Reading from any datasource

Last Updated Friday, 31 December 2004

If you would like to store your templates anywhere else, you can simply create a new reader. Just create a new subclass of patTemplate_Reader and place it into the reader directory.

You need to implement at least one method: readTemplates(). patTemplate will call this method, whenever the user calls patTemplate::readtemplatesFromInput() and will pass the unique identifier for the template to read. This can be a filename, a URL or the value of a primary key in a database.

After reading the template, you will have to return an array, that contains the template structure. In most cases you will be able to use patTemplate_Reader::parseString(), which will apply regular expressions to split the template source into several blocks and interprets all tags.

```
01Â Â Â Â Â Â Â  <?PHP
02Â Â Â Â Â Â Â  /**
03Â Â Â Â Â Â Â  * patTemplate Reader that reads from a file
04Â Â Â Â Â Â Â  *
05Â Â Â Â Â Â Â  * @packageÂ Â Â Â Â Â  patTemplate
06Â Â Â Â Â Â Â  * @subpackageÂ Â Â  Readers
07Â Â Â Â Â Â Â  * @authorÂ Â Â Â Â Â Â  Stephan Schmidt <schst@php.net>
08Â Â Â Â Â Â Â  */
09Â Â Â Â Â Â Â  class patTemplate_Reader_File extends patTemplate_Reader
10Â Â Â Â Â Â Â  {
11Â Â Â Â Â Â Â Â Â Â Â  /**
12Â Â Â Â Â Â Â Â Â Â Â  * reader name
13Â Â Â Â Â Â Â Â Â Â Â  * @accessÂ Â Â  private
14Â Â Â Â Â Â Â Â Â Â Â  * @varÂ Â Â Â Â Â Â  string
15Â Â Â Â Â Â Â Â Â Â Â  */
16Â Â Â Â Â Â Â Â Â Â Â  var $_name ='File';
17Â Â Â Â Â Â Â
18Â Â Â Â Â Â Â Â Â Â Â  /**
19Â Â Â Â Â Â Â Â Â Â Â  * read templates from any input
20Â Â Â Â Â Â Â Â Â Â Â  *
21Â Â Â Â Â Â Â Â Â Â Â  * @final
22Â Â Â Â Â Â Â Â Â Â Â  * @accessÂ Â Â  public
23Â Â Â Â Â Â Â Â Â Â Â  * @paramÂ Â Â  stringÂ Â Â  file to parse
24Â Â Â Â Â Â Â Â Â Â Â  * @returnÂ Â Â  arrayÂ Â Â  templates
25Â Â Â Â Â Â Â Â Â Â Â  */
26Â Â Â Â Â Â Â Â Â Â Â  function readTemplates( $input )
27Â Â Â Â Â Â Â Â Â Â Â  {
28Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â  $this->_currentInput = $input;
29Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â  $fullPathÂ  = $this->_resolveFullPath( $input );
30Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â  $contentÂ Â  = file_get_contents( $fullPath );
31Â Â Â Â Â Â Â
32Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â  $templates = $this->parseString( $content );
33Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â
34Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â  return $templates;
35Â Â Â Â Â Â Â Â Â Â Â  }
36Â Â Â Â Â Â Â
37Â Â Â Â Â Â Â Â Â Â Â  /**
38Â Â Â Â Â Â Â Â Â Â Â  * resolve path for a template
39Â Â Â Â Â Â Â Â Â Â Â  *
40Â Â Â Â Â Â Â Â Â Â Â  * @accessÂ Â Â  private
41Â Â Â Â Â Â Â Â Â Â Â  * @paramÂ Â Â  stringÂ Â Â Â Â Â Â  filename
42Â Â Â Â Â Â Â Â Â Â Â  * @returnÂ Â Â  stringÂ Â Â Â Â Â Â  full path
43Â Â Â Â Â Â Â Â Â Â Â  */Â Â Â
44Â Â Â Â Â Â Â Â Â Â Â  function _resolveFullPath( $filename )
45Â Â Â Â Â Â Â Â Â Â Â  {
46Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â  $baseDirÂ  = $this->_options['root'];
47Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â  $fullPath = $baseDir . '/' . $filename;
48Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â  return $fullPath;
49Â Â Â Â Â Â Â Â Â Â Â  }
50Â Â Â Â Â Â Â  }?>
```

If you want to support the parse="off" functionality for external templates, you will have to create a second method called loadTemplate(). This method will receive a unique identifier and should return the data associated with it as a string. In the above example, you would just strip the call to parseString().
Possible readers

Â Â Â * Read from database
Â Â Â * Read from a template server
Â Â Â * Read from shared memory

Other template engines

Readers may also be used to read templates that have been created for other template enginges than patTemplate. We already delivered a reader, that is able to read templates that have been created for HTML_Template_IT and treat them like patTemplate templates.
If you want to implement a reader for other engines, you'll have to make yourself fimiliar with the internal structure of patTemplate. We'll post more information on this subject at a latter point.
Caching

patTemplate supports caching of the returned templates structures. As the caching is still in beta state and the internal plugin API may change, documentation on this topic will follow, once the API is stable enough.