes implementation of a web content proxy server for interactions involving cookies between client devices and remote web servers.

[0011] Referring more specifically to FIG. 1, the web content proxy server 12 optimizes the handling of original server cookies from the web server devices 16(1)-16(n) for requesting client devices 14(1)-14(n) and the handling of web optimized client cookies, although the web content proxy server 12 can provide other numbers and types of functions. Although one web content proxy server 12 is shown, other numbers and types of web content proxy devices and systems can be used.

[0012] The web content proxy server 12 includes a central processing unit (CPU) or processor 13, a memory 15, and an interface system 17 which are coupled together by a bus 19 or other link, although other numbers and types of components, parts, devices, systems, and elements in other configurations and locations can be used. The processor 13 in the web content proxy server 12 executes a program of stored instructions one or more aspects of the present invention as described and illustrated by way of the embodiments herein, although the processor could execute other numbers and types of programmed instructions. The memory 15 in the web content proxy server 12 stores these programmed instructions for one or more aspects of the present invention as described and illustrated herein, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, DVD ROM, or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to the processor 13, can be used for the memory 15 in the web content proxy server 12. In these embodiments, the memory 15 includes a core module 21 and a cookie module 23 which store programmed instructions for one or more aspects of the present invention as described and illustrated herein, although the memory can comprise other types and numbers of systems, devices, and elements in other configurations which store other data. The cookie module 23 includes programmed instructions and/or logic configured to translate an original server cookie into a web optimized client cookie and to extract the original server cookie when a web optimized client cookie is received, although the cookie module 23 can have other types and numbers of functions as described and illustrated herein.

[0013] The interface system 17 in the web content proxy server 12 is used to operatively couple and communicate between the web content proxy server 12 and the client devices 14(1)-14(n) and the web server devices 16(1)-16(n) via the communication networks 18(1) and 18(2), although other types and numbers of communication networks with other types and numbers of connections and configurations can be used. By way of example only, the communication networks 18(1) and 18(2) can use TCP/IP over Ethernet and industry-standard protocols, including HTTP, HTTPS, WAP, and SOAP, although other types and numbers of communication networks, such as a direct connection, a local area network, a wide area network, modems and phone lines, e-mail, and wireless and hardwire communication technology, each having their own communications protocols, can be used.

[0014] Each of the client devices 14(1)-14(n) enables a user to request, get and interact with web pages from one or more web sites hosted by the web server devices 16(1)-16(n) through the web content proxy server 12 via one or more communication networks, although one or more of the client devices 14(1)-14(n) could access content and utilize other types and numbers of applications from other sources and could provide a wide variety of other functions for the user. Although multiple client devices 14(1)-14(n) are shown, other numbers and types of user computing systems could be used. In this example, the client devices 14(1)-14(n) comprise mobile devices with Internet access that permit a website form page or other retrieved data to be displayed, although each of the client devices 14(1)-14(n). By way of example only, one or more of the client devices 14(1)-14(n) can comprise smart phones, personal digital assistants, or computers.

[0015] Each of client devices 14(1)-14(n) in this exam-

ple is a computing device that includes a central processing unit (CPU) or processor 20, a memory 22, user input device 24, a display 26, and an interface system 28, and which are coupled together by a bus 30 or other link, although one or more of client devices 14(1)-14(n) can include other numbers and types of components, parts, devices, systems, and elements in other configurations. The processor 20 in each of client devices 14(1)-14(n) executes a program of stored instructions for one or more aspects of the present invention as described and illustrated herein, although the processor could execute other numbers and types of programmed instructions.

[0016] The memory 22 in each of the client devices 14(1)-14(n) stores these programmed instructions for one or more aspects of the present invention as described and illustrated herein as well as the web optimized client cookies, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to processor 20 can be used for the memory 22 in each of the client devices 14(1)-14(n.

[0017] The user input device 24 in each of the client devices 14(1)-14(n) is used to input selections, such as requests for a particular website form page or to enter data in fields of a form page, although the user input device could be used to input other types of data and interact with other elements. The user input device can include keypads, touch screens, and/or vocal input processing systems although other types and numbers of user input devices can be used.

[0018] The display 26 in each of the client devices 14(1)-14(n) is used to show data and information to the user, such as website page by way of example only. The display in each of the client devices 14(1)-14(n) is a phone screen display, although other types and numbers of displays could be used depending on the particular type of client device.

[0019] The interface system 28 in each of the client devices 14(1)-14(n) is used to operatively couple and communicate between the client devices 14(1)-14(n) and the web content proxy server 12 and web server devices 16(1)-16(n) over the communication networks 18(1) and 18(2), although other types and numbers of communication networks with other types and numbers of connections and configurations can be used.

[0020] The web server devices 16(1)-16(n) provide one or more pages from one or more web sites for use by one or more of the client devices 14(1)-14(n) via the web content proxy server 12, although the web server devices 16(1)-16(n) can provide other numbers and types of applications and/or content and can have provide other numbers and types of functions. Although web server devices 16(1)-16(n) are shown for ease of illustration and discussion, other numbers and types of web server systems and devices can be used.

[0021] Each of the web server devices 16(1)-16(n) include a central processing unit (CPU) or processor, a memory, and an interface system which are coupled together by a bus or other link, although each of the web server devices 16(1)-16(n) could have other numbers and types of components, parts, devices, systems, and elements in other configurations and locations can be used. The processor in each of the web server devices 16(1)-16(n) executes a program of stored instructions one or more aspects of the present invention as described and illustrated by way of the embodiments herein, although the processor could execute other numbers and types of programmed instructions.

[0022] The memory in each of the web server devices 16(1)-16(n) stores these programmed instructions for one or more aspects of the present invention as described and illustrated by way of the embodiments, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, DVD ROM, or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to the processor, can be used for the memory in each of the web server devices 16(1)-16(n).

[0023] The interface system in each of the web server devices 16(1)-16(n) is used to operatively couple and communicate between the web server devices 16(1)-16(n) and the web content proxy server 12 and the client devices 14(1)-14(n) via communication networks 18(1) and 18(2), although other types and numbers of communication networks with other types and numbers of connections and configurations can be used.

[0024] Although embodiments of the web content proxy server 12, the client devices 14(1)-14(n), and the web server devices 16(1)-16(n), are described and illustrated herein, each of the client devices 14(1)-14(n), the web content proxy server 12, and the web server devices 16(1)-16(n), can be implemented on any suitable computer system or computing device. It is to be understood that the devices and systems of the embodiments described herein are for exemplary purposes, as many variations of the specific hardware and software used to implement the embodiments are possible, as will be appreciated by those skilled in the relevant art(s).

[0025] Furthermore, each of the systems of the embodiments may be conveniently implemented using one or more general purpose computer systems, microprocessors, digital signal processors, and micro-controllers, programmed according to the teachings of the embodiments, as described and illustrated herein, and as will be appreciated by those ordinary skill in the art.

[0026] In addition, two or more computing systems or devices can be substituted for any one of the systems in

any embodiment of the embodiments. Accordingly, principles and advantages of distributed processing, such as redundancy and replication also can be implemented, as desired, to increase the robustness and performance of the devices and systems of the embodiments. The embodiments may also be implemented on computer system or systems that extend across any suitable network using any suitable interface mechanisms and communications technologies, including by way of example only telecommunications in any suitable form (e.g., voice and modem), wireless communications media, wireless communications networks, cellular communications networks, G3 communications networks, Public Switched Telephone Network (PSTNs), Packet Data Networks (PDNs), the Internet, intranets, and combinations thereof.

[0027] The embodiments may also be embodied as a computer readable medium having instructions stored thereon for one or more aspects of the present invention as described and illustrated by way of the embodiments herein, as described herein, which when executed by a processor, cause the processor to carry out the steps necessary to implement the methods of the embodiments, as described and illustrated herein.

[0028] An exemplary method for generating a web optimized client cookie from an original server cookie to optimize implementation of the web content proxy server 12 will now be described with reference to FIGS. 1-2C and 3. In step 50, in this example one of the client devices 14(1)-14(n) via a web browser requests a page A.html at the website, "www.example.com" as shown in one example in FIG. 2A. This request is transmitted to the web content proxy server 12 which processes and transmits the request to the one of the web servers 16(1)-16(n) hosting the website "www.example.com." The hosting one of the web servers 16(1)-16(n) provides a response in this example for the requested page A.html which also contains an original server cookie "SESSION" to the web content proxy server 12 as shown in FIG. 2B. In this example, SESSION has a value equal to "1234", the domain is equal to ".example.com" and the path is equal to "/". This response uses the HTTP header Field "Set-Cookie". The cookie is a string formed by the pair "name=value", followed by optional attributes, like those in this example indicating the server domain(s) and path accepting this cookie. Although one illustrative example is described herein, this technology can be used with specifications for all cookies.

[0029] Next, in step 52 the web content proxy server 12 determines whether the original server cookie includes the domain attribute for the requested web page provided by the hosting one of the web servers 16(1)-16(n). If in step 52 the web content proxy server 12 determines the original server cookie does not include the domain attribute, then the No branch is taken to step 54. In step 54, the web content proxy server 12 extracts the domain attribute from the requested web page provided by the hosting one of the web servers 16(1)-16(n). If in step 52 the web content proxy server 12 determines the original server cookie does include the domain attribute, then the Yes branch is taken to step 56.

[0030] In step 56, the web content proxy server 12 determines whether the original server cookie includes the path attribute for the requested web page provided by the hosting one of the web servers 16(1)-16(n). If in step 52 the web content proxy server 12 determines the original server cookie does not include the path attribute, then the No branch is taken to step 58. In step 58, the web content proxy server 12 extracts the path attribute from the requested web page provided by the hosting one of the web servers 16(1)-16(n). If in step 56 the web content proxy server 12 determines the original server cookie does include the path attribute, then the Yes branch is taken to step 60.

[0031] In step 60, the web content proxy server 12 generates a name for the new web optimized client cookie by concatenating the original name with domain and path, separated by spaces, although other manners for generating the new name can be used. The resulting new name is "universal resource locator encoded" to keep conformance to the cookie specification. Additionally, the resulting new name is unique even if different domains contain cookies with the same name. This new name contains all the information necessary for the web content proxy server 12 to extract the original server cookie later as described in greater detail by reference to FIG. 4,

[0032] In step 62, the web content proxy server 12 forms the new web optimized client cookie by associating the new name with the same value of the original server cookie, with the domain attribute not being specified, and with the path attribute being associated with a value "/", although other values can be used, such as one for the path attribute that corresponds to a prefix associated with this optimization method (by way of example only "/mt/").

[0033] By way of example only, when the web content proxy server 12 receives a response with the original server cookie as shown in FIG. 2B, the web content proxy server 12 generates a web optimized client cookie as shown in FIG. 2C. More specifically, the original server cookie: SESSION=1234; domain=.example.com; and path=/ is transformed by the web content proxy server 12 to a web optimized client cookie: SESSION+.example.com+%252F=1234; path=/mt/. Accoridingly, in this illustrative example the new web optimized client cookie name SESSION+.example.com+%252F is the encoded version of the concatenation of original server cookie name, domain attribute and path attribute, although other orders and manners for forming this name can be used. In this example, the new path attribute corresponds to a prefix "/mt/" associated with this optimization method.

[0034] Next, in step 64 the web content proxy server 12 copies the remaining attributes in the original server cookie, such as an expiration date for the original server cookie by way of example, in the web optimized client cookie, although other amounts of the remaining attributes could be copied and other information also could

be added.

[0035] Next, in step 66 the original server cookie which has been translated into the web optimized client cookie is now provided to the core module 21 in the web content proxy server 12. The core module 21 includes programmed instructions and/or logic to manage the transmission of the web optimized client cookie from the web content proxy server 12 to the requesting one of the client devices 14(1)-14(n). The web browser at the requesting one of the client devices 14(1)-14(n) receives and saves the web optimized client cookie in the memory 22 at the requesting one of the client devices 14(1)-14(n). In this illustrative example, the web optimized client cookie shown in FIG. 2C is stored in the memory 22 at the requesting one of the client devices 14(1)-14(n).

[0036] Referring now to FIGS. 1, 2D-2E and 4, an exemplary method for translating a web optimized client cookie back to an original server cookie to optimize the implementation of the web content proxy server 12 will now be described. In step 100, in this example one of the client devices 14(1)-14(n) via a web browser submits another request to the web content proxy server 12 for page B.html at the website, "www.example.com" as shown in one example in FIG. 2D. This request includes a web optimized client cookie which in this example comprises SESSION+.example.com+%252F.

[0037] In step 102, the web content proxy server 12 extracts the original server cookie name and the domain and path attributes from the name of the web optimized client cookie. In this illustrative example, the original server cookie name and the domain and path attributes are extracted by the web content proxy server from the name: SESSION+.example.com+%252F.

[0038] In step 104, the web content proxy server 12 determines whether the extracted domain and path attributes identify a web optimized client cookie that is a match to universal resource locator for the requested web page. If in step 104 the web content proxy server 12 determines the extracted domain and path attributes identify a web optimized client cookie is not a match, then the No branch is taken to step 106. In step 106, the web content proxy server 12 submits the request to the hosting one of the web servers 16(1)-16(n) hosting the request page without an original server cookie. In this illustrative example, the requested page is "B.html." If in step 104 the web content proxy server 12 determines the extracted domain and path attributes identify a web optimized client cookie is a match, then the Yes branch is taken to step 108.

[0039] In step 108, the web content proxy server 12 associates the extracted name from the web optimized client cookie with the value for the original server cookie. The extracted name and value comprise the original server cookie which is appended to the HTTP cookie header fields of the request to be sent to the one of the web servers 16(1)-16(n) hosting the requested web page. In this illustrative example, the extracted name SESSION is associated with the value 1234.

[0040] In step 110, the web content proxy server 12 submits the request with the reconstituted original server cookie to the one of the web servers 16(1)-16(n) hosting the requested page. In this illustrative example, the request with the reconstituted original server cookie as shown in FIG. 2E is transmitted to the one of the web servers 16(1)-16(n) hosting the requested page.

[0041] Accordingly, as illustrated and described herein this technology provides a number of advantages including providing a method, computer readable medium and an apparatus that optimizes implementation of a web content proxy server for interactions involving cookies between client devices and remote web servers. With this technology, the web content proxy server is much more scalable because of the reduced memory storage demands. Additionally, with this technology the web content proxy server poses a much lower security and privacy risk to information provided by the client devices 14(1)-14(n).

[0042] Having thus described the basic concept of the invention, it will be rather apparent to those skilled in the art that the foregoing detailed disclosure is intended to be presented by way of example only, and is not limiting. Various alterations, improvements, and modifications will occur and are intended to those skilled in the art, though not expressly stated herein. These alterations, improvements, and modifications are intended to be suggested hereby, and are within the scope of the invention. Additionally, the recited order of processing elements or sequences, or the use of numbers, letters, or other designations therefore, is not intended to limit the claimed processes to any order except as may be specified in the claims. Accordingly, the invention is limited only by the following claims and equivalents thereto.

Claims

1. A method for optimizing a web content proxy server (12), the method comprising:

   - obtaining at a proxy server (12) a web page with an original server cookie from one of one or more web content servers (16(1)-16(n) in response to a request for a web page from one of one or more client devices (14(1)-14(n));
   - generating at the proxy server (12) a web optimized client cookie based on the original server cookie;
   - providing with the proxy server (12) the obtained web page with the generated web optimized client cookie to the requesting one of one or more client devices (14(1)-14(n));

   wherein generating the web optimized client cookie comprises forming a new name for the web optimized client cookie by concatenating an original name of the original server cookie followed by a do-

main attribute of the obtained web page followed by a path attribute of the obtained web page;

- translating at the proxy server (12) the web optimized client cookie associated with another request from one of the one or more client devices (14(1)-14(n)) to the original server cookie;
- determining with the proxy server (12) whether the translated original server cookie corresponds with a universal resource locator of the another request;

and

- providing with the proxy server (12) the another request with the translated original server cookie to one of the one or more web servers (16(1)-16(n)) when the translated original server cookie is determined to correspond.

2. The method as set forth in claim 1 wherein the forming further comprises obtaining the domain and path attributes of the obtained web page from the original server cookie.

3. The method as set forth in claim 1 or 2 wherein the forming further comprises obtaining the domain and path attributes of the obtained web page from a universal resource locator for the obtained web page.

4. The method as set forth in claim 1 or 2 or 3 wherein the generating further comprises associating the new name with at least an original value for the original server cookie and the path attribute for the requested web page.

5. The method as set forth in claim 4 further comprising associating one or more original attributes to the translated original server cookie.

6. A computer readable medium having stored thereon instructions for optimizing a proxy server (12) comprising machine executable code which when executed by at least one processor, causes the processor to perform the steps of the method set forth in any of claims from 1 to 5.

7. A web proxy apparatus (12) comprising:

- one or more processors;
- a memory coupled to the one or more processors which are configured to execute programmed instructions stored in the memory comprising:
- obtaining a web page with an original server cookie from one or more web servers (16(1)-16(n)) in response to a request for a web page from one of one or more client devices (14(1)-14(n));
- generating a web optimized client cookie based on the original server cookie;
- transmitting the obtained web page with the generated web optimized client cookie to the requesting one of one or more client devices (14(1)-14(n)),

wherein generating the web optimized client cookie comprises forming a new name for the web optimized client cookie by concatenating an original name of the original server cookie followed by a domain attribute of the obtained web page followed by a path attribute of the obtained web page;

- translating at the proxy server (12) the web optimized client cookie associated with another request from one of the one or more client devices (14(1)-14(n)) to the original server cookie;
- determining with the proxy server (12) whether the translated original server cookie corresponds with a universal resource locator of the another request;

and

- providing with the proxy server (12) the another request with the translated original server cookie to one of the one or more web servers (16(1)-16(n)) when the translated original server cookie is determined to correspond.

8. The apparatus as set forth in claim 7 wherein the one or more processors is further configured to execute programmed instructions stored in the memory for the forming further comprising obtaining the domain and path attributes of the obtained web page from the original server cookie.

9. The apparatus as set forth in claim 7 or 8 wherein the one or more processors is further configured to execute programmed instructions stored in the memory for the forming further comprising obtaining the domain and path attributes of the obtained web page from a universal resource locator for the obtained web page.

10. The apparatus as set forth in claim 7 or 8 or 9 wherein the one or more processors is further configured to execute programmed instructions stored in the memory for the generating the web optimized client cookie further comprising associating the new name with at least an original value for the original server cookie and the path attribute for the requested web page.

11. The apparatus as set forth in claim 10 wherein the one or more processors is further configured to ex-

ecute programmed instructions stored in the memory comprising associating one or more original attributes to the translated original server cookie.

Patentansprüche

1. Verfahren zum Optimieren eines Web-Content-Proxy-Servers (12), wobei das Verfahren umfasst, dass:

   - an einem Proxy-Server (12) eine Webseite mit einem ursprünglichen Server-Cookie von einem von einem oder mehreren Web-Content-Servern (16(1) - 16(n)) in Ansprechen auf eine Anforderung hinsichtlich einer Webseite von einer von einer oder mehreren Client-Einrichtungen (14(1) - 14(n)) erhalten wird;
   - an dem Proxy-Server (12) ein weboptimiertes Client-Cookie basierend auf dem ursprünglichen Server-Cookie erzeugt wird;
   - mit dem Proxy-Server (12) die erhaltene Webseite mit dem erzeugten weboptimierten Client-Cookie für die Anfordernde einer oder mehrerer Client-Einrichtungen (14(1) - 14(n)) bereitgestellt wird;

   wobei das Erzeugen des weboptimierten Client-Cookies umfasst, dass ein neuer Name für das weboptimierte Client-Cookie gebildet wird, indem ein ursprünglicher Name des ursprünglichen Server-Cookies gefolgt von einem Domain-Attribut der erhaltenen Webseite gefolgt von einem Pfadattribut der erhaltenen Webseite verknüpft werden;

   - an dem Proxy-Server (12) das weboptimierte Client-Cookie, das mit einer anderen Anforderung von einer der einen oder mehreren Client-Einrichtungen (14(1) - 14(n)) in Verbindung steht, in das ursprüngliche Server-Cookie übersetzt wird;
   - mit dem Proxy-Server (12) ermittelt wird, ob das übersetzte ursprüngliche Server-Cookie einem Universal Resource Locator der anderen Anforderung entspricht; und
   - mit dem Proxy-Server (12) die andere Anforderung mit dem übersetzten ursprünglichen Server-Cookie einem des einen oder der mehreren Web-Server (16(1) - 16(n)) bereitgestellt wird, wenn eine Entsprechung des übersetzten ursprünglichen Server-Cookies ermittelt wird.

2. Verfahren nach Anspruch 1,
   wobei das Bilden ferner umfasst, dass das Domain- und das Pfadattribut der erhaltenen Webseite von dem ursprünglichen Server-Cookie erhalten werden.

3. Verfahren nach Anspruch 1 oder 2,
   wobei das Bilden ferner umfasst, dass das Domain- und das Pfadattribut der erhaltenen Webseite von einem Universal Resource Locator für die erhaltene Webseite erhalten werden.

4. Verfahren nach Anspruch 1 oder 2 oder 3,
   wobei das Erzeugen ferner umfasst, dass der neue Name mit zumindest einem ursprünglichen Wert für das ursprüngliche Server-Cookie und dem Pfadattribut für die angeforderte Webseite in Verbindung gebracht wird.

5. Verfahren nach Anspruch 4,
   das ferner umfasst, dass ein oder mehrere ursprüngliche Attribute mit dem übersetzten ursprünglichen Server-Cookie in Verbindung gebracht werden.

6. Von einem Computer lesbares Medium mit darauf gespeicherten Anweisungen zum Optimieren eines Proxy-Servers (12), die von einer Maschine ausführbaren Code umfassen, der bei einer Ausführung durch zumindest einen Prozessor bewirkt, dass der Prozessor die Schritte des Verfahrens nach einem der Ansprüche 1 bis 5 durchführt.

7. Web-Proxy-Vorrichtung (12), umfassend:

   - einen oder mehrere Prozessoren;
   - einen Speicher, der mit dem einen oder den mehreren Prozessoren gekoppelt ist, die ausgestaltet sind, um programmierte Anweisungen auszuführen, die in dem Speicher gespeichert sind und umfassen:
   - Erhalten einer Webseite mit einem ursprünglichen Server-Cookie von einem oder mehreren Web-Servern (16(1) - 16(n)) in Ansprechen auf eine Anforderung hinsichtlich einer Webseite von einer von einer oder mehreren Client-Einrichtungen (14(1) - 14(n));
   - Erzeugen eines weboptimierten Client-Cookies basierend auf dem ursprünglichen Server-Cookie;
   - Übertragen der erhaltenen Webseite mit dem erzeugten weboptimierten Client-Cookie an die Anfordernde einer oder mehrerer Client-Einrichtungen (14(1) - 14(n)),

   wobei das Erzeugen des weboptimierten Client-Cookies umfasst, dass ein neuer Name für das weboptimierte Client-Cookie gebildet wird, indem ein ursprünglicher Name des ursprünglichen Server-Cookies gefolgt von einem Domain-Attribut der erhaltenen Webseite gefolgt von einem Pfadattribut der erhaltenen Webseite verknüpft werden;

   - Übersetzen des weboptimierten Client-Cookies, das mit einer anderen Anforderung von einer der einen oder mehreren Client-Ein-

richtungen (14(1) - 14(n)) in Verbindung steht, in das ursprüngliche Server-Cookie an dem Proxy-Server (12);
- Ermitteln, ob das übersetzte ursprüngliche Server-Cookie einem Universal Resource Locator der anderen Anforderung entspricht, mit dem Proxy-Server (12); und
- Bereitstellen, mit dem Proxy-Server (12), der anderen Anforderung mit dem übersetzten ursprünglichen Server-Cookie für einen des einen oder der mehreren Web-Server (16(1) - 16(n)), wenn eine Entsprechung des übersetzten ursprünglichen Server-Cookies ermittelt wird.

8. Vorrichtung nach Anspruch 7, wobei der eine oder die mehreren Prozessoren ferner ausgestaltet ist oder sind, um programmierte Anweisungen auszuführen, die in dem Speicher gespeichert sind, sodass das Bilden ferner umfasst, dass das Domain- und das Pfadattribut der erhaltenen Webseite von dem ursprünglichen Server-Cookie erhalten werden.

9. Vorrichtung nach Anspruch 7 oder 8, wobei der eine oder die mehreren Prozessoren ferner ausgestaltet ist oder sind, um programmierte Anweisungen auszuführen, die in dem Speicher gespeichert sind, sodass das Bilden ferner umfasst, dass das Domain- und das Pfadattribut der erhaltenen Webseite von einem Universal Resource Locator für die erhaltene Webseite erhalten werden.

10. Vorrichtung nach Anspruch 7 oder 8 oder 9, wobei der eine oder die mehreren Prozessoren ferner ausgestaltet ist oder sind, um programmierte Anweisungen auszuführen, die in dem Speicher gespeichert sind, sodass das Erzeugen des weboptimierten Client-Cookies ferner umfasst, dass der neue Name mit zumindest einem ursprünglichen Wert für das ursprüngliche Server-Cookie und dem Pfadattribut für die angeforderte Webseite in Verbindung gebracht wird.

11. Vorrichtung nach Anspruch 10, wobei der eine oder die mehreren Prozessoren ferner ausgestaltet ist oder sind, um programmierte Anweisungen auszuführen, die in dem Speicher gespeichert sind und ein Inverbindungbringen eines oder mehrerer ursprünglicher Attribute mit dem übersetzten ursprünglichen Server-Cookie umfassen.

Revendications

1. Procédé d'optimisation d'un serveur mandataire (12) de contenu Web, le procédé comprenant :

- l'obtention auprès d'un serveur mandataire (12) d'une page Web avec un témoin de connexion de serveur d'origine provenant d'un ou plusieurs serveurs de contenu Web (16(1) à 16(n)) en réponse à une demande d'une page Web provenant d'un ou plusieurs dispositifs clients (14(1) à (14(n)) ;
- la génération auprès du serveur mandataire (12) d'un témoin de connexion de client optimisé pour le Web d'après le témoin de connexion de serveur d'origine ;
- la fourniture avec le serveur mandataire (12) de la page Web obtenue avec le témoin de connexion de client optimisé pour le Web généré au dispositif demandeur des un ou plusieurs dispositifs clients (14(1) à (14(n)) ;

dans lequel la génération du témoin de connexion de client optimisé pour le Web comprend la formation d'un nouveau nom pour le témoin de connexion de client optimisé pour le Web par concaténation d'un nom d'origine du témoin de connexion de serveur d'origine suivi d'un attribut de domaine de la page Web obtenue suivi d'un attribut de chemin de la page Web obtenue ;

- la traduction auprès du serveur mandataire (12) du témoin de connexion de client optimisé pour le Web associé à une autre demande provenant de l'un des un ou plusieurs dispositifs clients (14(1) à 14(n)) au témoin de connexion de serveur d'origine ;
- la détermination avec le serveur mandataire (12) selon laquelle le témoin de connexion de serveur d'origine traduit correspond à une adresse universelle de l'autre demande ;

et

- la fourniture avec le serveur mandataire (12) de l'autre demande avec le témoin de connexion de serveur d'origine traduit à l'un des un ou plusieurs serveurs Web (16(1) à 16(n)) lorsque le témoin de connexion de serveur d'origine traduit est déterminé correspondre.

2. Procédé selon la revendication 1, dans lequel la formation comprend en outre l'obtention des attributs de domaine et de chemin de la page Web obtenue à partir du témoin de connexion de serveur d'origine.

3. Procédé selon la revendication 1 ou 2, dans lequel la formation comprend en outre l'obtention des attributs de domaine et de chemin de la page Web obtenue à partir d'une adresse universelle pour la page Web obtenue.

4. Procédé selon la revendication 1 ou 2 ou 3, dans

lequel la génération comprend en outre l'association du nouveau nom à au moins une valeur d'origine pour le témoin de connexion de serveur d'origine et l'attribut de chemin pour la page Web demandée.

5. Procédé selon la revendication 4, comprenant en outre l'association d'un ou plusieurs attributs d'origine au témoin de connexion de serveur d'origine traduit.

6. Support lisible par ordinateur dans lequel sont stockées des instructions permettant d'optimiser un serveur mandataire (12), comprenant un code exécutable par machine qui, lorsqu'il est exécuté par au moins un processeur, amène le processeur à réaliser les étapes du procédé selon l'une quelconque des revendications 1 à 5.

7. Appareil mandataire (12) pour le Web comprenant :

- un ou plusieurs processeurs ;
- une mémoire couplée aux un ou plusieurs processeurs qui sont configurés pour exécuter des instructions programmées stockées dans la mémoire, comprenant :
- l'obtention d'une page Web avec un témoin de connexion de serveur d'origine à partir d'un ou plusieurs serveurs Web (16(1) à 16(n)) en réponse à une demande d'une page Web provenant de l'un ou plusieurs dispositifs clients (14(1) à (14(n)) ;
- la génération d'un témoin de connexion de client optimisé pour le Web d'après le témoin de connexion de serveur d'origine ;
- la transmission de la page Web obtenue avec le témoin de connexion de client optimisé pour le Web généré au dispositif demandeur des un ou plusieurs dispositifs clients (14(1) à 14(n)),

dans lequel la génération du témoin de connexion de client optimisé pour le Web comprend la formation d'un nouveau nom pour le témoin de connexion de client optimisé pour le Web par concaténation d'un nom d'origine du témoin de connexion de serveur d'origine suivi d'un attribut de domaine de la page Web obtenue suivi d'un attribut de chemin de la page Web obtenue ;

- la traduction auprès du serveur mandataire (12) du témoin de connexion de client optimisé pour le Web associé à une autre demande provenant de l'un des un ou plusieurs dispositifs clients (14(1) à 14(n)) au témoin de connexion de serveur d'origine ;
- la détermination avec le serveur mandataire (12) selon laquelle le témoin de connexion de serveur d'origine traduit correspond à une adresse universelle de l'autre demande ;

et

- la fourniture avec le serveur mandataire (12) de l'autre demande avec le témoin de connexion de serveur d'origine traduit à l'un des un ou plusieurs serveurs Web (16(1) à 16(n)) lorsque le témoin de connexion de serveur d'origine traduit est déterminé correspondre.

8. Appareil selon la revendication 7, dans lequel les un ou plusieurs processeurs sont en outre configurés pour exécuter des instructions programmées stockées dans la mémoire pour la formation comprenant en outre l'obtention des attributs de domaine et de chemin de la page Web obtenue à partir du témoin de connexion de serveur d'origine.

9. Appareil selon la revendication 7 ou 8, dans lequel les un ou plusieurs processeurs sont en outre configurés pour exécuter des instructions programmées stockées dans la mémoire pour la formation comprenant en outre l'obtention des attributs de domaine et de chemin de la page Web obtenue à partir d'une adresse universelle pour la page Web obtenue.

10. Appareil selon la revendication 7 ou 8 ou 9, dans lequel les un ou plusieurs processeurs sont en outre configurés pour exécuter des instructions programmées stockées dans la mémoire pour la génération du témoin de connexion de client optimisé pour le Web comprenant en outre l'association du nouveau nom à au moins une valeur d'origine pour le témoin de connexion de serveur d'origine et l'attribut de chemin pour la page Web demandée.

11. Appareil selon la revendication 10, dans lequel les un ou plusieurs processeurs sont en outre configurés pour exécuter des instructions programmées stockées dans la mémoire comprenant l'association d'un ou plusieurs attributs d'origine au témoin de connexion de serveur d'origine traduit.

