

# Smarty Manual

Monte Ohrt <monte at ohrt dot com>  
Andrei Zmievski <andrei@php.net>

Mario Ramírez <mario\_ramirez@fjcorona.com.mx>

---

## **Smarty - El motor compilador de Plantillas para PHP**

by Monte Ohrt <monte at ohrt dot com> and Andrei Zmievski <andrei@php.net>

by Mario Ramírez <mario\_ramirez@fjcorona.com.mx>

Publication date 2010-09-20

Copyright © 2001-2005 New Digital Group, Inc.

---

---

# Table of Contents

Prólogo .....	xii
I. Iniciando .....	1
1. Que es Smarty? .....	3
2. Instalación .....	5
Requerimientos .....	5
Instalación Básica .....	5
Expandiendo la configuración .....	9
II. Smarty For Template Designers .....	12
3. Basic Syntax .....	15
Comentarios .....	15
Variables .....	15
Funciones .....	16
Atributos .....	17
Colocando variables entre comillas dobles .....	17
Matemáticas .....	18
Escaping Smarty Parsing .....	18
4. Variables .....	20
Variables definidas desde PHP .....	20
Arreglos asociativos .....	21
Índices de Matrices .....	22
Objects .....	23
Variables cargadas desde archivos de configuración .....	24
La variable reservada { <code>\$smarty</code> } .....	26
Solicitud de Variables .....	26
{ <code>\$smarty.now</code> } .....	26
{ <code>\$smarty.const</code> } .....	27
{ <code>\$smarty.capture</code> } .....	27
{ <code>\$smarty.config</code> } .....	27
{ <code>\$smarty.section</code> }, { <code>\$smarty.foreach</code> } .....	27
{ <code>\$smarty.template</code> } .....	27
{ <code>\$smarty.version</code> } .....	27
{ <code>\$smarty.ldelim</code> }, { <code>\$smarty.rdelim</code> } .....	27
5. Modificadores de variables .....	28
capitalize .....	29
cat .....	29
count_characters .....	30
count_paragraphs .....	31
count_sentences .....	32
count_words .....	33
date_format .....	34
default .....	37
escape .....	38
indent .....	39
lower .....	41
nl2br .....	41
regex_replace .....	42
replace .....	43
spacify .....	44
string_format .....	45
strip .....	46
strip_tags .....	47

truncate .....	48
upper .....	50
wordwrap .....	51
6. Combinando Modificadores .....	53
7. Funciones Integradas .....	54
capture .....	54
config_load .....	55
{foreach},{foreachelse} .....	58
iteration .....	61
first .....	61
last .....	61
show .....	61
total .....	61
include .....	62
{include_php} .....	64
insert .....	65
if,elseif,else .....	66
{ldelim},{rdelim} .....	69
literal .....	69
{php} .....	70
section,sectionelse .....	71
index .....	78
index_prev .....	79
index_next .....	79
iteration .....	80
first .....	82
last .....	82
rownum .....	82
loop .....	82
show .....	83
total .....	84
{strip} .....	85
8. Custom Functions .....	87
{assign} .....	87
{counter} .....	88
cycle .....	89
{debug} .....	91
{eval} .....	91
{fetch} .....	93
{html_checkboxes} .....	94
{html_image} .....	96
{html_options} .....	97
{html_radios} .....	100
{html_select_date} .....	103
{html_select_time} .....	108
{html_table} .....	112
math .....	114
{mailto} .....	117
{popup_init} .....	118
popup .....	119
{textformat} .....	124
9. Config Files .....	128
10. Debugging Console .....	130
III. Smarty For Programmers .....	131

11. Constantes .....	135
SMARTY_DIR .....	135
SMARTY_CORE_DIR .....	135
12. Clase Variables de Smarty .....	136
\$template_dir .....	136
\$compile_dir .....	136
\$config_dir .....	136
\$plugins_dir .....	136
\$debugging .....	137
\$debug_tpl .....	137
\$debugging_ctrl .....	137
\$autoload_filters .....	137
\$compile_check .....	138
\$force_compile .....	138
\$caching .....	138
\$cache_dir .....	138
\$cache_lifetime .....	139
\$cache_handler_func .....	139
\$cache_modified_check .....	139
\$config_overwrite .....	139
\$config_booleanize .....	140
\$config_read_hidden .....	140
\$config_fix_newlines .....	140
\$default_template_handler_func .....	140
\$php_handling .....	141
\$security .....	141
\$secure_dir .....	141
\$security_settings .....	141
\$trusted_dir .....	142
\$left_delimiter .....	142
\$right_delimiter .....	142
\$compiler_class .....	142
\$request_vars_order .....	142
\$request_use_auto_globals .....	142
\$error_reporting .....	143
\$compile_id .....	143
\$use_sub_dirs .....	143
\$default_modifiers .....	144
\$default_resource_type .....	144
13. La clase Methods() de Smarty .....	145
append() .....	146
append_by_ref .....	147
assign() .....	148
assign_by_ref .....	149
clear_all_assign() .....	150
clear_all_cache .....	151
clear_assign() .....	152
clear_cache() .....	153
clear_compiled_tpl() .....	154
clear_config() .....	155
config_load() .....	156
display() .....	157
fetch() .....	159
get_config_vars() .....	162

get_registered_object() .....	163
get_template_vars() .....	164
is_cached() .....	165
load_filter() .....	167
register_block() .....	168
register_compiler_function .....	169
register_function() .....	170
register_modifier() .....	172
register_object() .....	173
register_outputfilter() .....	174
register_postfilter() .....	175
register_prefilter() .....	176
register_resource .....	177
trigger_error .....	178
template_exists() .....	179
unregister_block .....	180
unregister_compiler_function() .....	181
unregister_function() .....	182
unregister_modifier() .....	183
unregister_object() .....	184
unregister_outputfilter() .....	185
unregister_postfilter() .....	186
unregister_prefilter() .....	187
unregister_resource() .....	188
14. Cache .....	189
Configurando el Cache .....	189
Multiples caches por pagina .....	192
Cache Groups .....	193
Controlando salida de Cacheabilidad de plugins .....	194
15. Caracteristicas Avanzadas .....	197
Objetos .....	197
Prefilters .....	198
Postfilters .....	199
Filtros de salida .....	200
Función manipuladora de cache .....	200
Recursos .....	202
Templates desde \$template_dir .....	202
Templates partiendo de cualquier directorio .....	202
Templates partiendo de otras fuentes .....	203
Función manipuladora de Template por default .....	205
16. Extendiendo Smarty con plugins .....	206
Como funcionan los Plugins .....	206
Nombres convencionales .....	206
Escribiendo Plugins .....	207
Funciones de Template .....	207
Modificadores .....	209
Block Functions .....	211
Funciones Compiladoras .....	212
Prefiltros/Postfiltros .....	213
Filtros de Salida .....	215
Fuentes .....	216
Inserts .....	219
IV. Appendixes .....	220
17. Localización de Errores .....	222

Errores Smarty/PHP .....	222
18. Consejos y Trucos .....	224
Manipulación de Variables Vacías .....	224
Manipulación del valor default de una variable .....	224
Pasando la variable titulo a la cabecera del template .....	225
Fechas .....	226
WAP/WML .....	227
Templates con Componetes .....	229
Ofuscando direcciones de E-mail .....	230
19. Fuentes .....	231
20. ERRORES .....	232

---

## List of Examples

2.1. Archivos de la librería Smarty .....	5
2.2. Creando una instancia Smarty de Smarty .....	5
2.3. Reemplazar por la ruta absoluta de la librería del archivo .....	6
2.4. Adicionar el directorio de la librería para incluirlo en el <code>include_path</code> de PHP .....	6
2.5. Defina la constante <code>SMARTY_DIR</code> manualmente .....	6
2.6. Ejemplo de estructura de archivo .....	7
2.7. Configurando permisos de archivos .....	8
2.8. Editando <code>/web/www.example.com/smarty/guestbook/templates/index.tpl</code> .....	8
2.9. Editando <code>/web/www.example.com/docs/guestbook/index.php</code> .....	9
2.10. Editando <code>/php/includes/guestbook/setup.php</code> .....	10
2.11. Editando <code>/web/www.example.com/docs/guestbook/index.php</code> .....	11
3.1. Comentarios .....	15
3.2. Variables .....	16
3.3. Sintaxis de Funciones .....	16
3.4. Sintaxis de atributos de Funciones .....	17
3.5. Sintaxis entre comillas .....	18
3.6. Ejemplos de matemáticas .....	18
3.7. Ejemplo cambiando delimitadores .....	19
4.1. variables definidas .....	21
4.2. Accesando variables de matriz asociativa .....	22
4.3. Accesando matrices por sus índices .....	23
4.4. Accesando propiedades de los Objetos .....	24
4.5. Variables de configuración .....	25
4.6. Mostrando solicitud de variables .....	26
4.7. Usando <code>{\$smarty.now}</code> .....	27
4.8. Usando <code>{\$smarty.const}</code> .....	27
5.1. Ejemplo de modificador .....	28
5.2. <code>capitalize</code> .....	29
5.3. <code>cat</code> .....	30
5.4. <code>count_characters</code> .....	31
5.5. <code>count_paragraphs</code> .....	32
5.6. <code>count_sentences</code> .....	33
5.7. <code>count_words</code> .....	34
5.8. <code>date_format</code> .....	35
5.9. <code>default</code> .....	37
5.10. <code>escape</code> .....	38
5.11. <code>indent</code> .....	40
5.12. <code>lower</code> .....	41
5.13. <code>nl2br</code> .....	42
5.14. <code>regex_replace</code> .....	43
5.15. <code>replace</code> .....	44
5.16. <code>spacify</code> .....	45
5.17. <code>string_format</code> .....	46
5.18. <code>strip</code> .....	47
5.19. <code>strip_tags</code> .....	48
5.20. <code>truncate</code> .....	50
5.21. <code>upper</code> .....	51
5.22. <code>wordwrap</code> .....	52
6.1. Combinando Modificadores .....	53
7.1. capturando contenido de template .....	55
7.2. Función <code>{config_load}</code> .....	57



7.3. Función <code>config_load</code> con <code>section</code> .....	58
7.4. <code>foreach</code> .....	59
7.5. <code>foreach key</code> .....	60
7.6. Ejemplo de <code>{foreach}</code> - con base de datos (eg PEAR o ADODB) .....	61
7.7. función <code>{include}</code> .....	62
7.8. Función <code>{include}</code> pasando variables .....	63
7.9. <code>{include}</code> y asignacion de variables .....	63
7.10. Ejemplos de recursos para la función <code>include</code> .....	64
7.11. función <code>{include_php}</code> .....	65
7.12. función <code>{insert}</code> .....	66
7.13. sentencia <code>if</code> .....	68
7.14. <code>{ldelim}</code> , <code>{rdelim}</code> .....	69
7.15. Etiqueta literal .....	70
7.16. Etiqueta <code>{php}</code> .....	70
7.17. <code>section</code> .....	73
7.18. loop(ciclo) de la variable <code>section</code> .....	74
7.19. Nombres de <code>section</code> .....	75
7.20. <code>sections</code> anidadas .....	76
7.21. <code>sections</code> y matrices asociativas .....	77
7.22. <code>{sectionelse}</code> .....	78
7.23. <code>{section}</code> propiedades del <code>index</code> .....	78
7.24. <code>{section}</code> propiedades del <code>index_next</code> y <code>index_prev</code> .....	80
7.25. <code>{section}</code> propiedades de <code>iteration</code> .....	81
7.26. <code>{section}</code> propiedades <code>first</code> y <code>last</code> .....	82
7.27. <code>{section}</code> propiedades de <code>index</code> .....	83
7.28. <code>section</code> atributos de <code>show</code> .....	84
7.29. <code>{section}</code> propiedades de <code>total</code> .....	85
7.30. <code>{strip}</code> tags .....	86
8.1. <code>{assign}</code> .....	87
8.2. Accesando variables desde un script de PHP. <code>{assign}</code> .....	88
8.3. <code>counter</code> .....	89
8.4. <code>cycle</code> .....	91
8.5. <code>{eval}</code> .....	92
8.6. <code>fetch</code> .....	93
8.7. <code>{html_checkboxes}</code> .....	95
8.8. ejemplo de base de datos (eg PEAR o ADODB): .....	96
8.9. <code>html_image</code> example .....	97
8.10. <code>{html_options}</code> .....	99
8.11. <code>{html_options}</code> - Ejemplo con base de datos (eg PEAR o ADODB): .....	100
8.12. <code>{html_radios}</code> : Ejemplo 1 .....	101
8.13. <code>{html_radios}</code> : Ejemplo 2 .....	102
8.14. <code>{html_radios}</code> - Ejemplo con base de Datos (eg PEAR o ADODB): .....	103
8.15. <code>{html_select_date}</code> .....	107
8.16. <code>{html_select_date}</code> .....	108
8.17. <code>{html_select_time}</code> .....	111
8.18. <code>html_table</code> .....	114
8.19. <code>{math}</code> .....	116
8.20. <code>{mailto}</code> .....	118
8.21. <code>popup_init</code> .....	119
8.22. <code>popup</code> .....	124
8.23. <code>{textformat}</code> .....	126
9.1. Ejemplo de sintaxis de un archivo de configuración .....	128
11.1. <code>SMARTY_DIR</code> .....	135
11.2. <code>SMARTY_CORE_DIR</code> .....	135

12.1. multiple \$plugins_dir .....	137
12.2. Arreglo de variables de configuración .....	140
12.3. \$compile_id .....	143
13.1. append .....	146
13.2. append_by_ref .....	147
13.3. assign() .....	148
13.4. assign_by_ref() .....	149
13.5. clear_all_assign() .....	150
13.6. clear_all_cache .....	151
13.7. clear_assign() .....	152
13.8. clear_cache() .....	153
13.9. clear_compiled_tpl() .....	154
13.10. clear_config() .....	155
13.11. config_load() .....	156
13.12. display() .....	157
13.13. Ejemplos de recursos de la función display .....	158
13.14. fetch() .....	160
13.15. Usando fetch() y enviando a un e-mail .....	161
13.16. get_config_vars() .....	162
13.17. get_registered_object() .....	163
13.18. get_template_vars .....	164
13.19. is_cached() .....	165
13.20. is_cached() con templates con multiple-cache .....	165
13.21. loading filter plugins .....	167
13.22. register_block() .....	168
13.23. register_function() .....	170
13.24. register_modifier() .....	172
13.25. register_resource .....	177
13.26. template_exists() .....	179
13.27. unregister_function() .....	182
13.28. unregister_modifier() .....	183
13.29. unregister_resource() .....	188
14.1. Habilitando Cache .....	189
14.2. Configurando cache_lifetime por cache .....	190
14.3. Habilitando \$compile_check .....	190
14.4. Usando is_cached() .....	191
14.5. Limpiando el cache .....	191
14.6. Pasando un cache_id para display() .....	192
14.7. Pasando un cache_id para is_cached() .....	193
14.8. Limpando todos los caches para un cache_id en particular .....	193
14.9. Grupos de cache_id .....	194
14.10. Previendo que una saída de plugin de ser cacheada .....	195
14.11. Previendo una pasada entera del template para el cache .....	196
15.1. usando un objeto registrado o atribuido .....	198
15.2. usando un prefiltro prefilter de template .....	199
15.3. Usando un postfilter de template .....	199
15.4. Usando un filtro de salida de template .....	200
15.5. ejemplo usando MySQL como una fuente de cache .....	201
15.6. Usando templates desde \$template_dir .....	202
15.7. usando templates desde cualquier directorio .....	202
15.8. usando templates con rutas de archivos de windows .....	203
15.9. Usando recursos habituales .....	204
15.10. usando la función manipuladora de template por default .....	205
16.1. Función de plugin con salida .....	208

16.2. Función de plugin sin salida .....	209
16.3. Plugin modificador simple .....	210
16.4. Plugin modificador mas complejo .....	211
16.5. Función de bloque .....	212
16.6. Función compiladora simple .....	213
16.7. prefilter plugin .....	214
16.8. postfilter plugin .....	215
16.9. plugin de filtro de salida .....	216
16.10. Plugin resource (recurso) .....	218
16.11. insert plugin .....	219
17.1. Errores de Smarty .....	222
17.2. Errores de analisis gramatical de PHP .....	222
17.3. Otros errores comunes .....	223
18.1. Imprimiendo &nbsp; cuando una variable esta vacia .....	224
18.2. Atribuyendo el valor por default a una variable en el template .....	224
18.3. Pasando la variable titulo al encabezado del template .....	225
18.4. Usando date_format .....	226
18.5. Convirtiendo elementos en forma de fecha de vuelta a un timestamp .....	227
18.6. Usando insert para escribir un encabezado WML Content-Type .....	228
18.7. Templates con Componetes .....	229
18.8. Ejemplo de ofuscamiento de una direccion de E-mail .....	230

---

# Prólogo

Esta es indudablemente una de las preguntas que mas se hacen en las listas de correo de PHP: Como hacer mis scripts de PHP independientes del diseño?. Mientras PHP se encarga de como "incrustar scripts en lenguaje HTML", después de escribir los proyectos que mezclan PHP y HTML libremente, esto trae como consecuencia la idea de separar la forma y el contenido, muy buena idea[TM]. En adición, en muchas compañías la interpretación de esquema es diseñador y programador por separado. Por consiguiente, la busqueda trae como solución una plantilla(template).

Por ejemplo en nuestra compañía, el desarrollo de una aplicación es como sigue: Después de tener la documentación necesaria, el diseñador de web diseña el prototipo de la interfaz y la entrega al programador. El programador implementa las reglas de negocio en PHP y usa el prototipo para crear el "esqueleto" de la plantilla. El proyecto esta en manos de la persona responsable del HTML designer/web page que produzca la plantilla para su gloria completa. El proyecto debe ir y regresar entre programación/HTML varias veces. De esa manera, es importante para tener un buen soporte de templates porque los programadores no quieren hacer nada con HTML ni quieren diseño HTML al rededor del codigo PHP. Los diseñadores precisan de soporte para archivos de configuración, bloques dinámicos y otras interfaces usadas, mas ellos no quieren ocuparse con las complejidades del lenguaje de programación PHP.

Buscando, actualmente existen muchas soluciones de templates disponibles para PHP, la mayor parte de ellos les provee de una forma rudimentaria de sustitución de variables dentro del template y hace una forma limitada de la funcionalidad dinámica del bloque. Pero nuestras necesidades requieren mas que eso. Porque no queremos programadores que no quieran tener trato con HTML del todo, pero esto puede ser casi inevitable. Por ejemplo, si un diseñador quiere alternar colores de fondo sobre bloques dinámicos, esto tuvo que trabajarse con el programador anticipadamente. Nosotros necesitamos también que los diseñadores esten capacitados para usar archivos de configuración, y colocar variables de ellos dentro de los templates. La lista continua.

Nosotros empezamos escribiendo por fuera una especulación para un motor de plantillas(templates) atrasado de 1999. Después de terminar la especulación, comenzamos a trabajar un motor de plantillas escrito en C que esperanzadoramente fue aceptado para ser incorporado con PHP. No solamente nos encontramos con algunas complicadas barreras tecnicas, si no también hubo acalorados debates sobre lo que exactamente debia de hacer o no un motor de plantillas. De esta experiencia, decidimos que un motor de plantillas debería ser escrito en PHP como una clase, para que cualquiera lo use de la misma forma como ellos ven. Así nosotros escribimos un motor que es SmartTemplate™ nunca volvio a existir(nota: esa clase nunca fue enviada al público). Esta era una clase que realizaba casi todo lo que nosotros necesitabamos: sustitución de variables regulares, soporte incluso de otras plantillas, integración con archivos de configuración, incrustación de código PHP, funcionalidades 'if' limitada y muchos mas bloques dinámicos robustos que podrían ser anidados muchas veces. Todo esto con expresiones regulares y el código producido sería mejor, como diriamos nosotros, impenetrable. Eso era también notoriamente lento en grandes aplicaciones por todas las interpretaciones y expresiones regulares trabajando en cada requisición. El mayor problema del punto de vista de un programador era todo el trabajo necesario en el procesamiento del scripts PHP y procesamiento de bloques dinámicos de la plantilla. Como hacemos eso facilmente?

Entonces se origino la visión de que finalmente se convirtiera en Smarty. Nosotros sabemos que rápido es el código PHP sin las cabeceras y la interpretación de plantillas(templates). También sabemos que meticulouso y arrogante es el lenguaje PHP su poder debe ser aceptable para un diseñador, y este podría ser enmascarado con una simples sintaxis de plantillas(templates). Entonces que pasara si nosotros convinamos las dos fuerzas? De esta manera, nacio Smarty...

---

# Part I. Iniciando

---

---

## Table of Contents

1. Que es Smarty? .....	3
2. Instalación .....	5
Requerimientos .....	5
Instalación Básica .....	5
Expandiendo la configuración .....	9

---

# Chapter 1. Que es Smarty?

Smarty es un motor de plantillas para PHP. Mas especificamente, esta herramienta facilita la manera de separar la aplicación lógica y el contenido en la presentación. La mejor descripción esta en una situación donde la aplicación del programador y la plantilla del diseñador juegan diferentes roles, o en la mayoría de los casos no la misma persona.

Por ejemplo: Digamos que usted crea una pagina web, es decir, despliega el articulo de un diario. El encabezado del articulo, el rotulo, el autor y el cuerpo son elementos del contenido, estos no contiene información de como quieren ser presentados. Estos son pasados por la aplicación Smarty, donde el diseñador edita la plantilla, y usa una combinación de etiquetas HTML y etiquetas de plantilla para formatear la presentación de estos elementos (HTML, tablas, color de fondo, tamaño de letras, hojas de estilo, etc...). Un día el programador necesita cambiar la manera de recuperar el contenido del articulo(un cambio en la aplicación lógica.). Este cambio no afectara al diseñador de la plantilla, el contenido llegara a la plantilla exactamente igual. De la misma manera, si el diseñador de la plantilla quiere rediseñarla en su totalidad, estos cambios no afectaran la aplicación lógica. Por lo tanto, el programador puede hacer cambios en la aplicación lógica sin que sea necesario reestructurar la plantilla. Y el diseñador de la plantilla puede hacer cambios sin que haya rompimiento con la aplicación lógica.

One design goal of Smarty is the separation of business logic and presentation logic. This means templates can certainly contain logic under the condition that it is for presentation only. Things such as including other templates, altering table row colors, upper-casing a variable, looping over an array of data and displaying it, etc. are all examples of presentation logic. This does not mean that Smarty forces a separation of business and presentation logic. Smarty has no knowledge of which is which, so placing business logic in the template is your own doing. Also, if you desire *no* logic in your templates you certainly can do so by boiling the content down to text and variables only.

Ahora un pequeño resumen sobre que no hace Smarty. Smarty no intenta separar completamente la lógica de la plantilla. No hay problema entre la lógica y su plantilla bajo la condición que esta lógica sea estrictamente para presentación. Un consejo: mantener la aplicación lógica fuera de la plantilla, y la presentación fuera de la aplicación lógica. Esto tiene como finalidad tener un objeto mas manipulable y escalable para un futuro proximo.

Un único aspecto acerca de Smarty es la compilación de la plantilla. De esta manera Smarty lee la plantilla y crea los scripts de PHP. Una vez creados, son executados sobre él. Por consiguiente no existe ningún costo por analizar gramaticalmente cada archivo de template por cada requisición, y cada template puede llevar toda la ventaja del compilador de cache de PHP tal como Zend Accelerator (<http://www.zend.com/>) o PHP Accelerator (<http://www.php-accelerator.co.uk>).

Algunas de las características de Smarty:

- Es extremadamente rápido.
- Es eficiente ya que puede interpretar el trabajo mas sucio.
- No analiza gramaticalmente desde arriba el template, solo compila una vez.
- El esta atento para solo recompilar los archivos de plantilla que fueron cambiados.
- Usted puede crear funciones habituales y modificadores de variables customizados, de modo que el lenguaje de la platilla es altamente extensible.
- Sintaxis de etiquetas delimitadoras para configuración de la plantilla, así lo puede usar {}, {{}}, <!-- {}-->, etc.

- Los constructores `if/elseif/else/endif` son pasados por el interpretador de PHP, así la sintaxis de la expresión `{if ...}` puede ser compleja o simple de la forma que usted quiera.
- Permite un anidamiento ilimitado de sections, ifs, etc.
- Es posible incrustar directamente código PHP en los archivos de plantilla, aunque esto puede no ser necesario (no recomendado) dado que la herramienta se puede ajustar.
- Soporte de caching incrustado
- Fuentes de Plantilla absoluto
- Funciones habituales de manipulación de cache
- Arquitectura de Plugin



---

# Chapter 2. Instalación

## Requerimientos

Smarty Requiere un servidor web corriendo PHP 4.0.6 o posterior.

## Instalación Básica

Instale los archivos de la librería de Smarty que están en el directorio de distribución /libs/. Estos son los archivos PHP que usted NO EDITARA. Estos archivos son toda las aplicaciones comunes y ellos son actualizados cuando usted actualiza a una nueva versión de Smarty.

### Example 2.1. Archivos de la librería Smarty

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

Smarty utiliza una constante de PHP llamada SMARTY\_DIR que es la ruta para el directorio de la biblioteca de Smarty 'libs/'. Basicamente, si su aplicación puede encontrar el archivo Smarty.class.php , usted no necesita definir SMARTY\_DIR, Smarty lo encontrará. Por consiguiente si, Smarty.class.php no esta incluido en el path, y no es abastecido por una ruta absoluta para encontrar su aplicación, entonces usted debe definir SMARTY\_DIR manualmente. SMARTY\_DIR *debe* incluir una barra de seguimiento.

Aquí esta un ejemplo de como se crea una instancia de Smarty en sus scripts PHP:

### Example 2.2. Creando una instancia Smarty de Smarty

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

Intente correr el script de arriba. Si usted obtiene un error diciendo que el archivo Smarty.class.php no fue encontrado, puedes usar una de las siguientes opciones:

### Example 2.3. Reemplazar por la ruta absoluta de la librería del archivo

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

### Example 2.4. Adicionar el directorio de la librería para incluirlo en el include\_path de PHP

```
<?php
// Edite su archivo php.ini, y adicione el directorio de la librería de Smarty
// include_path y reinicie su servidor web.
// Entonces lo siguiente debe funcionar:
require('Smarty.class.php');
$smarty = new Smarty;
?>
```

### Example 2.5. Defina la constante SMARTY\_DIR manualmente

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

Ahora que la librería de archivos está en su sitio, es tiempo de configurar los directorios de Smarty para su aplicación.

Smarty requiere cuatro directorios (por default) llamados 'templates/', 'templates\_c/', 'configs/' y 'cache/'.

Cada uno de estos son para definir las propiedades de las clases de Smarty. `$template_dir`, `$compile_dir`, `$config_dir`, y `$cache_dir` respectivamente. Es altamente recomendado que usted configure un grupo separado de estos directorios para cada aplicación que utilice de Smarty.

Asegure que usted sabe la ubicación del document root de su servidor web. En nuestro ejemplo, el document root está en `/web/www.example.com/docs/`. Los directorios de Smarty solo son accesados por la librería de Smarty y nunca son accesados directamente por el navegador. Por consiguiente para evitar cualquier preocupación con la seguridad, es recomendado colocar estos directorios *fuera* del document root.

Para nuestro ejemplo de instalación, configuraremos el ambiente de Smarty para una aplicación de libro de visitas. Escojemos una aplicación solo con el proposito de crear un directorio de nombre convencional. Usted puede usar el mismo ambiente para cualquier aplicación, solamente sustituya "guestbook" con el nombre de su aplicación. Nosotros colocaremos nuestros directorios de Smarty dentro de `/web/www.example.com/smarty/guestbook/`.

Usted necesita tener por lo menos un archivo dentro de su document root, y que sea accesado por el navegador. Nosotros llamamos el script de `'index.php'`, y lo colocamos en un subdirectorio dentro del document root llamado `/guestbook/`.

### Nota Técnica:

Es conveniente configurar el servidor de forma que "index.php" pueda ser identificado como el índice del directorio padre, de esta manera si usted accesa `http://www.example.com/guestbook/`, el script `index.php` será ejecutado sin "index.php" ni la URL. En Apache usted puede definir el sitio adicionando "index.php" en el final de su configuración del directorio *DirectoryIndex* (separando cada uno con espacios.) como en el ejemplo de `httpd.conf`.

