

Smarty - the compiling PHP template engine

Monte Ohrt <monte@ispi.net>
Andrei Zmievski <andrei@php.net>

Copyright © 2001, 2002, 2003 by ispi of Lincoln, Inc.

[PHP](#)

[Table of Contents](#)

[Preface](#)

I. [Getting Started](#)

1. [What is Smarty?](#) [Smarty?]

2. [Installation](#)

[Requirements](#)

[Basic Installation](#)

[Extended Setup](#)

II. [Smarty For Template Designers](#)

3. [Basic Syntax](#)

[Comments](#)

[Functions](#)

[Attributes](#)

[Embedding Vars in Double Quotes](#)

[Math](#)

4. [Variables](#)

[Variables assigned from PHP](#) [PHP]

[Variables loaded from config files](#)

[{\\$smarty} reserved variable](#) [{\$smarty}]

5. [Variable Modifiers](#)

[capitalize](#)

[count_characters](#)

[cat](#)

[count_paragraphs](#)

[count_sentences](#)

[count_words](#)

[date_format](#)

[default](#)

[escape](#)

[indent](#) []
[lower](#) []
[nl2br](#) [
]
[regex_replace](#) []
[replace](#) []
[spacify](#) []
[string_format](#) []
[strip](#) []()
[strip_tags](#) [html]
[truncate](#) []
[upper](#) []
[wordwrap](#) []

6. [Combining Modifiers](#) []

7. [Built-in Functions](#) []

[capture](#)
[config_load](#)
[foreach,foreachelse](#)
[include](#)
[include_php](#)
[insert](#)
[if,elseif,else](#)
[ldelim,rdelim](#)
[literal](#)
[php](#)
[section,sectionelse](#)
[strip](#)

8. [Custom Functions](#) []

[assign](#)
[counter](#)
[cycle](#)
[debug](#)
[eval](#)
[fetch](#)
[html_checkboxes](#)
[html_image](#)
[html_options](#)
[html_radios](#)
[html_select_date](#)
[html_select_time](#)

[html_table](#)

[math](#)

[mailto](#)

[popup_init](#)

[popup](#)

[textformat](#)

9. [Config Files](#) []

10. [Debugging Console](#) []

III. [Smarty For Programmers](#) []

11. [Constants](#) []

[SMARTY_DIR](#) [Smarty]

12. [Variables](#) []

[\\$template_dir](#) []

[\\$compile_dir](#) []

[\\$config_dir](#) []

[\\$plugins_dir](#) []

[\\$debugging](#) []

[\\$debug_tpl](#) []

[\\$debugging_ctrl](#) []

[\\$global_assign](#) []

[\\$undefined](#) []

[\\$autoload_filters](#) []

[\\$compile_check](#) []

[\\$force_compile](#) []

[\\$caching](#) []

[\\$cache_dir](#) []

[\\$cache_lifetime](#) []

[\\$cache_handler_func](#) []

[\\$cache_modified_check](#) []

[\\$config_overwrite](#) []

[\\$config_booleanize](#) []

[\\$config_read_hidden](#) []

[\\$config_fix_newlines](#) []

[\\$default_template_handler_func](#) []

[\\$php_handling](#) [php]

[\\$security](#) []

[\\$secure_dir](#) []

[\\$security_settings](#) []

[\\$trusted_dir](#) []

[\\$left_delimiter](#) []
[\\$right_delimiter](#) []
[\\$compiler_class](#) []
[\\$request_vars_order](#) []
[\\$request_use_auto_globals](#) []
[\\$compile_id](#) [id]
[\\$use_sub_dirs](#) []
[\\$default_modifiers](#) []
[\\$default_resource_type](#) []

13. [Methods](#) []
- [append](#) []
 - [append_by_ref](#) []
 - [assign](#) []
 - [assign_by_ref](#) []
 - [clear_all_assign](#) []
 - [clear_all_cache](#) []
 - [clear_assign](#) []
 - [clear_cache](#) []
 - [clear_compiled_tpl](#) []
 - [clear_config](#) []
 - [config_load](#) []
 - [display](#) []
 - [fetch](#) []
 - [get_config_vars](#) []
 - [get_registered_object](#) []
 - [get_template_vars](#) []
 - [is_cached](#) []
 - [load_filter](#) []
 - [register_block](#) []
 - [register_compiler_function](#) []
 - [register_function](#) []
 - [register_modifier](#) []
 - [register_object](#) []
 - [register_outputfilter](#) []
 - [register_postfilter](#) []
 - [register_prefilter](#) []
 - [register_resource](#) []
 - [trigger_error](#) []
 - [template_exists](#) []

[unregister_block](#) []
[unregister_compiler_function](#) []
[unregister_function](#) []
[unregister_modifier](#) []
[unregister_object](#) []
[unregister_outputfilter](#) []
[unregister_postfilter](#) []
[unregister_prefilter](#) []
[unregister_resource](#) []

14. [Caching](#) []
[Setting Up Caching](#) []
[Multiple Caches Per Page](#) []
[Cache Groups](#) []
[Controlling Cacheability of Plugins' Output](#) []

15. [Advanced Features](#) []
[Objects](#) []
[Prefilters](#) []
[Postfilters](#) []
[Output Filters](#) []
[Cache Handler Function](#) []
[Resources](#) []

16. [Extending Smarty With Plugins](#) [Smarty]
[How Plugins Work](#) []
[Naming Conventions](#) []
[Writing Plugins](#) []
[Template Functions](#) []
[Modifiers](#) []
[Block Functions](#) []
[Compiler Functions](#) []
[Prefilters/Postfilters](#) [/]
[Output Filters](#) []
[Resources](#) []
[Inserts](#) []

IV. [Appendices](#) []

17. [Troubleshooting](#) []
[Smarty/PHP errors](#) [Smarty/PHP]
18. [Tips & Tricks](#) []
[Blank Variable Handling](#) []
[Default Variable Handling](#) []

[Passing variable title to header template](#) []

[Dates](#) []

[WAP/WML](#)

[Componentized Templates](#) []

[Obfuscating E-mail Addresses](#) []

19. [Resources](#) []

20. [BUGS](#) []

21. [LIST](#)

[Next](#)

Preface

Smarty - the compiling PHP template engine

Smarty - PHP

[Priv](#)

[PHP](#)

[Next](#)

Preface[]

It is undoubtedly one of the most asked questions on the PHP mailing lists: scripts independent of the layout? While PHP is billed as "HTML embedded", writing a couple of projects that mixed PHP and HTML freely one comes up with separation of form and content is a Good Thing [TM]. In addition, in many cases layout designer and programmer are separate. Consequently, the search for a solution ensues.

PHP PHP

PHP

In our company for example, the development of an application goes on a bit like this. First the requirements docs are done, the interface designer makes mockups of the user interface, and then they hand it off to the programmer. The programmer implements business logic in PHP and then hands it off to the designer to create skeleton templates. The project is then handed off to the HTML developer who adds the missing pieces and then passes it on to the person who brings the templates up to their full glory. The project may go through several iterations of design and programming/HTML a couple of times. Thus, it's important to have good templating support. Programmers don't want anything to do with HTML and don't want HTML developers to have to deal with PHP code. Designers need support for config files, dynamic blocks and other features, but they don't want to have to deal with intricacies of the PHP programming language.

,[],PHP,htmlhtmlphpphp

Looking at many templating solutions available for PHP today, most of them either do what we wanted or do what we didn't want. Most of them do the way of substituting variables into templates and do a limited form of dynamic blocks. None of them did what we wanted, so we had to come up with something else. Our needs required a bit more than that. We didn't want programmers to be forced to learn a new language, but we also didn't want designers to be forced to learn PHP at ALL, but this was almost inevitable. For instance, if a designer wanted to add a dynamic block to a template, they would have to learn how to use PHP to do it. If we wanted to alternate on dynamic blocks, this had to be worked out with the programmers. This was a problem because programmers don't want anything to do with HTML and designers don't want to deal with PHP code. We needed designers to be able to use their own configuration files, and pull them into templates. The list goes on.

php,phphtml:,

We started out writing out a spec for a template engine back in late 1999. I began to work on a template engine written in C that would hopefully be a good solution. Not only did we run into many complicated technical barriers, but there was also a lot of debate about exactly what a template engine should do and should not do. Finally, I decided that the template engine should be written in PHP as a class, for a few reasons. One reason was that it was easier to maintain and update (it was never submitted to the public). It was a class that did almost everything we wanted it to do, and it was easy to use.

variable substitution, supported including other templates, integration with
Prev [PHP](#) Home [Next](#)
PHP code, limited 'if' statement functionality and much more robust dynamic
multiple nested. It did all this with regular expressions and the tool started
[Smarty - The Compiling PHP Template Engine](#) Getting Started
say impenetrable. It was also noticeably slow in large applications from all
expression work it had to do on each invocation. The biggest problem [or
view was all the necessary work in the PHP script to setup and process] te
blocks. How do we make this easier?

1999,,cphp

'if'

Then came the vision of what ultimately became Smarty. We know how far
overhead of template parsing. We also know how meticulous and overbearing
may look to the average designer, and this could be masked with a much
So what if we combined the two strengths? Thus, Smarty was born...

smartyphpphp""

Smarty.....

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

I. Getting Started

Table of Contents

1. [What is Smarty?](#) smaty?
 2. [Installation](#)
-

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Preface

What is Smarty?

Smarty

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

Chapter 1. What is Smarty?

.Smarty?

Smarty is a template engine for PHP. More specifically, it facilitates a manageable way to separate application logic and content from its presentation. This is best described in a situation where the application programmer and the template designer play different roles, or in most cases are not the same person. For example, let's say you are creating a web page that is displaying a newspaper article. The article headline, tagline, author and body are content elements, they contain no information about how they will be presented. They are passed into Smarty by the application, then the template designer edits the templates and uses a combination of HTML tags and template tags to format the presentation of these elements (HTML tables, background colors, font sizes, style sheets, etc.) One day the programmer needs to change the way the article content is retrieved (a change in application logic.) This change does not affect the template designer, the content will still arrive in the template exactly the same. Likewise, if the template designer wants to completely redesign the templates, this requires no changes to the application logic. Therefore, the programmer can make changes to the application logic without the need to restructure templates, and the template designer can make changes to templates without breaking application logic.

Smartyphp,,,
,,,()0,,,,,

Now for a short word on what Smarty does NOT do. Smarty does not attempt to completely separate logic from the templates. There is no problem with logic in your templates under the condition that this logic is strictly for presentation. A word of advice: keep application logic out of the templates, and presentation logic out of the application. This will most definately keep things manageable and scalable for the foreseeable future.

smartysmarty,:,

One of the unique aspects about Smarty is the template compiling. This means Smarty reads the template files and creates PHP scripts from them. Once they are created, they are executed from then on. Therefore there is no costly template file parsing for each request, and each template can take full advantage of PHP compiler cache solutions such as Zend Accelerator (<http://www zend com>) or PHP Accelerator (<http://www php-accelerator co uk>).

Smarty'''Smartyphp,Zend(<http://www zend com>)
(<http://www php-accelerator co uk>)php

Some of Smarty's features:

Smaty:

- It is extremely fast.
!
- It is efficient since the PHP parser does the dirty work.
php
- No template parsing overhead, only compiles once.
,
- It is smart about recompiling only the template files that have changed.
- You can make [custom functions](#) and custom [variable modifiers](#), so the template language is extremely extensible.
''' ,
• Configurable template delimiter tag syntax, so you can use {}, {{}}, <!--{}-->, etc.
,{}, {{}}, <!--{}-->,
• The if/elseif/else/endif constructs are passed to the PHP parser, so the {if ...} expression syntax can be as simple or as complex as you like.
if/elseif/else/endif php, {if ...} ,
• Unlimited nesting of sections, ifs, etc. allowed.
,section

- It is possible to embed PHP code right in your template files, although this may not be needed (nor recommended) since the engine is so customizable.
.php,()
 - Built-in caching support
 - Arbitrary template sources
 - Custom cache handling functions
 - Plugin architecture
-

[Prev](#) [PHP](#)

Getting Started



[Home](#)

[Up](#)

[Next](#)

Installation



Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

Chapter 2. Installation

▪

Table of Contents

[Requirements](#)

[Basic Installation](#)

[Extended Setup](#)

Requirements

Smarty requires a web server running PHP 4.0.6 or later.
Smartywebphp4.0.6.

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

What is Smarty?
Smarty

[Up](#)

Basic Installation

Smarty - the compiling PHP template engine

[Prev](#) [PHP](#)

Chapter 2. Installation

[Next](#)

Basic Installation

Install the Smarty library files which are in the /libs/ directory of the distribution. These are the PHP files that you SHOULD NOT edit. They are shared among all applications and they only get updated when you upgrade to a new version of Smarty.

Smarty/libs/(). php.,smarty

Example 2-1. Smarty library files

2-1.Smarty

```
Smarty.class.php  
Smarty_Compiler.class.php  
Config_File.class.php  
debug.tpl  
/core/*.php (all of them)  
/plugins/*.php (all of them)
```

Smarty uses a PHP constant named [SMARTY_DIR](#) which is the system filepath Smarty library directory. Basically, if your application can find the *Smarty.class.php* file, you do not need to set SMARTY_DIR, Smarty will figure it out on its own. Therefore, if *Smarty.class.php* is not in your include_path, or you do not supply an absolute path to it in your application, then you must define SMARTY_DIR manually. SMARTY_DIR *must* include a trailing slash.

```
Smarty'SMARTY_DIR'php,  
SMARTY_DIR,Smarty,Smarty.class.phpinclude_path(  
,SMARTY_DIR ()SMARTY_DIR
```

Here is how you create an instance of Smarty in your PHP scripts:
phpsmarty:

Example 2-2. Create Smarty instance of Smarty

2-2.Smarty

```
require('Smarty.class.php');  
$smarty = new Smarty;
```

Try running the above script. If you get an error saying the *Smarty.class.php* file could not be found, you have to do one of the following:

,"
 Smarty.class.php " ,:;

Example 2-3. Supply absolute path to library file 2-3.

```
require('/usr/local/lib/php/Smarty/Smarty.class.php');  
$smarty = new Smarty;
```

Example 2-4. Add library directory to php_include_path 2-4.include_path

```
// Edit your php.ini file, add the Smarty library  
// directory to the include_path and restart web server:  
// Then the following should work:  
require('Smarty.class.php');  
$smarty = new Smarty;
```

Example 2-5. Set SMARTY_DIR constant manually 2-5.SMARTY_DIR

```
define('SMARTY_DIR', '/usr/local/lib/php/Smarty/');  
require(SMARTY_DIR.'Smarty.class.php');  
$smarty = new Smarty;
```

Now that the library files are in place, it's time to setup the Smarty directories for your application. Smarty requires four directories which are (by default) named *templates* , *templates_c* , *configs* and *cache* . Each of these are definable by the Smarty class properties *\$template_dir* , *\$compile_dir* , *\$config_dir* , and *\$cache_dir* respectively. It is highly recommended that you setup a separate set of these directories for each application that will use Smarty.

,SmartySmarty4,:;

Smarty: \$template_dir, \$compile_dir, \$config_dir, and \$cache_dir respectively!

Be sure you know the location of your web server document root. In our example, the document root is "/web/www.mydomain.com/docs/". The Smarty directories are only accessed by the Smarty library and never accessed directly by the web browser. Therefore to avoid any security concerns, it is recommended to place these directories *outside* of the document root.

web,:"/web/www.mydomain.com/docs/"Smarty4

For our installation example, we will be setting up the Smarty environment for a guest book application. We picked an application only for the purpose of a directory naming convention. You can use the same environment for any application, just replace "guestbook" with the name of your app. We'll place our Smarty directories under "/web/www.mydomain.com/smarty/guestbook/".

,smarty,"guestbook"Smarty
"/web/www.mydomain.com/smarty/guestbook/"

You will need at least one file under your document root, and that is the script accessed by the web browser. We will call our script "index.php", and place it in a subdirectory under the document root called "/guestbook/".

,, "index.php"."/"guestbook/"

Technical Note: It is convenient to setup the web server so that "index.php" can be identified as the default directory index, so if you access "http://www.mydomain.com/guestbook/", the index.php script will be executed without "index.php" in the URL. In Apache you can set this up by adding "index.php" onto the end of your DirectoryIndex setting (separate each entry with a space.)

webweb"http://www.mydomain.com/guestbook/"URL
"index.php"index.phpApacheDirectoryIndex"index.php"

Lets take a look at the file structure so far:

:

Example 2-6. Example file structure 2-6.

```
/usr/local/lib/php/Smarty/Smarty.class.php
/usr/local/lib/php/Smarty/Smarty_Compiler.class.php
/usr/local/lib/php/Smarty/Config_File.class.php
/usr/local/lib/php/Smarty/debug.tpl
/usr/local/lib/php/Smarty/core/*.php
/usr/local/lib/php/Smarty/plugins/*.php

/web/www.mydomain.com/smarty/guestbook/templates/
/web/www.mydomain.com/smarty/guestbook/templates_c/
/web/www.mydomain.com/smarty/guestbook/configs/
/web/www.mydomain.com/smarty/guestbook/cache/

/web/www.mydomain.com/docs/guestbook/index.php
```

Smarty will need write access to the `$compile_dir` and `$cache_dir`, so be sure the web server user can write to them. This is usually user "nobody" and group "nobody". For OS X users, the default is user "www" and group "www". If you are using Apache, you can look in your httpd.conf file (usually in "/usr/local/apache/conf/") to see what user and group are being used.

```
Smarty $compile_dir $cache_dir user "nobody" group "nob
OSX,user "web" group "web"Apache,httpd.conf
"/usr/local/apache/conf/")usergroup
```

Example 2-7. Setting file permissions 2-7

```
chown nobody:nobody /web/www.mydomain.com/smarty/guest
chmod 770 /web/www.mydomain.com/smarty/guestbook/templ

chown nobody:nobody /web/www.mydomain.com/smarty/guest
chmod 770 /web/www.mydomain.com/smarty/guestbook/cache
```

Technical Note: chmod 770 will be fairly tight security, it only allows user "nobody" and group "nobody" read/write access to the directories. If you would like to open up read access to anyone (mostly for your own convenience of viewing these files), you can use 775 instead.

:

```
chmod 770,user "nobody" group "nobody" / (),  
775
```

We need to create the index.tpl file that Smarty will load. This will be located in your \$template_dir.
index.tplsmarty. \$template_dir

Example 2-8. Editing

/web/www.mydomain.com/smarty/guestbook/templates/index.tpl

2-8 /web/www.mydomain.com/smarty/templates/index.tpl

```
{* Smarty *}
```

```
Hello, {$name} !
```

Technical Note: {* Smarty *} is a template comment. It is not required, but it is good practice to start all your template files with this comment. It makes the file easy to recognize regardless of the file extension. For example, text editors could recognize the file and turn on special syntax highlighting.

:

```
{* Smarty *} ,,
```

Now lets edit index.php. We'll create an instance of Smarty, assign a template variable and display the index.tpl file. In our example environment, "/usr/local/lib/php/Smarty" is in our include_path. Be sure you do the same, or use absolute paths.

```
index.phpSmarty,,  
include_path
```

```
index.tpl, "/usr
```

Example 2-9. Editing /web/www.mydomain.com/docs/guestbook/index.php

```
// load Smarty library
require('Smarty.class.php');

$smarty = new Smarty;

$smarty->template_dir = '/web/www.mydomain.com/smarty/';
$smarty->compile_dir = '/web/www.mydomain.com/smarty/g';
$smarty->config_dir = '/web/www.mydomain.com/smarty/g';
$smarty->cache_dir = '/web/www.mydomain.com/smarty/gue';

$smarty->assign('name', 'Ned');

$smarty->display('index.tpl');
```

Technical Note: In our example, we are setting absolute paths to all of the Smarty directories. If

'/web/www.mydomain.com/smarty/guestbook/' is within your PHP include_path, then these settings are not necessary. However, it is more efficient and (from experience) less error-prone to set them to absolute paths. This ensures that Smarty is getting files from the directories you intended.

```
:
,Smarty                                '/web/www.mydomain.
include_path,,,,
```

Now load the index.php file from your web browser. You should see
"Hello, Ned!"
index.php,"Hello, Porky!"

You have completed the basic setup for Smarty!
Smarty,!!

[Prev](#) [PHP](#)

[Home](#)

Installation

[Next](#)

Extended Setup

Up

Smarty - the compiling PHP template engine

[Prev](#) [PHP](#)

Chapter 2. Installation

[Next](#)

Extended Setup

This is a continuation of the [basic installation](#), please read that first!
!

A slightly more flexible way to setup Smarty is to extend the class and initialize your Smarty environment. So instead of repeatedly setting directory paths, assigning the same vars, etc., we can do that in one place. Lets create a new directory "/php/includes/guestbook/" and make a new file called "setup.php". In our example environment, "/php/includes" is in our include_path. Be sure you set this up too, or use absolute file paths.

```
Smarty,smarty
,
"/php/includes/guestbook/" "setup.php"
smarty
```

Example 2-10. Editing /php/includes/guestbook/setup.php

2-10. /php/includes/guestbook/setup.php

```
// load Smarty library
require('Smarty.class.php');

// The setup.php file is a good place to load
// required application library files, and you
// can do that right here. An example:
// require('guestbook/guestbook.lib.php');()
//index

class Smarty_GuestBook extends Smarty {

    function Smarty_GuestBook() {
        // Class Constructor. These automatica
    }
}
```

```

        $this->Smarty();

        $this->template_dir = '/web/www.mydomain.com/docs/guestbook';
        $this->compile_dir = '/web/www.mydomain.com/cache';
        $this->config_dir = '/web/www.mydomain.com/config';
        $this->cache_dir = '/web/www.mydomain.com/cache';

        $this->caching = true;
        $this->assign('app_name', 'Guest Book')
    }

}

```

Now lets alter the index.php file to use setup.php:
setupindex

Example 2-11. Editing

/web/www.mydomain.com/docs/guestbook/index.php
2-11./web/www.mydomain.com/docs/guestbook/index.php

```

require('guestbook/setup.php');

$smarty = new Smarty_GuestBook;

$smarty->assign('name', 'Ned');

$smarty->display('index.tpl');

```

Now you see it is quite simple to bring up an instance of Smarty, just use Smarty_GuestBook which automatically initializes everything for our application.

smarty.Smarty_GuestBook,^_^

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Basic Installation

[Up](#)

Smarty For Template
Designers

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

II. Smarty For Template Designers

Table of Contents[]

- 3. [Basic Syntax](#) []
 - 4. [Variables](#) []
 - 5. [Variable Modifiers](#) []
 - 6. [Combining Modifiers](#) []
 - 7. [Built-in Functions](#) []
 - 8. [Custom Functions](#) []
 - 9. [Config Files](#) []
 - 10. [Debugging Console](#) []
-

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Extended Setup []

[] Basic Syntax

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

Chapter 3. Basic Syntax

▪

Table of Contents[]

[Comments\[\]](#)

[Functions\[\]](#)

[Attributes\[\]](#)

[Embedding Vars in Double Quotes\[\]](#)

[Math\[\]](#)

All Smarty template tags are enclosed within delimiters. By default, these delimiters are { and }, but they can be changed.

smarty.

{ },.

For these examples, we will assume that you are using the default delimiters. In Smarty, all content outside of delimiters is displayed as static content, or unchanged. When Smarty encounters template tags, it attempts to interpret them, and displays the appropriate output in their place.

```
..  
smarty,,.  
smarty,, .
```

Comments[]

Template comments are surrounded by asterisks, and that is surrounded by the delimiter tags like so: {* this is a comment *} Smarty comments are not displayed in the final output of the template. They are used for making internal notes in the templates.

```
*, {* this is a comment *}  
smarty.  
. 
```

Example 3-1. Comments

3-1.

```
{* Smarty *}  
  
{* include the header file here *}  
{include file="header.tpl"}  
  
{include file=$includeFile}  
  
{include file=#includeFile#}  
  
{* display dropdown lists *}  
<SELECT name=company>  
{html_options values=$vals selected=$selected output=$}  
</SELECT>
```

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Smarty For Template
Designers

[Up](#)

Functions



Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 3. Basic Syntax[.]

[Next](#)

Functions

Each Smarty tag either prints a [variable](#) or invokes some sort of function. Functions are processed and displayed by enclosing the function and its attributes into delimiters like so: {funcname attr1="val" attr2="val"}.

smarty.

```
{'..:  
{funcname attr1="val" attr2="val"}.
```

Example 3-2. function syntax

3-2.

```
{config_load file="colors.conf"}  
  
{include file="header.tpl"}  
  
{if $highlight_name}  
    Welcome, <font color="#{$fontColor}">{$name} !<  
{else}  
    Welcome, {$name} !  
{/if}  
  
{include file="footer.tpl"}
```

Both built-in functions and custom functions have the same syntax in the templates. Built-in functions are the inner workings of Smarty, such as [{if}](#), [{section}](#) and [{strip}](#). They cannot be modified. Custom functions are additional functions implemented via plugins. They can be modified to your liking, or you can add new ones. [{html_options}](#) and [{html_select_date}](#) are examples of custom functions.

smarty,

[{if}](#), [{section}](#) and [{strip}](#)...

,..,..

{html_options} {html_select_date}

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Basic Syntax

[Up](#)

Attributes



Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 3. Basic Syntax[3.]

[Next](#)

Attributes



Most of the functions take attributes that specify or modify their behavior. Attributes to Smarty functions are much like HTML attributes. Static values don't have to be enclosed in quotes, but it is recommended for literal strings. Variables may also be used, and should not be in quotes.

smartyHTML.

..

..

Some attributes require boolean values (true or false). These can be specified as either unquoted true, on, and yes, or false, off, and no.

().
,true,on,yesfalse,off,no.

Example 3-3. function attribute syntax

3-3.

```
{include file="header.tpl"}  
  
{include file=$includeFile}  
  
{include file=#includeFile#}  
  
{html_select_date display_days=yes}  
  
<SELECT name=company>  
{html_options values=$vals selected=$selected output=$}  
</SELECT>
```

Functions



[Up](#)

Embedding Vars in
Double Quotes



Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 3. Basic Syntax[3.]

[Next](#)

Embedding Vars in Double Quotes



Smarty will recognize assigned variables embedded in double quotes so long as the variables contain only numbers, letters, underscores and brackets []. With any other characters (period, object reference, etc.) the variable must be surrounded by backticks.

Smarty[]: ``(``

Example 3-4. embedded quotes syntax

3-4.

SYNTAX EXAMPLES:

```
{func var="test $foo test"} <-- sees $foo
{func var="test $foo_bar test"} <-- sees $foo_bar
{func var="test $foo[0] test"} <-- sees $foo[0]
{func var="test $foo[bar] test"} <-- sees $foo[bar]
{func var="test $foo.bar test"} <-- sees $foo (not $fo
{func var="test `$foo.bar` test"} <-- sees $foo.bar
```

PRACTICAL EXAMPLES:

```
{include file="subdir/$tpl_name.tpl"} <-- will replace
{cycle values="one,two,`$smarty.config.myval`"} <-- mu
```

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Attributes

[Up](#)

Math



Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 3. Basic Syntax[3.]

[Next](#)

Math[]

Math can be applied directly to variable values.

Example 3-5. math examples

3-5.

```
{$foo+1}  
{$foo*$bar}  
{* some more complicated examples *}  
{$foo->bar-$bar[1]*$baz->foo->bar()-3*7}  
{if ($foo+$bar . test%$baz*134232+10+$b+10)}  
{$foo|truncate:"`$fooTruncCount/$barTruncFactor-1`"}  
{assign var="foo" value="`$foo+$bar`"}
```

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Embedding Vars in
Double Quotes

Variables

[Up](#)

[]



Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

Chapter 4. Variables[.]

Table of Contents[]

[Variables assigned from PHP\[PHP\]](#)

[Variables loaded from config files\[\]](#)

[{\\$smarty} reserved variable\[{\\$smarty} \]](#)

Smarty has several different types of variables.

The type of the variable depends on what symbol it is prefixed with (or enclosed within).

Smarty.

()

Variables in Smarty can be either displayed directly or used as arguments for function attributes and modifiers, inside conditional expressions, etc.

To print a variable, simply enclose it in the delimiters so that it is the only thing contained between them. Examples:

Smarty modifiers.,

,:

```
{$Name}
```

```
{$Contacts[row].Phone}
```

```
<body bgcolor="#bgcolor">
```

Variables assigned from PHP

PHP

Table of Contents

[Associative arrays](#)

[Array indexes](#)

[Objects](#)

Variables that are assigned from PHP are referenced by preceding them with a dollar sign \$.

Variables assigned from within the template with the [assign](#) function are also displayed this way.

PHP".\$php
assign.\$

Example 4-1. assigned variables

4-1.

index.php:

```
$smarty = new Smarty;  
$smarty->assign('firstname', 'Doug');  
$smarty->assign('lastLoginDate', 'January 11th, 2001')  
$smarty->display('index.tpl');
```

index.tpl:

```
Hello {$firstname}, glad to see you could make it.  
<p>  
Your last login was on {$lastLoginDate}.
```

OUTPUT:

```
Hello Doug, glad to see you could make it.  
<p>
```

Your last login was on January 11th, 2001.

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Attributes

[Up](#)

Associative arrays



Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 4. Variables[.]

[Next](#)

Variables loaded from config files

Variables that are loaded from the config files are referenced by enclosing them within hash marks (#), or with the smarty variable [\\$smarty.config](#).

The second syntax is useful for embedding into quoted attribute values.

```
"#"smarty      \$smarty.config.  
.  
          {include file="#includefile#"} #includefile#  
{include file="$smarty.config.includefile`"}``
```

Example 4-5. config variables

4-5.

```
foo.conf:  
  
pageTitle = "This is mine"  
bodyBgColor = "#eeeeee"  
tableBorderSize = "3"  
tableBgColor = "#bbbbbb"  
rowBgColor = "#cccccc"  
  
index.tpl:  
  
{config_load file="foo.conf"}  
<html>  
<title>{#pageTitle#}</title>  
<body bgcolor="{#bodyBgColor#}">  
<table border="{#tableBorderSize#}" bgcolor="{#tableBgColor#}">  
<tr bgcolor="{#rowBgColor#}">  
    <td>First</td>  
    <td>Last</td>  
    <td>Address</td>  
</tr>  
</table>  
</body>  
</html>
```

```
index.tpl: (alternate syntax)

{config_load file="foo.conf"}
<html>
<title>{$smarty.config.pageTitle}</title>
<body bgcolor="{$smarty.config.bodyBgColor}">
<table border="{$smarty.config.tableBorderSize}" bgcolor="{$smarty.config.rowBgColor}">
    <tr>
        <td>First</td>
        <td>Last</td>
        <td>Address</td>
    </tr>
</table>
</body>
</html>
```

OUTPUT: (same for both examples)

```
<html>
<title>This is mine</title>
<body bgcolor="#eeeeee">
<table border="3" bgcolor="#bbbbbb">
    <tr bgcolor="#cccccc">
        <td>First</td>
        <td>Last</td>
        <td>Address</td>
    </tr>
</table>
</body>
</html>
```

Config file variables cannot be used until after they are loaded in from a config file. This procedure is explained later in this document under [{config load}](#).