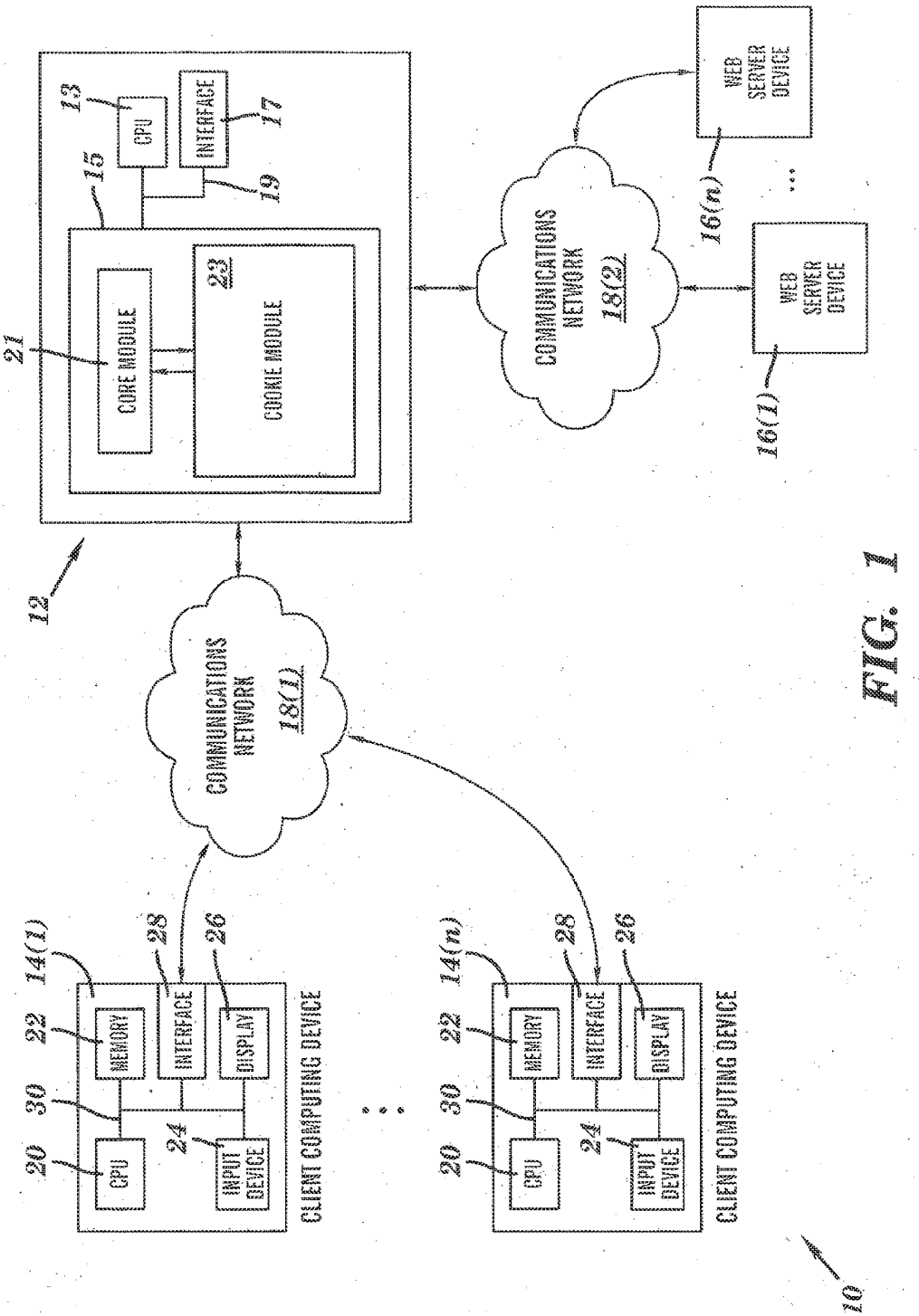


FIG. 1

```
GET /A.html HTTP/1.1
Host: www.example.com
Accept: */*
User-Agent: my-mobile-browser 1.0
```

FIG. 2A

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Content-Length: 5300
Set-Cookie: SESSION=1234; domain=.example.com; path=/
```

FIG. 2B

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Content-Length: 5300
Set-Cookie: SESSION+.example.com+%252F=1234; path=/mt/
```

FIG. 2C

```
GET /mt/www.example.com/B.html HTTP/1.1
User-Agent: my-mobile-browser 1.0
Host: m.proxy.com
Accept: */*
Cookie: SESSION+.example.com+%252F=1234
```

FIG. 2D

```
GET /B.html HTTP/1.1
User-Agent: my-mobile-browser 1.0
Host: www.example.com
Accept: */*
Cookie: SESSION=1234
```

FIG. 2E

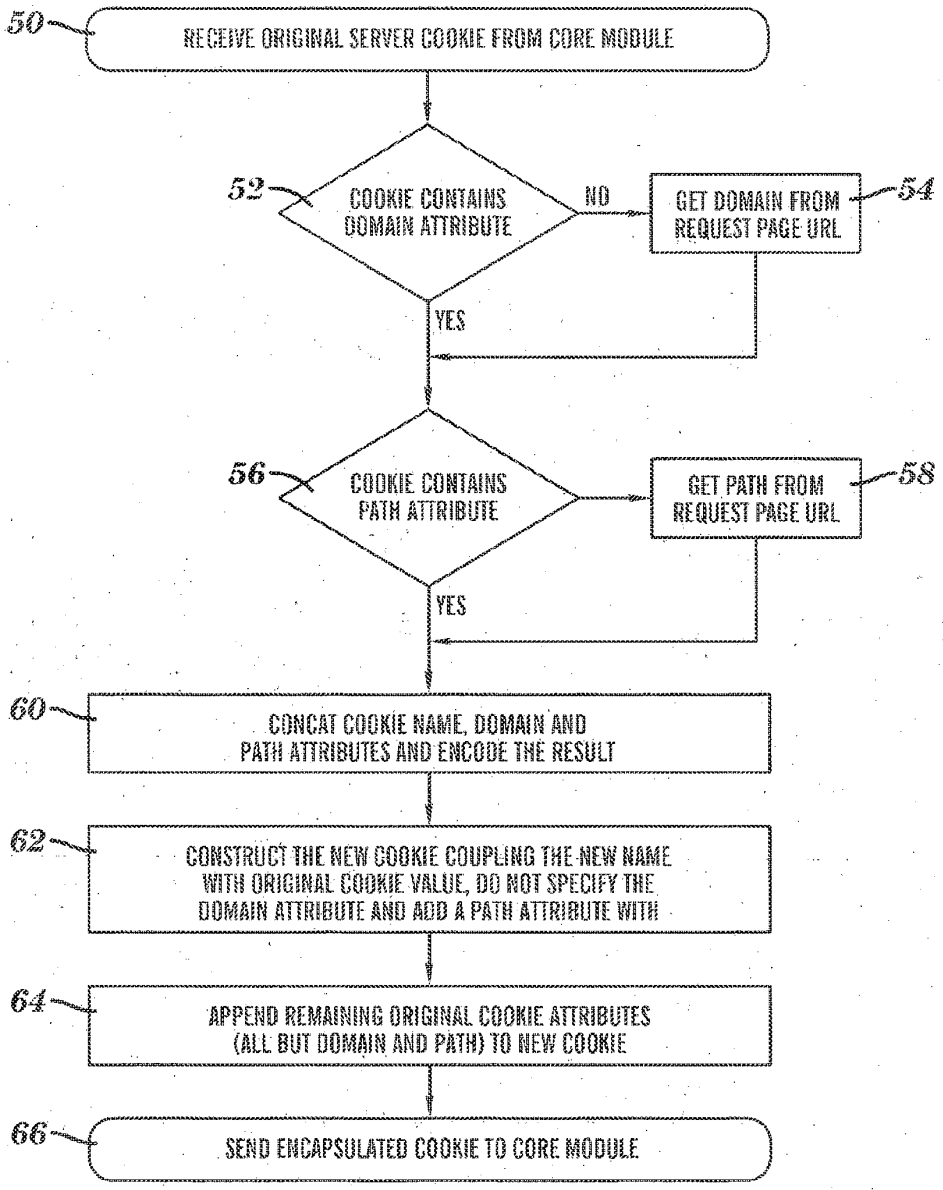


FIG. 3

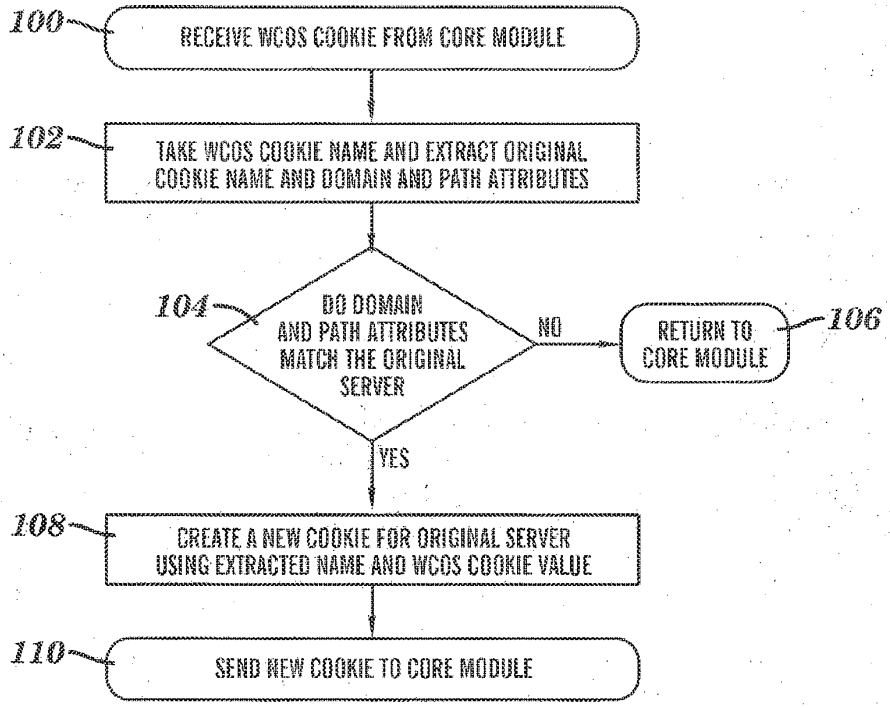


FIG. 4

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Patent documents cited in the description

- US 2004044768 A1 **[0004]**
- US 2001054020 A1 **[0004]**
- WO 0223375 A2 **[0004]**
- US 5826242 A **[0004]**

Europäisches Patentamt
European Patent Office
Office européen des brevets

SUPPLEMENTARY EUROPEAN SEARCH REPORT

Application Number
EP 10 81 6083

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (IPC)
	No further relevant documents disclosed -----		INV. G06F15/16 G06F17/30 G06F17/24
			TECHNICAL FIELDS SEARCHED (IPC) G06F

The supplementary search report has been based on the last set of claims valid and available at the start of the search.

Place of search	Date of completion of the search	Examiner
Berlin	17 January 2018	Écolivet, Stéphane

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

1

EPO FORM 1503 03.82 (P04C04)

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Veröffentlichungsnummer:
(11) Publication number: EP 2 580 686 A0
(11) Numéro de publication:

Internationale Anmeldung veröffentlicht durch die Weltorganisation für geistiges Eigentum unter der Nummer:

WO 2011/156743 (Art. 153(3) EPÜ).

International application published by the World Intellectual Property Organization under number:

WO 2011/156743 (Art. 153(3) EPC).

Demande internationale publiée par l'Organisation Mondiale de la Propriété Intellectuelle sous le numéro:

WO 2011/156743 (art. 153(3) CBE).

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
15 December 2011 (15.12.2011)

PCT

(10) International Publication Number
WO 2011/156743 A2

(51) International Patent Classification:
G06F 17/00 (2006.01) *G06F 9/455* (2006.01)

(21) International Application Number:
PCT/US2011/040026

(22) International Filing Date:
10 June 2011 (10.06.2011)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
12/802,670 11 June 2010 (11.06.2010) US

(71) Applicant *(for all designated States except US)*: **US-ABLENET INC.** [US/US]; 28 W. 23rd Street, 6th Floor, New York, NY 10010 (US).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: **SCODA, Enrico** [IT/IT]; Via Cividina 416/3, I-33035 Matignacco (ud) (IT).

(74) Agent: **LEINBERG, Gunnar, G.**; Leclairryan, 290 Linden Oaks, Rochester, NY 14625 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

(54) Title: METHODS FOR UTILIZING A JAVASCRIPT EMULATOR IN A WEB CONTENT PROXY SERVER AND DEVICES THEREOF

(57) Abstract: A method, computer readable medium and apparatus that utilize a JavaScript emulator in a proxy server to create and store an object model of a web page which has one or more JavaScript instruction sets. At least one of the one or more JavaScript instruction sets are extracted from the web page and a JavaScript field identifier is inserted into the web page to optimize the web page which is then provided.

WO 2011/156743 A2

METHODS FOR UTILIZING A JAVASCRIPT EMULATOR IN A WEB CONTENT PROXY SERVER AND DEVICES THEREOF

FIELD

[0001] This invention generally relates to proxy servers and, more particularly, to methods for utilizing a JavaScript emulator in a web content proxy server and apparatuses thereof.

BACKGROUND

[0002] A web content optimization server is a proxy server that optimizes web page interactions for client devices with special requirements, such as mobile phones, PDAs, and smartphones and for browsing tools used by visitors with special needs, such as visual impaired users. By way of example, a web content optimization server optimizes web page interactions as follows. A client device sends an HTTP request for a web page. The web content optimization server downloads the requested original web page from the content server listed in the received request. Next, the web content optimization server optimizes the content of the web page by applying transformation rules tailored to the requesting client device. This optimization process includes extracting the content relevant to the requesting client device and adapting this extracted content to fit the specifications of the requesting client device. By way of example, these transformations include JavaScript removal, content linearization, and small screen adaptation.

[0003] Although this process works well to optimize content for display at the requesting client device, the optimization process may fail when the original content heavily depends on JavaScript technology. For example, if the requested web page includes a JavaScript code or instruction set responsible for populating form fields, validating form submissions, retrieving data from external resources (based on AJAX technology), and even generating components that may change the structure of the web page, the page at the client device will not be able to properly function.

[0004] One example of how a JavaScript instruction set operates in a web page is illustrated with reference to FIGS. 2-4. Referring more specifically to FIG. 2, an exemplary web page of a login form 100 used to gain access to a web site is illustrated. To execute the login process, a user must enter the correct a user identification into the user id field 102 and a password in the illustrated password field 104 and then click on the login button 106 to submit the login inquiry. The specific action of clicking or otherwise engaging the login button 106 triggers the execution of the JavaScript validateLogin() function or instruction set illustrated in FIG. 3. This JavaScript validateLogin() function is responsible for validating that the user id field 102 and the password field 104 were properly filled before transmitting to the web content server for the requested web site.

[0005] Referring to FIG. 4, a scenario where the user entered the value "John" into the user id field 102, but did not enter any value into the password field 104 is illustrated. When the login button 106 is clicked or otherwise engaged, the validateLogin() function generates an error and provides the message "Password cannot be empty" to the HTML element of the displayed web page 100 whose attribute id has the value "error". When the validateFunction() generates an error the data entered by the user is not sent to the web content server for the requested web site.

[0006] If, by way of example, an existing web content optimization server optimizes the content of the web page 100 illustrated in FIGS. 2 and 4 by applying transformation rules to remove the JavaScript programmed instruction set that processes any values entered in user id field 102 and password field 104 and initiates an evaluation of entered values when login button 106 is clicked, any values entered could not be processed. Additionally, any attempt to click or otherwise engage the login button 106 would not provide the desired login functionality. As a result, the web page would have been optimized for viewing on the client device, but with substantially reduced functionality.

SUMMARY

[0007] A method for utilizing a JavaScript emulator in a proxy server includes creating and storing with a proxy server an object model of a web page which has one or more JavaScript instruction sets. At least one of the one or more JavaScript instruction sets are extracted from the web page and a JavaScript field identifier is inserted into the web page with the proxy server to optimize the web page which is then provided.

[0008] A computer readable medium having stored thereon instructions for utilizing a JavaScript emulator comprising machine executable code which when executed by at least one processor, causes the processor to perform steps including creating and storing an object model of a web page which has one or more JavaScript instruction sets. At least one of the one or more JavaScript instruction sets are extracted from the web page and a JavaScript field identifier is inserted into the web page to optimize the web page which is then provided.

[0009] A web proxy apparatus includes one or more processors and a memory coupled to the one or more processors which are configured to execute programmed instructions stored in the memory including creating and storing an object model of a web page which has one or more JavaScript instruction sets. At least one of the one or more JavaScript instruction sets are extracted from the web page and a JavaScript field identifier is inserted into the web page to optimize the web page which is then provided.

[00010] A method for utilizing a JavaScript emulator in a proxy server includes loading with the proxy server a stored object model of an original web page which had at least a portion of one or more JavaScript instruction sets previously extracted to form an optimized web page. Any data in a request received at the proxy server which corresponds to the loaded stored object model of the original web page is appended with the proxy server into the loaded stored object model of the original web page. The loaded stored object model of the original web page with any of the appended data is processed and then provided by the proxy server.

[00011] A computer readable medium having stored thereon instructions for utilizing a JavaScript emulator comprising machine executable code which when executed by at least one processor, causes the processor to perform steps including loading a stored object model of an original web page which had at least a portion of one or more JavaScript instruction sets previously extracted to form an optimized web page. Any data in a received request which corresponds to the loaded stored object model of the original web page is appended into the loaded stored object model of the original web page. The loaded stored object model of the original web page with any of the appended data is processed and then provided.

[00012] A web proxy apparatus includes one or more processors and a memory coupled to the one or more processors which are configured to execute programmed instructions stored in the memory including loading a stored object model of an original web page which had at least a portion of one or more JavaScript instruction sets previously extracted to form an optimized web page. Any data in a received request which corresponds to the loaded stored object model of the original web page is appended into the loaded stored object model of the original web page. The loaded stored object model of the original web page with any of the appended data is processed and then provided.

[00013] A method for utilizing a JavaScript emulator in a proxy server includes creating with a proxy server an object model of a web page which has at least one JavaScript instruction set. The at least one JavaScript instruction set from the web page is executed with the proxy server to obtain content. The at least one JavaScript instruction set is removed with the proxy server from the web page and the obtained content is appended to generate an optimized web page with the obtained content which is provided by the proxy server.

[00014] A computer readable medium having stored thereon instructions for utilizing a JavaScript emulator comprising machine executable code which when executed by at least one processor, causes the processor to perform steps including creating an object model of a web page which has at least one JavaScript instruction set. The at least one JavaScript instruction set from the web

page is executed to obtain content. The at least one JavaScript instruction set is removed from the web page and the obtained content is appended to generate an optimized web page with the obtained content which is provided.

[00015] A web proxy apparatus includes one or more processors and a memory coupled to the one or more processors which are configured to execute programmed instructions stored in the memory including creating an object model of a web page which has at least one JavaScript instruction set. The at least one JavaScript instruction set from the web page is executed to obtain content. The at least one JavaScript instruction set is removed from the web page and the obtained content is appended to generate an optimized web page with the obtained content which is provided.

[00016] This technology provides a number of advantages including providing a method, computer readable medium and an apparatus that further optimizes a web content proxy server through the utilization of a JavaScript emulator. With the JavaScript emulator, the web content proxy server can continue to apply optimizing transformation rules, while still enabling JavaScript functions in optimized HTTP requests, web pages, interactions with browsers, and other JavaScript events by to be executed with the JavaScript emulator. The JavaScript emulator emulates the behavior of all JavaScript objects that are supported by current web browsers and are used by web developers to access and change the components of web pages.

BRIEF DESCRIPTION OF THE DRAWINGS

[00017] FIG. 1 is a block diagram of an exemplary system environment with an optimized web content proxy server with a JavaScript emulator;

[00018] FIG. 2 is a screen shot of an exemplary login form;

[00019] FIG. 3 is an exemplary listing of JavaScript validation source code;

[00020] FIG. 4 is screen shot of an exemplary login form with an error message;

[00021] FIG. 5 is a flow chart of an example of a method for instantiating a JavaScript emulator and storing a document object model with an extracted JavaScript instruction set;

[00022] FIG. 6 is a flow chart of an example of a method for processing a received request at the optimized web content proxy server with the JavaScript emulator;

[00023] FIG. 7 is an exemplary listing JavaScript with a jQuery request; and

[00024] FIG. 8 is a flow chart of an example of a method for processing a received web page with a JavaScript instruction set at the optimized web content proxy server with a JavaScript emulator before extracting the JavaScript instruction set.

DETAILED DESCRIPTION

[00025] An exemplary environment 10 with a web content proxy server 12 with a JavaScript emulator is illustrated in FIG. 1. The exemplary environment 10 includes the web content proxy server or apparatus 12, client devices 14(1)-14(n), web server devices 16(1)-16(n), and communication networks 18(1)-18(2), although other numbers and types of systems, devices, and/or elements in other configurations and environments with other communication network topologies can be used. This technology provides a number of advantages including providing a method, computer readable medium and an apparatus that further optimizes a web content proxy server through the utilization of a JavaScript emulator.

[00026] Referring more specifically to FIG. 1, the web content proxy server 12 includes a central processing unit (CPU) or processor 13, a memory 15, and an interface system 17 which are coupled together by a bus 19 or other link, although other numbers and types of components, parts, devices, systems, and elements in other configurations and locations can be used. The processor 13 in the web content proxy server 12 executes a program of stored instructions one or

more aspects of the present invention as described and illustrated by way of the embodiments herein, although the processor could execute other numbers and types of programmed instructions.

[00027] The memory 15 in the web content proxy server 12 stores these programmed instructions for one or more aspects of the present invention as described and illustrated herein, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, DVD ROM, or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to the processor 13, can be used for the memory 15 in the web content proxy server 12. In these embodiments, the memory 15 includes a core module 21 and a JavaScript emulator module 23 with a memory cache 25 which store programmed instructions and other information for one or more aspects of the present invention as described and illustrated herein, although the memory can comprise other types and numbers of systems, devices, and elements in other configurations which store other data. The JavaScript emulator module 23 includes programmed instructions and/or logic configured to as described and illustrated herein including executing JavaScript instructions extracted from optimized web pages or HTTP requests, although the JavaScript emulator module 23 can have other types and numbers of functions as described and illustrated herein.

[00028] The interface system 17 in the web content proxy server 12 is used to operatively couple and communicate between the web content proxy server 12 and the client devices 14(1)-14(n) and the web server devices 16(1)-16(n) via the communication networks 18(1) and 18(2), although other types and numbers of communication networks with other types and numbers of connections and configurations can be used. By way of example only, the communication networks 18(1) and 18(2) can use TCP/IP over Ethernet and industry-standard protocols, including HTTP, HTTPS, WAP, and SOAP, although other types and numbers of communication networks, such as a direct connection, a local area

network, a wide area network, modems and phone lines, e-mail, and wireless and hardwire communication technology, each having their own communications protocols, can be used.

[00029] Each of the client devices 14(1)-14(n) enables a user to request, get and interact with web pages from one or more web sites hosted by the web server devices 16(1)-16(n) through the web content proxy server 12 via one or more communication networks, although one or more of the client devices 14(1)-14(n) could access content and utilize other types and numbers of applications from other sources and could provide a wide variety of other functions for the user. Although multiple client devices 14(1)-14(n) are shown, other numbers and types of user computing systems could be used. In this example, the client devices 14(1)-14(n) comprise mobile devices with Internet access that permit a website form page or other retrieved data to be displayed, although each of the client devices 14(1)-14(n). By way of example only, one or more of the client devices 14(1)-14(n) can comprise smart phones, personal digital assistants, or computers.

[00030] Each of client devices 14(1)-14(n) in this example is a computing device that includes a central processing unit (CPU) or processor 20, a memory 22, user input device 24, a display 26, and an interface system 28, and which are coupled together by a bus 30 or other link, although one or more of client devices 14(1)-14(n) can include other numbers and types of components, parts, devices, systems, and elements in other configurations. The processor 20 in each of client devices 14(1)-14(n) executes a program of stored instructions for one or more aspects of the present invention as described and illustrated herein, although the processor could execute other numbers and types of programmed instructions.

[00031] The memory 22 in each of the client devices 14(1)-14(n) stores these programmed instructions for one or more aspects of the present invention as described and illustrated herein, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, or

other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to processor 20 can be used for the memory 22 in each of the client devices 14(1)-14(n.

[00032] The user input device 24 in each of the client devices 14(1)-14(n) is used to input selections, such as requests for a particular website form page or to enter data in fields of a form page, although the user input device could be used to input other types of data and interact with other elements. The user input device can include keypads, touch screens, and/or vocal input processing systems although other types and numbers of user input devices can be used.

[00033] The display 26 in each of the client devices 14(1)-14(n) is used to show data and information to the user, such as website page by way of example only. The display in each of the client devices 14(1)-14(n) is a phone screen display, although other types and numbers of displays could be used depending on the particular type of client device.

[00034] The interface system 28 in each of the client devices 14(1)-14(n) is used to operatively couple and communicate between the client devices 14(1)-14(n) and the web content proxy server 12 and web server devices 16(1)-16(n) over the communication networks 18(1) and 18(2), although other types and numbers of communication networks with other types and numbers of connections and configurations can be used.

[00035] The web server devices 16(1)-16(n) provide one or more pages from one or more web sites for use by one or more of the client devices 14(1)-14(n) via the web content proxy server 12, although the web server devices 16(1)-16(n) can provide other numbers and types of applications and/or content and can have provide other numbers and types of functions. Although web server devices 16(1)-16(n) are shown for ease of illustration and discussion, other numbers and types of web server systems and devices can be used.

[00036] Each of the web server devices 16(1)-16(n) include a central processing unit (CPU) or processor, a memory, and an interface system which are

coupled together by a bus or other link, although each of the web server devices 16(1)-16(n) could have other numbers and types of components, parts, devices, systems, and elements in other configurations and locations can be used. The processor in each of the web server devices 16(1)-16(n) executes a program of stored instructions one or more aspects of the present invention as described and illustrated by way of the embodiments herein, although the processor could execute other numbers and types of programmed instructions.

[00037] The memory in each of the web server devices 16(1)-16(n) stores these programmed instructions for one or more aspects of the present invention as described and illustrated by way of the embodiments, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, DVD ROM, or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to the processor, can be used for the memory in each of the web server devices 16(1)-16(n).

[00038] The interface system in each of the web server devices 16(1)-16(n) is used to operatively couple and communicate between the web server devices 16(1)-16(n) and the web content proxy server 12 and the client devices 14(1)-14(n) via communication networks 18(1) and 18(2), although other types and numbers of communication networks with other types and numbers of connections and configurations can be used.

[00039] Although embodiments of the web content proxy server 12, the client devices 14(1)-14(n), and the web server devices 16(1)-16(n), are described and illustrated herein, each of the client devices 14(1)-14(n), the web content proxy server 12, and the web server devices 16(1)-16(n), can be implemented on any suitable computer system or computing device. It is to be understood that the devices and systems of the embodiments described herein are for exemplary purposes, as many variations of the specific hardware and software used to

implement the embodiments are possible, as will be appreciated by those skilled in the relevant art(s).

[00040] Furthermore, each of the systems of the embodiments may be conveniently implemented using one or more general purpose computer systems, microprocessors, digital signal processors, and micro-controllers, programmed according to the teachings of the embodiments, as described and illustrated herein, and as will be appreciated by those ordinary skill in the art.

[00041] In addition, two or more computing systems or devices can be substituted for any one of the systems in any embodiment of the embodiments. Accordingly, principles and advantages of distributed processing, such as redundancy and replication also can be implemented, as desired, to increase the robustness and performance of the devices and systems of the embodiments. The embodiments may also be implemented on computer system or systems that extend across any suitable network using any suitable interface mechanisms and communications technologies, including by way of example only telecommunications in any suitable form (e.g., voice and modem), wireless communications media, wireless communications networks, cellular communications networks, G3 communications networks, Public Switched Telephone Network (PSTNs), Packet Data Networks (PDNs), the Internet, intranets, and combinations thereof.

[00042] The embodiments may also be embodied as a computer readable medium having instructions stored thereon for one or more aspects of the present invention as described and illustrated by way of the embodiments herein, as described herein, which when executed by a processor, cause the processor to carry out the steps necessary to implement the methods of the embodiments, as described and illustrated herein.

[00043] An exemplary method for utilizing a JavaScript emulator in a web content proxy server 12 in an exemplary environment 10 will now be described with reference to FIGS. 1-8. Referring more specifically to FIGS. 1 and 5, in this

example in step 202, a form web page is received at the web content proxy server 12.

[00044] In step 204, the web content proxy server 12 determines whether the received web form page has any JavaScript code or instruction sets for execution during use of the web form page, such as for one or more forms or other fields in the received form web page. If in step 204 the web content proxy server 12 determines the received web form page does not have any JavaScript programmed instruction sets, then the No branch is taken to step 214 described below. If in step 204 the web content proxy server 12 determines the received web form page does has one or more JavaScript programmed instruction sets, then the Yes branch is taken to step 206.

[00045] In step 206, the web content proxy server 12 instantiates the received web form page with the JavaScript module 23 which is responsible for emulating the behavior of each of the identified JavaScript instruction sets. During the instantiation, the web content proxy server 12 creates a document object model of the web form page with each of the identified JavaScript instruction sets, although the other types of models could be created.

[00046] In this example, for reasons of security and performance four levels of detail can be used when the web content proxy server 12 creates the document object model, although other levels of detail can be used. In this particular example, the levels of detail for the document object model are: basic, attributes, all-but-scripts, and all, although other levels of detail in the document model object can be used. With basic, during the instantiation process when creating the document object model the JavaScript module 23 includes all elements in the web page removing their content, except for the elements of a form, such as checkboxes, radio boxes, combo boxes, buttons, and text fields by way of example. The JavaScript module 23 also includes all attributes of form controls along with only the id and class attributes of all elements. With attributes, during the instantiation process when creating the document object model the JavaScript module 23 includes all elements removing their content, except for the elements of a form, such as checkboxes, radio boxes, combo boxes,

buttons, and text fields by way of example. The JavaScript module 23 also includes all attributes of form controls. With all-but-scripts, during the instantiation process when creating the document object model the JavaScript module 23 includes all elements, including both content and attributes, however the JavaScript module 23 removes the content of the elements style and script. With all, during the instantiation process when creating the document object model the JavaScript module 23 includes all elements including both content and attributes.

[00047] Once the instantiation process is completed, in step 210 the JavaScript module 23 stores the created document object model for the web page with each of the JavaScript instruction sets in the cache memory 15 and assigns a JavaScript identifier to it, although each of the created document object models could be stored in other manners and in other locations. For security reasons the JavaScript module 23 executed by the web content proxy server 12 also provides the functionality to delete any values in fields before storing the created document object model. Accordingly, the web content proxy server 12 does not store sensitive data, such as credit card numbers, either temporary or permanently.

[00048] In step 212, the JavaScript module 23 executed by the web content proxy server 12 adds the assigned JavaScript field identifier for the created document object model in the optimized web form page at each location of an extracted JavaScript instruction set. In this particular example, the JavaScript identifier for the stored document object model is called un_jtt_jse and is stored as a hidden identifier in the optimized web page form provided to the requesting one of the client devices 14(1)-14(n), although other types and numbers of identifiers which are hidden or visible could be used.

[00049] In step 214, the JavaScript module 23 passes the optimized web form page to the core module 21 to be provided by the web content proxy server 12 for further use in the exemplary environment 10, such as in a web browser on one of the client devices 14(1)-14(n).

[00050] In another illustrative example, an exemplary method for processing a received request at the optimized web content proxy server 12 with the JavaScript emulator is illustrated with reference to FIGS. 1 and 6. In step 300, the JavaScript module 23 in web content proxy server 12 receives an HTTP request from the core module 21 that was received from one of the client devices 14(1)-14(n), although other types of requests, web pages, browser interactions or other types of JavaScript related actions could be received.

[00051] In step 302, the JavaScript module 23 in the web content proxy server 12 is executed to determine if the received HTTP request includes one or more hidden JavaScript identifiers or other marker. In this particular example, an un_jtt_jse JavaScript identifier is included in the received HTTP request. If in step 302, the received HTTP request does not include one or more hidden JavaScript identifiers, then the No branch is taken to step 310 as explained below. If in step 302, the received HTTP request does include one or more hidden JavaScript identifiers, then the Yes branch is taken to step 304.

[00052] In step 304, the web content proxy server 12 executes the JavaScript module 23 to retrieve from memory cache 25 the corresponding stored document object model for the identified JavaScript identifier. In this particular example, the document object model associated with the JavaScript identifier un_jtt_jse is retrieved.

[00053] In step 306, the web content proxy server 12 executes the JavaScript module 23 to validate all of the JavaScript instruction sets in the retrieved document object model.

[00054] In step 308, the web content proxy server 12 executes with the JavaScript module 23 each of the JavaScript instruction sets in the document object model with any corresponding values in the received HTTP request. By emulating the typical behavior of a web browser, the JavaScript module 23 executed by the web content proxy server 12 also assigns default values to all those properties whose values can neither be inferred from the received HTTP

request nor from the document object module. The web content proxy server 12 appends the output of this execution to the received HTTP request.

[00055] In step 310, the JavaScript module 23 is executed by the web content proxy server 12 to pass the core module 21 the response with the executed JavaScript functionality appended to the HTTP request. The core module 21 is executed by the web content proxy server 12 to determine where to provide the response with the executed JavaScript functionality in the exemplary environment 10, such as to a web browser on one of the client devices 14(1)-14(n). For example, the response to the received HTTP request might be an error message if incorrect or insufficient values were provided or might provide other information if all the needed values were provided.

[00056] By way of example now, consider the web page form represented in FIG. 2 that can be validated by the JavaScript instruction set or code shown in FIG. 3. The action of clicking on the login button 106 of the optimized form web page 100 triggers a new HTTP request that is sent to the web content proxy server 12 which executes the programmed instructions in the JavaScript module 23. In this example, the received request would includes an un_jtt_jse identifier or parameter so the 61 then the corresponding document object model is loaded from the memory cache 25. The JavaScript module 23 is executed by the web content proxy server 12 to use the form name or its index, i.e. position within the web page or document, to bind the parameters from the received HTTP request with the corresponding controls in the stored document object model representation of the previously extracted JavaScript instruction set during optimization.

[00057] At this point, the JavaScript module 23 is executed by the web content proxy server 12 to validate the received HTTP request with the corresponding controls in the stored document object model. Next, the JavaScript module 23 is executed by the web content proxy server 12 to append 64 the output of the validation process to the data included in the received HTTP request. The JavaScript module 23 also is executed by the web content proxy server 12 to retrieve values from JavaScript variables, properties of HTML elements, such as

the innerHTML property, input.value or document.location), or the strings sent to the window.alert() and the document.write() functions as necessary.

[00058] Once the HTTP request has been modified the JavaScript module 23 sends it to the core module 21. Based on the computed data, the core module 21 is executed by the web content proxy server 12 to determine whether to print an error message or send the data from the form to the original web content server.

[00059] In an another illustrative example, an exemplary method a method for processing a received web page at the optimized web content proxy server 12 with a JavaScript emulator for delivery to one of the client computing devices 10 is illustrated with reference to FIGS. 1, 7, and 8. In step 400, the JavaScript module 23 in web content proxy server 12 receives a web page which contains a JavaScript instruction set that creates dynamic content from one of the web server devices 16(1)-16(n), although other types of requests, web pages, browser interactions or other types of JavaScript related actions could be received and the content can be static or dynamic. By way of example, consider a web page with a JavaScript code or instruction set based on the jQuery library shown in FIG. 7. This is a JavaScript framework used to develop complex web applications based on Ajax technology. The event of receiving or loading this web page triggers the execution of the JavaScript code or instruction set shown in the example.

[00060] In step 402, the web content proxy server 12 instantiates this received web page which contains the JavaScript instruction set to create a document object model of the web page.

[00061] In step 404, the web content proxy server 12 adds the JavaScript programmed instruction set to retrieve dynamic content to the created document object model. In the example illustrated in FIG. 7, the web content proxy server 12 adds the JavaScript module 23 to retrieve the content of '/cart.html' web page and place it into the html element identified by the attribute id with value "cart".

[00062] In step 406, the web content proxy server 12 executes the JavaScript programmed instruction set in the document object model of the received web page to obtain the dynamic content. For example, when executing

the JavaScript programmed instruction set in the document object model, the JavaScript module 23 can dispatch Ajax calls to retrieve content from external sources, such as one of the web server devices 16(1)-16(n). In this example, the JavaScript programmed instruction set is programmed to retrieve dynamic content, although other types of dynamic or static data from other sources could be obtained.

[00063] In step 408, the web content proxy server 12 executes the JavaScript module 23 to append the obtained content in an optimized version of the received web page with the JavaScript programmed instruction set extracted.

[00064] In step 410, the JavaScript module 23 is executed by the web content proxy server 12 passes the optimized web page with the extracted JavaScript instruction set and appended dynamic content to the core module 21. The core module 21 is executed by the web content proxy server 12 to determine where to provide the web page with the executed JavaScript functionality in the exemplary environment 10, such as to a web browser on one of the client devices 14(1)-14(n), although other manners for passing the optimized web page with the extracted JavaScript instruction set and appended dynamic content could be used.