*DirectoryIndex index.htm index.html index.php index.php3 default.html index.cgi*

Veamos nuestra estructura de archivos hasta hora:

### Example 2.6. Ejemplo de estructura de archivo

```
/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/internals/*.php
/usr/local/lib/php/Smarty/plugins/*.php

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

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

Smarty necesitara **permisos de escritura** (usuarios de windows ingnorar) para `$compile_dir` y `$cache_dir`, esto garantiza que el usuario del servidor pueda escribir en ellos. Este es generalmente el usuarios "nobody" y el grupo "nobody". Para usuarios con X sistema operativo, el default es "www" y el grupo "www". Si usted esta usando Apache, puede ver en su archivo `httpd.conf` (normalmente en `"/usr/local/apache/conf/"`) cual es el usuario y grupo que estan siendo usados.

## Example 2.7. Configurando permisos de archivos

```
chown nobody:nobody /web/www.example.com/smarty/guestbook/templates_c/  
chmod 770 /web/www.example.com/smarty/guestbook/templates_c/
```

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

### Nota Técnica:

chmod 770 puede ser una seguridad bastante fuerte, solo le permite al usuario "nobody" y al grupo "nobody" acceso de lectura/escritura a los directorios. Si usted quiere abrir permiso de lectura a cualquiera (en la mayoría de las veces para su propia conveniencia de querer ver estos archivos), usted puede usar el 775 en lugar del 770.

Nosotros necesitamos crear el archivo index.tpl, para que Smarty lo pueda cargar. Este estara localizado en su \$template\_dir.

## Example 2.8. Editando /web/www.example.com/smarty/guestbook/templates/index.tpl

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

### Nota Técnica:

{\* Smarty \*} Esto es un comentario en el template. Este no es obligatorio, pero si una buena practica iniciar todos sus archivos de plantilla con estos comentarios. Esto hace facilmente reconocibles a los archivos a pesar la extensión del archivo. Por ejemplo, editores de texto pueden reconocer el archivo y habilitar un realce de sintaxis especial.

Ahora vamos a editar el index.php. crearemos una instancia de Smarty, daremos valor a las variables del template y mostraremos el archivo index.tpl. En el ambiente de nuestro ejemplo, "/usr/local/lib/php/Smarty" esta dentro de include\_path. Asegurese que exista el mismo, o utilice la ruta absoluta.

## Example 2.9. Editando /web/www.example.com/docs/guestbook/index.php

```
<?php

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

$smarty = new Smarty;

$smarty->template_dir = '/web/www.example.com/smarty/guestbook/templates/';
$smarty->compile_dir = '/web/www.example.com/smarty/guestbook/templates_c/';
$smarty->config_dir = '/web/www.example.com/smarty/guestbook/configs/';
$smarty->cache_dir = '/web/www.example.com/smarty/guestbook/cache/';

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

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

### Nota Técnica:

En nuestro ejemplo, estamos configurando rutas absolutas para todos los directorios de Smarty. Si `/web/www.example.com/smarty/guestbook/` está dentro de su `include_path` de PHP, entonces estas declaraciones no son necesarias. Sin embargo, esto es más eficiente y (por experiencia) tiene menos tendencia a errores en relación a determinar las rutas absolutas. Esto garantiza que Smarty está recibiendo los archivos del directorio que usted desea.

Ahora cargue el archivo `index.php` desde su navegador web. Usted debería ver "Hello, Ned!"

Usted ha completado la configuración básica para el Smarty!

## Expandiendo la configuración

Esta es una continuación de la instalación básica, por favor lea esta primero!

Una forma un poco más flexible de configurar el Smarty, expandir las clases e iniciar su ambiente de Smarty. Es, en vez de configurar rutas de directorios repetidamente, asigne esas mismas a variables, etc., nosotros podemos facilitar eso. Vamos a crear un nuevo directorio en `"/php/includes/guestbook/"` y llamemos al nuevo archivo `setup.php`. En nuestro ejemplo, `"/php/includes"` está en nuestro `include_path`. Verifique que usted también lo defina, o utilice rutas absolutas de los archivos.

**Example 2.10. Editando /php/includes/guestbook/setup.php**

```
<?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');

class Smarty_GuestBook extends Smarty {

    function Smarty_GuestBook()
    {

        // Class Constructor.
        // These automatically get set with each new instance.

        $this->Smarty();

        $this->template_dir = '/web/www.example.com/smarty/guestbook/templates/';
        $this->compile_dir  = '/web/www.example.com/smarty/guestbook/templates_c/';
        $this->config_dir   = '/web/www.example.com/smarty/guestbook/configs/';
        $this->cache_dir    = '/web/www.example.com/smarty/guestbook/cache/';

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

Ahora vamos a modificar el archivo index.php para usar el setup.php:

### **Example 2.11. Editando /web/www.example.com/docs/guestbook/index.php**

```
<?php  
  
require('guestbook/setup.php');  
  
$smarty = new Smarty_GuestBook;  
  
$smarty->assign('name', 'Ned');  
  
$smarty->display('index.tpl');  
?>
```

Ahora usted vera que es completamente simple crear una instancia de Smarty, solo use `Smarty_GuestBook`, que automáticamente inicializa todo para nuestra aplicación.

---

# Part II. Smarty For Template Designers

---



---

## Table of Contents

3. Basic Syntax .....	15
Comentarios .....	15
Variables .....	15
Funciones .....	16
Atributos .....	17
Colocando variables entre comillas dobles .....	17
Matemáticas .....	18
Escaping Smarty Parsing .....	18
4. Variables .....	20
Variables definidas desde PHP .....	20
Arreglos asociativos .....	21
Índices de Matrices .....	22
Objects .....	23
Variables cargadas desde archivos de configuración .....	24
La variable reservada {Smarty} .....	26
Solicitud de Variables .....	26
{Smarty.now} .....	26
{Smarty.const} .....	27
{Smarty.capture} .....	27
{Smarty.config} .....	27
{Smarty.section}, {Smarty.foreach} .....	27
{Smarty.template} .....	27
{Smarty.version} .....	27
{Smarty.lldelim}, {Smarty.rdelim} .....	27
5. Modificadores de variables .....	28
capitalize .....	29
cat .....	29
count_characters .....	30
count_paragraphs .....	31
count_sentences .....	32
count_words .....	33
date_format .....	34
default .....	37
escape .....	38
indent .....	39
lower .....	41
nl2br .....	41
regex_replace .....	42
replace .....	43
spacify .....	44
string_format .....	45
strip .....	46
strip_tags .....	47
truncate .....	48
upper .....	50
wordwrap .....	51
6. Combinando Modificadores .....	53
7. Funciones Integradas .....	54
capture .....	54
config_load .....	55
{foreach},{foreachelse} .....	58

iteration .....	61
first .....	61
last .....	61
show .....	61
total .....	61
include .....	62
{include_php} .....	64
insert .....	65
if,elseif,else .....	66
{ldelim},{rdelim} .....	69
literal .....	69
{php} .....	70
section,sectionelse .....	71
index .....	78
index_prev .....	79
index_next .....	79
iteration .....	80
first .....	82
last .....	82
rownum .....	82
loop .....	82
show .....	83
total .....	84
{strip} .....	85
8. Custom Functions .....	87
{assign} .....	87
{counter} .....	88
cycle .....	89
{debug} .....	91
{eval} .....	91
{fetch} .....	93
{html_checkboxes} .....	94
{html_image} .....	96
{html_options} .....	97
{html_radios} .....	100
{html_select_date} .....	103
{html_select_time} .....	108
{html_table} .....	112
math .....	114
{mailto} .....	117
{popup_init} .....	118
popup .....	119
{textformat} .....	124
9. Config Files .....	128
10. Debugging Console .....	130

---

# Chapter 3. Basic Syntax

Todas las etiquetas del template deben estar marcadas por delimitadores. Por default , estos delimitadores son { y }, sino estos pueden cambiar.

Para estos ejemplos, nosotros asumiremos que usted está usando los delimitadores por default. En Smarty, todo el contenido fuera de los delimitadores es mostrado como contenido estatico, o igual(sin cambios). Cuando Smarty encuentra etiquetas en el template, trata de interpretarlos, e intenta mostrar la salida apropiada en su lugar.

## Comentarios

Los comentarios en los templates son cercados por asteriscos, y por los delimitadores, así: { \* este es un comentario \* }. Los comentarios en Smarty no son mostrados en la salida final del template. semejantes a <!-- HTML comments --> Estos son usados para hacer notas internas dentro del template.

### Example 3.1. Comentarios

```
<body>
{ * this multiline
  comment is
  not sent to browser
 * }

{ * include the header file here * }
{include file="header.tpl" }

{ * Dev note:  $includeFile is assigned foo.php script  * }
<!-- this html comment is sent to browser -->
{include file=$includeFile}

{include file=#includeFile#}

{ * display dropdown lists * }
<select name="company">
  {html_options options=$vals selected=$selected_id}
</select>
</body>
```

## Variables

Las variable de Template que comiencen con signo de pesos. Pueden contener números, letras y guiones bajos, muy parecido a las variables de PHP [<http://php.net/language.variables>]. Usted también puede hacer referencia a arreglos que pueden ser numericos o no-numericos. También puede hacer referencia a métodos y propiedades de objetos. Config file variables es una excepción de la sintaxis del signo de pesos. También puede ser referenciado entre #signos de numeros#, o con la variable especial \$smarty.config.

### Example 3.2. Variables

```
{ $foo }           <-- displaying a simple variable (non array/object)
{ $foo[4] }        <-- display the 5th element of a zero-indexed array
{ $foo.bar }       <-- display the "bar" key value of an array, similar to PHP $foo['ba
{ $foo.$bar }      <-- display variable key value of an array, similar to PHP $foo[$bar
{ $foo->bar }       <-- display the object property "bar"
{ $foo->bar() }     <-- display the return value of object method "bar"
{ #foo# }          <-- display the config file variable "foo"
{ $smarty.config.foo } <-- synonym for { #foo# }
{ $foo[bar] }      <-- syntax only valid in a section loop, see {section}
{ assign var=foo value="baa" } { $foo } <-- displays "baa", see {assign}
```

Many other combinations are allowed

```
{ $foo.bar.baz }
{ $foo.$bar.$baz }
{ $foo[4].baz }
{ $foo[4].$baz }
{ $foo.bar.baz[4] }
{ $foo->bar($baz,2,$bar) } <-- passing parameters
{ "foo" }                 <-- static values are allowed
```

Vea también \$smarty reserved variables y Config Variables.

## Funciones

Cada etiqueta Smarty muestra una variable o utiliza algún tipo de función. Las funciones son procesadas y mostradas colocando los atributos de la función entre delimitadores, así: {funcname attr1="val" attr2="val"}.

### Example 3.3. Sintaxis de Funciones

```
{config_load file="colors.conf"}

{include file="header.tpl"}

{if $highlight_name}
    Welcome, <font color="{#fontColor#}">{ $name }!</font>
{else}
    Welcome, { $name }!
{/if}

{include file="footer.tpl"}
```

Las funciones internas y las funciones habituales, ambas deben tener la misma sintaxis dentro del template. Las funciones **internas** que funcionan en Smarty, son: {if}, {section} y {strip}. Estas no pueden ser modificadas. Las funciones habituales son funciones **adicionales** implementadas por plugins. Estas si pueden ser modificadas como usted quiera, o usted también puede adicionar nuevas. {html\_options} y {popup} son ejemplos de funciones habituales.

## Atributos

La mayoría de las funciones llevan atributos que especifican o cambian su funcionamiento. Los atributos para las funciones de Smarty son muy parecidos a los atributos de HTML. Los valores estaticos no necesitan estar entre comillas, pero si es recomendado para cadenas y literales. Las variables también pueden ser usadas y no precisamente estando entre comillas.

Algunos atributos requieren valores booleanos(true o false). Estos pueden ser especificados como cualquier otro valor sin comillas true, on, y yes, o false, off, y no.

### Example 3.4. Sintaxis de atributos de Funciones

```
{include file="header.tpl"}

{include file="header.tpl" attrib_name="attrib value"}

{include file=$includeFile}

{include file=#includeFile# title="Smarty is cool"}

{html_select_date display_days=yes}

<select name="company">
  {html_options options=$choices selected=$selected}
</select>
```

## Colocando variables entre comillas dobles

Smarty puede reconocer variables asignadas entre comillas aunque estas solo tengan números, letras, guiones bajos y corchetes[]. Con cualquier otro carácter(puntos, referencia de objetos, etc.) las variables deben estar entre apostrofos. Usted no puede incrustar modificadores, Estos deben ser siempre aplicados fuera de las comillas.

### Example 3.5. Sintaxis entre comillas

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 $foo.bar)
{func var="test ` $foo.bar ` test"}  <-- sees $foo.bar
{func var="test ` $foo.bar ` test"|escape} <-- modifiers outside quotes!
```

PRACTICAL EXAMPLES:

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

Ver también `escape`.

## Matemáticas

Las matemáticas pueden ser aplicadas directamente al los valores de las variables.

### Example 3.6. Ejemplos de matemáticas

```
{ $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`" }
```

Ver también la función `{math}`.

## Escaping Smarty Parsing

En algunas ocasiones es deseable o hasta necesario que Smarty tenga que ignorar sections o algún otro tipo análisis de sintaxis. Un ejemplo clásico es con el código JavaScript o CSS incrustado en el template. El problema se origina cuando aquellos lenguajes que utilizan los caracteres `{ }` los cuales son también los delimitadores por default para Smarty.

Esta puede ser una simple situación separando enteramente su código JavaScript y CSS dentro de un archivo personal y utilizar el método estándar del HTML para el acceso.

Es posible usar literal incluyendo el contenido del bloque {literal} .. {/literal}. Similar a usar entidades HTML, usted puede usar {ldelim},{rdelim} o {\$smarty.ldelim} para mostrar los delimitadores actuales.

Esto a menudo es conveniente para cambios simples a Smarty's \$left\_delimiter y \$right\_delimiter.

### **Example 3.7. Ejemplo cambiando delimitadores**

```
<?php

$smarty = new Smarty;
$smarty->left_delimiter = '<!--{';
$smarty->right_delimiter = '}->';
$smarty->assign('foo', 'bar');
$smarty->display('example.tpl');

?>
```

Donde example.tpl es:

```
<script language="javascript">
var foo = <!--{$foo}->;
function dosomething() {
    alert("foo is " + foo);
}
dosomething();
</script>
```

Ver También [escape modifier](#)

---

# Chapter 4. Variables

Smarty tiene varios tipos diferentes de variables. El tipo de variable depende de cual simbolo este prefijado(incluido dentro).

Las variables de Smarty no pueden ser mostradas directamente o usadas como argumentos para atributos, funciones y modificadores, dentro de expresiones condicionales, etc. Para mostrar una variable, simplemente coloque esta entre delimitadores siendo esta la única cosa entre ellos. Ejemplos:

```
{ $Name }
```

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

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

## Variables definidas desde PHP

Las variables que son asignadas desde PHP son referenciadas precedidas estas con una señal de cifrado \$. Las variables definidas dentro del template como una función assign también son mostradas de esta manera.



### Example 4.1. variables definidas

php script

```
<?php

$smarty = new Smarty;

$smarty->assign('firstname', 'Doug');
$smarty->assign('lastname', 'Evans');
$smarty->assign('meetingPlace', 'New York');

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

?>
```

Donde el contenido de index.tpl es:

```
Hello {$firstname} {$lastname}, glad to see you can make it.
<br />
{* this will not work as $vars are case sensitive *}
This weeks meeting is in {$meetingplace}.
{* this will work *}
This weeks meeting is in {$meetingPlace}.
```

esta es la salida:

```
Hello Doug Evans, glad to see you can make it.
<br />
This weeks meeting is in .
This weeks meeting is in New York.
```

## Arreglos asociativos

Usted también puede referenciar matrices asociativas en variables que son definidas desde PHP especificando la clave después del simbolo '.'(punto).

### Example 4.2. Accesando variables de matriz asociativa

```
<?php
$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');
?>
```

Donde el contenido de index.tpl es:

```
{ $Contacts.fax }<br />
{ $Contacts.email }<br />
{ * you can print arrays of arrays as well *}
{ $Contacts.phone.home }<br />
{ $Contacts.phone.cell }<br />
```

esta es la salida:

```
555-222-9876<br />
zaphod@slartibartfast.com<br />
555-444-3333<br />
555-111-1234<br />
```

## Índices de Matrices

Usted podrá referencia matrices por su índice, muy semejantes a la sintaxis de PHP.

### Example 4.3. Accesando matrices por sus índices

```
<?php
$smarty->assign('Contacts', array(
    '555-222-9876',
    'zaphod@slartibartfast.com',
    array('555-444-3333',
        '555-111-1234')
    ));
$smarty->display('index.tpl');
?>
```

Donde index.tpl es:

```
{ $Contacts[0] }<br />
{ $Contacts[1] }<br />
{ * you can print arrays of arrays as well *}
{ $Contacts[2][0] }<br />
{ $Contacts[2][1] }<br />
```

esta es la salida:

```
555-222-9876<br />
zaphod@slartibartfast.com<br />
555-444-3333<br />
555-111-1234<br />
```

## Objects

Las propiedades de los objetos definidos desde PHP pueden ser referenciados especificando el nombre de la propiedad después del símbolo '->'.

### Example 4.4. Accesando propiedades de los Objetos

```
name: {$person->name}<br />
email: {$person->email}<br />
```

esta es la salida:

```
name: Zaphod Beeblebrox<br />
email: zaphod@slartibartfast.com<br />
```

## Variables cargadas desde archivos de configuración

Las variables que son cargadas de archivos de configuración son referenciadas incluyendo entre ellas el signo(#), o como variables de Smarty \$smarty.config. La segunda sintaxis es util para incrustar valores de un atributo dentro de comillas.

**Example 4.5. Variables de configuración**

```
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: (sintaxis alternativa)

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

esta es la salida de ambos ejemplos:

```
<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>
```

Las variables de un archivo de configuración no pueden ser usadas hasta después de que son cargadas por los archivos de configuración. Este procedimiento es explicado posteriormente en este documento en `{config_load}`.

Ver también `Variables` y `$smarty reserved variables`

## La variable reservada `{$smarty}`

La variable reservada `{$smarty}` puede ser utilizada para acceder a variables especiales del template. A continuación una lista completa.

## Solicitud de Variables

La solicitud de variables [<http://php.net/reserved.variables>] como `$_GET`, `$_POST`, `$_COOKIE`, `$_SERVER`, `$_ENV` y `$_SESSION` (Ver `$request_vars_order` y `$request_use_auto_globals`) pueden ser accedidas como se muestra en los ejemplos de abajo:

### Example 4.6. Mostrando solicitud de variables

```
{* display value of page from URL (GET) http://www.domain.com/index.php?page=foo *}
{$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/cookies/server/env *}
{$smarty.request.username}
```

### Note

Por historicas razones `{$SCRIPT_NAME}` puede ser accedido directamente sin embargo `{$smarty.server.SCRIPT_NAME}` es el sugerido para acceder este valor.

## `{$smarty.now}`

El timestamp [<http://php.net/function.time>] actual puede ser accedido con `{$smarty.now}`. El número refleja el número de segundos pasados desde la llamada Epoca (1 de Enero de 1970) y puede ser pasado directamente para el modificador `date_format` para mostrar la fecha.

### Example 4.7. Usando `{Smarty.now}`

```
{* utilice el modificador date_format para mostrar la fecha y hora actual *}
{Smarty.now|date_format:"%Y-%m-%d %H:%M:%S"}
```

## `{Smarty.const}`

Usted puede acceder al valor de constantes PHP directamente. Ver también smarty constants

### Example 4.8. Usando `{Smarty.const}`

```
{Smarty.const._MY_CONST_VAL}
```

## `{Smarty.capture}`

La salida capturada via `{capture}..{/capture}` puede ser accesada usando la variable `{Smarty}`. vea la sección `{capture}` para un ejemplo.

## `{Smarty.config}`

La variable `{Smarty}` puede ser usada para referir variables de configuración cargadas. `{Smarty.config.foo}` es un sinónimo para `{#foo#}`. vea la sección sobre `{config_load}` para un ejemplo.

## `{Smarty.section}`, `{Smarty.foreach}`

La variable `{Smarty}` puede ser usada para hacer referencia a las propiedades 'section' y 'foreach' del loop. Ver la documentación sobre section y foreach.

## `{Smarty.template}`

Esta variable contiene el nombre actual del template que esta siendo procesado.

## `{Smarty.version}`

Esta variable contiene la versión Smarty con que es compilado el template.

## `{Smarty.ldelim}`, `{Smarty.rdelim}`

Esta variable es usada para imprimir literalmente el valor left-delimiter y right-delimiter. Ver tambien `{ldelim}`,`{rdelim}`.

Ver también Variables y Config Variables

---

# Chapter 5. Modificadores de variables

Los modificadores de variables pueden ser aplicados a variables, funciones habituales o cadenas. Para aplicar un modificador, especifique el valor seguido por | (pipe) y el nombre del modificador. Un modificador necesita parámetros adicionales que afectan en su funcionamiento. Estos parámetros siguen al nombre del modificador y son separados por : (dos puntos).

## Example 5.1. Ejemplo de modificador

```
{* apply modifier to a variable *}
{$title|upper}
{* modifier with parameters *}
{$title|truncate:40:"..."}

{* apply modifier to a function parameter *}
{html_table loop=$myvar|upper}
{* with parameters *}
{html_table loop=$myvar|truncate:40:"..."}

{* apply modifier to literal string *}
{"foobar"|upper}

{* using date_format to format the current date *}
{$smarty.now|date_format:"%Y/%m/%d"}

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

Si usted aplica un modificador a una matriz en lugar del valor de una variable, el modificador va a ser aplicado en cada uno de los valores de la matriz. Si usted realmente quisiera que el modificador funcionara en una matriz entera, debe colocar el símbolo @ antes del nombre del modificador, así como: {\$articleTitle|@count} (esto mostrara el número de elementos de la matriz \$articleTitle.)

Los modificadores pueden ser cargados automáticamente a partir de su \$plugins\_dir (vea también: Naming Conventions) o pueden ser registrados explícitamente (vea: register\_modifier). Adicionalmente, todas las funciones de php pueden ser utilizadas como modificadores implícitamente. (El ejemplo @count de arriba usa actualmente la función count de php y no un modificador de Smarty). Usar funciones de php como modificadores tiene dos pequeños problemas: Primero, algunas veces al ordenar los parámetros de una función esto no es aconsejable ({ "%2.f" |sprintf:\$float} actualmente funciona, pero existe algo mas intuitivo Por ejemplo: {\$float|string\_format:"%2.f"} que es proporcionado con la distribución de Smarty). Segundo: con \$security activado, todas las funciones de php que sean utilizadas como modificadores deben ser declaradas como variables de una matriz \$security\_settings['MODIFIER\_FUNCS'] .

Ver también register\_modifier(), register\_function(), Extending Smarty with plugins y modifiers,



## capitalize

Posicion del Parametro	Tipo	Requerido	Default	Descripción
1	boolean	No	false	Este determina que palabra con digitos no debe ser convertida

Este es usado para convertir a mayuscula la primera letra de todas la palabras de una variable.

### Example 5.2. capitalize

```
<?php
$smarty->assign('articleTitle', 'next x-men film, x3, delayed. ');
?>
```

Donde el template es:

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

Esta es la salida:

```
next x-men film, x3, delayed.
Next X-Men Film, x3, Delayed.
Next X-Men Film, X3, Delayed.
```

Ver también lower upper

## cat

Posicion del Parametro	Tipo	Requerido	cat	Descripción
1	string	No	<i>empty</i>	Este es el valor para concatenar con la variable dada.

Este valor es concatenado con la variable dada.

### Example 5.3. cat

```
<?php
$smarty->assign('articleTitle', "Psychics predict world didn't end");
?>
```

Donde el template es:

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

Esta es la salida:

```
Psychics predict world didn't end yesterday.
```

## count\_characters

Posición del Parametro	Tipo	Requerido	Default	Descripción
1	boolean	No	false	Este determina cuando incluir o no los espacios en blanco al contar.

Este es usado para contar el número de caracteres en una variable.

### Example 5.4. count\_characters

```
<?php
$smarty->assign('articleTitle', 'Cold Wave Linked to Temperatures. ');
?>
```

Donde el template es:

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

esta es la salida:

```
Cold Wave Linked to Temperatures.
29
33
```

ver también count\_words, count\_sentences y count\_paragraphs.

## count\_paragraphs

Este es usado para contar el número de párrafos en la variable.

### Example 5.5. count\_paragraphs

```
<?php
$smarty->assign('articleTitle',
               "War Dims Hope for Peace. Child's Death Ruins Couple's Holiday.\n
               Man is Fatally Slain. Death Causes Loneliness, Feeling of Isolati
               );
?>
```

Donde el template es:

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

Esta es la salida:

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

ver también `count_characters`, `count_sentences` y `count_words`.

## count\_sentences

Este es usado para contar el número de frases en la variable.

### Example 5.6. count\_sentences

```
<?php
$smarty->assign('articleTitle',
               'Two Soviet Ships Collide - One Dies.
               Enraged Cow Injures Farmer with Axe.'
               );
?>
```

Donde el template es:

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

esta es la salida:

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

ver también count\_characters, count\_paragraphs y count\_words.

## count\_words

Este es usado para contar el número de palabras en la variable.

### Example 5.7. count\_words

```
<?php
$smarty->assign('articleTitle', 'Dealers Will Hear Car Talk at Noon. ');
?>
```

Donde el template es:

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

esta es la salida:

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

ver también `count_characters`, `count_paragraphs` y `count_sentences`.

## date\_format

Posición del Parametro	Tipo	Requerido	Default	Descripción
1	string	No	%b %e, %Y	Este es el formato para la fecha mostrada.
2	string	No	n/a	Este es el default de la fecha si el valor de entrada es vacío.

Estos formatos de fecha y hora están dentro del formato determinado `strftime()` [<http://php.net/strftime>]. Las fechas pueden ser pasadas a Smarty como timestamps [<http://php.net/function.time>] `unix`, `timestamps mysql`, o como cualquier cadena compuesta de mes día año (pasada por `strtotime()` [<http://php.net/strtotime>]). El diseñador puede usar entonces `date_format` para tener un control completo del formateo de la fecha. Si la fecha pasada para **date\_format** estuviera vacía y un segundo parámetro fuera pasado, este será usado como la fecha a formatear.

### Example 5.8. date\_format

```
<?php  
  
$smarty->assign('yesterday', strtotime('-1 day'));  
  
?>
```

Where template is (uses \$smarty.now):

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

This will output:

```
Feb 6, 2001  
02/06/01  
02:33 pm  
Feb 5, 2001  
Monday, February 5, 2001  
14:33:00
```

**date\_format** especificadores de conversión:

- %a - nombre del día de la semana abreviado de acuerdo al local actual
- %A - nombre del día de la semana anterior de acuerdo al local actual
- %b - nombre del mes abreviado de acuerdo al local actual
- %B - nombre del mes anterior de acuerdo al local actual
- %c - Representación preferencial de la fecha y hora local actual
- %C - año con dos dígitos (o año dividido por 100 y truncado para un entero, intervalo de 00 a 99)
- %d - día del mes como un número decimal (intervalo de 00 a 31)
- %D - Lo mismo que %m/%d/%y
- %e - Día del mes como un número decimal, un único dígito y precedido por un espacio (intervalo de 1 a 31)

- %g - Año basado en la semana, sin el siglo [00,99]
- %G - Año basado en la semana, incluyendo el siglo [0000,9999]
- %h - Lo mismo que %b
- %H - Hora como un número decimal usando un reloj de 24 horas (intervalo de 00 a 23)
- %I - Hora como un número decimal usando un reloj de 12 horas (intervalo de 01 a 12)
- %j - Día del año como un número decimal (intervalo de 001 a 366)
- %k - Hora (reloj de 24 horas) dígitos únicos que son precedidos por un espacio en blanco (intervalo de 0 a 23)
- %l - Hora como un número decimal usando un reloj de 12 horas, dígitos únicos son precedidos por un espacio en blanco (intervalo de 1 a 12)
- %m - Mes como número decimal (intervalo de 01 a 12)
- %M - Minuto como un número decimal
- %n - Caracter de nueva línea
- %p - Cualquiera `am' o `pm' de acuerdo con el valor de la hora dado, o la cadena correspondiente a la local actual
- %r - Hora con notación a.m. y p.m.
- %R - Hora con notación de 24 horas
- %S - Segundo como número decimal
- %t - Caracter tab
- %T - Hora actual, igual a %H:%M:%S
- %u - Día de la semana como un número decimal [1,7], representando con 1 el lunes
- %U - Número de la semana del año actual como un número decimal, comenzando con el primer domingo como primer día de la primera semana
- %V - Número de la semana del año actual como número decimal de acuerdo con el ISO 8601:1988, intervalo de 01 a 53, en donde 1 es la primera semana que tenga por lo menos cuatro días en el año actual, siendo domingo el primer día de la semana.
- %w - Día de la semana como decimal, siendo domingo 0
- %W - Número de la semana del año actual como número decimal, comenzando con el primer lunes como primer día de la primera semana
- %x - Representación preferida para la fecha local actual sin la hora
- %X - Representación preferida de la hora local actual sin la fecha
- %y - Año como número decimal sin el siglo (intervalo de 00 a 99)
- %Y - Año como número decimal incluyendo el siglo
- %Z - Zona horaria, o nombre, o abreviación



- %% - Un carácter `%'`

## NOTA PARA PROGRAMADORES:

**date\_format** es esencialmente una envoltura para la función `strftime()` [<http://php.net/strftime>] de PHP. Usted debera tener mas o menos especificadores de conversiones disponibles de acuerdo con la función `strftime()` [<http://php.net/strftime>] del sistema operacional en donde PHP fue compilado. Cheque en la pagina del manual de su sistema una lista completa de especificadores validos.