{config_load} . .

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Variables



[Up](#)

{\$smarty} reserved
variable
[\$smarty]

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 4. Variables[.]

[Next](#)

{\$smarty} reserved variable {\$smarty}

Table of Contents

[Request variables](#) [](get,post,server,session)
[{\\$smarty.now}](#)
[{\\$smarty.const}](#)
[{\\$smarty.capture}](#)
[{\\$smarty.config}](#)
[{\\$smarty.section}, {\\$smarty.foreach}](#)
[{\\$smarty.template}](#)

The reserved {\$smarty} variable can be used to access several special template variables. The full list of them follows.

{\$smarty}.

:

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Variables loaded from
config files

[Up](#)

Request variables
[]



Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

Chapter 5. Variable Modifiers []

Table of Contents []

[capitalize](#)
[count_characters](#)
[cat](#)
[count_paragraphs](#)
[count_sentences](#)
[count_words](#)
[date_format](#)
[default](#)
[escape](#)
[indent](#)
[lower](#)
[nl2br](#)
[regex_replace](#)
[replace](#)
[spacify](#)
[string_format](#)
[strip](#)
[strip_tags](#)
[truncate](#)
[upper](#)
[wordwrap](#)

Variable modifiers can be applied to variables, custom functions or strings. To apply a modifier, specify the value followed by the | (pipe) and the modifier name. A modifier may accept additional parameters that affect its behavior. These parameters follow the modifier name and are separated by : (colon).

'|' "

Example 5-1. modifier example

5-1.

```
{* Uppercase the title *}
```

```
<h2>{$title|upper}</h2>

{* Truncate the topic to 40 characters use ... at the
Topic: {$topic|truncate:40:"...."}}

{* format a literal string *}
{"now"|date_format:@"%Y/%m/%d"}

{* apply modifier to a custom function *}
{mailto|upper address="me@domain.dom"}
```

If you apply a modifier to an array variable instead of a single value variable, the modifier will be applied to every value in that array. If you really want the modifier to work on an entire array as a value, you must prepend the modifier name with an @ symbol like so:

`{$articleTitle|@count}` (this will print out the number of elements in the \$articleTitle array.)

`@ {$articleTitle|@count} $articleTitle`

Modifiers can be autoloaded from your [\\$plugins_dir](#) (also see: [Naming Conventions](#)) or can be registered explicitly (see: [register_modifier](#)). Additionally all php-functions can be used as modifiers implicitly. (The @count-example above actually uses php's count-function and not a smarty-modifier). Using php-functions as modifiers has two little pitfalls: First: Sometimes the order of the function-parameters is not the desirable one (`{"%2.f" | sprintf:$float}` actually works, but asks for the more intuitive {For example: `$float|string_format:"%2.f"`} that is provided by the Smarty distribution). Second: with [\\$security](#) turned on all php-functions that are to be used as modifiers have to be declared trusted in the [\\$security_settings\['MODIFIER_FUNCS'\]](#)-array.

capitalize

This is used to capitalize the first letter of all words in a variable.

Example 5-2. capitalize 5-2.

index.php:

```
$smarty = new Smarty;  
$smarty->assign('articleTitle', 'Police begin campaign');  
$smarty->display('index.tpl');
```

index.tpl:

```
 {$articleTitle}  
 {$articleTitle|capitalize}
```

OUTPUT:

```
Police begin campaign to rundown jaywalkers.  
Police Begin Campaign To Rundown Jaywalkers.
```

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

[{\\$smarty.template}](#)

[Up](#)

[\[\] count_characters](#)

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 5. Variable Modifiers

[Next](#)

count_characters()

Parameter Position	Type	Required	Default	Description
1	boolean	No	false	This determines whether or not to include whitespace characters in the count.

This is used to count the number of characters in a variable.

Example 5-3. count_characters

index.php:

```
$smarty = new Smarty;  
$smarty->assign('articleTitle', 'Cold Wave Linked to Temperatures');  
$smarty->display('index.tpl');
```

index.tpl:

```
{$articleTitle}  
{$articleTitle|count_characters}  
{$articleTitle|count_characters:true}
```

OUTPUT:

Cold Wave Linked to Temperatures.

29

33

[Prev](#) [PHP](#)

Variable Modifiers

[Home](#)

[Up](#)

[Next](#)

cat

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 5. Variable Modifiers

[Next](#)

cat[]

Parameter Position	Type	Required	cat	Description
1	string	No	<i>empty</i>	This value to catenate to the given variable. cat.

This value is concatenated to the given variable.
cat.

Example 5-4. cat

index.php:

```
$smarty = new Smarty;  
$smarty->assign('articleTitle', "Psychics predict world  
$smarty->display('index.tpl');
```

index.tpl:

```
{$articleTitle|cat:" yesterday."}
```

OUTPUT:

Psychics predict world didn't end yesterday.

[Prev](#) [PHP](#)

count_characters

[Home](#)

[Up](#)

[Next](#)

count_paragraphs

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 5. Variable Modifiers

[Next](#)

count_paragraphs[]

This is used to count the number of paragraphs in a variable.

Example 5-5. count_paragraphs

index.php:

```
$smarty = new Smarty;  
$smarty->assign('articleTitle', "War Dims Hope for Peace  
Couple's Holiday.\n\nMan is Fatally Slain. Death Causes Loneliness,  
2  
$smarty->display('index.tpl');
```

index.tpl:

```
{$articleTitle}  
{$articleTitle|count_paragraphs}
```

OUTPUT:

```
War Dims Hope for Peace. Child's Death Ruins Couple's  
Man is Fatally Slain. Death Causes Loneliness, Feeling  
2
```

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 5. Variable Modifiers

[Next](#)

count_sentences[]

This is used to count the number of sentences in a variable.

Example 5-6. count_sentences

index.php:

```
$smarty = new Smarty;  
$smarty->assign('articleTitle', 'Two Soviet Ships Collide - One Dies. Enraged Cow Injures 2')  
$smarty->display('index.tpl');
```

index.tpl:

```
{$articleTitle}  
{$articleTitle|count_sentences}
```

OUTPUT:

```
Two Soviet Ships Collide - One Dies. Enraged Cow Injures 2  
2
```

[Prev](#) [PHP](#)

count_paragraphs

[Home](#)

[Up](#)

[Next](#)

count_words

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 5. Variable Modifiers

[Next](#)

count_words[]

This is used to count the number of words in a variable.

Example 5-7. count_words

index.php:

```
$smarty = new Smarty;  
$smarty->assign('articleTitle', 'Dealers Will Hear Car  
$smarty->display('index.tpl');
```

index.tpl:

```
{$articleTitle}  
{$articleTitle|count_words}
```

OUTPUT:

```
Dealers Will Hear Car Talk at Noon.  
7
```

[Prev](#) [PHP](#)

count_sentences

[Home](#)

[Up](#)

[Next](#)

date_format

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 5. Variable Modifiers

[Next](#)

date_format[]

Parameter Position	Type	Required	Default	Description
1	string	No	%b %e, %Y	This is the format for the outputted date.
2	string	No	n/a	This is the default date if the input is empty.

This formats a date and time into the given strftime() format.

Dates can be passed to Smarty as unix timestamps, mysql timestamps or any string made up of month day year (parsable by strtotime).

Designers can then use date_format to have complete control of the formatting of the date.

If the date passed to date_format is empty and a second parameter is passed, that will be used as the date to format.

strftime()
Unixmysql(parsable by strtotime)smarty
date_format
date_format,

Example 5-8. date_format[]

index.php:

```
$smarty = new Smarty;  
$smarty->assign('yesterday', strtotime('-1 day'));  
$smarty->display('index.tpl');
```

index.tpl:

```
{${smarty.now|date_format}  
{$smarty.now|date_format:"%A, %B %e, %Y"}  
{$smarty.now|date_format:"%H:%M:%S"}  
{$yesterday|date_format}  
{$yesterday|date_format:"%A, %B %e, %Y"}  
{$yesterday|date_format:"%H:%M:%S"}
```

OUTPUT:

```
Feb 6, 2001  
Tuesday, February 6, 2001  
14:33:00  
Feb 5, 2001  
Monday, February 5, 2001  
14:33:00
```

Example 5-9. date_format conversion specifiers[]

%a - abbreviated weekday name according to the current locale
""

%A - full weekday name according to the current locale
""

%b - abbreviated month name according to the current locale
""

%B - full month name according to the current locale
""

%c - preferred date and time representation for the current locale
""

%C - century number (the year divided by 100 and truncated to 0 or 1)
0

%d - day of the month as a decimal number (range 00 to 31)
0

%D - same as %m/%d/%y

%e - day of the month as a decimal number, a single digit space (range 1 to 31)

%g - Week-based year within century [00, 99]

%G - Week-based year, including the century [0000, 9999]

%h - same as %b

%H - hour as a decimal number using a 24-hour clock (range 00 to 23)

%I - hour as a decimal number using a 12-hour clock (range 01 to 12)

%j - day of the year as a decimal number (range 001 to 365)

%k - Hour (24-hour clock) single digits are preceded by a space

%l - hour as a decimal number using a 12-hour clock, single digit followed by a space (range 1 to 12)

%m - month as a decimal number (range 01 to 12)

%M - minute as a decimal number

%n - newline character

%p - either `am' or `pm' according to the given time value

%r - time in a.m. and p.m. notation

%R - time in 24 hour notation

%S - second as a decimal number

%t - tab character

%T - current time, equal to %H:%M:%S

```
%u - weekday as a decimal number [1,7], with 1 represent Sunday  
%U - week number of the current year as a decimal number  
%V - The ISO 8601:1988 week number of the current year is the first week that has at least 4 days in the current year  
%w - day of the week as a decimal, Sunday being 0  
%W - week number of the current year as a decimal number  
%x - preferred date representation for the current locale  
%X - preferred time representation for the current locale  
%y - year as a decimal number without a century (range 00-99)  
%Y - year as a decimal number including the century  
%Z - time zone or name or abbreviation  
%% - a literal '%' character
```

PROGRAMMERS NOTE: date_format is essentially a wrapper function. You may have more or less conversion specification than your system's strftime() function where PHP was compiled. See your system's manpage for a full list of valid specifiers.

:date_format
phpstrftime()
phpstrptime()

.

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 5. Variable Modifiers

[Next](#)

default[]

Parameter Position	Type	Required	Default	Description
1	string	No	<i>empty</i>	This is the default value to output if the variable is empty.

This is used to set a default value for a variable. If the variable is empty or unset, the given default value is printed instead. Default takes one argument.

,

Example 5-10. default

index.php:

```
$smarty = new Smarty;  
$smarty->assign('articleTitle', 'Dealers Will Hear Car  
$smarty->display('index.tpl');
```

index.tpl:

```
{$articleTitle|default:"no title"}  
{$myTitle|default:"no title"}
```

OUTPUT:

```
Dealers Will Hear Car Talk at Noon.  
no title
```

date_format

[Up](#)

escape

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 5. Variable Modifiers

[Next](#)

escape[]

Parameter Position	Type	Required	Possible Values
1	string	No	html,htmlall,url,quotes,hex,hexentity,javascript,html,htmlall,url,quotes,hex,hexentity,javascript

This is used to html escape, url escape, escape single quotes on a variable not already escaped, hex escape, hexentity or javascript escape. By default, the variable is html escaped.
html,url,,,javascripthtml

Example 5-11. escape

index.php:

```
$smarty = new Smarty;  
$smarty->assign('articleTitle', "'Stiff Opposition Expected'");  
$smarty->display('index.tpl');
```

index.tpl:

```
 {$articleTitle}  
 {$articleTitle|escape}  
 {$articleTitle|escape:"html"} {* escapes & " ' < > *}  
 {$articleTitle|escape:"htmlall"} {* escapes ALL html entities}  
 {$articleTitle|escape:"url"}  
 {$articleTitle|escape:"quotes"}  
<a href="mailto:{$EmailAddress|escape:"hex"}">{$EmailA
```

OUTPUT:

```
'Stiff Opposition Expected to Casketless Funeral Plan'  
&#039;Stiff Opposition Expected to Casketless Funeral
```

'Stiff Opposition Expected to Casketless Funeral
'Stiff Opposition Expected to Casketless Funeral
%27Stiff+Opposition+Expected+to+Casketless+Funeral+Pla
\'Stiff Opposition Expected to Casketless Funeral Plan
b

[Prev](#) [PHP](#)
default

[Home](#)
[Up](#)

[Next](#)
indent

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 5. Variable Modifiers

[Next](#)

indent()

Parameter Position	Type	Required	Default	Description
1	integer	No	4	This determines how many characters to indent to.
2	string	No	(one space)	This is the character used to indent with.

This indents a string at each line, default is 4. As an optional parameter, you can specify the number of characters to indent. As an optional second parameter, you can specify the character to use to indent with. (Use "\t" for tabs.)

4

HTML s p();

Example 5-12. indent

```
index.php:  
  
$smarty = new Smarty;  
$smarty->assign('articleTitle', 'NJ judge to rule on r  
$smarty->display('index.tpl');  
  
index.tpl:  
  
{$articleTitle}
```

```
{$articleTitle|indent}  
{$articleTitle|indent:10}  
{$articleTitle|indent:1:"\t"}
```

OUTPUT:

NJ judge to rule on nude beach.
Sun or rain expected today, dark tonight.
Statistics show that teen pregnancy drops off significantly.

NJ judge to rule on nude beach.
Sun or rain expected today, dark tonight.
Statistics show that teen pregnancy drops off significantly.

NJ judge to rule on nude beach.
Sun or rain expected today, dark tonight.
Statistics show that teen pregnancy drops off significantly.

NJ judge to rule on nude beach.
Sun or rain expected today, dark tonight.
Statistics show that teen pregnancy drops off significantly.

[Prev](#) [PHP](#)
escape

[Home](#)
[Up](#)

[Next](#)
lower

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 5. Variable Modifiers[]

[Next](#)

lower

This is used to lowercase a variable.

Example 5-13. lower 5-13.

index.php:

```
$smarty = new Smarty;  
$smarty->assign('articleTitle', 'Two Convicts Evade Noose');  
$smarty->display('index.tpl');
```

index.tpl:

```
 {$articleTitle}  
 {$articleTitle|lower}
```

OUTPUT:

```
Two Convicts Evade Noose, Jury Hung.  
two convicts evade noose, jury hung.
```

[Prev](#) [PHP](#)

indent[]

[Home](#)

[Up](#)

[Next](#)

nl2br

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 5. Variable Modifiers[]

[Next](#)

nl2br

All linebreaks will be converted to
 tags in the given variable. This is equivalent to the PHP nl2br() function.

.PHPnl2br().

Example 5-14. nl2br

5-14.

index.php:

```
$smarty = new Smarty;
$smarty->assign('articleTitle', "Sun or rain expected\n");
$smarty->display('index.tpl');
```

index.tpl:

```
{$articleTitle|nl2br}
```

OUTPUT:

```
Sun or rain expected<br />today, dark tonight
```

[Prev](#) [PHP](#)

lower[]

[Home](#)

[Up](#)

[Next](#)

[regex_replace\[\]](#)

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 5. Variable Modifiers[]

[Next](#)

regex_replace

Parameter Position	Type	Required	Default	Description
1	string	Yes	n/a	This is the regular expression to be replaced. .
2	string	Yes	n/a	This is the string of text to replace with.

A regular expression search and replace on a variable. Use the syntax for preg_replace() from the PHP manual.

.
Phppreg_replace().

Example 5-15. regex_replace 5-15.

index.php:

```
$smarty = new Smarty;  
$smarty->assign('articleTitle', "Infertility unlikely  
$smarty->display('index.tpl');
```

index.tpl:

```
{* replace each carriage return, tab & new line with a  
{* ,tab, *}  
{$articleTitle}  
{$articleTitle|regex_replace:"/[\r\t\n]/":" "}
```

OUTPUT:

Infertility unlikely to
be passed on, experts say.
Infertility unlikely to be passed on, experts say.

[Prev](#) [PHP](#)

nl2br[\n</br>

[Home](#)

[Up](#)

[Next](#)

[replace\[\]](#)

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 5. Variable Modifiers[]

[Next](#)

replace

Parameter Position	Type	Required	Default	Description
1	string	Yes	n/a	This is the string of text to be replaced.
2	string	Yes	n/a	This is the string of text to replace with.

A simple search and replace on a variable.

Example 5-16. replace 5-16.

index.php:

```
$smarty = new Smarty;  
$smarty->assign('articleTitle', "Child's Stool Great for Use in Garden.");  
$smarty->display('index.tpl');
```

index.tpl:

```
{$articleTitle}  
{$articleTitle|replace:"Garden":"Vineyard"}  
{$articleTitle|replace:" ":" "}
```

OUTPUT:

```
Child's Stool Great for Use in Garden.  
Child's Stool Great for Use in Vineyard.
```

Child's Stool Great for Use in Garden.

[Prev](#) [PHP](#)

regex_replace()

[Home](#)

[Up](#)

[Next](#)

spacify()

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 5. Variable Modifiers[]

[Next](#)

spacify

Parameter Position	Type	Required	Default	Description
1	string	No	one space	This what gets inserted between each character of the variable.

spacify is a way to insert a space between every character of a variable. You can optionally pass a different character (or string) to insert.

(,^^)().

Example 5-17. spacify

5-17.

index.php:

```
$smarty = new Smarty;  
$smarty->assign('articleTitle', 'Something Went Wrong');  
$smarty->display('index.tpl');
```

index.tpl:

```
{$articleTitle}  
{$articleTitle|spacify}  
{$articleTitle|spacify:"^^"}
```

OUTPUT:

```
Something Went Wrong in Jet Crash, Experts Say.  
S o m e t h i n g W e n t W r o n g i n J e t C r a s  
S^^o^^m^^e^^t^^h^^i^^n^^g^^ ^^^W^^e^^n^^t^^ ^^^W^^r^^o^^
```

[Prev](#) [PHP](#)

[replace\(\)](#)

[Home](#)

[Up](#)

[Next](#)

[string_format\(\)](#)

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 5. Variable Modifiers[]

[Next](#)

string_format

Parameter Position	Type	Required	Default	Description
1	string	Yes	n/a	This is what format to use. (sprintf)

This is a way to format strings, such as decimal numbers and such. Use the syntax for sprintf for the formatting.

..sprintf

Example 5-18. string_format

index.php:

```
$smarty = new Smarty;  
$smarty->assign('number', 23.5787446);  
$smarty->display('index.tpl');
```

index.tpl:

```
{$number}  
{$number|string_format:"%.2f"}  
{$number|string_format:"%d"}
```

OUTPUT:

```
23.5787446  
23.58  
24
```

spacify[]

[Up](#)

strip[()]

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 5. Variable Modifiers[]

[Next](#)

strip ()

This replaces all repeated spaces, newlines and tabs with a single space, or with a supplied string.

..

Note: If you want to strip blocks of template text, use the [strip function](#).

:, strip

Example 5-19. strip

5-19.()

```
index.php:
```

```
$smarty = new Smarty;  
$smarty->assign('articleTitle', "Grandmother of\nneighbor  
$smarty->display('index.tpl');
```

```
index.tpl:
```

```
{$articleTitle}  
{$articleTitle|strip}  
{$articleTitle|strip:"&nbsp;"}
```

OUTPUT:

```
Grandmother of  
eight makes hole in one.  
Grandmother of eight makes hole in one.  
Grandmother&nbsp;of&nbsp;eight&nbsp;makes&nbsp;hole&nb
```

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 5. Variable Modifiers[]

[Next](#)

strip_tags html

This strips out markup tags, basically anything between < and >.

<>, <>.

Example 5-20. strip_tags

5-20.Html

index.php:

```
$smarty = new Smarty;
$smarty->assign('articleTitle', "Blind Woman Gets <font face='helvetica'>New Kidney</font> from Dad she Hasn't Seen in Years");
$smarty->display('index.tpl');
```

index.tpl:

```
{$articleTitle}
{$articleTitle|strip_tags}
```

OUTPUT:

```
Blind Woman Gets New Kidney</font> from Dad she Hasn't Seen in Years
Blind Woman Gets New Kidney from Dad she Hasn't Seen in Years
```

[Prev](#) [PHP](#)

strip[()]

[Home](#)

[Up](#)

[Next](#)

[truncate\[\]](#)

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 5. Variable Modifiers[]

[Next](#)

truncate

Parameter Position	Type	Required	Default	Description
1	integer	No	80	This determines how many characters to truncate to.
2	string	No	...	This is the text to append if truncation occurs.
3	boolean	No	false	This determines whether or not to truncate at a word boundary (false), or at the exact character (true).

This truncates a variable to a character length, default is 80.

As an optional second parameter, you can specify a string of text to display at the end if the variable was truncated. The characters in the string are included with the original truncation length.

By default, truncate will attempt to cut off at a word boundary.

If you want to cut off at the exact character length, pass the optional third parameter of true.

.80.

,smarty
,"true"

Example 5-21. truncate 5-21.

index.php:

```
$smarty = new Smarty;  
$smarty->assign('articleTitle', 'Two Sisters Reunite after Eighteen Years at Checkout Counter');  
$smarty->display('index.tpl');
```

index.tpl:

```
{$articleTitle}  
{$articleTitle|truncate}  
{$articleTitle|truncate:30}  
{$articleTitle|truncate:30:""}  
{$articleTitle|truncate:30:"---"}  
{$articleTitle|truncate:30:"":true}  
{$articleTitle|truncate:30:"...":true}
```

OUTPUT:

```
Two Sisters Reunite after Eighteen Years at Checkout Counter  
Two Sisters Reunite after Eighteen Years at Checkout Counter  
Two Sisters Reunite after...  
Two Sisters Reunite after  
Two Sisters Reunite after---  
Two Sisters Reunite after Eigh  
Two Sisters Reunite after E...
```

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 5. Variable Modifiers[]

[Next](#)

upper

This is used to uppercase a variable.

Example 5-22. upper 5-22.

index.php:

```
$smarty = new Smarty;  
$smarty->assign('articleTitle', "If Strike isn't Settled");  
$smarty->display('index.tpl');
```

index.tpl:

```
{$articleTitle}  
{$articleTitle|upper}
```

OUTPUT:

```
If Strike isn't Settled Quickly it may Last a While.  
IF STRIKE ISN'T SETTLED QUICKLY IT MAY LAST A WHILE.
```

[Prev](#) [PHP](#)

truncate[]

[Home](#)

[Up](#)

[Next](#)

wordwrap[]

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 5. Variable Modifiers[]

[Next](#)

wordwrap

Parameter Position	Type	Required	Default	Description
1	integer	No	80	This determines how many columns to wrap to. ()
2	string	No	\n	This is the string used to wrap words with.
3	boolean	No	false	This determines whether or not to wrap at a word boundary (false), or at the exact character (true).

This wraps a string to a column width, default is 80.

As an optional second parameter, you can specify a string of text to wrap the text to the next line (default is carriage return \n).

By default, wordwrap will attempt to wrap at a word boundary. If you want to cut off at the exact character length, pass the optional third parameter of true.

(,).80.

,(\n).

smarty,ture

Example 5-23. wordwrap 5-23.

index.php:

```
$smarty = new Smarty;  
$smarty->assign('articleTitle', "Blind woman gets new  
$smarty->display('index.tpl');
```

index.tpl:

```
{$articleTitle}  
  
{$articleTitle|wordwrap:30}  
  
{$articleTitle|wordwrap:20}  
  
{$articleTitle|wordwrap:30:"<br>\n"}  
  
{$articleTitle|wordwrap:30:"\n":true}
```

OUTPUT:

Blind woman gets new kidney from dad she hasn't seen i

Blind woman gets new kidney
from dad she hasn't seen in
years.

Blind woman gets new
kidney from dad she
hasn't seen in
years.

Blind woman gets new kidney

from dad she hasn't seen in years.

Blind woman gets new kidney fr
om dad she hasn't seen in year

s .

[Prev](#) [PHP](#)

upper()

[Home](#)

[Up](#)

[Next](#)

Combining Modifiers[
]

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

Chapter 6. Combining Modifiers

6

You can apply any number of modifiers to a variable. They will be applied in the order they are combined, from left to right. They must be separated with a | (pipe) character.

"T"

Example 6-1. combining modifiers 6-1.

index.php:

```
$smarty = new Smarty;  
$smarty->assign('articleTitle', 'Smokers are Productive');  
$smarty->display('index.tpl');
```

index.tpl:

```
 {$articleTitle}  
 {$articleTitle|upper|spacify}  
 {$articleTitle|lower|spacify|truncate}  
 {$articleTitle|lower|truncate:30|spacify}  
 {$articleTitle|lower|spacify|truncate:30:". . ."}
```

OUTPUT:

```
Smokers are Productive, but Death Cuts Efficiency.  
S M O K E R S A R E P R O D U C T I V E , B U T D E A  
s m o k e r s a r e p r o d u c t i v e , b u t d e a  
s m o k e r s a r e p r o d u c t i v e , b u t . . .  
s m o k e r s a r e p. . .
```

[wordwrap\(\)](#)

[Up](#)

[Built-in Functions](#)

Smarty - the compiling PHP template engine

Smarty - php

[Prev](#)

[PHP](#)

[Next](#)

Chapter 7. Built-in Functions[.]

Table of Contents

[capture](#)
[config_load](#)
[foreach,foreachelse](#)
[include](#)
[include_php](#)
[insert](#)
[if,elseif,else](#)
[ldelim,rdelim](#)
[literal](#)
[php](#)
[section,sectionelse](#)
[strip](#)

Smarty comes with several built-in functions. Built-in functions are integral to the template language. You cannot create custom functions with the same names, nor can you modify built-in functions.

Smarty.

.

..

capture

Attribute Name	Type	Required	Default	Description
name	string	no	<i>default</i>	The name of the captured block
assign	string	No	<i>n/a</i>	The variable name where to assign the captured output to

name	string	no	<i>default</i>	
assign	string	No	<i>n/a</i>	name[]

capture is used to collect the output of the template into a variable instead of displaying it. Any content between {capture name="foo"} and{/capture} is collected into the variable specified in the name attribute. The captured content can be used in the template from the special variable \$smarty.capture.foo where foo is the value passed in the name attribute. If you do not supply a name attribute, then "default" will be used. All {capture} commands must be paired with{/capture}. You can nest capture commands.

```
capture,.  
{capture name="foo"}{/capture}$fooname.  
$smarty.capture.foo .  
name , "default" .  
{capture}{/capture},.
```

Technical Note: Smarty 1.4.0 - 1.4.4 placed the captured content into the variable named \$return. As of 1.4.5, this behavior was changed to use the name attribute, so update your templates accordingly.

:
Smarty 1.4.0 - 1.4.4 \$return .
1.4.5 name ..

Caution

Be careful when capturing [{\\$insert}](#) output. If you have caching turned on and you have [{\\$insert}](#) commands that you expect to run within cached content, do not capture this content.

[{\\$insert}](#).

[{\\$ins}](#)

Example 7-1. capturing template content
7-1.

```
{* we don't want to print a table row unless content is
{* *}
{capture name=banner}
{include file="get_banner.tpl"}
{/capture}
{if $smarty.capture.banner ne ""}
    <tr>
        <td>
            {$smarty.capture.banner}
        </td>
    </tr>
{/if}
```

[Prev](#) [PHP](#)

Combining Modifiers

[Home](#)

[Up](#)

[Next](#)

config_load

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 7. Built-in Functions[.]

[Next](#)

config_load

Attribute Name	Type	Required	Default	Description
file	string	Yes	<i>n/a</i>	The name of the config file to include
section	string	No	<i>n/a</i>	The name of the section to load
scope	string	no	<i>local</i>	How the scope of the loaded variables are treated, which must be one of local, parent or global. local means variables are loaded into the local template context. parent means variables are loaded into both the local context and the parent template that called it. global means variables are available to all

				templates.
global	boolean	No	No	Whether or not variables are visible to the parent template, same as scope=parent. NOTE: This attribute is deprecated by the scope attribute, but still supported. If scope is supplied, this value is ignored.
file	string	Yes	n/a	
section	string	No	n/a	
scope	string	no	local/	local, parent global. local . parent (). global .
global	boolean	No	No	scope=parent. : scope scope

This function is used for loading in variables from a configuration file into the template. See [Config Files](#) for more info.

Example 7-2. function config_load

7-2. config_load

```
{config_load file="colors.conf"}  
  
<html>  
<title>{#pageTitle#}</title>  
<body bgcolor="{#bodyBgColor#}">  
<table border="{#tableBorderSize#}" bgcolor="{#tableBgColor#}">  
    <tr bgcolor="{#rowBgColor#}">  
        <td>First</td>  
        <td>Last</td>  
        <td>Address</td>  
    </tr>  
</table>  
</body>  
</html>
```

Config files may also contain sections. You can load variables from within a section with the added attribute *section* .

section .

NOTE: *Config file sections* and the built-in template function called *section* have nothing to do with each other, they just happen to share a common naming convention.

section section

Example 7-3. function config_load with section

7-3. section config_load

```
{config_load file="colors.conf" section="Customer"}  
  
<html>  
<title>{#pageTitle#}</title>  
<body bgcolor="{#bodyBgColor#}">  
<table border="{#tableBorderSize#}" bgcolor="{#tableBgColor#}">
```

```
<tr bgcolor="#{rowBgColor}">
    <td>First</td>
    <td>Last</td>
    <td>Address</td>
</tr>
</table>
</body>
</html>
```

[Prev](#) [PHP](#)

Built-in Functions

[Home](#)

[Up](#)

[Next](#)

foreach,foreachelse

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 7. Built-in Functions[.]

[Next](#)

foreach,foreachelse

Table of Contents

[iteration](#)

[first](#)

[last](#)

[show](#)

[total](#)

Attribute Name	Type	Required	Default	Description
from	string	Yes	n/a	The name of the array you are looping through
item	string	Yes	n/a	The name of the variable that is the current element
key	string	No	n/a	The name of the variable that is the current key
name	string	No	n/a	The name of the foreach loop for accessing foreach properties

from	string	Yes	n/a	
item	string	Yes	n/a	
key	string	No	n/a	

<code>name</code>	<code>string</code>	<code>No</code>	<code>n/a</code>		
-------------------	---------------------	-----------------	------------------	--	--

foreach loops are an alternative to *section* loops. *foreach* is used to loop over a single associative array. The syntax for *foreach* is much easier than *section*, but as a tradeoff it can only be used for a single array. *foreach* tags must be paired with */foreach* tags. Required parameters are *from* and *item*. The name of the *foreach* loop can be anything you like, made up of letters, numbers and underscores. *foreach* loops can be nested, and the nested *foreach* names must be unique from each other. The *from* variable (usually an array of values) determines the number of times *foreach* will loop. *foreachelse* is executed when there are no values in the *from* variable.

```
foreach section () .
foreach ()      section .
foreach /foreach from item .
name () .
foreach      foreach .
from () .
foreachelse from .
```

Example 7-4. foreach

7-4. foreach

```
{* this example will print out all the values of the $*
{* $custid *}
{foreach from=$custid item=curr_id}
    id: {$curr_id}<br>
{/foreach}
```

OUTPUT:

```
id: 1000<br>
id: 1001<br>
id: 1002<br>
```

Example 7-5. foreach key

7-5. foreach

{* The key contains the key for each looped value
assignment looks like this:

```
$smarty->assign("contacts", array(array("phone" => "1"  
    array("phone" => "555-4444", "fax" => "555-3333", "ce  
*)  
{*  *}  
  
{foreach name=outer item=contact from=$contacts}  
  {foreach key=key item=item from=$contact}  
    {$key}: {$item}<br>  
 {/foreach}  
{/foreach}
```

OUTPUT:

```
phone: 1<br>  
fax: 2<br>  
cell: 3<br>  
phone: 555-4444<br>  
fax: 555-3333<br>  
cell: 760-1234<br>
```

Foreach-loops also have their own variables that handle foreach properties. These are indicated like so:

`{$smarty.foreach.foreachname.varname}` with foreachname being the name specified as the *name* attribute of foreach

```
foreach . {$smarty.foreach.foreachname.varname}  
foreachname foreach name .
```

[Prev](#) [PHP](#)
config_load

[Home](#)
[Up](#)

[Next](#)
iteration

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 7. Built-in Functions[.]