[00065] Accordingly, as illustrated and described herein this technology provides a number of advantages including providing a method, computer readable medium and an apparatus that further optimizes a web content proxy server through the utilization of a JavaScript emulator. The processes illustrated above are only exemplary and all JavaScript functionality can be emulated by the web content proxy server 12 with this technology. With the JavaScript emulator, the web content proxy server can continue to apply optimizing transformation rules, while still enabling JavaScript functions in optimized HTTP requests, web pages, interactions with browsers, and other JavaScript events by to be executed with the JavaScript emulator. The JavaScript emulator emulates the behavior of all JavaScript objects that are supported by current web browsers and are used by web developers to access and change the components of web pages.

[00066] Having thus described the basic concept of the invention, it will be rather apparent to those skilled in the art that the foregoing detailed disclosure is intended to be presented by way of example only, and is not limiting. Various alterations, improvements, and modifications will occur and are intended to those skilled in the art, though not expressly stated herein. These alterations, improvements, and modifications are intended to be suggested hereby, and are within the spirit and scope of the invention. Additionally, the recited order of processing elements or sequences, or the use of numbers, letters, or other designations therefore, is not intended to limit the claimed processes to any order except as may be specified in the claims. Accordingly, the invention is limited only by the following claims and equivalents thereto.

CLAIMS

What is claimed is:

1. A method for utilizing a JavaScript emulator in a proxy server, the method comprising:

creating and storing with a proxy server an object model of a web page which has one or more JavaScript instruction sets;

extracting with the proxy server at least one of the one or more JavaScript instruction sets from the web page and inserting a JavaScript field identifier into the web page to optimize the web page; and

providing with the proxy server the optimized web page with the inserted JavaScript field identifier.

2. The method as set forth in claim 1 wherein the creating and storing further comprises creating and storing with the proxy server the object model of the web page including all attributes and content of controls for one or more forms in the web page and all remaining elements of the web page with all content removed.

3. The method as set forth in claim 2 wherein the creating and storing further comprises creating and storing with the proxy server the object model of the web page with all attributes of identification and class for all the elements of the web page.

4. The method as set forth in claim 2 wherein the creating and storing with the proxy server the object model of the web page further comprises creating and storing the object model of the web page with all attributes of all the elements in the web page.

5. The method as set forth in claim 1 wherein the creating and storing further comprises creating and storing with the proxy server the object model of the web page with all elements and content retained and having elements of style and script removed.

6. The method as set forth in claim 1 wherein the extracting further comprises extracting all of the one or more JavaScript instruction sets.

7. The method as set forth in claim 1 further comprising deleting with the proxy server any values in form controls in the created object model.

8. A computer readable medium having stored thereon instructions for utilizing a JavaScript emulator comprising machine executable code which when executed by at least one processor, causes the processor to perform steps:

creating and storing an object model of a web page which has one or more JavaScript instruction sets;

extracting at least one of the one or more JavaScript instruction sets from the web page and inserting a JavaScript field identifier into the web page to optimize the web page; and

providing the optimized web page with the inserted JavaScript field identifier.

9. The medium as set forth in claim 8 wherein the creating and storing further comprises creating and storing the object model of the web page including all attributes and content of controls for one or more forms in the web page and all remaining elements of the web page with all content removed.

10. The medium as set forth in claim 9 wherein the creating and storing further comprises creating and storing the object model of the web page with all attributes of identification and class for all the elements of the web page.

11. The medium as set forth in claim 9 wherein the creating and storing the object model of the web page further comprises creating and storing the object model of the web page with all attributes of all the elements in the web page.

12. The medium as set forth in claim 8 wherein the creating and storing further comprises creating and storing the object model of the web page with all elements and content retained and having elements of style and script removed.

13. The medium as set forth in claim 8 wherein the extracting further comprises extracting all of the one or more JavaScript instruction sets.

14. The medium as set forth in claim 8 further comprising deleting any values in form controls in the created object model.

15. A web proxy apparatus comprising:

one or more processors;

a memory coupled to the one or more processors which are configured to execute programmed instructions stored in the memory comprising:

creating and storing an object model of a web page which has one or more JavaScript instruction sets;

extracting at least one of the one or more JavaScript instruction sets from the web page and inserting a JavaScript field identifier into the web page to optimize the web page; and

providing the optimized web page with the inserted JavaScript field identifier.

16. The apparatus as set forth in claim 15 wherein the one or more processors is further configured to execute programmed instructions stored in the memory for the creating and storing further comprises creating and storing the object model of the web page including all attributes and content of controls for one or more forms in the web page and all remaining elements of the web page with all content removed.

17. The apparatus as set forth in claim 16 wherein the one or more processors is further configured to execute programmed instructions stored

in the memory for the creating and storing further comprises creating and storing the object model of the web page with all attributes of identification and class for all the elements of the web page.

18. The apparatus as set forth in claim 16 wherein the one or more processors is further configured to execute programmed instructions stored in the memory for the creating and storing further comprises creating and storing the object model of the web page with all attributes of all the elements in the web page.

19. The apparatus as set forth in claim 15 wherein the one or more processors is further configured to execute programmed instructions stored in the memory for the creating and storing further comprises creating and storing the object model of the web page with all elements and content retained and having elements of style and script removed.

20. The apparatus as set forth in claim 15 wherein the one or more processors is further configured to execute programmed instructions stored in the memory for the extracting further comprises extracting all of the one or more JavaScript instruction sets.

21. The apparatus as set forth in claim 15 wherein the one or more processors is further configured to execute programmed instructions stored in the memory further comprising deleting any values in form controls in the created object model.

22. A method for utilizing a JavaScript emulator in a proxy server, the method comprising:

loading with the proxy server a stored object model of an original web page which had at least a portion of one or more JavaScript instruction sets previously extracted to form an optimized web page;

appending with the proxy server any data in a request received at the proxy server which corresponds to the loaded stored object model

of the original web page into the loaded stored object model of the original web page;

processing with the proxy server the loaded stored object model of the original web page with any of the appended data; and

providing with the proxy server the processed loaded stored object model of the original web page.

23. The method as set forth in claim 22 further comprising determining with the proxy server whether the request corresponds to the stored object model of an original web page, wherein the loading and the appending are executed when the determining determines the request corresponds to the stored object model of an original web page.

24. The method as set forth in claim 22 wherein the processing further comprises validating with the proxy server the loaded stored object model of the original web page with any of the appended data.

25. The method as set forth in claim 22 wherein the providing further comprises providing one or more functionalities of one or more resulting properties from the processing of the loaded stored object model of the original web page with any of the appended data to a determined destination.

26. A computer readable medium having stored thereon instructions for utilizing a JavaScript emulator comprising machine executable code which when executed by at least one processor, causes the processor to perform steps:

loading a stored object model of an original web page which had at least a portion of one or more JavaScript instruction sets previously extracted to form an optimized web page;

appending any data in a request received at the proxy server which corresponds to the loaded stored object model of the original web page into the loaded stored object model of the original web page;

processing the loaded stored object model of the original web page with any of the appended data; and

providing the processed loaded stored object model of the original web page.

27. The medium as set forth in claim 26 further comprising determining whether the request corresponds to the stored object model of an original web page, wherein the loading and the appending are executed when the determining determines the request corresponds to the stored object model of an original web page.

28. The medium as set forth in claim 26 wherein the processing further comprises validating the loaded stored object model of the original web page with any of the appended data.

29. The medium as set forth in claim 26 wherein the providing further comprises providing one or more functionalities of one or more resulting properties from the processing of the loaded stored object model of the original web page with any of the appended data to a determined destination.

30. A web proxy apparatus comprising:

one or more processors;

a memory coupled to the one or more processors which are configured to execute programmed instructions stored in the memory comprising:

loading a stored object model of an original web page which had at least a portion of one or more JavaScript instruction sets previously extracted to form an optimized web page;

appending any data in a request received at the proxy server which corresponds to the loaded stored object model of the original web page into the loaded stored object model of the original web page;

processing the loaded stored object model of the original web page with any of the appended data; and

providing the processed loaded stored object model of the original web page.

31. The apparatus as set forth in claim 30 wherein the one or more processors is further configured to execute programmed instructions stored in the memory further comprising determining whether the request corresponds to the stored object model of an original web page, wherein the loading and the appending are executed when the determining determines the request corresponds to the stored object model of an original web page.

32. The apparatus as set forth in claim 30 wherein the one or more processors is further configured to execute programmed instructions stored in the memory for the processing further comprising validating the loaded stored object model of the original web page with any of the appended data.

33. The apparatus as set forth in claim 30 wherein the one or more processors is further configured to execute programmed instructions stored in the memory for the providing further comprising providing one or more functionalities of one or more resulting properties from the processing of the loaded stored object model of the original web page with any of the appended data to a determined destination.

34. A method for utilizing a JavaScript emulator in a proxy server, the method comprising:

creating with a proxy server an object model of a web page which has at least one JavaScript instruction set;

executing with the proxy server the at least one JavaScript instruction set from the web page to obtain content;

removing with the proxy server the at least one JavaScript instruction set from the web page and appending the obtained content to generate an optimized web page with the obtained content; and

providing with the proxy server the optimized web page with the obtained content.

35. A computer readable medium having stored thereon instructions for utilizing a JavaScript emulator comprising machine executable code which when executed by at least one processor, causes the processor to perform steps:

creating an object model of a web page which has at least one JavaScript instruction set;

executing the at least one JavaScript instruction set from the web page to obtain content;

removing the at least one JavaScript instruction set from the web page and appending the obtained content to generate an optimized web page with the obtained content; and

providing the optimized web page with the obtained content.

36. A web proxy apparatus comprising:

one or more processors;

a memory coupled to the one or more processors which are configured to execute programmed instructions stored in the memory comprising:

creating an object model of a web page which has at least one JavaScript instruction set;

executing the at least one JavaScript instruction set from the web page to obtain content;

removing the at least one JavaScript instruction set from the web page and appending the obtained content to generate an optimized web page with the obtained content; and

providing the optimized web page with the obtained content.

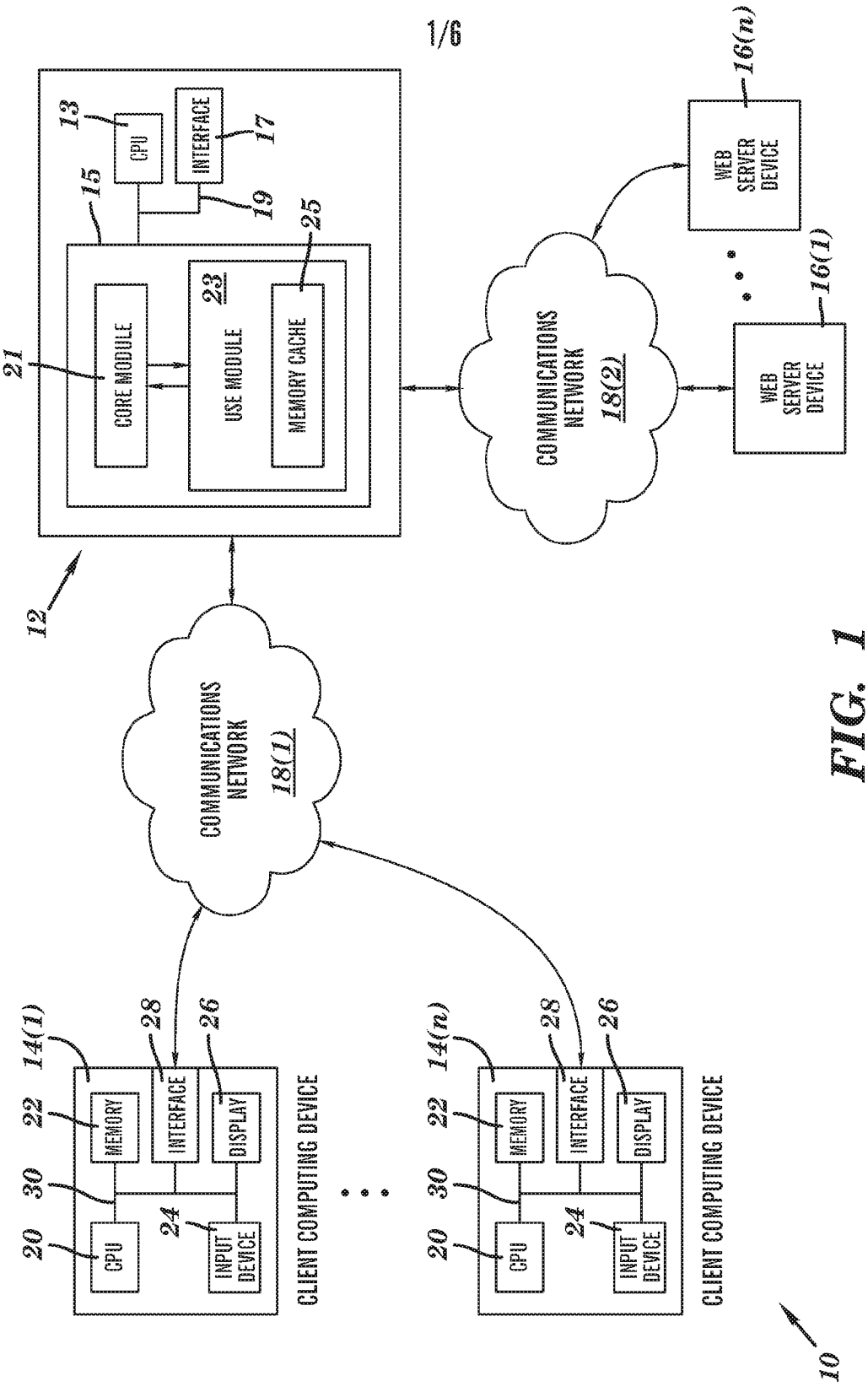

10
12
13
CPU
INTERFACE
17
15
19
25
21
CORE MODULE
23
USE MODULE
MEMORY CACHE
COMMUNICATIONS NETWORK 18(1)
COMMUNICATIONS NETWORK 18(2)
16(n)
WEB SERVER DEVICE
16(1)
WEB SERVER DEVICE
14(1)
28
26
22
MEMORY
INTERFACE
DISPLAY
30
20
CPU
24
INPUT DEVICE
CLIENT COMPUTING DEVICE
14(n)
28
26
22
MEMORY
INTERFACE
DISPLAY
30
20
CPU
24
INPUT DEVICE
CLIENT COMPUTING DEVICE

FIG. 1

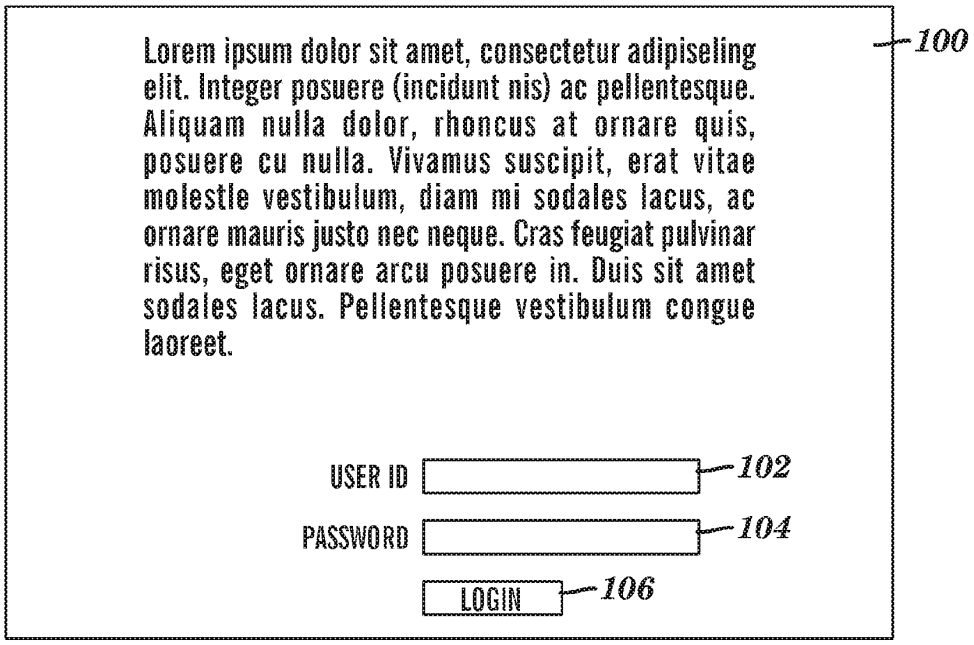


FIG. 2

```
function validateLogin() {
        var f = document.LoginForm;
        var error = document.getElementById('error');
        if (f.userId.value == '') {
                error.innerHTML = "USER ID cannot be empty";
                return false;
        }
        if (f.password.value == '') {
                error.innerHTML = "PASSWORD cannot be empty";
                return false;
        }
        return true;
}
```

FIG. 3

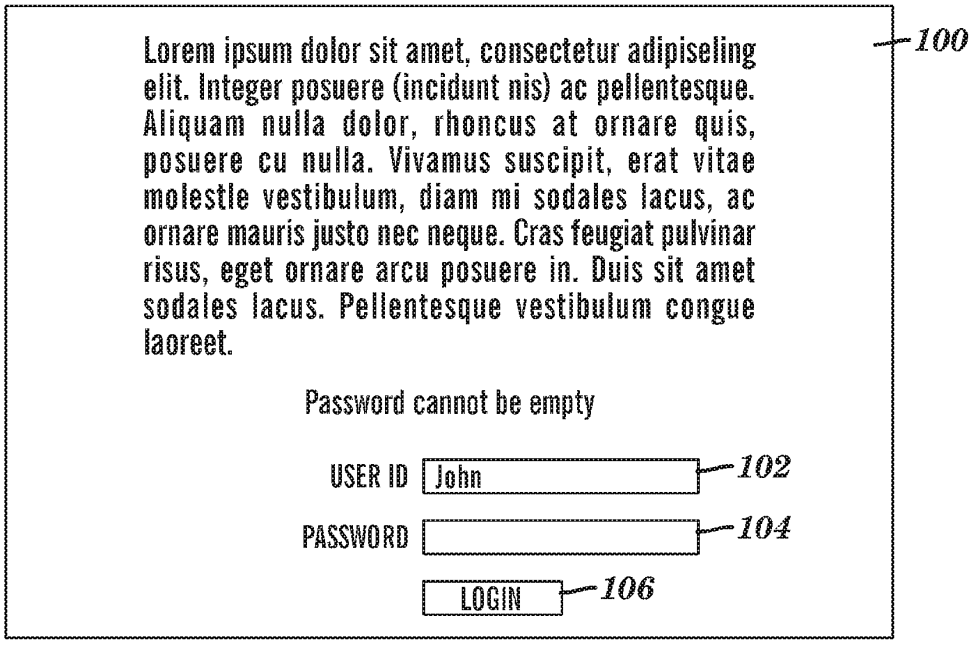

Lorem ipsum dolor sit amet, consectetur adipiseling elit. Integer posuere (incidunt nis) ac pellentesque. Aliquam nulla dolor, rhoncus at ornare quis, posuere cu nulla. Vivamus suscipit, erat vitae molestie vestibulum, diam mi sodales lacus, ac ornare mauris justo nec neque. Cras feugiat pulvinar risus, eget ornare arcu posuere in. Duis sit amet sodales lacus. Pellentesque vestibulum congue laoreet.
Password cannot be empty
USER ID
John
PASSWORD
LOGIN
100
102
104
106

FIG. 4

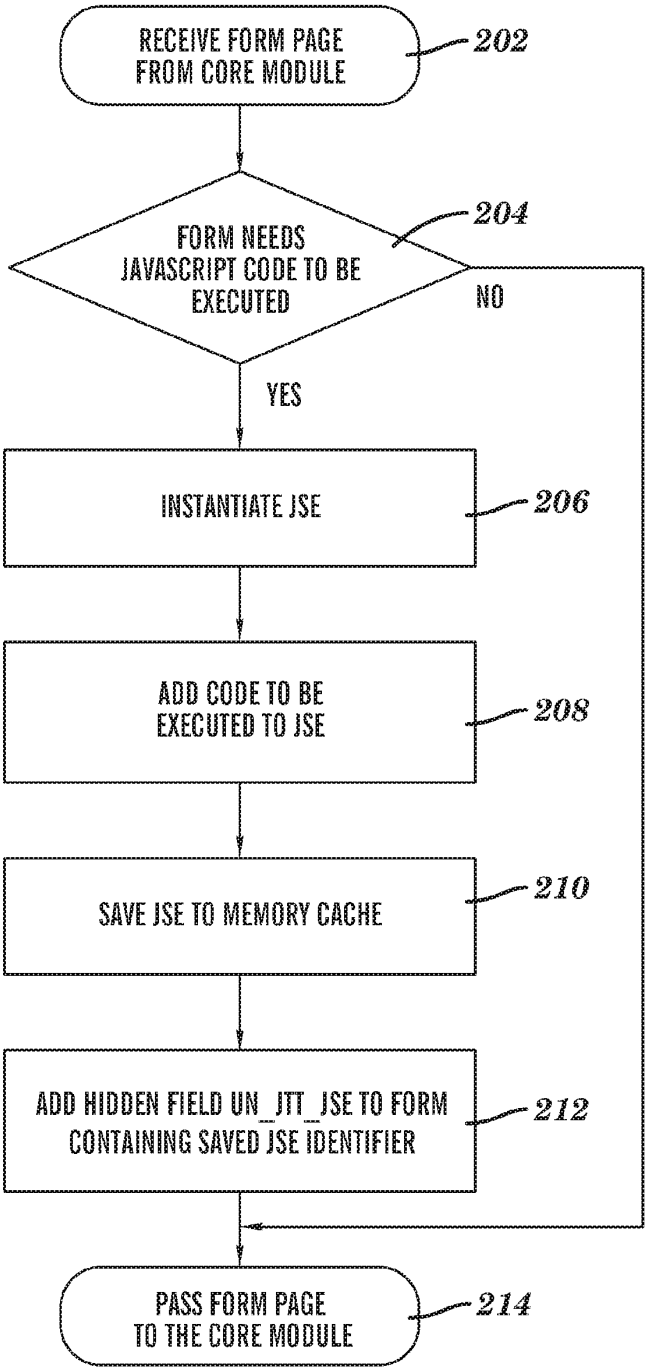

RECEIVE FORM PAGE FROM CORE MODULE
202
FORM NEEDS JAVASCRIPT CODE TO BE EXECUTED
204
NO
YES
INSTANTIATE JSE
206
ADD CODE TO BE EXECUTED TO JSE
208
SAVE JSE TO MEMORY CACHE
210
ADD HIDDEN FIELD UN_JTT_JSE TO FORM CONTAINING SAVED JSE IDENTIFIER
212
PASS FORM PAGE TO THE CORE MODULE
214

FIG. 5

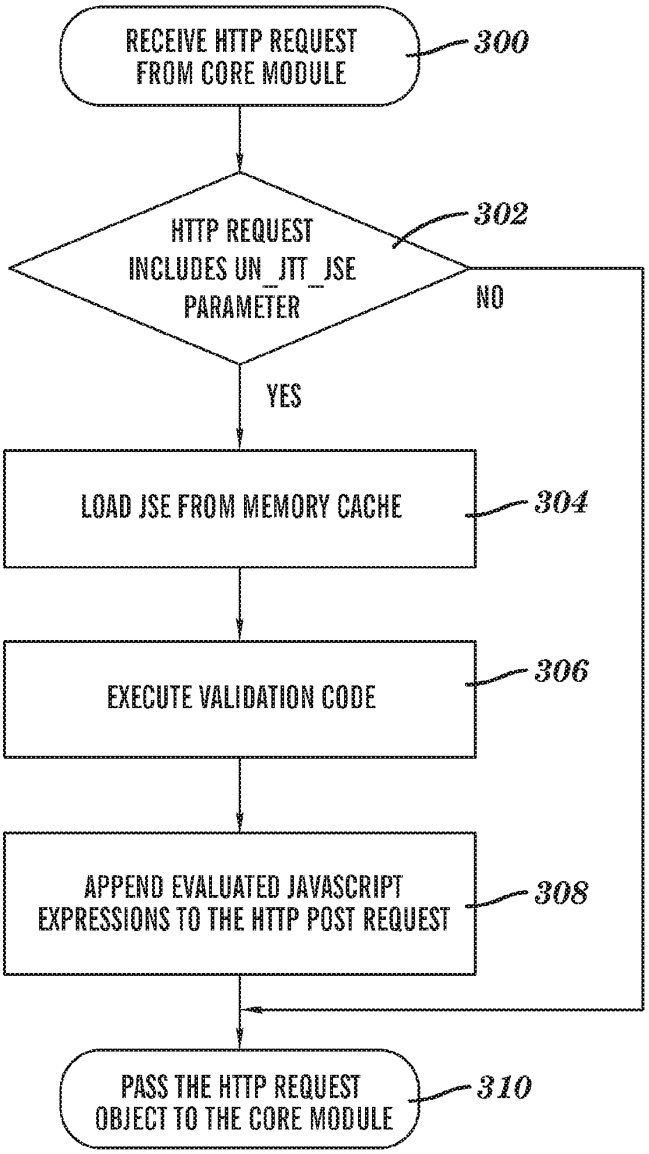


FIG. 6

```
|$(document).ready(function()  {
        $.get( '/cart.html', function(data) {
                $('#cart').html(data);}
        );
});
```

FIG. 7

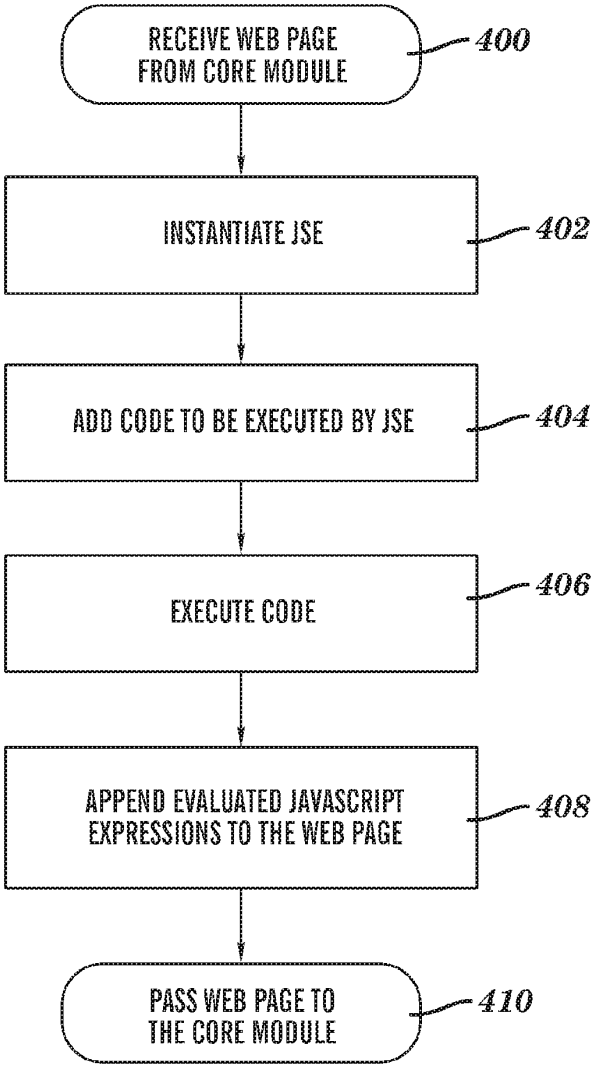

RECEIVE WEB PAGE FROM CORE MODULE
400
INSTANTIATE JSE
402
ADD CODE TO BE EXECUTED BY JSE
404
EXECUTE CODE
406
APPEND EVALUATED JAVASCRIPT EXPRESSIONS TO THE WEB PAGE
408
PASS WEB PAGE TO THE CORE MODULE
410

FIG. 8

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Veröffentlichungsnummer:
(11) Publication number: EP 2 580 699 A0
(11) Numéro de publication:

Internationale Anmeldung veröffentlicht durch die Weltorganisation für geistiges Eigentum unter der Nummer:

WO 2011/156739 (Art. 153(3) EPÜ).

International application published by the World Intellectual Property Organization under number:

WO 2011/156739 (Art. 153(3) EPC).

Demande internationale publiée par l'Organisation Mondiale de la Propriété Intellectuelle sous le numéro:

WO 2011/156739 (art. 153(3) CBE).

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
15 December 2011 (15.12.2011)

PCT

(10) International Publication Number
WO 2011/156739 A2

(51) International Patent Classification:
G06F 19/00 (2011.01) *G06Q 50/00* (2006.01)
G06Q 10/00 (2006.01)

(21) International Application Number:
PCT/US2011/040021

(22) International Filing Date:
10 June 2011 (10.06.2011)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
12/802,690 11 June 2010 (11.06.2010) US

(71) Applicant *(for all designated States except US)*: **US-ABLENET INC.** [US/US]; 28 W. 23rd Street, 6th Floor, New York, NY 10010 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **SCODA, Enrico** [IT/IT]; Via Cividina 416/3, I-33035 Matignacco (UD) (IT). **PEZZANO, Simone** [IT/IT]; Via Superiore, 21, I-33100 Udine (IT).

(74) Agents: **LEINBERG, Gunnar, G.** et al.; LECLAIR-RYAN, 290 Linden Oaks, Rochester, NY 14625 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

(54) Title: SOFTWARE PROJECT MANAGEMENT APPARATUSES AND METHODS THEREOF

(57) Abstract: A method, computer readable medium and apparatus that manages a software project includes assigning one of one or more virtual hosts in one of one or more workspaces in a development computing device to a remote computing device. The development computing device generates at least one link in the one of the one or more workspaces to at least one of one or more working copies of projects in one of one or more work benches in the one of the one or more virtual hosts. The development computing device generates at least one other link in the one of the one or more workspaces to the linked one of the one or more working copies of projects activated in a running area of the development computing device. The development computing device provides access to the activated one of the one or more working copies of projects to the remote computing device to execute one or more tasks.

WO 2011/156739 A2

SOFTWARE PROJECT MANAGEMENT APPARTUSES AND METHODS THEREOF

FIELD

[0001] This invention relates to software project management apparatuses and methods and, more particularly, to software project management apparatuses that manage projects which involve execution of tasks at remote workbenches, and methods thereof.

BACKGROUND

[0002] Currently, when managing the development of a software project which utilizes an interpreted computer language, such as HTML XML, JavaScript, JSP, and Python, a working copy of the program will be maintained on a production server. A project manager or managers of the software project will assign projects related to the working copy to developers located at remote working computing stations.

[0003] The developers at each remote working computing station will obtain a remote copy of the program and then develop and test code for the particular assigned project. Typically, each of these remote working computing stations will include an instance of an interpreter installed for execution and testing of the remote copy. When the remote copy is completed by the developer and needs to be published, the remote copy is transferred from the remote working computing station over to the working copy of the program on the production server. At that point, the developer or the project manager verifies the status of the particular assigned project on the production server.

[0004] Unfortunately, when multiple developers are working on different remote copies of the program at the remote working computing stations, there are possible conflicts in concurrency and loss of synchronization among remote copies and working copies. Additionally, each of the remote working computing stations requires a separate instance of the interpreter which can be expensive and also difficult to update and upgrade.

SUMMARY

[0005] A method for managing a software project includes assigning one of one or more virtual hosts in one of one or more workspaces in a development computing device to a remote computing device. The development computing device generates at least one link in the one of the one or more workspaces to at least one of one or more working copies of projects in one of one or more work benches in the one of the one or more virtual hosts. The development computing device generates at least one other link in the one of the one or more workspaces to the linked one of the one or more working copies of projects activated in a running area of the development computing device. The development computing device provides access to the activated one of the one or more working copies of projects to the remote computing device to execute one or more tasks.

[0006] A computer readable medium having stored thereon instructions for managing a software project comprising machine executable code which when executed by at least one processor, causes the processor to perform steps including assigning one of one or more virtual hosts in one of one or more workspaces in a development computing device to a remote computing device. At least one link is generated in the one of the one or more workspaces to at least one of one or more working copies of projects in one of one or more work benches in the one of the one or more virtual hosts. At least one other link is generated in the one of the one or more workspaces to the linked one of the one or more working copies of projects activated in a running area of the development computing device. Access is provided to the activated one of the one or more working copies of projects to the remote computing device to execute one or more tasks.

[0007] A software development management apparatus has one or more processors and a memory coupled to the one or more processors which are configured to execute programmed instructions stored in the memory including assigning one of one or more virtual hosts in one of one or more workspaces in a development computing device to a remote computing device. At least one link is generated in the one of the one or more workspaces to at least one of one or more working copies of projects in one of one or more work benches in the one of the

one or more virtual hosts. At least one other link is generated in the one of the one or more workspaces to the linked one of the one or more working copies of projects activated in a running area of the development computing device. Access is provided to the activated one of the one or more working copies of projects to the remote computing device to execute one or more tasks.

[0008] This technology provides a number of advantages including providing a more efficient and effective method for managing projects which involve execution of tasks at remote workbenches. This technology enables multiple software developers to accomplish tasks, such as developing, verifying, and testing, on remote computing workbenches and managers to supervise as if they were all located in one location. As a result, the developers at the remote computing workbenches and managers can be located anywhere. Additionally, this technology allows interpreted computer languages source files to be executed on a development server, eliminating the need to install an instance of the interpreter on each remote computing workbench. This helps to reduce costs and also simplifies updating and upgrading of the interpreter. Further, this technology enables managers to gain an easier and more efficient way to manage and control a software development project through its lifecycle.

BRIEF DESCRIPTIONS OF THE DRAWINGS

[0009] FIG. 1 is a partial block and partial functional diagram of an exemplary system environment with a development server;

[0010] FIG. 2 is a functional block diagram of workspaces in the development server shown in FIG. 1;

[0011] FIG. 3 is a functional block diagram of one of a workspace and a corresponding workbench in the development server shown in FIG. 1;

[0012] FIG. 4 is a flow chart of a method for creating a virtual host in the development server;

[0013] FIG. 5 is a flow chart of a method for activating a project in a virtual host in the development server;

[0014] FIG. 6 is a flow chart of a method for managing execution of an activation of a project in a virtual host in the development server;

[0015] FIG. 7 is a flow chart of a method for editing a project in a user workbench in the development server; and

[0016] FIG. 8 is a flow chart of a method for triggering an interpretation of a project in the development server.

DETAILED DESCRIPTION

[0017] An exemplary environment 10 with a development server 12 is illustrated in FIG. 1. The exemplary environment 10 includes the development server 12, a remote workbench computing device 14, a manager computing device 16, and a domain name server 18 are coupled together by one or more communication networks, although other numbers and types of systems, devices, and/or elements in other configurations and environments with other communication network topologies can be used. This technology provides a number of advantages including providing a more efficient and effective method for managing projects which involve execution of tasks at a remote workbench computing devices.