Ver también `$smarty.now`, `php function strftime()` [<http://php.net/strftime>], `{html_select_date}` y `date tips`.

## default

Pocisión del Parametro	Tipo	Requerido	Default	Descripción
1	string	No	<i>empty</i>	Este es el valor por defecto para mostrar una variable que estuviera vacia.

Este es usado para definir un valor por defecto para una variable. Si esta variable estuviera vacia o no estuviera definida, el valor por defecto es mostrado. El valor por defecto es usado como argumento.

### Example 5.9. default

```
<?php
$smarty->assign('articleTitle', 'Dealers Will Hear Car Talk at Noon. ');
?>
```

Donde el template es:

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

Esta es la salida:

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

Ver también Default Variable Handling y Blank Variable Handling.

## escape

Posición del Parametro	Tipo	Requerido	Posibles Valores	Default	Description
1	string	html,htmlall,url,quotes,hex,hexentity,javascript			Este es el formato de escape a utilizar.

Este es usado para escapar html, url, comillas simples para escapar una variable que no este escapada, escapar hex, hexentity o javascript. Por default, la variable html es escapada.

### Example 5.10. escape

```
<?php
$smarty->assign('articleTitle',
               "'Stiff Opposition Expected to Casketless Funeral Plan'"
               );

?>
```

Donde el template es:

```
{ $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"}">{$EmailAddress|escape:"hexentity"}</a>
```

esta es la salida:

```
'Stiff Opposition Expected to Casketless Funeral Plan'
&#039;Stiff Opposition Expected to Casketless Funeral Plan&#039;
&#039;Stiff Opposition Expected to Casketless Funeral Plan&#039;
&#039;Stiff Opposition Expected to Casketless Funeral Plan&#039;
%27Stiff+Opposition+Expected+to+Casketless+Funeral+Plan%27
\'Stiff Opposition Expected to Casketless Funeral Plan\'
<a href="mailto:%62%6f%..snip..%65%74">&#x62;&#x66;&#x62..snip..&#x65;&#x74;</a>
```

Ver también [Escaping Smarty Parsing](#) y [Obfuscating E-mail Addresses](#).

## indent

Posición del Parametro	Tipo	requerido	Default	Descripción
1	integer	No	4	Este define con cuantos caracteres endentar.
2	string	No	(un espacio)	Este define cual carácter va a ser usado para endentar.

Esta endenta una cadena en cada linea, el default es 4. Como parámetro opcional, usted puede especificar el número de caracteres para endentar. Como segundo parámetro opcional, usted puede especificar el carácter que desea usar para endentar. (Use "\t" para tabs.)

### Example 5.11. indent

```
<?php

$smarty->assign('articleTitle',
               'NJ judge to rule on nude beach.
               Sun or rain expected today, dark tonight.
               Statistics show that teen pregnancy drops off significantly after 25.'
               );

?>
```

Donde el template es:

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

esta es la salida:

```
NJ judge to rule on nude beach.
Sun or rain expected today, dark tonight.
Statistics show that teen pregnancy drops off significantly after 25.
```

```
    NJ judge to rule on nude beach.
    Sun or rain expected today, dark tonight.
    Statistics show that teen pregnancy drops off significantly after 25.
```

```
        NJ judge to rule on nude beach.
        Sun or rain expected today, dark tonight.
        Statistics show that teen pregnancy drops off significantly after 25.
```

```
            NJ judge to rule on nude beach.
            Sun or rain expected today, dark tonight.
            Statistics show that teen pregnancy drops off significantly after 25.
```

ver también strip y spacificy.

## lower

Esta es usada para convertir a minúsculas una variable.

### Example 5.12. lower

```
<?php
$smarty->assign('articleTitle', 'Two Convicts Evade Noose, Jury Hung. ');
?>
```

Donde el template es:

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

esta es la salida:

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

ver también upper y Capitalize.

## nl2br

Todos los saltos de línea serán convertidos a etiquetas `<br />` como datos de la variable. Esto equivale a la función `nl2br()` [<http://php.net/nl2br>] de PHP.

**Example 5.13. nl2br**

```
<?php
$smarty->assign('articleTitle',
               "Sun or rain expected\ntoday, dark tonight"
               );
?>
```

Donde el template es:

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

esta debe ser la salida:

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

Ver también `word_wrap`, `count_paragraphs` y `count_sentences`.

## regex\_replace

Posición del Parametro	Tipo	requerido	Default	Descripción
1	string	Si	<i>n/a</i>	Esta es la expresión regular a ser substituida.
2	string	Si	<i>n/a</i>	Esta es la cadena que sustituirá a la expresión regular.

Localiza una expresión regular y la reemplaza en la variable. Use la sintaxis para `preg_replace()` [[http://php.net/preg\\_replace](http://php.net/preg_replace)] del manual de PHP.

**Example 5.14. regex\_replace**

```
<?php
$smarty->assign('articleTitle', "Infertility unlikely to\nbe passed on, experts sa
?>
```

Donde index.tpl es:

```
{* replace each carriage return, tab and new line with a space *}

{$articleTitle}
{$articleTitle|regex_replace:"/[\r\t\n]/" : " "}
```

Esta es la salida:

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

Vea también `replace` y `escape`.

## replace

Posición del Parametro	Tipo	Requerido	Default	Descripción
1	string	Si	<i>n/a</i>	Esta es la cadena a ser substituida.
2	string	Si	<i>n/a</i>	Esta es la cadena que ira a substituir.

Una simple búsqueda y substituir en la variable. Esta es equivalente a la función `str_replace()` [[http://php.net/str\\_replace](http://php.net/str_replace)] de PHP.

**Example 5.15. replace**

```
<?php
$smarty->assign('articleTitle', "Child's Stool Great for Use in Garden.");
?>
```

Donde index.tpl es:

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

Esta es la Salida:

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

ver también `regex_replace` y `escape`.

## spacify

Posición del Parametro	Tipo	Requerido	Default	Descripción
1	string	No	<i>one space</i>	Este se inserta entre cada carácter de la variable.

Inserta un espacio entre cada carácter de una variable. Usted puede opcionalmente pasar un carácter (o una cadena) diferente para insertar.



### Example 5.16. spacyfy

```
<?php
$smarty->assign('articleTitle', 'Something Went Wrong in Jet Crash, Experts Say.')
?>
```

Donde index.tpl es:

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

Esta es la Salida:

```
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 h   ,   E x p e r t
S ^ o ^ m ^ e ^ t ^ h ^ i ^ n ^ g ^ ^ W ^ e ^ n ^ t ^ ^ W ^ r ^ o ^ n ^ g ^ ^ i ^ n ^ ^ J ^ e ^ t ^
```

Ver también wordwrap y nl2br.

## string\_format

Posición del Parametro	Tipo	Requerido	Default	Descripción
1	string	Si	<i>n/a</i>	Este es el formato que debera usar. (sprintf)

Esta es una manera de formatear cadenas, como números decimales y otros. Use la sintaxis de sprintf [<http://php.net/sprintf>] para formatearlo.

### Example 5.17. string\_format

```
<?php
$smarty->assign('number', 23.5787446);
?>
```

Donde index.tpl es:

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

Esta es la Salida:

```
23.5787446
23.58
24
```

Ver también `date_format`.

## strip

Este substituye todos los espacios repetidos, nuevas lineas y tabs por un unico espacio u otra cadena indicada.

### Nota

Si usted quiere substituir bloques de texto de un template use la función `{strip}`.

**Example 5.18. strip**

```
<?php
$smarty = new Smarty;
$smarty->assign('articleTitle', "Grandmother of\neight makes\t      hole in one.");
$smarty->display('index.tpl');
?>
```

Donde index.tpl es:

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

Esta es la Salida:

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

## strip\_tags

Posición del Parametro	Tipo	Requerido	Default	descripción
1	bool	No	true	Este determina cuando las etiquetas seran remplazadas por ' ' o por "

Este retira las etiquetas de marcación, basicamente todo entre < y >.

### Example 5.19. strip\_tags

```
<?php

$smarty->assign('articleTitle', "Blind Woman Gets <font face=\"helvetica\">New
Kidney</font> from Dad she Hasn't Seen in <b>years</b>.");

?>
```

Donde index.tpl es:

```
{ $articleTitle }
{ $articleTitle|strip_tags } { * same as { $articleTitle|strip_tags:true } * }
{ $articleTitle|strip_tags:false }
```

Esta es la Salida:

```
Blind Woman Gets <font face="helvetica">New Kidney</font> from Dad she Hasn't Seen
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.
```

## truncate

Posición del Parametro	Tipo	Requerido	Default	Desdcripción
1	integer	No	80	Este determina para cuantos caracteres truncar.
2	string	No	...	Este es el texto para adicionar si el truncamiento ocurre. La longitud NO se incluye para la logitud del truncamiento
3	boolean	No	false	Este determina cuando truncar o no o al final de una palabra(false),

Posición del Parametro	Tipo	Requerido	Default	Desdcripción
				o un carácter exacto(true).
3	boolean	No	false	Este determina cuando ocurre el truncamiento al final de la cadena(false), o en el centro de la cadena(true). Nota cuando este es true, entonces la palabra limite es ignorada.

Este trunca la variable en una cantidad de cacarteres, el default es 80. Como segundo parámetro opcional, usted puede especificar una cadena para mostrar al final si la variable fue truncada. Los caracteres en la cadena son incluidos tamando el original para el truncamiento. Por default, truncate intentara cortar al final de una palabra. Se usted quisiera cortar una cantidad exacta de caracteres, pase el tercer parámetro, que es opcional, como true.

### Example 5.20. truncate

```
<?php
$smarty->assign('articleTitle', 'Two Sisters Reunite after Eighteen Years at Check
?>
```

Donde index.tpl es:

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

Esta es la Salida:

```
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...
```

## upper

Este es usado para convertir a mayusculas una variable.

### Example 5.21. upper

```
<?php
$smarty->assign('articleTitle', "If Strike isn't Settled Quickly it may Last a Whi
?>
```

Donde index.tpl es:

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

Esta es la Salida:

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

Ver también lower y capitalize.

## wordwrap

Posición del Parametro	Tipo	Requerido	Default	Descripción
1	integer	No	80	Este determina en cuantas columnas cortar.
2	string	No	\n	Esta es la cadena usada para cortar.
3	boolean	No	false	Este determina cuando cortar o no, o al final de una palabra(false), o en un carácter exacto(true).

Este **wordwrap** corta una cadena para un ancho de columna, el default es 80. Como segundo parámetro opcional, usted puede especificar la cadena que será usada para cortar el texto para la próxima línea (el default es un retorno de carro \n). Por default, (wordwrap) intentara cortar al final de una palabra. Si usted quisiera cortar un tamaño exacto de cacarteres, pase al tercer parámetro, que es opcional, como true. Este es equivalente a la función wordwrap() [<http://php.net/wordwrap>] de PHP.

### Example 5.22. wordwrap

```
<?php
$smarty->assign('articleTitle', "Blind woman gets new kidney from dad she hasn't s
?>
```

Donde index.tpl es:

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

Esta es la Salida:

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<br />  
from dad she hasn't seen in<br />  
years.

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

Ver También nl2br y {textformat}.



---

# Chapter 6. Combinando Modificadores

Usted puede aplicar cualquier cantidad de modificadores para una variable. Estos seran aplicados en el orden en el que fueron combinados, de izquierda a derecha. Estos deben ser separados con el carácter | (pipe).

## Example 6.1. Combinando Modificadores

```
<?php
$smarty->assign('articleTitle', 'Smokers are Productive, but Death Cuts Efficiency
?>
```

Donde el template es:

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

La salida del ejemplode arriba:

```
Smokers are Productive, but Death Cuts Efficiency.
S M O K E R S   A R ....snip....   H   C U T S   E F F I C I E N C Y .
s m o k e r s   a r ....snip....   b u t   d e a t h   c u t s...
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 . . .
```

---

# Chapter 7. Funciones Integradas

Smarty cuenta con varias funciones integradas. Las funciones Integradas forman parte del lenguaje del template. Usted no puede crear funciones personalizadas con el mismo nombre, ni puede modificar las funciones integradas.

## capture

Nombre del Atributo	Tipo	Requerido	Default	Descripción
name	string	no	<i>default</i>	El nombre del bloque capturado
assign	string	No	<i>n/a</i>	El nombre de la variable para dar valor a la salida capturada

{capture} es usado para recolectar toda la salida del template en una variable en lugar de mostrarla. Cualquier contenido entre {capture name="foo"} y {/capture} es recolectado en una variable especificada y el atributo name. El contenido capturado puede ser usado en el template a partir de la variable especial \$smarty.capture.foo en donde foo es el valor pasado para el atributo name. Si usted no pasa un atributo name, entonces será usado "default". Todos los comandos {capture} deben estar entre {/capture}. Usted puede anidar(colocar uno dentro de otro) comandos capture.

### Nota Tecnica

Smarty 1.4.0 - 1.4.4 coloca el contenido capturado dentro de la variable llamada \$return. A partir de 1.4.5, este funcionamiento fue cambiado para usar el atributo name, entonces en consecuencia actualice sus templates.

### Caution

Tenga cuidado cuando capture la salida del comando {insert}. Si tuviera activo el cache y tuviera comandos {insert} y usted espera que funcione con contenido de cache, no se capturara este contenido.

### Example 7.1. capturando contenido de template

```
{* no queremos imprimir la fila de la tabla a menos que exista
    contenido para desplegar *}
{capture name=banner}
{include file="get_banner.tpl"}
{/capture}
{if $smarty.capture.banner ne ""}
<tr>
  <td>
    {$smarty.capture.banner}
  </td>
</tr>
{/if}
```

Ver También `$smarty.capture`, `{eval}`, `{fetch}`, `fetch()` y `{assign}`.

## config\_load

Nombre del Atributo	Tipo	Requerido	Default	Descripción
file	string	Si	<i>n/a</i>	El nombre del archivo de configuración a incluir
section	string	No	<i>n/a</i>	El nombre de la sección a cargar
scope	string	no	<i>local</i>	Como el scope carga las variables debe ser tratado de manera local, como padre y no como global. <i>local</i> indica que las variables son cargadas en el contexto del template local. <i>parent</i> indica que las variables son cargadas en el contexto actual y en el template que llamo. <i>global</i> indica que las variables estan disponibles para todos los templates.
global	boolean	No	<i>No</i>	Cuando las variables no son

Nombre del Atributo	Tipo	Requerido	Default	Descripción
				vistas en el template padre (al que llamo este), lo mismo que scope=parent. NOTA: este atributo esta obsoleto pero el atributo scope, puede dar el soporte. Si scope es el indicado, este valor es ignorado.

Esta función es usada para cargar las #variables# de un archivo de configuración dentro de un template. Vea Config Files para mayor información.

**Example 7.2. Función {config\_load}**

ejemplo.conf

```
#this is config file comment

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

#customer variables section
[Customer]
pageTitle = "Customer Info"
```

y el template

```
{config_load file="example.conf"}

<html>
  <title>{#pageTitle#|default:"No title"}</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>
```

Los archivos de configuración pueden contener secciones también. Usted puede cargar variables de una sección adicionando el atributo *'section'*.

**Note**

*Config file sections* es la función integrada de template *{section}* no tiene nada que ver uno con el otro, ellos justamente por casualidad tiene en común el convencionalismo del nombre.

### Example 7.3. Función `config_load` con `section`

```
{config_load file="ejemplo.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>
```

Vea también `Config files`, `Config variables`, `$config_dir`, `get_config_vars()` y `config_load()`.

## {foreach},{foreachelse}

Nombre del Atributo	Tipo	Requerido	Default	Descripción
from	array	Si	<i>n/a</i>	El nombre de la matriz a la que usted estara pegando los elementos
item	string	Si	<i>n/a</i>	El nombre de la variable que es el elemento actual
key	string	No	<i>n/a</i>	El nombre de la variable que es la llave actual
name	string	No	<i>n/a</i>	El nombre del ciclo <code>foreach</code> para acessar a las propiedades del <code>foreach</code>

Los ciclos(loop) *foreach* son una alternativa para `loop {section}`. *foreach* es usado para pegar cada elemento de una **matriz asociativa simple**. La sintaxis para *foreach* es mucho mas simple que *section*, pero tiene una desventaja de que solo puede ser usada en una única matriz. La etiqueta *foreach* debe tener su par */foreach*. Los parámetros requeridos son *from* e *item*. El nombre del ciclo(loop) `foreach` puede ser cualquier cosa que usted quiera, hecho de letras, números y subrayados. Los ciclos(loop) *foreach* pueden ser anidados, y el nombre de los ciclos(loop) anidados debe ser diferente uno de otro. La variable *from* (normalmente una matriz de valores) determina el número de veces del ciclo(loop) *foreach*. *foreachelse* y ejecutando cuando no hubieren mas valores en la variable *from*.

### Example 7.4. foreach

```
<?php
$arr = array( 1001,1002,1003);
$smarty->assign('custid', $arr);
?>
```

```
{* este ejemplo muestra todos los valores de la matriz $custid *}
{foreach from=$custid item=curr_id}
  id: {$curr_id}<br />
{/foreach}
```

Esta es la salida del ejemplo de arriba:

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

**Example 7.5. foreach key**

```
// La llave contiene la llave para cada valor del ciclo(loop)
//asignacion fisica de esta manera:
<?php
    $smarty->assign('contacts', array(
        array('phone' => '1',
              'fax' => '2',
              'cell' => '3'),
        array('phone' => '555-4444',
              'fax' => '555-3333',
              'cell' => '760-1234')
    ));
?>
```

```
{foreach name=outer item=contact from=$contacts}
  <hr />
  {foreach key=key item=item from=$contact}
    {$key}: {$item}<br />
  {/foreach}
{/foreach}
```

Esta es la salida del ejemplo de arriba:

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



**Example 7.6. Ejemplo de {foreach} - con base de datos (eg PEAR o ADODB)**

```
<?php
    $sql = 'select contact_id, name, nick from contacts order by contact';
    $smarty->assign("contacts", $db->getAssoc($sql));
?>

{foreach key=cid item=con from=$contacts}
    <a href="contact.php?contact_id={$cid}">{$con.name} - {$con.nick}</a><br />
{/foreach}
```

El ciclo(Loop) foreach también tiene sus propias variables para manipular las propiedades del foreach. Estas son indicadas así: {`$smarty.foreach.foreachname.varname`} con `foreachname` siendo el nombre especificado del atributo *name* del foreach.

Ver {`section`} para ejemplos ide las propiedades bajo las cuales son identicos.

## iteration

`iteration` es usado para mostrar la interacción actual del ciclo(loop). `iteration` siempre comienza en 1 incrementado en uno cada interacción.

## first

`first` Toma el valor true si la interacción actual del foreach es la primera.

## last

`last` Toma el valor de true si la interacción actual del foreach es la ultima.

## show

`show` Es usado como parámetro para el foreach. `show` Es un valor booleano, true o false. Si es false, el foreach no será mostrado. Si tuviera un `foreachelse` presente, este será alternativamente mostrado.

## total

`total` Es usado para mostrar el número de interacciones del foreach. Este puede ser usado dentro o después de el.

Ver tambien {`section`} y `$smarty.foreach`.

# include

Nombre del Atributo	Tipo	requerido	Default	Descripción
file	string	Si	<i>n/a</i>	El nombre del archivo de template a Incluir.
assign	string	No	<i>n/a</i>	El nombre de una variable que contendra toda la salida del template.
[var ...]	[var type]	No	<i>n/a</i>	Variable para pasar localmente a el template

Las etiquetas `{include}` son usadas para incluir otros templates en el template actual. Cualquier variable disponible en el template actual, también esta disponible dentro del template incluido. La etiqueta `{include}` debe tener el atributo "file", el cual contiene la ruta del archivo a incluir.

Usted puede opcionalmente pasar el atributo `'assign'`, el cual especificara el nombre de una variable de template para el cual contendra toda la salida de `{include}` en vez de mostrarla.

## Example 7.7. funcion `{include}`

```
<html>
<head>
  <title>{$title}</title>
</head>
<body>
{include file="page_header.tpl"}

{* el cuerpo del template va aqui *}
{include file="/$tpl_name.tpl"} <-- will replace $tpl_name with value

{include file="page_footer.tpl"}
</body>
</html>
```

Usted también puede pasar variables al template incluidas como atributos. Cualquier variable pasada al template incluidas como atributos estan disponibles solamente dentro el espacio del template. Las variables pasadas como atributos sobrescriben a las variables del template actual, en el caso en el que estas tengan el mismo nombre.

**Example 7.8. Función {include} pasando variables**

```
{include file="header.tpl" title="Main Menu" table_bgcolor="#c0c0c0"}

{* el cuerpo del template va aqui *}

{include file="footer.tpl" logo="http://my.example.com/logo.gif"}
```

Donde header.tpl puede ser

```
<table border='1' width='100%' bgcolor='{ $table_bgcolor | default: "#0000FF" }' >
  <tr><td>
    <h1>{ $title }</h1>
  </td></tr>
</table>
```

**Example 7.9. {include} y asignacion de variables**

En este ejemplo asignan el contenido de nav.tpl en la variable \$navbar, entonces la salida hasta arriba y hasta abajo de pagina.

```
<body>
{include file="nav.tpl" assign="navbar"}
{include file="header.tpl" title="Main Menu" table_bgcolor="#c0c0c0"}
{ $navbar }

{* el cuerpo del template va aqui *}

{include file="footer.tpl" logo="http://my.example.com/logo.gif"}
{ $navbar }
</body>
```

Use la sintaxis de template resources para incluir archivos fuera del directorio \$template\_dir.

**Example 7.10. Ejemplos de recursos para la función include**

```
{* ruta absoluta *}
{include file="/usr/local/include/templates/header.tpl"}

{* ruta absoluta (lo mismo) *}
{include file="file:/usr/local/include/templates/header.tpl"}

{* ruta absoluta de windows (DEBE usar el prefijo "file:") *}
{include file="file:C:/www/pub/templates/header.tpl"}

{* incluir a partir del recurso de template denominado "db" *}
{include file="db:header.tpl"}
```

ver también {include\_php}, {php}, Template Resources y Componentized Templates.

## {include\_php}

Nombre del Atributo	Tipo	Requerido	Default	Descripción
file	string	Si	<i>n/a</i>	El nombre del archivo php a incluir
once	boolean	No	<i>true</i>	Cuando incluir o no el archivo php mas de una vez, ser incluido varias veces
assign	string	No	<i>n/a</i>	El nombre de la variable que recibirá la salida del archivo php

### Nota técnica

{include\_php} es muy desaprovechado desde Smarty, usted puede lograr la misma funcionalidad por medio de las funciones de costumbre del template. La unica razón para usar {include\_php} es si usted en realidad tiene la necesidad de poner en cuarentena la funcion de php fuera del directorio de plugins y su codigo de la aplicación. Vea un ejemplo de templates componentizados para detalles.

Las etiquetas {include\_php} son usadas para incluir un script PHP dentro de su template. Si la seguridad estuviera activada, entonces el script PHP debe estar localizado en la ruta \$trusted\_dir. La etiqueta include\_php debe tener el atributo "file", el cual contiene la ruta del archivo PHP a ser incluido, o el relativo al \$trusted\_dir, o una ruta absoluta.

Por default, los archivos son incluidos solo una vez a un cuando son incluidos varias veces en el template. Usted puede especificar que este sea incluido todas la veces con un atributo *once*. Definiendo como false incluira el script php cada vez que este sea incluido en el template.

Usted puede opcionalmente pasar el atributo *assign*, el cual especificara una variable del template la cual contendra toda la salida del *{include\_php}* en vez de mostrarla.

El objeto smarty esta disponible como *\$this* dentro del script php que usted incluyo.

### Example 7.11. funcion `{include_php}`

load\_nav.php

```
<?php

// carga variables de una base de datos mysql y define esta para el template
require_once("MySQL.class.php");
$sql = new MySQL;
$sql->query("select * from site_nav_sections order by name",SQL_ALL);
$this->assign('sections', $sql->record);

?>
```

Donde index.tpl es:

```
{* ruta absoluta o relativa del $trusted_dir *}
{include_php file="/path/to/load_nav.php"}

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

Ver también `{include}`, `{php}`, `{capture}`, `Template Resources` y `Componentized Templates`

## insert

Nombre del Atributo	Tipo	Requerido	Default	Descripción
name	string	Si	<i>n/a</i>	El nombre de la función <code>insert(insert_name)</code>
assign	string	No	<i>n/a</i>	El nombre de la variable del template que recibirá la salida
script	string	No	<i>n/a</i>	El nombre de un php que será incluido antes que la función <code>insert</code> sea llamada

Nombre del Atributo	Tipo	Requerido	Default	Descripción
[var ...]	[var type]	No	<i>n/a</i>	Variable para pasar a la función insert

La etiqueta funciona parecido a las etiquetas {include}, excepto que las etiquetas insert no van para el cache cuando caching esta activado. Esta sera ejecutada a cada invocación del template.

Digamos que usted tiene un template con un banner en la parte de arriba de la pagina. El banner puede contener cualquier mezcla de HTML, imagenes, flash, etc. Así nosotros no podemos usar una liga(link) estatica aquí, y nosotros no queremos que este el contenido oculto con la pagina. Aquí vemos la etiqueta {insert}: el template conoce los valores #banner\_location\_id# y #site\_id# (obtenidos de un archivo de configuración), y necesita llamar una función para obtener el contenido del banner.

### Example 7.12. función {insert}

```
{* ejemplo de traer un banner *}
{insert name="getBanner" lid=#banner_location_id# sid=#site_id#}
```

En este ejemplo, nosotros estamos usando el nombre "getBanner" y pasando los parámetros #banner\_location\_id# y #site\_id#. El Smarty lo buscara en la función llamada insert\_getBanner() en su aplicación PHP, pasando los valores de #banner\_location\_id# y #site\_id# como primer argumento en una matriz asociativa. Todos los nombres de las funciones insert en su aplicación deben ser precedidas por "insert\_" para prevenir posibles problemas con nombres de funciones repetidos. Su función insert\_getBanner() debe hacer algo con los valores pasados y retornar los resultados. Estos resultados son mostrados en el template en lugar de la etiqueta insert. En este ejemplo, el Smarty llamara esta función: insert\_getBanner(array("lid" => "12345", "sid" => "67890")); y mostrara el resultado retornado en el lugar de la etiqueta insert.

Si usted proporciona el atributo "assign", la salida de la etiqueta {insert} será dada a esta variable en vez de ser una salida en el template. Nota: definir la salida a una variable no es util cuando el cache esta habilitado.

Si usted proporciona el atributo "script", este script php será incluido (solo una vez) antes de la ejecución de la función {insert}. Este es el caso donde la función insert no exista todavia, y el script php debe ser incluido antes para que pueda funcionar. La ruta puede ser absoluta o relativa a \$trusted\_dir. Cuando la seguridad esta activada, el script debe estar en \$trusted\_dir.

El objeto Smarty es pasado como segundo argumento. De este modo puede referenciar y modificar información del objeto Smarty dentro de la función.

### Nota Tecnica

Es posible tener partes del template fuera de la cache. Si usted tuviera caching activado, la etiqueta insert no podra heredar por la cache. Esta sera ejecutada dinámicamente cada vez que la pagina sea creada, igual con paginas en cache. Esto funciona bien para cosas como banners, encuestas, clima, busqueda de resultados, areas de opinión de usuario, etc.

## if,elseif,else

Los comandos {if} del Smarty tiene mucho de la flexibilidad del comando **if** [http://php.net/if] de php, con algunas adiciones para la herramienta de template. Todo {if} debe tener su {/if}. {else} y {elseif} también son permitidos. Toda las condicionales de PHP son reconocidas, tal como //, or, &&, and, etc.

La siguiente es una lista de calificadores reconocidos, los cuales deberan estar separados los dos elementos por espacios. Nota loas articulos pueden listarse [entre corchetes] es opcional. Equivalentes al lugar donde se apliquen en PHP.

Calificador	Alternativa	Ejemplo de Sintaxis	Significado	Equivalente en PHP
==	eq	\$a eq \$b	Iguales	==
!=	ne, neq	\$a neq \$b	Diferentes	!=
>	gt	\$a gt \$b	Mayor que	>
<	lt	\$a lt \$b	menor que	<
>=	gte, ge	\$a ge \$b	mayor que o igual	>=
<=	lte, le	\$a le \$b	menor que o igual	<=
===		\$a === 0	Igual e indentico	===
!	not	not \$a	negación (unary)	!
%	mod	\$a mod \$b	modulo	%
is [not] div by		\$a is not div by 4	divisible por	\$a % \$b == 0
is [not] even		\$a is not even	[not] es numero par (unary)	\$a % 2 == 0
is [not] even by		\$a is not even by \$b	agrupar niveles pares [not]	(\$a / \$b) % 2 == 0
is [not] odd		\$a is not odd	[not] el numero es impar (unary)	\$a % 2 != 0
is [not] odd by		\$a is not odd by \$b	[not] agrupa los niveles impares	(\$a / \$b) % 2 != 0

