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

=head1 NAME

CGI::Session::Driver - CGI::Session driver specifications

=head1 WARNING

Version 4.0 of CGI::Session's driver specification is B backward compatible with previous specification. If you already have a driver developed to work with the previous version you're highly encouraged to upgrade your driver code to make it compatible with the current version. Fortunately, current driver specs are a lot easier to adapt to.

If you need any help converting your driver to meet current specs, send me an e-mail. For support information see L<CGI::Session|CGI::Session>

=head1 SYNOPSIS

require CGI::Session::Driver; @ISA   = qw(CGI::Session::Driver);

=head1 DESCRIPTION

CGI::Session::Driver is a base class for all CGI::Session's native drivers. It also documents driver specifications for those willing to write drivers for different databases not currently supported by CGI::Session.

=head1 WHAT IS A DRIVER

Driver is a piece of code that helps CGI::Session library to talk to specific database engines, or storage mechanisms. To be more precise, driver is a F<.pm> file that inherits from CGI::Session::Driver and defines L<retrieve()|/"retrieve($self, $sid)">, L<store()|/"store($self, $sid, $datastr)"> and L<remove()|/"remove($self, $sid)"> methods.

=head2 BLUEPRINT

The best way of learning the specs is to look at a blueprint of a driver:

package CGI::Session::Driver::your_driver_name; use strict; use base qw(CGI::Session::Driver CGI::Session::ErrorHandler);

sub init { my ($self) = @_   ; # optional }

sub DESTROY { my ($self) = @_   ; # optional }

sub store { my ($self, $sid, $datastr) = @_   ; # Store $datastr, which is an already serialized string of data. }

sub retrieve { my ($self, $sid) = @_   ; # Return $datastr, which was previously stored using above store() method. # Return $datastr if $sid was found. Return 0 or "" if $sid doesn't exist }

sub remove { my ($self, $sid) = @_   ; # Remove storage associated with $sid. Return any true value indicating success, # or undef on failure. }

sub traverse { my ($self, $coderef) = @_   ; # execute $coderef for each session id passing session id as the first and the only # argument }

1;

All the attributes passed as the second argument to CGI::Session's new() or load() methods will automatically be made driver's object attributes. For example, if session object was initialized as following:

$s = CGI::Session->new("driver:your_driver_name", undef, {Directory=>'/tmp/sessions'});

You can access value of 'Directory' from within your driver like so:

sub store { my ($self, $sid, $datastr) = @_ ; my $dir = $self->{Directory}; # <-- in this example will be '/tmp/sessions' }

Optionally, you can define C<init()> method within your driver to do driver specific global initialization. C<init()> method will be invoked only once during the lifecycle of your driver, which is the same as the lifecycle of a session object.

For examples of C<init()> look into the source code of native CGI::Session drivers.

=head1 METHODS

This section lists and describes all driver methods. All the driver methods will receive driver object ($self) as the first argument. Methods that pertain to an individual session (such as C<retrieve()>, C<store()> and C<remove()>) will also receive session id ($sid) as the second argument.

Following list describes every driver method, including its argument list and what step of session's life they will be invoked. Understanding this may help driver authors.

=over 4

=item retrieve($self, $sid)

Called whenever a specific session is requested either via C<< CGI::Session->new() >> or C<< CGI::Session->load() >> syntax. Method should try to retrieve data associated with C< $sid > and return it. In case no data could be retrieved for C< $sid > 0 (zero) or "" should be returned. undef must be returned only to signal error. Error message should be set via set_error(), which can be inherited from L<CGI::Session::ErrorHandler|CGI::Session::ErrorHandler>.

Tip: set_error() always returns undef. Use it for your advantage.

=item store($self, $sid, $datastr)

Called whenever modified session data is to be stored back to disk. This happens whenever CGI::Session->flush() is called on modified session. Since CGI::Session->DESTROY() calls flush(), store() gets requested each time session object is to be terminated.

C< store() > is called both to store new sessions and to update already stored sessions. It's driver author's job to figure out which operation needs to be performed.

$datastr, which is passed as the third argument to represents B session data that needs to be saved.

store() can return any true value indicating success or undef on failure. Error message should be passed to set_error()

=item remove($self, $sid)

Called whenever session data is to be deleted, which is when CGI::Session->delete() is called. Should return any true value indicating success, undef on failure. Error message should be logged in set_error().

=item traverse($self, \&coderef)

Called only from within CGI::Session->find(). Job of traverse() is to call \&coderef for every single session stored in disk passing session's id as the first and only argument: C<< $coderef->($sid) >>

=item init($self)

Optional. Called whenever driver object is to be initialized, which happens only once during the lifecycle of CGI::Session object. Here you can do driver-wide initialization, such as to open connection to a database server.

=item DESTROY($self)

Optional. Perl automatically calls this method on objects just before they are to be terminated. This gives your driver chance to close any database connections or close any open file handles.

=back

=head2 NOTES

=over 4

=item *

All driver F<.pm> files must be lowercase!

=item *

DBI-related drivers are better off using L<CGI::Session::Driver::DBI|CGI::Session::Driver::DBI> as base, but don't have to.

=back

=head1 LICENSING

For support and licensing see L<CGI::Session|CGI::Session>.

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