[0018] Referring more specifically to FIG. 1, the development server 12 includes a central processing unit (CPU) or processor, a memory comprising a control version system repository and an interpreter, and an interface system which are coupled together by a bus or other link, although other numbers and types of components, parts, devices, systems, and elements in other configurations and locations can be used as well as other types of computing devices can be used. The processor in the development server 12 executes a program of stored instructions one or more aspects of the present invention as described and illustrated by way of the embodiments herein, although the processor could execute other numbers and types of programmed instructions.

[0019] The memory in the development server 12 stores these programmed instructions for one or more aspects of the present invention as

described and illustrated herein, including the execution of the methods described herein, although some or all of the programmed instructions as well as other data could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, DVD ROM, or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to the processor, can be used for the memory in the development server 12. In this example, the control version system repository and the module with programmed instructions for the interpreter are located in the memory of the development server, although each could be in other locations and have other numbers.

[0020] The interface system in the development server 12 is used to operatively couple and communicate between the development server 12 and the remote workbench computing device 14, the manager computing device 16, and the domain name server 18, although other types and numbers of communication networks with other types and numbers of connections and configurations can be used. By way of example only, one or more communication networks can use TCP/IP over Ethernet and industry-standard protocols, including HTTP, HTTPS, WAP, and SOAP, although other types and numbers of communication networks, such as a direct connection, a local area network, a wide area network, modems and phone lines, e-mail, and wireless and hardwire communication technology, each having their own communications protocols, can be used.

[0021] The remote workbench computing device 14 in this example is a computing device that includes a central processing unit (CPU) or processor 20, a memory 22, user input device 24, a display 26, and an interface system 28, and which are coupled together by a bus 30 or other link, although one or more of remote workbench computing device 14 can include other numbers and types of components, parts, devices, systems, and elements in other configurations. The processor 20 in the remote workbench computing device 14 executes a program of stored instructions for one or more aspects of the present invention as described and illustrated herein, although the processor could execute other numbers and types of programmed instructions. Although one remote workbench computing

device 14 is illustrated in this example, other numbers and types of computing devices could be used.

[0022] The memory 22 in the remote workbench computing device 14 stores these programmed instructions for one or more aspects of the present invention as described and illustrated herein, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to processor 20 can be used for the memory 22 in the remote workbench computing device 14.

[0023] The user input device 24 in the remote workbench computing device 14 is used to input selections, although the user input device could be used to input other types of data and interact with other elements. The user input device can include keypads, touch screens, and/or vocal input processing systems although other types and numbers of user input devices can be used.

[0024] The display 26 in the remote workbench computing device 14 is used to show data and information to the user. The display in the remote workbench computing device 14 is a computer screen display, although other types and numbers of displays could be used.

[0025] The interface system 28 in the remote workbench computing device 14 is used to operatively couple and communicate between the remote workbench computing device 14 and the development server 12, the manager computing device 16, and the domain name server 18 over one or more communication networks, although other types and numbers of communication networks with other types and numbers of connections and configurations can be used.

[0026] The manager computing device 16 in this example is a computing device that includes a central processing unit (CPU) or processor 32, a memory

34, user input device 36, a display 38, and an interface system 40, and which are coupled together by a bus 42 or other link, although the manager computing device can include other numbers and types of components, parts, devices, systems, and elements in other configurations. The processor 32 in the manager computing device 16 executes a program of stored instructions for one or more aspects of the present invention as described and illustrated herein, although the processor could execute other numbers and types of programmed instructions. Although one the manager computing device 16 is illustrated in this example, other numbers and types of computing devices could be used

[0027] The memory 34 in the manager computing device 16 stores these programmed instructions for one or more aspects of the present invention as described and illustrated herein, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to processor 32 can be used for the memory 34 in the manager computing device 16.

[0028] The user input device 36 in the manager computing device 16 is used to input selections, although the user input device could be used to input other types of data and interact with other elements. The user input device 36 can include keypads, touch screens, and/or vocal input processing systems although other types and numbers of user input devices can be used.

[0029] The display 38 in the manager computing device 16 is used to show data and information to the user. The display in the manager computing device 16 is a computer screen display, although other types and numbers of displays could be used.

[0030] The interface system 40 in the manager computing device 16 is used to operatively couple and communicate between the manager computing device 16 and the developmental management server 12, the remote workbench

computing device 14, and the domain name server 18 over one or more communication networks, although other types and numbers of communication networks with other types and numbers of connections and configurations can be used.

[0031] The domain name server 18 includes a central processing unit (CPU) or processor, a memory, and an interface system which are coupled together by a bus or other link, although other numbers and types of components, parts, devices, systems, and elements in other configurations and locations can be used as well as other types of computing devices can be used. The processor in the domain name server 18 executes a program of stored instructions one or more aspects of the present invention as described and illustrated by way of the embodiments herein including assigning domain name addresses, although the processor could execute other numbers and types of programmed instructions.

[0032] The memory in the domain name server 18 stores these programmed instructions for one or more aspects of the present invention as described and illustrated herein, including the execution of the methods described herein, although some or all of the programmed instructions as well as other data could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, DVD ROM, or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to the processor, can be used for the memory in the domain name server 18. In this example, the control version system repository and the module with programmed instructions for the interpreter are located in the memory of the development server, although each could be in other locations and have other numbers.

[0033] The interface system in the domain name server 18 is used to operatively couple and communicate between the domain name server 18 and the development server 12, the remote workbench computing device 14, and the manager computing device 16, although other types and numbers of

communication networks with other types and numbers of connections and configurations can be used.

[0034] Although embodiments of the development server 12, the remote workbench computing device 14, manager computing device 16, and domain name server are described and illustrated herein, each can be implemented on any suitable computer system or computing device. It is to be understood that the devices and systems of the embodiments described herein are for exemplary purposes, as many variations of the specific hardware and software used to implement the embodiments are possible, as will be appreciated by those skilled in the relevant art(s). Additionally, other numbers of the development server 12, the remote workbench computing device 14, manager computing device 16, and domain name server 18 could be used.

[0035] Furthermore, each of the systems of the embodiments may be conveniently implemented using one or more general purpose computer systems, microprocessors, digital signal processors, and micro-controllers, programmed according to the teachings of the embodiments, as described and illustrated herein, and as will be appreciated by those ordinary skill in the art.

[0036] In addition, two or more computing systems or devices can be substituted for any one of the systems in any embodiment of the embodiments. Accordingly, principles and advantages of distributed processing, such as redundancy and replication also can be implemented, as desired, to increase the robustness and performance of the devices and systems of the embodiments. The embodiments may also be implemented on computer system or systems that extend across any suitable network using any suitable interface mechanisms and communications technologies, including by way of example only telecommunications in any suitable form (e.g., voice and modem), wireless communications media, wireless communications networks, cellular communications networks, G3 communications networks, Public Switched Telephone Network (PSTNs), Packet Data Networks (PDNs), the Internet, intranets, and combinations thereof.

[0037] The embodiments may also be embodied as a computer readable medium having instructions stored thereon for one or more aspects of the present invention as described and illustrated by way of the embodiments herein, as described herein, which when executed by a processor, cause the processor to carry out the steps necessary to implement the methods of the embodiments, as described and illustrated herein.

[0038] An exemplary method for creating a virtual host in the development server 12 will now be described with reference to FIGS. 1-4. In step 50, a developer, also referred to as user01 or user1 in this example, at the remote workbench computing device 14 submits a request to create a new virtualhost01 for user01 to the development server 12. In step 52, the development server 12 processes the received request and determines which one of one or more control version system repository is appropriate for one or more projects assigned to the developer (user01 in this example) at the remote workstation computing device 14. In step 54, the development server 12 fetches the one of the control version system repositories determined to be appropriate for the one or more projects assigned to the developer (user01 in this example). In step 56, the development server 12 creates the new virtualhost01 in the workspace for user01 at the remote workstation computing device 14 to remotely access as illustrated in FIGS. 2-3. In step 58, the development server 12 creates the new virtualhost01 in the workbench for the developer (user01 in this example) which is at the development server 12 as illustrated in FIG. 3. In step 60, the developement server 12 runs a checkout so that the content for the one or more projects assigned to the developer (user01 in this example) are downloaded in the newly created virtualhost01 in the workspace for the developer (user01 in this example) at the remote workstation computing device 14 to acess via the internet connection in this example.

[0039] An exemplary method for activating a project in a virtual host in the development server will now be described with reference to FIGS. 1-3 and 5. In step 62, a manager at the manager computing device 16 requests to active project1 in the virtualhost01 for the developer (user01 in this example) in the development server 12 for user01 at the remote workbench computing device 14.

[0040] In step 64, the development server 12 determines whether the virtualhost01 for the developer (user01 in this example) exists. If in step 64 the development server 12 determines the virtualhost01 for the developer (user01 in this example) does not exist, then the No branch is taken to step 66. In step 66, the development server 12 creates the virtualhost01 for user01 using the method described with reference to FIG. 4. If in step 64 the development server 12 determines the virtualhost01 for the developer (user01 in this example) does exist, then the Yes branch is taken to step 68. In step 68, the development server 12 runs a control version system update for one or more of the projects assigned to the virtualhost01 on the workspace for the developer (user01 in this example).

[0041] In step 70, the development server 12 determines whether the project1 the manager at the manager computing device 16 has requested to active exists in the virtualhost01 on the workspace for the developer (user01 in this example), although one or more of the projects can be activated in other manners and from other sources. If in step 70 the development server 12 determines the project1 the manager at the manager computing device 16 has requested to active does not exist in the virtualhost01 on the workspace for the developer (user01 in this example), then the No branch is taken to step 72 resulting in an error message, although other manners for indicating an error can be used. If in step 70 the development server 12 determines the project1 the manager at the manager computing device 16 has requested to active does exist in the virtualhost01 on the workspace for the developer (user01 in this example), then the Yes branch is taken to step 74 where the execution of the activation of a project1 is carried out as explained in greater detail with reference to FIG. 6.

[0042] An exemplary method for managing execution of an activation of a project in a virtual host in the development server 12 will now be described with reference to FIGS. 1-3 and 6. In step 76, the development server 12 begins the execution of the activation of a project1 in the virtualhost01 in the development server 12. The developer at the remote workstation computing device 14 accesses the activated project1 in the virtualhost01 in the development server 12 remotely via an internet connection, although other manners for accessing the project can be used.

[0043] In step 78 the development server 12 determines whether the project1 exists in the workspace for user01 for the developer at the remote workstation computing device 14 to execute one or more tasks. If in step 78 the development server 12 determines the project1 does not exist in the workspace for the developer (user01 in this example), then the No branch is taken to step 80 resulting in an error message sent to the remote workstation computing device 14, although other manners for indicating an error can be used. If in step 78 the development server 12 determines the project1 does exist in the workspace for the developer (user01 in this example), then the Yes branch is taken to step 82.

[0044] In step 82 the development server 12 determines whether the workspace for the developer (user01 in this example) has an associated workbench for user 01 in the virtualhost 01. If in step 82 the development server 12 determines the workspace for user01 does not have an associated workbench for the developer (user01 in this example), then the No branch is taken to step 84. In step 84, the development server 12 creates a workbench for the developer (user01 in this example) associated with the workspace for user01 in the virtualhost 01. If in step 82 the development server 12 determines the workspace for the developer (user01 in this example) does have an associated workspace for user01, then the Yes branch is taken to step 86. In step 86, the development server 12 creates a link between the project1 in the workspace for the developer (user01 in this example) and the project1 in the workbench for the developer (user01 in this example) as shown in FIG. 3. In step 88, the development server 12 creates a link from the project1 in workbench for user01 to a running area in the development server 12 as illustrated in FIG. 1.

[0045] An exemplary method for editing a project in a user workbench in the development server 12 will now be described with reference to FIGS. 1-3 and 7. In step 90, a developer at the remote workstation computing device 12 begins this method for accessing a project, such as a computer file by way of example, for editing or other tasks. In step 92, the remote workstation computing device 12 establishes a secure channel with the development system 12 using a remote file system technology, although other manners for establishing a secure or unsecure connection could be used. The domain name server 18 is used to assign an IP

addresses to the virtualhost in the development server 12 which is used by the remote workstation computing device 14 to access and interact with one or more activated projects, although other manners for connecting and communicating between the devices can be used.

[0046] In step 94 the development server 12 determines whether the developer (user01) at the remote workstation computing device 12 is permitted to access the workbench for user01. If in step 94 the development server 12 determines the developer (user01) at the remote workstation computing device 12 is not permitted to access the workbench for developer (user01), then the No branch is taken to step 96 which returns an error message. If in step 94 the development server 12 determines the developer (user01) at the remote workstation computing device 12 is permitted to access the workbench for developer (user01), then the Yes branch is taken to step 98.

[0047] In step 98, the development system 12 synchronizes the links to the one or more projects in the workbench for user01 to the one or more real files for the projects located in the workbench for user01. In step 100, the development system 12 provides access to the synchronized projects to developer (user01) at the remote workstation computing device 12 through the virtualhost, although other manners for providing access can be used.

[0048] An exemplary method for triggering an interpretation of a project in the development server 12 will now be described with reference to FIGS. 1-3 and 8. In step 102, a developer (user01) at the remote workstation computing device 12 begins this method for triggering an interpretation of a project in the development server 12.

[0049] In step 104 the development server 12 determines whether the developer (user01) requesting an interpretation of the project1 is allowed to establish a connection with the virtualhost1, although other devices and manners for requesting an interpretatin can be used, such as a request from a manager at the manager computng device 16. If in step 104 the development server 12 determines the developer (user01) requesting an interpretation of the project1 is

not allowed to establish a connection with the virtualhost1, then the No branch is taken to step 106 where the request is rejected. If in step 104 the development server 12 determines the developer (user01) requesting an interpretation of the project1 is allowed to establish a connection with the virtualhost1, then the Yes branch is taken to step 108.

[0050] In step 108 the development server 12 determines whether the requested project1 or other file exists in the running area for interpretation. If in step 108 the development server 12 determines the requested project1 does not exist in the running area for interpretation, then the No branch is taken to step 110 where an error message is returned. If in step 108 the development server 12 determines the requested project1 does exist in the running area for interpretation, then the Yes branch is taken to step 112.

[0051] In step 112, the development server 12 synchronizes the project1 in the running area with the project1 in the workspace for user01. In step 114, the development server 12 executes the interpreter to interpret the instructions in project1 or other file being interpreted. In step 116 the development server prints the output generated by the interpretation, although the output could be handled or stored in other manners.

[0052] Accordingly as illustrated and described herein, this technology provides a remote workbench and a remote virtual host. The remote workbench provides a working area in the development server 12 that can be accessed remotely by software developers at a remote workstation computing device and lets them work on tasks on the project as if they were located locally. The remote virtual host is a virtual host that is assigned to the developer through which the developer can launch and execute tasks on any project that is in its workbench. The development server 12 in the examples described herein is TCP/IP accessible, executes the programmed instructions for one or more aspects of the invention and is where the interpreter is installed. Additionally, with this technology, a project manager at a manager computing device 16 can verify with the development server 12 the status of any project at any moment. Even though working remotely, the developers at each remote workbench computing device never face

concurrency because every developer owns a working copy in the development server 12.

[0053] Having thus described the basic concept of the invention, it will be rather apparent to those skilled in the art that the foregoing detailed disclosure is intended to be presented by way of example only, and is not limiting. Various alterations, improvements, and modifications will occur and are intended to those skilled in the art, though not expressly stated herein. These alterations, improvements, and modifications are intended to be suggested hereby, and are within the spirit and scope of the invention. Additionally, the recited order of processing elements or sequences, or the use of numbers, letters, or other designations therefore, is not intended to limit the claimed processes to any order except as may be specified in the claims. Accordingly, the invention is limited only by the following claims and equivalents thereto.

CLAIMS

What is claimed is:

1. A method for managing a software project, the method comprising:

assigning one of one or more virtual hosts in one of one or more workspaces in a development computing device to a remote computing device;

generating with the development computing device at least one link in the one of the one or more workspaces to at least one of one or more working copies of projects in one of one or more work benches in the one of the one or more virtual hosts;

generating with the development computing device at least one other link in the one of the one or more workspaces to the linked one of the one or more working copies of projects activated in a running area of the development computing device; and

providing with the development computing device access to the activated one of the one or more working copies of projects to the remote computing device to execute one or more tasks.

2. The method as set forth in claim 1 further comprising:

linking with the development computing device the activated one of the one or more working copies of projects to one of one or more control version repositories; and

replicating with the development computing device and the one of one or more control version repositories each of the executed one or more tasks on the activated one of the one or more working copies of projects on the other one or more workspaces.

3. The method as set forth in claim 2 further comprising identifying with the development computing device which one of the one of one or more control version repositories to link with the activated one of the one or more working copies of projects.

4. The method as set forth in claim 1 further comprising:

receiving a request at the development computing device to interpret the activated one of the one or more working copies of projects; and

interpreting with an interpreter in the development computing device the activated one of the one or more working copies of projects.

5. The method as set forth in claim 4 further comprising at least one of updating and upgrading the interpreter in the development computing device.

6. The method as set forth in claim 1 further comprising:

receiving at the development computing device an activation request for one of the one or more working copies of projects; and

activating with the development computing device the requested one of the one or more working copies of projects to create the activated one of the one or more working copies of projects.

7. The method as set forth in claim 1 further comprising creating with the development computing device the one or more virtual hosts.

8. The method as set forth in claim 7 further comprising associating a domain name for each of the one or more virtual hosts to an IP address for the development computing device.

9. A computer readable medium having stored thereon instructions for managing a software project comprising machine executable code which when executed by at least one processor, causes the processor to perform steps:

assigning one of one or more virtual hosts in one of one or more workspaces in a development computing device to a remote computing device;

generating at least one link in the one of the one or more

workspaces to at least one of one or more working copies of projects in one of one or more work benches in the one of the one or more virtual hosts;

generating at least one other link in the one of the one or more workspaces to the linked one of the one or more working copies of projects activated in a running area of the development computing device; and

providing access to the activated one of the one or more working copies of projects to the remote computing device to execute one or more tasks.

10. The medium as set forth in claim 9 further comprising:

linking the activated one of the one or more working copies of projects to one of one or more control version repositories; and

replicating each of the executed one or more tasks on the activated one of the one or more working copies of projects on the other one or more workspaces.

11. The medium as set forth in claim 10 further comprising identifying which one of the one of one or more control version repositories to link with the activated one of the one or more working copies of projects.

12. The medium as set forth in claim 9 further comprising:

receiving a request to interpret the activated one of the one or more working copies of projects; and

interpreting with an interpreter the activated one of the one or more working copies of projects.

13. The medium as set forth in claim 12 further comprising at least one of updating and upgrading the interpreter.

14. The medium as set forth in claim 9 further comprising:

receiving an activation request for one of the one or more working copies of projects; and

activating the requested one of the one or more working

copies of projects to create the activated one of the one or more working copies of projects.

15. The medium as set forth in claim 9 further comprising creating the one or more virtual hosts.

16. The medium as set forth in claim 15 further comprising associating a domain name for each of the one or more virtual hosts to an IP address for the development computing device.

17. A software development management apparatus comprising:

one or more processors;

a memory coupled to the one or more processors which are configured to execute programmed instructions stored in the memory comprising:

assigning one of one or more virtual hosts in one of one or more workspaces in a development computing device to a remote computing device;

generating at least one link in the one of the one or more workspaces to at least one of one or more working copies of projects in one of one or more work benches in the one of the one or more virtual hosts;

generating at least one other link in the one of the one or more workspaces to the linked one of the one or more working copies of projects activated in a running area of the development computing device; and

providing access to the activated one of the one or more working copies of projects to the remote computing device to execute one or more tasks.

18. The apparatus as set forth in claim 17 wherein the one or more processors is further configured to execute programmed instructions stored in the memory further comprising:

linking the activated one of the one or more working copies of projects to one of one or more control version repositories; and

replicating each of the executed one or more tasks on the activated one of the one or more working copies of projects on the other one or more workspaces.

19. The apparatus as set forth in claim 18 wherein the one or more processors is further configured to execute programmed instructions stored in the memory further comprising identifying which one of the one of one or more control version repositories to link with the activated one of the one or more working copies of projects.

20. The apparatus as set forth in claim 17 wherein the one or more processors is further configured to execute programmed instructions stored in the memory further comprising:

receiving a request to interpret the activated one of the one or more working copies of projects; and

interpreting with an interpreter the activated one of the one or more working copies of projects.

21. The apparatus as set forth in claim 20 wherein the one or more processors is further configured to execute programmed instructions stored in the memory further comprising at least one of updating and upgrading the interpreter.

22. The apparatus as set forth in claim 17 wherein the one or more processors is further configured to execute programmed instructions stored in the memory further comprising:

receiving an activation request for one of the one or more working copies of projects; and

activating the requested one of the one or more working copies of projects to create the activated one of the one or more working copies of projects.

23. The apparatus as set forth in claim 17 wherein the one or

more processors is further configured to execute programmed instructions stored in the memory further comprising creating the one or more virtual hosts.

24. The apparatus as set forth in claim 23 wherein the one or more processors is further configured to execute programmed instructions stored in the memory further comprising associating a domain name for each of the one or more virtual hosts to an IP address for the development computing device.

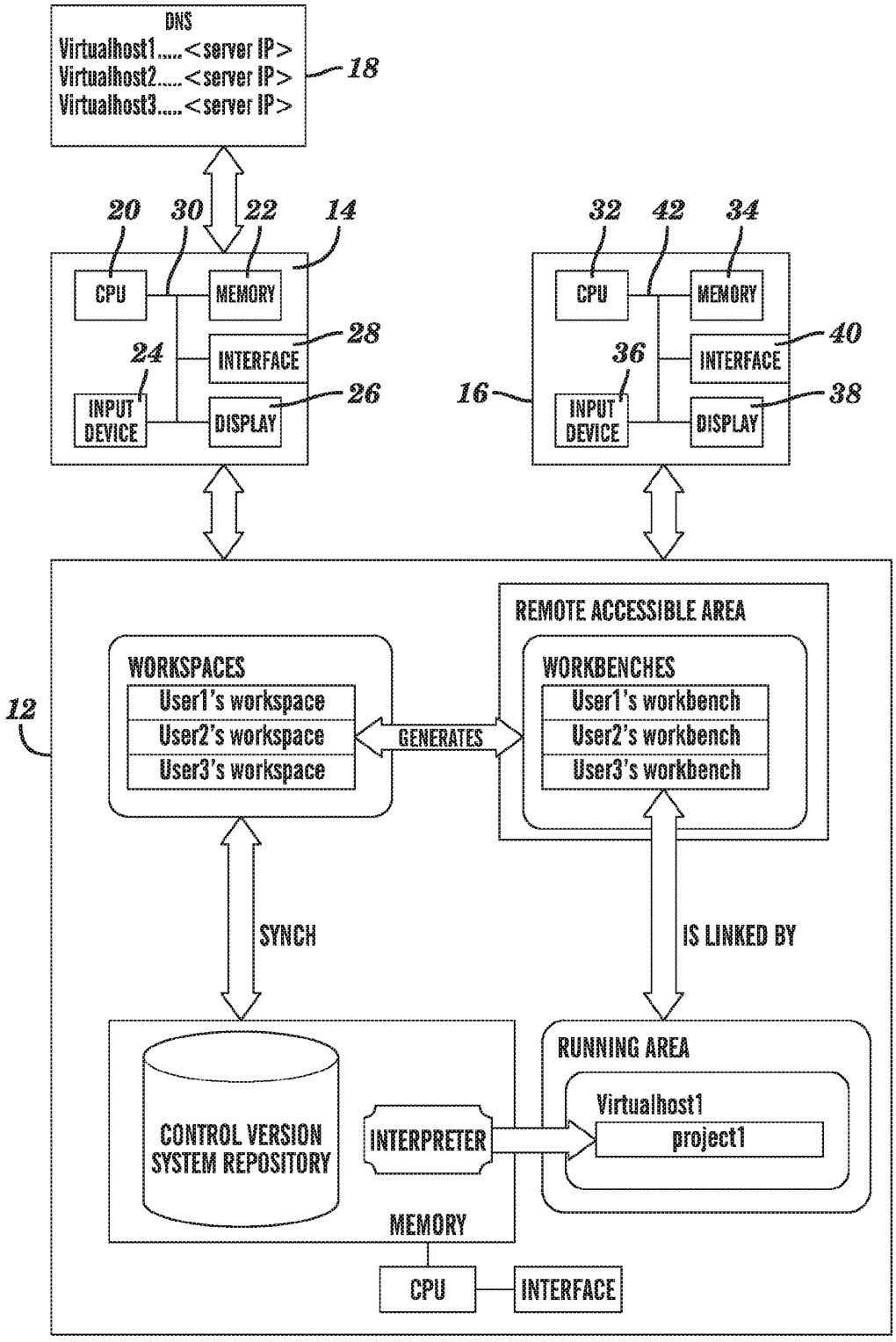

DNS
Virtualhost1.....<server IP>
Virtualhost2.....<server IP>
Virtualhost3.....<server IP>
18
20
30
22
14
CPU
MEMORY
24
INTERFACE
28
INPUT DEVICE
DISPLAY
26
32
42
34
CPU
MEMORY
36
INTERFACE
40
16
INPUT DEVICE
DISPLAY
38
12
REMOTE ACCESSIBLE AREA
WORKSPACES
User1's workspace
User2's workspace
User3's workspace
GENERATES
WORKBENCHES
User1's workbench
User2's workbench
User3's workbench
SYNCH
IS LINKED BY
CONTROL VERSION SYSTEM REPOSITORY
INTERPRETER
RUNNING AREA
Virtualhost1
project1
MEMORY
CPU
INTERFACE

FIG. 1

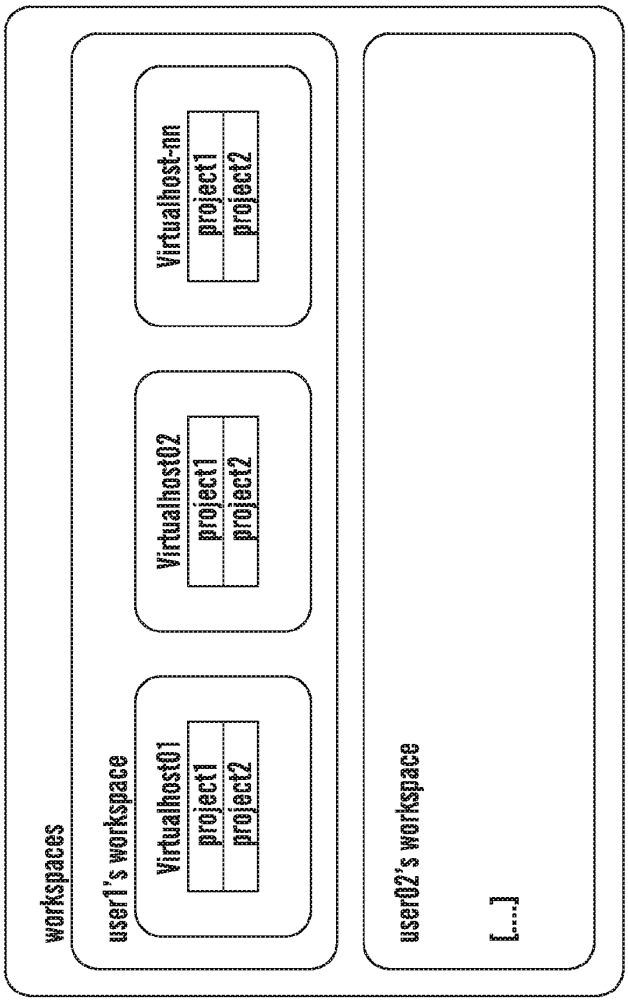

workspaces
user1's workspace
Virtualhost01
project1
project2
Virtualhost02
project1
project2
Virtualhost-nn
project1
project2
user02's workspace
[...]

FIG. 2

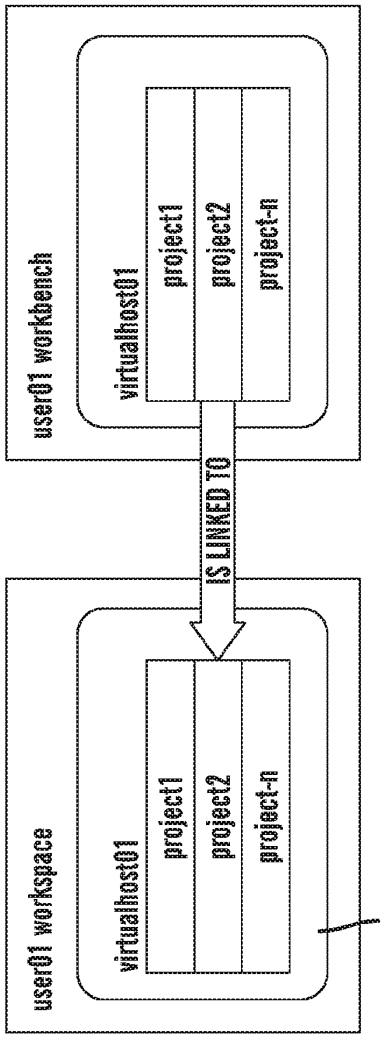

user01 workbench
virtualhost01
project1
project2
project-n
IS LINKED TO
user01 workspace
virtualhost01
project1
project2
project-n
3.1

FIG. 3

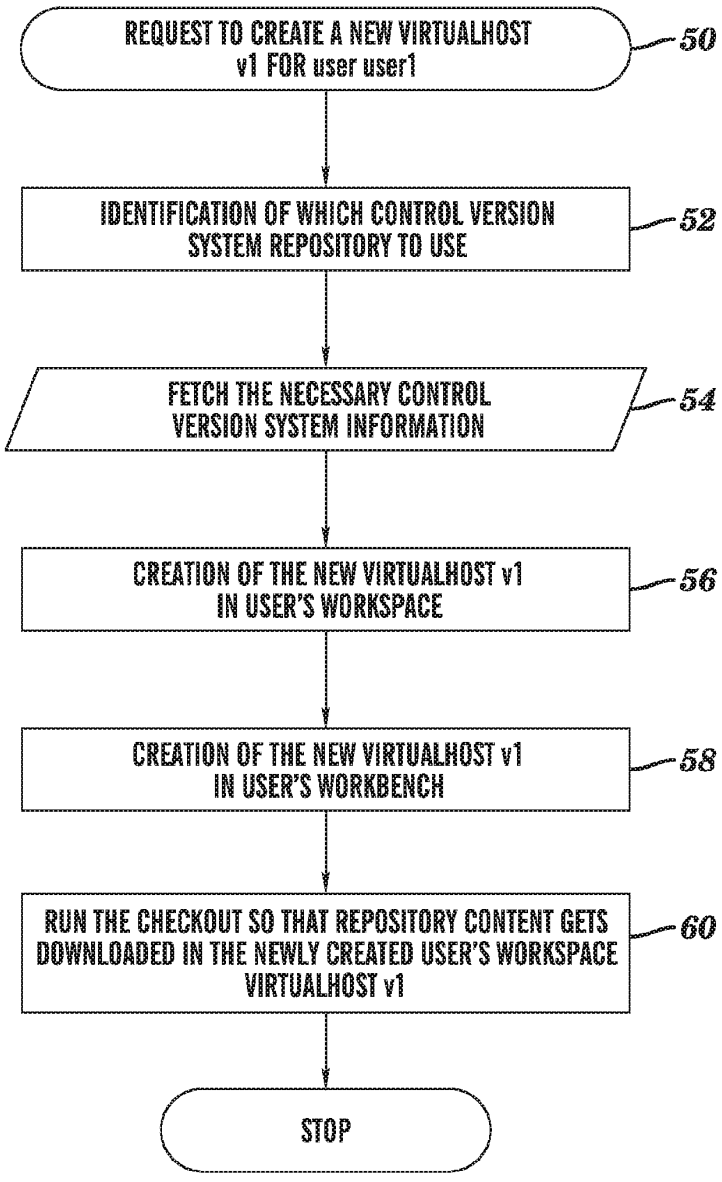

REQUEST TO CREATE A NEW VIRTUALHOST v1 FOR user user1
50
IDENTIFICATION OF WHICH CONTROL VERSION SYSTEM REPOSITORY TO USE
52
FETCH THE NECESSARY CONTROL VERSION SYSTEM INFORMATION
54
CREATION OF THE NEW VIRTUALHOST v1 IN USER'S WORKSPACE
56
CREATION OF THE NEW VIRTUALHOST v1 IN USER'S WORKBENCH
58
RUN THE CHECKOUT SO THAT REPOSITORY CONTENT GETS DOWNLOADED IN THE NEWLY CREATED USER'S WORKSPACE VIRTUALHOST v1
60
STOP