```

{elseif $name eq "Wilma"}
Welcome Ma'am.
{else}
Welcome, whatever you are.
{/if}

```

---

### Example 7.13. sentencia if

```

{* El mismo que arriba con operador logico *}
{if $name eq "Fred" or $name eq "Wilma"}
...
{/if}

{* El mismo que arriba *}
{if $name == "Fred" || $name == "Wilma"}
...
{/if}

{* La siguiente sintaxis no funcionara, el calificador de condición
deben estar separados entre ellos por espacios *}
{if $name=="Fred" || $name=="Wilma"}
...
{/if}

{* los parentesis son permitidos *}
{if ( $amount < 0 or $amount > 1000 ) and $volume >= #minVolAmt#}
...
{/if}

{* Usted también puede colocar funciones de PHP *}
{if count($var) gt 0}
...
{/if}

{* checa si el valor es par o impar *}
{if $var is even}
...
{/if}
{if $var is odd}
...
{/if}
{if $var is not odd}
...
{/if}

{* checa si la variable var es divisible por 4 *}
{if $var is div by 4}
...
{/if}

{* Checa si la variable var es igual, agrupandola por dos. 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}

```



## {ldelim},{rdelim}

{ldelim} y {rdelim} son usados para escapar delimitadores en el template, en nuestro caso "{" or "}". Usted puede usar solo {literal}{/literal} para escapar bloques de texto. Vea tambien {\$smarty.ldelim}.

### Example 7.14. {ldelim}, {rdelim}

```
{* Esto mostrara los delimitadores del template *}
{ldelim}funcname{rdelim} is how functions look in Smarty!
```

La salida del ejemplo de arriba:

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

Otros ejemplos con algunos javascript

```
<script language="JavaScript">
function foo() {ldelim}
    ... code ...
{rdelim}
</script>
```

esta es la salida

```
<script language="JavaScript">
function foo() {
    .... code ...
}
</script>
```

Vea también Escaping Smarty Parsing

## literal

Las etiquetas literal permiten que un block de datos sea tomado literalmente, no siendo interpretado por el smarty. Esto es generalmente utilizado alrededor de bloques javascript o stylesheet, en donde pueden

haber sintaxis delimitadoras que puedan interferir con el template. Cualquier cosa dentro de las etiquetas `{literal}{/literal}` no es interpretado, si no desplegado tal como esta. Si usted necesita en su template etiquetas incrustadas en su bloque de literal, considere usar `{ldelim}{rdelim}` para escapar delimitadores individuales en lugar de eso.

### Example 7.15. Etiqueta literal

```
{literal}
<script type="text/javascript">

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

</script>
{/literal}
```

Ver también Escaping Smarty Parsing.

## {php}

Las etiquetas `{php}` permiten a suetdd incrustar código php directamente en el template. No será escapado, no importando la definición de `$php_handling`. Esto es solo para usuario avanzados y normalmente no es necesario.

### Example 7.16. Etiqueta {php}

```
{php}
// incluyendo un script php
// directamente en el template.
include("/path/to/display_weather.php");
{/php}
```

### Nota técnica

Para poder tener acceso a las variables de PHP puede ser necesario usar la palabra clave global [<http://php.net/global>] de PHP.

ver También `$php_handling`, `{include_php}`, `{include}` y Componentized Templates.

## section,sectionelse

Nombre del Atributo	Tipo	Requerido	Default	Descripción
name	string	Si	<i>n/a</i>	El nombre de la section
loop	mixed	Si	<i>n/a</i>	El nombre de la variable para determinar el número de iteraciones
start	integer	No	<i>0</i>	La posición del índice de la section donde va a comenzar. Si el valor es negativo, la posición del inicio se calculara a partir del final de la matriz. Por ejemplo, si hubieran 7 valores en la matriz y comienza por -2, el índice inicial es 5. Valores inválidos (valores fuera del tamaño de la matriz) son automáticamente truncados para el valor valido mas próximo.
step	integer	No	<i>1</i>	El valor del step que sera usado para el loop de la matriz. Por ejemplo, step=2 realizara el loop con los índices 0,2,4, etc. Si step es negativo, este avanzara en la matriz de atras para adelante.
max	integer	No	<i>n/a</i>	Defíne el número máximo de ciclos(loops) para la section.

Nombre del Atributo	Tipo	Requerido	Default	Descripción
show	boolean	No	<i>true</i>	Determina cuando mostrar o no esta sección

Las section del template son usada para realizar un ciclo(loop) de un **arreglo de datos**. (al agiual que un {foreach}). Todas las etiquetas *section* deben tener su par */section*. Los parámetros requeridos son *name* y *loop*. El nombre de la section puede ser el que usted quiera, formado por letras, números y subrayados. Las sections pueden ser anidadas, y los nombres de la section anidadas deben ser diferentes unos de otros. Las variables del loop (normalmente una matriz de valores) determina el número de veces del loop de la section. Cuando estuviera mostrando una variable dentro de una section, el nombre de la section debe estar al lado de la variable dentro de corchetes []. *sectionelse* es ejecutado cuando no hubiera valores para la variable del loop(ciclo).

### Example 7.17. section

```
<?php

$data = array(1000,1001,1002);
$smarty->assign('custid',$data);

?>

{* this example will print out all the values of the $custid array *}
{section name=customer loop=$custid}
  id: {$custid[customer]}<br />
{/section}
<hr />
{* print out all the values of the $custid array reversed *}
{section name=foo loop=$custid step=-1}
  {$custid[foo]}<br />
{/section}
```

The above example will output:

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

Otro par de ejemplos sin un arreglo asignado.

```
{section name=foo start=10 loop=20 step=2}
  {$smarty.section.foo.index}
{/section}
<hr />
{section name=bar loop=21 max=6 step=-2}
  {$smarty.section.bar.index}
{/section}
```

Esta es la salida del ejemplo de arriba:

```
10 12 14 16 18
<hr />
20 18 16 14 12 10
```

### Example 7.18. loop(ciclo) de la variable section

```
<?php
$id = array(1001,1002,1003);
$smarty->assign('custid',$id);

$fullnames = array('John Smith','Jack Jones','Jane Munson');
$smarty->assign('name',$fullnames);

$addr = array('253 N 45th', '417 Mulberry ln', '5605 apple st');
$smarty->assign('address',$addr);

?>
```

```
{* la variable del loop solo determina el número de veces del ciclo.
   Usted puede acceder a cualquier variable del template dentro de la section.
   Este ejemplo asume que $custid, $name y $address son todas matrices
   conteniendo el mismo número de valores *}
{section name=customer loop=$custid}
  id: {$custid[customer]}&lt;br&gt;
  name: {$name[customer]}&lt;br&gt;
  address: {$address[customer]}&lt;br&gt;
  &lt;p&gt;
{/section}
```

La salida del ejemplo de arriba:

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

### Example 7.19. Nombres de section

```
{*
  El nombre de la section puede ser el que usted quiera,
  y es usado para referenciar los datos dentro de una section
*}
{section name=anything loop=$custid}
<p>
  id: {$custid[anything]}<br />
  name: {$name[anything]}<br />
  address: {$address[anything]}
</p>
{/section}
```

```
<?php
```

### Example 7.20. sections anidadas

```
$id = array(1001,1002,1003);
$smarty->assign('custid',$id);
```

```
$fullnames = array('John Smith','Jack Jones','Jane Munson');
$smarty->assign('name',$fullnames);
```

```
$addr = array('253 N 45th', '417 Mulberry ln', '5605 apple st');
$smarty->assign('address',$addr);
```

```
$types = array(
    array( 'home phone', 'cell phone', 'e-mail'),
    array( 'home phone', 'web'),
    array( 'cell phone')
);
```

```
$smarty->assign('contact_type', $types);
```

```
$info = array(
    array('555-555-5555', '666-555-5555', 'john@myexample.com'),
    array( '123-456-4', 'www.example.com'),
    array( '0457878')
);
```

```
$smarty->assign('contact_info', $info);
```

*\* Las sections pueden ser anidados tan profundamente como usted quiera. Con las sections anidadas, usted puede acceder a estructuras complejas, como una matriz multi-dimensional. En este ejemplo, \$contact\_type[customer] es una matriz de tipos de contacto para el cliente actual. \**

```
{section name=customer loop=$custid}
<hr>
  id: {$custid[customer]}<br />
  name: {$name[customer]}<br />
  address: {$address[customer]}<br />
  {section name=contact loop=$contact_type[customer]}
  la salida de este ejemplo es arriba.
  {section name=contact loop=$contact_type[customer]}: {$contact_info[customer][contact]}<br />
  {/section}
  {/section}
  name: John Smith<br />
  {/section}
  address: 253 N 45th<br />
    home phone: 555-555-5555<br />
    cell phone: 666-555-5555<br />
    e-mail: john@myexample.com<br />
<hr>
  id: 1001<br />
  name: Jack Jones<br />
  address: 417 Mulberry ln<br />
    home phone: 123-456-4<br />
    web: www.example.com<br />
<hr>
  id: 1002<br />
  name: Jane Munson<br />
  address: 5605 apple st<br />
    cell phone: 0457878<br />
```



```
<?php
```

### Example 7.21. sections y matrices asociativas

```
$data = array(
    array('name' => 'John Smith', 'home' => '555-555-5555',
        'cell' => '666-555-5555', 'email' => 'john@myexample.com'),
    array('name' => 'Jack Jones', 'home' => '777-555-5555',
        'cell' => '888-555-5555', 'email' => 'jack@myexample.com'),
    array('name' => 'Jane Munson', 'home' => '000-555-5555',
        'cell' => '123456', 'email' => 'jane@myexample.com')
);
$smarty->assign('contacts', $data);
```

```
?>
{*
```

Este es un ejemplo que muestra los datos de una matriz asociativa dentro de una section

```
*}
```

```
{section name=customer loop=$contacts}
```

```
<p>
```

```
name: {$contacts[customer].name}<br />
```

```
home: {$contacts[customer].home}<br />
```

```
<p>cell: {$contacts[customer].cell}<br />
```

```
e-mail: {$contacts[customer].email}
```

```
</p>
```

```
{/section}6-555-5555<br />
```

```
e-mail: john@myexample.com
```

```
</p>
```

```
<p>
```

```
name: Jack Jones<br />
```

```
home phone: 777-555-5555<br />
```

```
cell phone: 888-555-5555<br />
```

```
e-mail: jack@myexample.com
```

```
</p>
```

```
<p>
```

```
name: Jane Munson<br />
```

```
<p>home phone: 000-555-5555<br />
```

```
cell phone: 123456<br />
```

```
Ejemplo usando una base de datos (eg usando Pear o Adodb).
$smarty->assign('contacts', $db->getAll($sql));
```

```
</p>
```

```
{*
```

```
?> salida de la base de datos, resultado en una tabla
```

```
*}
```

```
<table>
```

```
<tr><th>&nbsp;</th><th>Name</th><th>Home</th><th>Cell</th><th>Email</th></tr>
```

```
{section name=co loop=$contacts}
```

```
<tr>
```

```
<td><a href="view.php?id={$contacts[co].id}">view<a></td>
```

```
<td>{$contacts[co].name}</td>
```

```
<td>{$contacts[co].home}</td>
```

```
<td>{$contacts[co].cell}</td>
```

```
<td>{$contacts[co].email}</td>
```

```
</tr>
```

```
{/section}
```

```
</table>
```

**Example 7.22. {sectionelse}**

```
{* sectionelse se ejecutara si no hubieran valores en $custid *}
{section name=customer loop=$custid}
  id: {$custid[customer]}<br />
{sectionelse}
  there are no values in $custid.
{/section}
```

Las sections también tiene sus propias variables que manipulan las propiedades de section. Estas son indicadas así: {\$smarty.section.sectionname.varname}

**Note**

NOTA: a partir de Smarty 1.5.0, la sintaxis de las variables de las propiedades de section ha sido cambiadas de {%sectionname.varname%} a {\$smarty.section.sectionname.varname}. La sintaxis antigua es aun soportada, pero usted puede ver la referencia de la sintaxis nueva en los ejemplos del manual.

## index

index es usado para mostrar el índice actual del cliclo(loop), comenzando en cero (o comienza con el atributo dado), e incrementando por uno (o por un atributo de paso dado).

**Nota Tecnica**

Si las propiedades de paso y comienzo del section son modificadas, entonces estas funcionan igual a las propiedades de iteration de la section, exepcto que comienzan en 0 en vez de 1.

**Example 7.23. {section} propiedades del index**

```
{* FYI, $custid[customer.index] y $custid[customer] are identical in meaning *}
{section name=customer loop=$custid}
  {$smarty.section.customer.index} id: {$custid[customer]}<br />
{/section}
```

salida del ejemplo de arriba:

```
0 id: 1000<br />
1 id: 1001<br />
2 id: 1002<br />
```

## **index\_prev**

El `index_prev` es usado para mostrar el índice anterior del loop(ciclo). del primer loop(ciclo) esto es definido como -1.

## **index\_next**

El `index_next` es usado para mostrar el próximo índice del loop. del último loop, esto es uno mas que el índice actual( respetando la definición del atributo `step` que se a dado.)

### Example 7.24. {section} propiedades del index\_next y index\_prev

```
<?php

$data = array(1001,1002,1003,1004,1005);
$smarty->assign('custid',$data);

?>

{* FYI, $custid[cus.index] and $custid[cus] are identical in meaning *}

<table>
  <tr>
    <th>index</th><th>id</th>
    <th>index_prev</th><th>prev_id</th>
    <th>index_next</th><th>next_id</th>
  </tr>
  {section name=cus loop=$custid}
  <tr>
    <td>{$smarty.section.cus.index}</td><td>{$custid[cus]}</td>
    <td>{$smarty.section.cus.index_prev}</td><td>{$custid[cus.index_prev]}</td>
    <td>{$smarty.section.cus.index_next}</td><td>{$custid[cus.index_next]}</td>
  </tr>
{/section}
</table>
```

la salida del ejemplo de arriba esta contenido en la siguiente tabla:

index	id	index_prev	prev_id	index_next	next_id
0	1001	-1		1	1002
1	1002	0	1001	2	1003
2	1003	1	1002	3	1004
3	1004	2	1003	4	1005
4	1005	3	1004	5	

## iteration

iteration es usado para mostrar la iteración actual del loop(ciclo).

### Note

Esto no es afectado por las propiedades del section start, step y max, distinto de las propiedades del index. Iteration también comienza con 1 en vez de 0 como index. rownum es un alias de iteration, estas funcionan de manera identica.

**Example 7.25. {section} propiedades de iteration**

```

<?php

// array of 3000 to 3015
$id = range(3000,3015);
$smarty->assign('custid',$id);

?>

{section name=cu loop=$custid start=5 step=2}
  iteration={$smarty.section.cu.iteration}
  index={$smarty.section.cu.index}
  id={$custid[cu]}<br />
{/section}

```

salida del ejemplo de arriba:

```

iteration=1 index=5 id=3005<br />
iteration=2 index=7 id=3007<br />
iteration=3 index=9 id=3009<br />
iteration=4 index=11 id=3011<br />
iteration=5 index=13 id=3013<br />
iteration=6 index=15 id=3015<br />

```

Este ejemplo utiliza la propiedad iteration para salida a una tabla bloqueando el encabezado para cada 5 renglones (utilice {if} con el operador mod).

```

<table>
{section name=co loop=$contacts}
  {if $smarty.section.co.iteration % 5 == 1}
    <tr><th>&nbsp;</th><th>Name</th><th>Home</th><th>Cell</th><th>Email</th></tr>
  {/if}
  <tr>
    <td><a href="view.php?id={$contacts[co].id}">view<a></td>
    <td>{$contacts[co].name}</td>
    <td>{$contacts[co].home}</td>
    <td>{$contacts[co].cell}</td>
    <td>{$contacts[co].email}</td>
  </tr>
{/section}
</table>

```

## first

first es definido como true se la iteración actual de la section es la primera.

## last

last es definido como true si la iteración actual del section es la ultima.

### Example 7.26. {section} propiedades first y last

En este ciclo de ejemplo el arreglo \$customer, en la salida es bloqueado el encabezado en la primera iteracion y en la ultima la salida es bloqueo para el pie de pagina. (Utilice la propiedad section total)

```
{section name=customer loop=$customers}
  {if $smarty.section.customer.first}
    <table>
      <tr><th>id</th><th>customer</th></tr>
    {/if}

    <tr>
      <td>{$customers[customer].id}</td>
      <td>{$customers[customer].name}</td>
    </tr>

    {if $smarty.section.customer.last}
      <tr><td></td><td>{$smarty.section.customer.total} customers</td></tr>
    </table>
  {/if}
{/section}
```

## rownum

rownum es usado para mostrar la interacción actual del loop(ciclo), comenzando con 1. Es un alias para iteration, estas funcionan de modo identico.

## loop

loop es usado para mostrar el ultimo número del índice del loop(ciclo) de esta section. Esto puede ser usado dentro o fuera del section.

### Example 7.27. {section} propiedades de index

```
{section name=customer loop=$custid}
  {$smarty.section.customer.index} id: {$custid[customer]}<br />
{/section}
```

There were {\$smarty.section.customer.loop} customers shown above.

La salida del ejemplo de arriba:

```
0 id: 1000<br />
1 id: 1001<br />
2 id: 1002<br />
```

There were 3 customers shown above.

## show

*show* Es usado como parámetro para *section*. *show* Es un valor booleano, true o false. Si es false, la *section* no será mostrada. Si existiera un *sectionelse* presente, este será alternativamente mostrado.

**Example 7.28. section atributos de show**

```
{*
  $show_customer_info debe ser pasado de la aplicacion PHP,
  para regular cuando mostrar o no esta section shows
*}
{section name=customer loop=$custid show=$show_customer_info}
  {$smarty.section.customer.rownum} id: {$custid[customer]}<br />
{/section}

{if $smarty.section.customer.show}
  the section was shown.
{else}
  the section was not shown.
{/if}
```

La salida del ejemplo de arriba:

```
1 id: 1000<br />
2 id: 1001<br />
3 id: 1002<br />
```

```
the section was shown.
```

**total**

total es usado para mostrar el número de iteraciones que está section tendrá. Este puede ser usado dentro o fuera del section.



### Example 7.29. {section} propiedades de total

```
{section name=customer loop=$custid step=2}
  {$smarty.section.customer.index} id: {$custid[customer]}<br />
{/section}
```

There were {\$smarty.section.customer.total} customers shown above.

The above example will output:

```
0 id: 1000<br />
2 id: 1002<br />
4 id: 1004<br />
```

There were 3 customers shown above.

Ver también {foreach} y \$smarty.section.

## {strip}

Muchas veces el diseñador de web tiene problemas con los espacios en blanco y retornos de carro que afectan la salida del HTML (browser "features"), si usted tiene que colocar todas sus etiquetas juntas para tener los resultados deseados. Esto normalmente termina en un template ilegible o que no se puede leer.

A cualquier cosa dentro de las etiquetas{strip}{/strip} en Smarty le son retirados los espacios en blanco y retornos de carro al inicio y al final de las líneas antes que sean mostrados. De este modo usted puede mantener su template legible, y no se preocupara de que los espacios en blanco extras le causen problemas.

### Nota Técnica

{strip}{/strip} no afecta el contenido de las variables del template. Vea la función strip modifier.

**Example 7.30. {strip} tags**

```
{* El siguiente código se ejecutara todo junto en una sola linea de salida *}
{strip}
<table border='0'>
  <tr>
    <td>
      <A HREF="{ $url }">
        <font color="red">This is a test</font>
      </A>
    </td>
  </tr>
</table>
{/strip}
```

salida del ejemplo de arriba:

```
<table border=0><tr><td><A HREF="http://w... snipped...</td></tr></table>
```

Note que en el ejemplo de arriba, todas las líneas comienzan y termina con etiquetas HTML. Tenga cuidado en que todas las líneas corran conjuntamente. Si usted tuviera textos planos simples en el inicio o en el final de una línea, este estaría junto, y puede no ser el resultado deseado.

Vea También strip modifier

---

# Chapter 8. Custom Functions

Smarty viene con varias funciones personalizadas que usted puede usar en sus templates.

## {assign}

Nombre del Atributo	Tipo	Requerido	Default	Descripción
var	string	Si	<i>n/a</i>	El nombre de la variable que esta ganando el valor
value	string	Si	<i>n/a</i>	El valor que esta siendo dado

{assign} es usado para definir valores a las variables de template **durante la ejecución** del template.

### Example 8.1. {assign}

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

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

Salida del ejemplo de arriba:

```
The value of $name is Bob.
```

### Example 8.2. Accesando variables desde un script de PHP. {assign}

Puedes acceder {assign} variables desde php usando `get_template_vars()`. sin embargo, las variables solo estan disponibles despues/durante la ejecución del template como en el siguiente ejemplo

```
{* index.tpl *}
{assign var="foo" value="Smarty"}

<?php

// this will output nothing as the template has not been executed
echo $smarty->get_template_vars('foo');

// fetch the template to a dead variable
$dead = $smarty->fetch('index.tpl');

// this will output 'smarty' as the template has been executed
echo $smarty->get_template_vars('foo');

$smarty->assign('foo','Even smarter');

// this will output 'Even smarter'
echo $smarty->get_template_vars('foo');

?>
```

La siguiente función *optionally* también puede asignar variables al template.

{capture}, {include}, {include\_php}, {insert}, {counter}, {cycle}, {eval}, {fetch}, {math}, {textformat}

Ver también `assign()` y `get_template_vars()`.

## {counter}

Nombre del Atributo	Tipo	Requerido	Default	Descripción
name	string	No	<i>default</i>	El nombre del contador
start	number	No	<i>1</i>	El número inicial para contar a partir de
skip	number	No	<i>1</i>	El intervalo para contar
direction	string	No	<i>up</i>	La dirección para contar (up/down)
print	boolean	No	<i>true</i>	Cuando mostrar o no el valor

Nombre del Atributo	Tipo	Requerido	Default	Descripción
assign	string	No	<i>n/a</i>	La variable del template que va a recibir la salida

{counter} es usada para mostrar un conteo. {counter} va a depender del conteo en cada iteración. Usted puede ajustar el número, el intervalo y la dirección del conteo, así como determinar cuando mostrar o no el conteo. Usted puede tener varios contadores al mismo tiempo, dando un nombre único para cada uno. Si usted no da un nombre, será usado 'default' como nombre.

Si usted indica el atributo especial "assign", la salida de la función counter se irá para esa variable del template en vez de ser mostrada en el template.

### Example 8.3. counter

```
{* Inicia el conteo *}
{counter start=0 skip=2}<br />
{counter}<br />
{counter}<br />
{counter}<br />
```

Esta es la salida:

```
0<br />
2<br />
4<br />
6<br />
```

## cycle

Nombre del Atributo	Tipo	Requerido	Default	Descripción
name	string	No	<i>default</i>	El nombre del ciclo
values	mixed	Si	<i>N/A</i>	Los valores del ciclo, o una lista delimitada por coma (vea el atributo delimiter), o una matriz de valores.
print	boolean	No	<i>true</i>	Cuando mostrar o no el valor

Nombre del Atributo	Tipo	Requerido	Default	Descripción
advance	boolean	No	<i>true</i>	Cuando avanzar o no hacia el siguiente valor
delimiter	string	No	,	El delimitador para usar el valor del atributo.
assign	string	No	<i>n/a</i>	La variable del template que recibirá la salida
reset	boolean	No	<i>false</i>	Este coloca al ciclo en el primer valor y no le permite avanzar

{Cycle} es usado para hacer un ciclo a través de un conjunto de valores. Esto hace mas fácil alternar entre dos o mas colores en una tabla, o ciclos a través de una matriz de valores.

Usted puede usar el {cycle} en mas de un conjunto de valores en su template supliendo el atributo name. De cada uno de los conjuntos de valores.

Usted puede forzar que el valor actual no sea mostrado definiendo el atributo print en false. Esto es útil para saltarse un valor.

El atributo advance es usado para repetir un valor. cuando se definido en false, la próxima llamada para cycle mostrara el mismo valor.

Si usted indica el atributo especial "assign", la salida de la función cycle ira a la variable del template en vez de ser mostrado directamente en el template.

### Example 8.4. cycle

```
{section name=rows loop=$data}
<tr bgcolor="{cycle values="#e0e0e0,#d0d0d0"}">
  <td>{$data[rows]}</td>
</tr>
{/section}
```

```
<tr bgcolor="#e0e0e0">
  <td>1</td>
</tr>
<tr bgcolor="#d0d0d0">
  <td>2</td>
</tr>
<tr bgcolor="#e0e0e0">
  <td>3</td>
</tr>
```

## {debug}

Nombre del Atributo	Tipo	Requerido	Default	Descripción
output	string	No	<i>javascript</i>	Tipo de salida, html o javascript

{debug} Muestra el debug de la consola en la pagina. Esto funciona independiente de la definición de debug. Ya que este es ejecutado en tiempo de ejecución, este solo puede mostrar las variables definidas, no en el template, es decir en uso. Usted puede ver todas las variables disponibles del template con scope.

Ver también Debugging console

## {eval}

Nombre del Atributo	Tipo	Requerido	Default	Descripción
var	mixed	Si	<i>n/a</i>	variable (o cadena) para evaluar
assign	string	No	<i>n/a</i>	La variable del template que recibirá la salida

{eval} es usado para evaluar una variable como de template. Esto puede ser usado para cosas como incrustar tags(etiquetas)/variables del template dentro de las variables o tags(etiquetas)/variables dentro de las variables de un archivo de configuración.

Si usted indica el atributo especial "assign", la salida de la función eval se ira para esta variable de template en vez de aparecer en el template.

### Nota Técnica

Al evaluar las variables son tratadas igual que el template. Ellas siguen el mismo funcionamiento para escape y seguridad tal como si ellas fueran templates.

### Nota Técnica

Las variables evaluadas son compiladas en cada invocación, las versiones compiladas no son salvadas. Sin embargo, si usted tiene activado el cache, la salida se va a fijar en la cache junto con el resto del template.

### Example 8.5. {eval}

```

setup.conf
-----

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

Where index.tpl is:

```

{config_load file="setup.conf"}

{eval var=$foo}
{eval var=#title#}
{eval var=#ErrorCity#}
{eval var=#ErrorState# assign="state_error"}
{$state_error}
    
```

La salida del ejemplo de arriba:

```

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



## {fetch}

Nombre del Atributo	Tipo	Requerido	Default	Descripción
file	string	Si	<i>n/a</i>	El archivo, sitio http o ftp para mandar llamar
assign	string	No	<i>n/a</i>	La variable del template que va a recibir la salida

{fetch} es usado para obtener archivos de sistema local, http o ftp, y mostrar el contenido. Si el nombre del archivo comienza con "http://", la página del web site sera traída y mostrada. Si el nombre del archivo comienza con "ftp://", el archivo será obtenido del servidor ftp y mostrado. Para archivos locales, debe ser dada la ruta completa del sistema de archivos, o una ruta relativa de el script php a ejecutar.

Si usted indica el atributo especial "assign", la salida de la función {fetch} se ira a una variable de template en vez de ser mostrada en el template. (nuevo en Smarty 1.5.0)

### Nota Técnica

Esto no soporta redireccionamiento http, tenga la certeza de incluirlo en la barra el seguimiento para ir a buscar donde sea necesario.

### Nota Técnica

Si tiene activada la seguridad en su template y usted estuviera recibiendo un archivo del sistema de archivos local, esto permitira que solo archivos de uno de los directorios estuviera definido como seguro. (\$secure\_dir)

### Example 8.6. fetch

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

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

{* fetch a news headline file via ftp *}
{fetch file="ftp://user:password@ftp.example.com/path/to/currentheadlines.txt"}

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

Ver también {capture}, {eval} y fetch().

## {html\_checkboxes}

Nombre del Atributo	Tipo	Requerido	Default	Descripción
name	string	No	<i>checkbox</i>	Nombre de la lista checkbox
values	array	Si, a menos que se este utilizando el atributo options	<i>n/a</i>	Una matriz de valores para los botones checkbox
output	array	Si, a menos que estuviera usando el atributo options	<i>n/a</i>	una matriz de salida para los botones checkbox
selected	string/array	No	<i>empty</i>	El(s) elemento(s) checkbox marcado(s)
options	arreglo asociativo	Si, a menos que este usando valores y output	<i>n/a</i>	Una matriz asociativa de valores y salida
separator	string	No	<i>empty</i>	Cadena de texto para separar cada checkbox
labels	boolean	No	<i>true</i>	Adicionar la etiqueta <label> para la salida

{html\_checkboxes} es una función personalizada que crea un grupo de checkbox con datos privistos. Este cuida cuales items(s) estan seleccionados por default. Los atributos requeridos son values y output, a menos que usted use options. Toda la salida es compatible con XHTML.

Todos los parámetros que no esten en la lista de arriba son mostrados como nombre/valor dentro de cada etiqueta <input> creada.

### Example 8.7. {html\_checkboxes}