[Next](#)

include

Attribute Name	Type	Required	Default	Description
file	string	Yes	n/a	The name of the template file to include
assign	string	No	n/a	The name of the variable that the output of <i>include</i> will be assigned to
[var ...]	[var type]	No	n/a	variable to pass local to template
file	string	Yes	n/a	
assign	string	No	n/a	
[var ...]	[var type]	No	n/a	

Include tags are used for including other templates in the current template. Any variables available in the current template are also available within the included template. The *include* tag must have the attribute "file", which contains the template resource path.

Include . . . file .

You can optionally pass the *assign* attribute, which will specify a template variable name that the output of *include* will be assigned to instead of displayed.

assign

Example 7-6. function include

7-6. include

```
{include file="header.tpl"}  
{* body of template goes here *}  
{include file="footer.tpl"}
```

You can also pass variables to included templates as attributes. Any variables explicitly passed to an included template as attributes are only available within the scope of the included file. Attribute variables override current template variables, in the case they are named alike.

...

Example 7-7. function include passing variables

7-7. include

```
{include file="header.tpl" title="Main Menu" table_bgcolor="white"}  
{* body of template goes here *}  
{include file="footer.tpl" logo="http://my.domain.com/logo.png"}
```

Use the syntax for [template resources](#) to include files outside of the \$template_dir directory.

\$template_dir ..

Example 7-8. function include template resource examples

7-8. include

```
{* absolute filepath *}  
{include file="/usr/local/include/templates/header.tpl"}  
  
{* absolute filepath (same thing) *}  
{include file="file:/usr/local/include/templates/header.tpl"}  
  
{* windows absolute filepath (MUST use "file:" prefix)}
```

```
{include file="file:C:/www/pub/templates/header.tpl"}  
{* include from template resource named "db" *}  
{include file="db:header.tpl"}
```

[Prev](#) [PHP](#)

total

[Home](#)

[Up](#)

[Next](#)

include_php

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 7. Built-in Functions[.]

[Next](#)

include_php

Attribute Name	Type	Required	Default	Description
file	string	Yes	n/a	The name of the php file to include
once	boolean	No	true	whether or not to include the php file more than once if included multiple times
assign	string	No	n/a	The name of the variable that the output of include_php will be assigned to
file	string	Yes	n/a	php
once	boolean	No	true	php/php include_once()
assign	string	No	n/a	php

include_php tags are used to include a php script in your template. If security is enabled, then the php script must be located in the \$trusted_dir path. The include_php tag must have the attribute "file", which contains the path to the included php file, either relative to \$trusted_dir, or an absolute path.

```
inlue_php php . $trusted_dir . include_php  
php $trusted_dir .
```

`include_php` is a nice way to handle componentized templates, and keep PHP code separate from the template files. Lets say you have a template that shows your site navigation, which is pulled dynamically from a database. You can keep your PHP logic that grabs database content in a separate directory, and include it at the top of the template. Now you can include this template anywhere without worrying if the database information was assigned by the application before hand.

`include_php php .`

By default, `php` files are only included once even if called multiple times in the template. You can specify that it should be included every time with the `once` attribute. Setting `once` to false will include the `php` script each time it is included in the template.

`php . once . once false.`

You can optionally pass the `assign` attribute, which will specify a template variable name that the output of `include_php` will be assigned to instead of displayed.

`assign php php`

The `smarty` object is available as `$this` within the PHP script that you include.

`php $this smarty .`

Example 7-9. function include_php

7-9. include_php

```
load_nav.php
-----
<?php
    // load in variables from a mysql db and assign them to smarty
    // mysql
    require_once("MySQL.class.php");
    $sql = new MySQL;
    $sql->query("select * from site_nav_sections");
    $this->assign('sections',$sql->record);
```

```
?>

index.tpl
-----
{* absolute path, or relative to $trusted_dir *}
{* $trusted_dir *}
{include_php file="/path/to/load_nav.php"}

{foreach item="curr_section" from=$sections}
    <a href="{$curr_section.url}">{$curr_section.r
{/foreach}
```

[Prev](#) [PHP](#)

include

[Home](#)

[Up](#)

[Next](#)

insert

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 7. Built-in Functions[.]

[Next](#)

insert

Attribute Name	Type	Required	Default	Description
name	string	Yes	n/a	The name of the insert function (insert_name)
assign	string	No	n/a	The name of the template variable the output will be assigned to
script	string	No	n/a	The name of the php script that is included before the insert function is called
[var ...]	[var type]	No	n/a	variable to pass to insert function

name	string	Yes	n/a	
assign	string	No	n/a	
script	string	No	n/a	php
[var ...]	[var type]	No	n/a	

Insert tags work much like include tags, except that insert tags are not cached when you have template [caching](#) enabled. They will be executed on every invocation of the template.

Insert inluce insert .

Let's say you have a template with a banner slot at the top of the page. The banner can contain any mixture of HTML, images, flash, etc. so we can't just use a static link here, and we don't want this contents cached with the page. In comes the insert tag: the template knows #banner_location_id# and #site_id# values (gathered from a config file), and needs to call a function to get the banner contents.

HTMLFLASH. .

#site_id# () .

Example 7-10. function insert 7-10. insert

```
{* example of fetching a banner *}
{insert name="getBanner" lid=#banner_location_id# sid=}
```

In this example, we are using the name "getBanner" and passing the parameters #banner_location_id# and #site_id#. Smarty will look for a function named insert_getBanner() in your PHP application, passing the values of #banner_location_id# and #site_id# as the first argument in an associative array. All insert function names in your application must be prepended with "insert_" to remedy possible function name-space conflicts. Your insert_getBanner() function should do something with the passed values and return the results. These results are then displayed in the template in place of the insert tag. In this example, Smarty would call this function: insert_getBanner(array("lid" => "12345", "sid" => "67890")); and display the returned results in place of the insert tag.

```
getBanner name #banner_location_id# #site_id# .
Smarty php insert_getBanner() #banner_location_id#
#site_id# . insert insert_. insert_getBanner()
Smarty insert_getBanner(array("lid"=>"12345","sid"=>67890));.
```

If you supply the "assign" attribute, the output of the insert tag will be assigned to this template variable instead of being output to the template. NOTE: assigning the output to a template variable isn't too useful with caching enabled.

assign ..

If you supply the "script" attribute, this php script will be included (only once) before the insert function is executed. This is the case where the insert function may not exist yet, and a php script must be included first to make it work. The path can be either absolute, or relative to \$trusted_dir. When security is enabled, the script must reside in \$trusted_dir.

script ()script php .

The Smarty object is passed as the second argument. This way you can reference and modify information in the Smarty object from within the insert function.

Smart \$this smarty .

Technical Note: It is possible to have portions of the template not cached. If you have [caching](#) turned on, insert tags will not be cached. They will run dynamically every time the page is created, even within cached pages. This works good for things like banners, polls, live weather, search results, user feedback areas, etc.

: . , insert . .

[Prev](#) [PHP](#)
include_php

[Home](#)
[Up](#)

[Next](#)
if,elseif,else

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 7. Built-in Functions[.]

[Next](#)

if,elseif,else

if statements in Smarty have much the same flexibility as php if statements, with a few added features for the template engine. Every *if* must be paired with an */if*. *else* and *elseif* are also permitted. "eq", "ne", "neq", "gt", "lt", "lte", "le", "gte", "ge", "is even", "is odd", "is not even", "is not odd", "not", "mod", "div by", "even by", "odd by", "==", "!=" , ">" , "<" , "<=" , ">=" are all valid conditional qualifiers, and must be separated from surrounding elements by spaces.

Smarty if php if . if /if . else
eqneneqgtltltelegegeis evenis oddis not evenis
not oddnotmoddiv byeven byodd by==!=><<=>= .

Example 7-11. if statements

7-11. if

```
{if $name eq "Fred"}  
    Welcome Sir.  
{elseif $name eq "Wilma"}  
    Welcome Ma'am.  
{else}  
    Welcome, whatever you are.  
{/if}  
  
{* an example with "or" logic *}  
{if $name eq "Fred" or $name eq "Wilma"}  
    ...  
{/if}  
  
{* same as above *}  
{if $name == "Fred" || $name == "Wilma"}  
    ...  
{/if}  
  
{* the following syntax will NOT work, conditional qua  
must be separated from surrounding elements by spaces  
{if $name=="Fred" || $name=="Wilma"}
```

```
...  
{/if} ...  
  
{* parenthesis are allowed *}  
{if ( $amount < 0 or $amount > 1000 ) and $volume >= #  
...  
{/if} ...  
  
{* you can also embed php function calls *}  
{if count($var) gt 0}  
...  
{/if} ...  
  
{* test if values are even or odd *}  
{if $var is even}  
...  
{/if}  
{if $var is odd}  
...  
{/if}  
{if $var is not odd}  
...  
{/if} ...  
  
{* test if var is divisible by 4 *}  
{if $var is div by 4}  
...  
{/if} ...  
  
{* test if var is even, grouped by two. i.e.,  
0=even, 1=even, 2=odd, 3=odd, 4=even, 5=even, etc. *}  
{if $var is even by 2}  
...  
{/if} ...  
  
{* 0=even, 1=even, 2=even, 3=odd, 4=odd, 5=odd, etc. *}  
{if $var is even by 3}  
...
```

{/if}

[Prev](#) [PHP](#)
insert

[Home](#)
[Up](#)

[Next](#)
Idelim,rdelim

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 7. Built-in Functions

[Next](#)

Idelim,rdelim

Idelim and rdelim are used for displaying the literal delimiter, in our case "{" or "}". The template engine always tries to interpret delimiters, so this is the way around that.

Idelim rdelim "{" "}"..

Example 7-12. Idelim, rdelim

7-12. Idelim, rdelim

```
{* this will print literal delimiters out of the templ
{ldelim}funcname{rdelim} is how functions look in Smar
```

OUTPUT:

```
{funcname} is how functions look in Smarty!
```

[Prev](#) [PHP](#)

if,elseif,else

[Home](#)

[Up](#)

[Next](#)

literal

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 7. Built-in Functions[.]

[Next](#)

literal

Literal tags allow a block of data to be taken literally, not being interpreted by the Smarty engine. This is handy for things like javascript sections, where there maybe curly braces and such things that would confuse the template parser. Anything within {literal}{/literal} tags is not interpreted, but displayed as-is.

Literal . javascript

Example 7-13. literal tags

7-13. literal

```
{literal}
    <script language=javascript>

        <!--
        function isblank(field) {
        if (field.value == '')
        { return false; }
        else
        {
        document.loginform.submit();
        return true;
        }
        }
        // -->

    </script>
{/literal}
```

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 7. Built-in Functions

[Next](#)

php

php tags allow php to be embedded directly into the template. They will not be escaped, regardless of the [\\$php_handling](#) setting. This is for advanced users only, not normally needed.

php php . [\\$php_handling](#) . .

Example 7-14. php tags

7-14. php

```
{php}
    // including a php script directly
    // from the template.
    include("/path/to/display_weather.php")
{/php}
```

[Prev](#) [PHP](#)

literal

[Home](#)

[Up](#)

[Next](#)

section,sectionelse

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 7. Built-in Functions[.]

[Next](#)

section,sectionelse

Table of Contents

[index](#)
[index_prev](#)
[index_next](#)
[iteration](#)
[first](#)
[last](#)
[rownum](#)
[loop](#)
[show](#)
[total](#)

Attribute Name	Type	Required	Default	Description
name	string	Yes	n/a	The name of the section
loop	[\$variable_name]	Yes	n/a	The name of the variable to determine # of loop iterations
				The index position that the section will begin looping. If the value is negative, the start position is calculated from the end of the array. For example, if there are

start	integer	No	0	seven values in the loop array and start is -2, the start index is 5. Invalid values (values outside of the length of the loop array) are automatically truncated to the closest valid value.
step	integer	No	1	The step value that will be used to traverse the loop array. For example, step=2 will loop on index 0,2,4, etc. If step is negative, it will step through the array backwards.
max	integer	No	1	Sets the maximum number of times the section will loop.
				determines

show	boolean	No	true	whether or not to show this section
name	string	Yes	n/a	
loop	[\$variable_name]	Yes	n/a	
start	integer	No	0	.. 7start -25. ().
step	integer	No	1	. step=2 024. step.
max	integer	No	1	.
show	boolean	No	true	.

Template sections are used for looping over arrays of data. All *section* tags must be paired with */section* tags. Required parameters are *name* and *loop*. The name of the section can be anything you like, made up of letters, numbers and underscores. Sections can be nested, and the nested section names must be unique from each other. The loop variable (usually an array of values) determines the number of times the section will loop. When printing a variable within a section, the section name must be given next to variable name within brackets []. *sectionelse* is executed when there are no values in the loop variable.

```
section .   section .   name   loop .. name .
loop () .                                         section name .
```

Example 7-15. **section**

7-15. **section**

```
{* this example will print out all the values of the $  

{section name=customer loop=$custid}  

    id: {$custid[customer]}<br>  

{/section}
```

OUTPUT:

```
id: 1000<br>
id: 1001<br>
id: 1002<br>
```

Example 7-16. section loop variable 7-16.loop

```
{* the loop variable only determines the number of times you can access any variable from the template within This example assumes that $custid, $name and $address arrays containing the same number of values *}
{section name=customer loop=$custid}
    id: {$custid[customer]}<br>
    name: {$name[customer]}<br>
    address: {$address[customer]}<br>
    <p>
{/section}
```

OUTPUT:

```
id: 1000<br>
name: John Smith<br>
address: 253 N 45th<br>
<p>
id: 1001<br>
name: Jack Jones<br>
address: 417 Mulberry ln<br>
<p>
id: 1002<br>
name: Jane Munson<br>
address: 5605 apple st<br>
<p>
```

Example 7-17. section names

7-17. section

```
{* the name of the section can be anything you like,  
and it is used to reference the data within the section  
{section name=mydata loop=$custid}  
    id: {$custid[mydata]}<br>  
    name: {$name[mydata]}<br>  
    address: {$address[mydata]}<br>  
    <p>  
{/section}
```

Example 7-18. nested sections

7-18. section

```
{* sections can be nested as deep as you like. With nested sections, you can access complex data structures, such as multi-dimensional arrays. In this example, $contact_type[customer] is an array containing all of the contact types for the current customer. *}  
{section name=customer loop=$custid}  
    id: {$custid[customer]}<br>  
    name: {$name[customer]}<br>  
    address: {$address[customer]}<br>  
    {section name=contact loop=$contact_type[customer]}  
        {$contact_type[customer][contact]}: {$contact[contact]}  
    {/section}  
    <p>  
{/section}
```

OUTPUT:

```
id: 1000<br>  
name: John Smith<br>  
address: 253 N 45th<br>  
home phone: 555-555-5555<br>  
cell phone: 555-555-5555<br>  
e-mail: john@mydomain.com<br>  
<p>
```

```
id: 1001<br>
name: Jack Jones<br>
address: 417 Mulberry ln<br>
home phone: 555-555-5555<br>
cell phone: 555-555-5555<br>
e-mail: jack@mydomain.com<br>
<p>
id: 1002<br>
name: Jane Munson<br>
address: 5605 apple st<br>
home phone: 555-555-5555<br>
cell phone: 555-555-5555<br>
e-mail: jane@mydomain.com<br>
<p>
```

Example 7-19. sections and associative arrays

7-19. section

```
{* This is an example of printing an associative array
of data within a section *}
{section name=customer loop=$contacts}
    name: {$contacts[customer].name}<br>
    home: {$contacts[customer].home}<br>
    cell: {$contacts[customer].cell}<br>
    e-mail: {$contacts[customer].email}<p>
{/section}
```

OUTPUT:

```
name: John Smith<br>
home: 555-555-5555<br>
cell: 555-555-5555<br>
e-mail: john@mydomain.com<p>
name: Jack Jones<br>
home phone: 555-555-5555<br>
cell phone: 555-555-5555<br>
e-mail: jack@mydomain.com<p>
```

```
name: Jane Munson<br>
home phone: 555-555-5555<br>
cell phone: 555-555-5555<br>
e-mail: jane@mydomain.com<p>
```

Example 7-20. sectionelse

7-20. sectionelse

```
{* sectionelse will execute if there are no $custid va
{section name=customer loop=$custid}
    id: {$custid[customer]}<br>
{sectionelse}
    there are no values in $custid.
{/section}
```

Sections also have their own variables that handle section properties. These are indicated like so: {\$smarty.section.sectionname.varname}

Section . {\$smarty.section.sectionname.varname}.

NOTE: As of Smarty 1.5.0, the syntax for section property variables has been changed from {%-sectionname.varname%} to {\$smarty.section.sectionname.varname}. The old syntax is still supported, but you will only see reference to the new syntax in the manual examples.

Smarty 1.5.0 section {%-sectionname.varname%}
{\$smarty.section.sectionname.varname}.

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 7. Built-in Functions[.]

[Next](#)

strip

Many times web designers run into the issue where white space and carriage returns affect the output of the rendered HTML (browser "features"), so you must run all your tags together in the template to get the desired results. This usually ends up in unreadable or unmanageable templates.

Web HTML("")�.

Anything within {strip}{/strip} tags in Smarty are stripped of the extra spaces or carriage returns at the beginnings and ends of the lines before they are displayed. This way you can keep your templates readable, and not worry about extra white space causing problems.

Smarty {strip}{/strip} . .

Technical Note: {strip}{/strip} does not affect the contents of template variables. See the [strip modifier function](#).

Example 7-31. strip tags

7-31. strip

```
{* the following will be all run into one line upon output *}
{strip}
<table border=0>
    <tr>
        <td>
            <A HREF="{$url}">
                <font color="red">This is a test</font>
            </A>
        </td>
    </tr>
</table>
{/strip}
```

OUTPUT:

```
<table border=0><tr><td><a href="http://my.domain.com"
```

Notice that in the above example, all the lines begin and end with HTML tags. Be aware that all the lines are run together. If you have plain text at the beginning or end of any line, they will be run together, and may not be desired results.

HTML...

[Prev](#) [PHP](#)

total

[Home](#)

[Up](#)

[Next](#)

Custom Functions

Smarty - the compiling PHP template engine

Smarty - php

[Prev](#)

[PHP](#)

[Next](#)

Chapter 8. Custom Functions[.]

Table of Contents

[assign](#)
[counter](#)
[cycle](#)
[debug](#)
[eval](#)
[fetch](#)
[html_checkboxes](#)
[html_image](#)
[html_options](#)
[html_radios](#)
[html_select_date](#)
[html_select_time](#)
[html_table](#)
[math](#)
[mailto](#)
[popup_init](#)
[popup](#)
[textformat](#)

Smarty comes with several custom functions that you can use in the templates.

Smarty .

assign

Attribute Name	Type	Required	Default	Description
var	string	Yes	n/a	The name of the variable being assigned
value	string	Yes	n/a	The value being assigned

var	string	Yes	n/a	
value	string	Yes	n/a	

assign is used for assigning template variables during the execution of the template.

assign .

Example 8-1. assign

8-1. assign

```
{assign var="name" value="Bob"}
```

The value of \$name is {\$name} .

OUTPUT:

The value of \$name is Bob.

Smarty - the compiling PHP template engine

Smarty - php

[Prev](#)

[PHP](#)

Chapter 8. Custom Functions[
]

[Next](#)

counter

Attribute Name	Type	Required	Default	Description
name	string	No	<i>default</i>	The name of the counter
start	number	No	<i>1</i>	The initial number to start counting from
skip	number	No	<i>1</i>	The interval to count by
direction	string	No	<i>up</i>	the direction to count (up/down)
print	boolean	No	<i>true</i>	Whether or not to print the value
assign	string	No	<i>n/a</i>	the template variable the output will be assigned to

name	string	No	<i>default</i>	
start	number	No	<i>1</i>	
skip	number	No	<i>1</i>	
direction	string	No	<i>up</i>	(<i>/</i>)
print	boolean	No	<i>true</i>	
assign	string	No	<i>n/a</i>	

counter is used to print out a count. counter will remember the count on each iteration. You can adjust the number, the interval and the direction of the count, as well as determine whether or not to print the value. You

can run multiple counters concurrently by supplying a unique name for each one. If you do not supply a name, the name 'default' will be used.

```
counter . counter . interval direction .
"default" .
```

If you supply the special "assign" attribute, the output of the counter function will be assigned to this template variable instead of being output to the template.

```
"assign" assign .
```

Example 8-2. counter

8-2. counter

```
{* initialize the count *}
{counter start=0 skip=2 print=false}

{counter}<br>
{counter}<br>
{counter}<br>
{counter}<br>
```

OUTPUT:

```
2<br>
4<br>
6<br>
8<br>
```

Smarty - the compiling PHP template engine

Smarty - php

[Prev](#)

[PHP](#)

Chapter 8. Custom Functions[.
]

[Next](#)

cycle

Attribute Name	Type	Required	Default	Description
name	string	No	<i>default</i>	The name of the cycle
values	mixed	Yes	<i>N/A</i>	The values to cycle through, either a comma delimited list (see delimiter attribute), or an array of values.
print	boolean	No	<i>true</i>	Whether to print the value or not
advance	boolean	No	<i>true</i>	Whether or not to advance to the next value
delimiter	string	No	,	The delimiter to use in the values attribute.
assign	string	No	<i>n/a</i>	the template variable the output will be assigned to

name	string	No	<i>default</i>	

values	mixed	Yes	N/A	(delimiter) . .
print	boolean	No	true	
advance	boolean	No	true	(false)
delimiter	string	No	,	values . .
assign	string	No	n/a	

Cycle is used to cycle though a set of values. This makes it easy to alternate between two or more colors in a table, or cycle through an array of values.

Cycle ..

You can cycle through more than one set of values in your template by supplying a name attribute. Give each set of values a unique name.

name .

You can force the current value not to print with the print attribute set to false. This would be useful for silently skipping a value.

print false ..

The advance attribute is used to repeat a value. When set to false, the next call to cycle will print the same value.

advance . false .

If you supply the special "assign" attribute, the output of the cycle function will be assigned to this template variable instead of being output to the template.

"assign" assign .

Example 8-3. cycle

8-3. cycle

```
{section name=rows loop=$data}
<tr bgcolor="{cycle values="#eeeeee,#d0d0d0"}">
<td>{$data[rows]}</td>
```

```
</tr>
{/section}
```

OUTPUT:

```
<tr bgcolor="#eeeeee">
<td>1</td>
</tr>
<tr bgcolor="#d0d0d0">
<td>2</td>
</tr>
<tr bgcolor="#eeeeee">
<td>3</td>
</tr>
```

[Prev](#) [PHP](#)
counter

[Home](#)
[Up](#)

[Next](#)
debug

Smarty - the compiling PHP template engine

Smarty - php

[Prev](#)

[PHP](#)

Chapter 8. Custom Functions[
]

[Next](#)

debug

Attribute Name	Type	Required	Default	Description
output	string	No	<i>html</i>	output type, html or javascript

output	string	No	<i>html</i>	html javascript

{debug} dumps the debug console to the page. This works regardless of the [debug](#) settings in Smarty. Since this gets executed at runtime, this is only able to show the assigned variables, not the templates that are in use. But, you see all the currently available variables within the scope of this template.

{debug} . Smarty [debug](#) . . .

[Prev](#) [PHP](#)
[cycle](#)

[Home](#)
[Up](#)

[Next](#)
[eval](#)

Smarty - the compiling PHP template engine

Smarty - php

[Prev](#)

[PHP](#)

Chapter 8. Custom Functions[
]

[Next](#)

eval

Attribute Name	Type	Required	Default	Description
var	mixed	Yes	n/a	variable (or string) to evaluate
assign	string	No	n/a	the template variable the output will be assigned to

var	mixed	Yes	n/a	()
assign	string	No	n/a	

eval is used to evaluate a variable as a template. This can be used for things like embedding template tags/variables into variables or tags/variables into config file variables.

eval . //.

If you supply the special "assign" attribute, the output of the eval function will be assigned to this template variable instead of being output to the template.

"assign" assign .

Technical Note: Evaluated variables are treated the same as templates. They follow the same escapement and security features just as if they were templates.

: ..

Technical Note: Evaluated variables are compiled on every invocation, the compiled versions are not saved! However if you have caching enabled, the output will be cached with the rest of the template.

: !.

Example 8-4. eval

8-4. eval

```
setup.conf
```

```
-----  
  
emphstart = <b>  
emphend = </b>  
title = Welcome to {$company}'s home page!  
ErrorCity = You must supply a {#emphstart#}city{#emphend#}  
ErrorState = You must supply a {#emphstart#}state{#emphend#}
```

```
index.tpl
```

```
-----  
  
{config_load file="setup.conf"}  
  
{eval var=$foo}  
{eval var=#title#}  
{eval var=#ErrorCity#}  
{eval var=#ErrorState# assign="state_error"}  
{$state_error}
```

OUTPUT:

```
This is the contents of foo.  
Welcome to Foobar Pub & Grill's home page!  
You must supply a <b>city</b>.  
You must supply a <b>state</b>.
```

Smarty - the compiling PHP template engine

Smarty - php

[Prev](#)

[PHP](#)

Chapter 8. Custom Functions[.
]

[Next](#)

fetch

Attribute Name	Type	Required	Default	Description
file	string	Yes	n/a	the file, http or ftp site to fetch
assign	string	No	n/a	the template variable the output will be assigned to

file	string	Yes	n/a	httpftp.
assign	string	No	n/a	

fetch is used to fetch files from the local file system, http, or ftp and display the contents. If the file name begins with "http://", the web site page will be fetched and displayed. If the file name begins with "ftp://", the file will be fetched from the ftp server and displayed. For local files, the full system file path must be given, or a path relative to the executed php script.

fetch HTTPFTP. "http://".

If you supply the special "assign" attribute, the output of the fetch function will be assigned to this template variable instead of being output to the template. (new in Smarty 1.5.0)

"assign" assign .(Smarty

Technical Note: This will not support http redirects, be sure to include a trailing slash on your web page fetches where necessary.

: HTTPwebwww.domain.comindex.phpindex.htm
default.phpurlurl.

Technical Note: If template security is turned on and you are

fetching a file from the local file system, this will only allow files from within one of the defined secure directories. (\$secure_dir)

: . (\$secure_dir)

Example 8-5. fetch

8-5. fetch

```
{* include some javascript in your template *}
{fetch file="/export/httpd/www.domain.com/docs/navbar.

{* embed some weather text in your template from another
{fetch file="http://www.myweather.com/68502/"}

{* fetch a news headline file via ftp *}
{fetch file="ftp://user:password@ftp.domain.com/path/t

{* assign the fetched contents to a template variable
{fetch file="http://www.myweather.com/68502/" assign="
{if $weather ne ""}
    <b>{$weather}</b>
{/if}
```

[Prev](#) [PHP](#)

eval

[Home](#)

[Up](#)

[Next](#)

html_checkboxes

Smarty - the compiling PHP template engine

Smarty - php

[Prev](#)

[PHP](#)

Chapter 8. Custom Functions[
]

[Next](#)

html_checkboxes

Attribute Name	Type	Required	Default	Description
name	string	No	<i>checkbox</i>	name of checkbox list
values	array	Yes, unless using options attribute	<i>n/a</i>	an array of values for checkbox buttons
output	array	Yes, unless using options attribute	<i>n/a</i>	an array of output for checkbox buttons
selected	string/array	No	<i>empty</i>	the checked checkbox element(s)
options	associative array	Yes, unless using values and output	<i>n/a</i>	an associative array of values and output
separator	string	No	<i>empty</i>	string of text to separate each checkbox item
labels	boolean	No	<i>true</i>	add <label>-tags to the output

name	string	No	<i>checkbox</i>	
		Yes,		

values	array	options	<i>n/a</i>	
output	array	Yes, options	<i>n/a</i>	
selected	string/array	No	<i>empty</i>	
options	associative array	Yes, values	<i>n/a</i>	
separator	string	No	<i>empty</i>	
labels	boolean	No	<i>true</i>	<label>

`html_checkboxes` is a custom function that creates an html checkbox group with provided data. It takes care of which item(s) are selected by default as well. Required attributes are `values` and `output`, unless you use `options` instead. All output is XHTML compatible.

```
html_checkboxes . . values ouput
XHTML .
```

All parameters that are not in the list above are printed as name/value-pairs inside each of the created `<input>`-tags.

```
<input> "/".
```

Example 8-6. `html_checkboxes`

8-6. `html_checkboxes`

```
index.php:
```

```
require('Smarty.class.php');
$smarty = new Smarty;
$smarty->assign('cust_ids', array(1000,1001,1002,1003)
$smarty->assign('cust_names', array('Joe Schmoe','Jack
$smarty->assign('customer_id', 1001);
$smarty->display('index.tpl');
```

```
index.tpl:
```

```
{html_checkboxes values=$cust_ids checked=$customer_id}
```

index.php:

```
require('Smarty.class.php');
$smarty = new Smarty;
$smarty->assign('cust_checkboxes', array(
    1000 => 'Joe Schmoe',
    1001 => 'Jack Smith',
    1002 => 'Jane Johnson',
    1003 => 'Charlie Brown'));
$smarty->assign('customer_id', 1001);
$smarty->display('index.tpl');
```

index.tpl:

```
{html_checkboxes name="id" options=$cust_checkboxes ch
```

OUTPUT: (both examples)

```
<label><input type="checkbox" name="checkbox[]" value=
<label><input type="checkbox" name="checkbox[]" value=
<label><input type="checkbox" name="checkbox[]" value=
<label><input type="checkbox" name="checkbox[]" value=
```

[Prev](#) [PHP](#)

fetch

[Home](#)

[Up](#)

[Next](#)

html_image

Smarty - the compiling PHP template engine

Smarty - php

[Prev](#)

[PHP](#)

Chapter 8. Custom Functions[
]

[Next](#)

html_image

Attribute Name	Type	Required	Default	Description
file	string	Yes	n/a	name/path to image
border	string	No	0	size of border around image
height	string	No	<i>actual image height</i>	height to display image
width	string	No	<i>actual image width</i>	width to display image
basedir	string	no	<i>web server doc root</i>	directory to base relative paths from
alt	string	no	""	alternative description of the image
href	string	no	n/a	href value to link the image to

file	string	Yes	n/a	
border	string	No	0	
height	string	No	<i>actual image height</i>	
width	string	No	<i>actual image width</i>	
basedir	string	no	<i>web server doc root</i>	
alt	string	no	""	(0)
href	string	no	n/a	

`html_image` is a custom function that generates an HTML tag for an image. The height and width are automatically calculated from the image file if none are supplied.