FIG. 4

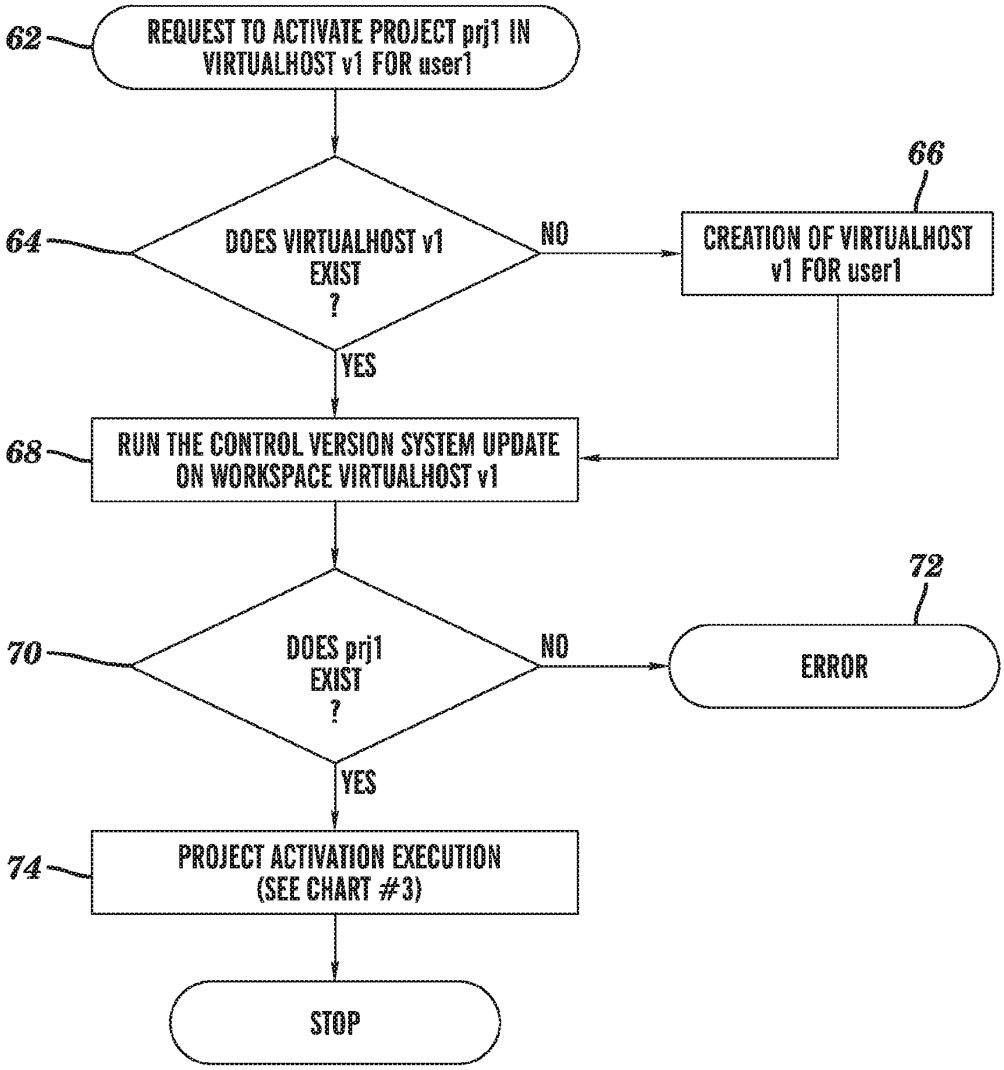

62
REQUEST TO ACTIVATE PROJECT prj1 IN VIRTUALHOST v1 FOR user1
64
DOES VIRTUALHOST v1 EXIST ?
NO
66
CREATION OF VIRTUALHOST v1 FOR user1
YES
68
RUN THE CONTROL VERSION SYSTEM UPDATE ON WORKSPACE VIRTUALHOST v1
70
DOES prj1 EXIST ?
NO
72
ERROR
YES
74
PROJECT ACTIVATION EXECUTION (SEE CHART #3)
STOP

FIG. 5

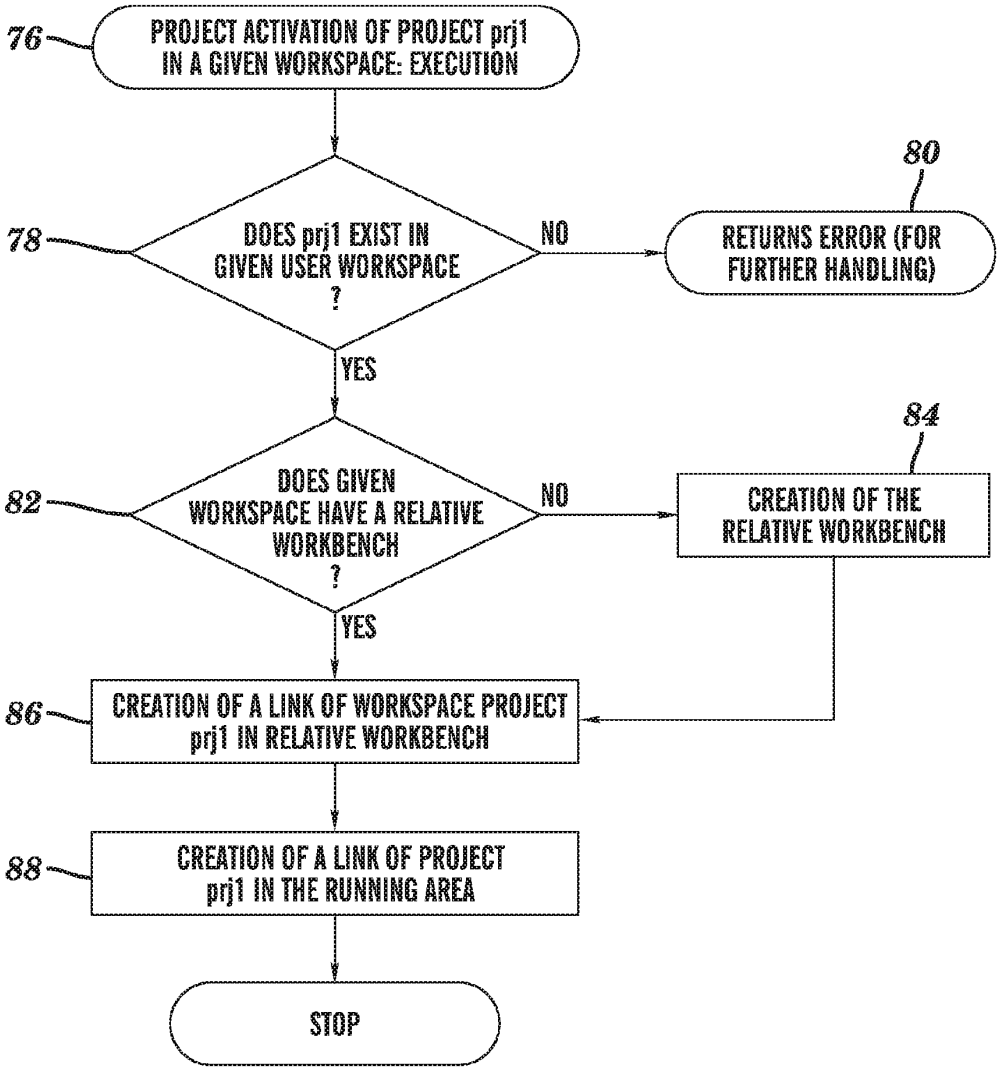

76
PROJECT ACTIVATION OF PROJECT prj1 IN A GIVEN WORKSPACE: EXECUTION
78
DOES prj1 EXIST IN GIVEN USER WORKSPACE ?
NO
80
RETURNS ERROR (FOR FURTHER HANDLING)
YES
82
DOES GIVEN WORKSPACE HAVE A RELATIVE WORKBENCH ?
NO
84
CREATION OF THE RELATIVE WORKBENCH
YES
86
CREATION OF A LINK OF WORKSPACE PROJECT prj1 IN RELATIVE WORKBENCH
88
CREATION OF A LINK OF PROJECT prj1 IN THE RUNNING AREA
STOP

FIG. 6

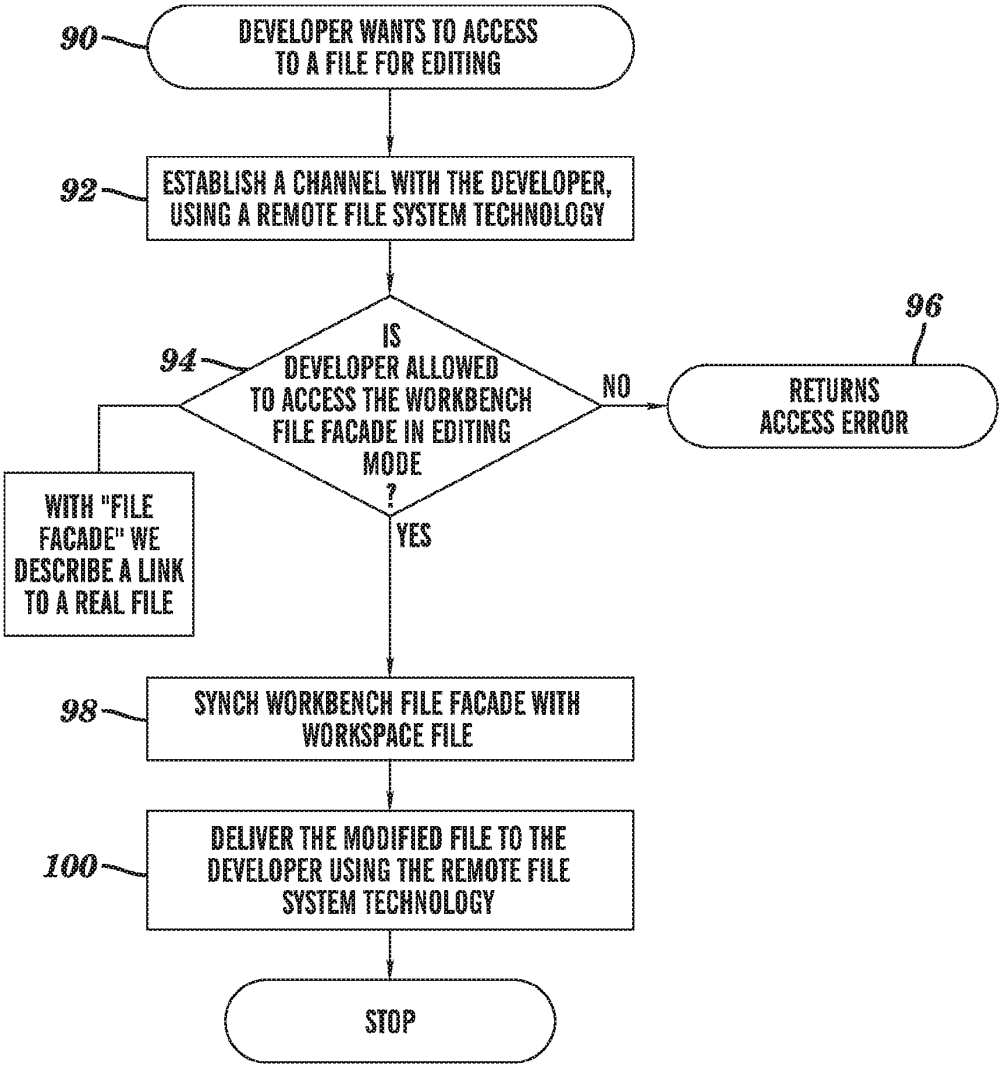

90
DEVELOPER WANTS TO ACCESS TO A FILE FOR EDITING
92
ESTABLISH A CHANNEL WITH THE DEVELOPER, USING A REMOTE FILE SYSTEM TECHNOLOGY
94
IS DEVELOPER ALLOWED TO ACCESS THE WORKBENCH FILE FACADE IN EDITING MODE ?
NO
96
RETURNS ACCESS ERROR
YES
WITH "FILE FACADE" WE DESCRIBE A LINK TO A REAL FILE
98
SYNCH WORKBENCH FILE FACADE WITH WORKSPACE FILE
100
DELIVER THE MODIFIED FILE TO THE DEVELOPER USING THE REMOTE FILE SYSTEM TECHNOLOGY
STOP

FIG. 7

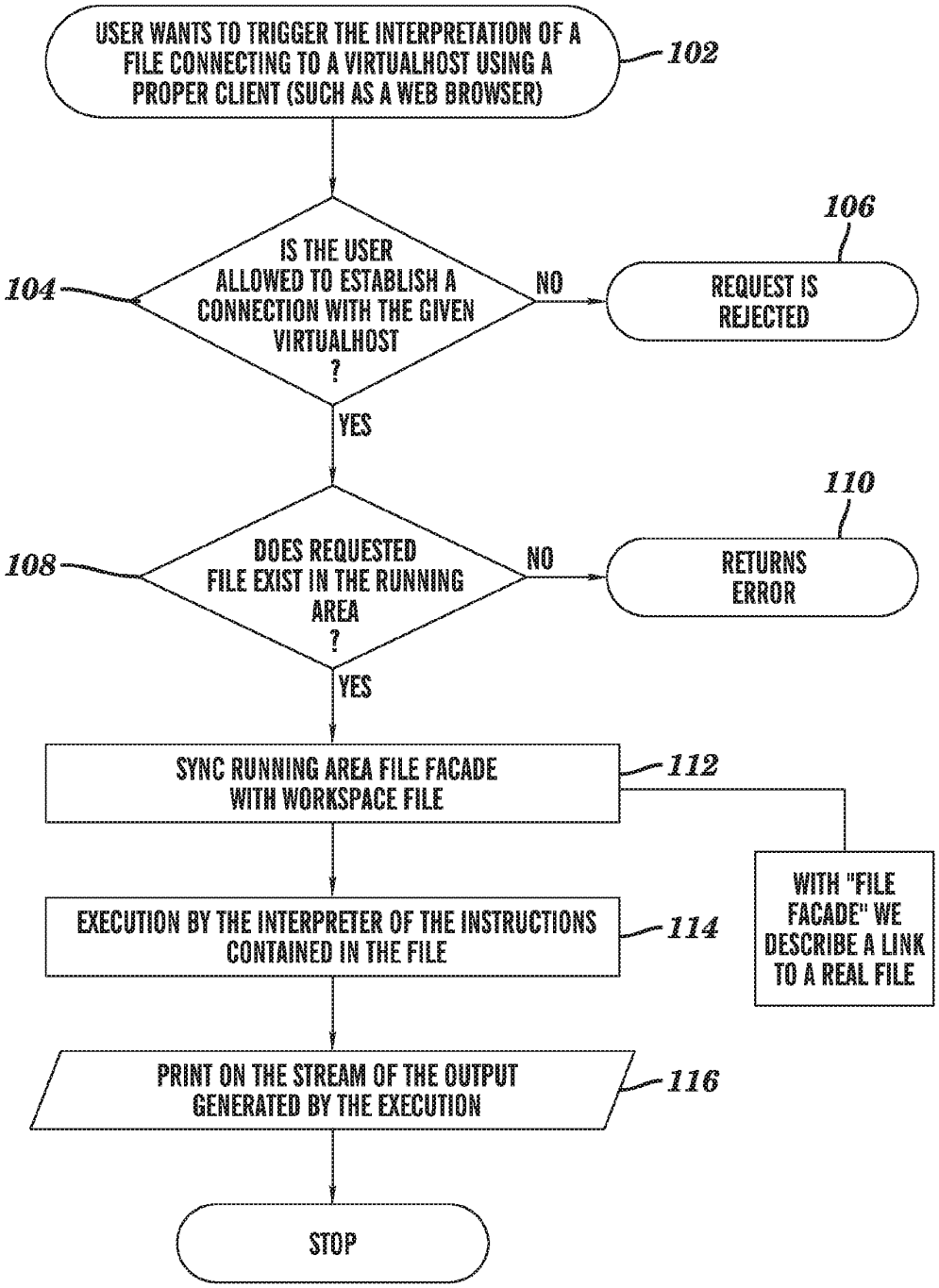

USER WANTS TO TRIGGER THE INTERPRETATION OF A FILE CONNECTING TO A VIRTUALHOST USING A PROPER CLIENT (SUCH AS A WEB BROWSER)
102
IS THE USER ALLOWED TO ESTABLISH A CONNECTION WITH THE GIVEN VIRTUALHOST ?
104
NO
106
REQUEST IS REJECTED
YES
DOES REQUESTED FILE EXIST IN THE RUNNING AREA ?
108
NO
110
RETURNS ERROR
YES
SYNC RUNNING AREA FILE FACADE WITH WORKSPACE FILE
112
WITH "FILE FACADE" WE DESCRIBE A LINK TO A REAL FILE
EXECUTION BY THE INTERPRETER OF THE INSTRUCTIONS CONTAINED IN THE FILE
114
PRINT ON THE STREAM OF THE OUTPUT GENERATED BY THE EXECUTION
116
STOP

FIG. 8

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Veröffentlichungsnummer:
(11) Publication number: EP 2 606 436 A0
(11) Numéro de publication:

Internationale Anmeldung veröffentlicht durch die Weltorganisation für geistiges Eigentum unter der Nummer:

WO 2012/024380 (Art. 153(3) EPÜ).

International application published by the World Intellectual Property Organization under number:

WO 2012/024380 (Art. 153(3) EPC).

Demande internationale publiée par l'Organisation Mondiale de la Propriété Intellectuelle sous le numéro:

WO 2012/024380 (art. 153(3) CBE).

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
23 February 2012 (23.02.2012)

PCT

(10) International Publication Number
WO 2012/024380 A2

(51) International Patent Classification:
G06F 17/21 (2006.01) *G06F 17/00* (2006.01)

(21) International Application Number:
PCT/US2011/048058

(22) International Filing Date:
17 August 2011 (17.08.2011)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
12/806,716 20 August 2010 (20.08.2010) US

(71) Applicant *(for all designated States except US)*: **US-ABLENET, INC.** [US/US]; 28 W. 23rd St., 6th Floor, New York, NY 10010 (US).

(72) Inventor: **SCODA, Enrico**; Via Cividina 416/3, I-33035 Martignacco (UD) (IT).

(74) Agents: **LEINBERG, Gunnar, G.** et al.; LeClairRyan, P.C., 290 Linden Oaks, Rochester, NY 14625 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

---- *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

(54) Title: METHODS FOR FURTHER ADAPTING XSL TO HTML DOCUMENT TRANSFORMATIONS AND DEVICES THEREOF

(57) Abstract: A method, computer readable medium and apparatus for further adapting XSL to HTML document transformations includes identifying with a web computing device one or more rules in an HTML document. An action associated with each of the identified one or more rules is identified with the web computing device. The identified actions are filtered with the web computing device based on one or more filtering rules when two or more of the identified actions have a match. The remaining identified actions after the filtering are applied with the web computing device to transform the one or more rules in the HTML document. The transformed HTML document is provided by the web computing device.

WO 2012/024380 A2

METHODS FOR FURTHER ADAPTING XSL TO HTML DOCUMENT TRANSFORMATIONS AND DEVICES THEREOF

FIELD

[0001] This technology generally relates to methods for adapting eXtensible Stylesheet Language (XSL) to HTML document transformations and devices thereof.

BACKGROUND

[0002] The introduction of eXtensible Markup Language (XML) and the EXtensibleStylesheet Language (XSL) specifications has provided an easy way to transform documents between various formats. This functionality has been included into Web development frameworks, giving them the ability to transform automatically an XML file into a document with different format such as HTML or XHTML, integrating the original data with graphic layout and user interface components. The XSL specifications are based on special constructs called templates that match a single element or a set of similar elements and rewrite them and their content based on instructions defined in the template.

[0003] Unfortunately, a problem arises when the structure of the XML document to process is not well defined. For example, the same element can be used for different purposes inside the XML document and based on these purposes multiple different transformations must be implemented. The problems get even worse when the task involves transforming HTML documents. For example the link element "a" can appear over a thousand times in different sections of a web page, such as in the main navigation bar, in hidden menus, to make images clickable, and as a button to execute JavaScript functions. Writing XSL templates that modify all these elements can increase complexity in an unpredictable way.

[0004] An illustrative example of these difficulties with a simple XSL file managing HTML links ("a" elements) is shown in FIG. 1. As illustrated, the XSL file:

(1) changes the "href" attribute using an XPath extension function called myext:normalize-url();

(2) if the link contains "target" attribute with value "_blank", remove it and set "class" attribute to "external" value, otherwise "class" attribute will get value "internal";

(3) if the "a" content is an image ("img" element), then set new content to image "alt" attribute otherwise apply templates to its children; and

(4) use the "identity" template (last one) to simply copy elements as they are if they are not "a" elements.

[0005] Accordingly, as shown the same instructions have to be written at least twice to keep templates simple and to cover all the combinations of the above transformations. More powerful XSL constructs like name templates or xsl:choose or xsl:if could be utilized and the resulting XSL document will be more optimized, but also will be more complex and less readable.

SUMMARY

[0006] A method for further adapting XSL to HTML document transformations includes identifying with a web computing device one or more rules matching one or more elements in an HTML document. An action associated with each of the identified one or more rules is identified with the web computing device. The identified actions are filtered with the web computing device based on one or more filtering rules when two or more of the identified actions have a match. The remaining identified actions after the filtering are applied with the web computing device to transform the one or more matching elements in the HTML document. The transformed HTML document is provided by the web computing device.

[0007] A computer readable medium having stored thereon instructions processing multiple documents from multiple sites comprising machine executable code which when executed by at least one processor, causes the processor to perform steps including identifying one or more rules matching one

or more elements in an HTML document. An action associated with each of the identified one or more rules is identified. The identified actions are filtered based on one or more filtering rules when two or more of the identified actions have a match. The remaining identified actions after the filtering are applied to transform the one or more matching elements in the HTML document. The transformed HTML document is provided.

[0008] A web computing apparatus includes one or more processors and a memory coupled to the one or more processors which are configured to execute programmed instructions stored in the memory including identifying one or more rules matching one or more elements in an HTML document. An action associated with each of the identified one or more rules is identified. The identified actions are filtered based on one or more filtering rules when two or more of the identified actions have a match. The remaining identified actions after the filtering are applied to transform the one or more matching elements in the HTML document. The transformed HTML document is provided.

[0009] This technology provides a number of advantages including providing a method, computer readable medium and apparatus that further adapts XSL to HTML document transformations. More specifically, examples of this technology identify a set of similar elements, i.e. sharing same properties, and then defines a set of actions to take on those elements, such as rename, set/change attributes, and set their content. With this technology, if one element is member of two or more sets, then all actions defined for these sets can be applied to the element. This enable smaller sets of instructions to be defined for well defined sets of elements without the need of rewriting same instructions for different sets. These sets of instructions are then transformed into XSL instructions that can be processed by any XSL processor.

BRIEF DESCRIPTION OF THE DRAWINGS

[00010] FIG. 1 is an exemplary XSL file managing HTML links;

[00011] FIG. 2 is a block diagram of an exemplary system with a proxy server configured to adapt XSL to HTML document transformations;

[00012] FIG. 3 is a flow chart of an exemplary method for adapting XSL to HTML document transformations;

[00013] FIG. 4 are three exemplary rules; and

[00014] FIG. 5 is a resulting XSL file from an automatic translation of the rules files shown in FIG. 4 into XSL format.

DETAILED DESCRIPTION

[00015] An exemplary environment 10 with a proxy server 12 configured to further adapt XSL to HTML document transformation is illustrated in FIG. 1, although this technology can be implemented on other types of devices, although this technology can be implemented on other types of devices, such as one of the web server devices 16(1)-16(n) by way of example only. The exemplary environment 10 includes the proxy server or apparatus 12, client devices 14(1)-14(n), the web server devices 16(1)-16(n), and communication networks 18(1)-18(2), although other numbers and types of systems, devices, and/or elements in other configurations and environments with other communication network topologies can be used. This technology provides a number of advantages including providing a method, computer readable medium and an apparatus that further adapts XSL to HTML document transformations.

[00016] Referring more specifically to FIG. 1, the proxy server 12 includes a central processing unit (CPU) or processor 13, a memory 15, and an interface system 17 which are coupled together by a bus 19 or other link, although other numbers and types of components, parts, devices, systems, and elements in other configurations and locations can be used. The processor 13 in the proxy server 12 executes a program of stored instructions one or more aspects of the present invention as described and illustrated by way of the embodiments herein, although the processor could execute other numbers and types of programmed instructions.

[00017] The memory 15 in the proxy server 12 stores these programmed instructions for one or more aspects of the present invention as described and illustrated herein, although some or all of the programmed instructions could be

stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, DVD ROM, or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to the processor 13, can be used for the memory 15 in the proxy server 12.

[00018] The interface system 17 in the proxy server 12 is used to operatively couple and communicate between the proxy server 12 and the client devices 14(1)-14(n) and the web server devices 16(1)-16(n) via the communication networks 18(1) and 18(2), although other types and numbers of communication networks with other types and numbers of connections and configurations can be used. By way of example only, the communication networks 18(1) and 18(2) can use TCP/IP over Ethernet and industry-standard protocols, including HTTP, HTTPS, WAP, and SOAP, although other types and numbers of communication networks, such as a direct connection, a local area network, a wide area network, modems and phone lines, e-mail, and wireless and hardwire communication technology, each having their own communications protocols, can be used.

[00019] Each of the client devices 14(1)-14(n) enables a user to request, get and interact with web pages from one or more web sites hosted by the web server devices 16(1)-16(n) through the proxy server 12 via one or more communication networks, although one or more of the client devices 14(1)-14(n) could access content and utilize other types and numbers of applications from other sources and could provide a wide variety of other functions for the user. Although multiple client devices 14(1)-14(n) are shown, other numbers and types of user computing systems could be used. In this example, the client devices 14(1)-14(n) comprise mobile devices with Internet access that permit a website form page or other retrieved data to be displayed, although each of the client devices 14(1)-14(n). By way of example only, one or more of the client devices 14(1)-14(n) can comprise smart phones, personal digital assistants, or computers.

[00020] Each of client devices 14(1)-14(n) in this example is a computing device that includes a central processing unit (CPU) or processor 20, a memory 22, user input device 24, a display 26, and an interface system 28, and which are coupled together by a bus 30 or other link, although one or more of client devices 14(1)-14(n) can include other numbers and types of components, parts, devices, systems, and elements in other configurations. The processor 20 in each of client devices 14(1)-14(n) executes a program of stored instructions for one or more aspects of the present invention as described and illustrated herein, although the processor could execute other numbers and types of programmed instructions.

[00021] The memory 22 in each of the client devices 14(1)-14(n) stores these programmed instructions for one or more aspects of the present invention as described and illustrated herein, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to processor 20 can be used for the memory 22 in each of the client devices 14(1)-14(n.

[00022] The user input device 24 in each of the client devices 14(1)-14(n) is used to input selections, such as requests for a particular website form page or to enter data in fields of a form page, although the user input device could be used to input other types of data and interact with other elements. The user input device can include keypads, touch screens, and/or vocal input processing systems although other types and numbers of user input devices can be used.

[00023] The display 26 in each of the client devices 14(1)-14(n) is used to show data and information to the user, such as website page by way of example only. The display in each of the client devices 14(1)-14(n) is a phone screen display, although other types and numbers of displays could be used depending on the particular type of client device.

[00024] The interface system 28 in each of the client devices 14(1)-14(n) is used to operatively couple and communicate between the client devices 14(1)-14(n) and the proxy server 12 and web server devices 16(1)-16(n) over the communication networks 18(1) and 18(2), although other types and numbers of communication networks with other types and numbers of connections and configurations can be used.

[00025] The web server devices 16(1)-16(n) provide one or more pages from one or more web sites for use by one or more of the client devices 14(1)-14(n) via the proxy server 12, although the web server devices 16(1)-16(n) can provide other numbers and types of applications and/or content and can have provide other numbers and types of functions. Although web server devices 16(1)-16(n) are shown for ease of illustration and discussion, other numbers and types of web server systems and devices can be used.

[00026] Each of the web server devices 16(1)-16(n) include a central processing unit (CPU) or processor, a memory, and an interface system which are coupled together by a bus or other link, although each of the web server devices 16(1)-16(n) could have other numbers and types of components, parts, devices, systems, and elements in other configurations and locations can be used. The processor in each of the web server devices 16(1)-16(n) executes a program of stored instructions one or more aspects of the present invention as described and illustrated by way of the embodiments herein, although the processor could execute other numbers and types of programmed instructions.

[00027] The memory in each of the web server devices 16(1)-16(n) stores these programmed instructions for one or more aspects of the present invention as described and illustrated by way of the embodiments, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, DVD ROM, or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system

that is coupled to the processor, can be used for the memory in each of the web server devices 16(1)-16(n).

[00028] The interface system in each of the web server devices 16(1)-16(n) is used to operatively couple and communicate between the web server devices 16(1)-16(n) and the proxy server 12 and the client devices 14(1)-14(n) via communication networks 18(1) and 18(2), although other types and numbers of communication networks with other types and numbers of connections and configurations can be used.

[00029] Although embodiments of the proxy server 12, the client devices 14(1)-14(n), and the web server devices 16(1)-16(n), are described and illustrated herein, each of the client devices 14(1)-14(n), the proxy server 12, and the web server devices 16(1)-16(n), can be implemented on any suitable computer system or computing device. It is to be understood that the devices and systems of the embodiments described herein are for exemplary purposes, as many variations of the specific hardware and software used to implement the embodiments are possible, as will be appreciated by those skilled in the relevant art(s).

[00030] Furthermore, each of the systems of the embodiments may be conveniently implemented using one or more general purpose computer systems, microprocessors, digital signal processors, and micro-controllers, programmed according to the teachings of the embodiments, as described and illustrated herein, and as will be appreciated by those ordinary skill in the art.

[00031] In addition, two or more computing systems or devices can be substituted for any one of the systems in any embodiment of the embodiments. Accordingly, principles and advantages of distributed processing, such as redundancy and replication also can be implemented, as desired, to increase the robustness and performance of the devices and systems of the embodiments. The embodiments may also be implemented on computer system or systems that extend across any suitable network using any suitable interface mechanisms and communications technologies, including by way of example only telecommunications in any suitable form (e.g., voice and modem), wireless

communications media, wireless communications networks, cellular communications networks, G3 communications networks, Public Switched Telephone Network (PSTNs), Packet Data Networks (PDNs), the Internet, intranets, and combinations thereof.

[00032] The embodiments may also be embodied as a computer readable medium having instructions stored thereon for one or more aspects of the present invention as described and illustrated by way of the embodiments herein, as described herein, which when executed by a processor, cause the processor to carry out the steps necessary to implement the methods of the embodiments, as described and illustrated herein.

[00033] An exemplary method for further adapting XSL to HTML document transformations with proxy server 12 will now be described with reference to FIGS. 2-5, although again this technology can be executed by other types of devices, such as by one of the web server devices 16(1)-16(n) and without a proxy server by way of example only. Referring more specifically to FIG. 3, in step 100 the proxy server 12 receives an HTML document to transform from one of the web server devices 16(1)-16(n) for one of the client computing devices 14(1)-14(n), although this exemplary method can be executed by other types and numbers of devices. The proxy server 12 traverses the HTML document to identify each element and generate rules for each identified element. In this particular example, the rules files illustrated in FIG. 4 are rules written for HTML element "a", although the HTML document can have other types of and numbers of elements and rules.

[00034] In step 102, the proxy server 12 identifies the action(s) defined for each rule in the rules files shown in FIG. 4 matching the HTML element being analyzed, although other manners for finding the action(s) could be used. The action(s) are statements that when executed change some properties of the matched element and are defined by the XML element(s) that are a child or children of the rule execute section. The match section of the rule is an XPath expression used to identify the other HTML elements matching the rule, although other matching expressions could be used. In this exemplary embodiment the

available actions are: (1) remove-element: to remove the matched element; (2) replace-element: to replace the matched element with a new element (changing its name); (3) linearize-table: to take some (or all) the cells of the matched HTML table and place their content in a different order thanks to multiple instructions called show-cell (for example <show-cell row="3" col="2"/>; (4) set-meta-category: to set the value of a special attribute called un-meta for the matched element; (5)move-bottom: take matched element and move it to the bottom of the document; (6) set-attribute: to set/change the value of a given attribute of matched element; (7) remove-attribute: to remove a given attribute of matched element; (8) set-content: to set new content for the matched element; (9) append-content: to append new content after the last child of current element. It's evident that some elements to be transformed can match two or even more sets of actions. In this case all sets actions will be "eligible" to be applied to the element. The priority/conflict rules defined for the language will be applied to filter actions and to decide which ones to use.

[00035] In step 104, the proxy server 12 determines whether any of the identified actions for the rules match. If in step 104, the proxy server 12 determines there are no matching actions, then the No branch is taken to step 106. In step 106, the proxy server 12 executes the actions on the rules to transform the HTML document. If in step 104, the proxy server 12 determines there are matching actions, then the Yes branch is taken to step 108.

[00036] In step 108, the proxy server 12 applies one or more filtering rules to filter the out the matching actions which are not applicable, although other manners for filtering the matching actions can be used. In this example, the filtering rules are: (1) group all matching actions based on document order of appearance (2) if the action is remove-element, then remove all of the following: remove-element; replace-element; linearize-table; set-meta-category; move-bottom; set-attribute; remove-attribute; set-content; and append-content; (3) if the action is replace-element, then remove all of the following: remove-element; replace-element; and linearize-table; (4) if the action is linearize-table, then remove all of the following: remove-element; replace-element; linearize-table; set-meta-category; set-attribute; remove-attribute; set-content; append-content.

Furthermore remove all preceding: set-meta-category; set-attribute; remove-attribute; set-content; and append-content. (5) if the action is set-meta-category, then remove all following: remove-element, set-meta-category; (6) if action is move-bottom, remove all following: remove-element and move-bottom; (7) if the action is set-attribute, then remove all following: remove-element; set-attribute if name parameter of following action is equal to the name parameter of the matching action; and remove-attribute if name parameter of following action is equal to the name parameter of the matching action; (8) if the action is remove-attribute, then remove all following: remove-element; set-attribute if name parameter of following action is equal to the name parameter of the matching action and remove-attribute if name parameter of following action is equal to the name parameter of the matching action; (9) if the action is set-content, then remove all following: remove-element; set-content and append-content; and (10) if the action is append-content, then remove all following: remove-element; set-content; and append-content.

[00037] In step 110, the proxy server 12 applies the remaining action(s) which remain after the filtering to transform the elements of the HTML document. In step 112, the proxy server 12 provides the transformed elements of the HTML document.

[00038] In this particular example, the XSL file resulting from the automatic translation of the rules files shown in FIG. 4 into the XSL format is illustrated in FIG. 5. As shown, the resulting XSL file can be even more complex than the original XSL file shown in FIG. 1. This is expected since the new language has been created to transfer complexity at the machine level.

[00039] Accordingly, as illustrated and described herein this technology provides a number of advantages including providing a method, computer readable medium and an apparatus that further adapts XSL to HTML document transformations. More specifically, examples of this technology identify a set of similar elements, i.e. sharing the same properties, and then defines a set of actions to take on those elements, such as rename, set/change attributes, and set their content. With this technology, if one element is member of two or more sets, then

all actions defined for these sets can be applied to the element. This enable smaller sets of instructions to be defined for well defined sets of elements without the need of rewriting same instructions for different sets. These sets of instructions are then transformed into XSL instructions that can be processed by any XSL processor.

[00040] Having thus described the basic concept of the invention, it will be rather apparent to those skilled in the art that the foregoing detailed disclosure is intended to be presented by way of example only, and is not limiting. Various alterations, improvements, and modifications will occur and are intended to those skilled in the art, though not expressly stated herein. These alterations, improvements, and modifications are intended to be suggested hereby, and are within the spirit and scope of the invention. Additionally, the recited order of processing elements or sequences, or the use of numbers, letters, or other designations therefore, is not intended to limit the claimed processes to any order except as may be specified in the claims. Accordingly, the invention is limited only by the following claims and equivalents thereto.

CLAIMS

What is claimed is:

1. A method for further adapting XSL to HTML document transformations, the method comprising:

identifying with a web computing device one or more rules matching one or more elements in an HTML document;

identifying with the web computing device an action associated with each of the identified one or more rules;

filtering with the web computing device the identified actions based on one or more filtering rules when two or more of the identified actions have a match; and

applying with the web computing device the remaining identified actions after the filtering to transform the one or more matching elements in the HTML document

providing with the web computing device the transformed HTML document.

