NAME

PSP::Engine - $psp object class and primary engine of the Perl Script Pages

SYNOPSIS

```
use PSP::Engine;
my $psp = new PSP::Engine(
  conf    => new PSP::Config(),
  cgi     => new PSP::CGI(),
  no_cgi  => 1,
  file    => 'file.psp',
  execute => \$text_string,
  args    => $args
 );

my $success = $psp->execute(\$text_string,'filename',$args);

my $success = $psp->file('file.psp',$args);

$psp->print($text);

$psp->printError($text,$filename);

my $cgi=$psp->cgi();

$value = $psp->setvar('variable',$value);
my $value = $psp->var('variable');

$psp->var->{variable} = $value;
my $value = $psp->var->{variable};

my $var = $psp->var();
$var->{variable} = $value;
my $value = $var->{variable};

my @vars = $psp->vars();

my $text = $psp->flush();

my $errors = $psp->flushErrors();
```

DESCRIPTION

This module provides the engine for the Perl Script Pages (PSP). PSP code is executed within a PSP::Engine object and the object exports itself to the PSP code being executed.

OVERVIEW

Object Constructors

new

Creates a new object of the PSP::Engine class. The new constructor is the only object constructor for this module. See *new object constructor* for details.

Object Methods

execute

Executes PSP code from a string reference. See *execute method* for details.

file

Loads PSP code directly from a file, changes directory to the file's directory and executes the code. See *file method* for details.

print

Appends data to the PSP output. See *print method* for details.

printError

Appends an error message to the PSP error output. See *printError method* for details.

cgi

Returns a reference to the PSP::CGI object associated to the object, if present. See *cgi method* and *PSP::CGI* for details.

setvar

Stores a value for a given variable name in the object. This is used for exchanging data between PSP code instances. See *setvar method* for details.

var

Either returns the value stored for a named variable inside the object, or a reference to the variables hash if variable name is omitted. See *var method* for details.

vars

Returns an array of variable names in use. See *vars method* for details.

flush

Returns the object's output data and resets the data storage. See *flush method* for details.

flushErrors

Returns the object's error messages and resets the error message storage. See *flushErrors method* for details.

SYNTAX

new Object Constructor

Creates a new object of the PSP::Engine class.

```
use PSP::Engine;
my $psp = new PSP::Engine(
  conf    => new PSP::Config(),
  cgi     => new PSP::CGI(),
  no_cgi  => 1,
  file    => 'file.psp',
  execute => \$text_string,
  args    => $args
 );
```

Required Parameters

None.

Optional Parameters

The following parameters are hash key/value parameters.

conf

May be an object of *PSP::Config*. If omitted, a new object will be created.

cgi

May be an object of *PSP::CGI*. If omitted, a new object will be created.

no_cgi

If set to a true value and cgi parameter is omitted, no *PSP::CGI* object will be created.

file

May be a PSP file to parse and execute. See the *file method* for details.

execute

May be a reference to a test string containing PSP code to parse and execute. execute is ignored if the file parameter is given. See the *execute method* for details.

args

May be a reference to i.e. an array of arguments to pass to a PSP string or file that is executed. See *file method* or *execute method* for details.

execute Method

Executes PSP code from a string reference.

```
my $success = $psp->execute(\$text_string,$filename,$args);
```

Required Parameters

\$text_string

A reference to a text string containing PSP code.

Optional Parameters

$filename

A text string containing a filename for the code being executed. It is only used for constructing an error message if execution fails and may be undef.

$args

If set, it will be available to the PSP code executed via the $args variable. It is intended to supply arguments to the code, i.e. via an array or hash reference. For example $psp->execute(\$data,[$a,$b]) will execute the PSP code stored in $data. The code can access the first argument ($a) via $args->[0] and the second one ($b) via $args->[1].

Return Values

$success

The execute method returns either 1 or undef to declare success or failure of code execution.

file Method

Loads PSP code directly from a file, changes directory to the file's directory and executes the code.

```
my $success = $psp->file('file.psp',$args);
```

Required Parameters

$file

A text string containing a filename to load the PSP code from for being executed.

Please note that paths/filenames with leading slash ('/') are not absolute to the filesystem but relative to the BaseDirectory, configured through PSP::Config! You may set BaseDirectory to '' if you wish to have "real" absolute paths.

Optional Parameters

$args

If set, it will be available to the PSP code executed via the $args variable. It is intended to supply arguments to the code, i.e. via an array or hash reference. For example $psp->file('file.psp',[$a,$b]) will execute file.psp, the code within file.psp can access the first argument ($a) via $args->[0] and the second one ($b) via $args->[1].

Return Values

$success

The file method returns either 1 or undef to declare success or failure of code execution.

print Method

Appends data to the PSP output.

```
$psp->print($text);
$psp->print(@text);
```

Required Parameters

$text or @text

One or more text strings to append to PSP output.

Optional Parameters

None.