`html_image` HTML . .

`basedir` is the base directory that relative image paths are based from. If not given, the web server document root (env variable `DOCUMENT_ROOT`) is used as the base. If security is enabled, the path to the image must be within a secure directory.

`basedir . WEB(DOCUMENT_ROOT) . .`

`href` is the href value to link the image to. If link is supplied, an `<a>` tag is put around the image tag.

`href` . .

Technical Note: `html_image` requires a hit to the disk to read the image and calculate the height and width. If you don't use template caching, it is generally better to avoid `html_image` and leave image tags static for optimal performance.

: `html_image` .

Example 8-7. `html_image`

8-7. `html_image`

`index.php`:

```
require('Smarty.class.php');
$smarty = new Smarty;
$smarty->display('index.tpl');
```

`index.tpl`:

```
{html_image file="pumpkin.jpg"}
{html_image file="/path/from/docroot/pumpkin.jpg"}
{html_image file="../path/relative/to/currdir/pumpkin.
```

OUTPUT: (possible)

```



```

[Prev](#) [PHP](#)

html_checkboxes

[Home](#)

[Up](#)

[Next](#)

html_options

Smarty - the compiling PHP template engine

Smarty - php

[Prev](#)

[PHP](#)

Chapter 8. Custom Functions[.
]

[Next](#)

html_options

Attribute Name	Type	Required	Default	Description
values	array	Yes, unless using options attribute	<i>n/a</i>	an array of values for dropdown
output	array	Yes, unless using options attribute	<i>n/a</i>	an array of output for dropdown
selected	string/array	No	<i>empty</i>	the selected option element(s)
options	associative array	Yes, unless using values and output	<i>n/a</i>	an associative array of values and output
name	string	No	<i>empty</i>	name of select group

values	array	Yes, unless using options attribute	<i>n/a</i>	
output	array	Yes, unless using options attribute	<i>n/a</i>	
selected	string/array	No	<i>empty</i>	
options	associative array	Yes, unless using values and output	<i>n/a</i>	
name	string	No	<i>empty</i>	

`html_options` is a custom function that creates html option group with provided data. It takes care of which item(s) are selected by default as well. Required attributes are values and output, unless you use options instead.

`html_options . . values ouput`

If a given value is an array, it will treat it as an html OPTGROUP, and display the groups. Recursion is supported with OPTGROUP. All output is XHTML compatible.

`OPTGROUP . XHTML .`

If the optional `name` attribute is given, the `<select name="groupname"></select>` tags will enclose the option list. Otherwise only the option list is generated.

`name<select name="groupname"></select>.`

All parameters that are not in the list above are printed as name/value-pairs inside the `<select>`-tag. They are ignored if the optional `name` is not given.

`<select> "/". name .`

Example 8-8. `html_options`

8-8. `html_options`

`index.php:`

```
require('Smarty.class.php');
$smarty = new Smarty;
$smarty->assign('cust_ids', array(1000,1001,1002,1003));
$smarty->assign('cust_names', array('Joe Schmoe','Jack
Johnson','Carlie Brown'));
$smarty->assign('customer_id', 1001);
$smarty->display('index.tpl');
```

`index.tpl:`

```
<select name=customer_id>
    {html_options values=$cust_ids selected=$custo
```

```
</select>
```

index.php:

```
require('Smarty.class.php');
$smarty = new Smarty;
$smarty->assign('cust_options', array(
    1001 => 'Joe Schmoe',
    1002 => 'Jack Smith',
    1003 => 'Jane Johnson',
    1004 => 'Charlie Brown'));
$smarty->assign('customer_id', 1001);
$smarty->display('index.tpl');
```

index.tpl:

```
<select name=customer_id>
    {html_options options=$cust_options selected=$customer_id}
</select>
```

OUTPUT: (both examples)

```
<select name=customer_id>
    <option value="1000">Joe Schmoe</option>
    <option value="1001" selected="selected">Jack
    <option value="1002">Jane Johnson</option>
    <option value="1003">Charlie Brown</option>
</select>
```

Smarty - the compiling PHP template engine

Smarty - php

[Prev](#)

[PHP](#)

Chapter 8. Custom Functions[
]

[Next](#)

html_radios

Attribute Name	Type	Required	Default	Description
name	string	No	<i>radio</i>	name of radio list
values	array	Yes, unless using options attribute	<i>n/a</i>	an array of values for radio buttons
output	array	Yes, unless using options attribute	<i>n/a</i>	an array of output for radio buttons
checked	string	No	<i>empty</i>	the checked radio element
options	associative array	Yes, unless using values and output	<i>n/a</i>	an associative array of values and output
separator	string	No	<i>empty</i>	string of text to separate each radio item

name	string	No	<i>radio</i>	
values	array	Yes, options	<i>n/a</i>	
output	array	Yes, options	<i>n/a</i>	
checked	string	No	<i>empty</i>	
options	associative array	Yes, values	<i>n/a</i>	

separator		string		No		<i>empty</i>	
-----------	--	--------	--	----	--	--------------	--

html_radios is a custom function that creates html radio button group with provided data. It takes care of which item is selected by default as well. Required attributes are values and output, unless you use options instead. All output is XHTML compatible.

```
html_radios .. values ouput
XHTML .
```

All parameters that are not in the list above are printed as name/value-pairs inside each of the created <input>-tags.

```
<input> "/".
```

Example 8-9. html_radios

8-9. html_radios

index.php:

```
require('Smarty.class.php');
$smarty = new Smarty;
$smarty->assign('cust_ids', array(1000,1001,1002,1003)
$smarty->assign('cust_names', array('Joe Schmoe','Jack
Johnson','Carlie Brown'));
$smarty->assign('customer_id', 1001);
$smarty->display('index.tpl');
```

index.tpl:

```
{html_radios values=$cust_ids checked=$customer_id out
```

index.php:

```
require('Smarty.class.php');
$smarty = new Smarty;
$smarty->assign('cust_radios', array(
    1001 => 'Joe Schmoe',
```

```
        1002 => 'Jack Smith',
        1003 => 'Jane Johnson',
        1004 => 'Charlie Brown'));
$smarty->assign('customer_id', 1001);
$smarty->display('index.tpl');
```

index.tpl:

```
{html_radios name="id" options=$cust_radios checked=$c
```

OUTPUT: (both examples)

```
<input type="radio" name="id[]" value="1000">Joe Schmo
<input type="radio" name="id[]" value="1001" checked="checked"
<input type="radio" name="id[]" value="1002">Jane John
<input type="radio" name="id[]" value="1003">Charlie B
```

[Prev](#) [PHP](#)
[html_options](#)

[Home](#)
[Up](#)

[Next](#)
[html_select_date](#)

Smarty - the compiling PHP template engine

Smarty - php

[Prev](#)

[PHP](#)

Chapter 8. Custom Functions[
]

[Next](#)

html_select_date

Attribute Name	Type	Required	Default	Description
prefix	string	No	Date_	what to prefix the var name with
time	timestamp/YYYY-MM-DD	No	current time in unix timestamp or YYYY-MM-DD format	what date/time to use
start_year	string	No	current year	the first year in the dropdown, either year number, or relative to current year (+/- N)
end_year	string	No	same as start_year	the last year in the dropdown, either year number, or relative to current year (+/- N)
display_days	boolean	No	true	whether to display days or not
display_months	boolean	No	true	whether to display months or r

display_years	boolean	No	true	whether to display years or not
month_format	string	No	%B	what format the month should be in (strftime)
day_format	string	No	%02d	what format the day output should be in (sprintf)
day_value_format	string	No	%d	what format the day value should be in (sprintf)
year_as_text	boolean	No	false	whether or not to display the year as text
reverse_years	boolean	No	false	display years in reverse order
field_array	string	No	null	if a name is given, the select boxes will be drawn such that the results will be returned to PHP in the form of name[Day], name[Year] name[Month]
day_size	string	No	null	adds size attribute to select tag if

				given
month_size	string	No	null	adds size attribute to select tag if given
year_size	string	No	null	adds size attribute to select tag if given
all_extra	string	No	null	adds extra attributes to all select/input tags if given
day_extra	string	No	null	adds extra attributes to select/input tags if given
month_extra	string	No	null	adds extra attributes to select/input tags if given
year_extra	string	No	null	adds extra attributes to select/input tags if given
field_order	string	No	MDY	the order in which to display the fields
field_separator	string	No	\n	string printed between different fields
				strftime format of th

month_value_format	string	No	%m	month values, default is % for month numbers.
--------------------	--------	----	----	---

prefix	string	No	Date_	
time	timestamp/YYYY-MM-DD	No	UNIX--	(data/time)
start_year	string	No		(/)
end_year	string	No	start_year	(/)
display_days	boolean	No	true	
display_months	boolean	No	true	
display_years	boolean	No	true	
month_format	string	No	%B	(strftime)
day_format	string	No	%02d	(sprintf)
day_value_format	string	No	%d	(sprintf)
year_as_text	boolean	No	false	
reverse_years	boolean	No	false	
field_array	string	No	null	[Day], [Year], [Month] PHP()
day_size	string	No	null	
month_size	string	No	null	
year_size	string	No	null	
all_extra	string	No	null	
day_extra	string	No	null	
month_extra	string	No	null	
year_extra	string	No	null	

field_order	string	No	MDY	
field_separator	string	No	\n	
month_value_format	string	No	%m	strftime %m

html_select_date is a custom function that creates date dropdowns for you. It can display any or all of year, month, and day.

html_select_date . .

Example 8-10. html_select_date

8-10. html_select_date

```
{html_select_date}
```

OUTPUT:

```
<select name="Date_Month">
<option value="1">January</option>
<option value="2">February</option>
<option value="3">March</option>
<option value="4">April</option>
<option value="5">May</option>
<option value="6">June</option>
<option value="7">July</option>
<option value="8">August</option>
<option value="9">September</option>
<option value="10">October</option>
<option value="11">November</option>
<option value="12" selected>December</option>
</select>
<select name="Date_Day">
<option value="1">01</option>
<option value="2">02</option>
<option value="3">03</option>
<option value="4">04</option>
<option value="5">05</option>
<option value="6">06</option>
```

```

<option value="7">07</option>
<option value="8">08</option>
<option value="9">09</option>
<option value="10">10</option>
<option value="11">11</option>
<option value="12">12</option>
<option value="13" selected>13</option>
<option value="14">14</option>
<option value="15">15</option>
<option value="16">16</option>
<option value="17">17</option>
<option value="18">18</option>
<option value="19">19</option>
<option value="20">20</option>
<option value="21">21</option>
<option value="22">22</option>
<option value="23">23</option>
<option value="24">24</option>
<option value="25">25</option>
<option value="26">26</option>
<option value="27">27</option>
<option value="28">28</option>
<option value="29">29</option>
<option value="30">30</option>
<option value="31">31</option>
</select>
<select name="Date_Year">
<option value="2001" selected>2001</option>
</select>

```

Example 8-11. html_select_date

8-11. html_select_date

```

{* start and end year can be relative to current year
{html_select_date prefix="StartDate" time=$time start_
OUTPUT: (current year is 2000)

```

```
<select name="StartDateMonth">
<option value="1">January</option>
<option value="2">February</option>
<option value="3">March</option>
<option value="4">April</option>
<option value="5">May</option>
<option value="6">June</option>
<option value="7">July</option>
<option value="8">August</option>
<option value="9">September</option>
<option value="10">October</option>
<option value="11">November</option>
<option value="12" selected>December</option>
</select>
<select name="StartDateYear">
<option value="1999">1995</option>
<option value="1999">1996</option>
<option value="1999">1997</option>
<option value="1999">1998</option>
<option value="1999">1999</option>
<option value="2000" selected>2000</option>
<option value="2001">2001</option>
</select>
```

[Prev](#) [PHP](#)

html_radios

[Home](#)

[Up](#)

[Next](#)

html_select_time

Smarty - the compiling PHP template engine

Smarty - php

[Prev](#)

[PHP](#)

Chapter 8. Custom Functions[.
]

[Next](#)

html_select_time

Attribute Name	Type	Required	Default	Description
prefix	string	No	Time_	what to prefix the var name with
time	timestamp	No	current time	what date/time to use
display_hours	boolean	No	true	whether or not to display hours
display_minutes	boolean	No	true	whether or not to display minutes
display_seconds	boolean	No	true	whether or not to display seconds
display_meridian	boolean	No	true	whether or not to display meridian (am/pm)
use_24_hours	boolean	No	true	whether or not to use 24 hour clock
minute_interval	integer	No	1	number interval in minute dropdown
second_interval	integer	No	1	number interval in second dropdown
				outputs

field_array	string	No	n/a	values to array of this name
all_extra	string	No	null	adds extra attributes to select/input tags if given
hour_extra	string	No	null	adds extra attributes to select/input tags if given
minute_extra	string	No	null	adds extra attributes to select/input tags if given
second_extra	string	No	null	adds extra attributes to select/input tags if given
meridian_extra	string	No	null	adds extra attributes to select/input tags if given

prefix	string	No	Time_	
time	timestamp	No	UNIX--	(data/time)
display_hours	boolean	No	true	
display_minutes	boolean	No	true	
display_seconds	boolean	No	true	
display_meridian	boolean	No	true	(/)
use_24_hours	boolean	No	true	24
minute_interval	integer	No	1	
second_interval	integer	No	1	

field_array	string	No	n/a	
all_extra	string	No	null	
hour_extra	string	No	null	
minute_extra	string	No	null	
second_extra	string	No	null	
meridian_extra	string	No	null	

html_select_time is a custom function that creates time dropdowns for you. It can display any or all of hour, minute, second and meridian.

html_select_time . .

Example 8-12. html_select_time

8-12. html_select_time

```
{html_select_time use_24_hours=true}
```

OUTPUT:

```
<select name="Time_Hour">
<option value="00">00</option>
<option value="01">01</option>
<option value="02">02</option>
<option value="03">03</option>
<option value="04">04</option>
<option value="05">05</option>
<option value="06">06</option>
<option value="07">07</option>
<option value="08">08</option>
<option value="09" selected>09</option>
<option value="10">10</option>
<option value="11">11</option>
<option value="12">12</option>
<option value="13">13</option>
<option value="14">14</option>
<option value="15">15</option>
<option value="16">16</option>
```

```
<option value="17">17</option>
<option value="18">18</option>
<option value="19">19</option>
<option value="20">20</option>
<option value="21">21</option>
<option value="22">22</option>
<option value="23">23</option>
</select>
<select name="Time_Minute">
<option value="00">00</option>
<option value="01">01</option>
<option value="02">02</option>
<option value="03">03</option>
<option value="04">04</option>
<option value="05">05</option>
<option value="06">06</option>
<option value="07">07</option>
<option value="08">08</option>
<option value="09">09</option>
<option value="10">10</option>
<option value="11">11</option>
<option value="12">12</option>
<option value="13">13</option>
<option value="14">14</option>
<option value="15">15</option>
<option value="16">16</option>
<option value="17">17</option>
<option value="18">18</option>
<option value="19">19</option>
<option value="20" selected>20</option>
<option value="21">21</option>
<option value="22">22</option>
<option value="23">23</option>
<option value="24">24</option>
<option value="25">25</option>
<option value="26">26</option>
<option value="27">27</option>
<option value="28">28</option>
<option value="29">29</option>
```

```
<option value="30">30</option>
<option value="31">31</option>
<option value="32">32</option>
<option value="33">33</option>
<option value="34">34</option>
<option value="35">35</option>
<option value="36">36</option>
<option value="37">37</option>
<option value="38">38</option>
<option value="39">39</option>
<option value="40">40</option>
<option value="41">41</option>
<option value="42">42</option>
<option value="43">43</option>
<option value="44">44</option>
<option value="45">45</option>
<option value="46">46</option>
<option value="47">47</option>
<option value="48">48</option>
<option value="49">49</option>
<option value="50">50</option>
<option value="51">51</option>
<option value="52">52</option>
<option value="53">53</option>
<option value="54">54</option>
<option value="55">55</option>
<option value="56">56</option>
<option value="57">57</option>
<option value="58">58</option>
<option value="59">59</option>
</select>
<select name="Time_Second">
<option value="00">00</option>
<option value="01">01</option>
<option value="02">02</option>
<option value="03">03</option>
<option value="04">04</option>
<option value="05">05</option>
<option value="06">06</option>
```

```
<option value="07">07</option>
<option value="08">08</option>
<option value="09">09</option>
<option value="10">10</option>
<option value="11">11</option>
<option value="12">12</option>
<option value="13">13</option>
<option value="14">14</option>
<option value="15">15</option>
<option value="16">16</option>
<option value="17">17</option>
<option value="18">18</option>
<option value="19">19</option>
<option value="20">20</option>
<option value="21">21</option>
<option value="22">22</option>
<option value="23" selected>23</option>
<option value="24">24</option>
<option value="25">25</option>
<option value="26">26</option>
<option value="27">27</option>
<option value="28">28</option>
<option value="29">29</option>
<option value="30">30</option>
<option value="31">31</option>
<option value="32">32</option>
<option value="33">33</option>
<option value="34">34</option>
<option value="35">35</option>
<option value="36">36</option>
<option value="37">37</option>
<option value="38">38</option>
<option value="39">39</option>
<option value="40">40</option>
<option value="41">41</option>
<option value="42">42</option>
<option value="43">43</option>
<option value="44">44</option>
<option value="45">45</option>
```

```
<option value="46">46</option>
<option value="47">47</option>
<option value="48">48</option>
<option value="49">49</option>
<option value="50">50</option>
<option value="51">51</option>
<option value="52">52</option>
<option value="53">53</option>
<option value="54">54</option>
<option value="55">55</option>
<option value="56">56</option>
<option value="57">57</option>
<option value="58">58</option>
<option value="59">59</option>
</select>
<select name="Time_Meridian">
<option value="am" selected>AM</option>
<option value="pm">PM</option>
</select>
```

[Prev](#) [PHP](#)

html_select_date

[Home](#)

[Up](#)

[Next](#)

html_table

Smarty - the compiling PHP template engine

Smarty - php

[Prev](#)

[PHP](#)

Chapter 8. Custom Functions[
]

[Next](#)

html_table

Attribute Name	Type	Required	Default	Description
loop	array	Yes	<i>n/a</i>	array of data to loop through
cols	integer	No	3	number of columns in the table
table_attr	string	No	<i>border="1"</i>	attributes for table tag
tr_attr	string	No	<i>empty</i>	attributes for tr tag (arrays are cycled)
td_attr	string	No	<i>empty</i>	attributes for td tag (arrays are cycled)
trailpad	string	No	 	value to pad the trailing cells on last row with (if any)
hdir	string	No	<i>right</i>	direction of one row to be rendered. possible values: <i>left / right</i>
vdir	string	No	<i>down</i>	direction of the columns to be rendered. possible values: <i>up /</i>

				<i>down</i>
loop	array	Yes	<i>n/a</i>	
cols	integer	No	3	
table_attr	string	No	<i>border="1"</i>	
tr_attr	string	No	<i>empty</i>	()
td_attr	string	No	<i>empty</i>	()
trailpad	string	No	 	()
hdir	string	No	<i>right</i>	<i>left</i> <i>right</i>
vdir	string	No	<i>down</i>	<i>up</i> <i>down</i>

html_table is a custom function that dumps an array of data into an HTML table. The *cols* attribute determines how many columns will be in the table. The *table_attr* , *tr_attr* and *td_attr* values determine the attributes given to the table, tr and td tags. If *tr_attr* or *td_attr* are arrays, they will be cycled through. *trailpad* is the value put into the trailing cells on the last table row if there are any present.

```
html_table HTML .      cols .  table_attr , tr_attr td_attr tr td
.      tr_attr   td_attr .                                trailpad .
```

Example 8-13. *html_table*

8-13. *html_table*

```
index.php:
```

```
require('Smarty.class.php');
$smarty = new Smarty;
$smarty->assign('data',array(1,2,3,4,5,6,7,8,9));
$smarty->assign('tr',array('bgcolor="#eeeeee"','bgcolc
$smarty->display('index.tpl');
```

```
index.tpl:
```

```
{html_table loop=$data}
{html_table loop=$data cols=4 table_attr='border="0"' }
{html_table loop=$data cols=4 tr_attr=$tr}
```

OUTPUT:

```
<table border="1">
<tr><td>1</td><td>2</td><td>3</td></tr>
<tr><td>4</td><td>5</td><td>6</td></tr>
<tr><td>7</td><td>8</td><td>9</td></tr>
</table>
<table border="0">
<tr><td>1</td><td>2</td><td>3</td><td>4</td></tr>
<tr><td>5</td><td>6</td><td>7</td><td>8</td></tr>
<tr><td>9</td><td>&ampnbsp</td><td>&ampnbsp</td><td>&ampnbsp</td>
</table>
<table border="1">
<tr bgcolor="#eeeeee"><td>1</td><td>2</td><td>3</td><td>4</td></tr>
<tr bgcolor="#dddddd"><td>5</td><td>6</td><td>7</td><td>8</td></tr>
<tr bgcolor="#eeeeee"><td>9</td><td>&ampnbsp</td><td>&ampnbsp</td><td>&ampnbsp</td>
</table>
```

[Prev](#) [PHP](#)

html_select_time

[Home](#)

[Up](#)

[Next](#)

math

Smarty - the compiling PHP template engine

Smarty - php

[Prev](#)

[PHP](#)

Chapter 8. Custom Functions[
]

[Next](#)

math

Attribute Name	Type	Required	Default	Description
equation	string	Yes	n/a	the equation to execute
format	string	No	n/a	the format of the result (sprintf)
var	numeric	Yes	n/a	equation variable value
assign	string	No	n/a	template variable the output will be assigned to
[var ...]	numeric	Yes	n/a	equation variable value

equation	string	Yes	n/a	
format	string	No	n/a	(sprintf)
var	numeric	Yes	n/a	
assign	string	No	n/a	
[var ...]	numeric	Yes	n/a	

math allows the template designer to do math equations in the template. Any numeric template variables may be used in the equations, and the result is printed in place of the tag. The variables used in the equation are passed as parameters, which can be template variables or static values. +, -, /, *, abs, ceil, cos, exp, floor, log, log10, max, min, pi, pow, rand, round, sin, sqrt, srans and tan are all valid operators. Check the PHP documentation for further information on these math functions.

```
math . math . .
log, log10, max, min, pi, pow, rand, round, sin, sqrt, srans tan .
PHP .
```

If you supply the special "assign" attribute, the output of the math function will be assigned to this template variable instead of being output to the template.

```
"assign" assign .
```

Technical Note: math is an expensive function in performance due to its use of the php eval() function. Doing the math in PHP is much more efficient, so whenever possible do the math calculations in PHP and assign the results to the template. Definately avoid repetitive math function calls, like within section loops.

```
: php eval() math . PHP
section math .
```

Example 8-14. math 8-14. math

```
{* $height=4, $width=5 *}

{math equation="x + y" x=$height y=$width}
```

OUTPUT:

9

```
{* $row_height = 10, $row_width = 20, #col_div# = 2, a
{math equation="height * width / division"
height=$row_height
width=$row_width
division=#col_div#}
```

OUTPUT:

100

```
{* you can use parenthesis *}
```

```
{math equation="(( x + y ) / z )" x=2 y=10 z=2}
```

OUTPUT:

6

```
{* you can supply a format parameter in sprintf format
```

```
{math equation="x + y" x=4.4444 y=5.0000 format=".2f"
```

OUTPUT:

9.44

[Prev](#) [PHP](#)

html_table

[Home](#)

[Up](#)

[Next](#)

[mailto](mailto:)

Smarty - the compiling PHP template engine

Smarty - php

[Prev](#)

[PHP](#)

Chapter 8. Custom Functions[
]

[Next](#)

mailto

Attribute Name	Type	Required	Default	Description
address	string	Yes	n/a	the e-mail address
text	string	No	n/a	the text to display, default is the e-mail address
encode	string	No	none	How to encode the e-mail. Can be one of none, hex or javascript.
cc	string	No	n/a	e-mail addresses to carbon copy. Separate entries by a comma.
bcc	string	No	n/a	e-mail addresses to blind carbon copy. Separate entries by a comma.
subject	string	No	n/a	e-mail subject.
newsgroups	string	No	n/a	newsgroups to post to. Separate

				entries by a comma.
followupto	string	No	n/a	addresses to follow up to. Separate entries by a comma.
extra	string	No	n/a	any extra information you want passed to the link, such as style sheet classes

address	string	Yes	n/a	
text	string	No	n/a	
encode	string	No	none	none hexjavascript
cc	string	No	n/a	
bcc	string	No	n/a	
subject	string	No	n/a	
newsgroups	string	No	n/a	
followupto	string	No	n/a	
extra	string	No	n/a	css

mailto automates the creation of mailto links and optionally encodes them. Encoding e-mails makes it more difficult for web spiders to pick up e-mail addresses off of your site.

mailto . .

Technical Note: javascript is probably the most thorough form of encoding, although you can use hex encoding too.

: hex javascript .

Example 8-15. mailto 8-15. mailto

```
{mailto address="me@domain.com"}  
{mailto address="me@domain.com" text="send me some mai  
{mailto address="me@domain.com" encode="javascript"}  
{mailto address="me@domain.com" encode="hex"}  
{mailto address="me@domain.com" subject="Hello to you!  
{mailto address="me@domain.com" cc="you@domain.com, the  
{mailto address="me@domain.com" extra='class="email'']}
```

OUTPUT:

```
<a href="mailto:me@domain.com" >me@domain.com</a>  
<a href="mailto:me@domain.com" >send me some mail</a>  
<SCRIPT language="javascript">eval(unescape('%64%6f%63  
9%74%65%28%27%3c%61%20%68%72%65%66%3d%22%6d%61%69%6c%7  
61%69%6e%2e%63%6f%6d%22%20%3e%6d%65%40%64%6f%6d%61%69%  
%27%29%3b'))</SCRIPT>  
<a href="mailto:%6d%65@%64%6f%6d%61%69%6e.%63%6f%6d" >  
#x6f;#x6d;#x61;#x69;#x6e;#x2e;#x63;#x6f;#x6d;<  
<a href="mailto:me@domain.com?subject=Hello%20to%20you  
<a href="mailto:me@domain.com?cc=you@domain.com%2Cthey  
<a href="mailto:me@domain.com" class="email">me@domair
```

[Prev](#) [PHP](#)
[math](#)

[Home](#)
[Up](#)

[Next](#)
[popup_init](#)

Smarty - the compiling PHP template engine

Smarty - php

[Prev](#)

[PHP](#)

Chapter 8. Custom Functions[
]

[Next](#)

popup_init

popup is an integration of overLib, a library used for popup windows. These are used for context sensitive information, such as help windows or tooltips. `popup_init` must be called once at the top of any page you plan on using the [popup](#) function. overLib was written by Erik Bosrup, and the homepage is located at <http://www.bosrup.com/web/overlib/>.

`popup overLib() . . . popup popup_init . overLib
Erik Bosrup http://www.bosrup.com/web/overlib/.`

As of Smarty version 2.1.2, overLib does NOT come with the release. Download overLib, place the overlib.js file under your document root and supply the relative path to this file as the "src" parameter to `popup_init`.

Smarty 2.1.2 `overLib . overlib.js popup_init
"src".`

Example 8-16. `popup_init`

8-16. `popup_init`

```
{* popup_init must be called once at the top of the pa  
{popup_init src="/javascripts/overlib.js"}
```

[Prev](#) [PHP](#)

mailto

[Home](#)

[Up](#)

[Next](#)

popup

Smarty - the compiling PHP template engine

Smarty - php

[Prev](#)

[PHP](#)

Chapter 8. Custom Functions[
]

[Next](#)

popup

Attribute Name	Type	Required	Default	Description
text	string	Yes	n/a	the text/html to display in the popup window
trigger	string	No	onMouseOver	What is used to trigger the popup window. Can be one of onMouseOver or onClick
sticky	boolean	No	false	Makes the popup stick around until closed
caption	string	No	n/a	sets the caption to title
fgcolor	string	No	n/a	color of the inside of the popup box
bgcolor	string	No	n/a	color of the border of the popup box
textcolor	string	No	n/a	sets the color of the text inside the box
capcolor	string	No	n/a	sets color of the box's caption
closecolor	string	No	n/a	sets the color of the close

				text
textfont	string	No	n/a	sets the font to be used by the main text
captionfont	string	No	n/a	sets the font of the caption
closefont	string	No	n/a	sets the font for the "Close" text
textsize	string	No	n/a	sets the size of the main text's font
captionsize	string	No	n/a	sets the size of the caption's font
closesize	string	No	n/a	sets the size of the "Close" text's font
width	integer	No	n/a	sets the width of the box
height	integer	No	n/a	sets the height of the box
left	boolean	No	false	makes the popups go to the left of the mouse
right	boolean	No	false	makes the popups go to the right of the mouse
center	boolean	No	false	makes the popups go to the center of the mouse
				makes the

above	boolean	No	<i>false</i>	popups go above the mouse. NOTE: only possible when height has been set
below	boolean	No	<i>false</i>	makes the popups go below the mouse
border	integer	No	<i>n/a</i>	makes the border of the popups thicker or thinner
offsetx	integer	No	<i>n/a</i>	how far away from the pointer the popup will show up, horizontally
offsety	integer	No	<i>n/a</i>	how far away from the pointer the popup will show up, vertically
fgbackground	url to image	No	<i>n/a</i>	defines a picture to use instead of color for the inside of the popup.
				defines a picture to use instead of

bgbbackground	url to image	No	n/a	color for the border of the popup. NOTE: You will want to set bgcolor to "" or the color will show as well. NOTE: When having a Close link, Netscape will re-render the table cells, making things look incorrect
closetext	string	No	n/a	sets the "Close" text to something else
noclose	boolean	No	n/a	does not display the "Close" text on stickies with a caption
status	string	No	n/a	sets the text in the browsers status bar
autostatus	boolean	No	n/a	sets the status bar's text to the popup's text. NOTE: overrides status setting
				sets the

autostatuscap	string	No	n/a	status bar's text to the caption's text. NOTE: overrides status and autostatus settings
inarray	integer	No	n/a	tells overLib to read text from this index in the ol_text array, located in overlib.js. This parameter can be used instead of text
caparray	integer	No	n/a	tells overLib to read the caption from this index in the ol_caps array
capicon	url	No	n/a	displays the image given before the popup caption
snapx	integer	No	n/a	snaps the popup to an even position in a horizontal grid
snapy	integer	No	n/a	snap the popup to an even position

				in a vertical grid
fixx	integer	No	n/a	locks the popups horizontal position Note: overrides all other horizontal placement
fixy	integer	No	n/a	locks the popups vertical position Note: overrides all other vertical placement
background	url	No	n/a	sets image to be used instead of table box background
padx	integer,integer	No	n/a	pads the background image with horizontal whitespace for text placement. Note: this is a two parameter command
				pads the background image with vertical

pady	integer,integer	No	<i>n/a</i>	whitespace for text placement. Note: this is a two parameter command
fullhtml	boolean	No	<i>n/a</i>	allows you to control the html over a background picture completely. The html code is expected in the "text" attribute
frame	string	No	<i>n/a</i>	controls popups in a different frame. See the overlib page for more info on this function
timeout	string	No	<i>n/a</i>	calls the specified javascript function and takes the return value as the text that should be displayed in the popup window
				makes that

delay	integer	No	n/a	popup behave like a tooltip. It will popup only after this delay in milliseconds
hauto	boolean	No	n/a	automatically determine if the popup should be to the left or right of the mouse.
vauto	boolean	No	n/a	automatically determine if the popup should be above or below the mouse.

text	string	Yes	n/a	
trigger	string	No	onMouseOver onMouseOver() onClick()	
sticky	boolean	No	false	
caption	string	No	n/a	
fgcolor	string	No	n/a	
bgcolor	string	No	n/a	
textcolor	string	No	n/a	
capcolor	string	No	n/a	
closecolor	string	No	n/a	""
textfont	string	No	n/a	

captionfont	string	No	n/a	
closefont	string	No	n/a	""
textsize	string	No	n/a	
captionsize	string	No	n/a	
closesize	string	No	n/a	""
width	integer	No	n/a	sets the width of the box
height	integer	No	n/a	sets the height of the box
left	boolean	No	false	
right	boolean	No	false	
center	boolean	No	false	
above	boolean	No	false	. : height
below	boolean	No	false	
border	integer	No	n/a	
offsetx	integer	No	n/a	
offsety	integer	No	n/a	
fgbackground	url to image	No	n/a	
bgbackground	url to image	No	n/a	. 1: bgcolor "". 2: Netscape . .
closetext	string	No	n/a	
noclose	boolean	No	n/a	
status	string	No	n/a	
autostatus	boolean	No	n/a	. : status
autostatuscap	string	No	n/a	. : status autostatus
inarray	integer	No	n/a	overLibol_text text. text
				overLibol_caps

caparray	integer	No	n/a	caption.
capicon	url	No	n/a	
snapx	integer	No	n/a	
snapy	integer	No	n/a	
fixx	integer	No	n/a	.:
fixy	integer	No	n/a	.:
background	url	No	n/a	
padx	integer,integer	No	n/a	.:
pady	integer,integer	No	n/a	.:
fullhtml	boolean	No	n/a	HTML. HTML "text"
frame	string	No	n/a	. overlib . .
timeout	string	No	n/a	javascript.
delay	integer	No	n/a	()
hauto	boolean	No	n/a	.
vauto	boolean	No	n/a	.

popup is used to create javascript popup windows.

popup javascript.

