```
#!/usr/local/bin/sybperl5
#-T
# -w
# $Header: /usr/local/cvsroot/webley/agents/www/webget.pl,v 1.9
1999/12/13 22:54:20 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;
#print "Sybase::CTlib $DB_USR, $DB_PWD, $DB_SRV;";
open( STDERR, ">>$0.log" );
$log = `date`;
chop( $log );

$debug2 = 0;

my $service = shift;
$log .= " $service: ". join( ':', @ARGV ) . "\n";
print STDERR $log;
my @ini = &read_ini( $service );
chop( @ini );
my $section = "";
do { $section = &process_section( $section ) } while $section;
exit;
############################################################
sub read_ini {
     my ( $service ) = @_;
     my @ini = ();
     # first, try to read file
     $0 =~ m|^(.*/)[^/]*|;
     $service = $1 . $service;
     if ( open( INI, "$service.ini" ) ) {
          @ini = ( <INI> );
          return @ini unless ( $DB_SRV );
          # update datatable
          my $file_time = time - int( ( -M "$service.ini" ) * 24
* 3600 );
#         print "time $file_time\n";
          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;
          }
#         print "time $file_time:".$row_refs[ 0 ]->
{ 'lastUpdate' }."\n";
          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;
     }
     else {
          print STDERR "$0: WARNING: $service.ini n/a in " .
`pwd`
                . "Try to read DB\n";
     }
     # 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, $content );
      my %Param;
      my %Content;
#     print"##################################\n";
      foreach ( @ini ) {
#           print;
#           chop;
            s/\s+$//;
            s/^\s+//;
            # get section name
            if ( /^\[(.*)\]/ ) {
#                 print "$_: $section:$prev_section\n";
                  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
/([^=]+)=(.*)/;
      }
#     print"++++++++++++++++++++++++++++++++++\n";
      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[ $_ ] =~ /^\[.+\]/;
             $output .= $ini[ $_ ] . "\n";
            }
            last;
      }

      # prepare output
      map { $output =~ s/$_/$Content{ $_ }/g
      } keys %Content;
```

```
      print $output;
      return 0;
}
############################################################
sub get_url_content {
      my ( $url ) = @_;
      my $ua = LWP::UserAgent->new;
      $ua->agent( 'Mozilla/4.0 [en] (X11; I; FreeBSD 2.2.8-STABLE
i386)' );
      $ua->proxy( ['http', 'https'],
'http://proxy.webley:3128/' );
      $ua->no_proxy( 'webley', 'vail' );
      my $cookie = HTTP::Cookies->new;
      $ua->cookie_jar( $cookie );
      $url = url $url;
      print "$url\n" if $debug2;
      my $time = time;
      my $res = $ua->request( GET $url );
      print "Response: " . ( time - $time ) . "sec\n" if $debug2;
      return( $time - time, $res->content );
}
############################################################
sub die_hard {
      my( $re, $content ) = @_;
      my ( $re_end, $pattern );
      while( $content !~ /$re/ ) {
            if ( $re =~ s/(\([^\(\)]+\)[^\(\)]*$)// ) {
                  $re_end = $1 . $re_end;
            }
            else {
                  $re_end = $re;
                  last;
            }
      }
      $content =~ /$re/;
      print STDERR "The regular expression did not match:\n
$re\n
Possible misuse:
$re_end:\n
Matched:
$&\n
Mismatched:
$'\n
";
    print STDERR "Content:\n $content\n" unless $';
}
############################################################
```