Chapter 10. Template

Table of Contents

Note: If you think that this is like FastTemplates, read carefully. It isn't.

The template class allows you to keep your HTML code in some external files which are completely free of PHP code, but contain replacement fields. The class provides you with functions which can fill in the replacement fields with arbitrary strings. These strings can become very large, e.g. entire tables.

NOTE: This version of the documentation is no longer maintained. Please see the phpdoc comments in the template.inc source file for the definitive documentation.

Template Instance variables

Accessible instance variables

classname	String. Serialization helper: The name of this class.
debug	Integer, Bitmask: set 1 bit to see all variable assignments,set 2 bit to see all variable accesses,set 4 bit to see internal function references.
unknowns	One of "keep", "comment", "remove" (Default).Determines how to handle unresolved variable names intemplates upon output. If set to "keep", those are leftuntouched. If set to "comment", unresolved variable names aretransformed into HTML comments reporting the error. If set to"remove", unresolved variable names are silently removed (thedefault).
halt_on_error = "yes"	One of "yes"(Default), "report", "no". Determines how Template handleserror conditions. If set to "yes" (the Default), the error isreported, then execution is halted. If set to "report", theerror is reported, then execution continues by returning"false". If set to "no", errors are silently ignored, andexecution resumes reporting "false".
last_error = ""	The last error message iskept in this variable.

Internal instance variables

file	Hash of strings. A translation table whichtranslates variable names into filenames.

root	String (Pathname). The base directory from whichtemplate files are being loaded.
varkeys	Hash of strings. A translation table whichtranslates variable names into regular expressions forthemselves.
varvals	Hash of strings. A translation table whichtranslates variable names into replacement values for theirrespective varkeys.

Template Instance methods

Accessible Instance methods

Template($root = ".", $unknowns = "remove")

Constructor. May be called with two optional parameters. The first parameter sets the template directory (see *set_root()*, the second parameter sets the policy regarding handling of unknown variables.

set_root($root)

The function checks that $root is a valid directory and sets this directory as the base directory where templates are being stored.

set_unknowns($unknowns = "remove")

The function sets the policy for dealing with unresolved variable names. Must be either "remove", "comment" or "keep". If set to "keep", those are left untouched. If set to "comment", unresolved variable names are transformed into HTML comments reporting the error. If set to "remove", unresolved variable names are silently removed (the default).

set_file($varname, $filename = "")

The function defines a filename for the initial value of a variable. It may be called with either a $varname/$filename pair or with a hash of $varname/$filename pairs. The files are not loaded until they are needed.

set_block($parent, $varname, $name = "")

A variable $parent may contain a variable block named by $varname. The function removes that block from $parent and replaces it with a variable reference named $name. If $name is omitted, it is assumed to be the same as $varname.

set_var($varname, $value = "")

The functions sets the inital value of a variable. It may be called with either a $varname/$value pair or with a hash of $varname/$value pairs.

subst($varname)

The function returns the value of the variable named $varname, with all defined variable values filled in. The resulting string is not "finished", that is, the unresolved variable name policy has not been applied yet.

psubst($varname)

This is a shorthand for *print $this->subst($varname)*.

parse($target, $varname, $append = false)

The function substitutes the values of all defined variables in the variable named $varname and stores or appends the result in the variable named $target.

If $varname is an array of variable names, $append is ignored. The variables named by $varname are being sequentially substituted and the result of each substitution step is stored in $target. The resulting substitution is available in the variable named by $target, as is each intermediate step for the next $varname in sequence.

pparse($target, $varname, $append = false)

A shorthand for *print $this->parse(...)*.

get_vars()

Returns a hash of all defined values, keyed by their names.

query_id()

Returns the value of the variable named by $varname. If $varname references a file and that file has not been loaded, yet, the variable will be reported as empty. When called with an array of variable names, an hash of values, keyed by their names, will be returned.

get_undefined($varname)

The function will return a hash of unresolved variable names in $varname, keyed by their names (that is, the hash has the form $a[$name] = $name).

finish($str)

The function will returned the finished version of $str, that is, the policy regarding unresolved variable names will be applied to $str.

p($varname)

The function will print the finished version of the value of the variable named by $varname.

get($varname)

The function will return the finished version of the value of the variable named by $varname.

haltmsg($msg)

This function can be overridden by your subclass of Template. It will be called with an error message to print.

Internal instance methods

filename($filename)

When called with a relative pathname, this function will return the pathname with the appropriate directory from $this->root prepended. Absolute pathnames are taken unchanged.