```
<?php

$smarty->assign('cust_ids', array(1000,1001,1002,1003));
$smarty->assign('cust_names', array(
    'Joe Schmoe',
    'Jack Smith',
    'Jane Johnson',
    'Charlie Brown')
);
$smarty->assign('customer_id', 1001);

?>
```

donde el template es

```
{html_checkboxes name="id" values=$cust_ids output=$cust_names
    selected=$customer_id separator="<br />"}
```

o donde el codigo es:

```
<?php

$smarty->assign('cust_checkboxes', array(
    1000 => 'Joe Schmoe',
    1001 => 'Jack Smith',
    1002 => 'Jane Johnson',
    1003 => 'Charlie Brown')
);
$smarty->assign('customer_id', 1001);

?>
```

y el template es

```
{html_checkboxes name="id" options=$cust_checkboxes selected=$customer_id separato
```

salida de ambos ejemplos:

```
<label><input type="checkbox" name="id[]" value="1000" />Joe Schmoe</label><br />
<label><input type="checkbox" name="id[]" value="1001" checked="checked" />Jack Sm
<br />
<label><input type="checkbox" name="id[]" value="1002" />Jane Johnson</label><br />
<label><input type="checkbox" name="id[]" value="1003" />Charlie Brown</label><br />
```

**Example 8.8. ejemplo de base de datos (eg PEAR o ADODB):**

```

<?php

$sql = 'select type_id, types from types order by type';
$smarty->assign('types', $db->getAssoc($sql));

$sql = 'select * from contacts where contact_id=12';
$smarty->assign('contact', $db->getRow($sql));

?>

{html_checkboxes name="type" options=$types selected=$contact.type_id separator="<

```

Vea también {html\_radios} y {html\_options}

## {html\_image}

Nombre del Atributo	Tipo	Requerido	Default	Descripción
file	string	Si	<i>n/a</i>	nombre/ruta de la imagen
height	string	No	<i>Altura actual de la imagen</i>	altura con la cual la imagen debe ser mostrada
width	string	No	<i>Largo actual de la imagen</i>	largo con el cual la imagen debe ser mostrada
basedir	string	no	<i>document root del servidor web</i>	ruta relativa para la base del directorio
alt	string	no	""	descripción alternativa de la imagen
href	string	no	<i>n/a</i>	valor href a donde la imagen será ligada

{html\_image} es una función habitual que genera una etiqueta HTML para una imagen. La altura y lo largo son automáticamente calculadas a partir del archivo de la imagen si ningún valor suplido.

basedir es el directorio base en el cual las rutas relativas de las imagenes estan basados. Si no lo proporciona, el document root del servidor (env variable de ambiente DOCUMENT\_ROOT) es usada como el directorio base. Si la \$security esta habilitada, la ruta para la imagen debe estar dentro de un directorio seguro.

*href* es el valor href a donde la imagen sera ligada. Si un link es proporcionado, una etiqueta `<a href="LINKVALUE"><a>` es puesta alrededor de la imagen.

Todos los parametros que no esten dentro de la lista de arriba son mostrados como pares de nombre/valor dentro de la etiqueta creada `<img>`.

### Nota Técnica

`{html_image}` requiere un acceso a disco para leer la imagen y calcular la altura y el largo. Si usted no usa cache en el template, generalmente es mejor evitar `{html_image}` y utilizar las etiquetas de imagen estáticas para un optimo funcionamiento.

### Example 8.9. html\_image example

where `index.tpl` is:

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

la posible salida puede ser:

```



```

## {html\_options}

Nombre del Atributo	Tipo	Requerido	Default	Descripción
values	array	Si, a menos que use el atributo options	<i>n/a</i>	una matriz de valores para el menu dropdown
output	array	Si, a menos que use el atributo options	<i>n/a</i>	una matriz de salida para el menu dropdown
selected	string/array	No	<i>empty</i>	los elemento(s) de la option selecionado(s)
options	arreglo asociativo	Si, a menos que utilize valores y salida	<i>n/a</i>	una matriz asociativa de valores y salida

Nombre del Atributo	Tipo	Requerido	Default	Descripción
name	string	No	<i>empty</i>	nombre del grupo seleccionado

{html\_options} es una función customizada que crea un grupo html <select><option> con los datos proporcionados. Este se encarga de cuidar cuales datos han sido seleccionado por default. Los atributos son valores y salidas, a menos que usted utilice options en lugar de eso.

Si un valor es una matriz, este será tratado como un <optgroup> html, y mostrara los grupos. La recursión es soportada por <optgroup>. Todas las salidas son compatibles con XHTML.

Si el atributo opcional *name* es dado, las etiquetas <select name="groupname"></select> encapsularan la lista de opciones. De otra manera solo es generada la lista de opciones.

Todos los parámetros que no estan en la lista de arriba son exhibidos como name/value-pairs dentro de las etiquetas <select>. Estas son ignoradas si la opcion *name* no es dada.

**Example 8.10. {html\_options}**

```
<?php

$smarty->assign('cust_ids', array(1000,1001,1002,1003));
$smarty->assign('cust_names', array(
    'Joe Schmoe',
    'Jack Smith',
    'Jane Johnson',
    'Charlie Brown'));
$smarty->assign('customer_id', 1001);

?>
```

donde el template es:

```
<select name=customer_id>
    {html_options values=$cust_ids output=$cust_names selected=$customer_id}
</select>
```

**Ejemplo 2:**

```
<?php

$smarty->assign('cust_options', array(
    1000 => 'Joe Schmoe',
    1001 => 'Jack Smith',
    1002 => 'Jane Johnson',
    1003 => 'Charlie Brown')
);
$smarty->assign('customer_id', 1001);

?>
```

donde el template es:

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

Salida de ambos ejemplos de arriba:

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

**Example 8.11. {html\_options} - Ejemplo con base de datos (eg PEAR o ADODB):**

```
<?php

$sql = 'select type_id, types from types order by type';
$smarty->assign('types', $db->getAssoc($sql));

$sql = 'select contact_id, name, email, type_id
        from contacts where contact_id='.$contact_id;
$smarty->assign('contact', $db->getRow($sql));

?>
```

Donde el template es:

```
<select name="type_id">
  <option value='null'>-- none --</option>
  {html_options name="type" options=$types selected=$contact.type_id}
</select>
```

vea también {html\_checkboxes} y {html\_radios}

## {html\_radios}

Nombre del Atributo	Tipo	Requerido	Default	Descripción
name	string	No	<i>radio</i>	Nombre de la lista del radio
values	array	Si, a menos que utilice el atributo options	<i>n/a</i>	una matriz de valores para radio buttons
output	array	Si, a menos que utilice el atributo options	<i>n/a</i>	una matriz de salida para radio buttons
selected	string	No	<i>empty</i>	El elemento del radio selccionado
options	arreglo asociativo	Si, a menos que utilice valores y salida	<i>n/a</i>	una matriz asociativa de valores y salida
separator	string	No	<i>empty</i>	cadena de texto para separar cada objeto de radio

{html\_radios} es una función customizada que crea grupos de botones de radio html con los datos proporcionados. Este esta atento para saber cual objeto esta selccionado por default. Los atributos



requeridos son valores y salidas, a menos que usted use option en lugar de eso. Toda salida es compatible con XHTML.

Todos los parámetros que no estan en la lista de arriba son impresos como pares de name/value dentro de cada etiqueta <input> creada.

### **Example 8.12. {html\_radios} : Ejemplo 1**

```
<?php
$smarty->assign('cust_ids', array(1000,1001,1002,1003));
$smarty->assign('cust_names', array(
    'Joe Schmoe',
    'Jack Smith',
    'Jane Johnson',
    'Charlie Brown')
);
$smarty->assign('customer_id', 1001);
?>
```

Donde el template es:

```
{html_radios name="id" values=$cust_ids output=$cust_names
    selected=$customer_id separator="<br />"}
```

**Example 8.13. {html\_radios} : Ejemplo 2**

```
<?php
$smarty->assign('cust_radios', array(
    1000 => 'Joe Schmoie',
    1001 => 'Jack Smith',
    1002 => 'Jane Johnson',
    1003 => 'Charlie Brown'));
$smarty->assign('customer_id', 1001);
?>
```

Donde index.tpl es:

```
{html_radios name="id" options=$cust_radios selected=$customer_id separator="<br /
```

Salida de ambos ejemplos:

```
<label for="id_1000">
<input type="radio" name="id" value="1000" id="id_1000" />Joe
Schmoie</label><br />
<label for="id_1001"><input type="radio" name="id" value="1001" id="id_1001"
checked="checked" />Jack
Smith</label><br />
<label for="id_1002"><input type="radio" name="id" value="1002" id="id_1002" />Jan
Johnson</label><br />
<label for="id_1003"><input type="radio" name="id" value="1003" id="id_1003" />Cha
Brown</label><br />
```

**Example 8.14. {html\_radios}- Ejemplo con base de Datos (eg PEAR o ADODB):**

```
<?php

$sql = 'select type_id, types from types order by type';
$smarty->assign('types', $db->getAssoc($sql));

$sql = 'select contact_id, name, email, type_id
       from contacts where contact_id='.$contact_id;
$smarty->assign('contact', $db->getRow($sql));

?>
```

y el template:

```
{html_radios name="type" options=$types selected=$contact.type_id separator="<br /
```

ver también {html\_checkboxes} y {html\_options}

## {html\_select\_date}

Nombre del Atributo	Tipo	Requerido	Default	Descripción
prefix	string	No	Date_	Con el prefijo el nombre de la variable
time	timestamp/ YYYY-MM-DD	No	Tiempo actual en el timestamp de unix o el formato YYYY-MM-DD	Cual date/time a usar
start_year	string	No	Año actual	El primer año primero en el menu dropdown, o el número de año, o el relativo al año actual (+/- N)
end_year	string	No	de la misma forma que start_year	El ultimo año en el menu dropdown, o el número de año, o el relativo al año actual (+/- N)
display_days	boolean	No	true	Muestra los días o no
display_months	boolean	No	true	Muestra los meses o no

Nombre del Atributo	Tipo	Requerido	Default	Descripción
display_years	boolean	No	true	Muestra los años o no
month_format	string	No	%B	Cual debe ser el formato de salida del mes dentro de (strftime)
day_format	string	No	%02d	Cual debe ser el formato de salida del día dentro de (sprintf)
day_value_format	string	No	%d	Cual debe ser el formato de salida del valor del día dentro de (sprintf)
year_as_text	boolean	No	false	Se mostrara o no el año como texto
reverse_years	boolean	No	false	Muestra los años en orden inverso
field_array	string	No	null	si un nombre es dado, las cajas de selección serán exhibidas semejantes a los resultados que estarán retornando al PHP en la forma. name[Day], name[Year], name[Month].
day_size	string	No	null	adiciona el tamaño al atributo para la etiqueta select si fue dada
month_size	string	No	null	adiciona el tamaño del atributo para la etiqueta select si fue dada
year_size	string	No	null	adiciona el tamaño del atributo para la etiqueta select si fue dada
all_extra	string	No	null	adiciona atributos extras para todas las etiquetas select/ input si fueron dadas
day_extra	string	No	null	adiciona atributos extras para todas

Nombre del Atributo	Tipo	Requerido	Default	Descripción
				las etiquetas select/input si fueron dadas
month_extra	string	No	null	adiciona atributos extras para todas las etiquetas select/input si fueron dadas
year_extra	string	No	null	adiciona atributos extras para todas las etiquetas select/input si fueron dadas
field_order	string	No	MDY	El orden para ser mostrados los campos
field_separator	string	No	\n	Cadena a mostrar entre los diferentes campos
month_value_format	string	No	%m	formato strftime de los valores del mes, el default es %m para el número del mes.
year_empty	string	No	null	Si es proporcionado entonces el primer elemento es el año select-box tiene este valor como etiqueta y "" como valor. Esto es util para hacer una lectura en el select-box por ejemplo "por favor seccione el año". Note que este puede usar valores como "-MM-DD" como atributos de time indicando que el año sea desmarcado.
month_empty	string	No	null	Si es proporcionado entonces el mes es el primer elemento select-box tiene este valor como etiqueta y "" como valor.

Nombre del Atributo	Tipo	Requerido	Default	Descripción
				Note que usted puede usar valores como "YYYY--DD" como atributos de time indicando que el mes sea desmarcado.
day_empty	string	No	null	Si es proporcionado entonces es dias es el primer elemento select-box tiene este valor como etiqueta y "" como valor. Note que usted puede usar valores como "YYYY-MM--" como atributos de time indicando que el dia sea desmarcado.

{html\_select\_date} es una función customizada que crea menus dropdowns de fechas para usted. Este puede mostrar algunos o todos por año, mes y dia.

**Example 8.15. {html\_select\_date}**

Codigo del Template

```
{html_select_date}
```

Esta es la salida:

```
<select name="Date_Month">
<option value="1">January</option>
<option value="2">February</option>
<option value="3">March</option>
..... snipped .....
<option value="10">October</option>
<option value="11">November</option>
<option value="12" selected="selected">December</option>
</select>
<select name="Date_Day">
<option value="1">01</option>
<option value="2">02</option>
<option value="3">03</option>
..... snipped .....
<option value="11">11</option>
<option value="12">12</option>
<option value="13" selected="selected">13</option>
<option value="14">14</option>
<option value="15">15</option>
..... snipped .....
<option value="29">29</option>
<option value="30">30</option>
<option value="31">31</option>
</select>
<select name="Date_Year">
<option value="2001" selected="selected">2001</option>
</select>
```

**Example 8.16. {html\_select\_date}**

```
{* el año seleccionado puede ser relativo al año actual *}
{html_select_date prefix="StartDate" time=$time start_year="-5"
  end_year="+1" display_days=false}
```

esta es la salida: (el año actual es 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="selected">December</option>
</select>
<select name="StartDateYear">
<option value="1995">1995</option>
<option value="1996">1996</option>
<option value="1997">1997</option>
<option value="1998">1998</option>
<option value="1999">1999</option>
<option value="2000" selected="selected">2000</option>
<option value="2001">2001</option>
</select>
```

Ver también {html\_select\_time}, date\_format, \$smarty.now y date tips.

**{html\_select\_time}**

Nombre del Atributo	Tipo	Requerido	Default	Descripción
prefix	string	No	Time_	con el prefijo el nombre de la variable
time	timestamp	No	current time	cual date/time va a usar
display_hours	boolean	No	true	Mostrar o no las horas



Nombre del Atributo	Tipo	Requerido	Default	Descripción
display_minutes	boolean	No	true	Mostrar o no los minutos
display_seconds	boolean	No	true	Mostrar o no los segundos
display_meridian	boolean	No	true	Mostrar o no el meridiano (am/pm)
use_24_hours	boolean	No	true	Usar o no reloj de 24 horas
minute_interval	integer	No	1	número de los intervalos de los minutos del menu dropdown
second_interval	integer	No	1	número de los intervalos de los segundos del menu dropdown
field_array	string	No	n/a	muestra los valores del arreglo con este nombre
all_extra	string	No	null	adiciona atributos extras a las etiquetas select/input si fueron proporcionados
hour_extra	string	No	null	adiciona atributos extras a las etiquetas select/input si fueron proporcionados
minute_extra	string	No	null	adiciona atributos extras a las etiquetas select/input si fueron proporcionados
second_extra	string	No	null	adiciona atributos extras a las etiquetas select/input si fueron proporcionados
meridian_extra	string	No	null	adiciona atributos extras a las etiquetas select/input si fueron proporcionados

{html\_select\_time} es una función customizada que crea menus dropdowns de tiempo para usted. Esta puede mostrar algunos valores, o todo en hora, minuto, segundo y am/pm.

Los atributos de time pueden tener diferentes formatos. Este puede ser un unico timestamp o una cadena conteniendo YYYYMMDDHHMMSS o una cadena parseada por php's strtotime() [<http://php.net/strtotime>].

```
<option value="07">07</option>
<option value="08">08</option>
<option value="09">09</option>
<option value="10">10</option>
```

Custom Functions

---

```
<option value="11">11</option>
<option value="12">12</option>
<option value="13">13</option>
Example 8.17. (html select time)
<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>
```

Ver también `$smarty.now`, `{html_select_date}` y `date tips`.

## {html\_table}

Nombre del Atributo	Tipo	Requerido	Default	Descripción
loop	array	Si	<i>n/a</i>	matriz de datos para el ciclo(loop)
cols	integer	No	3	Número de columnas para la tabla. Si el atributo cols esta vacío, los renglones serán determinados, entonces el numero de columnas sera calculado por el numero de renglones y el numero de elementos a mostrar para ser ajustado a las columnas de todos los elementos que serán mostrados, si ambos, renglones y columnas, son omitidos las columnas por default son 3.
rows	integer	No	<i>empty</i>	Número de renglones en la tabla. Si el atributo rows es vacío, las columnas serán determinadas, entonces el numero de renglones sera calculado por el numero de columnas y el numero de elementos a mostrar para ser ajustado el numero de renglones al total de elementos a ser mostrados.
inner	string	No	<i>cols</i>	Dirección consecutiva de los

Nombre del Atributo	Tipo	Requerido	Default	Descripción
				elementos en el arreglo para ser representados. <i>cols</i> manera en que los elementos son mostrados columna por columna. <i>rows</i> manera en que los elementos son mostrados renglon por renglon.
<code>table_attr</code>	string	No	<i>border="1"</i>	atributos para la etiqueta table
<code>tr_attr</code>	string	No	<i>empty</i>	atributos para la etiqueta tr (arreglos del ciclo)
<code>td_attr</code>	string	No	<i>empty</i>	atributos para la etiqueta td (arreglos del ciclo)
<code>trailpad</code>	string	No	<i>&amp;nbsp;</i>	valor de relleno de las celdas para el ultimo renglon con (si hay alguno)
<code>hdir</code>	string	No	<i>right</i>	dirección de una línea para ser representada. posibles valores: <i>left</i> (left-to-right), <i>right</i> (right-to-left)
<code>vdir</code>	string	No	<i>down</i>	Dirección de las columnas para ser representadas. posibles valores: <i>down</i> (top-to-bottom), <i>up</i> (bottom-to-top)

{html\_table} Es una función customizada que transforma un arreglo de datos en una tabla HTML. El atributo *cols* determina el número de columnas que tendrá la tabla. Los valores *table\_attr*, *tr\_attr* y *td\_attr* determinan los atributos dados para las etiquetas table, tr y td. Si *tr\_attr* o *td\_attr* son arreglos, ellos entrarán en un ciclo. *trailpad* y el valor depositado dentro de trailing cells en la última línea de la tabla si existe alguna presente.

**Example 8.18. html\_table**

php code:

```

-----
<?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"', 'bgcolor="#dddddd"'));
$smarty->display('index.tpl');
?>

```

template code:

```

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

```

La salida de ambos ejemplos:

```

<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>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td></tr>
</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>&nbsp;</td><td>&nbsp;</td><td>&nbsp;</td></tr>
</table>

```

**math**

Nombre del Atributo	Tipo	Requerido	Default	Descripción
equation	string	Si	<i>n/a</i>	La ecuación a ejecutar

Nombre del Atributo	Tipo	Requerido	Default	Descripción
format	string	No	<i>n/a</i>	El formato del resultado (sprintf)
var	numeric	Si	<i>n/a</i>	Valor de la variable de la ecuación
assign	string	No	<i>n/a</i>	Variable de template cuya salida sera asignada
[var ...]	numeric	Si	<i>n/a</i>	Valor de la variable de la ecuación

{math} permite diseñar ecuaciones matemáticas dentro del template. Cualquier variable numérica del template puede ser usada en ecuaciones, y el resultado es mostrado en lugar de la etiqueta. Las variables usadas en ecuaciones son pasadas como parámetros, que pueden ser variables de template o valores estáticos. +, -, /, \*, abs, ceil, cos, exp, floor, log, log10, max, min, pi, pow, rand, round, sin, sqrt, srans y tan son todos los operadores validos. Verifique la documentación de PHP para mas información acerca de estas funciones matemáticas.

Si usted proporciona el atributo especial "assign", la salida de la función matemática será atribuido a esta variable de template en vez de ser mostrada en el template.

### Nota Técnica

{math} es una función de muy alto rendimiento debido a que se puede usar con la función eval() [<http://php.net/eval>] de PHP. Hacer las matemáticas en PHP es mucho mas eficiente, asi en cualquier momento es posible hacer calculos matemáticos en PHP asignarlos a una variable y lanzar los resultados al template. Defínitivamente evite llamadas repetitivas de funciones matemáticas, dentro de los ciclos {section}.

**Example 8.19. {math}**  
**Ejemplo a:**

```
{* $height=4, $width=5 *}
{math equation="x + y" x=$height y=$width}
```

Salida del ejemplo de arriba:

9

**Ejemplo b:**

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

Salida del ejemplo de arriba:

100

**Ejemplo c:**

```
{* you can use parenthesis *}
{math equation="(( x + y ) / z )" x=2 y=10 z=2}
```

Salida del ejemplo de arriba:

6

**Ejemplo d:**

```
{* you can supply a format parameter in sprintf format *}
{math equation="x + y" x=4.4444 y=5.0000 format="%.2f"}
```

Salida del ejemplo de arriba:

9.44



## {mailto}

Nombre del Atributo	Tipo	Requerido	Default	Descripción
address	string	Yes	<i>n/a</i>	La dirección de correo electrónico(e-mail)
text	string	No	<i>n/a</i>	El texto para mostrar, el default es la dirección de correo (e-mail)
encode	string	No	<i>none</i>	Como codificar el e-mail. Puede ser none, hex, javascript o javascript_charcode.
cc	string	No	<i>n/a</i>	La dirección de correo(e-mail) para mandar una copia el carbon(cc). Separados por una coma.
bcc	string	No	<i>n/a</i>	Dirección de correo electrónico(e-mail) para mandar una copia al carbon ofuscada(bcc). Separando las direcciones por comas.
subject	string	No	<i>n/a</i>	Asunto del correo electrónico(e-mail).
newsgroups	string	No	<i>n/a</i>	newsgroup para enviar. separando las direcciones por comas.
followupto	string	No	<i>n/a</i>	Direcciones para acompañar. Separe las direcciones con comas.
extra	string	No	<i>n/a</i>	Cualquier otra información que usted quiera pasar por el link, tal como plantillas de estilo

{mailto} automatiza el proceso de creación de links de correo electrónico(e-mail) y opcionalmente los codifica. Codificar el correo electrónico(e-mail) hace mas difícil que las web spiders tomen las direcciones de nuestro sitio.

## Nota Técnica

javascript es probablemente el codificador mas utilizado, aunque usted puede utilizar también codificación hexadecimal.

### Example 8.20. {mailto}

```
{mailto address="me@example.com"}
<a href="mailto:me@example.com" >me@example.com</a>

{mailto address="me@example.com" text="send me some mail"}
<a href="mailto:me@example.com" >send me some mail</a>

{mailto address="me@example.com" encode="javascript"}
<script type="text/javascript" language="javascript">
  eval(unescape('%64%6f% ... snipped ...%61%3e%27%29%3b'))
</script>

{mailto address="me@example.com" encode="hex"}
<a href="mailto:%6d%65.. snipped..%3f%6d">&#x6d;&#x6d;..snipped...&#x6f;&#x6d;</a>

{mailto address="me@example.com" subject="Hello to you!"}
<a href="mailto:me@example.com?subject=Hello%20to%20you%21" >me@example.com</a>

{mailto address="me@example.com" cc="you@example.com,they@example.com"}
<a href="mailto:me@example.com?cc=you@example.com%20they@example.com" >me@examp

{mailto address="me@example.com" extra='class="email"'}
<a href="mailto:me@example.com" class="email">me@example.com</a>

{mailto address="me@example.com" encode="javascript_charcode"}
<script type="text/javascript" language="javascript">
  <!--
  {document.write(String.fromCharCode(60,97, ... snipped ....60,47,97,62))}
  //-->
</script>
```

ver también `escape`, `Obfuscating E-mail Addresses` y `{textformat}`

## {popup\_init}

{popup} es una integración de overLib [<http://www.bosrup.com/web/overlib/>], una biblioteca usada para ventanas popup. Esta es usada como contexto de infomación sensitiva, como ventanas de ayuda o herramientas. {popup\_init} debe ser usada una vez hasta arriba de cada pagina donde usted planea usar la función popup. overLib [<http://www.bosrup.com/web/overlib/>] fue escrita por Erik Bosrup, y la pagina esta localizada en <http://www.bosrup.com/web/overlib/>.

A partir da versión 2.1.2 de Smarty, overLib NO viene con la distribución. Descargar el overLib, coloque el archivo overlib.js dentro de su document root e indique la ruta relativa en el parámetro "src" de {popup\_init}.

**Example 8.21. popup\_init**

```

<head>
{* popup_init debe ser llamado una sola vez hasta arriba de la pagina *}
{popup_init src="/javascripts/overlib.js"}
</head>

```

**popup**

Nombre del Atributo	Tipo	Requerido	Default	Descripción
text	string	Si	<i>n/a</i>	El text/html para mostrar en la ventana popup
trigger	string	No	<i>onMouseOver</i>	El que va a ser usado para que aparezca la ventana. Puede ser onMouseOver u onClick
sticky	boolean	No	<i>false</i>	Hace que el poppup se quede cerca hasta que se cierre
caption	string	No	<i>n/a</i>	Defíne el texto para el título
fgcolor	string	No	<i>n/a</i>	El color que va a ser usado dentro de la caja popup
bgcolor	string	No	<i>n/a</i>	El color del borde de la caja popup
textcolor	string	No	<i>n/a</i>	Defíne el color del texto dentro de la caja popup
capcolor	string	No	<i>n/a</i>	Defíne el color del título de la caja
closecolor	string	No	<i>n/a</i>	Defíne el color del texto para cerrar
textfont	string	No	<i>n/a</i>	Defíne el color del texto para ser usado en el texto principal
captionfont	string	No	<i>n/a</i>	Defíne el tipo de letra para ser usado en el Título

Nombre del Atributo	Tipo	Requerido	Default	Descripción
closefont	string	No	<i>n/a</i>	Defíne el tipo de letra para el texto "Close"
textsize	string	No	<i>n/a</i>	Defíne el tipo de letra del texto principal
captionsize	string	No	<i>n/a</i>	Defíne el tamaño del tipo de letra del título
closesize	string	No	<i>n/a</i>	Defíne el tamaño del tipo de letra del texto "Close"
width	integer	No	<i>n/a</i>	Defíne el ancho de la caja
height	integer	No	<i>n/a</i>	Defíne la altura de la caja
left	boolean	No	<i>false</i>	Hace que el popups vaya para la izquierda del ratón
right	boolean	No	<i>false</i>	Hace que el popups vaya para la derecha del ratón
center	boolean	No	<i>false</i>	Hace que el popups vaya al centro del ratón
above	boolean	No	<i>false</i>	Hace que el popups vaya por encima del rató. NOTA:solamente es posible si el height fue definido
below	boolean	No	<i>false</i>	Hace que el popups vaya por abajo del ratón
border	integer	No	<i>n/a</i>	Torna en gruesos o finos los bordes del popups
offsetx	integer	No	<i>n/a</i>	A que distancia del ratón aparecera el popup, horizontalmente
offsety	integer	No	<i>n/a</i>	A que distancia del ratón aparecera el popup, verticalmente

Nombre del Atributo	Tipo	Requerido	Default	Descripción
fgbackground	url to image	No	<i>n/a</i>	Defíne una imagen para usar en vez del color del popup.
bgbackground	url to image	No	<i>n/a</i>	defíne una imagen para ser usada como borde en vez de un color para el popup. NOTA:Usted debe definir bgcolor como "" o el color aparecera también. NOTA: Cuando tuviera un link "Close", el Netscape rediseñara las celdas de la tabla, haciendo que las cosas aparezcan incorrectamente
closetext	string	No	<i>n/a</i>	Defíne el texto "Close" a otra cosa
noclose	boolean	No	<i>n/a</i>	No muestra el texto "Close" pegado con el título
status	string	No	<i>n/a</i>	Defíne el texto en la barra de estado del navegador
autostatus	boolean	No	<i>n/a</i>	Defíne el texto en la barra de estado para el texto del popup. NOTA: sobrescribe la definición del status.
autostatuscap	string	No	<i>n/a</i>	Defíne el texto de la barra de estado como el texto del título NOTA: sobrescribe el status y autostatus
inarray	integer	No	<i>n/a</i>	Indica al overLib desde que índice de la matriz ol_text debe leer el texto, localizada en overlib.js. Este parámetro puede ser

Nombre del Atributo	Tipo	Requerido	Default	Descripción
				usado en vez del texto
caparray	integer	No	<i>n/a</i>	Indica al overLib a partir de que índice de la matriz ol_caps leer el título
capicon	url	No	<i>n/a</i>	Muestra la imagen antes del título
snapx	integer	No	<i>n/a</i>	Instantanea el popup a una posición constante en una cuadrícula horizontal
snapy	integer	No	<i>n/a</i>	Instantanea el popup a una posición constante en una cuadrícula vertical
fixx	integer	No	<i>n/a</i>	Cierra el popups en una posición horizontal Nota: pasa por encima de otros colocados horizontal
fixy	integer	No	<i>n/a</i>	Cierra popups en posición vertical Note: pasa por encima de otros colocados vertical
background	url	No	<i>n/a</i>	Defíne una imagen para ser usada como fondo en vez de la tabla
padx	integer,integer	No	<i>n/a</i>	Rellena el fondo de la imagen con espacios en blanco horizontal para colocar el texto. Nota: este es un comando de dos parámetros
pady	integer,integer	No	<i>n/a</i>	Rellena el fondo de la imagen con espacios en blanco vertical para colocar el texto. Nota: este es un

Nombre del Atributo	Tipo	Requerido	Default	Descripción
				comando de dos parámetros
fullhtml	boolean	No	<i>n/a</i>	Permite a usted controlar completamente el html sobre la figura de fondo. El código HTML es esperado en el atributo "text"
frame	string	No	<i>n/a</i>	Controla popups en frames diferentes. Para mayores informes sobre esta función vea la pagina de overlib
timeout	string	No	<i>n/a</i>	LLama específicamente a una función javascript y toma el valor que retorna, como el texto que se va a mostrar en la ventana popup
delay	integer	No	<i>n/a</i>	Hace que el popup funcione como un tooltip. Este aparecera solo con un retraso en milesimas de segundo
hauto	boolean	No	<i>n/a</i>	Determina automáticamente si el popup debe aparecer a la izquierda o a la derecha del ratón.
vauto	boolean	No	<i>n/a</i>	Determina automáticamente si el popup debe aparecer abajo o arriba del ratón.

