```
#!/usr/local/bin/sybperl5
# -T
# -w
# $Header: /usr/local/cvsroot/webley/agents/www/html2csv.pl,v 1.1
1999/11/02 00:24:52 zhukoff Exp $
# Agent to get info from the web.
# Parameters: service_name [service_parameters], i.e. stock msft
or weather 60645
# Configuration stored in files service_name.ini
# if this file is absent the configuration is received from
mcServices table
# This script provides autoupdate to datatable if the .ini file
is newer.

use URI::URL;
use LWP::UserAgent;
use HTTP::Request::Common;
use Vail::VarList;
use Sybase::CTlib;
use HTTP::Cookies;

my ( $action, @urls ) = @ARGV;

$debug2 = 0;
$IMAGES = 1;

my $service = shift;

#my @ini = &read_ini( $service );
my $section = "";
# get page content
#print $urls[ 0 ];
if ( $action eq "follow" ) {
#     print "test\n";
      ( $Content{ 'TIME' }, $content ) =
&get_url_content( $urls[ 0 ] );
      $path = $urls[ 0 ];
      $path =~ s/\/[^\/]*$/\//;

}
elsif ( $action eq "read" ) {
      open( IN, $urls[ 0 ] );
      $content = join( "", (<IN>) );
      close IN;
}
#print $content;
$content =~ s/<!--webley/<--webley/g;
# change double qoutes to single ones
#$content =~ s/\"/\'/g;
# delete new line chars
$content =~ s/\n//g;
$content =~ s/\r//g;
```

```
# delete comments and scripts
$content =~ s/<!.*?->//g;
$content =~ s/<[Ss][Cc][Rr][Ii][Pp][Tt].*?<\/[Ss][Cc][Rr][Ii][Pp]
[Tt]>//g;
# replace <br> and </head> <li> </ul> </h\d> tag with a period
$content =~ s/<[Bb][Rr]>/\./g;
$content =~ s/<[lL][iI]>/\./g;
$content =~ s/<\/[uU][lL]>/\./g;
$content =~ s/<\/[hH]\d+>/\./g;
$content =~ s/<\/[Hh][Ee][Aa][Dd]>/./g;
# replace &copy with Copyright
$content =~ s/\&copy/Copyright/g;
#print $content;

#$content =~ /<!--webley\s+([^>])+>([^<])+<!--webley-->/;
#$content =~ s/<[Aa]{1}\s+[Hh]{1}[Rr]{1}[Ee]{1}[Ff]{1}\s*=
\s*[\"\']?([^\s\'\"\>])[ \"\'][^>]*>[^<]*<\/[Aa]{1}>/<--webley
name=\"link\" value=\"\1\" text=\"\2\"-->\2<--webley-->/g;

$content =~ s/<[iI][mM][gG][^>]*?\s+[aA][lL][tT]\s*=\s*[\"\']?([^
\'\"\>]+)[\"\']?.*?>/\1\./g
     if $IMAGES;
#print $content;
# replace hrefs with webley links
$content =~ s/<[aA]\s+[hH][rR][eE][fF]\s*=\s*[\"\']?[mM][aA][iI]
[lL][tT][oO]:([^\s\'\"\>]+)[\"\']?.*?>(.+?)<\/[aA]>/\n<--webley
name=\"email\" value=\"\1\" text=\"\2\"-->\2<--webley-->\n/g;
$content =~ s/<[aA]\s+[hH][rR][eE][fF]\s*=\s*[\"\']?([^\s
\'\"\>]+)[\"\']?.*?>(.+?)<\/[aA]>/\n<--webley name=\"link\"
value=\"$path\1\" text=\"\2\"-->\2<--webley-->\n/g;
$content =~ s/$path(http|ftp)/\1/g;
#print $content;
$content =~ s/<[^-<>][^<>]*>//g;
$content =~ s/\s+/ /g;
$content =~ s/\s+\./\./g;
$content =~ s/\.{2,}/\./g;
$content =~ s/>[^<]*<--webley-->/>/g;
$content =~ s/>[\.\s]+</>\n</g;
$content =~ s/>(.+?)</\n<--webley name=\"text\" value=\"\" text=
\"\1\"-->\n</g;
$content =~ s/(^[^<]+)/\n<--webley name=\"text\" value=\"\" text=
\"\1\"-->\n/g;
$content =~ s/([^>]+$)/\n<--webley name=\"text\" value=\"\" text=
\"\1\"-->\n/g;
                   $content =~ s/<--webley name=//g;
                   $content =~ s/ value=/,/g;
                   $content =~ s/\.([A-Z]{1})/\. \1/g;
                   $content =~ s/ text=/,/g;
                   $content =~ s/-->?//g;
                   $content =~ s/.*,\"\"\n//g;
                   $content =~ s/.*\"\. \"\n//g;
#                  $content =~ s/\"link\",\"mailto:/\"email\",
```

```
\"/g;
print $content;
#do { $section = &process_section( $section ) } while $section;
exit;
#######################################################
sub read_ini {
      my ( $service ) = @_;
      my @ini = ();
      # first, try to read file
      if ( open( INI, "$service.ini" ) ) {
            @ini = ( <INI> );
            return @ini unless ( $DB_SRV );
            # update datatable
            my $file_time = time - int( ( -M "$service.ini" ) * 24
* 3600 );
            my $dbh = new Sybase::CTlib $DB_USR, $DB_PWD, $DB_SRV;
            unless ( $dbh ) {
                  print STDERR "webget.pl: Cannot connect to
dataserver $DB_SRV:$DB_USR:$DB_PWD\n";
                  return @ini;
            }
            my @row_refs = $dbh->ct_sql( "select lastUpdate from
mcServices where service = '$service'", undef, 1 );
            if ( $dbh->{ RC } == CS_FAIL ) {
                  print STDERR "webget.pl: DB select from
mcServices failed\n";
                  return @ini;
            }