2. The method as set forth in claim 1 further comprising determining with the web computing device when two or more of the identified actions have a match.

3. The method as set forth in claim 1 wherein the filtering further comprises:

grouping with the web computing device each of the identified actions which have a match together; and

removing with the web computing device one or more of the actions in each of the groups based on the one or more filtering rules.

4. The method as set forth in claim 3 wherein the one or more filtering rules comprises removing remove-element, replace-element, linearize-table, set-meta-category, move-bottom, set-attribute, remove-attribute, set-content, and append-content when the identified action in the group is remove-element.

5. The method as set forth in claim 3 wherein the one or more filtering rules comprises removing remove-element, replace-element, and linearize-table when the identified action in the group is replace-element.

6. The method as set forth in claim 3 wherein the one or more filtering rules comprises removing remove-element, replace-element, linearize-table, set-meta-category, set-attribute, remove-attribute, set-content, append-content and removing all preceding set-meta-category, set-attribute, remove-attribute, set-content, and append-content when the identified action in the group is linearize-table.

7. The method as set forth in claim 3 wherein the one or more filtering rules comprises removing remove-element and set-meta-category when the identified action in the group is set-meta-category.

8. The method as set forth in claim 3 wherein the one or more filtering rules comprises removing remove-element and move-bottom when the identified action in the group is move-bottom.

9. The method as set forth in claim 3 wherein the one or more filtering rules comprises removing remove-element, set-attribute if name parameter of following action is equal to the name parameter of the matching action, and remove-attribute if name parameter of following action is equal to the name parameter of the matching action when the identified action in the group is set-attribute.

10. The method as set forth in claim 3 wherein the one or more filtering rules comprises removing remove-element, set-attribute if name parameter of following action is equal to the name parameter of the matching action and remove-attribute if name parameter of following action is equal to the name parameter of the matching action when the identified action in the group is remove-attribute.

11. The method as set forth in claim 3 wherein the one or more filtering rules comprises removing remove-element, set-content, and append-content when the identified action in the group is set-content.

12. The method as set forth in claim 3 wherein the one or more filtering rules comprises removing remove-element, set-content, and append-content when the identified action in the group is append-content.

13. The method as set forth in claim 1 wherein one or more of the actions can comprise one or more of remove-element replace-element, linearize-table, set-meta-category, move-bottom, set-attribute, remove-attribute, set-content, and append-content.

14. A computer readable medium having stored thereon instructions processing multiple documents from multiple sites comprising machine executable code which when executed by at least one processor, causes the processor to perform steps comprising:

identifying one or more rules matching one or more elements in an HTML document;

identifying an action associated with each of the identified one or more rules;

filtering the identified actions based on one or more filtering rules when two or more of the identified actions have a match; and

applying the remaining identified actions after the filtering to transform the one or more matching elements in the HTML document

providing the transformed HTML document.

15. The medium as set forth in claim 14 further comprising determining when two or more of the identified actions have a match.

16. The medium as set forth in claim 14 wherein the filtering further comprises:

grouping each of the identified actions which have a match together; and

removing one or more of the actions in each of the groups based on the one or more filtering rules.

17. The medium as set forth in claim 16 wherein the one or more filtering rules comprises removing remove-element, replace-element, linearize-table, set-meta-category, move-bottom, set-attribute, remove-attribute, set-content, and append-content when the identified action in the group is remove-element.

18. The medium as set forth in claim 16 wherein the one or more filtering rules comprises removing remove-element, replace-element, and linearize-table when the identified action in the group is replace-element.

19. The medium as set forth in claim 16 wherein the one or more filtering rules comprises removing remove-element, replace-element, linearize-table, set-meta-category, set-attribute, remove-attribute, set-content, append-content and removing all preceding set-meta-category, set-attribute, remove-attribute, set-content, and append-content when the identified action in the group is linearize-table.

20. The medium as set forth in claim 16 wherein the one or more filtering rules comprises removing remove-element and set-meta-category when the identified action in the group is set-meta-category.

21. The medium as set forth in claim 16 wherein the one or more filtering rules comprises removing remove-element and move-bottom when the identified action in the group is move-bottom.

22. The medium as set forth in claim 16 wherein the one or more filtering rules comprises removing remove-element, set-attribute if name parameter of following action is equal to the name parameter of the matching action, and remove-attribute if name parameter of following action is equal to the name parameter of the matching action when the identified action in the group is set-attribute.

23. The medium as set forth in claim 16 wherein the one or more filtering rules comprises removing remove-element, set-attribute if name parameter of following action is equal to the name parameter of the matching action and remove-attribute if name parameter of following action is equal to the name parameter of the matching action when the identified action in the group is remove-attribute.

24. The medium as set forth in claim 16 wherein the one or more filtering rules comprises removing remove-element, set-content, and append-content when the identified action in the group is set-content.

25. The medium as set forth in claim 16 wherein the one or more filtering rules comprises remvoing remove-element, set-content, and append-content when the identified action in the group is append-content.

26. The medium as set forth in claim 14 wherein one or more of the actions can comprise one or more of remove-element replace-element, linearize-table, set-meta-category, move-bottom, set-attribute, remove-attribute, set-content, and append-content.

27. A web proxy apparatus comprising:

one or more processors;

a memory coupled to the one or more processors which are configured to execute programmed instructions stored in the memory comprising:

identifying one or more rules matching one or more elements in an HTML document;

identifying an action associated with each of the identified one or more rules;

filtering the identified actions based on one or more filtering rules when two or more of the identified actions have a match; and

applying the remaining identified actions after the filtering to transform the one or more matching elements in the HTML document

providing the transformed HTML document.

28. The apparatus as set forth in claim 27 wherein the one or more processors is further configured to execute programmed instructions stored in the memory further comprising determining when two or more of the identified actions have a match.

29. The apparatus as set forth in claim 27 wherein the one or more processors is further configured to execute programmed instructions stored in the memory for the filtering further comprising:

grouping each of the identified actions which have a match together; and

removing one or more of the actions in each of the groups based on the one or more filtering rules.

30. The apparatus as set forth in claim 29 wherein the one or more filtering rules comprises removing remove-element, replace-element, linearize-table, set-meta-category, move-bottom, set-attribute, remove-attribute, set-content, and append-content when the identified action in the group is remove-element.

31. The apparatus as set forth in claim 30 wherein the one or more filtering rules comprises removing remove-element, replace-element, and linearize-table when the identified action in the group is replace-element.

32. The apparatus as set forth in claim 30 wherein the one or more filtering rules comprises removing remove-element, replace-element,

linearize-table, set-meta-category, set-attribute, remove-attribute, set-content, append-content and removing all preceding set-meta-category, set-attribute, remove-attribute, set-content, and append-content when the identified action in the group is linearize-table.

33. The apparatus as set forth in claim 30 wherein the one or more filtering rules comprises removing remove-element and set-meta-category when the identified action in the group is set-meta-category.

34. The apparatus as set forth in claim 30 wherein the one or more filtering rules comprises removing remove-element and move-bottom when the identified action in the group is move-bottom.

35. The apparatus as set forth in claim 30 wherein the one or more filtering rules comprises removing remove-element, set-attribute if name parameter of following action is equal to the name parameter of the matching action, and remove-attribute if name parameter of following action is equal to the name parameter of the matching action when the identified action in the group is set-attribute.

36. The apparatus as set forth in claim 30 wherein the one or more filtering rules comprises removing remove-element, set-attribute if name parameter of following action is equal to the name parameter of the matching action and remove-attribute if name parameter of following action is equal to the name parameter of the matching action when the identified action in the group is remove-attribute.

37. The apparatus as set forth in claim 30 wherein the one or more filtering rules comprises removing remove-element, set-content, and append-content when the identified action in the group is set-content.

38. The apparatus as set forth in claim 30 wherein the one or more filtering rules comprises remvoing remove-element, set-content, and append-content when the identified action in the group is append-content.

39. The apparatus as set forth in claim 27 wherein one or more of the actions can comprise one or more of remove-element replace-element, linearize-table, set-meta-category, move-bottom, set-attribute, remove-attribute, set-content, and append-content.

```
<?xml version="1.0" encoding= "UTF-8"?>
<xsl:stylesheet xnlns:xsl-"http://www.w3.org/1999/XSL/Transform" version="1.0")

    <xsl:template match="a[@target='_blank' and img]">
        <a href="{myext:normalize-url(@href)}" class="external">,
            <xsl:value-of select="img/@alt"/>
        </a>
    </xsl:template>

    <xsl:template match="a[@target='_blank']">
        <a href="{myext:normalize-url(@href)}" class="external">,
            <xsl:apply-templates/>
        </a>
    </xsl:template>

    <xsl:template match="a[img]">
        <a href="{myext:normalize-url(@href)}" class="internal">,
            <xst:value-of select="img/@alt"/>
        </a>
    </xsl:template>

    <xsl:template match="a">
        <a hret="{myext:normalize-url(@href)}" class=internal">,
            <xsl:apply-templates/>
        </a>
    </xsl:template>

    <xsl:template match="*">
        <xsl:copy>
            <xsl:copy-of select="@*"/>
            </xsl:apply-templates/>
        </xsl:copy>
    </xsl:template>

</xsl:stylesheet>
```

FIG. 1

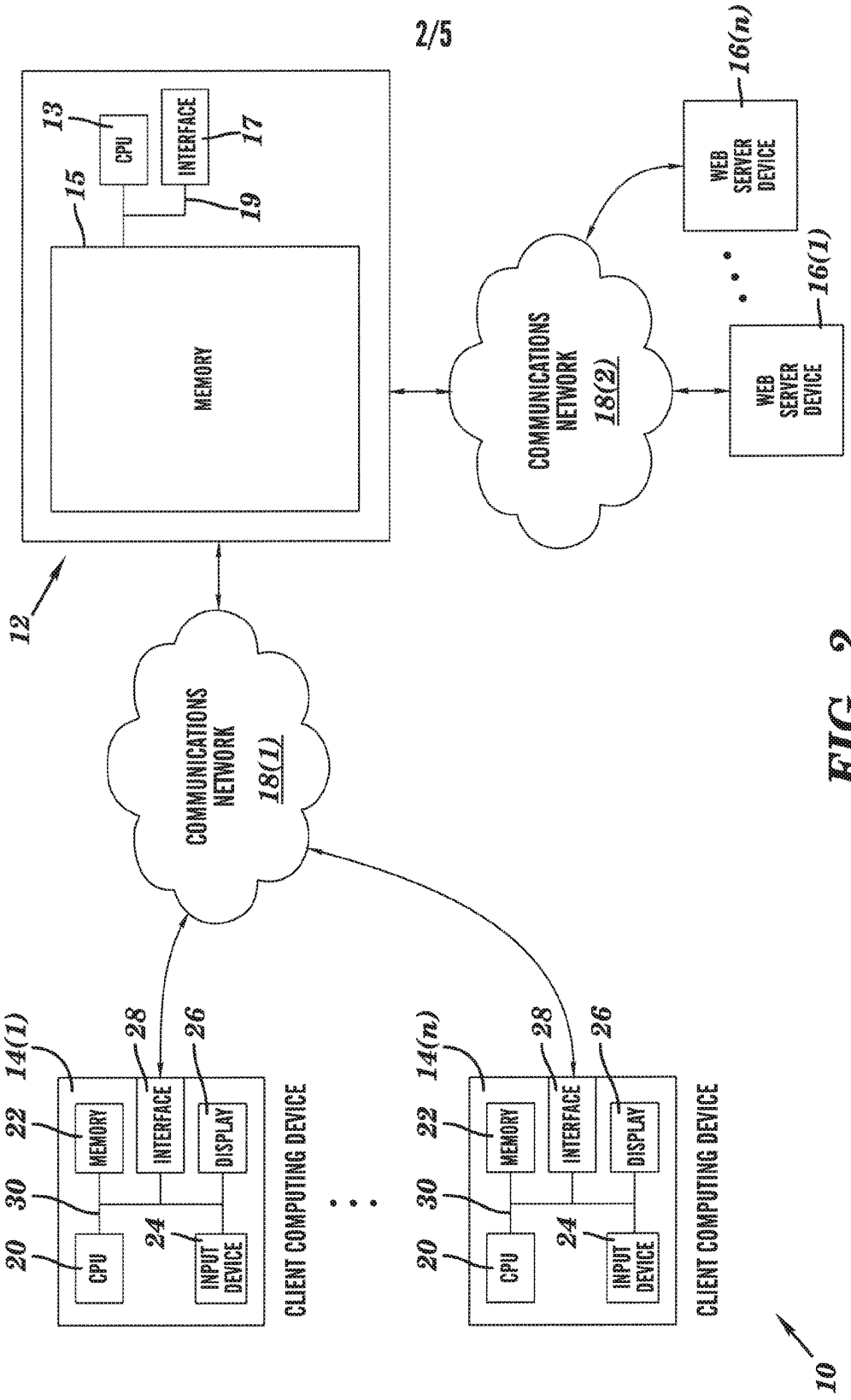

12
13
CPU
INTERFACE
17
15
19
MEMORY
COMMUNICATIONS NETWORK 18(2)
16(n)
WEB SERVER DEVICE
16(1)
WEB SERVER DEVICE
COMMUNICATIONS NETWORK 18(1)
14(1)
28
26
22
MEMORY
INTERFACE
DISPLAY
30
20
CPU
24
INPUT DEVICE
CLIENT COMPUTING DEVICE
14(n)
28
26
22
MEMORY
INTERFACE
DISPLAY
30
20
CPU
24
INPUT DEVICE
CLIENT COMPUTING DEVICE
10

FIG. 2

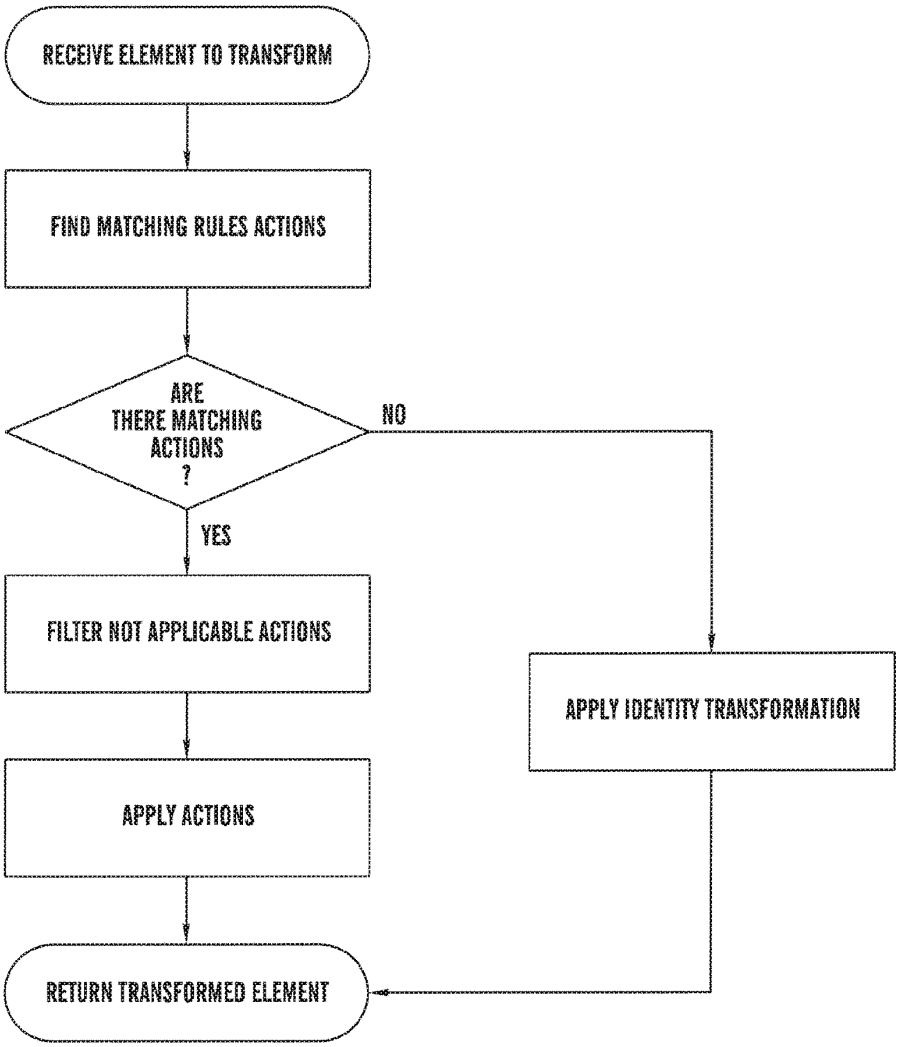

RECEIVE ELEMENT TO TRANSFORM
FIND MATCHING RULES ACTIONS
ARE THERE MATCHING ACTIONS ?
NO
YES
FILTER NOT APPLICABLE ACTIONS
APPLY ACTIONS
APPLY IDENTITY TRANSFORMATION
RETURN TRANSFORMED ELEMENT

FIG. 3

```
<rules>

    <rule for = "a">
        <match>@target='_blank'</match>
        <execute>
            <set-attribute name = "class">external</set-attribute>
            <remove-attribute name = "target"/>
        </execute>
    </rule>

    <rule for = "a">
        <match>img</match>
        <execute>
            <set-content>
                <xsl:value-of, select = "img/@alt"/>
            </set-content>
        </execute>
    </rule>

    <rule for ="a">
        <match>true()</match>
        <execute>
            <set-attribute name = "class">internal</set-attribute>
            <set-attribute name = "href" select = "myext:normalize-url(@href)"/>
        </execute>
    </rule>

</rules>
```

FIG. 4

```
<?xml version = "1.0" encoding = "UTF-8?>
<xsl:stylesheet xmln:xsl="http://ww.w3.org/1999/XSL/Transform" version="1.0"

    <xsl:template match="a">
        <xsl:copy>
        <xsl:attribute name="href"><xsl:value-of select-"myext:normalize-
        url(@href)"/></xsl:attribute>
        <xsl:choose>
            <xsl:when test = "@target='_blank">
                <xsl:attribute name="class">external</xsl:attribute>
            </xsl:when>
            <xsl:otherwise>
                <xsl:attribute name = "class">internal</xsl:attribute>
                <xsl:if test="@target">
                    <xsl:copy-of select="@target"/>
                </xsl:if>
            </xsl:otherwise>
        </xsl:choose>
        <xsl:choose>
            <xsl:when test = "img">
                <xsl:value-of select = "img/@alt"/>
            </xsl:when>
            <xsl:otherwise>
                <xsl:apply-templates/>
            </xsl:otherwise>
        </xsl:choose>
    </xsl:copy>
</xsl:template>

    <xsl:template match-"*">
        <xsl:copy>
            <xsl:copy-of select="@*"/>
            <xsl:apply-templates/>
        </xsl:copy>
    </xsl:template>

</xsl:stylesheet>
```

FIG. 5

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Veröffentlichungsnummer:
(11) Publication number: EP 2 616 962 A0
(11) Numéro de publication:

Internationale Anmeldung veröffentlicht durch die Weltorganisation für geistiges Eigentum unter der Nummer:

WO 2012/036833 (Art. 153(3) EPÜ).

International application published by the World Intellectual Property Organization under number:

WO 2012/036833 (Art. 153(3) EPC).

Demande internationale publiée par l'Organisation Mondiale de la Propriété Intellectuelle sous le numéro:

WO 2012/036833 (art. 153(3) CBE).

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
22 March 2012 (22.03.2012)

PCT

(10) International Publication Number
WO 2012/036833 A2

(51) International Patent Classification:
G06F 17/21 (2006.01) *G06F 17/00* (2006.01)

(21) International Application Number:
PCT/US2011/048060

(22) International Filing Date:
17 August 2011 (17.08.2011)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
12/807,766 14 September 2010 (14.09.2010) US

(71) Applicant *(for all designated States except US)*: **US-ABLENET, INC.** [US/US]; 28 W. 23rd St., 6th Floor, New York, NY 10010 (US).

(72) Inventor: **SCODA, Enrico**; Via Cividina 416/3, I-33035 Martignacco (UD) (IT).

(74) Agents: **LEINBERG, Gunnar, G.** et al.; LeClairRyan, P.C., 290 Linden Oaks, Rochester, NY 14625 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

---- *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

(54) Title: METHODS FOR EXTENDING A DOCUMENT TRANSFORMATION SERVER TO PROCESS MULTIPLE DOCUMENTS FROM MULTIPLE SITES AND DEVICES THEREOF

(57) Abstract: A method, computer readable medium and device that extends a document transformation server to process multiple documents from multiple websites includes obtaining with a document transformation server a document to process based on at least a URL of the document to process in an HTTP request URL. An XML source representing the document to process is generated with the document transformation server. One or more XML processors associated with at least a portion of the HTTP request URL from a plurality of stored associations are identified with the document transformation server. The XML source is transformed by the document transformation server with each of the identified one identified XML processors. The generated document is provided in an HTTP response to the HTTP request URL by the document transformation server.

WO 2012/036833 A2

METHODS FOR EXTENDING A DOCUMENT TRANSFORMATION SERVER TO PROCESS MULTIPLE DOCUMENTS FROM MULTIPLE SITES AND DEVICES THEREOF

FIELD

[0001] This technology generally relates to methods and devices for transforming and rendering documents and, more particularly, to methods for extending a document transformation server to process multiple documents from multiple sites and devices thereof.

BACKGROUND

[0002] The introduction of eXtensible Markup Language (XML) and the EXtensibleStylesheet Language (XSL) specifications has provided an easy way to transform documents between various formats. This functionality has been included into Web development frameworks enabling them to automatically transform an XML file into a document with different format, such as HTML or XHTML, and integrate the original data with a graphic layout and user interface parts.

[0003] This transformation process is based on the following basic scheme. Each URL on a website is associated to a single XML file and to one or more XSL style sheets. The associated XML file and XSL style sheet(s) include the transformation rules used to customize the webpage into a desired format.

[0004] An example of a prior art method for transforming and rendering a single document is illustrated in FIG. 2. In step 100, a server receives an HTTP request. In step 102, the server extracts the parts from the PATH and QUERY fields in the HTTP request to identify and obtain the document to be processed.

[0005] In step 104, the server determines whether the extracted parts in the PATH and QUERY fields of the HTTP request match an XML source for the document to be processed. If in step 104 the server determines the extracted parts in the PATH and QUERY fields of the HTTP request do not have an association with an XML source, then the No branch is taken to step 106. In step 106, the

server generates an error page document. In step 108 the server returns the generated error page document in an HTTP response to the HTTP request.

[0006] If in step 104 the server determines the extracted parts in the PATH and QUERY fields of the HTTP request do have an association with an XML source for the document to be processed, then the Yes branch is taken to step 110. In step 110, the server obtains the document to be processed from the XML source and generates an XML document.

[0007] In step 112, the server determines whether the extracted parts in the PATH and QUERY fields of the HTTP request match one of one or more stored XML processors. If in step 112 the server determines the extracted parts in the PATH and QUERY fields of the HTTP request do not match one of one or more stored XML processors, then the No branch is taken to step 114. In step 114, the server converts the XML document into a desired format. Next, in step 108 the returns the converted XML document in an HTTP response to the HTTP request.

[0008] If in step 112 the server determines the extracted parts in the PATH and QUERY fields of the HTTP request do match one or more stored of the XML processors, then the Yes branch is taken to step 116. In step 116, the server runs the one or more matching XML processors on the previously generated XML document. In step 114, the server converts the processed XML document into a desired format. Next, in step 108 the server returns the converted XML document in an HTTP response to the HTTP request.

[0009] Referring to FIG. 3, an exemplary prior art fragment of a sitemap.xmap document which is a configuration file of Apache Cocoon is illustrated. This fragment has two sets of prior art instructions to process documents.

[00010] The first set of instructions executed starts with XML element <map:match pattern=""> and matches the document with an empty component in the PATH field that is the home page of a site (for example www.sample.com). The document is obtained by loading the XML file "welcome.xml" as the

document or other file to be processed, then applying the XSL transformation described in welcome.xsl using XHTML format to return it to the browser because of the <map:serialize> instruction.

[00011] The second set of instructions matches an entire set of web site pages: all pages whose PATH field starts with "static-site/" and ends with a name followed by ".xml" extension (for example, www.sample.com/static-site/news.xml). The first instruction loads the corresponding documents or other files from the xdocs directory (for example xdocs/news.xml). The second instruction applies an XSL transformation using one or more identified matching XML processors that transforms the original xml document into an HTML document adding web site user interface (web site navigation links, site logo etc). The last instruction returns the document to the browser in HTML format.

[00012] Accordingly, as illustrated and described above, the prior art provides methods for transforming and rendering documents. Unfortunately, these transformation rules currently can not be utilized in an effective and efficient manner by a document transformation server in a single application instance to transform multiple documents from multiple websites.

SUMMARY

[00013] A method for extending a document transformation server to process multiple documents from multiple websites includes obtaining with a document transformation server a document to process based on at least a URL of the document to process in an HTTP request URL. An XML source representing the document to process is generated with the document transformation server. One or more XML processors associated with at least a portion of the HTTP request URL from a plurality of stored associations are identified with the document transformation server. The XML source is transformed by the document transformation server with each of the identified one identified XML processors. The generated document is provided in an HTTP response to the HTTP request URL by the document transformation server.

[00014] A non-transitory computer readable medium having stored thereon instructions for extending a document transformation server to process multiple documents from multiple websites comprising machine executable code which when executed by at least one processor, causes the processor to perform steps including obtaining a document to process based on at least a URL of the document to process in an HTTP request URL. An XML source representing the obtained document to process is generated and one or more XML processors that are associated with at least a portion of the HTTP request URL from a plurality of stored associations are identified. The XML source is transformed with each of the identified one identified XML processors and the generated document is provided in an HTTP response to the HTTP request URL.

[00015] A document transformation apparatus includes one or more processors and a memory coupled to the one or more processors which are configured to execute programmed instructions stored in the memory obtaining a document to process based on at least a URL of the document to process in an HTTP request URL. An XML source representing the obtained document to process is generated and one or more XML processors that are associated with at least a portion of the HTTP request URL from a plurality of stored associations are identified. The XML source is transformed with each of the identified one identified XML processors and the generated document is provided in an HTTP response to the HTTP request URL.

[00016] This technology provides a number of advantages including providing a method, computer readable medium and an apparatus that extends a document transformation server to process multiple documents from multiple sites. With this technology, an exemplary document transformation server can efficiently manage the transformation and rendering of documents using a single web application instance.

BRIEF DESCRIPTION OF THE DRAWINGS

[00017] FIG. 1 is a block diagram of an exemplary environment with an exemplary document transformation server configured to process multiple documents from multiple sites;

[00018] FIG. 2 is a flow chart of a prior art method for transforming and rendering a document;

[00019] FIG. 3 is a prior art fragment of a configuration file;

[00020] FIG. 4 is an exemplary flow chart of a method for extending a document transformation server to process multiple documents from multiple sites;

[00021] FIG. 5 is an exemplary flow chart of a method for obtaining a mapping file; and

[00022] FIG. 6 is an exemplary implementation of a mapping file.

DETAILED DESCRIPTION

[00023] An exemplary environment 10 with a document transformation server 12 configured to process multiple documents from multiple websites is illustrated in FIG. 1. The exemplary environment 10 includes the document transformation server or apparatus 12, client devices 14(1)-14(n), web server devices 16(1)-16(n), and communication networks 18(1)-18(2), although other numbers and types of systems, devices, and/or elements in other configurations and environments with other communication network topologies can be used. This technology provides a number of advantages including providing a method, computer readable medium and an apparatus that extends a document transformation server to process multiple documents from multiple sites.

[00024] Referring more specifically to FIG. 1, the document transformation server 12 includes a central processing unit (CPU) or processor 13, a memory 15, and an interface system 17 which are coupled together by a bus 19

or other link, although other numbers and types of components, parts, devices, systems, and elements in other configurations and locations can be used. Additionally, other types and numbers of proxy servers or other computing devices could be configured to execute the exemplary methods illustrated and described herein. The processor 13 in the document transformation server 12 executes a program of stored instructions one or more aspects of the present invention as described and illustrated by way of the embodiments herein, although the processor could execute other numbers and types of programmed instructions.

[00025] The memory 15 in the document transformation server 12 stores these programmed instructions for one or more aspects of the present invention as described and illustrated herein, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, DVD ROM, or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to the processor 13, can be used for the memory 15 in the document transformation server 12.

[00026] The interface system 17 in the document transformation server 12 is used to operatively couple and communicate between the document transformation server 12 and the client devices 14(1)-14(n) and the web server devices 16(1)-16(n) via the communication networks 18(1) and 18(2), although other types and numbers of communication networks with other types and numbers of connections and configurations can be used. By way of example only, the communication networks 18(1) and 18(2) can use TCP/IP over Ethernet and industry-standard protocols, including HTTP, HTTPS, WAP, and SOAP, although other types and numbers of communication networks, such as a direct connection, a local area network, a wide area network, modems and phone lines, e-mail, and wireless and hardwire communication technology, each having their own communications protocols, can be used.

[00027] Each of the client devices 14(1)-14(n) enables a user to request, get and interact with documents and other files from one or more web sites hosted by the web server devices 16(1)-16(n) through the document transformation server 12 via one or more communication networks, although one or more of the client devices 14(1)-14(n) could access content and utilize other types and numbers of applications from other sources and could provide a wide variety of other functions for the user. Although multiple client devices 14(1)-14(n) are shown, other numbers and types of user computing systems could be used.

[00028] Each of client devices 14(1)-14(n) in this example is a computing device that includes a central processing unit (CPU) or processor 20, a memory 22, user input device 24, a display 26, and an interface system 28, and which are coupled together by a bus 30 or other link, although one or more of client devices 14(1)-14(n) can include other numbers and types of components, parts, devices, systems, and elements in other configurations. The processor 20 in each of client devices 14(1)-14(n) executes a program of stored instructions for one or more aspects of the present invention as described and illustrated herein, although the processor could execute other numbers and types of programmed instructions.

[00029] The memory 22 in each of the client devices 14(1)-14(n) stores these programmed instructions for one or more aspects of the present invention as described and illustrated herein, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to processor 20 can be used for the memory 22 in each of the client devices 14(1)-14(n.

[00030] The user input device 24 in each of the client devices 14(1)-14(n) is used to input selections and other data, although the user input device could provide other functions and interact with other elements. The user input device

can include keypads, touch screens, and/or vocal input processing systems although other types and numbers of user input devices can be used.

[00031] The display 26 in each of the client devices 14(1)-14(n) is used to show data and information to the user, such as a website page by way of example only. The display in each of the client devices 14(1)-14(n) is a computer screen display, although other types and numbers of displays could be used depending on the particular type of client device.

[00032] The interface system 28 in each of the client devices 14(1)-14(n) is used to operatively couple and communicate between the client devices 14(1)-14(n) and the document transformation server 12 and web server devices 16(1)-16(n) over the communication networks 18(1) and 18(2), although other types and numbers of communication networks with other types and numbers of connections and configurations can be used.

[00033] The web server devices 16(1)-16(n) provide one or more pages from one or more web sites for use by one or more of the client devices 14(1)-14(n) via the document transformation server 12, although the web server devices 16(1)-16(n) can provide other numbers and types of applications and/or content and can have provide other numbers and types of functions. Although web server devices 16(1)-16(n) are shown for ease of illustration and discussion, other numbers and types of web server systems and devices can be used.

[00034] Each of the web server devices 16(1)-16(n) include a central processing unit (CPU) or processor, a memory, and an interface system which are coupled together by a bus or other link, although each of the web server devices 16(1)-16(n) could have other numbers and types of components, parts, devices, systems, and elements in other configurations and locations can be used. The processor in each of the web server devices 16(1)-16(n) executes a program of stored instructions one or more aspects of the present invention as described and illustrated by way of the embodiments herein, although the processor could execute other numbers and types of programmed instructions.

[00035] The memory in each of the web server devices 16(1)-16(n) stores these programmed instructions for one or more aspects of the present invention as described and illustrated by way of the embodiments, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, DVD ROM, or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to the processor, can be used for the memory in each of the web server devices 16(1)-16(n).

[00036] The interface system in each of the web server devices 16(1)-16(n) is used to operatively couple and communicate between the web server devices 16(1)-16(n) and the document transformation server 12 and the client devices 14(1)-14(n) via communication networks 18(1) and 18(2), although other types and numbers of communication networks with other types and numbers of connections and configurations can be used.

[00037] Although embodiments of the document transformation server 12, the client devices 14(1)-14(n), and the web server devices 16(1)-16(n), are described and illustrated herein, each of the client devices 14(1)-14(n), the document transformation server 12, and the web server devices 16(1)-16(n), can be implemented on any suitable computer system or computing device. It is to be understood that the devices and systems of the embodiments described herein are for exemplary purposes, as many variations of the specific hardware and software used to implement the embodiments are possible, as will be appreciated by those skilled in the relevant art(s).

[00038] Furthermore, each of the systems of the embodiments may be conveniently implemented using one or more general purpose computer systems, microprocessors, digital signal processors, and micro-controllers, programmed according to the teachings of the embodiments, as described and illustrated herein, and as will be appreciated by those ordinary skill in the art.

[00039] In addition, two or more computing systems or devices can be substituted for any one of the systems in any embodiment of the embodiments. Accordingly, principles and advantages of distributed processing, such as redundancy and replication also can be implemented, as desired, to increase the robustness and performance of the devices and systems of the embodiments. The embodiments may also be implemented on computer system or systems that extend across any suitable network using any suitable interface mechanisms and communications technologies, including by way of example only telecommunications in any suitable form (e.g., voice and modem), wireless communications media, wireless communications networks, cellular communications networks, G3 communications networks, Public Switched Telephone Network (PSTNs), Packet Data Networks (PDNs), the Internet, intranets, and combinations thereof.

[00040] The embodiments may also be embodied as non-transitory computer readable medium having instructions stored thereon for one or more aspects of the present invention as described and illustrated by way of the embodiments herein, as described herein, which when executed by a processor, cause the processor to carry out the steps necessary to implement the methods of the embodiments, as described and illustrated herein.

[00041] An exemplary method for extending a document transformation server 12 to process multiple documents from multiple sites, such as web server devices 16(1)-16(n), in an exemplary environment 10 will now be described with reference to FIGS. 1 and 4-6. Referring more specifically to FIG. 4, in step 200 the document transformation server 12 receives an HTTP request whose url contains the URI of the document to process, from one of the client computing devices 14(1)-14(n), although other types of requests could be received and from other types of devices. Optionally, if the URI of the document to process is encoded, the document transformation server 12 will decode the URI of the document to process, although other manners for processing an encoded URI or other encoded portion of a URI HTTP request could be used. Since manners for encoding and decoding are well known to those of ordinary skill in the art, they will not be described in detail here. By way of example only, consider an HTTP

request from one of the client computing devices 14(1)-14(n) where the URL to be processed is embedded in the PATH field of the HTTP request url: http://processor.com/app/www.acme.com/anyproduct.html. The transformer server applications is identified by "processor.com/app"; the address www.acme.com/anyproduct.html is the resource to be processed.