The resulting filename must exist or an error is generated.

varname($varname)

The function will construct a variable name regexp for a given variable name.

loadfile($varname)

If a variable is undefined or empty and is backed by a filename, the backing file will be loaded and the files contents will be assigned as the variables value.

halt($msg)

This function is called whenever an error occurs and will handle the error according to the policy defined in $this->halt_on_error.

Template Examples

The class manages a set of variables which are text strings. These strings may contain references to other variables in the form of "{variable}". When parsed or substituted, a variable reference is being replaced by the value of that variable. For example, if you

```
<?php
 $t = new Template;

 $t->set_var("a", "defined as hugo");
 $t->set_var("b", "the value of a is {a}");
```

```
          print $t->subst("b")
         ?>
```

will print *the value of a is defined as hugo*.

A variable value may be defined manually by calling *set_var("name", "value");* or it may be defined from a file by calling *set_file("name","filename.ihtml");*. In the latter case, the contents of the file are being loaded when needed (as late as possible) and set as the value of that variable.

A third way to define a variable value is to call *set_block("parent", "block", "name");*. In this case, the variable named *parent* is being searched for a block that starts with *<!-- BEGIN block -->* and ends with *<!-- END block -->*. This string is removed from the variable *parent* and assigned to the variable named *block*. In *parent*, a variable reference to *name* is placed instead. If the optional parameter *"name"* is left out, *"block"* is being used instead.

For example, if you

```
        <?php
          $t = new Template;

          $t->set_var("a", "front matter

          <!-- BEGIN b -->
          this is block b
          <!-- END b -->

          end matter");
          $t->set_block("a", "b", "bb");
        ?>
```

this will define the variable *a* as *front matter {bb} end matter* and the variable *b* as *this is block b*. All of this will become more clear when you set the instance variable *debug* to 7 in the following example and trace the variable accesses.

Use *Template* direcly or define a subclass of *Template* as needed.

Define a template file named page.ihtml as follows:

```
        <html>
         <head><title>{PAGETITLE}</title></head>
         <body bgcolor="#ffffff">
         <table border=1 cellpadding=4 cellspacing=0 bgcolor="#eeeeee">
          <tr>
           <td colspan=2><h1>{PAGETITLE}</h1></td>
          </tr>
          <tr>
           <td>{OUT}</td>
           <td>Content<br>{UNDEFINED_VARIABLE}</td>
          </tr>
         </table>
         </body>
        </html>
```

This file contains a reference to the variable *PAGETITLE* and a reference to the variable named *OUT*. Another reference to the variable named *UNDEFINED_VARIABLE* is never resolved. Another template file, named box.ihtml, contains a block named *row* with three variable references {TITLE}, {NUM} and {BIGNUM}:

```
        <!-- start box.ihtml -->
        <table border=1 bgcolor="#cccccc" cellpadding=4 cellspacing=0>
         <tr>
          <td colspan=2><b>{TITLE}</b></td>
         </tr>
          <!-- BEGIN row -->
          <tr>
           <td>{NUM}</td>
           <td>{BIGNUM}
          </tr>
          <!-- END row -->
        </table>
        <!-- end box.ihtml -->
```

The following php3 file demonstrates how to use these templates:

```
        <?php
          /* include the Template class */
          include("template.inc");

          /* create an instance of Template with desired parameters */
          $t = new
Template("/home/kris/www/test.koehntopp.de/pages/template", "keep");
          /* $t->debug = 7; */ /* activate for full debugging */

          /* define two variables from files */
          $t->set_file(array(
             "page" => "page.ihtml",
             "box"  => "box.ihtml"));

          /* define a variable contained in another */
          $t->set_block("box", "row", "rows");

          /* define two variables manually */
          $t->set_var(array("TITLE"     => "Testseite",
                            "PAGETITLE" => "hugo"));

          for ($i=1; $i<=3; $i++) {
            $n  = $i;
            $nn = $i*10;

            /* define values for NUM and BIGNUM */
            $t->set_var(array("NUM" => $n, "BIGNUM" => $nn));

            /* replace NUM and BIGNUM in row and
             * append the result to rows */
            $t->parse("rows", "row", true);
          }
          /* replace all variables in box, result in OUT,
           * then replace all variables in page, result in OUT */
          $t->parse("OUT", array("box", "page"));

          /* print OUT */
          $t->p("OUT");
        ?>
        <hr>
        Undefined variables in OUT:
        <?php
          print @implode(", ", $t->get_undefined("OUT"));
         ?>
```

User Contributed Notes

Template Examples

02-Feb-2005 08:02

rating: **3.0** | rate: 1 | 2 | 3 | 4 | 5

```
You have to read the above message like this: (this works)
(you can copy the template.inc for testing, to avoid
instalation...

list_in_list.php:
<?php
include('./template.inc');


$t = new Template("./");
```