            unless ( defined @row_refs ) {
                  # have to insert
                  my ( @ini_escaped ) = map {
                        ( my $x = $_ ) =~ s/\'/\'\'/g;
                        $x;
                  } @ini;
                  $dbh->ct_sql( "insert mcServices
values( '$service', '@ini_escaped', $file_time )" );
                  if ( $dbh->{ RC } == CS_FAIL ) {
                        print STDERR "webget.pl: DB insert to
mcServices failed\n";
                  }
                  return @ini;
            }
            if ( $file_time > $row_refs[ 0 ]->{ 'lastUpdate' } ) {
                  # have to update
                  my ( @ini_escaped ) = map {
                        ( my $x = $_ ) =~ s/\'/\'\'/g;
                        $x;
                  } @ini;
                  $dbh->ct_sql( "update mcServices set config =
'@ini_escaped', lastUpdate = $file_time where service =
'$service'" );
                  if ( $dbh->{ RC } == CS_FAIL ) {
```

```
                        print STDERR "webget.pl: DB update to
mcServices failed\n";
                  }
            }
            return @ini;
      }
      # then try to read datatable
      die "webget.pl: Unable to find service $service\n" unless
( $DB_SRV );
      my $dbh = new Sybase::CTlib $DB_USR, $DB_PWD, $DB_SRV;
      die "webget.pl: Cannot connect to dataserver $DB_SRV:
$DB_USR:$DB_PWD\n" unless ( $dbh );
      my @row_refs = $dbh->ct_sql( "select config from mcServices
where service = '$service'", undef, 1 );
      die "webget.pl: DB select from mcServices failed\n" if
$dbh->{ RC } == CS_FAIL;
      die "webget.pl: Unable to find service $service\n" unless
( defined @row_refs );
      $row_refs[ 0 ]->{ 'config' } =~ s/\n /\n\r/g;
      @ini = split( /\r/, $row_refs[ 0 ]->{ 'config' } );
      return @ini;
}
#######################################################
sub process_section {
      my ( $prev_section ) = @_;
      my ( $section, $output, $content );
      my %Param;
      my %Content;

      foreach ( @ini ) {
            chop;
            s/\s+$//;
            s/^\s+//;
            # get section name
            if ( /^\[(.*)\]/ ) {
                  last if $section;
                  next if $1 eq "print";
                  next if $prev_section ne "" and $prev_section ne
$1;
                  if ( $prev_section eq $1 ) {
                        $prev_section = "";
                        next;
                  }
                  $section = $1;
            }
            # get parameters
            push( @{ $Param{ $1 } }, $2 ) if $section and
/([^=]+)=(.*)/;
      }
      return 0 unless $section;
#     print "section $section\n";
      # substitute parameters with values
```

```
      map { $Param{ URL }->[ 0 ] =~ s/$Param{ Input }->[ $_ ]/
$ARGV[ $_ ]/g
      } 0 .. $#{ $Param{ Input } };

      # get page content
      ( $Content{ 'TIME' }, $content ) =
&get_url_content( ${ $Param{ URL } }[ 0 ] );

      # filter it
      map {
            if ( /\"([^\"]+)\"([^\"]*)\"/ or /\/([^\/]+)\/([^
\/]*)\// ) {
                  my $out = $2; $content =~ s/$1/$out/g;
            }
      } @{ $Param{ "Pre-filter" } };

      # do main regular expression
      unless( @values = $content =~ /
${ $Param{ Regular_expression } }[ 0 ]/ ) {
            &die_hard( ${ $Param{ Regular_expression } }[ 0 ],
$content );
            return $section;
      }

      %Content = map { ( $Param{ Output }->[ $_ ], $values[ $_ ] )
      } 0 .. $#{ $Param{ Output } };

      # filter it
      map {
            if ( /([^\"]+)\"([^\"]+)\"([^\"]*)\"/
                  or /([^\/]+)\/([^\/]+)\/([^\/]*)\// ) {
                   my $out = $3;
                  $Content{ $1 } =~ s/$2/$out/g;
            }
      } @{ $Param{ "Post-filter" } };

      # calculate it
      map {
            if ( /([^=]+)=(.*)/ ) {
                  my $eval = $2;
                  map { $eval =~ s/$_/$Content{ $_ }/g
                  } keys %Content;
             $Content{ $1 } = eval( $eval );
            }

      } @{ $Param{ Calculate } };

      # read section [print]
    foreach $i ( 0 .. $#ini ) {
            next unless $ini[ $i ] =~ /^\[print\]/;
            foreach ( $i + 1 ..  $#ini ) {
               last if $ini[ $_ ] =~ /^\[.+\]/;
```

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.