Return Values

None.

printError Method

Appends an error message to the PSP error output.

```
$psp->printError($text,$filename);
```

At the moment, errors printed through this method are also appended to the PSP output. This may change (i.e. become optional) in future versions.

Required Parameters

$text

Text string to append to PSP error messages.

Optional Parameters

$filename

If given, the error message will be prepended with this filename.

Return Values

None.

cgi Method

Returns a reference to the PSP::CGI object associated to the object, if present.

```
my $cgi = psp$->cgi();
```

Required Parameters

None.

Optional Parameters

None.

Return Values

$cgi

A reference to the *PSP::CGI* object associated to the object, if present. See *PSP::CGI* for details.

setvar Method

Stores a value for a given variable name in the object. This is used for exchanging data between PSP code instances.

```
my $varvalue = psp$->setvar($varname,$varvalue);
```

Required Parameters

$varname

The name used to identify the variable.

$varvalue

The value to set the variable given above to.

Optional Parameters

None.

Return Values

$varvalue

A copy of the $varvalue parameter.

var Method

Either returns the value stored for a named variable inside the object, or a reference to the variables hash if variable name is omitted.

```
my $varvalue = psp$->var($varname);

my $var = psp$->var();
$var->{$varname} = $varvalue;
my $varvalue = $var->{$varname};
```

Required Parameters

None.

Optional Parameters

$varname

The name used to identify the variable.

Return Values

$varvalue

The value stored in the object for a named variable, undef if no such variable exists.

$var

A reference to the variables hash if $varname was omitted/undef.

vars Method

Returns an array of variable names in use.

```
my @vars = $psp->vars();
```

Required Parameters

None.

Optional Parameters

None.

Return Values

@vars

Array of variable names/hash search keys used to store data.

flush Method

Returns the object's output data and resets the data storage.

```
my $text = $psp->flush();
```

Required Parameters

None.

Optional Parameters

None.

Return Values

$text

The value of the output data queue.

flushErrors Method

Returns the object's error messages and resets the error message storage.

```
my $errros = $psp->flushErrors();
```

Required Parameters

None.

Optional Parameters

None.

Return Values

$errors

The value of the error messages queue.

USAGE INSTRUCTIONS

Object Construction

When constructing a PSP::Engine object, it expects a PSP::Config and a PSP::CGI object as hash parameters:

```
my $psp = new PSP::Engine(
  conf    => new PSP::Config(),
  cgi     => new PSP::CGI()
 );
```

If one or both objects are omitted, the "missing" object(s) will be created.

See also *PSP::Config* and *PSP::CGI*.

The no_cgi => 1 option prevents the PSP::Engine constructor from creating a new PSP::CGI object:

```
my $psp = new PSP::Engine(
  no_cgi  => 1
 );
```

This is useful if PSP is not used for CGI but for i.e. templating tasks.

Both the *file method* and the *execute method* may be used directly from the object constructor for comfort:

```
my $psp = new PSP::Engine(
  file    => 'file.psp',
  args    => $args
 );

my $psp = new PSP::Engine(
  execute => \$text_string,
  args    => $args
 );
```

Executing PSP Code

PSP code may either be loaded from a file using the *file method* or directly from a text string reference using the *execute method*:

```
my $success = $psp->execute(\$text_string,'filename',$args);

my $success = $psp->file('file.psp',$args);
```

Both methods return either 1 or undef to declare success or failure of code execution.

The $args parameter is optional. If set, it will be available to the PSP code executed via the $args variable. It is intended to supply arguments to the code, i.e. via an array or hash reference. For example $psp->file('file.psp',[$a,$b]) will execute file.psp, the code within file.psp can access the first argument ($a) via $args->[0] and the second one ($b) via $args->[1].

Please note that paths/filenames with leading slash ('/') are not absolute to the filesystem but relative to the BaseDirectory, configured through PSP::Config! You may set BaseDirectory to '' if you wish to have "real" absolute paths.

The second parameter for the *execute method*, 'filename', is optional and may be undef. It is

only used for constructing an error message if execution fails.

You can use the *execute method* and *file method* more than once, resulting output is appended.

Printing Data to PSP Output

Using the *print method*, you may append data to the PSP output to be flushed later:

```
$psp->print($text);
```

For error messages, there is a *printError method*, also accepting a text string as parameter:

```
$psp->printError($text,$filename);
```

The $filename parameter is optional and may be omitted. If given, the error message will be prepended with this filename.

At the moment, errors printed through the *printError method* are also appended to the PSP output. This may change (i.e. become optional) in future versions.

Resulting Output

After PSP code execution, the resulting output is to be flushed from the object to a string using the *flush method*:

```
my $text = $psp->flush();
```

The same applies to code execution error messages via the *flushErrors method*:

```
my $errors = $psp->flushErrors();
```

The object can be re-used after flushing.

AUTHOR

Veit Wahlich

E-Mail: cru |at| zodia |dot| de

WWW: *http://home.ircnet.de/cru/*

VERSION

v0.7 Wednesday, 18 January 2006

COPYRIGHT/LICENSE