---+ Package == **extends** `CGI::Session::ErrorHandler`

=head1 NAME

CGI::Session - persistent session data in CGI applications

=head1 SYNOPSIS

Object initialization: use CGI::Session; $session = new CGI::Session();

$CGISESSID = $session->id();

send proper HTTP header with cookies: print $session->header();

storing data in the session $session->param('f_name', 'Sherzod'); # or $session->param(-name=>'l_name', -value=>'Ruzmetov');

flush the data from memory to the storage driver at least before your # program finishes since auto-flushing can be unreliable $session->flush();

retrieving data my $f_name = $session->param('f_name'); # or my $l_name = $session->param(-name=>'l_name');

clearing a certain session parameter $session->clear(["l_name", "f_name"]);

expire '_is_logged_in' flag after 10 idle minutes: $session->expire('is_logged_in', '+10m')

expire the session itself after 1 idle hour $session->expire('+1h');

delete the session for good $session->delete();

=head1 DESCRIPTION

CGI-Session is a Perl5 library that provides an easy, reliable and modular session management system across HTTP requests. Persistency is a key feature for such applications as shopping carts, login/authentication routines, and application that need to carry data across HTTP requests. CGI::Session does that and many more.

=head1 TRANSLATIONS

This document is also available in Japanese.

=over 4

=item o

Translation based on 4.14: http://digit.que.ne.jp/work/index.cgi?Perldoc/ja

=item o

Translation based on 3.11, including Cookbook and Tutorial: http://perldoc.jp/docs/modules/CGI-Session-3.11/

=back

=head1 TO LEARN MORE

Current manual is optimized to be used as a quick reference. To learn more both about the philosophy and CGI::Session programming style, consider the following:

=over 4

=item *

L<CGI::Session::Tutorial|CGI::Session::Tutorial> - extended CGI::Session manual. Also includes library architecture and driver specifications.

=item *

We also provide mailing lists for CGI::Session users. To subscribe to the list or browse the archives visit https://lists.sourceforge.net/lists/listinfo/cgi-session-user

=item *

B - "HTTP State Management Mechanism" found at ftp://ftp.isi.edu/in-notes/rfc2965.txt

=item *

L<CGI|CGI> - standard CGI library

=item *

L<Apache::Session|Apache::Session> - another fine alternative to CGI::Session.

=back

=head1 METHODS

Following is the overview of all the available methods accessible via CGI::Session object.

=head2 new()

=head2 new($sid)

=head2 new($query)

=head2 new($dsn, $query||$sid)

=head2 new($dsn, $query||$sid, \%dsn_args)

Constructor. Returns new session object, or undef on failure. Error message is accessible through L<errstr() - class method|CGI::Session::ErrorHandler/errstr>. If called on an already initialized session will re-initialize the session based on already configured object. This is only useful after a call to L<load()|/"load">.

Can accept up to three arguments, $dsn - Data Source Name, $query||$sid - query object OR a string representing session id, and finally, \%dsn_args, arguments used by $dsn components.

If called without any arguments, $dsn defaults to I<driver:file;serializer:default;id:md5>, $query||$sid defaults to C<< CGI->new() >>, and C<\%dsn_args> defaults to I.

If called with a single argument, it will be treated either as C<$query> object, or C<$sid>, depending on its type. If argument is a string , C<new()> will treat it as session id and will attempt to retrieve the session from data store. If it fails, will create a new session id, which will be accessible through L<id() method|/"id">. If argument is an object, L<cookie()|CGI/cookie> and L<param()|CGI/param> methods will be called on that object to recover a potential C<$sid> and retrieve it from data store. If it fails, C<new()> will create a new session id, which will be accessible through L<id() method|/"id">. C<name()> will define the name of the query parameter and/or cookie name to be requested, defaults to I.

If called with two arguments first will be treated as $dsn, and second will be treated as $query or $sid or undef, depending on its type. Some examples of this syntax are:

$s = CGI::Session->new("driver:mysql", undef); $s = CGI::Session->new("driver:sqlite", $sid); $s = CGI::Session->new("driver:db_file", $query); $s = CGI::Session->new("serializer:storable;id:incr", $sid); # etc...