Example 8-17. popup

8-17. popup

```
{* popup_init must be called once at the top of the page *}
{popup_init src="/javascripts/overlib.js"}
```

```
{* create a link with a popup window when you move your mouse over it *}
<A href="mypage.html" {popup text="This link takes you to another page."}
```

```
{* you can use html, links, etc in your popup text *}
<A href="mypage.html" {popup sticky=true caption="mypage" text="<UL><LI>links<LI>pages<LI>images</UL>" snapx=10}
```

[Prev](#) [PHP](#)

popup_init

[Home](#)

[Up](#)

[Next](#)

textformat

Smarty - the compiling PHP template engine

Smarty - php

[Prev](#)

[PHP](#)

Chapter 8. Custom Functions[
]

[Next](#)

textformat

Attribute Name	Type	Required	Default	Description
style	string	No	n/a	preset style
indent	number	No	0	The number of chars to indent every line
indent_first	number	No	0	The number of chars to indent the first line
indent_char	string	No	(single space)	The character (or string of chars) to indent with
wrap	number	No	80	How many characters to wrap each line to
wrap_char	string	No	\n	The character (or string of chars) to break each line with
wrap_cut	boolean	No	false	If true, wrap will break the line at the exact character instead of at a word

				boundary
assign	string	No	n/a	the template variable the output will be assigned to

style	string	No	n/a	
indent	number	No	0	
indent_first	number	No	0	
indent_char	string	No	(single space)	()
wrap	number	No	80	
wrap_char	string	No	\n	()
wrap_cut	boolean	No	false	..
assign	string	No	n/a	

textformat is a block function used to format text. It basically cleans up spaces and special characters, and formats paragraphs by wrapping at a boundary and indenting lines.

textformat . .

You can set the parameters explicitly, or use a preset style. Currently "email" is the only available style.

. "email".

Example 8-18. textformat 8-18. textformat

```
{textformat wrap=40}
```

```
This is foo.  
This is foo.  
This is foo.
```

```
This is foo.  
This is foo.  
This is foo.  
  
This is bar.  
  
bar foo bar foo foo.  
bar foo bar foo foo.
```

```
{/textformat}
```

OUTPUT:

```
This is foo. This is foo. This is foo.  
This is foo. This is foo. This is foo.
```

This is bar.

```
bar foo bar foo foo. bar foo bar foo  
foo. bar foo bar foo foo. bar foo bar  
foo foo. bar foo bar foo foo. bar foo  
bar foo foo. bar foo bar foo foo.
```

```
{textformat wrap=40 indent=4}
```

```
This is foo.  
This is foo.
```

This is bar.

```
bar foo bar foo foo.  
bar foo bar foo foo.
```

```
{/textformat}
```

OUTPUT:

```
This is foo. This is foo. This is  
foo. This is foo. This is foo. This  
is foo.
```

```
This is bar.
```

```
bar foo bar foo foo. bar foo bar foo  
foo. bar foo bar foo foo. bar foo  
bar foo foo. bar foo bar foo foo.  
bar foo bar foo foo. bar foo bar  
foo foo.
```

```
{textformat wrap=40 indent=4 indent_first=4}
```

```
This is foo.  
This is foo.
```

```
This is bar.
```

```
bar foo bar foo foo.  
bar foo bar foo foo.  
bar foo bar foo foo.
```

bar	foo	bar	foo	foo
bar	foo	bar	foo	foo
bar	foo	bar	foo	foo
bar	foo	bar	foo	foo

{/textformat}

OUTPUT:

This is foo. This is foo. This
is foo. This is foo. This is foo.
This is foo.

This is bar.

bar foo bar foo foo. bar foo bar
foo foo. bar foo bar foo foo. bar
foo bar foo foo. bar foo bar foo
foo. bar foo bar foo foo. bar foo
bar foo foo.

{textformat style="email"}

This is foo.
This is foo.

This is bar.

```
{/textformat}
```

OUTPUT:

This is foo. This is foo. This is foo. This is foo. Th
foo.

This is bar.

bar foo bar foo foo. bar foo bar foo foo. bar foo bar
bar foo foo. bar foo bar foo foo. bar foo bar foo foo.
foo.

[Prev](#) [PHP](#)
popup

[Home](#)
[Up](#)

[Next](#)
Config Files

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

Chapter 9. Config Files

- Config files are handy for designers to manage global template variables from one file. One example is template colors. Normally if you wanted to change the color scheme of an application, you would have to go through each and every template file and change the colors. With a config file, the colors can be kept in one place, and only one file needs to be updated.

Example 9-1. Example of config file syntax

9-1

```
# global variables
pageTitle = "Main Menu"
bodyBgColor = #000000
tableBgColor = #000000
rowBgColor = #00ff00

[Customer]
pageTitle = "Customer Info"

[Login]
pageTitle = "Login"
focus = "username"
Intro = """This is a value that spans more
than one line. you must enclose
it in triple quotes."""

# hidden section
[.Database]
host=my.domain.com
db=ADDRESSBOOK
user=php-user
pass=foobar
```

Values of config file variables can be in quotes, but not necessary. You can use either single or double quotes. If you have a value that spans more than one line, enclose the entire value with triple quotes (""""). You can put comments into config files by any syntax that is not a valid config file syntax. We recommend using a # (hash) at the beginning of the line.

("""")

This config file example has two sections. Section names are enclosed in brackets []. Section names can be arbitrary strings not containing [or] symbols. The four variables at the top are global variables, or variables not within a section. These variables are always loaded from the config file. If a particular section is loaded, then the global variables and the variables from that section are also loaded. If a variable exists both as a global and in a section, the section variable is used. If you name two variables the same within a section, the last one will be used.

["","[",""]"]

Config files are loaded into templates with the built-in function [{config load}](#).

[{ config load }](#)

You can hide variables or entire sections by prepending the variable name or section name with a period. This is useful if your application reads the config files and gets sensitive data from them that the template engine does not need. If you have third parties doing template editing, you can be certain that they cannot read sensitive data from the config file by loading it into the template.

[Prev](#) [PHP](#)

textformat

[Home](#)

[Up](#)

[Next](#)

Debugging Console

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

Chapter 10. Debugging Console

- There is a debugging console included with Smarty. The console informs you of all the included templates, assigned variables and config file variables for the current invocation of the template. A template named "debug.tpl" is included with the distribution of Smarty which controls the formatting of the console. Set \$debugging to true in Smarty, and if needed set \$debug_tpl to the template resource path for debug.tpl (this is in SMARTY_DIR by default.) When you load the page, a javascript console window should pop up and give you the names of all the included templates and assigned variables for the current page. To see the available variables for a particular templates, see the [{\\$debug}](#) template function. To disable the debugging console, set \$debugging to false. You can also temporarily turn on the debugging console by putting SMARTY_DEBUG in the URL if you enable this option with [\\$debugging_ctrl](#).

SMARTY debug.tpl

true \$debug_tpl

[{\\$debug}](#)

SMARTY_DEBUG URL

SMARTY_DIRJAVASCRIPT

\$debugging false

Technical Note: The debugging console does not work when you use the fetch() API, only when using display(). It is a set of javascript statements added to the very bottom of the generated template. If you do not like javascript, you can edit the debug.tpl template to format the output however you like. Debug data is not cached and debug.tpl info is not included in the output of the debug console.

fetch() API display() javascript
debug.tpl

deb

Note: The load times of each template and config file are in seconds, or fractions thereof.

[Prev](#) [PHP](#)

Config Files

[Home](#)

[Up](#)

[Next](#)

Smarty For
Programmers

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

III. Smarty For Programmers

Table of Contents ▾

- 11. [Constants](#)
 -
 - 12. [Variables](#)
 -
 - 13. [Methods](#)
 -
 - 14. [Caching](#)
 -
 - 15. [Advanced Features](#)
 -
 - 16. [Extending Smarty With Plugins \[Smarty\]](#)
-

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Debugging Console

Constants

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

Chapter 11. Constants

Table of Contents

[SMARTY_DIR](#) [Smarty]

SMARTY_DIR

Smarty

This should be the full system path to the location of the Smarty class files. If this is not defined, then Smarty will attempt to determine the appropriate value automatically. If defined, the path must end with a slash.

SmartySmartySmarty,

Example 11-1. SMARTY_DIR

```
// set path to Smarty directory
define("SMARTY_DIR", "/usr/local/lib/php/Smarty/");

require_once(SMARTY_DIR."Smarty.class.php");
```

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Smarty For
Programmers

[Up](#)

Variables

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

Chapter 12. Variables

Table of Contents []

[\\$template_dir](#) []
[\\$compile_dir](#) []
[\\$config_dir](#) []
[\\$plugins_dir](#) []
[\\$debugging](#) []
[\\$debug_tpl](#)
[\\$debugging_ctrl](#) []
[\\$global_assign](#) []
[\\$undefined](#) []
[\\$autoload_filters](#) []
[\\$compile_check](#) []
[\\$force_compile](#) []
[\\$caching](#) []
[\\$cache_dir](#) []
[\\$cache_lifetime](#) []
[\\$cache_handler_func](#) []
[\\$cache_modified_check](#) []
[\\$config_overwrite](#) []
[\\$config_booleanize](#) []
[\\$config_read_hidden](#) []
[\\$config_fix_newlines](#) []
[\\$default_template_handle_r_func](#) []
[\\$php_handling](#) [PHP]
[\\$security](#) []
[\\$secure_dir](#) []
[\\$security_settings](#) []
[\\$trusted_dir](#) []
[\\$left_delimiter](#) []
[\\$right_delimiter](#) []
[\\$compiler_class](#) []
[\\$request_vars_order](#) []
[\\$request_use_auto_globals](#) []

[\\$compile_id](#) [ID]
[\\$use_sub_dirs](#) []
[\\$default_modifiers](#) []
[\\$default_resource_type](#) []

\$template_dir

This is the name of the default template directory. If you do not supply a resource type when including files, they will be found here. By default this is "./templates", meaning that it will look for the templates directory in the same directory as the executing php script.

“./templates”php

Technical Note: It is not recommended to put this directory under the web server document root.

web

[Prev](#) [PHP](#)

[Constants](#)[]

[Home](#)

[Up](#)

[Next](#)

[]\$compile_dir

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$compile_dir

This is the name of the directory where compiled templates are located. By default this is "./templates_c", meaning that it will look for the compile directory in the same directory as the executing php script.

“./templates_c”php

Technical Note: This setting must be either a relative or absolute path. include_path is not used for writing files.

Technical Note: It is not recommended to put this directory under the web server document root.

web

[Prev](#) [PHP](#)

Variables[]

[Home](#)

[Up](#)

[Next](#)

[]\$config_dir

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$config_dir

This is the directory used to store config files used in the templates.
Default is "./configs", meaning that it will look for the configs directory in
the same directory as the executing php script.

“./configs”php

Technical Note: It is not recommended to put this directory under
the web server document root.

web

[Prev](#) [PHP](#)

[\\$compile_dir\[\]](#)

[Home](#)

[Up](#)

[Next](#)

[\[\]\\$plugins_dir](#)

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$plugins_dir

This is the directories where Smarty will look for the plugins that it needs. Default is "plugins" under the SMARTY_DIR. If you supply a relative path, Smarty will first look under the SMARTY_DIR, then relative to the cwd (current working directory), then relative to each entry in your PHP include path.

SmartySMARTY“plugins”SmartySMARTYphp

Technical Note: For best performance, do not setup your plugins_dir to have to use the PHP include path. Use an absolute pathname, or a path relative to SMARTY_DIR or the cwd.
phpSMARTY

[Prev](#) [PHP](#)

[\\$config_dir\[\]](#)

[Home](#)

[Up](#)

[Next](#)

[\[\]\\$debugging](#)

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$debugging

This enables the [debugging console](#). The console is a javascript window that informs you of the included templates and assigned variables for the current template page.

javascript

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

`$plugins_dir[]`

[Up](#)

`[]$debug_tpl`

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$debug_tpl

This is the name of the template file used for the debugging console. By default, it is named debug.tpl and is located in the [SMARTY_DIR](#).
debug.tplSmarty

[Prev](#) [PHP](#)

\$debugging[]

[Home](#)

[Up](#)

[Next](#)

[]\$debugging_ctrl

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$debugging_ctrl

This allows alternate ways to enable debugging. NONE means no alternate methods are allowed. URL means when the keyword SMARTY_DEBUG is found in the QUERY_STRING, debugging is enabled for that invocation of the script. If \$debugging is true, this value is ignored.

NONEURLSMARTY_DEBUGQUERY_STRING

\$debugging

[Prev](#) [PHP](#)

\$debug_tpl[]

[Home](#)

[Up](#)

[Next](#)

[]\$global_assign

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$global_assign

This is a list of variables that are always implicitly assigned to the template engine. This is handy for making global variables or server variables available to all templates without having to manually assign them. Each element in the \$global_assign should be either a name of the global variable, or a key/value pair, where the key is the name of the global array and the value is the array of variables to be assigned from that global array. \$SCRIPT_NAME is globally assigned by default from \$HTTP_SERVER_VARS.

/\$SCRIPT_NAME\$HTTP_SERVER_VARS

Technical Note: Server variables can be accessed through the \$smarty variable, such as {\$smarty.server.SCRIPT_NAME}. See the section on the [\\$smarty](#) variable.

\$smarty{\$smarty.server.SCRIPT_NAME}\$smarty

[Prev](#) [PHP](#)

[\\$debugging_ctrl\(\)](#)

[Home](#)

[Up](#)

[Next](#)

[\[\]\\$undefined](#)

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$undefined

This sets the value of \$undefined for Smarty, default is null. Currently this is only used to set undefined variables in \$global_assign to a default value.

Smarty,..

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

[\\$global_assign\[\]](#)

[Up](#)

[\[\]\\$autoload_filters](#)

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$autoload_filters

If there are some filters that you wish to load on every template invocation, you can specify them using this variable and Smarty will automatically load them for you. The variable is an associative array where keys are filter types and values are arrays of the filter names. For example:

,Smarty.,,:

```
$smarty->autoload_filters = array('pre' => array('tr  
'output' => array('convert')));
```

[Prev](#) [PHP](#)

[\\$undefined\[\]](#)

[Home](#)

[Up](#)

[Next](#)

[\[\]\\$compile_check](#)

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$compile_check

Upon each invocation of the PHP application, Smarty tests to see if the current template has changed (different time stamp) since the last time it was compiled. If it has changed, it recompiles that template. If the template has not been compiled, it will compile regardless of this setting. By default this variable is set to true. Once an application is put into production (templates won't be changing), the compile_check step is no longer needed. Be sure to set \$compile_check to "false" for maximal performance. Note that if you change this to "false" and a template file is changed, you will *not* see the change since the template will not get recompiled. If caching is enabled and compile_check is enabled, then the cache files will get regenerated if an involved template file or config file was updated. See [\\$force_compile](#) or [clear_compiled_tpl](#).

PHP,Smarty ,,,,true.(),,\$compile_check"false":

"false",,,,,,

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

[\\$autoload_filters\(\)](#)

[Up](#)

[\[\]\\$force_compile](#)

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$force_compile

This forces Smarty to (re)compile templates on every invocation. This setting overrides \$compile_check. By default this is disabled. This is handy for development and debugging. It should never be used in a production environment. If caching is enabled, the cache file(s) will be regenerated every time.

Smarty().\$compile_check.,....,

[Prev](#) [PHP](#)

[\\$compile_check\(\)](#)

[Home](#)

[Up](#)

[Next](#)

[\[\]\\$caching](#)

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$caching

This tells Smarty whether or not to cache the output of the templates. By default this is set to 0, or disabled. If your templates generate redundant redundant content, it is advisable to turn on caching. This will result in significant performance gains. You can also have multiple caches for the same template. A value of 1 or 2 enables caching. 1 tells Smarty to use the current \$cache_lifetime variable to determine if the cache has expired. A value of 2 tells Smarty to use the cache_lifetime value at the time the cache was generated. This way you can set the cache_lifetime just before fetching the template to have granular control over when that particular cache expires. See also [is_cached](#).

Smarty.,0.,...12.1Smarty
cache_lifetime..

If \$compile_check is enabled, the cached content will be regenerated if any of the templates or config files that are part of this cache are changed. If \$force_compile is enabled, the cached content will always be regenerated.

,(),...

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

[\\$force_compile\[\]](#)

[Up](#)

[\[\]\\$cache_dir](#)

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$cache_dir

This is the name of the directory where template caches are stored. By default this is "./cache", meaning that it will look for the cache directory in the same directory as the executing php script. You can also use your own custom cache handler function to control cache files, which will ignore this setting.

.,"./cache",php..,

Technical Note: This setting must be either a relative or absolute path. include_path is not used for writing files.

...

Technical Note: It is not recommended to put this directory under the web server document root.

:web.

[Prev](#) [PHP](#)

\$caching[]

[Home](#)

[Up](#)

[Next](#)

[]\$cache_lifetime

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$cache_lifetime

This is the length of time in seconds that a template cache is valid. Once this time has expired, the cache will be regenerated. \$caching must be set to "true" for \$cache_lifetime to have any purpose. A value of -1 will force the cache to never expire. A value of 0 will cause the cache to always regenerate (good for testing only, to disable caching a more efficient method is to set [\\$caching = false.](#))

((),,\$caching\$cache_lifetime"true".-1,.0(, [\\$caching = false.](#))

If [\\$force_compile](#) is enabled, the cache files will be regenerated every time, effectively disabling caching. You can clear all the cache files with the [clear_all_cache\(\)](#) function, or individual cache files (or groups) with the [clear_cache\(\)](#) function.

,,, [clear_all_cache\(\)](#) , [clear_cache\(\)](#) ().

Technical Note: If you want to give certain templates their own cache lifetime, you could do this by setting [\\$caching = 2](#), then set \$cache_lifetime to a unique value just before calling display() or fetch().

:display()fetch(), [\\$caching = 2](#),\$cache_lifetime.

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

\$cache_dir[]

[Up](#)

[
]\$cache_handler_func

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$cache_handler_func

You can supply a custom function to handle cache files instead of using the built-in method using the \$cache_dir. See the custom [cache handler function section](#) for details.

,\$cache_dir.: [cache handler function section](#)

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

\$cache_lifetime[]

[Up](#)

[
]\$cache_modified_check

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$cache_modified_check

If set to true, Smarty will respect the If-Modified-Since header sent from the client. If the cached file timestamp has not changed since the last visit, then a "304 Not Modified" header will be sent instead of the content. This works only on cached content without [{\\$insert}](#) tags.

,SmartyIf-Modified-Since.,"304 Not Modified",,. [{\\$insert}](#).

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

\$cache_handler_func[
]

[Up](#)

[]\$config_overwrite

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$config_overwrite

If set to true, variables read in from config files will overwrite each other. Otherwise, the variables will be pushed onto an array. This is helpful if you want to store arrays of data in config files, just list each element multiple times. true by default.

.....

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

\$cache_modified_check[
]

[Up](#)

[
]\$config_booleanize

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$config_booleanize

If set to true, config file values of on/true/yes and off/false/no get converted to boolean values automatically. This way you can use the values in the template like so: {if #foobar#} ... {/if}. If foobar was on, true or yes, the {if} statement will execute. true by default.

,on/true/yes
yes, {if}...
off/false/no.,{if}

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

\$config_overwrite[]

[Up](#)

[
]\$config_read_hidden

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$config_read_hidden

If set to true, hidden sections (section names beginning with a period) in config files can be read from templates. Typically you would leave this false, that way you can store sensitive data in the config files such as database parameters and not worry about the template loading them. false by default.

,(*****),,,,...

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

\$config_booleanize[
]

[Up](#)

[
]\$config_fix_newlines

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$config_fix_newlines

If set to true, mac and dos newlines (\r and \r\n) in config files are converted to \n when they are parsed. true by default.
,mac dos (\r and \r\n)\n.

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

\$config_read_hidden[
]

[Up](#)

[
]\$default_template_handler_func

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$default_template_handler_func

This function is called when a template cannot be obtained from its resource.

.

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

\$config_fix_newlines[
]

[Up](#)

[php]\$php_handling

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$php_handling php

This tells Smarty how to handle PHP code embedded in the templates.
There are four possible settings, default being

SMARTY_PHP_PASSTHRU. Note that this does NOT affect php code
within `{php}{/php}` tags in the template.

Smartyphp.,SMARTY_PHP_PASSTHRU. `{php}{/php}`php.

- SMARTY_PHP_PASSTHRU - Smarty echos tags as-is.Smarty
- SMARTY_PHP_QUOTE - Smarty quotes the tags as html
entities.Smarty
html
- SMARTY_PHP_REMOVE - Smarty removes the tags from the
templates.Smarty
- SMARTY_PHP_ALLOW - Smarty will execute the tags as PHP
code.Smarty
php.

NOTE: Embedding PHP code into templates is highly discouraged. Use
[custom functions](#) or [modifiers](#) instead.

php.

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

\$default_template_handler_func[
]

[Up](#)

[]\$security

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$security

\$security true/false, default is false. Security is good for situations when you have untrusted parties editing the templates (via ftp for example) and you want to reduce the risk of system security compromises through the template language. Turning on security enforces the following rules to the template language, unless specifically overridden with \$security_settings: ...(ftp),...,\$security_settings.

- If \$php_handling is set to SMARTY_PHP_ALLOW, this is implicitly changed to SMARTY_PHP_PASSTHRU
\$php_handlingSMARTY_PHP_ALLOW,SMARTY_PHP_PASSTHRU
- PHP functions are not allowed in IF statements, except those specified in the \$security_settings
PHPIF,\$security_settings.
- templates can only be included from directories listed in the \$secure_dir array
\$secure_dir.
- local files can only be fetched from directories listed in the \$secure_dir array using {fetch}
{fetch}{\$secure_dir}.
- {php}{/php} tags are not allowed
{php}{/php}.
- PHP functions are not allowed as modifiers, except those specified in the \$security_settings
PHP,\$security_settings.

[Prev](#) [PHP](#)

\$php_handling[php]

[Home](#)

[Up](#)

[Next](#)

[]\$secure_dir

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$secure_dir

This is an array of all local directories that are considered secure.

{include} and {fetch} use this when security is enabled.

.{include} {fetch}.

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

[\\$security\[\]](#)

[Up](#)

[\[\]\\$security_settings](#)

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$security_settings

These are used to override or specify the security settings when security is enabled. These are the possible settings:

..

- PHP_HANDLING - true/false. If set to true, the \$php_handling setting is not checked for security.
PHP_HANDLING - .,\$php_handling.
- IF_FUNCS - This is an array of the names of permitted PHP functions in IF statements.
IF_FUNCS - ifphp.
- INCLUDE_ANY - true/false. If set to true, any template can be included from the file system, regardless of the \$secure_dir list.
INCLUDE_ANY - .,\$secure_dir.
- PHP_TAGS - true/false. If set to true, {php}{/php} tags are permitted in the templates.
PHP_TAGS - .,{php}{/php}.
- MODIFIER_FUNCS - This is an array of the names of permitted PHP functions used as variable modifiers.
MODIFIER_FUNCS - php.

[Prev](#) [PHP](#)

[\\$secure_dir](#)

[Home](#)

[Up](#)

[Next](#)

[\[\]\\$trusted_dir](#)

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$trusted_dir

\$trusted_dir is only for use when \$security is enabled. This is an array of all directories that are considered trusted. Trusted directories are where you keep php scripts that are executed directly from the templates with [{\\$include_php}](#).

\$security..php, [{\\$include_php}](#).

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

\$security_settings[]

[Up](#)

[]\$left_delimiter

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$left_delimiter

This is the left delimiter used by the template language. Default is "{".
,"{".

[Prev](#) [PHP](#)

\$trusted_dir[]

[Home](#)

[Up](#)

[Next](#)

[]\$right_delimiter

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$right_delimiter

This is the right delimiter used by the template language. Default is "}".
,"}".

[Prev](#) [PHP](#)

\$left_delimiter[]

[Home](#)

[Up](#)

[Next](#)

[]\$compiler_class

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$compiler_class

Specifies the name of the compiler class that Smarty will use to compile the templates. The default is 'Smarty_Compiler'. For advanced users only.`Smarty::'Smarty_Compiler'..`

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

`$right_delimiter[]`

[Up](#)

[
]`]$request_vars_order`

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$request_vars_order

The order in which request variables are registered, similar to
variables_order in php.ini
,php.ini.

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

[\\$compiler_class\[\]](#)

[Up](#)

[
]\$request_use_auto_globals

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$request_use_auto_globals

Specifies if Smarty should use php's \$HTTP_*_VARS[] (\$request_use_auto_globals=false which is the default value) or \$_*[] (\$request_use_auto_globals=true). This affects templates that make use of {\$smarty.request.*}, {\$smarty.get.*} etc.. Caution: If you set \$request_use_auto_globals to true, [variable.request.vars.order](#) has no effect but php's configuration value gpc_order is used.

Smartyphp\$HTTP_*_VARS[](\$request_use_auto_globals=false)\$_*[]
(\$request_use_auto_globals=true).{\$smarty.request.*}, {\$sma
.:\$request_use_auto_globals, [variable.request.vars.](#)
gpc_order.

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

\$request_vars_order[
]

[Up](#)

[id]\$compile_id

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$compile_id id

Persistent compile identifier. As an alternative to passing the same compile_id to each and every function call, you can set this compile_id and it will be used implicitly thereafter.

.,id,id.

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

\$request_use_auto_globals[
]

[Up](#)

[\[\]\\$use_sub_dirs](#)

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$use_sub_dirs

Set this to false if your PHP environment does not allow the creation of sub directories by Smarty. Sub directories are more efficient, so use them if you can.

phpSmarty,...

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

[\\$compile_id\[id\]](#)

[Up](#)

[\[\]\\$default_modifiers](#)

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$default_modifiers

This is an array of modifiers to implicitly apply to every variable in a template. For example, to HTML-escape every variable by default, use array('escape:"htmlall"'); To make a variable exempt from default modifiers, pass the special "smarty" modifier with a parameter value of "nodefaults" modifier to it, such as {\$var|smarty:nodefaults}.
..:HTML,(‘escape:”htmlall”’);,”nodefaults””smarty”,:
{\$var|smarty:nodefaults}.

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

[\\$use_sub_dirs\[\]](#)

[Up](#)

[
]\$default_resource_type

Smarty - the compiling PHP template engine

Chapter 12. Variables

[Prev](#)

[PHP](#)

[Next](#)

\$default_resource_type

This tells smarty what resource type to use implicitly. The default value is 'file', meaning that \$smarty->display('index.tpl'); and \$smarty->display('file:index.tpl'); are identical in meaning. See the [resource](#) chapter for details.

smarty.'file',,\$smarty->display('index.tpl');and
>display('file:index.tpl');.

[Home](#)
[Up](#)

[Methods](#)

[resource](#)

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

Chapter 13. Methods 13.

Table of Contents

[append](#)
[append_by_ref](#)
[assign](#)
[assign_by_ref](#)
[clear_all_assign](#)
[clear_all_cache](#)
[clear_assign](#)
[clear_cache](#)
[clear_compiled_tpl](#)
[clear_config](#)
[config_load](#)
[display](#)
[fetch](#)
[get_config_vars](#)
[get_registered_object](#)
[get_template_vars](#)
[is_cached](#)
[load_filter](#)
[register_block](#)
[register_compiler_function](#)
[register_function](#)
[register_modifier](#)
[register_object](#)
[register_outputfilter](#)
[register_postfilter](#)
[register_prefilter](#)
[register_resource](#)
[trigger_error](#)
[template_exists](#)
[unregister_block](#)
[unregister_compiler_function](#)
[unregister_function](#)
[unregister_modifier](#)
[unregister_object](#)

[unregister_outputfilter](#)
[unregister_postfilter](#)
[unregister_prefilter](#)
[unregister_resource](#)

append

```
void append (mixed var)
void append (string varname, mixed var)
void append (string varname, mixed var, boolean merge)
```

This is used to append an element to an assigned array. If you append to a string value, it is converted to an array value and then appended to. You can explicitly pass name/value pairs, or associative arrays containing the name/value pairs. If you pass the optional third parameter of true, the value will be merged with the current array instead of appended.

, “=>” TRUE

Technical Note: The merge parameter respects array keys, so if you merge two numerically indexed arrays, they may overwrite each other or result in non-sequential keys. This is unlike the array_merge() function of PHP which wipes out numerical keys and renames them.

