Filters

Last Updated Tuesday, 01 November 2005

Creating you own output filter

An output filter has to be a class that extends from patTemplate_OutputFilter. You have to place the file that conatains your filter in the folder patTemplate/OutputFilter.

In this class you only need to implement one method called 'apply()'. This method will be called by patTemplate when patTemplate::displayParsedTemplate() is called by the script. Before the resulting HTML code is sent to the browser, your filter will have the oportunity to modify or filter the resulting HTML.

The apply method has to accept one string parameter, in which it will receive the HTML code. After modifying it, you just have to return to modified HTML.
A simple example

The following example shows how to implement an output filter, that removes all linebreaks and extra spaces, indentations, etc. from the HTML code before sending it to the browser. The class has to be placed in the file patTemplate/Output Filter / Strip Whitespace.php? (actually it already is there as this example is included in the distribution).

```
<?PHP
/**
* patTemplate StripWhitespace output filter
*
* Will remove all whitespace and replace it with a single space.
*
* @package       patTemplate
* @subpackage    Filters
* @author        Stephan Schmidt <schst@php.net>
*/
class patTemplate_OutputFilter_StripWhitespace extends patTemplate_OutputFilter
{
   /**
    * filter name
    *
    * @access    protected
    * @abstract
    * @var    string
    */
    var    $_name    =    'StripWhitespace';

   /**
    * remove all whitespace from the output
    *
    * @access    public
    * @param    string        data
    * @return    string        data without whitespace
    */
    function apply( $data )
    {
        $data = str_replace( "n", ' ', $data );
        $data = preg_replace( '/ss+/', ' ', $data );

        return $data;
    }
}
?>
```

Applying output filters

Applying an output filter is an easy task. You just have to call one method on your patTemplate object and pass the desired output filter. You may create a filter chain by applying as many output filters as you like.

If you applied more than one filter, the will be called in the same order as you applied them.

```
<?PHP
require_once 'pat/patErrorManager.php';
require_once 'pat/patTemplate.php';

$tmpl = &new patTemplate();
$tmpl->setRoot( 'templates' );
$tmpl->applyOutputFilter( 'StripWhitespace' );

$tmpl->readTemplatesFromInput( 'page.tmpl', 'File' );

/**
* output filter will be applied here
*/
$tmpl->displayParsedTemplate();
?>
```

Passing parameters to the filter

You may also create an output filter that can be parameterised by the script that applies the filter. If the filter class needs to access the parameters set by the script, you may use the method patTemplate_OutputFilter::getParam().
When applying a filter, all parameters have to be passed as an array in the second parameter of patTemplate::applyOutputFilter().

Input Filters
Why input filters?

You may use input filters for various tasks. You could strip all whitespace from you templates to speed up the parsing process, remove unneeded comments or unpack templates if they are stored in zipped format.

You may also use it in some special cases, where you need to modify the templates but are not able to modify them in their original storage location.
Implementing an Input Filter

Implementing an Input Filter is exactly the same as creating a new Output Filter. You have to extend a new class from patTemplate_InputFilter and place it in a file in the patTemplate/InputFilter directory. The last part of the class name has to be identical to the name of the file.

In this class, you simply have to implement one method:

string patTemplate_OutputFilter::apply(string templateCode)

patTemplate_Reader will pass the template code to this method before it is analyzed by the lexer and you may modify it according to your needs.
A simple example

The following example strips HTML comments from the templates before they are analyzed. This allows you to place them between the <patTemplate:tmpl> and <patTemplate:sub> tags, although it is not allowed to place data there.

```
<?PHP
/**
* patTemplate StripComments output filter
*
* Will remove all HTML comments.
*
* @package        patTemplate
* @subpackage    Filters
* @author        Stephan Schmidt <schst@php.net>
*/
class patTemplate_InputFilter_StripComments extends patTemplate_InputFilter
{
   /**
    * filter name
    *
```

```
16          * @access private
17          * @var    string
18          */
19          var    $_name    =    'StripComments';
20
21         /**
22          * compress the data
23          *
24          * @access    public
25          * @param     string        data
26          * @return    string        data without whitespace
27          */
28          function apply( $data )
29          {
30              $data = preg_replace( '<!--.*-->msU', '', $data );
31
32              return $data;
33          }
34      }
35      ?>
```

Applying input filters

Applying an input filter is an easy task. You just have to call one method on your patTemplate object and pass the desired output filter. You may create a filter chain by applying as many input filters as you like.

If you applied more than one filter, the will be called in the same order as you applied them.

```
01      <?PHP
02      require_once 'pat/patErrorManager.php';
03      require_once 'pat/patTemplate.php';
04
05      $tmpl = &new patTemplate();
06      $tmpl->setRoot( 'templates' );
07      $tmpl->applyInputFilter( 'StripComments' );
08
09      $tmpl->readTemplatesFromInput( 'page.tmpl', 'File' );
10
11      /**
12      * output filter will be applied here
13      */
14      $tmpl->displayParsedTemplate();
15      ?>
```

Passing parameters to the filter

You may also create an input filter that can be parameterised by the script that applies the filter. If the filter class needs to access the parameters set by the script, you may use the method patTemplate_InputFilter::getParam().

When applying a filter, all parameters have to be passed as an array in the second parameter of patTemplate::applyInputFilter().