Following data source components are supported:

=over 4

=item *

B - CGI::Session driver. Available drivers are L<file|CGI::Session::Driver::file>, L<db_file|CGI::Session::Driver::db_file>, L<mysql|CGI::Session::Driver::mysql> and L<sqlite|CGI::Session::Driver::sqlite>. Third party drivers are welcome. For driver specs consider L<CGI::Session::Driver|CGI::Session::Driver>

=item *

B - serializer to be used to encode the data structure before saving in the disk. Available serializers are L<storable|CGI::Session::Serialize::storable>, L<freezethaw|CGI::Session::Serialize::freezethaw> and L<default|CGI::Session::Serialize::default>. Default serializer will use L<Data::Dumper|Data::Dumper>.

=item *

B - ID generator to use when new session is to be created. Available ID generator is L<md5|CGI::Session::ID::md5>

=back

For example, to get CGI::Session store its data using DB_File and serialize data using FreezeThaw:

$s = new CGI::Session("driver:DB_File;serializer:FreezeThaw", undef);

If called with three arguments, first two will be treated as in the previous example, and third argument will be C<\%dsn_args>, which will be passed to C<$dsn> components (namely, driver, serializer and id generators) for initialization purposes. Since all the $dsn components must initialize to some default value, this third argument should not be required for most drivers to operate properly.

undef is acceptable as a valid placeholder to any of the above arguments, which will force default behavior.

=head2 load()

=head2 load($query||$sid)

=head2 load($dsn, $query||$sid)

=head2 load($dsn, $query, \%dsn_args);

Accepts the same arguments as new(), and also returns a new session object, or undef on failure. The difference is, L<new()|/"new"> can create new session if it detects expired and non-existing sessions, but C<load()> does not.

C<load()> is useful to detect expired or non-existing sessions without forcing the library to create new sessions. So now you can do something like this:

$s = CGI::Session->load() or die CGI::Session->errstr(); if ($s->is_expired) { print $s->header(), $cgi->start_html(), $cgi->p("Your session timed out! Refresh the screen to start new session!") $cgi->end_html(); exit(0); }

if ($s->is_empty) { $s = $s->new() or die $s->errstr; }

Notice, all I sessions are empty, but not all I sessions are expired!

=head2 id()

Returns effective ID for a session. Since effective ID and claimed ID can differ, valid session id should always be retrieved using this method.

=head2 param($name)

=head2 param(-name=E$name)

Used in either of the above syntax returns a session parameter set to $name or undef if it doesn't exist. If it's called on a deleted method param() will issue a warning but return value is not defined.

=head2 param($name, $value)

=head2 param(-name=E$name, -value=E$value)

Used in either of the above syntax assigns a new value to $name parameter, which can later be retrieved with previously introduced param() syntax. C<$value> may be a scalar, arrayref or hashref.

Attempts to set parameter names that start with I will trigger a warning and undef will be returned.

=head2 param_hashref()

B. Use L<dataref()|/"dataref"> instead.

=head2 dataref()

Returns reference to session's data table:

$params = $s->dataref(); $sid = $params->{_SESSION_ID}; $name= $params->{name}; # etc...

Useful for having all session data in a hashref, but too risky to update.

=head2 save_param()

=head2 save_param($query)

=head2 save_param($query, \@list)

Saves query parameters to session object. In other words, it's the same as calling L<param($name, $value)|/"param"> for every single query parameter returned by C<< $query->param() >>. The first argument, if present, should be either CGI object or any object which can provide param() method. If it's undef, defaults to the return value of L<query()|/"query">, which returns C<< CGI->new >>. If second argument is present and is a reference to an array, only those query parameters found in the array will be stored in the session. undef is a valid placeholder for any argument to force default behavior.

=head2 load_param()

=head2 load_param($query)

=head2 load_param($query, \@list)

Loads session parameters into a query object. The first argument, if present, should be query object, or any other object which can provide param() method. If second argument is present and is a reference to an array, only parameters found in that array will be loaded to the query object.