“merge” PHPArray_merge()

Example 13-1. append 13-1.

```
// passing name/value pairs
$smarty->append("Name", "Fred");
$smarty->append("Address", $address);

// passing an associative array
$smarty->append(array("city" => "Lincoln", "state" => "
```

[Prev](#) [PHP](#)

\$default_resource_type

[Home](#)

[Up](#)

[Next](#)

append_by_ref

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

append_by_ref

```
void append_by_ref (string varname, mixed var)
```

```
void append_by_ref (string varname, mixed var, boolean merge)
```

This is used to append values to the templates by reference. If you append a variable by reference then change its value, the appended value sees the change as well. For objects, append_by_ref() also avoids an in-memory copy of the appended object. See the PHP manual on variable referencing for an in-depth explanation. If you pass the optional third parameter of true, the value will be merged with the current array instead of appended.

append_by_ref()PHP true

Technical Note: The merge parameter respects array keys, so if you merge two numerically indexed arrays, they may overwrite each other or result in non-sequential keys. This is unlike the array_merge() function of PHP which wipes out numerical keys and renames them.

“merge” PHParray_merge()

Example 13-2. append_by_ref

13-2.

```
// appending name/value pairs
$smarty->append_by_ref("Name",$myname);
$smarty->append_by_ref("Address",$address);
```

[Prev](#) [PHP](#)

Methods

[Home](#)

[Up](#)

[Next](#)

assign

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

assign

```
void assign (mixed var)
void assign (string varname, mixed var)
```

This is used to assign values to the templates. You can explicitly pass name/value pairs, or associative arrays containing the name/value pairs.

/ /

Example 13-3. assign 13-3.

```
// passing name/value pairs /
$smarty->assign("Name", "Fred");
$smarty->assign("Address", $address);

// passing an associative array
$smarty->assign(array("city" => "Lincoln", "state" => "
```

[Prev](#) [PHP](#)
[append_by_ref](#)

[Home](#)
[Up](#)

[Next](#)
[assign_by_ref](#)

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

assign_by_ref

```
void assign_by_ref (string varname, mixed var)
```

This is used to assign values to the templates by reference instead of making a copy. See the PHP manual on variable referencing for an explanation.

PHP

Technical Note: This is used to assign values to the templates by reference. If you assign a variable by reference then change its value, the assigned value sees the change as well. For objects, assign_by_ref() also avoids an in-memory copy of the assigned object. See the PHP manual on variable referencing for an in-depth explanation.

[assign_by_ref\(\)](#) PHP

Example 13-4. assign_by_ref **13-4.**

```
// passing name/value pairs
$smarty->assign_by_ref("Name",$myname);
$smarty->assign_by_ref("Address",$address);
```

[Prev](#) [PHP](#)

assign

[Home](#)

[Up](#)

[Next](#)

clear_all_assign

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

clear_all_assign

```
void clear_all_assign ()
```

This clears the values of all assigned variables.

Example 13-5. clear_all_assign

13-5.

```
// clear all assigned variables  
$smarty->clear_all_assign();
```

[Prev](#) [PHP](#)
[assign_by_ref](#)

[Home](#)
[Up](#)

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

clear_all_cache

```
void clear_all_cache (int expire time)
```

This clears the entire template cache. As an optional parameter, you can supply a minimum age in seconds the cache files must be before they will get cleared.

“expire time”

Example 13-6. clear_all_cache 13-6.

```
// clear the entire cache
$smarty->clear_all_cache();
```

[Prev](#) [PHP](#)

clear_all_assign

[Home](#)

[Up](#)

[Next](#)

clear_assign

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

clear_assign

```
void clear_assign (string var)
```

This clears the value of an assigned variable. This can be a single value, or an array of values.

Example 13-7. clear_assign

13-7.

```
// clear a single variable  
$smarty->clear_assign("Name");  
  
// clear multiple variables  
$smarty->clear_assign(array("Name", "Address", "Zip"));
```

[Prev](#) [PHP](#)

clear_all_cache

[Home](#)

[Up](#)

[Next](#)

clear_cache

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

clear_cache

```
void clear_cache (string template [, string cache id [, string compile id [, int expire time]]])
```

This clears the cache for a specific template. If you have multiple caches for this template, you can clear a specific cache by supplying the cache id as the second parameter. You can also pass a compile id as a third parameter. You can "group" templates together so they can be removed as a group. See the [caching section](#) for more information. As an optional fourth parameter, you can supply a minimum age in seconds the cache file must be before it will get cleared.

Example 13-8. clear_cache

13-8.

```
// clear the cache for a template
//  
$smarty->clear_cache("index.tpl");  
  
// clear the cache for a particular cache id in an mul
//  
$smarty->clear_cache("index.tpl","CACHEID");
```

[Prev](#) [PHP](#)

clear_assign

[Home](#)

[Up](#)

[Next](#)

clear_compiled_tpl

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

clear_compiled_tpl

```
void clear_compiled_tpl (string tpl_file)
```

This clears the compiled version of the specified template resource, or all compiled template files if one is not specified. This function is for advanced use only, not normally needed.

Fwolf:templates_c

Example 13-9. clear_compiled_tpl 13-9.

```
// clear a specific template resource
//
$smarty->clear_compiled_tpl("index.tpl");

// clear entire compile directory
//
$smarty->clear_compiled_tpl();
```

[Prev](#) [PHP](#)

clear_cache

[Home](#)

[Up](#)

[Next](#)

clear_config

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

clear_config

```
void clear_config ([string var])
```

This clears all assigned config variables. If a variable name is supplied, only that variable is cleared.

Example 13-10. clear_config 13-10.

```
// clear all assigned config variables.  
//  
$smarty->clear_config();  
  
// clear one variable  
//  
$smarty->clear_config('foobar');
```

[Prev](#) [PHP](#)

clear_compiled_tpl

[Home](#)

[Up](#)

[Next](#)

config_load

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

config_load

```
void config_load (string file [, string section])
```

This loads config file data and assigns it to the template. This works identical to the template [config_load](#) function.

[config_load](#)

Technical Note: As of Smarty 2.4.0, assigned template variables are kept across invocations of fetch() and display(). Config vars loaded from config_load() are always global scope. Config files are also compiled for faster execution, and respect the [force_compile](#) and [compile_check](#) settings.

Smarty 2.4.0fetch()
display()
[config_load\(\)](#)
[force_compile](#)
[compile_check](#)

Example 13-11. config_load 13-11.

```
// load config variables and assign them
//
$smarty->config_load('my.conf');

// load a section
//
$smarty->config_load('my.conf', 'foobar');
```

[Prev](#) [PHP](#)
[clear_config](#)

[Home](#)
[Up](#)

[Next](#)
[display](#)

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

display

```
void display (string template [, string cache_id [, string compile_id]])
```

This displays the template. Supply a valid [template resource](#) type and path. As an optional second parameter, you can pass a cache id. See the [caching section](#) for more information.

As an optional third parameter, you can pass a compile id. This is in the event that you want to compile different versions of the same template, such as having separate templates compiled for different languages. Another use for compile_id is when you use more than one \$template_dir but only one \$compile_dir. Set a separate compile_id for each \$template_dir, otherwise templates of the same name will overwrite each other. You can also set the [\\$compile_id](#) variable once instead of passing this to each call to display().

```
$template_dir$compile_dir$template_dir  
display() $compile_id
```

Example 13-12. display 13-12.

```
include("Smarty.class.php");  
$smarty = new Smarty;  
$smarty->caching = true;  
  
// only do db calls if cache doesn't exist  
//  
if (!$smarty->is_cached("index.tpl"))  
{  
  
    // dummy up some data
```

```

$address = "245 N 50th";
$db_data = array(
    "City" => "Lincoln",
    "State" => "Nebraska",
    "Zip" => "68502"
);

$smarty->assign("Name", "Fred");
$smarty->assign("Address", $address);
$smarty->assign($db_data);

}

// display the output
//
$smarty->display("index.tpl");

```

Use the syntax for [template resources](#) to display files outside of the \$template_dir directory.

\$template_dir

Example 13-13. function display template resource examples 13-13.

```

// absolute filepath
//
$smarty->display("/usr/local/include/templates/header.

// absolute filepath (same thing)
//
$smarty->display("file:/usr/local/include/templates/he

```

```
// windows absolute filepath (MUST use "file:" prefix)
// WINDOS"file:"
$smarty->display("file:C:/www/pub/templates/header.tpl

// include from template resource named "db"
// "db"
$smarty->display("db:header.tpl");
```

[Prev](#) [PHP](#)

config_load

[Home](#)

[Up](#)

[Next](#)

fetch

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

fetch

```
string fetch (string template [, string cache_id [, string compile_id]])
```

This returns the template output instead of displaying it. Supply a valid [template resource](#) type and path. As an optional second parameter, you can pass a cache id. See the [caching section](#) for more information.

HTML

As an optional third parameter, you can pass a compile id. This is in the event that you want to compile different versions of the same template, such as having separate templates compiled for different languages. Another use for compile_id is when you use more than one \$template_dir but only one \$compile_dir. Set a separate compile_id for each \$template_dir, otherwise templates of the same name will overwrite each other. You can also set the [\\$compile_id](#) variable once instead of passing this to each call to fetch().

```
$template_dir$compile_dir$template_dir  
display() $compile_id
```

Example 13-14. fetch 13-14.

```
include("Smarty.class.php");  
$smarty = new Smarty;  
  
$smarty->caching = true;  
  
// only do db calls if cache doesn't exist  
//  
if (!$smarty->is_cached("index.tpl"))  
{
```

```
// dummy up some data
$address = "245 N 50th";
$db_data = array(
    "City" => "Lincoln",
    "State" => "Nebraska",
    "Zip" => "68502"
);

$smarty->assign("Name", "Fred");
$smarty->assign("Address", $address);
$smarty->assign($db_data);

}

// capture the output
//
$output = $smarty->fetch("index.tpl");

// do something with $output here
//

echo $output;
```

[Prev](#) [PHP](#)

display

[Home](#)

[Up](#)

[Next](#)

[get_config_vars](#)

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

get_config_vars

array **get_config_vars** ([string varname])

This returns the given loaded config variable value. If no parameter is given, an array of all loaded config variables is returned.

Example 13-15. get_config_vars 13-15.

```
// get loaded config template var 'foo'  
// "foo"  
$foo = $smarty->get_config_vars('foo');  
  
// get all loaded config template vars  
//  
$config_vars = $smarty->get_config_vars();  
  
// take a look at them  
//  
print_r($config_vars);
```

[Prev](#) [PHP](#)

fetch

[Home](#)

[Up](#)

[Next](#)

get_registered_object

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

get_registered_object

array **get_registered_object** (string object_name)

This returns a reference to a registered object. This is useful from within a custom function when you need direct access to a registered object.

Example 13-16. get_registered_object 13-16.

```
function smarty_block_foo($params, &$smarty) {
    if (isset[$params['object']]) {
        // get reference to registered object
        //
        $obj_ref =& $smarty->&get_registered_o
        // use $obj_ref is now a reference to
        // $obj_ref
    }
}
```

[Prev](#) [PHP](#)

get_config_vars

[Home](#)

[Up](#)

[Next](#)

get_template_vars

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

get_template_vars

array **get_template_vars** ([string varname])

This returns the given assigned variable value. If no parameter is given, an array of all assigned variables is returned.

Example 13-17. get_template_vars 13-17.

```
// get assigned template var 'foo'  
$foo = $smarty->get_template_vars('foo');
```



```
// get all assigned template vars  
$tpl_vars = $smarty->get_template_vars();
```



```
// take a look at them  
print_r($tpl_vars);
```

[Prev](#) [PHP](#)

[get_registered_object](#)

[Home](#)

[Up](#)

[Next](#)

[is_cached](#)

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

is_cached

```
void is_cached (string template, [string cache_id])
```

This returns true if there is a valid cache for this template. This only works if [caching](#) is set to true.

Example 13-18. is_cached 13-18.

```
$smarty->caching = true;  
  
if( !$smarty->is_cached("index.tpl")) {  
    // do database calls, assign vars here  
    //  
}  
  
$smarty->display("index.tpl");
```

You can also pass a cache id as an optional second parameter in case you want multiple caches for the given template.

Example 13-19. is_cached with multiple-cache template 13-19.

```
$smarty->caching = true;  
  
if( !$smarty->is_cached("index.tpl", "FrontPage")) {  
    // do database calls, assign vars here
```

```
}
```

```
$smarty->display("index.tpl","FrontPage");
```

[Prev](#) [PHP](#)

[get_template_vars](#)

[Home](#)

[Up](#)

[Next](#)

[load_filter](#)

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

load_filter

```
void load_filter (string type, string name)
```

This function can be used to load a filter plugin. The first argument specifies the type of the filter to load and can be one of the following: 'pre', 'post', or 'output'. The second argument specifies the name of the filter plugin, for example, 'trim'.

“pre”“post”“output” “trim”

Example 13-20. loading filter plugins

13-20.

```
$smarty->load_filter('pre', 'trim'); // load prefilter  
$smarty->load_filter('pre', 'datefooter'); // load and  
$smarty->load_filter('output', 'compress'); // load ou
```

[Prev](#) [PHP](#)

is_cached

[Home](#)

[Up](#)

[Next](#)

register_block

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

register_block

```
void register_block (string name, mixed impl, bool cacheable, array or  
null cache_attrs)
```

Use this to dynamically register block functions plugins. Pass in the block function name, followed by the PHP function callback that implements it.

/

The php-function callback *impl* can be either (a) a string containing the function name or (b) an array of the form `array(&$object, $method)` with `&$object` being a reference to an object and `$method` being a string containing the method-name or (c) an array of the form `array(&$class, $method)` with `$class` being a classname and `$method` being a class method of that class.

```
array(&$object, $method)&$object$methodarray(&$ class,  
$method)$class$method
```

`$cacheable` and `$cache_attrs` can be omitted in most cases. See [Controlling Cacheability of Plugins' Output](#) on how to use them properly.

`$cacheable` `$cache_attrs` .

Example 13-21. register_block 13-21.

```
/* PHP */  
$smarty->register_block("translate", "do_translation")  
  
function do_translation ($params, $content, &$smarty,  
if (isset($content)) {  
$lang = $params['lang'];  
// do some translation with $content  
return $translation;  
}
```

```
}
```

```
{* template *}
{translate lang="br"}
Hello, world!
{/translate}
```

[Prev](#) [PHP](#)

load_filter

[Home](#)

[Up](#)

[Next](#)

register_compiler_function

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

register_compiler_function

```
void register_compiler_function (string name, mixed impl, bool  
cacheable)
```

Use this to dynamically register a compiler function plugin. Pass in the compiler function name, followed by the PHP function that implements it.

The php-function callback *impl* can be either (a) a string containing the function name or (b) an array of the form `array(&$object, $method)` with `&$object` being a reference to an object and `$method` being a string containing the method-name or (c) an array of the form `array(&$class, $method)` with `$class` being a classname and `$method` being a class method of that class.

`array(&$object, $method)&$object$method`
`array(&$class, $method)&$class$method`

`$cacheable` can be omitted in most cases. See [Controlling Cacheability of Plugins' Output](#) on how to do it properly.

`$cacheable`

[Prev](#) [PHP](#)

register_block

[Home](#)

[Up](#)

[Next](#)

register_function

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

register_function

```
void register_function (string name, mixed impl, bool cacheable, array  
or null cache_attrs)
```

Use this to dynamically register template function plugins. Pass in the template function name, followed by the PHP function name that implements it.

The php-function callback *impl* can be either (a) a string containing the function name or (b) an array of the form `array(&$object, $method)` with `&$object` being a reference to an object and `$method` being a string containing the method-name or (c) an array of the form `array(&$class, $method)` with `$class` being a classname and `$method` being a class method of that class.

```
array(&$object, $method)&$object$methodarray(&$ class,  
$method)$class$method
```

`$cacheable` and `$cache_attrs` can be omitted in most cases. See [Controlling Cacheability of Plugins' Output](#) on how to use them properly.

`$cacheable` `$cache_attrs` -

Example 13-22. register_function 13-22.

```
$smarty->register_function("date_now", "print_current_  
  
function print_current_date ($params) {  
    extract($params);  
    if(empty($format))  
        $format="%b %e, %Y";  
    return strftime($format,time());  
}
```

```
// now you can use this in Smarty to print the current  
// or, {date_now format="%Y/%m/%d"} to format it.
```

```
// {date_now}  
// {date_now format="%Y/%m/%d"}
```

[Prev](#) [PHP](#)

register_compiler_function

[Home](#)

[Up](#)

[Next](#)

register_modifier

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

register_modifier

```
void register_modifier (string name, mixed impl)
```

Use this to dynamically register modifier plugin. Pass in the template modifier name, followed by the PHP function that it implements it.

The php-function callback *impl* can be either (a) a string containing the function name or (b) an array of the form `array(&$object, $method)` with `&$object` being a reference to an object and `$method` being a string containing the method-name or (c) an array of the form `array(&$class, $method)` with `$class` being a classname and `$method` being a class method of that class.

```
array(&$object, $method)&$object$methodarray(&$ class,  
$method)$class$method
```

Example 13-23. register_modifier 13-23.

```
// let's map PHP's stripslashes function to a Smarty modifier  
// PHPstripslashesSmarty  
  
$smarty->register_modifier("sslash","stripslashes");  
  
// now you can use {$var|sslash} to strip slashes from  
// {$var|sslash}
```

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

register_object

void **register_object** (string object_name, object \$object, array allowed methods/properties, boolean format, array block methods)

This is to register an object for use in the templates. See the [object section](#) of the manual for examples.

[Prev](#) [PHP](#)

[register_modifier](#)

[Home](#)

[Up](#)

[Next](#)

[register_outputfilter](#)

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

register_outputfilter

void **register_outputfilter** (mixed function)

Use this to dynamically register outputfilters to operate on a template's output before it is displayed. See [template output filters](#) for more information on how to set up an output filter function.

The php-function callback *function* can be either (a) a string containing the function name or (b) an array of the form `array(&$object, $method)` with `&$object` being a reference to an object and `$method` being a string containing the method-name or (c) an array of the form `array(&$class, $method)` with `$class` being a classname and `$method` being a class method of that class.

`array(&$object, $method)&$object$methodarray(&$ class,
$method)$class$method`

[Prev](#) [PHP](#)

register_object

[Home](#)

[Up](#)

[Next](#)

register_postfilter

Smarty - the compiling PHP template engine

Chapter

13.

Methods

[Prev](#)

[PHP](#)

[Next](#)

register_postfilter

```
void register_postfilter (mixed function)
```

Use this to dynamically register postfilters to run templates through after they are compiled. See [template postfilters](#) for more information on how to setup a postfiltering function.

The php-function callback *function* can be either (a) a string containing the function name or (b) an array of the form `array(&$object, $method)` with `&$object` being a reference to an object and `$method` being a string containing the method-name or (c) an array of the form `array(&$class, $method)` with `$class` being a classname and `$method` being a class method of that class.

```
array(&$object, $method)&$object$methodarray(&$ class,  
$method)$class$method
```

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

register_outputfilter

[Up](#)

register_prefilter

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

register_prefilter

```
void register_prefilter (mixed function)
```

Use this to dynamically register prefilters to run templates through before they are compiled. See [template prefilters](#) for more information on how to setup a prefiltering function.

The php-function callback *function* can be either (a) a string containing the function name or (b) an array of the form `array(&$object, $method)` with `&$object` being a reference to an object and `$method` being a string containing the method-name or (c) an array of the form `array(&$class, $method)` with `$class` being a classname and `$method` being a class method of that class.

```
array(&$object, $method)&$object$methodarray(&$ class,  
$method)$class$method
```

[Prev](#) [PHP](#)

register_postfilter

[Home](#)

[Up](#)

[Next](#)

register_resource

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

register_resource

```
void register_resource (string name, array resource_funcs)
```

Use this to dynamically register a resource plugin with Smarty. Pass in the name of the resource and the array of PHP functions implementing it. See [template resources](#) for more information on how to setup a function for fetching templates.

Technical Note: A resource name must be at least two characters in length. One character resource names will be ignored and used as part of the file path, such as \$smarty->display('c:/path/to/index.tpl');

```
$smarty->display('c:/path/to/index.tpl');
```

The php-function-array *resource_funcs* must have 4 or 5 elements. With 4 elements the elements are the functions-callbacks for the respective "source", "timestamp", "secure" and "trusted" functions of the resource. With 5 elements the first element has to be an object reference or a class name of the object or class implementing the resource and the 4 following elements have to be the method names implementing "source", "timestamp", "secure" and "trusted".

```
454“source ”“timestamp ”“secure ”“trusted ”  
54“source ”“timestamp ”“secure ”“trusted  
”
```

Example 13-24. register_resource 13-24.

```
$smarty->register_resource("db", array("db_get_template",  
"db_get_timestamp",  
"db_get_secure",  
"db_get_trusted"));
```

[Prev](#) [PHP](#)

register_prefilter

[Home](#)

[Up](#)

[Next](#)

trigger_error

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

trigger_error

```
void trigger_error (string error_msg, [int level])
```

This function can be used to output an error message using Smarty.
level parameter can be one of the values used for trigger_error() PHP
function, i.e. E_USER_NOTICE, E_USER_WARNING, etc. By default it's
E_USER_WARNING.

Smarty *level* PHPtrigger_error()
E_USER_NOTICE,
E_USER_WARNING
E_USER_WARNING

[Prev](#) [PHP](#)

register_resource

[Home](#)

[Up](#)

[Next](#)

template_exists

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

template_exists

bool **template_exists** (string template)

This function checks whether the specified template exists. It can accept either a path to the template on the filesystem or a resource string specifying the template.

template

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

[trigger_error](#)

[Up](#)

[unregister_block](#)

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

unregister_block

```
void unregister_block (string name)
```

Use this to dynamically unregister block function plugin. Pass in the block function name.

[Prev](#) [PHP](#)

template_exists

[Home](#)

[Up](#)

[Next](#)

unregister_compiler_function

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

unregister_compiler_function

```
void unregister_compiler_function (string name)
```

Use this to dynamically unregister a compiler function. Pass in the name of the compiler function.

name

[Prev](#) [PHP](#)

[unregister_block](#)

[Home](#)

[Up](#)

[Next](#)

[unregister_function](#)

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

unregister_function

```
void unregister_function (string name)
```

Use this to dynamically unregister template function plugin. Pass in the template function name.

Example 13-25. unregister_function 13-35.

```
// we don't want template designers to have access to
//
$smarty->unregister_function("fetch");
```

[Prev](#) [PHP](#)

unregister_compiler_function

[Home](#)

[Up](#)

[Next](#)

unregister_modifier

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

unregister_modifier

```
void unregister_modifier (string name)
```

Use this to dynamically unregister modifier plugin. Pass in the template modifier name.

Example 13-26. unregister_modifier 13-26.

```
// we don't want template designers to strip tags from
//  
  
$smarty->unregister_modifier("strip_tags");
```

[Prev](#) [PHP](#)

unregister_function

[Home](#)

[Up](#)

[Next](#)

unregister_object

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

unregister_object

`void unregister_object (string object_name)`

Use this to unregister an object.

[Prev](#) [PHP](#)

unregister_modifier

[Home](#)

[Up](#)

[Next](#)

unregister_outputfilter

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

unregister_outputfilter

`void unregister_outputfilter (string function_name)`

Use this to dynamically unregister an output filter.

[Prev](#) [PHP](#)

[unregister_object](#)

[Home](#)

[Up](#)

[Next](#)

[unregister_postfilter](#)

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

unregister_postfilter

```
void unregister_postfilter (string function_name)
```

Use this to dynamically unregister a postfilter.

[Prev](#) [PHP](#)

[unregister_outputfilter](#)

[Home](#)

[Up](#)

[Next](#)

[unregister_prefilter](#)

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

unregister_prefilter

`void unregister_prefilter (string function_name)`

Use this to dynamically unregister a prefilter.

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

[unregister_postfilter](#)

[Up](#)

[unregister_resource](#)

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 13. Methods

[Next](#)

unregister_resource

```
void unregister_resource (string name)
```

Use this to dynamically unregister a resource plugin. Pass in the name of the resource.

Example 13-27. unregister_resource 13-27.

```
$smarty->unregister_resource("db");
```

[Prev](#) [PHP](#)

unregister_prefilter

[Home](#)

[Up](#)

[Next](#)

Caching

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

Chapter 14. Caching []

Table of Contents []

[Setting Up Caching \[\]](#)

[Multiple Caches Per Page \[\]](#)

[Cache Groups \[\]](#)

[Controlling Cacheability of Plugins' Output \[\]](#)

Caching is used to speed up a call to [display\(\)](#) or [fetch\(\)](#) by saving its output to a file. If a cached version of the call is available, that is displayed instead of regenerating the output. Caching can speed things up tremendously, especially templates with longer computation times. Since the output of [display\(\)](#) or [fetch\(\)](#) is cached, one cache file could conceivably be made up of several template files, config files, etc.

[display\(\)](#)[fetch\(\)](#)[display\(\)](#)[fetch\(\)](#)

Since templates are dynamic, it is important to be careful what you are caching and for how long. For instance, if you are displaying the front page of your website that does not change its content very often, it might work well to cache this page for an hour or more. On the other hand, if you are displaying a page with a weather map containing new information by the minute, it would not make sense to cache this page.

Setting Up Caching []

The first thing to do is enable caching. This is done by setting [\\$caching = true](#) (or 1.)

```
$caching = true( 1.)
```

Example 14-1. enabling caching 14-1.

```
require('Smarty.class.php');
$smarty = new Smarty;

$smarty->caching = true;

$smarty->display('index.tpl');
```

With caching enabled, the function call to `display('index.tpl')` will render the template as usual, but also saves a copy of its output to a file (a cached copy) in the [\\$cache_dir](#). Upon the next call to `display('index.tpl')`, the cached copy will be used instead of rendering the template again.
`display('index.tpl')`
`copyn.$cache_dir.display('index.tpl')`
`copyn.`

Technical Note: The files in the `$cache_dir` are named similar to the template name. Although they end in the ".php" extention, they are not really executable php scripts. Do not edit these files!

```
$chche_dir.phpphp
```

Each cached page has a limited lifetime determined by [\\$cache_lifetime](#). The default value is 3600 seconds, or 1 hour. After that time expires, the cache is regenerated. It is possible to give individual caches their own expiration time by setting `$caching = 2`. See the documentation on [\\$cache_lifetime](#) for details.

```
\$cache\_lifetime3600$caching=2
```

Example 14-2. setting cache_lifetime per cache

14-2

```
require('Smarty.class.php');
$smarty = new Smarty;

$smarty->caching = 2; // lifetime is per cache

// set the cache_lifetime for index.tpl to 5 minutes
$smarty->cache_lifetime = 300;
$smarty->display('index.tpl');

// set the cache_lifetime for home.tpl to 1 hour
$smarty->cache_lifetime = 3600;
$smarty->display('home.tpl');

// NOTE: the following $cache_lifetime setting will no
// $chching=2$chche_lifetime
// The cache lifetime for home.tpl has already been se
// to 1 hour, and will no longer respect the value of
// home.tpl1$cache_lifetime
// The home.tpl cache will still expire after 1 hour.
// home.tpl1$smarty->cache_lifetime = 30; // 30 seconds
$smarty->display('home.tpl');
```

If [\\$compile_check](#) is enabled, every template file and config file that is involved with the cache file is checked for modification. If any of the files have been modified since the cache was generated, the cache is immediately regenerated. This is a slight overhead so for optimum performance, leave [\\$compile_check](#) set to false.

[\\$compile_check](#)\$compile_checkfalse:D

Example 14-3. enabling [\\$compile_check](#)

14-3.\$compile_check

```
require('Smarty.class.php');
$smarty = new Smarty;
```

```
$smarty->caching = true;  
$smarty->compile_check = true;  
  
$smarty->display('index.tpl');
```

If [\\$force_compile](#) is enabled, the cache files will always be regenerated. This effectively turns off caching. [\\$force_compile](#) is usually for debugging purposes only, a more efficient way of disabling caching is to set [\\$caching](#) = false (or 0.)

[\\$force_compile](#) \$force_compile [\\$caching](#) = false(0.)

The [is_cached\(\)](#) function can be used to test if a template has a valid cache or not. If you have a cached template that requires something like a database fetch, you can use this to skip that process.

[is_cached\(\)](#)

Example 14-4. using is_cached() 14-4.is_cached()

```
require('Smarty.class.php');  
$smarty = new Smarty;  
  
$smarty->caching = true;  
  
if (!$smarty->is_cached('index.tpl')) {  
    // No cache available, do variable assignments  
    $contents = get_database_contents();  
    $smarty->assign($contents);  
}  
  
$smarty->display('index.tpl');
```

You can keep parts of a page dynamic with the [insert](#) template function. Let's say the whole page can be cached except for a banner that is displayed down the right side of the page. By using an insert function for the banner, you can keep this element dynamic within the cached content. See the documentation on [insert](#) for details and examples.

[insert](#) [insert](#)

You can clear all the cache files with the [clear_all_cache\(\)](#) function, or individual cache files (or groups) with the [clear_cache\(\)](#) function.

[clear_all_cache\(\)](#) [clear_cache\(\)](#)

Example 14-5. clearing the cache 14-5.

```
require('Smarty.class.php');
$smarty = new Smarty;

$smarty->caching = true;

// clear out all cache files
$smarty->clear_all_cache();

// clear only cache for index.tpl
$smarty->clear_cache('index.tpl');

$smarty->display('index.tpl');
```

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

[unregister_resource](#)

[Up](#)

Multiple Caches Per
Page

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 14. Caching

[Next](#)

Multiple Caches Per Page

You can have multiple cache files for a single call to display() or fetch(). Let's say that a call to display('index.tpl') may have several different output contents depending on some condition, and you want separate caches for each one. You can do this by passing a cache_id as the second parameter to the function call.

display()fetch()display('index.tpl')cache_id

Example 14-6. passing a cache_id to display()

14-6.display()cache_id

```
require('Smarty.class.php');
$smarty = new Smarty;

$smarty->caching = true;

$my_cache_id = $_GET['article_id'];

$smarty->display('index.tpl', $my_cache_id);
```

Above, we are passing the variable \$my_cache_id to display() as the cache_id. For each unique value of \$my_cache_id, a separate cache will be generated for index.tpl. In this example, "article_id" was passed in the URL and is used as the cache_id.

\$my_cache_idcache_iddisplay()index.tpl\$my_cache_id
"article_id"URLcache_id

Technical Note: Be very cautious when passing values from a client (web browser) into Smarty (or any PHP application.) Although the above example of using the article_id from the URL looks handy, it could have bad consequences. The cache_id is used to create a directory on the file system, so if the user decided to pass an extremely large value for article_id, or write a script that sends random article_ids at a rapid pace, this could possibly cause problems at the server level. Be sure to sanitize any data passed in before using it. In this instance, maybe you know the article_id has a

length of 10 characters and is made up of alpha-numerics only, and must be a valid article_id in the database. Check for this!

(web)Smarty(PHP)article_idURL[]cache_id
article_idarticle_idsarticle_id()10-
article_idCheck for this!!

Be sure to pass the same cache_id as the second parameter to [is_cached\(\)](#) and [clear_cache\(\)](#).