{popup} es usado para crear ventanas popup con javascript. {popup\_init} DEBE ser llamado primero para poder trabajar.

### Example 8.22. popup

```
{* popup_init debe ser llamado una vez hasta arriba de la pagina *}
{popup_init src="/jascripts/overlib.js"}

{* crea un link con una ventana popup que aparece cuando se pasa el mouse sobre es
<a href="mypage.html" {popup text="This link takes you to my page!"}>mypage</a>

{* usted puede usar html, links, etc en el texto del popup *}
<a href="mypage.html" {popup sticky=true caption="mypage contents"
text="<ul><li>links</li><li>pages</li><li>images</li></ul>"
snapx=10 snapy=10}>mypage</a>
```

Ver también {popup\_init} y overLib [<http://www.bosrup.com/web/overlib/>].

## {textformat}

Nombre del Atributo	Tipo	Requerido	Default	Descripción
style	string	No	<i>n/a</i>	estilo pre-definido
indent	number	No	<i>0</i>	Número de caracteres para endentar cada línea.
indent_first	number	No	<i>0</i>	Número de caracteres para endentar la primera línea
indent_char	string	No	<i>(single space)</i>	El carácter (o cadena de caracteres) para endentar
wrap	number	No	<i>80</i>	Cuantos caracteres tendrá cada línea
wrap_char	string	No	<i>\n</i>	Caracter (o cadena de caracteres) a usar para saltar cada línea
wrap_cut	boolean	No	<i>false</i>	Si es true, wrap saltará la línea en el carácter exacto en vez de saltar al final de la palabra.
assign	string	No	<i>n/a</i>	La variable del template que recibirá la salida



`{textformat}` es una función de bloque usada para formatear texto. Básicamente limpia espacios y caracteres especiales, y formatea los párrafos cortando el texto al final de la palabra y endentando líneas.

Usted puede definir los parámetros explícitamente, o usar un estilo pre-definido. Actualmente el único estilo disponible es "email".

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

```
This is foo.
```

```
This is foo.
```

Custom Functions

```
This is foo.
```

```
This is foo.
```

**Example 8.23. {textformat}**

```
This is foo.
```

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

```
bar foo bar foo    foo.
This is bar: foo
{textformat wrap=40 indent=4}
```

Salida del ejemplo de arriba:

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

```
This is foo.
{/textformat}
This is foo.
```

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

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

Salida del ejemplo de arriba:

```
foo. bar foo bar foo bar foo bar foo
bar foo bar foo    foo. bar foo
This is bar: bar foo bar foo foo.
bar foo bar foo    foo.
bar foo bar foo    bar foo bar
This is foo.
{/textformat}
This is foo.
```

```
This is foo.
```

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

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

Salida del ejemplo de arriba:

```
foo. bar foo bar foo bar foo bar foo
bar foo bar foo    foo. bar
foo foo. bar foo bar foo foo. bar
This is bar: foo. bar foo bar foo
bar foo bar foo    foo. bar foo
bar foo bar foo    bar foo bar foo
bar foo bar foo    foo.
This is foo.
{/textformat}
This is foo.
```

```
This is foo.
```

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

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

Salida del ejemplo de arriba:

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

Ver también `{strip}` y `{wordwrap}`.

---

# Chapter 9. Config Files

Los archivos de configuración son útiles para diseñar y administrar variables globales para los templates a partir de un archivo. Un ejemplo son los colores del template. Normalmente si usted quiere cambiar el esquema de colores de una aplicación, usted debe ir a cada uno de los archivos del template y cambiar los colores. Con un archivo de configuración, los colores pueden estar mantenidos en un lugar y solo necesita actualizar este para cambiar los colores.

## Example 9.1. Ejemplo de sintaxis de un archivo de configuración

```
# 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
```

Los valores de las variables pueden estar entre comillas, mas no es necesario. Usted puede usar comillas simples o dobles. Si usted tuviera un valor que ocupe mas de una linea, coloque todo el valor entre tres comillas ("""). Usted puede colocar comentarios en un archivo de configuración con cualquier sintaxis que no sea valida en un archivo de configuración. Nosotros recomendamos usar un # en el princio de cada linea.

Este archivo de configuración tiene dos secciones. Los nombres de secciones debe estar entre corchetes []. Los nombres de sección pueden ser cadenas arbitrarias que no contengan los simbolos [ or ]. Las cuatro variables en la cabecera son variables globales, o no son variables de sección. Estas variables son siempre cargadas del archivo de configuración. Si una sección en particular fuera cargada, entonces las variables globales y las variables de esta sección son cargadas. Si una variable existe como global y dentro de una sección, la variable de sección será utilizada. Si usted tuviera dos variables en la misma sección con el mismo nombre, la ultima será utilizada.

Los archivos de configuración son cargados en el template con una función incrustada {**config\_load**}. (Ver También **config\_load()** ).

Usted puede ocultar variables o secciones enteras colocando un punto antes del nombre de la variable. Esto es útil si su aplicación lee los archivos de configuración y los datos sensibles a partir de ellos que la herramienta del template no lo necesita. Si usted tiene a otras personas realizando la edición de templates, usted tendrá la certeza que ellos no leerán datos sensibles del archivo de configuración cargando estos en el template.

Ver También `{config_load}`, `$config_overwrite`, `get_config_vars()`, `clear_config()` y `config_load()`

---

# Chapter 10. Debugging Console

Incluso en Smarty existe una consola para debug. La consola informa a usted de todos los templates incluidos, las variables definidas y las variables de archivos de configuración de la llamada actual del template. Incluso un template llamado "debug.tpl" viene con la distribución de Smarty el cual controla el formateo de la consola. Defina `$debugging` en `true` en el Smarty, y si es necesario defina `$debug_tpl` para la ruta del recurso debug.tpl (Esto es `SMARTY_DIR` por default). Cuando usted carga una pagina, una consola en javascript abrira una ventana popup y dara a usted el nombre de todos los templates incluidos y las variables definidas en la pagina actual. Para ver las variables disponibles para un template en particular, vea la función `{debug}`. Para deshabilitar la consola del debug, defina `$debugging` en `false`. Usted puede activar temporalmente la consola del debug colocando `SMARTY_DEBUG` en la URL si usted activo esta opción con `$debugging_ctrl`.

## Nota Técnica

La consola de debug no funciona cuando usted usa la API `fetch()`, solo cuando estuviera usando `display()`. Es un conjunto de comandos javascript adicionados al final del template generado. Si a usted no le gusta el javascript, usted puede editar el template debug.tpl para formatear la salida como usted quiera. Los datos del debug no son guardados en cache y los datos del debug.tpl no son incluidos en la consola debug.

## Note

Los tiempos de carga de cada template y de archivos de configuración son en segundos, o en fracciones de segundo.

Vea también `troubleshooting`, `$error_reporting` y `trigger_error()`.

---

# **Part III. Smarty For Programmers**

---

---

## Table of Contents

11. Constantes .....	135
SMARTY_DIR .....	135
SMARTY_CORE_DIR .....	135
12. Clase Variables de Smarty .....	136
\$template_dir .....	136
\$compile_dir .....	136
\$config_dir .....	136
\$plugins_dir .....	136
\$debugging .....	137
\$debug_tpl .....	137
\$debugging_ctrl .....	137
\$autoload_filters .....	137
\$compile_check .....	138
\$force_compile .....	138
\$caching .....	138
\$cache_dir .....	138
\$cache_lifetime .....	139
\$cache_handler_func .....	139
\$cache_modified_check .....	139
\$config_overwrite .....	139
\$config_booleanize .....	140
\$config_read_hidden .....	140
\$config_fix_newlines .....	140
\$default_template_handler_func .....	140
\$php_handling .....	141
\$security .....	141
\$secure_dir .....	141
\$security_settings .....	141
\$trusted_dir .....	142
\$left_delimiter .....	142
\$right_delimiter .....	142
\$compiler_class .....	142
\$request_vars_order .....	142
\$request_use_auto_globals .....	142
\$error_reporting .....	143
\$compile_id .....	143
\$use_sub_dirs .....	143
\$default_modifiers .....	144
\$default_resource_type .....	144
13. La clase Methods() de Smarty .....	145
append() .....	146
append_by_ref .....	147
assign() .....	148
assign_by_ref .....	149
clear_all_assign() .....	150
clear_all_cache .....	151
clear_assign() .....	152
clear_cache() .....	153
clear_compiled_tpl() .....	154
clear_config() .....	155
config_load() .....	156



display()	157
fetch()	159
get_config_vars()	162
get_registered_object()	163
get_template_vars()	164
is_cached()	165
load_filter()	167
register_block()	168
register_compiler_function	169
register_function()	170
register_modifier()	172
register_object()	173
register_outputfilter()	174
register_postfilter()	175
register_prefilter()	176
register_resource	177
trigger_error	178
template_exists()	179
unregister_block	180
unregister_compiler_function()	181
unregister_function()	182
unregister_modifier()	183
unregister_object()	184
unregister_outputfilter()	185
unregister_postfilter()	186
unregister_prefilter()	187
unregister_resource()	188
14. Cache	189
Configurando el Cache	189
Multiples caches por pagina	192
Cache Groups	193
Controlando salida de Cacheabilidad de plugins	194
15. Caracteristicas Avanzadas	197
Objetos	197
Prefilters	198
Postfilters	199
Filtros de salida	200
Función manipuladora de cache	200
Recursos	202
Templates desde \$template_dir	202
Templates partiendo de cualquier directorio	202
Templates partiendo de otras fuentes	203
Función manipuladora de Template por default	205
16. Extendiendo Smarty con plugins	206
Como funcionan los Plugins	206
Nombres convencionales	206
Escribiendo Plugins	207
Funciones de Template	207
Modificadores	209
Block Functions	211
Funciones Compiladoras	212
Prefiltros/Postfiltros	213
Filtros de Salida	215
Fuentes	216

Inserts ..... 219

---

# Chapter 11. Constantes

## SMARTY\_DIR

Esta debe ser la ruta completa del path para la localización de los archivos de clases de Smarty. Si esta no fuera definida, Entonces Smarty intentara determinar el valor apropiado automáticamente. Si es definido, el path **debe finalizar con una diagonal**.

### Example 11.1. SMARTY\_DIR

```
<?php
// set path to Smarty directory *nix style
define('SMARTY_DIR', '/usr/local/lib/php/Smarty/libs/');

// path to Smarty windows style
define('SMARTY_DIR', 'c:/webroot/libs/Smarty/libs/');

// hack (not recommended) that works on both *nix and wind
// Smarty is assumend to be in 'includes' dir under script
define('SMARTY_DIR', str_replace("\\", "/", getcwd()).'/includes/Smarty/libs/');

// include the smarty class Note 'S' is upper case
require_once(SMARTY_DIR.'Smarty.class.php');
?>
```

Ver también \$smarty.const y \$php\_handling constants

## SMARTY\_CORE\_DIR

Esta debe ser la ruta completa de localización del sistema de archivos de Smarty core. Si no es definido, Smarty tomara por default esta constante de *libs/* bajo el sub-directory SMARTY\_DIR. Si es definida, la ruta debe terminar con una diagonal. Use esta constante cuando necesite incluir manualmente algun archivo de core.\*

### Example 11.2. SMARTY\_CORE\_DIR

```
<?php

// load core.get_microtime.php
require_once(SMARTY_CORE_DIR.'core.get_microtime.php');

?>
```

Ver también \$smarty.const

---

# Chapter 12. Clase Variables de Smarty

## **\$template\_dir**

Este es el nombre por default del directorio del template. Si usted no proporciona un tipo de recurso que incluya archivos, entonces estos se encontraran aquí. Por default `./templates`, esto significa que lo buscara en el directorio del `templates` en el mismo directorio que esta ejecutando el script PHP.

### **Nota Técnica**

No es recomendado colocar este directorio bajo el directorio document root de su servidor web.

## **\$compile\_dir**

Ese es el nombre del directorio donde los templates compilados están localizados, Por default están en `./templates_c`, esto significa que lo buscara en el directorio de templates en el mismo directorio que esta ejecutando el script php. **Este directorio debe tener permiso de escritura para el servidor web** (ver la instalación). también `$use_sub_dirs`.

### **Nota Técnica**

Esa configuración debe ser un path relativo o un path absoluto. `include_path` no se usa para escribir archivos.

### **Nota Técnica**

No es recomendado colocar este directorio bajo el directorio document root de su servidor web.

Ver también `$compile_id` y `$use_sub_dirs`.

## **\$config\_dir**

Este es el directorio usado para almacenar archivos de configuración usados en los templates. El default es `./configs`, esto significa que lo buscara en el directorio de templates en el mismo directorio que esta ejecutando el script php.

### **Nota Técnica**

No es recomendado colocar este directorio bajo el directorio document root de su servidor web.

## **\$plugins\_dir**

Este es el directorio (o directorios) donde Smarty procurara ir a buscar los plugins que sean necesarios. El default es `plugins` bajo el `SMARTY_DIR`. Si usted proporciona un path relativo, Smarty procurara ir primero bajo el `SMARTY_DIR`, Entonces relativo para el `cwd`(current working directory), Entonces relativo para cada entrada de su PHP `include path`. Si `$plugins_dir` es un arreglo de directorios, Smarty buscara los plugins para cada directorio de plugins en el orden en el que fueron proporcionados.

## Nota Técnica

Para un mejor funcionamiento, no configure su `plugins_dir` para que use el include path PHP. Use un path absoluto, o un path relativo para `SMARTY_DIR` o el `cwd` (current working directory).

### Example 12.1. multiple `$plugins_dir`

```
<?php

$smarty->plugins_dir = array(
    'plugins', // the default under SMARTY_DIR
    '/path/to/shared/plugins',
    '../../../../includes/my/plugins'
);

?>
```

## `$debugging`

Este habilita el debugging console. La consola es una ventana de javascript que informa a usted sobre los archivos del template incluidos y las variables destinadas a la pagina del template actual.

Ver también `{debug}`, `$debug_tpl`, y `$debugging_ctrl`

## `$debug_tpl`

Este es el nombre del archivo de template usado para el debug de la consola. Por default, es nombrado `debug.tpl` y esta localizado en el `SMARTY_DIR`.

Ver también `$debugging` y `Debugging console`

## `$debugging_ctrl`

Esto permite rutas alternativas para habilitar el debug. `NONE` no significa que métodos alternativos son permitidos. `URL` significa cuando la palabra `SMARTY_DEBUG` fue encontrada en el `QUERY_STRING`, que el debug está habilitado para la llamada del script. Si `$debugging` es `true`, ese valor es ignorado.

Ver también `Debugging console`.

## `$autoload_filters`

Si existe algun filtro que usted desea cargar en cada llamada de template, usted puede especificar cual variable usar y el Smarty ira automáticamente a cargarlos para usted. La variable es un arreglo asociativo donde las llaves son tipos de filtro y los valores son arreglos de nombres de filtros. Por ejemplo:

```
<?php
$smarty->autoload_filters = array('pre' => array('trim', 'stamp'),
                                'output' => array('convert'));
?>
```

Ver también `register_outputfilter()`, `register_prefilter()`, `register_postfilter()` y `load_filter()`

## \$compile\_check

En cada llamada de la aplicación PHP, Smarty prueba para ver si el template actual fue modificado (diferentes time stamp) desde la última compilación. Si este fue modificado, se recompilará el template. Si el template no fue compilado, este irá a compilar de cualquier manera esa configuración. Por default esta variable es determinada como true. Una vez que la aplicación está en producción (los templates no serán modificados), el paso `compile_check` no es necesario. asegúrese de determinar `$compile_check` a "false" para un mejor funcionamiento. Note que si usted modifica está para "false" y el archivo de template está modificado, usted \*no\* verá los cambios desde el template hasta que no sea recompilado. Si el caching está habilitado y `compile_check` está habilitado, entonces los archivos de cache tienen que ser regenerados si el archivo de template es muy complejo o el archivo de configuración fue actualizado. vea `$force_compile` o `clear_compiled_tpl()`.

## \$force\_compile

Este fuerza al Smarty a (re)compilar templates en cada llamada. Esta configuración sobrescribe `$compile_check`. Por default este es deshabilitado. Es útil para el desarrollo y debug. Nunca debe ser usado en ambiente de producción. Si el cache está habilitado, los archivo(s) de cache serán regenerados todo el tiempo.

## \$caching

Este informa al Smarty si hay o no salida de cache para el template en el `$cache_dir`. Por default tiene asignado 0, o deshabilitado. Si su template genera contenido redundante, es necesario ligar el `$caching`. Esto tendrá un beneficio significativo en el rendimiento. Usted puede tener múltiples caches para el mismo template. Un valor de 1 o 2 `caching` habilitados. 1 anuncia a Smarty para usar la variable actual `$cache_lifetime` hasta determinar si el cache expira. Un valor 2 anuncia a Smarty para usar el valor `$cache_lifetime` al tiempo en que le cache fue generado. De esta manera usted puede determinar el `$cache_lifetime` inmediatamente antes de buscar el template para tener el control cuando este cache en particular expira. Vea también `is_cached()`.

Si `$compile_check` está habilitado, el contenido del cache se regenerará si alguno de los dos templates o archivos de configuración que son parte de este cache estuviera modificado. Si `$force_compile` está habilitado, el contenido del cache siempre será regenerado.

Ver También `$cache_dir`, `$cache_lifetime`, `$cache_handler_func`, `$cache_modified_check` y `Caching section`.

## \$cache\_dir

Este es el nombre del directorio donde los caches del template son almacenados. Por default es `./cache`, esto significa que buscará el directorio de cache en el mismo directorio que ejecuta el script PHP. **Este directorio debe ser regrabable por el servidor web** (ver instalación). Usted puede usar también

su propia función habitual de mantenimiento de cache para manipular los archivos de cache, que ignorará esta configuración. Ver también `$use_sub_dirs`.

### **Nota Técnica**

Esta configuración debe ser cualquiera de estas dos, un path relativo o absoluto. `include_path` no es usado para escribir archivos.

### **Nota Técnica**

No es recomendado colocar este directorio bajo el directorio document root del servidor web.

Ver también `$caching`, `$use_sub_dirs`, `$cache_lifetime`, `$cache_handler_func`, `$cache_modified_check` y Caching section.

## **`$cache_lifetime`**

Este es la duración del tiempo en segundos que un cache de template es valido. Una vez que este tiempo está expirado, el cache sera regenerado. `$caching` debe ser asignado a "true" para `$cache_lifetime` hasta tener algún propósito. Un valor de -1 forza el cache a nunca expirar. Un valor de 0 forzara a que el cache sea siempre regenerado (bueno solo para probar, el método mas eficiente para desabilitar cache es asignar `$caching = false`.)

Si `$force_compile` está habilitado, los archivos de cache serán regenerados todo el tiempo, efectivamente desabilitando `caching`. Usted puede limpiar todos los archivos de cache con la función `clear_all_cache()`, o archivos individuales de cache (o grupos) con la función `clear_cache()`.

### **Nota Técnica**

Si usted quisiera dar a ciertos templates su propio tiempo de vida de cache, usted puede hacer esto asignando `$caching = 2`, entonces determina `$cache_lifetime` a un único valor justo antes de llamar `display()` o `fetch()`.

## **`$cache_handler_func`**

Usted puede proporcionar una función por default para manipular archivos de cache en vez de usar el metodo incrustado usando el `$cache_dir`. Para mas detalles vea la sección `cache handler function section`.

## **`$cache_modified_check`**

Si es asignado true, Smarty respetara el If-Modified-Since encabezado enviado para el cliente. Si el timestamp del archivo de cache no fue alterado desde la ultima visita, entonces un encabezado "304 Not Modified" sera enviado en vez del contenido. Esto funciona solamente en archivos de cache sin etiquetas `{insert}`.

Ver también `$caching`, `$cache_lifetime`, `$cache_handler_func`, y Caching section.

## **`$config_overwrite`**

Si es asignado true, las variables leidas en el archivo de configuración se sobrescribieran unas a otras. De lo contrario, las variables seran guardadas en un arreglo. Esto es útil si usted quiere almacenar arreglos de datos en archivos de configuración, solamente lista tiempos de cada elemento múltiplo. true por default.

## Example 12.2. Arreglo de variables de configuración

Este ejemplo utiliza `{cycle}` para la salida a una tabla alternando colores por renglon rojo/verde/azul con `$config_overwrite = false`.

El archivo de configuración.

```
# row colors
rowColors = #FF0000
rowColors = #00FF00
rowColors = #0000FF
```

El template con ciclo `{section}`.

```
<table>
  {section name=r loop=$rows}
  <tr bgcolor="{cycle values=#rowColors#}">
    <td> ....etc.... </td>
  </tr>
{/section}
</table>
```

Ver también `{config_load}`, `config files`, `get_config_vars()`, `clear_config()` y `config_load()`.

## `$config_booleanize`

Si es asignado `true`, los valores del archivo de configuración de `on/true/yes` y `off/false/no` se convierten en valores booleanos automáticamente. De esta forma usted puede usar los valores en un template como: `{if #foobar#} ... {/if}`. Si `foobar` estuviera `on`, `true` o `yes`, la condición `{if}` se ejecutara. `true` es el default.

## `$config_read_hidden`

Si es asignado `true`, esconde secciones (nombres de secciones comenzando con un periodo) en el archivo de configuración pueden ser leídos del template. Tipicamente desearia esto como `false`, de esta forma usted puede almacenar datos sensibles en el archivo de configuración como un parámetro de base de datos y sin preocuparse que el template los cargue. `false` es el default.

## `$config_fix_newlines`

Si es asignado `true`, `mac` y dos `newlines` (`\r` y `\r\n`) en el archivo de configuración serán convertidos a `\n` cuando estos fueran interpretados. `true` es el default.

## `$default_template_handler_func`

Esta función es llamada cuando un template no puede ser obtenido desde su recurso.



## \$php\_handling

Este informa al Smarty como manipular códigos PHP contenidos en los templates. Hay cuatro posibles configuraciones, siendo el default `SMARTY_PHP_PASSTHRU`. Observe que esto NO afectara los códigos php dentro de las etiquetas `{php}{/php}` en el template.

- `SMARTY_PHP_PASSTHRU` - Smarty echos tags as-is.
- `SMARTY_PHP_QUOTE` - Smarty abre comillas a las etiquetas de entidades html.
- `SMARTY_PHP_REMOVE` - Smarty borra las etiquetas del template.
- `SMARTY_PHP_ALLOW` - Smarty ejecuta las etiquetas como código PHP.

### Note

Incrustar codigo PHP dentro del template es sumamente desalentador. Use custom functions o modifiers en vez de eso.

## \$security

`$security true/false`, el default es `false`. Security es bueno para situaciones cuando usted tiene partes inconfiables editando el template (via ftp por ejemplo) y usted quiere reducir los riesgos de comportamiento de seguridad del sistema a través del lenguaje del template. Al habilitar la seguridad fuerza las siguientes reglas del lenguaje del template, a menos que especifique control con `$security_settings`:

- Si `$php_handling` está asignado a `SMARTY_PHP_ALLOW`, este es implícitamente cambiado a `SMARTY_PHP_PASSTHRU`
- Las funciones PHP no son permitidas en sentencias `{if}`, excepto aquellas que esten especificadas en `$security_settings`
- Los templates solo pueden ser incluidos en el directorio listado en arreglo `$secure_dir`
- Los archivos locales solamente pueden ser traídos del directorio listado en `$secure_dir` usando el arreglo `{fetch}`
- Estas etiquetas `{php}{/php}` no son permitidas
- Las funciones PHP no son permitidas como modificadores, excepto si estan especificados en el `$security_settings`

## \$secure\_dir

Este es un arreglo de todos los directorios locales que son considerados seguros. `{include}` y `{fetch}` usan estos (directorios) cuando security está habilitada.

Ver también Security settings, y `$trusted_dir`.

## \$security\_settings

Estas son usadas para cancelar o especificar configuraciones de seguridad cuando security esta habilitado. Estas son las posibles configuraciones.

- `PHP_HANDLING` - true/false. la configuracion de `$php_handling` no es checada por security.
- `IF_FUNCS` - Este es un arreglo de nombres de funciones PHP permitidas en los bloques IF.
- `INCLUDE_ANY` - true/false. Si es asignado true, algun template puede ser incluido para un archivo de sistema, a pesar de toda la lista de `$secure_dir`.
- `PHP_TAGS` - true/false. Si es asignado true, las etiquetas `{php}{/php}` son permitidas en los templates.
- `MODIFIER_FUNCS` - Este es un arreglo de nombres de funciones PHP permitidas usadas como modificadores de variables.
- `ALLOW_CONSTANTS` - true/false. Si es asignado true, la constante a travez de `{Smarty.const.name}` es autorizada en el template. El default es asignado "false" por seguridad.

## **`$trusted_dir`**

`$trusted_dir` solamente es usado cuando `$security` está habilitado. Este es un arreglo de todos los directorios que son considerados confiables. Los directorios confiables son de donde usted extraera sus script PHP que son ejecutados directamente desde el template con `{include_php}`.

## **`$left_delimiter`**

Este es el delimitador izquierdo usado por el lenguaje de template. El default es "`{`".

Ver también `$right_delimiter` y `escaping smarty parsing`.

## **`$right_delimiter`**

Este es el delimitador derecho usado por el lenguaje de template. El default es "`}`".

ver también `$left_delimiter` y `escaping smarty parsing`.

## **`$compiler_class`**

Especifica el nombre del compilador de clases que Smarty usara para compilar los templates. El default es 'Smarty\_Compiler'. Solo para usuarios avanzados.

## **`$request_vars_order`**

El orden en el cual las variables requeridas seran registradas, similar al `variables_order` en el `php.ini`

Ver también `$smarty.request` y `$request_use_auto_globals`.

## **`$request_use_auto_globals`**

Especifica si el Smarty debe usar variables globales del php `$HTTP_*_VARS[]` (`$request_use_auto_globals=false` valor por default) o `$_*[]` (`$request_use_auto_globals=true`). Esto afecta a los templates que hacen uso de `{Smarty.request.*}`, `{Smarty.get.*}` etc. . Atención: Si usted asigna `$request_use_auto_globals` a true, `variable.request.vars.order` no tendran efecto los valores de configuracion de php `gpc_order` sera usados.

## \$error\_reporting

Cuando este valor es asignado a non-null-value este valor es usado como un nivel error\_reporting [[http://php.net/error\\_reporting](http://php.net/error_reporting)] dentro de display() y fetch(). Cuando debugging es habilitado este valor es ignorado y el error-level no es tocado.

Ver también trigger\_error(), debugging y Troubleshooting.