=head2 clear()

=head2 clear('field')

=head2 clear(\@list)

Clears parameters from the session object.

With no parameters, all fields are cleared. If passed a single parameter or a reference to an array, only the named parameters are cleared.

=head2 flush()

Synchronizes data in memory with the copy serialized by the driver. Call flush() if you need to access the session from outside the current session object. You should at least call flush() before your program exits.

As a last resort, CGI::Session will automatically call flush for you just before the program terminates or session object goes out of scope. This automatic behavior was the recommended behavior until the 4.x series. Automatic flushing has since proven to be unreliable, and in some cases is now required in places that worked with 3.x. For further details see:

http://rt.cpan.org/Ticket/Display.html?id=17541 http://rt.cpan.org/Ticket/Display.html?id=17299

=head2 atime()

Read-only method. Returns the last access time of the session in seconds from epoch. This time is used internally while auto-expiring sessions and/or session parameters.

=head2 ctime()

Read-only method. Returns the time when the session was first created in seconds from epoch.

=head2 expire()

=head2 expire($time)

=head2 expire($param, $time)

Sets expiration interval relative to L<atime()|/"atime">.

If used with no arguments, returns the expiration interval if it was ever set. If no expiration was ever set, returns undef. For backwards compatibility, a method named C<etime()> does the same thing.

Second form sets an expiration time. This value is checked when previously stored session is asked to be retrieved, and if its expiration interval has passed, it will be expunged from the disk immediately. Passing 0 cancels expiration.

By using the third syntax you can set the expiration interval for a particular session parameter, say I<~logged-in>. This would cause the library call clear() on the parameter when its time is up. Note it only makes sense to set this value to something I than when the whole session expires. Passing 0 cancels expiration.

All the time values should be given in the form of seconds. Following keywords are also supported for your convenience:

+-----------+---------------+

alias	meaning
s	Second
m	Minute
h	Hour
d	Day
w	Week
M	Month
y	Year

+-----------+---------------+

Examples:

$session->expire("2h"); # expires in two hours $session->expire(0); # cancel expiration $session->expire("~logged-in", "10m"); # expires '~logged-in' parameter after 10 idle minutes

Note: all the expiration times are relative to session's last access time, not to its creation time. To expire a session immediately, call L<delete()|/"delete">. To expire a specific session parameter immediately, call L<clear([$name])|/"clear">.

=head2 is_new()

Returns true only for a brand new session.

=head2 is_expired()

Tests whether session initialized using L<load()|/"load"> is to be expired. This method works only on sessions initialized with load():

$s = CGI::Session->load() or die CGI::Session->errstr; if ($s->is_expired) { die "Your session expired. Please refresh"; } if ($s->is_empty) { $s = $s->new() or die $s->errstr; }

=head2 is_empty()

Returns true for sessions that are empty. It's preferred way of testing whether requested session was loaded successfully or not:

$s = CGI::Session->load($sid); if ($s->is_empty) { $s = $s->new(); }

Actually, the above code is nothing but waste. The same effect could've been achieved by saying:

$s = CGI::Session->new($sid);

L<is_empty()|"is_empty"> is useful only if you wanted to catch requests for expired sessions, and create new session afterwards. See L<is_expired()|"is_expired"> for an example.

=head2 delete()

Deletes a session from the data store and empties session data from memory, completely, so subsequent read/write requests on the same object will fail. Technically speaking, it will only set object's status to I and will trigger L<flush()|"flush">, and flush() will do the actual removal.

=head2 find(\&code)

=head2 find($dsn, \&code)

=head2 find($dsn, \&code, \%dsn_args)

Experimental feature. Executes \&code for every session object stored in disk, passing initialized CGI::Session object as the first argument of \&code. Useful for housekeeping purposes, such as for removing expired sessions. Following line, for instance, will remove sessions already expired, but are still in disk:

The following line, for instance, will remove sessions already expired, but which are still on disk:

CGI::Session->find(sub {});

Notice, above \&code didn't have to do anything, because load(), which is called to initialize sessions inside find(), will automatically remove expired sessions. Following example will remove all the objects that are 10+ days old:

CGI::Session->find(\&purge); sub purge { my ($session) = @_ ; next if $session->is_empty; # <-- already expired?! if (($session->ctime + 3600*240) <= time()) { $session->delete() or warn "couldn't remove " . $session->id . ": " . $session->errstr; } }

B: find will not change the modification or access times on the sessions it returns.