[00042] In step 202, the document transformation server 12 extracts the domain in the URI of the document to process, although other fields of the URI could be extracted and used.

[00043] In step 204, the document transformation server 12 determines whether the transformation of the website in the URI of the document to process is allowed. If in step 204, the document transformation server 12 determines the transformation of the website in the URI of the document to process is not allowed, then the No branch is taken to step 206. In step 206, the document transformation server 12 generates an error page document. In step 208 the document transformation server 12 returns the generated error page document in an HTTP response to the one of the client computing devices 14(1)-14(n) that provided the HTTP request in this example.

[00044] If in step 204, the document transformation server 12 determines the transformation of the website in the URI of the document to process is allowed, then the Yes branch is taken to step 210. In step 210, the document transformation server 12 downloads the original document or other resource linked by the given URL and transforms the downloaded document into an XML source, although other manners for processing the obtained document or documents can be used. In step 212, document transformation server 12 looks for a mapping or configuration file for the directory that matches the extracted domain. An exemplary method for locating a mapping file from manifest.xml is described herein with reference to FIG. 5.

[00045] In step 214, the document transformation server 12 determines whether the mapping file for the extracted domain exists. If in step 214 the document transformation server 12 determines the mapping file does not exist,

then the No branch is taken to step 216. In step 216, the document transformation server 12 converts the document into the desired format. Next, in step 208 the document transformation server 12 returns the converted document in an HTTP response to the one of the client computing devices 14(1)-14(n) that provided the HTTP request.

[00046] If in step 214 the document transformation server 12 determines the mapping file for the extracted domain does exist, then the Yes branch is taken to step 218. In step 218, the document transformation server 12 use the mapping file to identify all of the XSL processors linked to the corresponding URL and applies the identified XSL processors to the document to be processed. Optionally the document transformation server 12 may apply one or more best matching rules to filter the identified one or more XML processors in the matching directory. By way of example only, one best matching rule is to take the XML processor with the longest matching pattern string, although other types and numbers of best matching rules can be used.

[00047] In step 216, the document transformation server 12 converts the processed document into the desired format. Next, in step 208 the document transformation server returns the converted document in an HTTP response to the one of the client computing devices 14(1)-14(n) that provided the HTTP request.

[00048] Referring to FIG. 5, an exemplary method for obtaining a mapping file with the document transformation server 12 is illustrated. In step 300, the document transformation server 12 receives a URI to match from one of the client computing devices 14(1)-14(n), although other types of requests could be received and from other types of devices. In step 302, the document transformation server 12 extracts from the URI the AUTHORITY portion corresponding to the associated domain and port, for example "ssl.example.com:8443", although other types and numbers of fields could be extracted.

[00049] In this example, if the port is either 80 or 443 (default values for HTTP and HTTPS respectively) the string representing the authority corresponds to the domain portion of the URL. However, if the port does not correspond to a

standard value, the ':' character is substituted with '_' in order to avoid potential problems in the file system (i.e. ssl.example.com_8443). The string representing the authority is further stripped by the document transformation server 12 removing the 'www.' from the extracted string.

[00050] In step 304, the document transformation server 12 determines whether there is a directory with the same name as the extracted domain and containing the mapping file, although other manners for determining matches with other portions of the domain can be used. By way of example only, consider the following directories: (1) "www.acme.com"; (2) "_.example.com"; and (3) "ssl.example.com". The www.acme.com directory will match all the requests for www.acme.com and acme.com domains. With this technology, the "_.example.com" directory will match all the requests for domains ending with ".example.com" that do not have more specific matching directories. Accordingly, "_.example.com" will match "products.example.com", but it will not match "ssl.example.com" because an "ssl.example.com" directory exists. The "_" character in front of ".example.com" is used to make the directory visible when using file systems that use "." characters in front of files to hide them to users.

[00051] If in step 304 the document transformation server 12 determines there is no directory with the same name as the extracted domain, then the No branch is taken to step 306. In step 306, the document transformation server 12 determines whether the domain contains a valid subdomain. If in step 306 the document transformation server 12 determines the domain does contain a valid subdomain, then the Yes branch is taken to step 308. In step 308, the document transformation server 12 extracts the next level subdomain name and returns to step 304 as described earlier.

[00052] If in step 304 the document transformation server 12 determines there is a directory with the same name as the extracted domain and containing the mapping file, then the Yes branch is taken to step 310, although other manners for determining matches with other portions of the domain can be used. In another example, the document transformation server 12 may also use another optional mapping file that stores known aliases of domains. When a known alias of a

domain is identified using this optional mapping file, the document transformation server 12 can use the same one or more stored directories for the known and identified aliases. As a result, this reduces the number of stored directories when handling domains with multiple aliases (i.e. defined with DNS records of type CNAME).

[00053] In step 310, the document transformation server 12 extracts the parts from the PATH and QUERY fields in the URI, although the parts can be extracted from other types and numbers of fields in the URI.

[00054] In step 312, the document transformation server 12 determines whether the extracted parts in the PATH and QUERY fields of the URI match or otherwise have an association with one of one or more XML processors in the identified mapping file, although other types of associations between other types and numbers of parts can be used. If in step 312 the document transformation server 12 determines the extracted parts in the PATH and QUERY fields of the URI match or otherwise have an association with one of one or more XML processors, then the Yes branch is taken to step 314 where the matching XML processors are collected. Next, in step 316, the document transformation server 12 returns the collected XML processors to the requesting one of the client computing devices 14(1)-14(n).

[00055] If in step 312 the document transformation server 12 determines the extracted parts in the PATH and QUERY fields of the URI do not match one of one or more XML processors, then the No branch is taken to step 316 where no matching XML processors are returned to the requesting one of the client computing devices 14(1)-14(n).

[00056] Referring to FIG. 6, an exemplary implementation of the mapping file "manifest.xml" is illustrated. Each map element identifies an XSL file (inside file element) and a substring to match a URL PATH component (inside the PATH element). The map element with the longest string matching the PATH component will be used to setup the XSL transformer identified by the level attribute.

[00057] In these examples, two XSL transformers are defined: level 1 transformer; and level 2 transformer. If the request URL has the PATH field containing "/content/news/july.html", then the original site web page is downloaded (for example, www.sample.com/content/news/july.html). Next, the level 1 transformer uses the news.xsl stylesheet document to transform the downloaded page. The level 2 transformer uses content2.xsl stylesheet document to transform the document obtained from the level 1 transformer.

[00058] If the request URL has the PATH field containing "/content/privacy.html", then the original site web page is downloaded (for example www.sample.com/content/privacy.html). Next, the level 1 transformer uses the content.xsl stylesheet document to transform the downloaded page. Next, the level 2 transformer uses the content2.xsl stylesheet document to transform the document obtained from the level 1 transformer.

[00059] Accordingly, as illustrated and described herein this technology provides a number of advantages including providing a method, computer readable medium and an apparatus that extends a document transformation server to process multiple documents from multiple sites. One of the advantages of this technology is that a single association or match between a name of a directory and at least a portion of a domain name or other part of a URI can be used for hundreds or thousands of URIs. As a result, only a few associations with directories or other stored tables can be enough for the document transformation server to transform the content of an entire website.

[00060] Having thus described the basic concept of the invention, it will be rather apparent to those skilled in the art that the foregoing detailed disclosure is intended to be presented by way of example only, and is not limiting. Various alterations, improvements, and modifications will occur and are intended to those skilled in the art, though not expressly stated herein. These alterations, improvements, and modifications are intended to be suggested hereby, and are within the spirit and scope of the invention. Additionally, the recited order of processing elements or sequences, or the use of numbers, letters, or other designations therefore, is not intended to limit the claimed processes to any order

except as may be specified in the claims. Accordingly, the invention is limited only by the following claims and equivalents thereto.

CLAIMS

What is claimed is:

1. A method for extending a document transformation server to process multiple documents from multiple websites, the method comprising:

obtaining with a document transformation server a document to process based on at least a URL of the document to process in an HTTP request URL;

generating with the document transformation server an XML source representing the document to process;

identifying with the document transformation server one or more XML processors associated with at least a portion of the HTTP request URL from a plurality of stored associations;

transforming with the document transformation server the XML source with each of the identified one identified XML processors; and

providing with the document transformation server the generated document in an HTTP response to the HTTP request URL.

2. The method as set forth in claim 1 wherein the obtaining the document to process is further based on one or more additional identifiers in the HTTP request URL.

3. The method as set forth in claim 2 wherein the one or more additional identifiers comprise at least one of an HTTP header and an HTTP POST parameter.

4. The method as set forth in claim 1 wherein the obtaining the document to process further comprises decoding with the document transformation server the obtained URL of the document to process prior to obtaining the document to process.

5. The method as set forth in claim 1 further comprising:

storing with the document transformation server the one or

more XML processors associated to at least a portion of a particular domain in a directory having a directory name equal to the at least a portion of the particular domain; and

storing with the document transformation server a mapping file between the one or more XML processor and one or more parts of one or more fields in an HTTP request URL with the stored directory.

6. The method as set forth in claim 5 wherein the one or more fields of the HTTP requests URL comprise one or more of a URL path, one or more query parameters, one or more post parameters, and one or more HTTP headers.

7. The method as set forth in claim 6 wherein the identifying with the document transformation server the one or more XML processors further comprises:

locating with the document transformation server the directory comprising one or more XML processors which has a directory name equal to at least a portion of the particular domain in the HTTP request URL;

opening with the document transformation server the mapping file with the located directory comprising the mapping file between the one or more XML processor and one or more parts of one or more fields in an HTTP request URL; and

identifying with the document transformation server the one or more XML processors in the located directory that match the one or more parts of the one or more fields in an HTTP request URL and filtering out any non-matching XML processors.

8. The method as set forth in claim 7 further comprising applying with the document transformation server one or more best match rules to the identified one or more XML processors, wherein the transforming with the document transformation server the XML source is based on the identified best matches of the identified one or more XML processors.

9. The method as set forth in claim 7 wherein the locating with the document transformation server the directory further comprises:

determining with the document transformation server when more than one directory has a directory name equal to at least a portion of the particular domain in the HTTP request URL; and

selecting with the document transformation server the directory which has a name which is the longest among the located directories as the located directory.

10. The method as set forth in claim 1 wherein each of the one or more XML processors comprises a set of one or more XSL transformers with transformation rules in one or more XSL files.

11. A non-transitory computer readable medium having stored thereon instructions for extending a document transformation server to process multiple documents from multiple websites comprising machine executable code which when executed by at least one processor, causes the processor to perform steps comprising:

obtaining a document to process based on at least a URL of the document to process in an HTTP request URL;

generating an XML source representing the document to process;

identifying one or more XML processors associated with at least a portion of the HTTP request URL from a plurality of stored associations;

transforming the XML source with each of the identified one identified XML processors; and

providing the generated document in an HTTP response to the HTTP request URL.

12. The medium as set forth in claim 11 wherein the obtaining the document to process is further based on one or more additional identifiers in the HTTP request URL.

13. The medium as set forth in claim 12 wherein the one or more additional identifiers comprise at least one of an HTTP header and an HTTP POST parameter.

14. The medium as set forth in claim 11 wherein the obtaining the document to process further comprises decoding the obtained URL of the document to process prior to obtaining the document to process.

15. The medium as set forth in claim 11 further comprising:

storing the one or more XML processors associated to at least a portion of a particular domain in a directory having a directory name equal to the at least a portion of the particular domain; and

storing a mapping file between the one or more XML processor and one or more parts of one or more fields in an HTTP request URL with the stored directory.

16. The medium as set forth in claim 15 wherein the one or more fields of the HTTP requests URL comprise one or more of a URL path, one or more query parameters, one or more post parameters, and one or more HTTP headers.

17. The medium as set forth in claim 16 wherein the identifying the one or more XML processors further comprises:

locating the directory comprising one or more XML processors which has a directory name equal to at least a portion of the particular domain in the HTTP request URL;

opening the mapping file with the located directory comprising the mapping file between the one or more XML processor and one or more parts of one or more fields in an HTTP request URL; and

identifying the one or more XML processors in the located directory that match the one or more parts of the one or more fields in an HTTP request URL and filtering out any non-matching XML processors.

18. The medium as set forth in claim 17 further comprising applying one or more best match rules to the identified one or more XML processors, wherein the transforming the XML source is based on the identified best matches of the identified one or more XML processors.

19. The medium as set forth in claim 17 wherein the locating the directory further comprises:

determining when more than one directory has a directory name equal to at least a portion of the particular domain in the HTTP request URL; and

selecting the directory which a name which is the longest among the located directories as the located directory.

20. The medium as set forth in claim 11 wherein each of the one or more XML processors comprises a set of one or more XSL transformers with transformation rules in one or more XSL files.

21. A document transformation apparatus comprising:

one or more processors;

a memory coupled to the one or more processors which are configured to execute programmed instructions stored in the memory comprising:

obtaining a document to process based on at least a URL of the document to process in an HTTP request URL;

generating an XML source representing the document to process;

identifying one or more XML processors associated with at least a portion of the HTTP request URL from a plurality of stored associations;

transforming the XML source with each of the identified one identified XML processors; and

providing the generated document in an HTTP response to the HTTP request URL.

22. The apparatus as set forth in claim 21 wherein the one or more processors is further configured to execute programmed instructions stored in the memory for the obtaining the document to process further comprising obtaining the document to process further based on one or more additional identifiers in the HTTP request URL.

23. The apparatus as set forth in claim 22 wherein the one or more additional identifiers comprise at least one of an HTTP header and an HTTP POST parameter.

24. The apparatus as set forth in claim 21 wherein the one or more processors is further configured to execute programmed instructions stored in the memory further comprising decoding the obtained URL of the document to process prior to obtaining the document to process.

25. The apparatus as set forth in claim 21 wherein the one or more processors is further configured to execute programmed instructions stored in the memory further comprising:

storing the one or more XML processors associated to at least a portion of a particular domain in a directory having a directory name equal to the at least a portion of the particular domain; and

storing a mapping file between the one or more XML processor and one or more parts of one or more fields in an HTTP request URL with the stored directory.

26. The apparatus as set forth in claim 25 wherein the one or more fields of the HTTP requests URL comprise one or more of a URL path, one or more query parameters, one or more post parameters, and one or more HTTP headers.

27. The apparatus as set forth in claim 26 wherein the one or more processors is further configured to execute programmed instructions stored in the memory for the identifying the one or more XML processors further

comprising:

locating the directory comprising one or more XML processors which has a directory name equal to at least a portion of the particular domain in the HTTP request URL;

opening the mapping file with the located directory comprising the mapping file between the one or more XML processor and one or more parts of one or more fields in an HTTP request URL; and

identifying the one or more XML processors in the located directory that match the one or more parts of the one or more fields in an HTTP request URL and filtering out any non-matching XML processors.

28. The apparatus as set forth in claim 27 wherein the one or more processors is further configured to execute programmed instructions stored in the memory further comprising applying one or more best match rules to the identified one or more XML processors, wherein the transforming the XML source is based on the identified best matches of the identified one or more XML processors.

29. The apparatus as set forth in claim 27 wherein the one or more processors is further configured to execute programmed instructions stored in the memory for the locating the directory further comprising:

determining when more than one directory has a directory name equal to at least a portion of the particular domain in the HTTP request URL; and

selecting the directory which has a name which is the longest among the located directories as the located directory.

30. The apparatus as set forth in claim 21 wherein each of the one or more XML processors comprises a set of one or more XSL transformers with transformation rules in one or more XSL files.

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 2 638 681 B1

(12) EUROPEAN PATENT SPECIFICATION

(45) Date of publication and mention of the grant of the patent:
19.07.2017 Bulletin 2017/29

(21) Application number: **11839018.6**

(22) Date of filing: **09.11.2011**

(51) Int Cl.:
H04L 29/06 *(2006.01)* ***G06F 15/16*** *(2006.01)*
H04L 29/08 *(2006.01)*

(86) International application number:
PCT/US2011/059992

(87) International publication number:
WO 2012/064856 (18.05.2012 Gazette 2012/20)

(54) **METHODS FOR REDUCING LATENCY IN NETWORK CONNECTIONS AND SYSTEMS THEREOF**

VERFAHREN ZUR VERRINGERUNG DER LATENZ IN NETZWERKVERBINDUNGEN UND SYSTEME DAFÜR

PROCÉDÉS DE RÉDUCTION DE LATENCE DANS DES CONNEXIONS DE RÉSEAU ET SYSTÈMES CORRESPONDANTS

(84) Designated Contracting States:
AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO RS SE SI SK SM TR

(30) Priority: **09.11.2010 US 927169**

(43) Date of publication of application:
18.09.2013 Bulletin 2013/38

(73) Proprietor: **Usablenet Inc.**
New York, NY 10019 (US)

(72) Inventor: **SCODA, Enrico**
Martignacco (ud) (IT)

(74) Representative: **Cinquantini, Bruno et al**
Notarbartolo & Gervasi S.p.A.
Corso di Porta Vittoria, 9
20122 Milano (IT)

(56) References cited:
US-A1- 2004 215 717
US-A1- 2005 060 410
US-A1- 2005 060 410
US-A1- 2008 172 488
US-A1- 2008 172 488
US-A1- 2009 193 129
US-A1- 2009 271 497
US-A1- 2010 250 757
US-B1- 6 992 983
US-B1- 6 992 983

Printed by Jouve, 75001 PARIS (FR)

EP 2 638 681 B1

Description

FIELD

[0001] The embodiments of the present invention generally relate to proxy server devices and, more particularly, to methods for reducing latency in network connections utilizing proxy server devices, and systems thereof.

BACKGROUND

[0002] When a client device connects to a server (*e.g.*, a web server, or a content server) to get a network resource using a network protocol, *e.g.*, the Hyper-text Transfer protocol (HTTP), the server responds by sending the network resource or by sending a redirect message back to the client device over a communication channel. If the client device receives a redirect message, it will need to send a new request to the server based upon the redirect message, and the server will again respond with a redirect or a real resource. This communication process between the client device and the server repeats until the client device is able to get the resource, if available.

[0003] However, when the client device, *e.g.*, a cell phone using a radio network, or a computer having a slow Internet connection, requests a network resource and has to perform more than one redirects to obtain the network resource, the client device will experience substantial delay and will spend a considerable amount of time to execute the whole process before finally being provided with the network resource. The delay can occur, for example, because of a large time to establish a connection and send the HTTP request, also referred to as latency time of radio networks or other slow network connections (*e.g.*, Internet via a dial-up connection). Unfortunately, this delay can often lead to the client device not being able to obtain the network resource at all, or the client device giving up or relinquishing attempts to obtain the network resource under time constraints.

[0004] One conventional solution built to obtain faster HTTP responses for slow connection networks utilizes one or more proxy server devices (*e.g.*, web proxy servers). Another conventional solution utilizes telephone carrier data centers that handle the network traffic for each client device by handling one or more requests when the client device is a mobile telephone, or a mobile personal digital assistant (PDA) device, for example. Unfortunately, the above-noted conventional solutions do not resolve the redirection problem associated with network resources stored on the servers since the redirect messages are forwarded on to the client devices for handling resulting in multiple back and forth communication between the client devices and the servers. US patent application n. US2008/172488 discloses a method performed by a proxy server located between a content server and a client browser for reducing effects of network latency there between comprises intercepting a request from the client browser for a resource at the content server, obtaining a response by the content server to the request, determining that the response would, if unmodified, require a plurality of communications between the content server and the client browser in the absence of the proxy server, modifying the response to reduce a network latency associated with the plurality of communications for accessing information located externally to the response, and transmitting the modified response to the client browser for use thereby. US 2005/0060410 A1 discloses a system in which a network proxy provides redirection of service requests in order to decrease latency.

SUMMARY

[0005] A method for reducing latency in network connections is presented according to claim 1.

[0006] A computer readable medium having stored thereon instructions for reducing latency in network connections is presented according to claim 7.

[0007] An apparatus configured to reduce latency is presented according to claim 8.

[0008] This technology provides a number of advantages including providing a method, computer readable medium and an apparatus that adds an exemplary redirect module to the proxy server devices to efficiently manage the whole redirect chain returning only the last redirect message to the client device, which client device can then obtain the requested network resource from the server on which the network resource is stored, without exchanging multiple intermediate redirect messages back and forth with the server. In one exemplary scenario, cookies received by the proxy server device from the content servers are collected and forwarded to the client device. Accordingly, this technology provides substantial reduction in latency of network connections because the number of redirect messages between the client devices and the content/resource servers is reduced, and takes advantage of the high speed and high bandwidth communication infrastructure between the proxy server device(s) and the server devices on which content is stored.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009]

FIG. 1 is a block diagram of an exemplary network environment with a proxy server device interposed between client devices and server devices;

FIG. 2 is an example of a redirect response message chain exchanged between the client devices and the server devices of FIG. 1, as handled by a conventional proxy server device;

FIG. 3 is an example of an optimized redirect response message chain handled by the proxy server

device of FIG. 1 on behalf of client devices of FIG. 1 for obtaining a network resource; and

FIG. 4 is an exemplary flowchart for reducing latency in network connections by optimizing redirect response message chains at the proxy server device.

DETAILED DESCRIPTION

[0010] An exemplary environment 10 in which a proxy server device 12 is optimized for reducing latency in network connections is illustrated in FIG. 1. By way of example only, proxy server device 12 can be a web content proxy server, or other types of proxy servers well known to those of ordinary skill in the art. The exemplary environment 10 includes the proxy server device or apparatus 12, client devices 14(1)-14(n), server devices 16(1)-16(n), and communication networks 18(1)-18(2), although other numbers and types of systems, devices, and/or elements in other configurations and environments with other communication network topologies can be used. This technology provides a number of advantages including providing a method, computer readable medium, and an apparatus that reduces latency in network connections, for example, HTTP connections.

[0011] Referring more specifically to FIG. 1, the proxy server device 12 manages handling of redirect messages or redirection responses from the server devices 16(1)-16(n) for and/or on behalf of requesting client devices 14(1)-14(n) and provides updated cookie information to the client devices 14(1)-14(n) for future requests for network resources from the client devices 14(1)-14(n), although the proxy server device 12 can provide other numbers and types of functions. Although one proxy server device 12 is shown, other numbers and types of web content proxy devices and systems can be used. By way of example only, according to one embodiment of the present invention, proxy server device 12 can be a carrier gateway device communicating with one or more content servers, e.g., server devices 16(1)-16(n), for managing network resource requests from client devices 14(1)-14(n). Further by way of example only, the proxy server device 12 can be an HTTP proxy server or a carrier gateway configured to reduce latency in network connections by receiving a first response from one or more server devices 16(1)-16(n) in response to a first request from a client device for a network resource stored on the one or more server devices 16(1)-16(n). Proxy server device 12 determines whether or not the first response is a redirect message, and sends a second request to the one or more server devices 16(1)-16(n) when the first response includes the first redirect message, such that the sending is performed at least partially based upon the received redirect message.

[0012] The proxy server device 12 includes a central processing unit (CPU) or processor 13, a memory 15, and an interface system 17 which are coupled together by a bus 19 or other link, although other numbers and types of components, parts, devices, systems, and elements in other configurations and locations can be used. The processor 13 in the proxy server device 12 executes a program of stored instructions to carry out or perform one or more aspects of the present invention as described and illustrated by way of the embodiments herein, although the processor could execute other numbers and types of programmed instructions.

[0013] The memory 15 in the proxy server device 12 stores these programmed instructions for one or more aspects of the present invention as described and illustrated herein, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, DVD ROM, or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to the processor 13, can be used for the memory 15 in the proxy server device 12. In these embodiments, the memory 15 includes a core module 21 and a redirect module 23 which store programmed instructions for one or more aspects of the present invention as described and illustrated herein, although the memory can comprise other types and numbers of systems, devices, and elements in other configurations which store other data. As discussed in more detail below in FIG. 3, only the last network location (*e.g.*, a Uniform Resource Identifier or URI) of a redirect chain 300 is forwarded by redirect module 23 to the requesting client device among client devices 14(1)-14(n), although the redirect module 23 can have other types and numbers of functions as described and illustrated herein.

[0014] The interface system 17 in the proxy server device 12 is used to operatively couple and communicate between the proxy server device 12 and the client devices 14(1)-14(n) and the server devices 16(1)-16(n) via the communication networks 18(1) and 18(2), although other types and numbers of communication networks with other types and numbers of connections and configurations can be used. By way of example only, the communication networks 18(1) and 18(2) can use TCP/IP over Ethernet and industry-standard protocols, including HTTP, HTTPS, WAP, and SOAP, although other types and numbers of communication networks, such as a direct connection, a local area network, a wide area network, modems and phone lines, e-mail, and wireless and hardwire communication technology, each having their own communications protocols, can be used. In one exemplary embodiment, one of communication networks 18(1) and 18(2) can be operating over one or more low-speed connections (*e.g.*, a dial-up connection) while the other one of the communication networks 18(1) and 18(2) can be operating over a high speed, high bandwidth connection (*e.g.*, optical fiber based communication network). In yet another exemplary embodiment, one or more of communication networks 18(1) and 18(2) can be a radio network,

a satellite network, an Internet connection, a wired cable network, or combinations thereof, well known to one of ordinary skill in the art reading this disclosure.

[0015] Each of the client devices 14(1)-14(n) enables a user to request, obtain, and interact with one or more network resources, e.g., web pages from one or more web sites, hosted by server devices 16(1)-16(n) through the proxy server device 12 via one or more communication networks (*e.g.*, communication network 18(1)), although one or more of the client devices 14(1)-14(n) could access content and utilize other types and numbers of applications from other sources and could provide a wide variety of other functions for the user. Although multiple client devices 14(1)-14(n) are shown, other numbers and types of user computing systems could be used. In one example, the client devices 14(1)-14(n) comprise mobile devices with Internet access that permit a website form page or other retrieved data that is a part of a requested network resource to be displayed, although each of the client devices 14(1)-14(n) can connect to server devices 16(1)-16(n) via other types of network connections directly or indirectly, depending upon specific scenarios, as can be contemplated by one of ordinary skill in the art, after reading this disclosure. By way of example only, one or more of the client devices 14(1)-14(n) can comprise smart phones, personal digital assistants, computers, or other computing devices.

[0016] Each of client devices 14(1)-14(n) in this example is a computing device that includes a central processing unit (CPU) or processor 20, a memory 22, user input device 24, a display 26, and an interface system 28, and which are coupled together by a bus 30 or other link, although one or more of client devices 14(1)-14(n) can include other numbers and types of components, parts, devices, systems, and elements in other configurations. The processor 20 in each of client devices 14(1)-14(n) executes a program of stored instructions for aiding one or more aspects of the present invention as described and illustrated herein, although the processor could execute other numbers and types of programmed instructions.

[0017] The memory 22 in each of the client devices 14(1)-14(n) stores these programmed instructions for one or more aspects of the present invention as described and illustrated herein as well as updated cookies associated with a network resource and received as part of one or more redirect chains forwarded by proxy server device 12, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to processor 20 can be used for the memory 22 in each of the client devices 14(1)-14(n).

[0018] The user input device 24 in each of the client devices 14(1)-14(n) is used to input selections, such as requests for a network resource, *e.g.*, a particular website form page or to enter data in fields of a form page, although the user input device could be used to input other types of data and interact with other elements of exemplary environment 10. The user input device 24 can include keypads, touch screens, and/or vocal input processing systems, although other types and numbers of user input devices can be used.

[0019] The display 26 in each of the client devices 14(1)-14(n) is used to show data and information to the user, such as website page by way of example only. The display in each of the client devices 14(1)-14(n) is a mobile phone screen display, although other types and numbers of displays could be used depending on the particular type of client device, as can be contemplated by one of ordinary skill in the art, after reading this disclosure.

[0020] The interface system 28 in each of the client devices 14(1)-14(n) is used to operatively couple and communicate between the client devices 14(1)-14(n) and the proxy server device 12 and server devices 16(1)-16(n) over the communication networks 18(1) and 18(2), although other types and numbers of communication networks with other types and numbers of connections and configurations can be used.

[0021] The server devices 16(1)-16(n) provide one or more pages from one or more web sites for use by one or more of the client devices 14(1)-14(n) via the proxy server device 12, although the server devices 16(1)-16(n) can provide other numbers and types of applications and/or content and can have provide other numbers and types of functions. Although server devices 16(1)-16(n) are shown for ease of illustration and discussion, other numbers and types of server systems, for example, web servers, and devices can be used. In one example, server devices 16(1)-16(n) can be web servers having dedicated hardware with software executing on the dedicated hardware to facilitate the proxy server device 12 and client devices 14(1)-14(n) in their functioning. In another example, server devices 16(1)-16(n) can be content servers that are configured to deliver network resources stored thereupon using the HTTP protocol, or other network protocols for example. Content stored on server devices 16(1)-16(n) that can be part of the network resources requested by client devices 14(1)-14(n) can be web pages, electronic files and documents, configuration data, metadata, or other network data and files, by way of example only and not by way of limitation.

[0022] Each of the server devices 16(1)-16(n) include a central processing unit (CPU) or processor, a memory, and an interface system which are coupled together by a bus or other link, although each of the server devices 16(1)-16(n) could have other numbers and types of components, parts, devices, systems, and elements in other configurations and locations can be used. The processor in each of the server devices 16(1)-16(n) executes a program of stored instructions one or more aspects of the present invention as described and illustrated by way of

the embodiments herein, although the processor could execute other numbers and types of programmed instructions. When one of the server devices 16(1)-16(n) does not store the requested content, the server device may respond by sending a redirect message to the proxy server device 12, which the proxy server device 12 sends to the appropriate server device indicated by the redirect message instead of forwarding the redirect message back to the requesting one of the client devices 14(1)-14(n).

[0023] The memory in each of the server devices 16(1)-16(n) stores these programmed instructions for one or more aspects of the present invention as described and illustrated by way of the embodiments, although some or all of the programmed instructions could be stored and/or executed elsewhere. A variety of different types of memory storage devices, such as a random access memory (RAM) or a read only memory (ROM) in the system or a floppy disk, hard disk, CD ROM, DVD ROM, or other computer readable medium which is read from and/or written to by a magnetic, optical, or other reading and/or writing system that is coupled to the processor, can be used for the memory in each of the server devices 16(1)-16(n).

[0024] The interface system in each of the server devices 16(1)-16(n) is used to operatively couple and communicate between the server devices 16(1)-16(n) and the proxy server device 12 and the client devices 14(1)-14(n) via communication networks 18(1) and 18(2), although other types and numbers of communication networks with other types and numbers of connections and configurations can be used.

[0025] Although embodiments of the proxy server device 12, the client devices 14(1)-14(n), and the server devices 16(1)-16(n), are described and illustrated herein, each of the client devices 14(1)-14(n), the proxy server device 12, and the server devices 16(1)-16(n), can be implemented on any suitable computer system or computing device. It is to be understood that the devices and systems of the embodiments described herein are for exemplary purposes, as many variations of the specific hardware and software used to implement the embodiments are possible, as will be appreciated by those skilled in the relevant art(s).

[0026] Furthermore, each of the systems of the embodiments may be conveniently implemented using one or more general purpose computer systems having computer readable medium, microprocessors, digital signal processors, and micro-controllers, programmed according to the teachings of the embodiments, as described and illustrated herein, and as will be appreciated by those ordinary skill in the art.

[0027] In addition, two or more computing systems or devices can be substituted for any one of the systems in any embodiment of the embodiments. Accordingly, principles and advantages of distributed processing, such as redundancy and replication also can be implemented, as desired, to increase the robustness and performance of the devices and systems of the embodiments. The embodiments may also be implemented on computer system or systems that extend across any suitable network using any suitable interface mechanisms and communications technologies, including by way of example only telecommunications in any suitable form (*e.g.*, voice and modern), wireless communications media, wireless communications networks, cellular communications networks, G3 communications networks, Public Switched Telephone Network (PSTNs), Packet Data Networks (PDNs), the Internet, intranets, and combinations thereof.

[0028] The embodiments may also be embodied as a computer readable medium having instructions stored thereon for one or more aspects of the present invention as described and illustrated by way of the embodiments herein, as described herein, which when executed by a processor, cause the processor to carry out the steps necessary to implement the methods of the embodiments, as described and illustrated herein.

[0029] Referring to FIG. 2, an exemplary redirect chain 200 is illustrated for a conventional scenario where proxy server device 12 is not configured to automatically handle redirect messages. It is to be noted that the sequence of steps for handling redirect chain 200 is only exemplary and one of ordinary skill in the art, after reading this disclosure, can contemplate alternative sequences of steps that achieve substantially the same result as shown by the sequence of steps for redirect chain 200.

[0030] More specifically, an example of redirect chain 200 including multiple redirects to obtain a network resource is illustrated in FIG. 2 starting at step 202 where one of the client devices 14(1)-14(n) (*e.g.*, a mobile device) sends an HTTP request for a network resource at a Uniform Resource Locator (URL) http://www.example.com/A.html, although other types of requests for other types of network resources may be sent. Although in this example one of the client devices 14(1)-14(n) via a web browser requests a page A.html at the website, "www.example.com" as shown, by way of example only, client devices 14(1)-14(n) may send a request for a network shared data file using a file transfer protocol instead of a URL using the HTTP protocol. In one example, this request is transmitted to the proxy server device 12 which processes and transmits the request to the one of the server device 16(1)-16(n) hosting the exemplary website www.example.com.

[0031] In step 204, the client device requesting the network resource gets a response from the hosting one of the server devices 16(1)-16(n) storing the network resource, in this example, the requested page A.html. The response includes a temporary redirect message 204a (shown as an exemplary status code 302 with a message "Moved Temporarily") to a different URL 204b. URL 204b is shown exemplarily as http://www.example.com/B.html with a header field 204c shown as "Set-Cookie" including a cookie named "SESSION" set to value "1234," with domain equal to "example.com" and the path equal to "/."

[0032] Generally, the cookie is a string formed by the pair "name=value" (*e.g.*, "SESSION=1234", followed by optional attributes, like those in this example indicating the server domain(s) and path accepting this cookie. Although one illustrative example is described herein, this technology can be used with specifications for all cookies.