## \$compile\_id

Identificador de compilación persistente. Como una alternativa para pasar el mismo compile\_id a cada llamada de función, usted puede asignar este compile\_id y este será usado implícitamente después.

Con el compile\_id usted puede trabajar con limitacion porque usted no puede usar el mismo \$compile\_dir para diferentes \$template\_dirs. Si usted asigna distintos compile\_id para cada template\_dir entonces Smarty puede hacer la compilacion de los templates por cada compile\_id.

Si usted tiene por ejemplo un prefilter este localiza su template (es decir: traduce al lenguaje las dependencias por partes) y lo compila, entonces usted debe usar el lenguaje actual como \$compile\_id y usted obtendrá un conjunto de plantillas compiladas para cada idioma que usted utilice.

otro ejemplo puede ser si usa el mismo directorio para compilar multiples dominios / multiples host virtuales.

### Example 12.3. \$compile\_id

```
<?php
$smarty->compile_id = $_SERVER['SERVER_NAME'];
$smarty->compile_dir = 'path/to/shared_compile_dir';
?>
```

## \$use\_sub\_dirs

Smarty puede crear subdirectorios bajo los directorios templates\_c y cache si \$use\_sub\_dirs es asignado true. En un ambiente donde hay potencialmente decenas de miles de archivos creados, esto puede ayudar la velocidad de sistema de archivos. Por otro lado, algunos ambientes no permiten que procesos de PHP creen directorios, este debe ser desabilitado. El valor por default es false (disabled). Los Sub directorios son mas eficientes, entonces aprovechelo si puede.

Teóricamente usted obtiene mayor eficiencia en sun sistema de archivos con 10 directorios que contengan 100 archivos, que con un directorio que contenga 1000 archivos. Este era ciertamente un caso con Solaris 7 (UFS)... con un nuevo sistema de archivos como ext3 y un levantado especial, la diferencia es casi nula.

### Nota Técnica

\$use\_sub\_dirs=true doesn't trabaja con safe\_mode=On [<http://php.net/features.safe-mode>], esto es porque es switchable y porque puede estar en off por default.

## Nota Técnica

Desde Smarty-2.6.2 `use_sub_dirs` esta por default en false.

Ver también `$compile_dir`, y `$cache_dir`.

## **\$default\_modifiers**

Este es un arreglo de modificadores implícitamente aplicados para cada variable en el template. Por Ejemplo, cada variable HTML-escape por default, usa el arreglo('escape:"htmlall"'); Para hacer que las variables excenten los modificadores por default, pase el modificador especial "smarty" con un valor de parámetro "nodefaults" modificando esto, tal como `{$var|smarty:nodefaults}`.

## **\$default\_resource\_type**

Este anuncia a Smarty el tipo de recurso a usar implícitamente. El valor por default es 'file', significa que `$smarty->display('index.tpl')`; y `$smarty->display('file:index.tpl')`; son identicos en el significado. Para mas detalles vea el capitulo resource.

---

## **Chapter 13. La clase Methods() de Smarty**

## Name

append() — agregando elementos a una matriz asignada

## Descripción

```
void append(mixed var);

void append(string varname,
             mixed var,
             bool merge);
```

Este es usado para adicionar un elemento en un arreglo asignado. Si usted adiciona una cadena como valor, este se convertira en un valor del arreglo y entonces lo adiciona. Usted puede explicitamente pasar pares de nombres/valores, o arreglos asociativos conteniendo los pares nombre/valor. Si usted pasa el tercer parámetro opcional como true, el valor se únira al arreglo actual en vez de ser adicionado.

### Nota Tecnica

El parametro *merge* es la llave respectiva del arreglo, asi si usted asocia dos arreglos indexados numericamente, estos se sobre escriben uno al otro o tener como resultado llaves no-secuenciales. Este es diferente a la funcion `array_merge()` de PHP la cual limpia las llaves numericas y las vuelve a reenumerar.

### Example 13.1. append

```
<?php
// passing name/value pairs
$smarty->append("Name", "Fred");
$smarty->append("Address", $address);

// passing an associative array
$smarty->append(array('city' => 'Lincoln', 'state' => 'Nebraska'));
?>
```

Ver también `append_by_ref()`, `assign()` y `get_template_vars()`

## Name

append\_by\_ref — pasando valores por referencia

## Descripción

```
void append_by_ref(string varname,  
                  mixed var,  
                  bool merge);
```

Este es usado para pasar valores al template por referencia. Si usted pasa una variable por referencia entonces cambiara su valor, el valor asignado sufrira el cambio también. Para objetos, `append_by_ref()` también envía una copia en memoria del objeto adicionado. Vea el manual de PHP en referenciando variables para una mejor explicación del asunto. Si usted pasa el tercer parámetro en `true`, el valor será mezclado con el arreglo en ves de ser adicionado.

### Nota Tecnica

El parametro *merge* es la llave respectiva del arreglo, asi si usted asocia dos arreglos indexados numericamente, estos se sobre escriben uno al otro o tener como resultado llaves no-secuenciales. Este es diferente a la funcion `array_merge()` de PHP la cual limpia las llaves numericas y las vuelve a reenumerar.

### Example 13.2. append\_by\_ref

```
<?php  
// appending name/value pairs  
$smarty->append_by_ref("Name", $myname);  
$smarty->append_by_ref("Address", $address);  
?>
```

Ver también `append()` y `assign()`.

## Name

assign() — pasando valores para el template

## Descripción

```
void assign(mixed var);
```

```
void assign(string varname,  
             mixed var);
```

Usted puede explícitamente pasar pares de nombres/valores, o un arreglo asociativo conteniendo el par de nombre/valor.

### Example 13.3. assign()

```
<?php  
// pasando pares de nombre/valor  
$smarty->assign('Name', 'Fred');  
$smarty->assign('Address', $address);  
  
// pasando un arreglo asociativo  
$smarty->assign(array('city' => 'Lincoln', 'state' => 'Nebraska'));  
  
// pasando un row desde una base de datos (eg adodb)  
$sql = 'select id, name, email from contacts where contact = '.$sid;  
$smarty->assign('contact', $db->getRow($sql));  
?>
```

Accesando estos en el template con

```
{ $Name }  
{ $Address }  
{ $city }  
{ $state }  
  
{ $contact.id }, { $contact.name }, { $contact.email }  
?>
```

Para ver una asignacion de arreglos mas compleja {foreach} y {section}

Vea también assign\_by\_ref(), get\_template\_vars(), clear\_assign(), append() y {assign}



## Name

`assign_by_ref` — pasando valores por referencia

## Descripción

```
void assign_by_ref(string varname,  
                  mixed var);
```

Este es usado para asignar valores por referencia al template en vez de hacer una copia. Vea el manual de PHP en la parte sobre referencia de variables para una explicación mas detallada.

### Nota Técnica

Este es usado para asignar valores por referencia al template. Si usted asigna una variable por referencia entonces cambiara este valor, el valor asignado experimentara el cambio también. Para objetos, `assign_by_ref()` también existe una copia del objetos asignado en memoria. Vea el manual de PHP en refereciando variables para una mejor explicación.

### Example 13.4. `assign_by_ref()`

```
<?php  
// passing name/value pairs  
$smarty->assign_by_ref('Name', $myname);  
$smarty->assign_by_ref('Address', $address);  
?>
```

Ver también `assign()`, `clear_all_assign()`, `append()` y `{assign}`

## Name

`clear_all_assign()` — >limpia el valor de todas las variables asignadas

## Descripción

```
void clear_all_assign();
```

### Example 13.5. `clear_all_assign()`

```
<?php
// passing name/value pairs
$smarty->assign('Name', 'Fred');
$smarty->assign('Address', $address);

// will output above
print_r( $smarty->get_template_vars() );

// clear all assigned variables
$smarty->clear_all_assign();

// will output nothing
print_r( $smarty->get_template_vars() );
?>
```

Ver también `clear_assign()`, `clear_config()`, `assign()` y `append()`

## Name

`clear_all_cache` — limpia completamente el cache del template

```
void clear_all_cache(int expire_time);
```

Como un parámetro opcional, usted puede proporcionar un periodo minimo en segundos que el archivo de cache debe tener antes de ser anulado.

### Example 13.6. `clear_all_cache`

```
<?php
// clear the entire cache
$smarty->clear_all_cache();
?>
```

Ver también `clear_cache()`, `is_cached()` y `caching`

## Name

`clear_assign()` — limpia el valor de una variable asignada

## Descripción

```
void clear_assign(mixed var);
```

Este puede ser un valor simple, o un arreglo de valores.

### Example 13.7. `clear_assign()`

```
<?php
// clear a single variable
$smarty->clear_assign('Name');

// clear multiple variables
$smarty->clear_assign(array('Name', 'Address', 'Zip'));
?>
```

Ver también `clear_all_assign()`, `clear_config()`, `get_template_vars()`, `assign()` y `append()`

## Name

`clear_cache()` — Esto limpia el cache de un template específico

## Descripción

```
void clear_cache(string template,  
                 string cache_id,  
                 string compile_id,  
                 int expire_time);
```

Si usted tiene multiples caches en este archivo, usted puede limpiar un cache específico proporcionando el `cache_id` como segundo parámetro. Usted también puede pasar el `$compile_id` como un tercer parámetro. Usted puede "agrupar" templates conjuntamente de esta manera estos pueden ser removidos como un grupo. Vea el caching section para mayor información. Como un cuarto parámetro opcional, usted puede proporcionar un periodo mínimo en segundos que el archivo de cache debe tener antes de ser anulado.

### Example 13.8. `clear_cache()`

```
<?php  
// clear the cache for a template  
$smarty->clear_cache('index.tpl');  
  
// clear the cache for a particular cache id in an multiple-cache template  
$smarty->clear_cache('index.tpl', 'CACHEID');  
?>
```

Ver también `clear_all_cache()` y `caching`.

## Name

`clear_compiled_tpl()` — Esto limpia la vesion compilada del recurso de un template especifico

## Descripción

```
void clear_compiled_tpl(string tpl_file,  
                        string compile_id,  
                        int exp_time);
```

Esto limpia la versión compilada del recurso del template especificado, o todos los archivos de templates compilados si no fueron especificados. si usted pasa `$compile_id` solo sera compilado este template especificado `$compile_id` es limpiado. Si usted lo pasa con `ex_time`, entonces solo compilara los templates anteriores al `exp_time` segundo seran limpiados, por default todos los templates son compilados y limpiados independientemente de su tiempo de vida. Esta función es solo para uso avanzado, normalmente no es necesaria.

### Example 13.9. `clear_compiled_tpl()`

```
<?php  
// clear a specific template resource  
$smarty->clear_compiled_tpl("index.tpl");  
  
// clear entire compile directory  
$smarty->clear_compiled_tpl();  
?>
```

## Name

`clear_config()` — Esto limpia todas las variables de configuración

## Descripción

```
void clear_config(string var);
```

Esto limpia todas las variables de configuración asignadas. Si es proporcionado el nombre de una variable, solo esa variable es limpiada.

### Example 13.10. `clear_config()`

```
<?php
// clear all assigned config variables.
$smarty->clear_config();

// clear one variable
$smarty->clear_config('foobar');
?>
```

Ver también `get_config_vars()`, `config variables`, `config files`, `{config_load}`, `config_load()` y `clear_assign()`

## Name

`config_load()` — Carga el archivo de configuración y lo asigna al template

## Descripción

```
void config_load(string file,  
                 string section);
```

Esto carga el archivo de configuración de datos y lo asigna al template. Esto funciona idéntico a la función `{config_load}`.

### Nota Técnica

A partir de Smarty 2.4.0, las variables de template asignadas son mantenidas a través de `fetch()` y `display()`. Las variables de configuración cargadas de `config_load()` son siempre de alcance global. Los archivos de configuración también son compilados para ejecución rápida, y respetar el `force_compile` y `compile_check` de configuración.

### Example 13.11. `config_load()`

```
<?php  
// load config variables and assign them  
$smarty->config_load('my.conf');  
  
// load a section  
$smarty->config_load('my.conf', 'foobar');  
?>
```

Ver también `{config_load}`, `get_config_vars()`, `clear_config()`, y `config variables`



## Name

display() — Despliega el Template

## Descripción

```
void display(string template,  
             string cache_id,  
             string compile_id);
```

Este despliega el template diferente de fetch(). Cargando un tipo valido de path template resource. Como un segundo parámetro opcional, usted puede pasar un identificador de cache. Vea el caching section para mayor información.

Como tercer parametro opcional, usted puede pasar *compile\_id*. Este en el caso que usted quiera compilar diferentes versiones del mismo Tempalte, tal como tener separadas varios Templates compilados de diferentes lenguajes. Otro uso para *compile\_id* es cuando usted usa mas de un *\$template\_dir* pero solo un *\$compile\_dir*. Ponga separado *compile\_id* por cada *\$template\_dir*, de otra manera los tempate con el mismo nombre se sobre escibiran uno sobre otro. Uste puede poner también la variable *\$compile\_id* una vez en lugar de pasar esta por cada llamada a la función.

### Example 13.12. display()

```
<?php  
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 la sintaxis template resources para mostrar archivos fuera del directorio *\$template\_dir*.

### Example 13.13. Ejemplos de recursos de la función display

```
<?php
// absolute filepath
$smarty->display('/usr/local/include/templates/header.tpl');

// absolute filepath (same thing)
$smarty->display('file:/usr/local/include/templates/header.tpl');

// windows absolute filepath (MUST use "file:" prefix)
$smarty->display('file:C:/www/pub/templates/header.tpl');

// include from template resource named "db"
$smarty->display('db:header.tpl');
?>
```

Ver también `fetch()` y `template_exists()`.

## Name

fetch() — Retorna la salida del template

## Descripción

```
string fetch(string template,  
             string cache_id,  
             string $compile_id);
```

Este retorna la salida del template en vez de desplegarla. Proporcionando un tipo y path valido template resource. Como un segundo parámetro opcional, usted puede pasar el identificador de cache. vea el caching section para mayor información.

Como tercer parametro opcional, usted puede pasar *compile\_id*. Este en el caso que usted quiera compilar diferentes versiones del mismo Tempalte, tal como tener separadas varios Templates compilados de diferentes lenguajes. Otro uso para *compile\_id* es cuando usted usa mas de un *\$template\_dir* pero solo un *\$compile\_dir*. Ponga separado *compile\_id* por cada *\$template\_dir*, de otra manera los tempate con el mismo nombre se sobre escibiran uno sobre otro. Uste puede poner también la variable *\$compile\_id* una vez en lugar de pasar esta por cada llamada a la función.

### Example 13.14. fetch()

```
<?php
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;
?>
```

### Example 13.15. Usando fetch() y enviando a un e-mail

El template email\_body.tpl

```
Dear {$contact.name},

Welcome and thankyou for signing up as a member of our user group,

Click on the link below to login with your user name of '{$contact.login_id}'
so you can post in our forums.

http://{$smarty.server.SERVER_NAME}/index.php?page=login

List master
Some user group

{include file="email_disclaimer.tpl"}
```

El template email\_disclaimer.tpl usando el modificador {textformat}.

```
{textformat wrap=40}
Unless you are named "{$contact.name}", you may read only the "odd numbered
words" (every other word beginning with the first) of the message above. If you ha
violated that, then you hereby owe the sender 10 GBP for each even
numbered word you have read
{/textformat}
```

y el script de PHP usando la función mail() [<http://php.net/function.mail>]

```
<?php

// get contact from database eg using pear or adodb
$query = 'select name, email, login_id from contacts where contact_id='.$contact_id;
$contact = $db->getRow($sql);
$smarty->assign('contact', $contact);

mail($contact['email'], 'Subject', $smarty->fetch('email_body.tpl'));

?>
```

Ver también {fetch} display(), {eval}, y template\_exists().

## Name

`get_config_vars()` — retorna el valor asignado a la variable de configuración

## Descripción

```
array get_config_vars(string varname);
```

Si no tiene un parámetro asignado, un arreglo de todas las variables de los archivos de configuración es retornado.

### Example 13.16. `get_config_vars()`

```
<?php
// get loaded config template var '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);
?>
```

Ver también `clear_config()`, `{config_load}`, `config_load()` y `get_template_vars()`.

## Name

`get_registered_object()` — Este retorna una referencia para un objeto registrado.

## Descripción

```
array get_registered_object(string object_name);
```

Este es útil dentro de una función habitual cuando usted necesita acesar directamente a un objeto registrado. Ver [objects](#) para mas información;

### Example 13.17. `get_registered_object()`

```
<?php
function smarty_block_foo($params, &$smarty)
{
    if (isset($params['object'])) {
        // get reference to registered object
        $obj_ref = &$smarty->get_registered_object($params['object']);
        // use $obj_ref is now a reference to the object
    }
}
?>
```

Ver también [register\\_object\(\)](#), [unregister\\_object\(\)](#) y [objects section](#)

## Name

`get_template_vars()` — Retorna el valor asignado a una variable

## descripción

```
array get_template_vars(string varname);
```

Si no tiene un parámetro dado, un arreglo de todas las variables asignadas es retornado.

### Example 13.18. `get_template_vars`

```
<?php
// 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);
?>
```

Ver también `assign()`, `{assign}`, `assign_by_ref()`, `append()`, `clear_assign()`, `clear_all_assign()` y `get_config_vars()`



## Name

`is_cached()` — Retorna true si hay cache valido para ese template

## Descripción

```
bool is_cached(string template,
               string cache_id,
               string compile_id);
```

Esto solamente funciona si `caching` está asignado a `true`. ver también `caching` section.

### Example 13.19. `is_cached()`

```
<?php
$smarty->caching = true;

if(!$smarty->is_cached("index.tpl")) {
// do database calls, assign vars here
}

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

Usted también puede pasar un identificador de `$cache` como un segundo parámetro opcional en el caso que usted quiera multiples caches para el template dado.

Usted puede proporcionar el identificador como un tercer parametro opcional. Si usted omite ese parametro la persistencia del `$compile_id` es usada.

Si usted no quiere pasar el identificador de cache solamente quiere pasar el compile id debe pasar `null` como el identificador de cache.

### Example 13.20. `is_cached()` con templates con `multiple-cache`

```
<?php
$smarty->caching = true;

if(!$smarty->is_cached("index.tpl", "FrontPage")) {
// do database calls, assign vars here
}

$smarty->display("index.tpl", "FrontPage");
?>
```

## Nota técnica

Si `is_cached` retorna true el carga actualmente la salida del cache y lo guarda internamente. cualquier subsecuente llama a `display()` o `fetch()` y retorna este internamente guardando la salida y no intenta volver a cargar el archivo del cache. Esto previene una condicion de la carrera que puede ocurrir cuando un segundo proceso limpie el cache entre las llamadas a `is_cached` mostradas en el ejemplo de arriba. Esto significa tambien llamar al `clear_cache()` y otros cambios en el cache-settings que no tiene efecto despues que `is_cached` retorna true.

Ver también `clear_cache()`, `clear_all_cache()`, y `caching section`.

## Name

load\_filter() — Carga un filtro de plugin

## Descripción

```
void load_filter(string type,  
                 string name);
```

El primer argumento especifica el tipo de filtro a cargar y puede ser uno de los siguientes: 'pre', 'post', o 'output'. El segundo argumento especifica el nombre del filtro del plugin, por ejemplo, 'trim'.

### Example 13.21. loading filter plugins

```
<?php  
$smarty->load_filter('pre', 'trim');           // load prefilter named 'trim'  
$smarty->load_filter('pre', 'datefooter');    // load another prefilter named 'datef  
$smarty->load_filter('output', 'compress');   // load output filter named 'compress'  
?>
```

Ver también register\_prefilter(), register\_postfilter(), register\_outputfilter(), \$autoload\_filters y Advanced features.

## Name

register\_block() — Registra dinámicamente bloques de funciones de plugins

## Descripción

```
void register_block(string name,  
                    mixed impl,  
                    bool cacheable,  
                    mixed cache_attrs);
```

Use este para registrar dinámicamente bloques de funciones de plugins. Pase el bloque de nombres de función, seguido por una llamada de función PHP que implemente esto.

La llamada de una función-*php* *impl* puede ser cualquier (a) cadena conteniendo el nombre de la función o (b) un arreglo con el formato `array(&$object, $method)` con `&$object` siendo la referencia a un objeto y `$method` siendo una cadena conteniendo el nombre del método o (c) un arreglo con el formato `array(&$class, $method)` con `$class` siendo un nombre de clase y `$method` siendo un método de esta clase.

*cacheable* y *cache\_attrs* pueden ser omitidos en la mayoría de los casos. Vea Controlando modos de salida de cache de los plugins para saber como usar las propiedades.

### Example 13.22. register\_block()

```
<?php  
$smarty->register_block('translate', 'do_translation');  
  
function do_translation ($params, $content, &$smarty, &$repeat)  
{  
    if (isset($content)) {  
        $lang = $params['lang'];  
        // do some translation with $content  
        return $translation;  
    }  
}  
?>
```

Donde el template es:

```
{* template *}  
{translate lang="br"}  
Hello, world!  
{/translate}
```

Ver también `unregister_block()` y `Plugin Block Functions`.

## Name

register\_compiler\_function — Registra dinamicamente un plugin de una funcion compiladora

## Descripción

```
bool register_compiler_function(string name,  
                               mixed impl,  
                               bool cacheable);
```

Pase el nombre de la función compiladora, seguido por la función PHP que implemente esto.

La llamada a la funcion-php *impl* puede ser

- a. a una cadena conteniendo el nombre de la función
- b. un arreglo con la forma `array(&$object, $method)` con `&$object` siendo una referencia para un objeto y `$method` siendo una cadena conteniendo el nombre del método
- c. un arreglo con la forma `array(&$class, $method)` con `$class` siendo el nombre de una clase y `$method` siendo el método de esta clase.

*cacheable* puede ser omitido en la mayoría de los casos. Vea Controlando modos de Salida de Cache de los Plugins para obtener mayor información.

Ver también `unregister_compiler_function()` y `Plugin Compiler Functions`.

## Name

register\_function() — Registra dinamicamente un plugin de función para un template

## Descripción

```
void register_function(string name,  
                        mixed impl,  
                        bool cacheable,  
                        mixed cache_attrs);
```

Pase en el template el nombre de la función, seguido por el nombre de la función PHP que implementa esto.

La llamada a la función-*php impl* puede ser:

- a. a una cadena conteniendo el nombre de la función o
- b. un arreglo con la forma `array(&$object, $method)` con `&$object` siendo una referencia para un objeto y `$method` siendo una cadena conteniendo el nombre del método
- c. un arreglo con la forma `array(&$class, $method)` con `$class` siendo el nombre de una clase y `$method` siendo un método de esa clase.

*cacheable* y *cache\_attrs* pueden ser omitidos en la mayoría de los casos. Vea Controlando modos de Salida Cache de los Plugins para obtener mayores informes.

### Example 13.23. register\_function()

```
<?php  
$smarty->register_function('date_now', 'print_current_date');  
  
function print_current_date($params, &$smarty)  
{  
    if(empty($params['format'])) {  
        $format = "%b %e, %Y";  
    } else {  
        $format = $params['format'];  
        return strftime($format, time());  
    }  
}  
?>
```

y en el template

```
{date_now}  
  
{* or to format differently *}  
{date_now format="%Y/%m/%d"}
```

Ver también `unregister_function()` y `Plugin functions`.

## Name

register\_modifier() — modifica dinámicamente plugins registrados

## Descripción

```
void register_modifier(string name,  
                        mixed impl);
```

Pase en el template el nombre del modificador, seguido de la función PHP que implemente esto.

La llamada de la función-*php* *impl* puede ser

- una cadena conteniendo el nombre de la función
- un arreglo con la forma `array(&$object, $method)` con `&$object` siendo una referencia para un objeto y `$method` siendo una cadena conteniendo el nombre de un metodo
- un arreglo con la forma `array(&$class, $method)` con `$class` siendo el nombre de una clase y `$method` siendo un metodo de esta clase.

### Example 13.24. register\_modifier()

```
<?php  
  
// let's map PHP's stripslashes function to a Smarty modifier.  
$smarty->register_modifier('sslash', 'stripslashes');  
  
?>
```

template

```
<?php  
{* use 'sslash' to strip slashes from variables *}  
{ $var|sslash }  
?>
```

Ver También `unregister_modifier()`, `register_function()`, `modifiers`, `Extending Smarty with plugins y Creating Plugin modifiers`,



## Name

register\_object() — Registr un objeto para usar en el template

## Descripción

```
void register_object(string object_name,  
                    object object,  
                    array allowed_methods_properties,  
                    boolean format,  
                    array block_methods);
```

Vea object section para ejemplos.

Vea también get\_registered\_object(), y unregister\_object().

## Name

register\_outputfilter() — Registra dinamicamente filtros de salida

## Descripción

```
void register_outputfilter(mixed function);
```

Use este para registrar dinámicamente filtros de salida para operaciones en la salida del template antes de mostrarlo. Vea [Filtros de Salida de Templates](#) para mayores informes de como configurar una función de filtro de salida.

La llamada de la funcion-php *function* puede ser

- a. una cadena conteniendo un nombre de función
- b. un arreglo con la forma `array(&$object, $method)` con `&$object` siendo referencia a un objeto y `$method` siendo una cadena conteniendo el nombre de un metodo
- c. un arreglo con la forma `array(&$class, $method)` con `$class` siendo el nombre de la clase y `$method` siendo un método de esta clase.

Vea también `unregister_outputfilter()`, `register_prefilter()`, `register_postfilter()`, `load_filter()`, `$autoload_filters` y `template output filters`.

## Name

register\_postfilter() — Resgistr dinamicamente postfiltros

## Descripción

```
void register_postfilter(mixed function);
```

Use esto para registrar dinámicamente postfiltros para correr templates directos después de ser compilados. Vea postfiltros de template para mayores informes de como configurar funciones de postfiltering.

La llamada de la funcion-php *function* puede ser:

- a. una cadena conteniendo un nombre de función
- b. un arreglo con la forma `array(&$object, $method)` con `&$object` siendo una referencia para un objeto y `$method` siendo una cadena conteniendo el nombre de un método
- c. un arreglo con la forma `array(&$class, $method)` con `$class` siendo un nombre de clase y `$method` siendo un método de esta clase.

Vea También `unregister_postfilter()`, `register_prefilter()`, `register_outputfilter()`, `load_filter()`, `$autoload_filters` y `template output filters`.

## Name

register\_prefilter() — Registra dinamicamente prefiltros

## Descripción

```
void register_prefilter(mixed function);
```

Use esto para registrar prefiltros dinámicamente para correr templates antes de que estos sean compilados. Vea `template prefilters` para mayores informes de como configurar una función de `prefiltering`.

La llamada de la funcion-php *function* puede ser:

- a. una cadena conteniendo un nombre de función
- b. un arreglo con la forma `array(&$object, $method)` con `&$object` siendo una referencia para un objeto y `$method` siendo una cadena conteniendo el nombre de un método
- c. un arreglo con la forma `array(&$class, $method)` con `$class` siendo un nombre de clase y `$method` siendo un método de esta clase.

Ver también `unregister_prefilter()`, `register_postfilter()`, `register_ouputfilter()`, `load_filter()`, `$autoload_filters` y `template output filters`.

## Name

register\_resource — Registra dinamicamente un plugin de recurso

## Descripción

```
void register_resource(string name,  
                      array resource_funcs);
```

Use esto para registrar dinamicamente un recurso de plugin con Smarty. Pase el nombre o el recurso y o el arreglo de funciones que implementa esto. Vea [template resources](#) para mayor información de como configurar una función para mandar llamar templates.

### Nota técnica

El nombre del recurso debe tener al menos dos caracteres de largo. Un nombre de recurso de un carácter será ignorado y usado como parte del path del archivo como, `$smarty->display('c:/path/to/index.tpl');`

La php-funcion-array *resource\_funcs* debe tener 4 o 5 elementos. Con 4 elementos los elementos son las llamadas para las respectivas funciones de recurso "source", "timestamp", "secure" y "trusted". Con 5 elementos el primer elemento tiene que ser un objeto por referencia o un nombre de clase del objeto o una clase implementando el recurso y los 4 elementos siguientes tiene que ser los nombres de los métodos implementando "source", "timestamp", "secure" y "trusted".

### Example 13.25. register\_resource

```
<?php  
$smarty->register_resource('db', array('db_get_template',  
                                     'db_get_timestamp',  
                                     'db_get_secure',  
                                     'db_get_trusted'));  
?>
```

Ver también [unregister\\_resource\(\)](#) y [template resources](#)

## Name

trigger\_error — Despliega un mensaje de error

## Descripción

```
void trigger_error(string error_msg,  
                  int level);
```

Esta función puede ser usada para la salida de un mensaje de error usando Smarty. El parámetro *level* es uno de los valores usados para la función de php `trigger_error()` [[http://php.net/trigger\\_error](http://php.net/trigger_error)], ex.: `E_USER_NOTICE`, `E_USER_WARNING`, etc. Por default es `E_USER_WARNING`.