Explanation of the 3 parameters to C<find()>:

=over 4

=item $dsn

This is the DSN (Data Source Name) used by CGI::Session to control what type of sessions you previously created and what type of sessions you now wish method C<find()> to pass to your callback.

The default value is defined above, in the docs for method C<new()>, and is

'driver:file;serializer:default;id:md5'.

Do not confuse this DSN with the DSN arguments mentioned just below, under \%dsn_args.

=item \&code

This is the callback provided by you (i.e. the caller of method C<find()>) which is called by CGI::Session once for each session found by method C<find()> which matches the given $dsn.

There is no default value for this coderef.

When your callback is actually called, the only parameter is a session. If you want to call a subroutine you already have with more parameters, you can achieve this by creating an anonymous subroutine that calls your subroutine with the parameters you want. For example:

CGI::Session->find($dsn, sub { my_subroutine(@_ , 'param 1', 'param 2') }); CGI::Session->find($dsn, sub { $coderef->(@_ , $extra_arg) });

Or if you wish, you can define a sub generator as such:

sub coderef_with_args { my ($coderef, @params) = @_ ; return sub { $coderef->(@_ , @params) }; }

CGI::Session->find($dsn, coderef_with_args($coderef, 'param 1', 'param 2'));

=item \%dsn_args

If your $dsn uses file-based storage, then this hashref might contain keys such as:

{ Directory => Value 1, NoFlock => Value 2, UMask => Value 3 }

If your $dsn uses db-based storage, then this hashref contains (up to) 3 keys, and looks like:

{ DataSource => Value 1, User => Value 2, Password => Value 3 }

These 3 form the DSN, username and password used by DBI to control access to your database server, and hence are only relevant when using db-based sessions.

The default value of this hashref is undef.

=back

B<Note:> find() is meant to be convenient, not necessarily efficient. It's best suited in cron scripts.

=head1 MISCELLANEOUS METHODS

=head2 remote_addr()

Returns the remote address of the user who created the session for the first time. Returns undef if variable REMOTE_ADDR wasn't present in the environment when the session was created.

=head2 errstr()

Class method. Returns last error message from the library.

=head2 dump()

Returns a dump of the session object. Useful for debugging purposes only.

=head2 header()

Replacement for L<CGI.pm|CGI>'s header() method. Without this method, you usually need to create a CGI::Cookie object and send it as part of the HTTP header:

$cookie = CGI::Cookie->new(-name=>$session->name, -value=>$session->id); print $cgi->header(-cookie=>$cookie);

You can minimize the above into:

print $session->header();

It will retrieve the name of the session cookie from C<$session->name()> which defaults to C<$CGI::Session::NAME>. If you want to use a different name for your session cookie, do something like following before creating session object:

CGI::Session->name("MY_SID"); $session = new CGI::Session(undef, $cgi, \%attrs);

Now, $session->header() uses "MY_SID" as a name for the session cookie.

=head2 query()

Returns query object associated with current session object. Default query object class is L<CGI.pm|CGI>.

=head2 DEPRECATED METHODS

These methods exist solely for for compatibility with CGI::Session 3.x.

=head3 close()

Closes the session. Using flush() is recommended instead, since that's exactly what a call to close() does now.

=head1 DISTRIBUTION

CGI::Session consists of several components such as L<drivers|"DRIVERS">, L<serializers|"SERIALIZERS"> and L. This section lists what is available.