[is_cached\(\)](#)[clear_cache\(\)](#)cache_id

Example 14-7. passing a cache_id to is_cached()

14-7.is_cached()cache_id

```
require('Smarty.class.php');
$smarty = new Smarty;

$smarty->caching = true;

$my_cache_id = $_GET['article_id'];

if(!$smarty->is_cached('index.tpl',$my_cache_id)) {
    // No cache available, do variable assignments
    $contents = get_database_contents();
    $smarty->assign($contents);
}

$smarty->display('index.tpl',$my_cache_id);
```

You can clear all caches for a particular cache_id by passing null as the first parameter to [clear_cache\(\)](#).

[clear_cache\(\)](#)nullcache_id

Example 14-8. clearing all caches for a particular cache_id

14-8.cache_id

```
require('Smarty.class.php');
$smarty = new Smarty;
```

```
$smarty->caching = true;  
  
// clear all caches with "sports" as the cache_id  
$smarty->clear_cache(null,"sports");  
  
$smarty->display('index.tpl',"sports");
```

In this manner, you can "group" your caches together by giving them the same `cache_id`.

`cache_id`

[Prev](#) [PHP](#)

Caching

[Home](#)

[Up](#)

[Next](#)

Cache Groups

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 14. Caching

[Next](#)

Cache Groups []

You can do more elaborate grouping by setting up cache_id groups. This is accomplished by separating each sub-group with a vertical bar "|" in the cache_id value. You can have as many sub-groups as you like.
cache_idcache_id|"|"

Example 14-9. cache_id groups

14-9.cache_id

```
require('Smarty.class.php');
$smarty = new Smarty;

$smarty->caching = true;

// clear all caches with "sports|basketball" as the first cache_id
$smarty->clear_cache(null,"sports|basketball");

// clear all caches with "sports" as the first cache_id
// include "sports|basketball", or "sports|(anything)"
$smarty->clear_cache(null,"sports");

$smarty->display('index.tpl',"sports|basketball");
```

Technical Note: The cache grouping does NOT use the path to the template as any part of the cache_id. For example, if you have display('themes/blue/index.tpl'), you cannot clear the cache for everything under the "themes/blue" directory. If you want to do that, you must group them in the cache_id, such as
display('themes/blue/index.tpl','themes|blue'); Then you can clear the caches for the blue theme with clear_cache(null,'themes|blue');
:cache_iddisplay('themes/blue/index.tpl'),"themes/blue"
cache_iddisplay('themes/blue/index.tpl','themes|blue')
clear_cache(null,'themes|blue')blue theme(...)

Multiple Caches Per
Page



[Up](#)

Controlling
Cacheability of Plugins'
Output



Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 14. Caching

[Next](#)

Controlling Cacheability of Plugins' Output

Since Smarty-2.6.0 plugins the cacheability of plugins can be declared when registering them. The third parameter to register_block, register_compiler_function and register_function is called `$cacheable` and defaults to true which is also the behaviour of plugins in Smarty versions before 2.6.0

Smarty-2.6.0
register_block
register_compiler_function
register_function
3\$ `cacheable` , true
2.6.0

When registering a plugin with `$cacheable=false` the plugin is called everytime the page is displayed, even if the page comes from the cache. The plugin function behaves a little like an [insert](#) function.

`$cacheable=false` [insert](#)

In contrast to [{insert}](#) the attributes to the plugins are not cached by default. They can be declared to be cached with the fourth parameter `$cache_attrs`. `$cache_attrs` is an array of attribute-names that should be cached, so the plugin-function get value as it was the time the page was written to cache everytime it is fetched from the cache.

[{insert}](#) `$cache_attrs` `$cache_attrs` -----

Example 14-10. Preventing a plugin's output from being cached

14-10.

index.php:

```
require('Smarty.class.php');
$smarty = new Smarty;
$smarty->caching = true;

function remaining_seconds($params, &$smarty) {
    $remain = $params['endtime'] - time();
    if ($remain >=0)
```

```

    return $remain . " second(s)";
else
    return "done";
}

$smarty->register_function('remaining', 'remaining_sec');

if (!$smarty->is_cached('index.tpl')) {
    // fetch $obj from db and assign...
    $smarty->assign_by_ref('obj', $obj);
}

$smarty->display('index.tpl');

index.tpl:

Time Remaining: {remain endtime=$obj->endtime}

```

The number of seconds till the endtime of \$obj is reached changes on each request. Since the endtime attribute is cached the object only has to be updated once when it is written to the cache but not on subsequent requests of the page.

\$obj

Example 14-11. Preventing a whole passage of a template from being cached

```

index.php:

require('Smarty.class.php');
$smarty = new Smarty;
$smarty->caching = true;

function smarty_block_dynamic($param, $content, &$smarty) {
    return $content;
}

```

```
}

$smarty->register_block( 'dynamic', 'smarty_block_dynamic' );
$smarty->display( 'index.tpl' );

index.tpl:

Page created: {"0" | date_format:"%D %H:%M:%S"}

{dynamic}

Now is: {"0" | date_format:"%D %H:%M:%S"}

... do other stuff ...

{/dynamic}
```

When reloading the page you will notice that both dates differ. One is "dynamic" one is "static". You can do everything between {dynamic}...{/dynamic} and be sure it will not be cached like the rest of the page.

""""{dynamic}...{/dynamic}

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Cache Groups

[Up](#)

Advanced Features



Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

Chapter 15. Advanced Features

Table of Contents

- [Objects](#)
- [Prefilters](#)
- [Postfilters](#)
- [Output Filters](#)
- [Cache Handler Function](#)
- [Resources](#)

Objects

Smarty allows access to PHP objects through the templates. There are two ways to access them. One way is to register objects to the template, then use access them via syntax similar to custom functions. The other way is to assign objects to the templates and access them much like any other assigned variable. The first method has a much nicer template syntax. It is also more secure, as a registered object can be restricted to certain methods or properties. However, a registered object cannot be looped over or assigned in arrays of objects, etc. The method you choose will be determined by your needs, but use the first method whenever possible to keep template syntax to a minimum.

SMARTYPHP,

If security is enabled, no private methods or functions can be accessed (beginning with `_`). If a method and property of the same name exist, the method will be used.

" "
—

You can restrict the methods and properties that can be accessed by listing them in an array as the third registration parameter.

By default, parameters passed to objects through the templates are passed the same way custom functions get them. An associative array is passed as the first parameter, and the smarty object as the second. If you want the parameters passed one at a time for each argument like traditional object parameter passing, set the fourth registration parameter to false.

SMARTY

Example 15-1. using a registered or assigned object

```
<?php
```

```

// the object

class My_Object() {
    function meth1($params, &$smarty_obj) {
        return "this is my meth1";
    }
}

$smarty = new My_Object;
// registering the object (will be by reference)
$smarty->register_object("foobar",$myobj);
// if we want to restrict access to certain methods or
$smarty->register_object("foobar",$myobj,array('meth1'
// if you want to use the traditional object parameter
$smarty->register_object("foobar",$myobj,null,false);

// We can also assign objects. Assign by ref when poss
$smarty->assign_by_ref("myobj", $myobj);

$smarty->display("index.tpl");
?>

```

TEMPLATE:

```

{* access our registered object *}
{foobar->meth1 p1="foo" p2=$bar}

{* you can also assign the output *}
{foobar->meth1 p1="foo" p2=$bar assign="output"}
the output was {$output}

{* access our assigned object *}
{$myobj->meth1("foo",$bar)}

```

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 15. Advanced Features

[Next](#)

Prefilters

Template prefilters are PHP functions that your templates are ran through before they are compiled. This is good for preprocessing your templates to remove unwanted comments, keeping an eye on what people are putting in their templates, etc. Prefilters can be either [registered](#) or loaded from the plugins directory by using [load_filter\(\)](#) function or by setting [\\$autoload_filters](#) variable. Smarty will pass the template source code as the first argument, and expect the function to return the resulting template source code.

PHP
SMARTY

Example 15-2. using a template prefilter

```
<?php
// put this in your application
function remove_dw_comments($tpl_source, &$smarty)
{
    return preg_replace("/<!--#.*-->/U","", $tpl_source);
}

// register the prefilter
$smarty->register_prefilter("remove_dw_comments");
$smarty->display("index.tpl");
?>

{* Smarty template index.tpl *}
<!--# this line will get removed by the prefilter -->
```

[Prev](#)

Advanced Features

[Home](#)

[Up](#)

[Next](#)

Postfilters

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 15. Advanced Features

[Next](#)

Postfilters

Template postfilters are PHP functions that your templates are ran through after they are compiled. Postfilters can be either [registered](#) or loaded from the plugins directory by using [load_filter\(\)](#) function or by setting [\\$autoload_filters](#) variable. Smarty will pass the compiled template code as the first argument, and expect the function to return the result of the processing.

PHP [load filter\(\)](#) [\\$autoload filters](#) SMARTY

Example 15-3. using a template postfilter

```
<?php
// put this in your application
function add_header_comment($tpl_source, &$smarty)
{
    return "<?php echo \"<!-- Created by Smarty! -->\n\""
}

// register the postfilter
$smarty->register_postfilter("add_header_comment");
$smarty->display("index.tpl");
?>

{* compiled Smarty template index.tpl *}
<!-- Created by Smarty! -->
{* rest of template content... *}
```

[Prev](#)[Prefilters](#)[Home](#)[Up](#)[Next](#)[Output Filters](#)

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 15. Advanced Features

[Next](#)

Output Filters

When the template is invoked via `display()` or `fetch()`, its output can be sent through one or more output filters. This differs from postfilters because postfilters operate on compiled templates before they are saved to the disk, and output filters operate on the template output when it is executed.

`display()` `fetch()`

Output filters can be either [registered](#) or loaded from the plugins directory by using [`load_filter\(\)`](#) function or by setting [`\$autoload_filters`](#) variable. Smarty will pass the template output as the first argument, and expect the function to return the result of the processing.

[`load filter\(\)`](#) [`\$autoload filters`](#) SMARTY

Example 15-4. using a template outputfilter

```
<?php
// put this in your application
function protect_email($tpl_output, &$smarty)
{
    $tpl_output =
        preg_replace('!(\S+)(@[a-zA-Z0-9\.\-]+\.([a-zA-Z]{2,3})$1%40$2', $tpl_output);
    return $tpl_output;
}

// register the outputfilter
$smarty->register_outputfilter("protect_email");
$smarty->display("index.tpl");

// now any occurrence of an email address in the templ
// a simple protection against spambots
?>
```

[Prev](#)[Home](#)[Next](#)

Postfilters

[Up](#)

Cache Handler
Function

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 15. Advanced Features

[Next](#)

Cache Handler Function

As an alternative to using the default file-based caching mechanism, you can specify a custom cache handling function that will be used to read, write and clear cached files.

Create a function in your application that Smarty will use as a cache handler. Set the name of it in the [\\$cache_handler_func](#) class variable. Smarty will now use this to handle cached data. The first argument is the action, which will be one of 'read', 'write' and 'clear'. The second parameter is the Smarty object. The third parameter is the cached content. Upon a write, Smarty passes the cached content in these parameters. Upon a 'read', Smarty expects your function to accept this parameter by reference and populate it with the cached data. Upon a 'clear', pass a dummy variable here since it is not used. The fourth parameter is the name of the template file (needed for read/write), the fifth parameter is the cache_id (optional), and the sixth is the compile_id (optional).

SMARTY [\\$cache_handler_func](#) Smarty"""
 Smarty Smarty Smarty
IDID

Note: The last parameter (\$exp_time) was added in Smarty-2.6.0.

\$exp_time) Smarty-2.6.0

Example 15-5. example using MySQL as a cache source

```
<?php
/*
example usage:

include('Smarty.class.php');
```

```
include('mysql_cache_handler.php');

$smarty = new Smarty;
$smarty->cache_handler_func = 'mysql_cache_handler';

$smarty->display('index.tpl');

mysql database is expected in this format:

create database SMARTY_CACHE;

create table CACHE_PAGES(
CacheID char(32) PRIMARY KEY,
CacheContents MEDIUMTEXT NOT NULL
);

*/
function mysql_cache_handler($action, &$smarty_obj, &$
{
    // set db host, user and pass here
    $db_host = 'localhost';
    $db_user = 'myuser';
    $db_pass = 'mypass';
    $db_name = 'SMARTY_CACHE';
    $use_gzip = false;

    // create unique cache id
    $CacheID = md5($tpl_file.$cache_id.$compile_id);

    if(! $link = mysql_pconnect($db_host, $db_user,
        $smarty_obj->trigger_error_msg("cache
        return false;
    }
    mysql_select_db($db_name);

    switch ($action) {
        case 'read':
```

```
// save cache to database
$results = mysql_query("select * from $cache_table where id = '$cache_id'");
if(!$results) {
    $smarty_obj->_trigger_error("Cache error: failed to query database");
}
$row = mysql_fetch_array($results);
if($use_gzip && function_exists("gzuncompress")) {
    $cache_contents = gzuncompress($row['contents']);
} else {
    $cache_contents = $row['contents'];
}
$return = $results;
break;
case 'write':
    // save cache to database
    if($use_gzip && function_exists("gzcompress")) {
        // compress the content
        $contents = gzcompress($cache_contents, 9);
    } else {
        $contents = $cache_contents;
    }
    $results = mysql_query("replace into $cache_table values('$cache_id', '$cache_contents')");
    if(!$results) {
        $smarty_obj->_trigger_error("Cache error: failed to update database");
    }
}
$return = $results;
break;
case 'clear':
    // clear cache info
    if(empty($cache_id) && empty($cache_start)) {
        // clear them all
        $results = mysql_query("delete from $cache_table");
    } else {
        $results = mysql_query("delete from $cache_table where id = '$cache_id' or start_time < '$cache_start'");
    }
}
$return = $results;
break;
```

```
        }
        if(!$results) {
            $smarty_obj->_trigger_
        }
        $return = $results;
        break;
    default:
        // error, unknown action
        $smarty_obj->_trigger_error_ms
        $return = false;
        break;
    }
mysql_close($link);
return $return;

}
?>
```

[Prev](#) [PHP](#)
Output Filters

[Home](#)
[Up](#)

[Next](#)
Resources

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 15. Advanced Features

[Next](#)

Resources

Table of Contents

[Templates from \\$template_dir](#)
[Templates from any directory](#)
[Templates from other sources](#)
[Default template handler function](#)

The templates may come from a variety of sources. When you display or fetch a template, or when you include a template from within another template, you supply a resource type, followed by the appropriate path and template name. If a resource is not explicitly given the value of [\\$default_resource_type](#) is assumed.

[Prev](#) [PHP](#)

Cache Handler
Function

[Home](#)

[Up](#)

[Next](#)

Templates from
\$template_dir

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

Chapter 16. Extending Smarty With Plugins[.Smarty]

Table of Contents

[How Plugins Work](#)
[Naming Conventions](#)
[Writing Plugins](#)
[Template Functions](#)
[Modifiers](#)
[Block Functions](#)
[Compiler Functions](#)
[Prefilters/Postfilters /](#)
[Output Filters](#)
[Resources](#)
[Inserts](#)

Version 2.0 introduced the plugin architecture that is used for almost all the customizable functionality of Smarty. This includes:

2.0Smarty

- functions
- modifiers
- block functions
- compiler functions
- prefilters
- postfilters
- outputfilters
- resources
- inserts

With the exception of resources, backwards compatibility with the old way of registering handler functions via `register_*` API is preserved. If you did not use the API but instead modified the class variables `$custom_funcs`, `$custom_mods`, and other ones directly, then you will need to adjust your scripts to either use the API or convert your custom functionality into plugins.

`register_* API API`

How Plugins Work

Plugins are always loaded on demand. Only the specific modifiers, functions, resources, etc invoked in the templates scripts will be loaded. Moreover, each plugin is loaded only once, even if you have several different instances of Smarty running within the same request.

Smarty

Pre/postfilters and output filters are a bit of a special case. Since they are not mentioned in the templates, they must be registered or loaded explicitly via API functions before the template is processed. The order in which multiple filters of the same type are executed depends on the order in which they are registered or loaded.

/API

The [plugins directory](#) can be a string containing a path or an array containing multiple paths. To install a plugin, simply place it in one of the directories and Smarty will use it automatically.

Smarty

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Default template
handler function

[Up](#)

Naming Conventions

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 16. Extending Smarty With
Plugins

[Next](#)

Naming Conventions

Plugin files and functions must follow a very specific naming convention in order to be located by Smarty.

Smarty

The plugin files must be named as follows:

type . name .php

Where type is one of these plugin types:

type

- function
- modifier
- block
- compiler
- prefilter
- postfilter
- outputfilter
- resource
- insert

And name should be a valid identifier (letters, numbers, and underscores only).

name

Some examples: `function.html_select_date.php`, `resource.db.php`,
`modifier.spacify.php`.

The plugin functions inside the plugin files must be named as follows:

smarty_ type _ name ()

The meanings of type and name are the same as before.

`typename`

Smarty will output appropriate error messages if the plugin file it needs is not found, or if the file or the plugin function are named improperly.

Smarty

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Extending Smarty With
Plugins

[Up](#)

Writing Plugins

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 16. Extending Smarty With
Plugins

[Next](#)

Writing Plugins

Plugins can be either loaded by Smarty automatically from the filesystem or they can be registered at runtime via one of the register_* API functions. They can also be unregistered by using unregister_* API functions.

Smartyregister_* API unregister_* API

For the plugins that are registered at runtime, the name of the plugin function(s) does not have to follow the naming convention.

If a plugin depends on some functionality provided by another plugin (as is the case with some plugins bundled with Smarty), then the proper way to load the needed plugin is this:

Smarty

```
require_once $smarty->_get_plugin_filepath('function',
```

As a general rule, Smarty object is always passed to the plugins as the last parameter (with two exceptions: modifiers do not get passed the Smarty object at all and blocks get passed *&\$repeat* after the Smarty object to keep backwards compatibility to older versions of Smarty).

Smarty1Smarty2Smarty

[Prev](#) [PHP](#)

Naming Conventions

[Home](#)

[Up](#)

[Next](#)

Template Functions

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 16. Extending Smarty With
Plugins

[Next](#)

Template Functions

```
void smarty_function_ name (array $params, object &$smarty)
```

All attributes passed to template functions from the template are contained in the `$params` as an associative array. Either access those values directly, e.g. `$params['start']` or use `extract($params)` to import them into the symbol table.

```
$params           $params[ 'sta
```

The output (return value) of the function will be substituted in place of the function tag in the template (`fetch()` function, for example). Alternatively, the function can simply perform some other task without any output (`assign()` function).

fetch()

If the function needs to assign some variables to the template or use some other Smarty-provided functionality, it can use the supplied `$smarty` object to do so.

```
Smarty           $smarty
```

See also: [register_function\(\)](#), [unregister_function\(\)](#).

Example 16-1. function plugin with output16-1

```
<?php
/*
 * Smarty plugin
 * -----
 * File: function.eightball.php
 * Type: function
 * Name: eightball
 * Purpose: outputs a random magic answer
 *
 */
function smarty_function_eightball($params, &$smarty)
{
```

```

$answers = array( 'Yes',
'No',
'No way',
'Outlook not so good',
'Ask again soon',
'Maybe in your reality');

$result = array_rand($answers);
return $answers[$result];
}
?>

```

which can be used in the template as:

```

Question: Will we ever have time travel?
Answer: {eightball}.

```

Example 16-2. function plugin without output16-2

```

<?php
/*
 * Smarty plugin
 * -----
 * File: function.assign.php
 * Type: function
 * Name: assign
 * Purpose: assign a value to a template variable
 * -----
 */
function smarty_function_assign($params, &$smarty)
{
extract($params);

if (empty($var)) {
$smarty->trigger_error("assign: missing 'var' parameter");
}
}

```

```
return;
}

if (!in_array('value', array_keys($params))) {
$smarty->trigger_error("assign: missing 'value' parameter");
return;
}

$smarty->assign($var, $value);
}
?>
```

[Prev](#) [PHP](#)
Writing Plugins

[Home](#)
[Up](#)

[Next](#)
Modifiers

Smarty - the compiling PHP template engine

Chapter 16. Extending Smarty With

[Prev](#) [PHP](#)

Plugins
Smarty

[Next](#)

Modifiers

Modifiers are little functions that are applied to a variable in the template before it is displayed or used in some other context. Modifiers can be chained together.

```
mixed smarty_modifier_ name (mixed $value, [mixed $param1, ...])
```

The first parameter to the modifier plugin is the value on which the modifier is supposed to operate. The rest of the parameters can be optional, depending on what kind of operation is supposed to be performed.

The modifier has to return the result of its processing.

See also [register_modifier\(\)](#), [unregister_modifier\(\)](#),
[register_modifier\(\)](#), [unregister_modifier\(\)](#).

Example 16-3. simple modifier plugin

This plugin basically aliases one of the built-in PHP functions. It does not additional parameters.

PHP

```
<?php
/*
 * Smarty plugin
 * -----
 * File: modifier.capitalize.php
 * Type: modifier
 * Name: capitalize
 * Purpose: capitalize words in the string
 * -----
 */
function smarty_modifier_capitalize($string)
{
    return ucwords($string);
```

```
}
```

```
?>
```

Example 16-4. more complex modifier plugin

```
<?php
/*
 * Smarty plugin
 * -----
 * File: modifier.truncate.php
 * Type: modifier
 * Name: truncate
 * Purpose: Truncate a string to a certain length if requested,
 * optionally splitting in the middle of a word, and
 * appending the $etc string.
 */
function smarty_modifier_truncate($string, $length = 80,
    $break_words = false)
{
    if ($length == 0)
        return '';

    if (strlen($string) > $length) {
        $length -= strlen($etc);
        $fragment = substr($string, 0, $length+1);
        if ($break_words)
            $fragment = substr($fragment, 0, -1);
        else
            $fragment = preg_replace('/\s+(\S+)?$/ ', '', $fragment);
        return $fragment.$etc;
    } else
        return $string;
}
?>
```

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Template Functions[
]

[Up](#)

[]Block Functions

Smarty - the compiling PHP template engine

Chapter 16. Extending Smarty With

[Prev](#) [PHP](#)

Plugins
Smarty

[Next](#)

Block Functions

```
void smarty_block_ name (array $params, mixed $content, object  
&$smarty)
```

Block functions are functions of the form: {func} .. {/func}. In other words, they enclose a template block and operate on the contents of this block. Block functions take precedence over custom functions of the same name, that is, you cannot have both custom function {func} and block function {func} .. {/func}.

```
{func} .. {/func}{func}{func} ..
```

By default your function implementation is called twice by Smarty: once for the opening tag, and once for the closing tag (see &\$repeat below how to change this).

```
Smarty      &$repeat
```

Only the opening tag of the block function may have attributes. All attributes passed to template functions from the template are contained in the *\$params* as an associative array. You can either access those values directly, e.g. *\$params['start']* or use *extract(\$params)* to import them into the symbol table. The opening tag attributes are also accessible to your function when processing the closing tag.

```
$params['start']extract ($params)
```

The value of *\$content* variable depends on whether your function is called for the opening or closing tag. In case of the opening tag, it will be *null*, and in case of the closing tag it will be the contents of the template block. Note that the template block will have already been processed by Smarty, so all you will receive is the template output, not the template source.

```
$content Smarty,
```

The parameter *&\$repeat* is passed by reference to the function implementation and provides a possibility for it to control how many times the block is displayed. By default *\$repeat* is true at the first call of the block-function (the block opening tag) and false on all subsequent calls to the block function (the block's closing tag). Each time the function implementation returns with *&\$repeat* being true, the contents between

{func} .. {/func} are evaluated and the function implementation is called again with the new block contents in the parameter `$content` .

```
&$repeat           $repeat           &$repeat {func} .. {/func},  
$content
```

If you have nested block functions, it's possible to find out what the parent block function is by accessing `$smarty->_tag_stack` variable. Just do a `var_dump()` on it and the structure should be apparent.

```
$smarty->_tag_stack var_dump()
```

See also: [register_block\(\)](#), [unregister_block\(\)](#).

Example 16-5. block function

```
<?php  
/*  
 * Smarty plugin  
 * -----  
 * File: block.translate.php  
 * Type: block  
 * Name: translate  
 * Purpose: translate a block of text  
 * -----  
 */  
function smarty_block_translate($params, $content, &$content)  
{  
    if (isset($content)) {  
        $lang = $params['lang'];  
        // do some intelligent translation thing here with $content  
        return $translation;  
    }  
}
```

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Modifiers[]

[Up](#)

[]Compiler
Functions

Smarty - the compiling PHP template engine

Chapter 16. Extending Smarty With

[Prev](#) [PHP](#)

Plugins
smarty

[Next](#)

Compiler Functions

Compiler functions are called only during compilation of the template. They are useful for injecting PHP code or time-sensitive static content into the template. If there is both a compiler function and a custom function registered under the same name, the compiler function has precedence.

PHP

```
mixed smarty_compiler_ name (string $tag_arg, object &$smarty)
```

The compiler function is passed two parameters: the tag argument string - basically, everything from the function name until the ending delimiter, and the Smarty object. It's supposed to return the PHP code to be injected into the compiled template.

—SmartyPHP

See also [register_compiler_function\(\)](#), [unregister_compiler_function\(\)](#).

Example 16-6. simple compiler function

```
<?php
/*
 * Smarty plugin
 * -----
 * File: compiler.tplheader.php
 * Type: compiler
 * Name: tplheader
 * Purpose: Output header containing the source file name
 *          and the time it was compiled.
 *
 */
function smarty_compiler_tplheader($tag_arg, &$smarty)
{
    return "\necho '" . $smarty->_current_file . " compiled at "
        . date("Y-m-d H:i:s") . "'";
?>
```

This function can be called from the template as:

```
{* this function gets executed at compile time only *}
{tplheader}
```

The resulting PHP code in the compiled template would be something like

PHP

```
<php
echo 'index.tpl compiled at 2002-02-20 20:02';
?>
```

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Block Functions[]

[Up](#)

[/
]Prefilters/Postfilters

Smarty - the compiling PHP template engine

Chapter 16. Extending Smarty With

[Prev](#) [PHP](#)

Plugins
smarty

[Next](#)

Prefilters/Postfilters

Prefilter and postfilter plugins are very similar in concept; where they differ is in the execution -- more precisely the time of their execution.

```
string smarty_prefilter_ name (string $source, object &$smarty)
```

Prefilters are used to process the source of the template immediately before compilation. The first parameter to the prefILTER function is the template source, possibly modified by some other prefilters. The plugin is supposed to return the modified source. Note that this source is not saved anywhere, it is only used for compilation.

```
string smarty_postfilter_ name (string $compiled, object &$smarty)
```

Postfilters are used to process the compiled output of the template (the PHP code) immediately after the compilation is done but before the compiled template is saved to the filesystem. The first parameter to the postfilter function is the compiled template code, possibly modified by other postfilters. The plugin is supposed to return the modified version of this code.

PHP

Example 16-7. prefILTER plugin

```
<?php
/*
 * Smarty plugin
 * -----
 * File: prefILTER.pre01.php
 * Type: prefILTER
 * Name: pre01
 * Purpose: Convert html tags to be lowercase.
 *
 */
function smarty_prefILTER_pre01($source, &$smarty)
{
```

```
    return preg_replace('!<(\w+)[^>]+>!e', 'strtolower("$  
}  
?>
```

Example 16-8. postfilter plugin

```
<?php  
/*  
 * Smarty plugin  
 * -----  
 * File: postfilter.post01.php  
 * Type: postfilter  
 * Name: post01  
 * Purpose: Output code that lists all current template  
 * variables.  
 */  
function smarty_postfilter_post01($compiled, &$smarty)  
{  
$compiled = "<pre>\n<?php print_r(\$this->get_template_vars()); ?>\n</pre>";  
return $compiled;  
}  
?>
```

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Compiler Functions[
]

[Up](#)

[]Output Filters

Smarty - the compiling PHP template engine

Chapter 16. Extending Smarty With

[Prev](#) [PHP](#)

Plugins
smarty

[Next](#)

Output Filters

Output filter plugins operate on a template's output, after the template is loaded and executed, but before the output is displayed.

```
string smarty_outputfilter_ name (string $template_output, object  
&$smarty)
```

The first parameter to the output filter function is the template output that needs to be processed, and the second parameter is the instance of Smarty invoking the plugin. The plugin is supposed to do the processing and return the results.

Smarty

Example 16-9. output filter plugin

```
/*  
 * Smarty plugin  
 * -----  
 * File: outputfilter.protect_email.php  
 * Type: outputfilter  
 * Name: protect_email  
 * Purpose: Converts @ sign in email addresses to %40  
 * a simple protection against spambots  
 * -----  
 */  
function smarty_outputfilter_protect_email($output, &  
{  
    return preg_replace('!(\S+)(@[a-zA-Z0-9\.\-]+\.( [a-zA-Z0-9]+)%40$2', '$1%40$2', $output);  
}
```

Smarty - the compiling PHP template engine

Chapter 16. Extending Smarty With

[Prev](#) [PHP](#)

Plugins
smarty

[Next](#)

Resources

Resource plugins are meant as a generic way of providing template sources or PHP script components to Smarty. Some examples of resources: databases, LDAP, shared memory, sockets, and so on.

SmartyPHP:LDAPsockets

There are a total of 4 functions that need to be registered for each type of resource. Every function will receive the requested resource as the first parameter and the Smarty object as the last parameter. The rest of parameters depend on the function.

Smarty

```
bool smarty_resource_ name _source (string $rsrc_name, string  
&$source, object &$smarty)
```

```
bool smarty_resource_ name _timestamp (string $rsrc_name, int  
&$timestamp, object &$smarty)
```

```
bool smarty_resource_ name _secure (string $rsrc_name, object  
&$smarty)
```

```
bool smarty_resource_ name _trusted (string $rsrc_name, object  
&$smarty)
```

The first function is supposed to retrieve the resource. Its second parameter is a variable passed by reference where the result should be stored. The function is supposed to return true if it was able to successfully retrieve the resource and false otherwise.

 true false

The second function is supposed to retrieve the last modification time of the requested resource (as a UNIX timestamp). The second parameter is a variable passed by reference where the timestamp should be stored. The function is supposed to return true if the timestamp could be successfully determined, and false otherwise.

UNIX true false

The third function is supposed to return true or false, depending on whether the requested resource is secure or not. This function is used only for template resources but should still be defined.

truefalse

The fourth function is supposed to return true or false, depending on whether the requested resource is trusted or not. This function is used for only for PHP script components requested by [**{include php}**](#) tag or [**{insert}**](#) tag with *src* attribute. However, it should still be defined even for template resources.

truefalse [**{include php}**](#) [**{insert}**](#) *src* PHP

See also [register_resource\(\)](#), [unregister_resource\(\)](#).

