NAME
HTTP::Cookies – HTTP cookie jars

VERSION
version 6.08

SYNOPSIS
```
  use HTTP::Cookies;
  $cookie_jar = HTTP::Cookies->new(
    file => "$ENV{'HOME'}/lwp_cookies.dat",
    autosave => 1,
  );

  use LWP;
  my $browser = LWP::UserAgent->new;
  $browser->cookie_jar($cookie_jar);
```

Or for an empty and temporary cookie jar:

```
  use LWP;
  my $browser = LWP::UserAgent->new;
  $browser->cookie_jar( {} );
```

DESCRIPTION
This class is for objects that represent a "cookie jar" — that is, a database of all the HTTP cookies that a given LWP::UserAgent object knows about.

Cookies are a general mechanism which server side connections can use to both store and retrieve information on the client side of the connection. For more information about cookies refer to Cookie Spec <http://curl.haxx.se/rfc/cookie_spec.html> and Cookie Central <http://www.cookiecentral.com>. This module also implements the new style cookies described in RFC 2965 <https://tools.ietf.org/html/rfc2965>. The two variants of cookies are supposed to be able to coexist happily.

Instances of the class *HTTP::Cookies* are able to store a collection of Set–Cookie2: and Set-Cookie: headers and are able to use this information to initialize Cookie-headers in *HTTP::Request* objects. The state of a *HTTP::Cookies* object can be saved in and restored from files.

LIMITATIONS
This module does not support Public Suffix <https://publicsuffix.org/> encouraged by a more recent standard, RFC 6265 <https://tools.ietf.org/html/rfc6265>.

This module's shortcomings mean that a malicious Web site can set cookies to track your user agent across all sites under a top level domain. See *t/publicsuffix.t* in this module's distribution for details.

HTTP::CookieJar::LWP supports Public Suffix, but only provides a limited subset of this module's functionality and does not support standards older than *RFC 6265*.

METHODS
The following methods are provided:

`$cookie_jar` = HTTP::Cookies–>new

The constructor takes hash style parameters. The following parameters are recognized:

```
  file:            name of the file to restore cookies from and save cookies to
  autosave:        save during destruction (bool)
  ignore_discard:  save even cookies that are requested to be discarded (bool)
  hide_cookie2:    do not add Cookie2 header to requests
```

Future parameters might include (not yet implemented):

```
  max_cookies               300
  max_cookies_per_domain    20
  max_cookie_size           4096

  no_cookies   list of domain names that we never return cookies to
```

`$cookie_jar`->get_cookies(`$url_or_domain`)
`$cookie_jar`->get_cookies(`$url_or_domain`, `$cookie_key`,...)

Returns a hash of the cookies that applies to the given URL. If a domainname is given as argument, then a prefix of "https://" is assumed.

If one or more `$cookie_key` parameters are provided return the given values, or `undef` if the cookie isn't available.

`$cookie_jar`->add_cookie_header(`$request`)

The **add_cookie_header()** method will set the appropriate Cookie:–header for the *HTTP::Request* object given as argument. The `$request` must have a valid url attribute before this method is called.

`$cookie_jar`->extract_cookies(`$response`)

The **extract_cookies()** method will look for Set-Cookie: and Set–Cookie2: headers in the *HTTP::Response* object passed as argument. Any of these headers that are found are used to update the state of the `$cookie_jar`.

`$cookie_jar`->set_cookie(`$version`, `$key`, `$val`, `$path`, `$domain`, `$port`, `$path_spec`, `$secure`, `$maxage`, `$discard`, \%rest)

The **set_cookie()** method updates the state of the `$cookie_jar`. The `$key`, `$val`, `$domain`, `$port` and `$path` arguments are strings. The `$path_spec`, `$secure`, `$discard` arguments are boolean values. The `$maxage` value is a number indicating number of seconds that this cookie will live. A value of `$maxage` <= 0 will delete this cookie. The `$version` argument sets the version of the cookie; the default value is 0 (original Netscape spec). Setting `$version` to another value indicates the RFC to which the cookie conforms (e.g. version 1 for RFC 2109). `%rest` defines various other attributes like "Comment" and "CommentURL".

`$cookie_jar`->save
`$cookie_jar`->save(`$file`)
`$cookie_jar`->save(file => `$file`, ignore_discard => `$ignore_discard`)

This method file saves the state of the `$cookie_jar` to a file. The state can then be restored later using the **load()** method. If a filename is not specified we will use the name specified during construction. If the `$ignore_discard` value is true (or not specified, but attribute *ignore_discard* was set at cookie jar construction), then we will even save cookies that are marked to be discarded.

The default is to save a sequence of "Set–Cookie3" lines. "Set–Cookie3" is a proprietary LWP format, not known to be compatible with any browser. The *HTTP::Cookies::Netscape* sub-class can be used to save in a format compatible with Netscape.

`$cookie_jar`->load
`$cookie_jar`->load(`$file`)

This method reads the cookies from the file and adds them to the `$cookie_jar`. The file must be in the format written by the **save()** method.

`$cookie_jar`->revert

This method empties the `$cookie_jar` and re-loads the `$cookie_jar` from the last save file.

`$cookie_jar`->clear
`$cookie_jar`->clear(`$domain`)
`$cookie_jar`->clear(`$domain`, `$path`)
`$cookie_jar`->clear(`$domain`, `$path`, `$key`)

Invoking this method without arguments will empty the whole `$cookie_jar`. If given a single argument only cookies belonging to that domain will be removed. If given two arguments, cookies belonging to the specified path within that domain are removed. If given three arguments, then the cookie with the specified key, path and domain is removed.

`$cookie_jar`−>clear_temporary_cookies
Discard all temporary cookies. Scans for all cookies in the jar with either no expire field or a true `discard` flag. To be called when the user agent shuts down according to RFC 2965.

`$cookie_jar`−>scan(\&callback)
The argument is a subroutine that will be invoked for each cookie stored in the `$cookie_jar`. The subroutine will be invoked with the following arguments:

```
  0  version
  1  key
  2  val
  3  path
  4  domain
  5  port
  6  path_spec
  7  secure
  8  expires
  9  discard
 10  hash
```

`$cookie_jar`−>as_string
`$cookie_jar`−>as_string(`$skip_discardables`)
The **as_string()** method will return the state of the `$cookie_jar` represented as a sequence of "Set−Cookie3" header lines separated by "\n". If `$skip_discardables` is TRUE, it will not return lines for cookies with the *Discard* attribute.

SEE ALSO

HTTP::Cookies::Netscape, HTTP::Cookies::Microsoft

AUTHOR

Gisle Aas <gisle@activestate.com>

COPYRIGHT AND LICENSE