=head2 DRIVERS

Following drivers are included in the standard distribution:

=over 4

=item *

L<file|CGI::Session::Driver::file> - default driver for storing session data in plain files. Full name: B<CGI::Session::Driver::file>

=item *

L<db_file|CGI::Session::Driver::db_file> - for storing session data in BerkelyDB. Requires: L. Full name: B<CGI::Session::Driver::db_file>

=item *

L<mysql|CGI::Session::Driver::mysql> - for storing session data in MySQL tables. Requires L<DBI|DBI> and L<DBD::mysql|DBD::mysql>. Full name: B<CGI::Session::Driver::mysql>

=item *

L<sqlite|CGI::Session::Driver::sqlite> - for storing session data in SQLite. Requires L<DBI|DBI> and L<DBD::SQLite|DBD::SQLite>. Full name: B<CGI::Session::Driver::sqlite>

=back

=head2 SERIALIZERS

=over 4

=item *

L<default|CGI::Session::Serialize::default> - default data serializer. Uses standard L<Data::Dumper|Data::Dumper>. Full name: B<CGI::Session::Serialize::default>.

=item *

L<storable|CGI::Session::Serialize::storable> - serializes data using L. Requires L. Full name: B<CGI::Session::Serialize::storable>.

=item *

L<freezethaw|CGI::Session::Serialize::freezethaw> - serializes data using L. Requires L. Full name: B<CGI::Session::Serialize::freezethaw>

=item *

L<yaml|CGI::Session::Serialize::yaml> - serializes data using YAML. Requires L or L<YAML::Syck>. Full name: B<CGI::Session::Serialize::yaml>

=item *

L<json|CGI::Session::Serialize::json> - serializes data using JSON. Requires L<JSON::Syck>. Full name: B<CGI::Session::Serialize::json>

=back

=head2 ID GENERATORS

Following ID generators are available:

=over 4

=item *

L<md5|CGI::Session::ID::md5> - generates 32 character long hexadecimal string. Requires L<Digest::MD5|Digest::MD5>. Full name: B<CGI::Session::ID::md5>.

=item *

L<incr|CGI::Session::ID::incr> - generates incremental session ids.

=item *

L<static|CGI::Session::ID::static> - generates static session ids. B<CGI::Session::ID::static>

=back

=head1 CREDITS

CGI::Session evolved to what it is today with the help of following developers. The list doesn't follow any strict order, but somewhat chronological. Specifics can be found in F file

=over 4

=item Andy Lester

=item Brian King Emrbbking@mac.comE

=item Olivier Dragon Edragon@shadnet.shad.caE

=item Adam Jacob Eadam@sysadminsith.orgE

=item Igor Plisco Eigor@plisco.ruE

=item Mark Stosberg

=item Matt LeBlanc Emleblanc@cpan.orgE

=item Shawn Sorichetti

=back

=head1 COPYRIGHT

=head1 PUBLIC CODE REPOSITORY

You can see what the developers have been up to since the last release by checking out the code repository. You can browse the Subversion repository from here:

http://svn.cromedome.net/

Or check it directly with C from here:

svn://svn.cromedome.net/CGI-Session

=head1 SUPPORT

If you need help using CGI::Session consider the mailing list. You can ask the list by sending your questions to cgi-session-user@lists.sourceforge.net .

You can subscribe to the mailing list at https://lists.sourceforge.net/lists/listinfo/cgi-session-user .

Bug reports can be submitted at http://rt.cpan.org/NoAuth/ReportBug.html?Queue=CGI-Session

=head1 AUTHOR

Sherzod Ruzmetov Esherzodr@cpan.orgE, http://author.handalak.com/

Mark Stosberg became a co-maintainer during the development of 4.0. C<markstos@cpan.org>.

=head1 SEE ALSO

=over 4

=item *

L<CGI::Session::Tutorial|CGI::Session::Tutorial> - extended CGI::Session manual

=item *

B - "HTTP State Management Mechanism" found at ftp://ftp.isi.edu/in-notes/rfc2965.txt

=item *

L<CGI|CGI> - standard CGI library

=item *

L<Apache::Session|Apache::Session> - another fine alternative to CGI::Session

=back

This topic: TWiki > CGISessionDotPm
Topic revision: r1 - 2008-01-22 - TWikiContributor