[0033] In step 206, the client device will then send a new HTTP request 206a to the server devices 16(1)-16(n) for a URL http://www.example.com/B.html also passing the cookie 206b shown as "SESSION=1234." In response, at step 208 of the redirect chain 200, the requesting client device obtains a new temporary redirect message 208b to another URL http://www.example.com/C.html with a new cookie named "LANG" set to a value "en" in a header field 208c.

[0034] In step 210, since the response at step 208 included a new cookie "LANG" with the new redirect response message 208b to a URL http://www.example.com/C.html, the client device will then send a new HTTP request 210a for http://www.example.com/C.html passing the two cookies "LANG=en" and "SESSION=1234" in a header 210b. In response, in step 212, the client device will get a real network resource 212b as a response from the hosting one of server devices 16(1)-16(n) with a status identifier message 212a shown as status code 200 set to "OK." It is to be noted although steps 202-212 are shown, a higher or a lower number of steps may be realized to obtain the real network resource 212b in response to the initial request in step 202 from one of the client devices 14(1)-14(n) for the network resource A.html by back and forth redirect communications as shown by redirect chain 200 between the client device 14(1)-14(n) and server devices 16(1)-16(n) via proxy server device 12. By way of example only, in some scenarios the client device may never be able to obtain the network resources for various reasons, including non-availability of the resources.

[0035] Referring to FIG. 3, an exemplary aspect of the present technology will now be described. FIG. 3 shows an exemplary redirect chain 300 when handled by proxy server device 12 interposed between client devices 14(1)-14(n) and server devices 16(1)-16(n). Advantageously, in this exemplary embodiment, back and forth communication between the client devices 14(1)-14(n) and hosting server devices 16(1)-16(n) with respect to intermediate redirect message portions of redirect chain 300 received for a requested network resource is reduced/minimized or even eliminated, thereby reducing latency in network connection between client devices 14(1)-14(n) and hosting server devices 16(1)-16(n). Redirect chain 300 is handled by proxy server device 12 as illustrated in the sequence of steps shown in FIG. 3 and described below, although the sequence of steps for handling redirect chain 300 is only exemplary and one of ordinary skill in the art, after reading this disclosure, can contemplate alternative sequences of steps that achieve the same, or substantially the same result as shown by the sequence of steps for redirect chain 300.

[0036] In step 302, one of client devices 14(1)-14(n) sends an initial request for a URI, *e.g.*, a URL http://www.example.com/A.html to the proxy server device 12. In step 304, unlike steps 204-208 of FIG. 2, proxy server device 12 executes all the intermediate redirects on its side and returns a temporary redirect response message 304a (shown as an exemplary status code 302 with a message "Moved Temporarily") for a URL 304b identified as a URL http://www.example.com/C.html to the requesting one of client devices 14(1)-14(n). In this example, both "LANG" and "SESSION" cookies will be set as shown by respective "Set-Cookie" header fields 304c such that they correspond to the updated state of the requested URL www.example.com. Although a single redirect message is illustrated in FIG. 3, the process can be extended to proxy server device 12 handling all the redirect messages from server devices 14(1)-14(n) until a real resource (*e.g.*, with an HTTP status message 200) is obtained. The information, including cookies, contained in the intermediate or temporary redirect response message 304a is forwarded to the requesting one of client devices 14(1)-14(n). For example, the forwarded message can include merged cookies previously received.

[0037] In step 306, the requesting client device will then send a request including a last redirect message 306a from the redirect chain 300 to the proxy server device 12 for the network resource URL http://www.example.com/C.html with updated cookies "SESSION" and "LANG" in the header field 306b as illustrated. The requesting client device forms the last redirect message 306a based upon the information forwarded by proxy server device 12 in step 304, and cookie information stored at the requesting client device because of prior navigation by the requesting client device, although the requesting client device may obtain the updated state of cookies from other sources, for example, one of server devices 16(1)-16(n).

[0038] In step 308, in response to the request sent in step 306, the requesting one of the client devices 14(1)-14(n) gets a real network resource 308b as a response from the hosting one of the server devices 16(1)-16(n) with a status identifier 308a set to "OK" (shown as status code 200). Advantageously, the redirect chain 300 is executed substantially between the proxy server device 12 and the hosting one of server devices 16(1)-16(n) in a substantially lesser time as compared to the intermediate redirect responses shown in redirect chain 200 exchanged substantially between the client devices 14(1)-14(n) and the hosting one of server devices 16(1)-16(n) with minimal to zero functionality of proxy server device 12 with respect to handling redirect response message chain 200. This is because typically there is a faster connection between proxy server device 12 and server devices 16(1)-16(n) than between client devices 14(1)-14(n) and server devices 16(1)-16(n).

[0039] Referring to FIG. 4, an exemplary method for reducing latency in network connections (*e.g.*, HTTP con-

nections) will now be described using a flowchart 400 with reference back to FIGS. 1-3. In this example, an exemplary sequence of steps performed by proxy server device 12 to handle redirects as discussed above is described. The flowchart 300 is representative of example machine readable instructions to implementing reducing latency in network connections, for example, at proxy server device 12. In this example, the machine readable instructions comprise an algorithm for execution by: (a) a processor (*e.g.*, CPU 13), (b) a controller, and/or (c) one or more other suitable processing device(s) within proxy server device 12, for example. The algorithm may be implemented in software stored on tangible computer readable media such as, for example, a flash memory, a CD-ROM, a floppy disk, a hard drive, a digital video (versatile) disk (DVD), or other memory devices, but persons of ordinary skill in the art will readily appreciate that the entire algorithm and/or parts thereof could alternatively be executed by a device other than a processor and/or implemented in firmware or dedicated hardware in a well known manner (*e.g.*, it may be implemented by an application specific integrated circuit (ASIC), a programmable logic device (PLD), a field programmable logic device (FPLD), a field programmable gate array (FPGA), discrete logic, or the like). For example, at least some of the components of the proxy server device 12 could be implemented by software, hardware, and/or firmware. Also, some or all of the machine readable instructions represented by the process of flowchart 400 of FIG. 4 may be implemented manually at the proxy server device 12, for example, using a command line interface (CLI) prompt window operated by a system administrator. Further, although the example algorithm is described with reference to flowchart 400, persons of ordinary skill in the art will readily appreciate that many other methods of implementing the example machine readable instructions may alternatively be used. For example, the order of execution of the blocks in flowchart 400 may be changed, and/or some of the blocks described may be changed, eliminated, or combined.

[0040] The flow begins in step 402, when proxy server device 12 receives a request from one of the client devices 14(1)-14(n). By way of example only, the request can be an HTTP request, although other types of requests (e.g., any type of HxxP or HTTPS request). In step 406, the received request is processed by proxy server device 12 and forwarded to one of the content hosting server devices 16(1)-16(n) based upon the information in the request. In response, in step 408, proxy server device 12 receives a first response from the hosting one of the server devices 16(1)-16(n), the first response being formed based upon the information in the forwarded request received by the hosting one of server devices 16(1)-16(n). In step 410, proxy server device 12 determines whether the response from the hosting one of server devices 16(1)-16(n) is a real network resource (*e.g.*, a status code 200 message) or a redirect message. If a real resource is received, the Yes branch is followed to step 412, where proxy server device 12 sends the first response including the link to the actual network resource to the requesting client device.

[0041] However, as shown in step 414, if the first response from the hosting one of the server devices 16(1)-16(n) is not a real network resource but is a redirect message (for example, a temporary redirect 204a with status code 302, shown in FIG. 2), proxy server device 12 will collect the cookies set in the first response and merge them with those sent with the last and/or prior requests for the network resource, including the original cookie received by the proxy server device 12 from the hosting one of the server devices 16(1)-16(n).

[0042] In step 416, proxy server device 12 will then send a new request (or, a second request) for the network resource (*e.g.*, URL "www.example.com/C.html") specified in the last redirect response (*e.g.*, redirect message 306a) to the hosting one of the server devices 16(1)-16(n), passing the cookies used in the first request (shown in step 302 of FIG. 3) and all the cookies collected from the previous redirect response messages 304a, 306a. It is to be noted the proxy server device 12 does not send the redirect messages that do not include real network resource to the client devices 14(1)-14(n). In step 418, proxy server device 12 receives a response back from the hosting one of server devices 16(1)-16(n), which response is examined again, similar to step 410. In step 420, if the response from the hosting one of server devices 16(1)-16(n) is again a redirect message (e.g., redirect message 306a), then the proxy server device 12 will continue to repeat steps 414-418 by subsequently creating a new request for the server devices 16(1)-16(n) based upon the last redirect response message obtained from server devices 16(1)-16(n). Using the technique discussed above, instead of the requesting one of client devices 14(1)-14(n) handling all intermediate redirect messages from server devices 14(1)-104(n), proxy server device 12 is configured to handle the intermediate redirects until a real network resource is obtained from server devices 16(1)-16(n). An advantage of the proxy server device 12 handling the redirect message chain instead of client devices 14(1)-14(n) is to reduce latency in provisioning requests from client devices 14(1)-14(n) since connection between proxy server device 12 and server devices 16(1)-16(n) is faster than connection between client devices 14(1)-14(n) and server devices 16(1)-16(n).

[0043] However, if the response from the hosting one of server devices 16(1)-16(n) is a real network resource, in step 422, proxy server device 12 will send a redirect response 304a to the requesting one of the client devices 14(1)-14(n) specifying the last URL 304b of the redirect chain 300 and all the cookies (*e.g.*, cookies in header field 306b) collected from the redirect response messages 304a, 306a. Subsequently, the requesting one of the client devices 14(1)-14(n) can use the most updated URL 304b to the hosting one of the server devices 16(1)-16(n) via proxy server device 12, and receive the requested

network resource. It is to be noted although in the examples above, URLs are being discussed, the technology works in substantially the same manner for other types of resource identifier schemes, e.g., Uniform Resource Names (URNs), as URLs are being discussed by way of example only and not by way of limitation, as can be contemplated by one of ordinary skill in the art after reading this disclosure. Accordingly, the steps shown in flowchart 400 advantageously reduce the slow communication of intermediate redirect response messages between the client devices 14(1)-14(n) and the hosting server devices 16(1)-16(n), thus reducing latency in the network connection, which in this example is an HTTP connection, although other types of connections (*e.g.*, FTP connections) may be used.

[0044] Accordingly, as illustrated and described herein this technology provides a number of advantages including providing a method, computer readable medium and an apparatus that reduces latency in network connections, for example, by managing handling of redirect messages from server devices 16(1)-16(n) substantially at proxy server device 12 interposed between the client devices 14(1)-14(n) and server devices 16(1)-16(n) on which various network resources are stored, which network resources are requested by client devices 14(1)-14(n). With this technology, the proxy server device 12 optimizes network resource provisioning by returning only the last redirect message along with all cookies collected from intermediate redirect messages from a chain of redirect messages from server devices 16(1)-16(n) to client devices 14(1)-14(n) for a particular network resource, thereby reducing latency in network connections between client devices and server devices.

[0045] Having thus described the basic concept of the invention, it will be rather apparent to those skilled in the art that the foregoing detailed disclosure is intended to be presented by way of example only, and is not limiting. Various alterations, improvements, and modifications will occur and are intended to those skilled in the art, though not expressly stated herein. These alterations, improvements, and modifications are intended to be suggested hereby as well. Additionally, the recited order of processing elements or, and are within the scope of the invention. Additionally, the recited order of processing elements or sequences, or the use of numbers, letters, or other designations therefore, is not intended to limit the claimed processes to any order except as may be specified in the claims. Accordingly, the invention is limited only by the following claims.

Claims

1. A method for reducing latency in network connections, the method comprising:

   receiving at a proxy server device (12) a first redirect message including a first cookie and a first network address from one or more server devices (16(1)-16(n)) in response to a first request from a client device (14(1)-14(n)) for a network resource stored on the one or more server devices (16(1)-16(n));

   sending a second request including the first cookie from the proxy server device (12) to the one or more server devices (16(1)-16(n)) based upon the first network address;

   receiving at the proxy server device (12) a second redirect message including a second cookie and a second network address from the one or more server devices (16(1)-16(n)) in response to the second request;

   sending a third request including the first and second cookies from the proxy server device (12) to the one or more server devices (16(1)-16(n)) based on the second network address;

   receiving at the proxy server device (12) a third response including a network resource from the one or more server devices (16(1)-16(n)) in response to the third request; and

   sending with the proxy server device (12) a third redirect message to a requesting client device (14(1)-14(n)), the third redirect message including at least the second network address and first and second cookies included in the first and second redirect messages, respectively.

2. The method as set forth in claim 1, wherein the first cookie is merged with the second cookie as a part of the third redirect message.

3. The method as set forth in one or more of claims from 1 to 2, wherein the second network address is a Uniform Resource Identifier (URI).

4. The method as set forth in one or more of claims from 1 to 3, wherein the first redirect message and subsequent redirect messages that do not include a real network resource are not sent to the client device (14(1)-14(n)).

5. The method as set forth in one or more of claims from 1 to 4, wherein the client device (14(1)-14(n)) is a mobile client device.

6. A computer readable medium having stored thereon instructions for reducing latency in network connections comprises machine executable code which when executed by at least one processor, causes the processor to perform the steps of the method according to claims 1 to 5.

7. An apparatus comprising one or more processors (13) and a memory (15) coupled to the one or more processors which are configured to execute programmed instructions stored in the memory (15) in-

cluding instructions for implementing:

receiving a first redirect message including a first cookie and a first network address from one or more server devices (16(1)-16(n)) in response to a first request from a client device (14(1)-14(n)) for a network resource stored on the one or more server devices (16(1)-16(n));
sending a second request including the first cookie to the one or more server devices (16(1)-16(n)) based upon the first network address;
receiving a second redirect message including a second cookie and a second network address from the one or more server devices (16(1)-16(n)) in response to the second request; sending a third request including the first and second cookies to the one or more server devices (16(1)-16(n)) based on the second network address;
receiving a third response including a network resource from the one or more server devices (16(1)-16(n)) in response to the third request; and
sending a third redirect message to a requesting client device (14(1)-14(n)), the third redirect message including at least the second network address and first and second cookies included in the first and second redirect messages, respectively.

8. The apparatus as set forth in claim 7, wherein the first cookie is merged with the second cookie as a part of the third redirect message.

9. The apparatus as set forth in one or more of claims from 7 to 8, wherein the second network address is a Uniform Resource Identifier (URI).

10. The apparatus as set forth in one or more of claims from 7 to 9, wherein the first redirect message and subsequent redirect messages that do not include a real network resource are not sent to the client device (14(1)-14(n)) until a network link to a real network resource is obtained from the one or more server devices.

11. The apparatus as set forth in one or more of claims from 7 to 10, wherein the client device (14(1)-14(n)) is a mobile client device.

Patentansprüche

1. Verfahren zum Verringern einer Latenz in Netzwerkverbindungen, wobei das Verfahren umfasst, dass:

von einer oder mehreren Servervorrichtungen (16(1)-16(n)) bei einer Proxyservervorrichtung (12) eine erste Weitersendenachricht, die ein erstes Cookie und eine erste Netzwerkadresse enthält, in Ansprechen auf eine erste Anforderung von einer Clientvorrichtung (14(1)-14(n)) für eine Netzwerkressource, die auf der einen oder den mehreren Servervorrichtungen (16(1)-16(n)) gespeichert ist, empfangen wird;
auf der Grundlage der ersten Netzwerkadresse eine zweite Anforderung, welche das erste Cookie enthält, von der Proxyservervorrichtung (12) an die eine oder die mehreren Servervorrichtungen (16(1)-16(n)) gesendet wird;
von der einen oder den mehreren Servervorrichtungen (16(1)-16(n)) bei der Proxyservervorrichtung (12) eine zweite Weitersendenachricht, die ein zweites Cookie und eine zweite Netzwerkadresse enthält, in Ansprechen auf die zweite Anforderung empfangen wird;
auf der Grundlage der zweiten Netzwerkadresse eine dritte Anforderung, welche das erste und zweite Cookie enthält, von der Proxyservervorrichtung (12) an die eine oder die mehreren Servervorrichtungen (16(1)-16(n)) gesendet wird;
von der einen oder den mehreren Servervorrichtungen (16(1)-16(n)) bei der Proxyservervorrichtung (12) eine dritte Antwort, die eine Netzwerkressource enthält, in Ansprechen auf die dritte Anforderung empfangen wird; und
mit der Proxyservervorrichtung (12) eine dritte Weitersendenachricht an eine anfordernde Clientvorrichtung (14(1)-14(n)) gesendet wird, wobei die dritte Weitersendenachricht mindestens die zweite Netzwerkadresse und erste und zweite Cookies enthält, die in der ersten bzw. zweiten Weitersendenachricht enthalten sind.

2. Verfahren nach Anspruch 1, wobei als Teil der dritten Weitersendenachricht das erste Cookie mit dem zweiten Cookie verschmolzen wird.

3. Verfahren nach einem oder mehreren der Ansprüche 1 bis 2, wobei die zweite Netzwerkadresse ein Uniform Resource Identifier (URI) ist.

4. Verfahren nach einem oder mehreren der Ansprüche 1 bis 3, wobei die erste Weitersendenachricht und nachfolgende Weitersendenachrichten, die keine reale Netzwerkressource enthalten, nicht an die Clientvorrichtung (14(1)-14(n)) gesendet werden.

5. Verfahren nach einem oder mehreren der Ansprüche 1 bis 4, wobei die Clientvorrichtung (14(1)-14(n)) eine mobile Clientvorrichtung ist.

6. Computerlesbares Medium mit darin gespeicherten Anweisungen zum Verringern einer Latenz in Netzwerkverbindungen, das einen von einer Maschine ausführbaren Code umfasst, der veranlasst, wenn

er von mindestens einem Prozessor ausgeführt wird, dass der Prozessor die Schritte des Verfahrens nach Anspruch 1 bis 5 ausführt.

7. Vorrichtung, die einen oder mehrere Prozessoren (13) und einen Speicher (15) umfasst, der mit dem einen oder den mehreren Prozessoren gekoppelt ist, welche ausgestaltet sind, um programmierte Anweisungen auszuführen, die in dem Speicher (15) gespeichert sind, die Anweisungen enthalten, um zu implementieren, dass:

von einer oder mehreren Servervorrichtungen (16(1)-16(n)) in Ansprechen auf eine erste Anforderung von einer Clientvorrichtung (14(1)-14(n)) für eine Netzwerkressource, die in der einen oder den mehreren Servervorrichtungen (16(1)-16(n)) gespeichert ist, eine erste Weitersendenachricht empfangen wird, die ein erstes Cookie und eine erste Netzwerkadresse enthält; auf der Grundlage der ersten Netzwerkadresse eine zweite Anforderung, die das erste Cookie enthält, an die eine oder die mehreren Servervorrichtungen (16(1)-16(n)) gesendet wird;
von der einen oder den mehreren Servervorrichtungen (16(1)-16(n)) in Ansprechen auf die zweite Anforderung eine zweite Weitersendenachricht empfangen wird, die ein zweites Cookie und eine zweite Netzwerkadresse enthält; auf der Grundlage der zweiten Netzwerkadresse eine dritte Anforderung, die das erste und zweite Cookie enthält, an die eine oder die mehreren Servervorrichtungen (16(1)-16(n)) gesendet wird;
von der einen oder den mehreren Servervorrichtungen (16(1)-16(n)) in Ansprechen auf die dritte Anforderung eine dritte Antwort, die eine Netzwerkressource enthält, empfangen wird; und
eine dritte Weitersendenachricht an eine anfordernde Clientvorrichtung (14(1)-14(n)) gesendet wird, wobei die dritte Weitersendenachricht mindestens die zweite Netzwerkadresse und erste und zweite Cookies enthält, die in der ersten bzw. zweiten Weitersendenachricht enthalten sind.

8. Vorrichtung nach Anspruch 7, wobei das erste Cookie mit dem zweiten Cookie als Teil der dritten Weitersendenachricht verschmolzen wird.

9. Vorrichtung nach einem oder mehreren der Ansprüche 7 bis 8, wobei die zweite Netzwerkadresse ein Uniform Resource Identifier (URI) ist.

10. Vorrichtung nach einem oder mehreren der Ansprüche 7 bis 9, wobei die erste Weitersendenachricht und nachfolgende Weitersendenachrichten, welche keine reale Netzwerkressource enthalten, nicht an die Clientvorrichtung (14(1)-14(n)) gesendet werden, bis eine Netzwerkverbindung zu einer realen Netzwerkressource von der einen oder den mehreren Servervorrichtungen erhalten wird.

11. Vorrichtung nach einem oder mehreren der Ansprüche 7 bis 10, wobei die Clientvorrichtung (14(1)-14(n)) eine mobile Clientvorrichtung ist.

Revendications

1. Procédé de réduction de latence dans des connexions de réseau, le procédé comprenant :

la réception dans un dispositif serveur mandataire (12) d'un premier message de redirection incluant un premier témoin et une première adresse de réseau en provenance d'un ou de plusieurs dispositifs serveurs (16(1) à 16(n)) en réponse à une première demande en provenance d'un dispositif client (14(1) à 14(n)) d'une ressource de réseau stockée sur les un ou plusieurs dispositifs serveurs (16(1) à 16(n)) ;
l'envoi d'une deuxième demande incluant le premier témoin en provenance du dispositif serveur mandataire (12) aux un ou plusieurs dispositifs serveurs (16(1) à 16(n)) d'après la première adresse de réseau ;
la réception au niveau du dispositif serveur mandataire (12) d'un deuxième message de redirection incluant un second témoin et une seconde adresse de réseau en provenance des un ou plusieurs dispositifs serveurs (16(1) à 16(n)) en réponse à la deuxième demande ;
l'envoi d'une troisième demande incluant les premier et second témoins en provenance du dispositif serveur mandataire (12) aux un ou plusieurs dispositifs serveurs (16(1) à 16(n)) d'après la seconde adresse de réseau ;
la réception au niveau du dispositif serveur mandataire (12) d'une troisième réponse incluant une ressource de réseau en provenance des un ou plusieurs dispositifs serveurs (16(1) à 16(n)) en réponse à la troisième demande ; et
l'envoi avec le dispositif serveur mandataire (12) d'un troisième message de redirection à un dispositif client demandeur (14(1) à 14(n)), le troisième message de redirection incluant au moins la seconde adresse de réseau et des premier et second témoins inclus dans les premier et deuxième messages de redirection, respectivement.

2. Procédé selon la revendication 1, dans lequel le premier témoin est fusionné au second témoin dans le cadre du troisième message de redirection.

3. Procédé selon une ou plusieurs des revendications 1 à 2, dans lequel la seconde adresse de réseau est un identifiant de ressource uniforme (URI).

4. Procédé selon une ou plusieurs des revendications 1 à 3, dans lequel le premier message de redirection et les messages de redirection suivants qui n'incluent pas une ressource de réseau réelle ne sont pas envoyés au dispositif client (14(1) à 14(n)).

5. Procédé selon une ou plusieurs des revendications 1 à 4, dans lequel le dispositif client (14(1) à 14(n)) est un dispositif client mobile.

6. Support lisible par ordinateur sur lequel sont stockées des instructions pour réduire la latence dans des connexions de réseau qui comprend un code exécutable par machine qui, lorsqu'il est exécuté par au moins un processeur, amène le processeur à réaliser les étapes du procédé selon les revendications 1 à 5.

7. Appareil comprenant un ou plusieurs processeurs (13) et une mémoire (15) couplée aux un ou plusieurs processeurs qui sont configurés pour exécuter des instructions programmées stockées dans la mémoire (15) incluant des instructions pour implémenter :

la réception d'un premier message de redirection incluant un premier témoin et une première adresse de réseau en provenance d'un ou de plusieurs dispositifs serveurs (16(1) à 16(n)) en réponse à une première demande en provenance d'un dispositif client (14(1) à 14(n)) d'une ressource de réseau stockée sur les un ou plusieurs dispositifs serveurs (16(1) à 16(n)) ;
l'envoi d'une deuxième demande incluant le premier témoin aux un ou plusieurs dispositifs serveurs (16(1) à 16(n)) d'après la première adresse de réseau ;
la réception d'un deuxième message de redirection incluant un second témoin et une seconde adresse de réseau en provenance des un ou plusieurs dispositifs serveurs (16(1) à 16(n)) en réponse à la deuxième demande ; l'envoi d'une troisième demande incluant les premier et second témoins aux un ou plusieurs dispositifs serveurs (16(1) à 16(n)) d'après la seconde adresse de réseau ;
la réception d'une troisième réponse incluant une ressource de réseau en provenance des un ou plusieurs dispositifs serveurs (16(1) à 16(n)) en réponse à la troisième demande ; et
l'envoi d'un troisième message de redirection à un dispositif client demandeur (14(1) à 14(n)), le troisième message de redirection incluant au moins la seconde adresse de réseau et des premier et second témoins inclus dans les premier et deuxième messages de redirection, respectivement.

8. Appareil selon la revendication 7, dans lequel le premier témoin est fusionné au second témoin dans le cadre du troisième message de redirection.

9. Appareil selon une ou plusieurs des revendications 7 à 8, dans lequel la seconde adresse de réseau est un identifiant de ressource uniforme (URI).

10. Appareil selon une ou plusieurs des revendications 7 à 9, dans lequel le premier message de redirection et les messages de redirection suivants qui n'incluent pas une ressource de réseau réelle ne sont pas envoyés au dispositif client (14(1) à 14(n)) jusqu'à ce qu'une liaison de réseau à une ressource de réseau réelle soit obtenue des un ou plusieurs dispositifs serveurs.

11. Appareil selon une ou plusieurs des revendications 7 à 10, dans lequel le dispositif client (14(1) à 14(n)) est un dispositif client mobile.

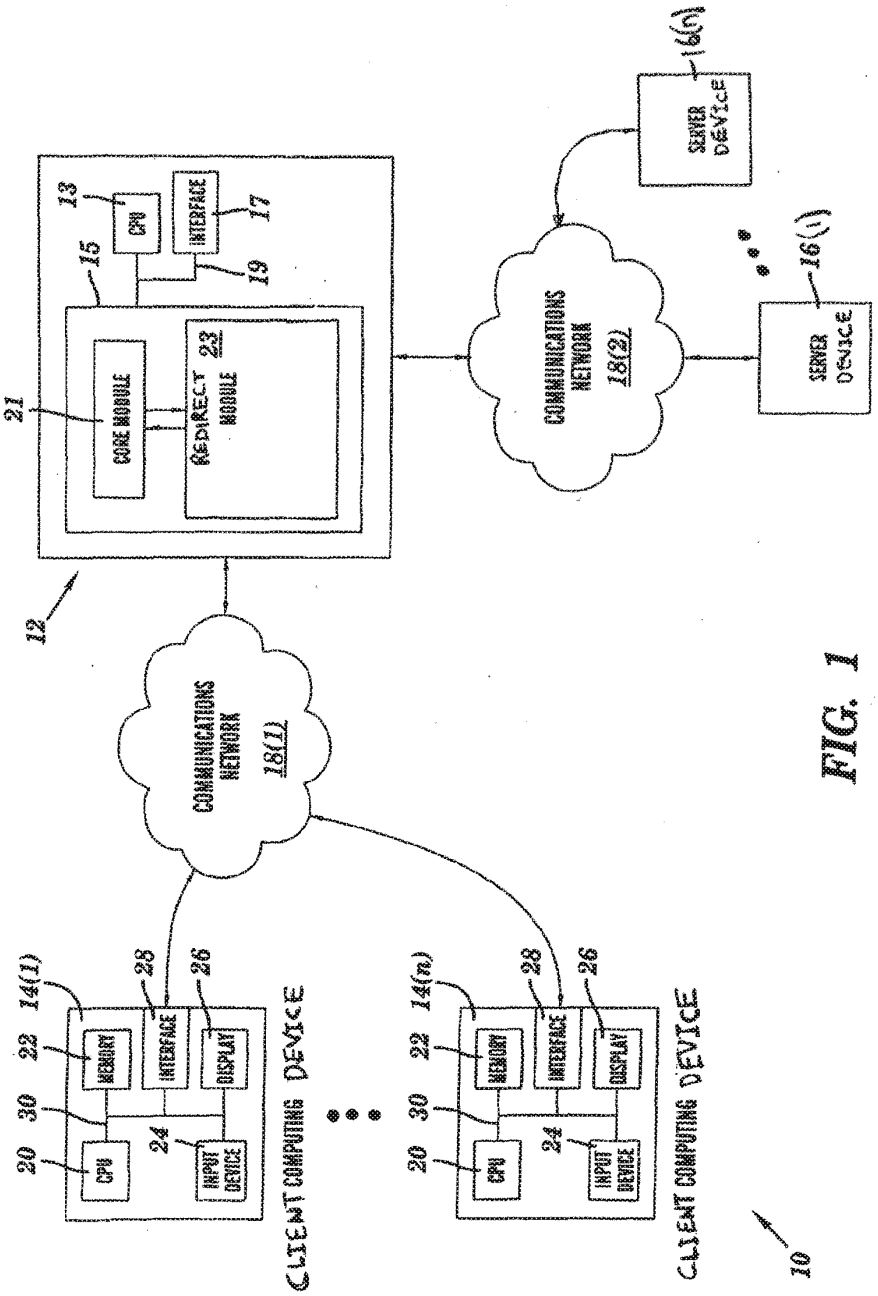


FIG. 1

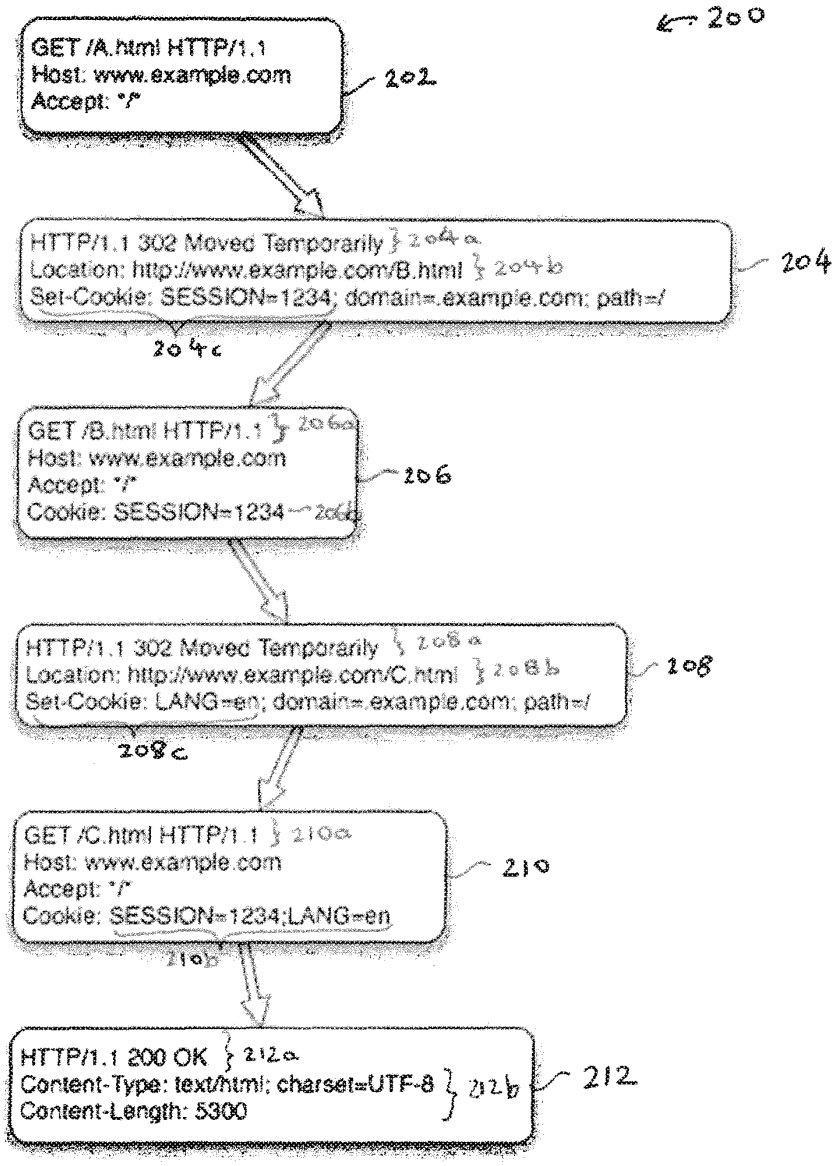


FIG. 2

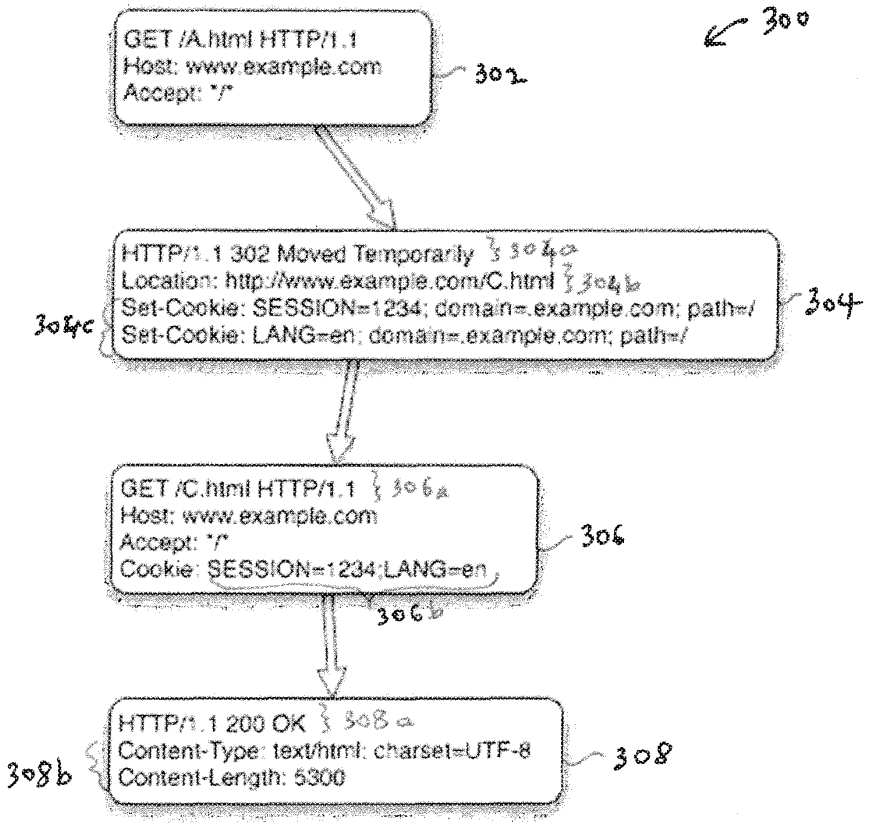


FIG. 3