## NAME

LWP::Simple – simple procedural interface to LWP

## SYNOPSIS

```
perl -MLWP::Simple -e 'getprint "http://www.sn.no"'

use LWP::Simple;
$content = get("http://www.sn.no/");
die "Couldn't get it!" unless defined $content;

if (mirror("http://www.sn.no/", "foo") == RC_NOT_MODIFIED) {
    ...
}

if (is_success(getprint("http://www.sn.no/"))) {
    ...
}
```

## DESCRIPTION

This module is meant for people who want a simplified view of the libwww-perl library. It should also be suitable for one-liners. If you need more control or access to the header fields in the requests sent and responses received, then you should use the full object-oriented interface provided by the LWP::UserAgent module.

The module will also export the LWP::UserAgent object as `$ua` if you ask for it explicitly.

The user agent created by this module will identify itself as `LWP::Simple/#.##` and will initialize its proxy defaults from the environment (by calling `$ua->env_proxy`).

## FUNCTIONS

The following functions are provided (and exported) by this module:

### get

```
my $res = get($url);
```

The **get()** function will fetch the document identified by the given URL and return it. It returns `undef` if it fails. The `$url` argument can be either a string or a reference to a URI object.

You will not be able to examine the response code or response headers (like `Content-Type`) when you are accessing the web using this function. If you need that information you should use the full OO interface (see LWP::UserAgent).

### head

```
my $res = head($url);
```

Get document headers. Returns the following 5 values if successful: ($content_type, `$document_length`, `$modified_time`, `$expires`, `$server`)

Returns an empty list if it fails. In scalar context returns TRUE if successful.

### getprint

```
my $code = getprint($url);
```

Get and print a document identified by a URL. The document is printed to the selected default filehandle for output (normally STDOUT) as data is received from the network. If the request fails, then the status code and message are printed on STDERR. The return value is the HTTP response code.

### getstore

```
my $code = getstore($url, $file)
```

Gets a document identified by a URL and stores it in the file. The return value is the HTTP response code.

### mirror

```
    my $code = mirror($url, $file);
```

Get and store a document identified by a URL, using *If-modified-since*, and checking the *Content-Length*. Returns the HTTP response code.

## STATUS CONSTANTS

This module also exports the HTTP::Status constants and procedures. You can use them when you check the response code from "getprint" in LWP::Simple, "getstore" in LWP::Simple or "mirror" in LWP::Simple. The constants are:

```
   RC_CONTINUE
   RC_SWITCHING_PROTOCOLS
   RC_OK
   RC_CREATED
   RC_ACCEPTED
   RC_NON_AUTHORITATIVE_INFORMATION
   RC_NO_CONTENT
   RC_RESET_CONTENT
   RC_PARTIAL_CONTENT
   RC_MULTIPLE_CHOICES
   RC_MOVED_PERMANENTLY
   RC_MOVED_TEMPORARILY
   RC_SEE_OTHER
   RC_NOT_MODIFIED
   RC_USE_PROXY
   RC_BAD_REQUEST
   RC_UNAUTHORIZED
   RC_PAYMENT_REQUIRED
   RC_FORBIDDEN
   RC_NOT_FOUND
   RC_METHOD_NOT_ALLOWED
   RC_NOT_ACCEPTABLE
   RC_PROXY_AUTHENTICATION_REQUIRED
   RC_REQUEST_TIMEOUT
   RC_CONFLICT
   RC_GONE
   RC_LENGTH_REQUIRED
   RC_PRECONDITION_FAILED
   RC_REQUEST_ENTITY_TOO_LARGE
   RC_REQUEST_URI_TOO_LARGE
   RC_UNSUPPORTED_MEDIA_TYPE
   RC_INTERNAL_SERVER_ERROR
   RC_NOT_IMPLEMENTED
   RC_BAD_GATEWAY
   RC_SERVICE_UNAVAILABLE
   RC_GATEWAY_TIMEOUT
   RC_HTTP_VERSION_NOT_SUPPORTED
```

## CLASSIFICATION FUNCTIONS

The HTTP::Status classification functions are:

### is_success

```
    my $bool = is_success($rc);
```

True if response code indicated a successful request.

**is_error**

```
    my $bool = is_error($rc)
```

True if response code indicated that an error occurred.

## CAVEAT

Note that if you are using both LWP::Simple and the very popular CGI module, you may be importing a `head` function from each module, producing a warning like `Prototype mismatch: sub main::head ($) vs none`. Get around this problem by just not importing LWP::Simple's `head` function, like so:

```
        use LWP::Simple qw(!head);
        use CGI qw(:standard);  # then only CGI.pm defines a head()
```

Then if you do need LWP::Simple's `head` function, you can just call it as `LWP::Simple::head($url)`.

## SEE ALSO

LWP, lwpcook, LWP::UserAgent, HTTP::Status, lwp-request, lwp-mirror