Example 16-10. resource plugin

```
<?php
/*
 * Smarty plugin
 * -----
 * File: resource.db.php
 * Type: resource
 * Name: db
 * Purpose: Fetches templates from a database
 *
 */
function smarty_resource_db_source($tpl_name, &$tpl_sc
{
    // do database call here to fetch your template,
    // populating $tpl_source
    $sql = new SQL;
    $sql->query("select tpl_source
from my_table
where tpl_name='$tpl_name'");
    if ($sql->num_rows) {
        $tpl_source = $sql->record['tpl_source'];
        return true;
    } else {
        return false;
    }
}

function smarty_resource_db_timestamp($tpl_name, &$tpl
```

```
{  
    // do database call here to populate $tpl_timestamp.  
    $sql = new SQL;  
    $sql->query("select tpl_timestamp  
    from my_table  
    where tpl_name='$tpl_name'");  
    if ($sql->num_rows) {  
        $tpl_timestamp = $sql->record['tpl_timestamp'];  
        return true;  
    } else {  
        return false;  
    }  
}  
  
function smarty_resource_db_secure($tpl_name, &$smarty)  
{  
    // assume all templates are secure  
    return true;  
}  
  
function smarty_resource_db_trusted($tpl_name, &$smarty)  
{  
    // not used for templates  
}  
?>
```

[Prev](#) [PHP](#)

Output Filters[]

[Home](#)

[Up](#)

[Next](#)

[]Inserts

Smarty - the compiling PHP template engine

Chapter 16. Extending Smarty With

[Prev](#) [PHP](#)

Plugins
smarty

[Next](#)

Inserts

Insert plugins are used to implement functions that are invoked by [{insert}](#) tags in the template.

[{insert}](#)

```
string smarty_insert_ name (array $params, object &$smarty)
```

The first parameter to the function is an associative array of attributes passed to the insert. Either access those values directly, e.g.

\$params['start'] or use extract(\$params) to import them into the symbol table.

```
$params['start']=extract($params)
```

The insert function is supposed to return the result which will be substituted in place of the [{insert}](#) tag in the template.

[{insert}](#)

Example 16-11. insert plugin

```
<?php
/*
 * Smarty plugin
 * -----
 * File: insert.time.php
 * Type: time
 * Name: time
 * Purpose: Inserts current date/time according to for
 * -----
 */
function smarty_insert_time($params, &$smarty)
{
    if (empty($params['format'])) {
        $smarty->trigger_error("insert time: missing 'format'");
        return;
    }

    $datetime = strftime($params['format']);
    return $datetime;
```

```
}
```

```
?>
```

[Prev](#) [PHP](#)
Resources[]

[Home](#)
[Up](#)

[Next](#)
[]Appendices

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

IV. Appendixes []

Table of Contents []

- 17. [Troubleshooting \[\]](#)
 - 18. [Tips & Tricks \[\]](#)
 - 19. [Resources \[\]](#)
 - 20. [BUGS \[\]](#)
-

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

[Inserts \[\]](#)

[] Troubleshooting

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

Chapter 17. Troubleshooting

Table of Contents

[Smarty/PHP errors](#)

Smarty/PHP errors

Smarty can catch many errors such as missing tag attributes or malformed variable names. If this happens, you will see an error similar to the following:

Smarty

Example 17-1. Smarty errors

```
Warning: Smarty: [in index.tpl line 4]: syntax error:  
in /path/to/smarty/Smarty.class.php on line 1041  
  
Smarty: index.tpl4:'%blah'  
  
Fatal error: Smarty: [in index.tpl line 28]: syntax er  
in /path/to/smarty/Smarty.class.php on line 1041  
  
Smarty: index.tpl28: /path/to/smarty/Smarty.class.php1
```

Smarty shows you the template name, the line number and the error. After that, the error consists of the actual line number in the Smarty class that the error occurred.

Smartysmarty

There are certain errors that Smarty cannot catch, such as missing close tags. These types of errors usually end up in PHP compile-time parsing errors.

Smartyphp .

Example 17-2. PHP parsing errors

Parse error: parse error in /path/to/smarty/templates_

When you encounter a PHP parsing error, the error line number will correspond to the compiled PHP script, not the template itself. Usually you can look at the template and spot the syntax error. Here are some common things to look for: missing close tags for {if}{/if} or {section}{/section}, or syntax of logic within an {if} tag. If you can't find the error, you might have to open the compiled PHP file and go to the line number to figure out where the corresponding error is in the template.

```
phpphpif}{/if}  
{section}{/section}{if}php
```

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Appendixes

[Up](#)

Tips & Tricks

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

Chapter 18. Tips & Tricks

18

Table of Contents

- []
[Blank Variable Handling](#)[]
- []
[Default Variable Handling](#)[]
- []
[Passing variable title to header template](#)[]
- []
[Dates](#)[]
- []
[WAP/WML](#)[WAP/WML]
- []
[Componentized Templates](#)[]
- []
[Obfuscating E-mail Addresses](#)[]

Blank Variable Handling

There may be times when you want to print a default value for an empty variable instead of printing nothing, such as printing " " so that table backgrounds work properly. Many would use an {if} statement to handle this, but there is a shorthand way with Smarty, using the *default* variable modifier.

```
'&nbsp;['      HTML]{if}Smaty
```

Example 18-1. Printing when a variable is empty

18-1 ---

```
{* the long way *}

{if $title eq ""}
    &nbsp;
{else}
    {$title}
{/if}
```

```
{* the short way *}

{$title|default:"&nbsp;"}
```

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Troubleshooting

[Up](#)

Default Variable
Handling

□

□

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 18. Tips & Tricks[]

[Next](#)

Default Variable Handling

If a variable is used frequently throughout your templates, applying the default modifier every time it is mentioned can get a bit ugly. You can remedy this by assigning the variable its default value with the [assign](#) function.

[assign\[\]](#)

Example 18-2. Assigning a template variable its default value

18-2.

```
{* do this somewhere at the top of your template *}
{assign var="title" value=$title|default:"no title"}

{* if $title was empty, it now contains the value "no
{$title}
```

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Tips & Tricks

[Up](#)

Passing variable title to
header template



Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 18. Tips & Tricks[]

[Next](#)

Passing variable title to header template

When the majority of your templates use the same headers and footers, it is common to split those out into their own templates and include them. But what if the header needs to have a different title, depending on what page you are coming from? You can pass the title to the header when it is included.

Example 18-3. Passing the title variable to the header template

18-3.

```
mainpage.tpl
```

```
{include file="header.tpl" title="Main Page"}  
{* template body goes here *}  
{include file="footer.tpl"}
```

```
archives.tpl
```

```
{config_load file="archive_page.conf"}  
{include file="header.tpl" title=#archivePageTitle#}  
{* template body goes here *}  
{include file="footer.tpl"}
```

```
header.tpl
```

```
<HTML>  
<HEAD>  
<TITLE>{$title|default:"BC News"}</TITLE>
```

```
</HEAD>
<BODY>

footer.tpl
-----
</BODY>
</HTML>
```

When the main page is drawn, the title of "Main Page" is passed to the header.tpl, and will subsequently be used as the title. When the archives page is drawn, the title will be "Archives". Notice in the archive example, we are using a variable from the archives_page.conf file instead of a hard coded variable. Also notice that "BC News" is printed if the \$title variable is not set, using the *default* variable modifier.

```
[main page]‘Main Page’[header.tpl][archives
‘Archives’[archives_page.conf][$title]‘BC
```

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Default Variable
Handling

[Up](#)

Dates

[]

[]

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 18. Tips & Tricks[]

[Next](#)

Dates

As a rule of thumb, always pass dates to Smarty as timestamps. This allows template designers to use [date_format](#) for full control over date formatting, and also makes it easy to compare dates if necessary.

Smarty [date_format\[\]](#)

NOTE: As of Smarty 1.4.0, you can pass dates to Smarty as unix timestamps, mysql timestamps, or any date parsable by strtotime().

Smarty 1.4.0unix,mysql,[strtotime()]Smarty

Example 18-4. using date_format

18-4.

```
{$startDate|date_format}
```

OUTPUT:

Jan 4, 2001

```
{$startDate|date_format:"%Y/%m/%d"}
```

OUTPUT:

2001/01/04

```
{if $date1 < $date2}
    ...
{/if}
```

When using {html_select_date} in a template, The programmer will most

likely want to convert the output from the form back into timestamp format. Here is a function to help you with that.

{html_select_date}

Example 18-5. converting form date elements back to a timestamp

```
// this assumes your form elements are named  
// startDate_Day, startDate_Month, startDate_Year  
  
$startDate = makeTimeStamp($startDate_Year,$startDate_  
  
function makeTimeStamp($year="", $month="", $day="")  
{  
    if(empty($year))  
        $year = strftime("%Y");  
    if(empty($month))  
        $month = strftime("%m");  
    if(empty($day))  
        $day = strftime("%d");  
  
    return mktime(0,0,0,$month,$day,$year);  
}
```

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Passing variable title to
header template

[Up](#)

[WAP/WML](#)



[WAP/WML]

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 18. Tips & Tricks[]

[Next](#)

WAP/WML

WAP/WML templates require a php Content-Type header to be passed along with the template. The easiest way to do this would be to write a custom function that prints the header. If you are using caching, that won't work so we'll do it using the insert tag (remember insert tags are not cached!) Be sure that there is nothing output to the browser before the template, or else the header may fail.

WAP/WML

Example 18-6. using insert to write a WML Content-Type header

18-6. WML

```
// be sure apache is configured for the .wml extensions
// put this function somewhere in your application, or
function insert_header() {
    // this function expects $content argument
    extract(func_get_arg(0));
    if(empty($content))
        return;
    header($content);
    return;
}

// your Smarty template _must_ begin with the insert tag
{insert name=header content="Content-Type: text/vnd.wap.wml; charset=utf-8; version=1.0"}

<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/wml_1.1.xml">

<!-- begin new wml deck -->
<wml>
<!-- begin first card -->
<card>
```

```
<do type="accept">
<go href="#two"/>
</do>
<p>
Welcome to WAP with Smarty!
Press OK to continue...
</p>
</card>
<!-- begin second card -->
<card id="two">
<p>
Pretty easy isn't it?
</p>
</card>
</wml>
```

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Dates

[Up](#)

Componentized
Templates



Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 18. Tips & Tricks[]

[Next](#)

Componentized Templates

This tip is a bit of a hack, but still a neat idea. Use at your own risk. ;-)

;-)

Traditionally, programming templates into your applications goes as follows: First, you accumulate your variables within your PHP application, (maybe with database queries.) Then, you instantiate your Smarty object, assign the variables and display the template. So lets say for example we have a stock ticker on our template. We would collect the stock data in our application, then assign these variables in the template and display it. Now wouldn't it be nice if you could add this stock ticker to any application by merely including the template, and not worry about fetching the data up front?

PHPSmarty

You can embed PHP into your templates with the {php}{/php} tags. With this, you can setup self contained templates with their own data structures for assigning their own variables. With the logic embedded like this, you can keep the template & logic together. This way no matter where the template source is coming from, it is always together as one component.

{php}{/php}PHP

Example 18-7. componentized template

18-7.

```
{* Smarty *}

{php}
    // setup our function for fetching stock data
    function fetch_ticker($symbol,&$ticker_name,&$
```

```

        // put logic here that fetches $ticker
        // and $ticker_price from some resource
    }

    // call the function
fetch_ticker("YHOO",$ticker_name,$ticker_price)

    // assign template variables
$this->assign("ticker_name",$ticker_name);
$this->assign("ticker_price",$ticker_price);

{/php}

Stock Name: {$ticker_name} Stock Price: {$ticker_price}

```

As of Smarty 1.5.0, there is even a cleaner way. You can include php in your templates with the {include_php ...} tag. This way you can keep your PHP logic separated from the template logic. See the [include_php](#) function for more information.

Smarty 1.5.0{include_php ...}phpPHP [include_php](#)

Example 18-8. componentized template with include_php

18-8. include_php

```
load_ticker.php
```

```
-----
```

```

<?php
    // setup our function for fetching stock data
    function fetch_ticker($symbol,&$ticker_name,&$ticker_price)
        // put logic here that fetches $ticker
        // and $ticker_price from some resource
    }

    // call the function
fetch_ticker("YHOO",$ticker_name,$ticker_price)

```

```
// assign template variables
$this->assign("ticker_name",$ticker_name);
$this->assign("ticker_price",$ticker_price);
?>

index.tpl
-----
{* Smarty *}

{include_php file="load_ticker.php"}

Stock Name: {$ticker_name} Stock Price: {$ticker_price}
```

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

WAP/WML

[Up](#)

Obfuscating E-mail
Addresses

[WAP/WML]



Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

Chapter 18. Tips & Tricks[]

[Next](#)

Obfuscating E-mail Addresses

Do you ever wonder how your E-mail address gets on so many spam mailing lists? One way spammers collect E-mail addresses is from web pages. To help combat this problem, you can make your E-mail address show up in scrambled javascript in the HTML source, yet it will look and work correctly in the browser. This is done with the mailto plugin.

HTMLjavascriptmailto

Example 18-9. Example of Obfuscating an E-mail Address

18-9.

```
index.tpl
```

```
-----
```

```
Send inquiries to  
{mailto address=$EmailAddress encode="javascript" subj
```

Technical Note: This method isn't 100% foolproof. A spammer could conceivably program his e-mail collector to decode these values, but not likely.

100%

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Componentized
Templates

[Up](#)

Resources



Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

Chapter 19. Resources

- Smarty's homepage is located at <http://smarty.php.net/>. You can join the mailing list by sending an e-mail to smarty-general-subscribe@lists.php.net. An archive of the mailing list can be viewed at <http://marc.theaimsgroup.com/?l=smarty&r=1&w=2>
Smarty <http://smarty.php.net/>
smarty-general-subscribe@lists.php.net
<http://marc.theaimsgroup.com/?l=smarty&r=1&w=2>

[PHP](#) [PHP](#)

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Obfuscating E-mail
Addresses

[Up](#)

[BUGS](#)

Smarty - the compiling PHP template engine

[Prev](#) [PHP](#)

Chapter 20. BUGS

▪

Check the BUGS file that comes with the latest distribution of Smarty, or check the website.

Smarty

[Prev](#) [PHP](#)

Resources

[Home](#)

[Up](#)

Smarty - the compiling PHP template engine

[Prev](#) [PHP](#)

Chapter 21. LIST

zhangyx@12365.sd.cn
kolinwebmaster@im22.uni.cc
libk libkhorse@sohu.com
huix huix22@hotmail.com
flyhan hanbd7@sina.com
bralf ab_blunt@yahoo.com.cn
yuguanglou yuguanglou@21cn.com
fwolf fwolf001@tom.com
gavinguo gavin@jqw.cn
cnhfyycnhfyy@126.com
webmaster@ehome.51.net

[Prev](#)

[PHP](#)

[BUG](#)

[Home](#)

[Up](#)

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

Associative arrays

You can also reference associative array variables that are assigned from PHP by specifying the key after the '.' (period) symbol.

Example 4-2. accessing associative array variables

index.php:

```
$smarty = new Smarty;
$smarty->assign('Contacts',
  array('fax' => '555-222-9876',
    'email' => 'zaphod@slartibartfast.com',
    'phone' => array('home' => '555-444-3333',
      'cell' => '555-111-1234')));
$smarty->display('index.tpl');
```

index.tpl:

```
{$Contacts.fax}<br>
{$Contacts.email}<br>
{* you can print arrays of arrays as well *}
{$Contacts.phone.home}<br>
{$Contacts.phone.cell}<br>
```

OUTPUT:

```
555-222-9876<br>
zaphod@slartibartfast.com<br>
555-444-3333<br>
555-111-1234<br>
```

[Prev](#) [PHP](#)

Variables

[Home](#)

[Up](#)

[Next](#)

Array indexes

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

Array indexes

You can reference arrays by their index, much like native PHP syntax.

Example 4-3. accessing arrays by index

index.php:

```
$smarty = new Smarty;
$smarty->assign('Contacts',
array('555-222-9876',
'zaphod@slartibartfast.com',
array('555-444-3333',
'555-111-1234')));
$smarty->display('index.tpl');
```

index.tpl:

```
{$Contacts[0]}<br>
{$Contacts[1]}<br>
{* you can print arrays of arrays as well *}
{$Contacts[2][0]}<br>
{$Contacts[2][1]}<br>
```

OUTPUT:

```
555-222-9876<br>
zaphod@slartibartfast.com<br>
555-444-3333<br>
555-111-1234<br>
```

[Prev](#) [PHP](#)

Associative arrays

[Home](#)

[Up](#)

[Next](#)

Objects

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

Objects

Properties of objects assigned from PHP can be referenced by specifying the property name after the '->' symbol.

Example 4-4. accessing object properties

```
name: {$person->name}<br>
email: {$person->email}<br>
```

OUTPUT:

```
name: Zaphod Beeblebrox<br>
email: zaphod@slartibartfast.com<br>
```

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Array indexes

[Up](#)

Variables loaded from
config files

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

{\$smarty.config}

{\$smarty} variable can be used to refer to loaded config variables.
{\$smarty.config.foo} is a synonym for {#foo#}. See the section on
[config_load](#) for an example.

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

 {\$smarty.capture}

[Up](#)

 {\$smarty.section},
 {\$smarty.foreach}

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

Request variables

The request variables such as get, post, cookies, server, environment, and session variables can be accessed as demonstrated in the examples below:

Example 4-6. displaying request variables

```
{* display value of page from URL (GET) http://www.domain.com *}
{$smarty.get.page}

{* display the variable "page" from a form (POST) *}
{$smarty.post.page}

{* display the value of the cookie "username" *}
{$smarty.cookies.username}

{* display the server variable "SERVER_NAME" *}
{$smarty.server.SERVER_NAME}

{* display the system environment variable "PATH" *}
{$smarty.env.PATH}

{* display the php session variable "id" *}
{$smarty.session.id}

{* display the variable "username" from merged get/post variables *}
{$smarty.request.username}
```

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

{\$smarty} reserved
variable

[Up](#)

{\$smarty.now}

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

{\$smarty.now}

The current timestamp can be accessed with {\$smarty.now}. The number reflects the number of seconds passed since the so-called Epoch (January 1, 1970) and can be passed directly to date_format modifier for display purposes.

Example 4-7. using {\$smarty.now}

```
{* use the date_format modifier to show current date and time *}
{$smarty.now|date_format:"%Y-%m-%d %H:%M:%S"}
```

[Prev](#) [PHP](#)

Request variables

[Home](#)

[Up](#)

[Next](#)

 {\$smarty.const}

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

{\$smarty.const}

You can access PHP constant values directly.

Example 4-8. using {\$smarty.const}

```
{$smarty.const._MY_CONST_VAL}
```

[Prev](#) [PHP](#)

[{\\$smarty.now}](#)

[Home](#)

[Up](#)

[Next](#)

[{\\$smarty.capture}](#)

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

{\$smarty.capture}

The output captured via {capture}..{/capture} construct can be accessed using {\$smarty} variable. See section on [capture](#) for an example.

[Prev](#) [PHP](#)

{\$smarty.const}

[Home](#)

[Up](#)

[Next](#)

{\$smarty.config}

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

{\$smarty.section}, {\$smarty.foreach}

{\$smarty} variable can be used to refer to 'section' and 'foreach' loop properties. See docs for [section](#) and [foreach](#).

[Prev](#) [PHP](#)

 {\$smarty.config}

[Home](#)

[Up](#)

[Next](#)

 {\$smarty.template}

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

{\$smarty.template}

This variable contains the name of the current template being processed.

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

`{$smarty.section},
{$smarty.foreach}`

[Up](#)

Variable Modifiers

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

iteration

iteration is used to display the current loop iteration.

iteration []

Iteration always starts with 1 and is incremented by one each iteration.

iteration 1 1.]

[Prev](#) [PHP](#)

foreach,foreachelse

[Home](#)

[Up](#)

[Next](#)

first

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

first

first is set to true if the current foreach iteration is the first one.

foreach **first** true.

[Prev](#) [PHP](#)
iteration

[Home](#)
[Up](#)

[Next](#)
last

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

last

last is set to true if the current foreach iteration is the last one.

foreach **last** true.

[Prev](#) [PHP](#)
first

[Home](#)
[Up](#)

[Next](#)
show

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

show

show is used as a parameter to foreach. *show* is a boolean value, true or false. If false, the foreach will not be displayed. If there is a foreachelse present, that will be alternately displayed.

show foreach . true false. false
show .

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

last

[Up](#)

total

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

total

total is used to display the number of iterations that this foreach will loop. This can be used inside or after the foreach.

***total* .**

[Prev](#) [PHP](#)
show

[Home](#)
[Up](#)

[Next](#)
include

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

index

index is used to display the current loop index, starting with zero (or the start attribute if given), and incrementing by one (or by the step attribute if given.)

index 0(start)1(step).

Technical Note: If the step and start section properties are not modified, then this works the same as the iteration section property, except it starts on 0 instead of 1.

stepstartiteration0.

Example 7-21. section property index 7-21. section index

```
{section name=customer loop=$custid}
{$smarty.section.customer.index} id: {$custid[
{/section}]}
```

OUTPUT:

```
0 id: 1000<br>
1 id: 1001<br>
2 id: 1002<br>
```

[Prev](#) [PHP](#)

section,sectionelse

[Home](#)

[Up](#)

[Next](#)

[index_prev](#)

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

index_prev

index_prev is used to display the previous loop index. on the first loop, this is set to -1.

index_prev . -1.

Example 7-22. section property index_prev

7-22. section index_prev

```
{section name=customer loop=$custid}
{$smarty.section.customer.index} id: {$custid[
{* FYI, $custid[customer.index] and $custid[cu
{if $custid[customer.index_prev] ne $custid[cu
The customer id changed<br>
{/if}
{/section}
```

OUTPUT:

```
0 id: 1000<br>
The customer id changed<br>
1 id: 1001<br>
The customer id changed<br>
2 id: 1002<br>
The customer id changed<br>
```

[Prev](#) [PHP](#)

index

[Home](#)

[Up](#)

[Next](#)

index_next

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

index_next

index_next is used to display the next loop index. On the last loop, this is still one more than the current index (respecting the setting of the step attribute, if given.)
index_next . 1(step).

Example 7-23. section property index_next

7-22. section index_next

```
{section name=customer loop=$custid}
{$smarty.section.customer.index} id: {$custid[0]
{* FYI, $custid[customer.index] and $custid[cu
{if $custid[customer.index_next] ne $custid[cu
The customer id will change<br>
{/if}
{/section}
```

OUTPUT:

```
0 id: 1000<br>
The customer id will change<br>
1 id: 1001<br>
The customer id will change<br>
2 id: 1002<br>
The customer id will change<br>
```

[Prev](#) [PHP](#)
[index_prev](#)

[Home](#)
[Up](#)

[Next](#)
[iteration](#)

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

iteration

iteration is used to display the current loop iteration.

iteration .

NOTE: This is not affected by the section properties start, step and max, unlike the index property. Iteration also starts with 1 instead of 0 like index. rownum is an alias to iteration, they work identical.

iteration indexstartstepmax1(index0).rownum iteration.

Example 7-24. section property iteration

7-24. section iteration

```
{section name=customer loop=$custid start=5 step=2}
current loop iteration: {$smarty.section.customer.index}
{$smarty.section.customer.index} id: {$custid[$smarty.section.customer.index]}
{* FYI, $custid[customer.index] and $custid[customer.index]_next are identical}
{if $custid[customer.index_next] neq $custid[customer.index]}
The customer id will change<br>
{/if}
{/section}
```

OUTPUT:

```
current loop iteration: 1
5 id: 1000<br>
The customer id will change<br>
current loop iteration: 2
7 id: 1001<br>
The customer id will change<br>
current loop iteration: 3
9 id: 1002<br>
The customer id will change<br>
```

[index](#) [next](#)

[Up](#)

[first](#)

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

first

first is set to true if the current section iteration is the first one.

first true.

Example 7-25. section property first 7-25. section first

```
{section name=customer loop=$custid}
{if $smarty.section.customer.first}
<table>
{/if}

<tr><td>{$smarty.section.customer.index} id:
{$custid[customer]}</td></tr>

{if $smarty.section.customer.last}
</table>
{/if}
{/section}
```

OUTPUT:

```
<table>
<tr><td>0 id: 1000</td></tr>
<tr><td>1 id: 1001</td></tr>
<tr><td>2 id: 1002</td></tr>
</table>
```

[Prev](#) [PHP](#)

iteration

[Home](#)

[Up](#)

[Next](#)

last

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

last

last is set to true if the current section iteration is the last one.

last true.

Example 7-26. section property last 7-26. section last

```
{section name=customer loop=$custid}
{if $smarty.section.customer.first}
<table>
{/if}

<tr><td>{$smarty.section.customer.index} id:
{$custid[customer]}</td></tr>

{if $smarty.section.customer.last}
</table>
{/if}
{/section}
```

OUTPUT:

```
<table>
<tr><td>0 id: 1000</td></tr>
<tr><td>1 id: 1001</td></tr>
<tr><td>2 id: 1002</td></tr>
</table>
```

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

rownum

rownum is used to display the current loop iteration, starting with one. It is an alias to iteration, they work identically.

rownum . iteration.

Example 7-27. section property rownum

7-27. section rownum

```
{section name=customer loop=$custid}
{$smarty.section.customer.rownum} id: {$custid}
{/section}
```

OUTPUT:

```
1 id: 1000<br>
2 id: 1001<br>
3 id: 1002<br>
```

[Prev](#)

[PHP](#)

last

[Home](#)

[Up](#)

[Next](#)

loop

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

loop

loop is used to display the last index number that this section looped. This can be used inside or after the section.

loop . .

Example 7-28. section property index

```
{section name=customer loop=$custid}
{$smarty.section.customer.index} id: {$custid[{$smarty.section.customer.index}]}
{/section}
```

There were {\$smarty.section.customer.loop} cus

OUTPUT:

```
0 id: 1000<br>
1 id: 1001<br>
2 id: 1002<br>
```

There were 3 customers shown above.

[Prev](#) [PHP](#)
rownum

[Home](#)
[Up](#)

[Next](#)
show

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

show

show is used as a parameter to section. *show* is a boolean value, true or false. If false, the section will not be displayed. If there is a sectionelse present, that will be alternately displayed.

show section . **show** true false. false.

Example 7-29. section attribute show

7-29. section show

```
{* $show_customer_info may have been passed fr  
application, to regulate whether or not this s  
{section name=customer loop=$custid show=$show  
{$smarty.section.customer.rownum} id: {$custid}  
{/section}  
  
{if $smarty.section.customer.show}  
the section was shown.  
{else}  
the section was not shown.  
{/if}
```

OUTPUT:

```
1 id: 1000<br>  
2 id: 1001<br>  
3 id: 1002<br>
```

the section was shown.

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

total

total is used to display the number of iterations that this section will loop. This can be used inside or after the section.

total . .

Example 7-30. section property total

7-30. section total

```
{section name=customer loop=$custid step=2}
{$smarty.section.customer.index} id: {$custid[{$smarty.section.customer.index}]}
{/section}
```

There were {\$smarty.section.customer.total} cu

OUTPUT:

```
0 id: 1000<br>
2 id: 1001<br>
4 id: 1002<br>
```

There were 3 customers shown above.

[Prev](#) [PHP](#)
show

[Home](#)
[Up](#)

[Next](#)
strip

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

Templates from \$template_dir

Templates from the \$template_dir do not require a template resource, although you can use the file: resource for consistency. Just supply the path to the template you want to use relative to the \$template_dir root directory.

Example 15-6. using templates from \$template_dir

```
// from PHP script
$smarty->display("index.tpl");
$smarty->display("admin/menu.tpl");
$smarty->display("file:admin/menu.tpl"); // same as or

{* from within Smarty template *}
{include file="index.tpl"}
{include file="file:index.tpl"} {* same as one above *}
```

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Resources

[Up](#)

Templates from any
directory

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

Templates from any directory

Table of Contents

[Windows Filepaths](#)

Templates outside of the \$template_dir require the file: template resource type, followed by the absolute path and name of the template.

Example 15-7. using templates from any directory

```
// from PHP script
$smarty->display("file:/export/templates/index.tpl");
$smarty->display("file:/path/to/my/templates/menu.tpl"

{* from within Smarty template *}
{include file="file:/usr/local/share/templates/navigation.tpl"}
```

Windows Filepaths

If you are using a Windows machine, filepaths usually include a drive letter (C:) at the beginning of the pathname. Be sure to use "file:" in the path to avoid namespace conflicts and get the desired results.

Example 15-8. using templates from windows file paths

```
// from PHP script
$smarty->display("file:C:/export/templates/index.tpl")
$smarty->display("file:F:/path/to/my/templates/menu.tpl"

{* from within Smarty template *}
{include file="file:D:/usr/local/share/templates/navigation.tpl"}
```

[Prev](#) [PHP](#)

[Home](#)

[Next](#)

Templates from
\$template_dir

[Up](#)

Templates from other
sources

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

Templates from other sources

You can retrieve templates using whatever possible source you can access with PHP: databases, sockets, LDAP, and so on. You do this by writing resource plugin functions and registering them with Smarty.

See [resource plugins](#) section for more information on the functions you are supposed to provide.

Note: Note that you cannot override the built-in `file` resource, but you can provide a resource that fetches templates from the file system in some other way by registering under another resource name.

Example 15-9. using custom resources

```
// from PHP script

// put these function somewhere in your application
function db_get_template ($tpl_name, &$tpl_source, &$tpl_timestamp)
{
    // do database call here to fetch your template,
    // populating $tpl_source
    $sql = new SQL;
    $sql->query("select tpl_source
from my_table
where tpl_name='$tpl_name'");
    if ($sql->num_rows) {
        $tpl_source = $sql->record['tpl_source'];
        return true;
    } else {
        return false;
    }
}

function db_get_timestamp($tpl_name, &$tpl_timestamp,
{
```

```
// do database call here to populate $tpl_timestamp.
$sql = new SQL;
$sql->query("select tpl_timestamp
from my_table
where tpl_name='$tpl_name'");
if ($sql->num_rows) {
$tpl_timestamp = $sql->record['tpl_timestamp'];
return true;
} else {
return false;
}
}

function db_get_secure($tpl_name, &$smarty_obj)
{
// assume all templates are secure
return true;
}

function db_get_trusted($tpl_name, &$smarty_obj)
{
// not used for templates
}

// register the resource name "db"
$smarty->register_resource("db", array("db_get_template",
"db_get_timestamp",
"db_get_secure",
"db_get_trusted"));

// using resource from php script
$smarty->display("db:index.tpl");

{* using resource from within Smarty template *}
{include file="db:/extras/navigation.tpl"}
```

Templates from any
directory

[Up](#)

Default template
handler function

Smarty - the compiling PHP template engine

[Prev](#)

[PHP](#)

[Next](#)

Default template handler function

You can specify a function that is used to retrieve template contents in the event the template cannot be retrieved from its resource. One use of this is to create templates that do not exist on-the-fly.

Example 15-10. using the default template handler function

```
<?php
// put this function somewhere in your application

function make_template ($resource_type, $resource_name
{
    if( $resource_type == 'file' ) {
        if ( ! is_readable ( $resource_name ) )
            // create the template file, r
            $template_source = "This is a
            $template_timestamp = time();
            $smarty_obj->_write_file($resou
            return true;
    }
} else {
    // not a file
    return false;
}
}

// set the default handler
$smarty->default_template_handler_func = 'make_template'
?>
```

[Prev](#) [PHP](#)

Templates from other
sources

[Home](#)

[Up](#)

[Next](#)

Extending Smarty With
Plugins