Ver también `$error_reporting`, `debugging` y `Troubleshooting`.

## Name

template\_exists() — Verifica si el template especificado existe

## Descripción

```
bool template_exists(string template);
```

Este puede aceptar un path para el template en el filesystem o un recurso de cadena especificando el template.

### Example 13.26. template\_exists()

Este ejemplo utiliza \$\_GET['page'] este incluye el contenido del template. Si el template no existe entonces un error de pagina es desplegado en su lugar.

El page\_container.tpl

```
<html>
<head><title>{$title}</title></head>
<body>
{include file='page_top.tpl'}

{* include middle content page *}
{include file=$page_mid}

{include file='page_footer.tpl'}
</body>
```

y el script PHP

```
<?php

// set the filename eg index.inc.tpl
$mid_template = $_GET['page'].'.inc.tpl';

if( !$smarty->template_exists($mid_template) ){
    $mid_template = 'page_not_found.inc.tpl';
}
$smarty->assign('page_mid', $mid_template);

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

?>
```

Ver también display(), fetch(), {include} y {insert}

## Name

unregister\_block — Des-registra dinamicamente un plugin de bloque de funciones

## Descripción

```
void unregister_block(string name);
```

Use esto para des-registrar dinámicamente un bloque de funciones de plugin. Pase en el bloque el *nombre* de la función.

Ver también register\_block() y Block Functions Plugins.



## Name

`unregister_compiler_function()` — des-registrar dinámicamente una función de compilación

## Descripción

```
void unregister_compiler_function(string name);
```

Pase el *nombre* de la función compiladora.

Ver también `register_compiler_function()` y Plugin Compiler Functions.

## Name

unregister\_function() — des-registrar dinámicamente una función de plugin del template

## Descripción

```
void unregister_function(string name);
```

Pase en el template el nombre de la función.

### Example 13.27. unregister\_function()

```
<?php
// nosotros no queremos que el diseñador del template tenga acceso a
// nuestros archivos de sistema

$smarty->unregister_function("fetch");
?>
```

Ver también register\_function() .

## Name

unregister\_modifier() — des-registrar dinámicamente un modificador de plugin

## Descripción

```
void unregister_modifier(string name);
```

Pase en el template el nombre del modificador.

### Example 13.28. unregister\_modifier()

```
<?php
// nosotros no queremos que el dise&ntilde;ador de template use strip tags
// para los elementos

$smarty->unregister_modifier("strip_tags");
?>
```

Ver también register\_modifier() y Plugin modifiers,

## Name

unregister\_object() — Des-registra dinamicamente un objeto

## Descripción

```
void unregister_object(string object_name);
```

Ver también register\_object() y objects section

## Name

`unregister_outputfilter()` — des-registra dinámicamente un filtro de salida

## Descripción

```
void unregister_outputfilter(string function_name);
```

Use este para des-registrar dinámicamente un filtro de salida.

Ver también `register_outputfilter()` y `template output filters`.

## Name

`unregister_postfilter()` — Des-registra dinamicamente un postfiltro

## Descripción

```
void unregister_postfilter(string function_name);
```

Ver también `register_postfilter()` y `template post filters`.

## Name

unregister\_prefilter() — Des-registra dinamicamente un prefiltro

## Descripción

```
void unregister_prefilter(string function_name);
```

Ver también register\_prefilter() y pre filters.

## Name

unregister\_resource() — Des-registra dinamicamente un plugin de recurso

## Descripción

```
void unregister_resource(string name);
```

Pase en el parámetro el nombre del recurso.

### Example 13.29. unregister\_resource()

```
<?php
$smarty->unregister_resource("db");
?>
```

Ver también register\_resource() y template resources



---

# Chapter 14. Cache

Caching es usado para aumentar la velocidad de llamada de `display()` o `fetch()` salvando esto en un archivo de salida. Si hay una versión de cache disponible para la llamada, este es mostrado en vez de regresar la salida de datos. Caching puede hacer cosas tremendamente rápidas, especialmente templates con largo tiempo de computo. Desde la salida de datos de `display()` o `fetch()` está en cache, un archivo de cache podría ser compuesto por diversos archivos de templates, archivos de configuración, etc.

Dado que sus templates son dinámicos, es importante tener cuidado de como usa la cache y por cuanto tiempo. Por ejemplo, si usted esta mostrando la pagina principal de su web site y esta no tiene cambios muy frecuentes en su contenido, esta puede funcionar bien en la cache por una hora o mas. por otro lado, si usted esta mostrando una pagina con un mapa de tiempo que contenga nueva información por minuto, no tiene sentido hacer cache nuestra página.

## Configurando el Cache

Lo primero que se tiene que hacer es habilitar el cache. esto es configurar `$caching = true` (o 1.)

### Example 14.1. Habilitando Cache

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

$smarty->caching = true;

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

Con el caching habilitado, la llamada a la función `display('index.tpl')` traera el template como siempre, pero también salvara una copia en el archivo de salida (una copia de cache) en el `$cache_dir`. En la proxima llamada de `display('index.tpl')`, la copia en cache sera usada en vez de traer nuevamente el template.

### Nota Técnica

Los archivos en el `$cache_dir` son nombrados similarmente al nombre del archivo de template. Aunque ellos tengan una extensión ".php", ellos no son realmente scripts ejecutables de php. No edite estos archivos!

Cada pagina en cache tiene un periodo de tiempo limitado determinado por `$cache_lifetime`. El default del valor es 3600 segundos, o 1 hora. Después de este tiempo expira, el cache es regenerado. Es posible dar tiempos individuales para caches con su propio tiempo de expiración para configuración `$caching = 2`. Vea la documentación en `$cache_lifetime` para mas detalles.

**Example 14.2. Configurando cache\_lifetime por cache**

```
<?php
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 not work when $caching = 2.
// The cache lifetime for home.tpl has already been set
// to 1 hour, and will no longer respect the value of $cache_lifetime.
// The home.tpl cache will still expire after 1 hour.
$smarty->cache_lifetime = 30; // 30 seconds
$smarty->display('home.tpl');
?>
```

Si `$compile_check` está habilitado, cada archivo de template y archivo de configuración que está involucrado con el archivo en cache es checado por modificadores. Si alguno de estos archivos fue modificado desde que el ultimo cache fue generado, el cache es regenerado inmediatamente. Esto es una forma de optimizar ligeramente el rendimiento de las cabeceras, dejar `$compile_check` determinado false.

**Example 14.3. Habilitando \$compile\_check**

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

$smarty->caching = true;
$smarty->compile_check = true;

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

Si `$force_compile` está habilitado, los archivos de cache siempre seran regenerados. Esto definitivamente desactiva el caching. `$force_compile` generalmente es usado para propósitos de debug solamente, una forma mas eficiente de desactivar el caching es asignando `$caching = false` (ó 0.)

La función `is_cached()` puede ser usada para testar si un template tiene un cache valido o no. Si usted tiene un template con cache que requiera alguna cosa como un retorno de base de datos, usted puede usar esto para saltar este proceso.

#### **Example 14.4. Usando `is_cached()`**

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

$smarty->caching = true;

if(!$smarty->is_cached('index.tpl')) {
    // No cache available, do variable assignments here.
    $contents = get_database_contents();
    $smarty->assign($contents);
}

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

Usted puede guardar partes de su pagina dinámica con la función de template `{insert}`. Vamos a decir que su pagina entera puede tener cache excepto para un banner que es mostrado abajo del lado derecho de su pagina. Usando la función `insert` para el banner, usted puede guardar ese elemento dinámico dentro de un contenido de cache. Vea la documentación en `{insert}` para detalles y ejemplos.

Usted puede limpiar todos los archivos de cache con la función `clear_all_cache()`, los archivos de cache individuales (o grupos) con la función `clear_cache()`.

#### **Example 14.5. Limpiando el cache**

```
<?php
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');
?>
```

## Multiples caches por pagina

Usted puede tener multiples archivos de cache para una simples llamada de `display()` o `fetch()`. Vamos a decir que una llamada a `display('index.tpl')` debe tener varios contenidos de salida diferentes dependiendo de alguna condición, y usted quiere separar los caches para cada una. Usted puede hacer esto pasando un `cache_id` como un segundo parámetro en la llamada de la función.

### Example 14.6. Pasando un `cache_id` para `display()`

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

$smarty->caching = true;

$my_cache_id = $_GET['article_id'];

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

Arriba, nosotros pasamos la variable `$my_cache_id` a `display()` con el `cache_id`. Para cada valor unico de `$my_cache_id`, un cache por separado sera generado para cada `index.tpl`. En este ejemplo, "article\_id" fue pasado en URL y es usado como el `cache_id`.

### Nota Técnica

Tenga mucho cuidado cuando pase valores del cliente (web browser) dentro de Smarty (o alguna aplicación PHP). Aunque el ejemplo de arriba usar el `article_id` desde una URL parece facil, esto podría tener fatales consecuencias. El `cache_id` es usado para crear un directorio en el sistema de archivos, entonces si el usuario decide pasar un valor extremadamente largo para `article_id`, o escribir un script que envia `article_ids` aleatorios en un paso rápido, esto posiblemente podría causar problemas a nivel del servidor. Tenga la certeza de limpiar algún dato pasado antes de usarlo. En este ejemplo, tal vez usted sabia que el `article_id` tiene un largo de 10 caracteres este es constituido solamente de alfanuméricos, y debe ser un `article_id` valido en la base de datos. Verifique esto!

Asegurarse de pasar el mismo `cache_id` como el segundo parámetro para `is_cached()` y `clear_cache()`.

**Example 14.7. Pasando un cache\_id para is\_cached()**

```
<?php
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 here.
    $contents = get_database_contents();
    $smarty->assign($contents);
}

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

Usted puede limpiar todos los caches para un cache\_id en particular pasando el primer parámetro null a clear\_cache().

**Example 14.8. Limpando todos los caches para un cache\_id en particular**

```
<?php
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");
?>
```

De esta manera, usted puede "agrupar" sus caches conjuntamente dando les el mismo cache\_id.

## Cache Groups

Usted puede hacer agrupamientos mas elaborados configurando grupos de cache\_id. Esto se logra con la separación de cada sub-grupo con una barra vertical "|" en el valor del cache\_id. Usted puede tener tantos sub-grupos como guste.

Usted puede pensar que los grupos de cache son parecidos a un directorio para organizar. por ejemplo, un grupo de cache con "a|b|b" podria pensarse como la estructura del directorio "a/b/c/". clear\_cache(null,"a|b|c") esto seria para quitar los archivos "/a/b/c/\*". clear\_cache(null,"a|b") esto seria para quitar los archivos

"/a/b/\*". Si usted especifica el `compile_id` como `clear_cache(null,"a|b","foo")` este tratara de agregarlo al grupo de cache `"/a/b/c/foo/"`. Si usted especifica el nombre del template tal como `clear_cache("foo.tpl","a|b|c")` entonces el smarty intentara borrar `"/a/b/c/foo.tpl"`. Usted no puede borrar un nombre de template especifico bajo multiples grupos de cache como `"/a/b/*/foo.tpl"`, el grupo de cache trabaja solo de izquierda a derecha. Usted puede necesitar para su grupos de templates un unico grupo de cache jerarquico para poder limpiarlos como grupos.

El agrupamiento de cache no debe ser confundido con su directorio jerarquico del template, El agrupamiento de cache no tiene ninguna ciencia de como sus templates son estructurados. Por ejemplo, si usted tiene una estructura `display('themes/blue/index.tpl')`, usted no puede limpiar el cache para todo bajo el directorio `"themes/blue"`. Si usted quiere hacer esto, usted debe agruparlos en el `cache_id`, como `display('themes/blue/index.tpl','themes|blue')`; Entonces usted puede limpiar los caches para el tema azul con `clear_cache(null,'themes|blue')`;

### Example 14.9. Grupos de `cache_id`

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

$smarty->caching = true;

// clear all caches with "sports|basketball" as the first two cache_id groups
$smarty->clear_cache(null,"sports|basketball");

// clear all caches with "sports" as the first cache_id group. This would
// include "sports|basketball", or "sports|(anything)|(anything)|(anything)|..."
$smarty->clear_cache(null,"sports");

// clear the foo.tpl cache file with "sports|basketball" as the cache_id
$smarty->clear_cache("foo.tpl","sports|basketball");

$smarty->display('index.tpl',"sports|basketball");
?>
```

## Controlando salida de Cacheabilidad de plugins

Desde Smarty-2.6.0 los caches de plugins pueden ser declarados o registrados. El tercer parámetro para `register_block()`, `register_compiler_function()` y `register_function()` es llamado `$cacheable` y el default es true que es también el comportamiento de plugins en la versiones anteriores a Smarty 2.6.0.

Cuando registre un plugin con `$cacheable=false` el plugin es llamado todo el tiempo en la pagina que está siendo mostrada, aun si la pagina viene desde el cache. La función de plugin tiene un comportamiento parecido al de la función `insert`.

En contraste con `{insert}` el atributo para el plugin no está en cache por default. Ellos pueden ser declarados para ser cacheados con el cuarto parámetro `$cache_attrs`. `$cache_attrs` es un arreglo de nombres

de atributos que deben ser cacheados, entonces la función de plugin pega el valor como si fuera el tiempo en que la página fue escrita para el cache todo el tiempo este es traído desde el cache.

### Example 14.10. Previniendo que una salida de plugin de ser cacheada

```
<?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_seconds', false, array('endtime' => true));

if (!$smarty->is_cached('index.tpl')) {
    // fetch $obj from db and assign...
    $smarty->assign_by_ref('obj', $obj);
}

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

Donde index.tpl es:

```
Time Remaining: {remaining endtime=$obj->endtime}
```

El número en segundos hasta el endtime del \$obj este sufre cambios en cada display de la página, aun si la página esta en cache. Desde que el atributo endtime sea cacheado el objeto solamente tiene que ser jalado de la base de datos cuando la página esta escrita en la cache mas no en requisiciones de la página.

**Example 14.11. Previendo una pasada entera del template para el cache**

index.php:

```
<?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', false);

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

Donde index.tpl es:

```
Page created: {"0"|date_format:"%D %H:%M:%S"}
```

```
{dynamic}
```

```
Now is: {"0"|date_format:"%D %H:%M:%S"}
```

```
... do other stuff ...
```

```
{/dynamic}
```

Cuando recarga la pagina usted notara que ambas fechas son diferentes. Una es "dinamica" y la otra es "estática". Usted puede hacer todo entre las etiquetas {dynamic}...{/dynamic} y tener la certeza de que no sera cacheado como el resto de la pagina.



---

# Chapter 15. Características Avanzadas

## Objetos

El Smarty permite acceso a objetos de PHP a través de sus templates. Hay dos formas de accederlos. Una forma es registrando objetos para el template, entonces accéselos mediante sintaxis similar a las funciones habituales. La otra es asignar objetos al template y accederlos como si fueran una variable asignada. El primer método tiene una sintaxis de template mucho mas agradable. Y también mas segura, a medida que un objeto registrado puede ser reescrito a ciertos métodos y propiedades. Sin embargo tanto, **un objeto registrado no puede ser puesto en loop o ser asignado en arreglos de objetos**, etc. El método que usted escoja sera determinado por sus necesidades, pero utilice el primero método si es posible para mantener un minimo de sintaxis en el template.

Si \$security esta habilitada, ninguno de los dos métodos privados o funciones pueden ser accésados (comenzando con "\_"). Si un metodo y propiedades de un mismo nombre existe, el método será usado.

Usted puede restringir los métodos y propiedades que pueden ser accésados listandolos en un arreglo como el tercer parámetro de registro.

Por default, los parámetros pasados a los objetos a a través de los templates son pasados de la misma forma en que las funciones de costumbre los obtienen. Un arreglo asociativo es pasado como el primer parámetro, y el objeto smarty como el segundo. Si usted quiere que los parámetros pasados uno de cada vez por cada argumento pasen como parámetros de un objeto tradicional, defina el cuarto parámetro de registro en falso.

El quinto parámetro opcional solo tiene efecto con *format* siendo *true* y conteniendo una lista de métodos de ob que seran tratados como bloques. Esto significa que estos métodos tienen una etiqueta de cierre en el template (`{foobar->meth2} . . . {/foobar->meth2}`) y los parámetros para los métodos tienen la misma sinopsis como los parámetros de block-function-plugins: Ellos reciben 4 parámetros *\$params*, *\$content*, *&\$smarty* y *&\$repeat* también se comportan como block-function-plugins.

**Example 15.1. usando un objeto registrado o atribuido**

```
<?php
// el objeto

class My_Object {
    function meth1($params, &$smarty_obj) {
        return "this is my meth1";
    }
}

$myobj = new My_Object;
// registrando el objeto (será por referencia)
$smarty->register_object("foobar",$myobj);
// Si usted quiere restringir acceso a ciertos metodos o propiedades, listelos
$smarty->register_object("foobar",$myobj,array('meth1','meth2','prop1'));
// Si usted quiere usar el formato de parámetro del objeto tradicional, pase un bo
$smarty->register_object("foobar",$myobj,null,false);

// también puede asignar ojetos. Posible cuando se asignan por referencia.
$smarty->assign_by_ref("myobj", $myobj);

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

y como debiera acceder a su objeto en index.tpl

```
{* accedando a nuestro objeto registrado *}
{foobar->meth1 p1="foo" p2=$bar}

{* usted también puede asignar la salida *}
{foobar->meth1 p1="foo" p2=$bar assign="output"}
the output was {$output}

{* accedando a nuestro objeto asignado *}
{$myobj->meth1("foo",$bar)}
```

## Prefilters

Los prefilters de Template son funciones de PHP que corren sus templates antes de ser compilados. Esto es bueno para procesar por adelantado sus templates y remover comentarios no deseados, vigilando a las personas que coloquen en sus templates, etc.

Los Prefilters pueden ser registrado o cargado del directorio de plugins usando la función `load_filter()` o por la configuración de la variable `$autoload_filters`.

El Smarty pasara el código fuente del template como el primer argumento, y espera que la función le retorne el código fuente del template resultante.

### Example 15.2. usando un prefiltro prefilter de template

```
<?php
// ponga esto en su aplicación
function remove_dw_comments($tpl_source, &$smarty)
{
    return preg_replace("/<!--#. *-->/U", "", $tpl_source);
}

// registrar el prefilter
$smarty->register_prefilter("remove_dw_comments");
$smarty->display("index.tpl");
?>
```

Esto eliminara todos los comentarios en el codigo del template.

## Postfilters

Los postfilters de template son funciones de PHP con las cuales sus templates son corridos inmediatamente después de ser compilados. Los postfilters pueden ser registrado o cargados del directorio de plugins usando la función `load_filter()` o por la variable de configuración `$autoload_filters`. El Smarty pasara el código fuente del template compilado como el primer argumento, y espera que la función retorne el resultado del procesamiento.

### Example 15.3. Usando un postfilter de template

```
<?php
// ponga esto en su aplicaci&oacute;n
function add_header_comment($tpl_source, &$smarty)
{
    return "<?php echo \"<!-- Created by Smarty! -->;\n\" ?>;\n\".$tpl_source;
}

// registra el postfilter
$smarty->register_postfilter("add_header_comment");
$smarty->display("index.tpl");
?>
```

Observe como hacer la compilacion para Smarty del template index.tpl:

```
<!-- Created by Smarty! -->
{* rest of template content... *}
```

## Filtros de salida

Cuando el template es invocado a través de `display()` o `fetch()`, su salida puede ser enviada a través de uno o mas filtros de salida. Este es diferente a los postfilters porque los postfilters operan en los templates compilados antes de ser salvados en disco, y los filtros de salida operan en la salida del template cuando este es ejecutado.

Los Filtros de Salida pueden ser registrado o cargados del directorio de plugins usando la función `load_filter()` o configurando a variable `$autoload_filters`. El Smarty pasara la salida como el primer argumento, y espera que la función retorne el resultado del procesamiento.

### Example 15.4. Usando un filtro de salida de template

```
<?php
// ponga esto en su aplicación
function protect_email($tpl_output, &$smarty)
{
    $tpl_output =
        preg_replace('!(\S+)@([a-zA-Z0-9\.\-]+\.\.([a-zA-Z]{2,3}|[0-9]{1,3}))!','
            '$1%40$2', $tpl_output);
    return $tpl_output;
}

// registra el outputfilter
$smarty->register_outputfilter("protect_email");
$smarty->display("index.tpl");

// Ahora cualquier ocurrencia de una direcci&oacute;n de email en la salida
// del template tendra una simple protecci&oacute;n contra spambots
?>
```

## Función manipuladora de cache

Como una alternativa al uso del mecanismo de caching por default basado en archivo, usted puede especificar una función habitual de manipulación de cache que será usada para leer, escribir y limpiar archivos de cache.

Cree una función en su aplicación para que Smarty la use como un manipulador de cache. Defina el nombre de la variable de clase en el `$cache_handler_func`. El Smarty ahora usara esta para manipular datos en el cache. El primer parámetro es la acción, que puede ser uno de estos 'read', 'write' y 'clear'. El segundo parámetro es el objeto de Smarty. El tercer parámetro es el contenido que esta en el cache. Sobre 'write', el Smarty pasa el contenido en cache en estos parámetros. sobre 'read', el Smarty espera que su función acepte este parámetro por referencia y poblar estos con los datos en cache. Sobre 'clear', el Smarty pasa una variable en cero desde aquí que esta no es usada. El cuarto parámetro es el nombre del archivo de template(necesario para leer/escribir). El quinto parámetro es la `cache_id` (opcional). El sexto parámetro es la `compile_id` (opcional).

NOTA: El ultimo parámetro (`$exp_time`) fue adicionado en el Smarty-2.6.0.

```

switch($action) {
case 'read':
    // read cache from database
    $results = mysql_query("select CacheContents from CACHE_PAGES where CacheID='$CacheID'");
    if(!$results) {
        $smarty_obj->trigger_error_msg("cache_handler: query failed.");
    }
    $row = mysql_fetch_array($results,MYSQL_ASSOC);

    if($use_gzip && function_exists("gzuncompress")) {
        $cache_content = gzuncompress($row["CacheContents"]);
    } else {
        $cache_content = $row["CacheContents"];
    }
    $return = $results;
    break;
case 'write':
    // save cache to database

    if($use_gzip && function_exists("gzcompress")) {
        // compress the contents for storage efficiency
        $contents = gzcompress($cache_content);
    } else {
        $contents = $cache_content;
    }
    $results = mysql_query("replace into CACHE_PAGES values(
        '$CacheID',
        '".addslashes($contents)."'
    );");
    if(!$results) {
        $smarty_obj->trigger_error_msg("cache_handler: query failed.");
    }
    $return = $results;
    break;
case 'clear':
    // clear cache info
    if(empty($cache_id) && empty($compile_id) && empty($tpl_file)) {
        // clear them all
        $results = mysql_query("delete from CACHE_PAGES");
    } else {
        $results = mysql_query("delete from CACHE_PAGES where CacheID='$CacheID'");
    }
    if(!$results) {
        $smarty_obj->trigger_error_msg("cache_handler: query failed.");
    }
    $return = $results;
    break;
default:
    // error, unknown action
    $smarty_obj->trigger_error_msg("cache_handler: unknown action \"\$action\");");
    $return = false;
    break;
}
mysql_close($link);
return $return;
}

?>

```

**Example 15.5. ejemplo usando MySQL como una fuente de cache**

## Recursos

Los templates pueden venir de una variedad de fuentes. Cuando usted muestra un template con `display()` o `fetch()`, o incluye un template dentro de otro template, usted suministra un tipo de recurso, seguido por la ruta correcta y el nombre del template. Si un recurso no es dado explícitamente el valor de `$default_resource_type` es asumido.

## Templates desde `$template_dir`

Los templates desde el `$template_dir` no requieren recursos del template, aunque usted puede usar el `file:resource` for consistancy(recurso por consistencia). Justamente proporcionando la ruta(path) del template que usted quiere usar en relación al directorio root `$template_dir`.

### Example 15.6. Usando templates desde `$template_dir`

```
<?php
// desde el script de PHP
$smarty->display("index.tpl");
$smarty->display("admin/menu.tpl");
$smarty->display("file:admin/menu.tpl"); // igual al de arriba
?>

{* dentro del template de Smarty *}
{include file="index.tpl"}
{include file="file:index.tpl"} {* igual al de arriba *}
```

## Templates partiendo de cualquier directorio

Los templates de fuera del `$template_dir` requieren el `file:` tipo de recurso del template, seguido por la ruta absoluta y el nombre del template.

### Example 15.7. usando templates desde cualquier directorio

```
<?php
// desde el script de PHP
$smarty->display("file:/export/templates/index.tpl");
$smarty->display("file:/path/to/my/templates/menu.tpl");
?>
```

dentro del template Smarty:

```
{include file="file:/usr/local/share/templates/navigation.tpl"}
```

## Rutas de archivos de Windows

Si usted esta utilizando una maquina con windows, las rutas de los archivos normalmente incluyen la letra del drive (C:) en el comienzo del nombre de la ruta. Asegurarse de usar "file:" en la ruta para evitar conflictos de nombres y poder obtener los resultados desados.

### Example 15.8. usando templates con rutas de archivos de windows

```
<?php
// dentro del script de PHP
$smarty->display("file:C:/export/templates/index.tpl");
$smarty->display("file:F:/path/to/my/templates/menu.tpl");
?>
```

dentro del template de Smarty

```
{include file="file:D:/usr/local/share/templates/navigation.tpl" }
```

## Templates partiendo de otras fuentes

Se pueden retomar templates usando cualquier fuente posible a la que usted pueda acceder con PHP: base de datos, sockets, LDAP, etc. Usted puede hacer esto escribiendo las funciones de plugin de recurso y registrandolas con Smarty.

Vea la sección resource plugins para mayor informacion sobre las funciones que puede utilizar.

### Note

Nota Usted puede activar manualmente el recurso `file` incrustado, pero no puede suministrar un recurso que busca templates a partir del sistema de archivos de alguna otra forma registrando bajo otro nombre de recurso.

**Example 15.9. Usando recursos habituales**

```

?>
// ponga estas funciones en algun lugar de su aplicación
function db_get_template ($tpl_name, &$tpl_source, &$smarty_obj)
{
    // llame su base de datos para traer los datos al template,
    // poblando el $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, &$smarty_obj)
{
    // llame su base de datos para traer los datos y poblar el $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)
{
    // asume que todos los templates son seguros
    return true;
}

function db_get_trusted($tpl_name, &$smarty_obj)
{
    // no usar para templates
}

// registrar el nombre del recurso "db"
$smartyy->register_resource("db", array("db_get_template",
                                       "db_get_timestamp",
                                       "db_get_secure",
                                       "db_get_trusted"));

// usando el recurso a partir de un script PHP
// usando el recurso dentro del template de Smarty
$smartyy->display("db:index.tpl");
?>

```



## Función manipuladora de Template por default

Usted puede especificar la función que será usada para devolver el contenido del template dentro del evento del template no puede ser retomado desde su recurso. Un uso distinto es para crear templates que no existen "on-the-fly" (templates cuyo contenido cambia mucho, bastante variable).

### Example 15.10. usando la función manipuladora de template por default

```
<?php
// ponga esta funci&oacute;n en algun lugar de su aplicaci&oacute;n

function make_template ( $resource_type, $resource_name, &$template_source, &$templ
{
  if( $resource_type == 'file' ) {
    if ( ! is_readable ( $resource_name )) {
      // create the template file, return contents.
      $template_source = "This is a new template.";
      $template_timestamp = time();
      $smarty_obj->_write_file($resource_name,$template_source);
      return true;
    }
  } else {
    // not a file
    return false;
  }
}

// defina la funci&oacute;n manipuladora por default
$smarty->default_template_handler_func = 'make_template';
?>
```

---

# Chapter 16. Extendiendo Smarty con plugins

La version 2.0 introduce la arquitectura de plugin que es usada para casi todas las funcionalidades adaptables del Smarty. Esto incluye:

- funciones
- modificadores
- funciones de bloque
- funciones de compilación
- prefiltros
- postfiltros
- filtros de salida
- recursos(fuentes)
- inserts

Con la excepción de recursos, la compatibilidad con la forma antigua de funciones de manipulación de registro via `register_*` API es conservada. Si usted no uso el API en lugar de eso modifiko las variables de clase `$custom_funcs`, `$custom_mods`, y otras directamente, entonces usted va a necesitar ajustar sus scripts para cualquiera que use el API o convertir sus funciones habituales en plugins.

## Como funcionan los Plugins

Los plugins son siempre cargados cuando son requeridos. solo los calificativos especificos, funciones, recursos, etc convocados en scripts del template seran leidos. Además, cada plugin es cargado una sola vez, aun si usted tiene corriendo varias instancias diferentes de Smarty dentro de la misma petición.

Pre/posfiltros y salidas de filtros son una parte de un caso especial. Dado que ellos no son mencionados en los templates, ellos deben ser registrados o leidos explicitamente mediante funciones de API antes de que el template sea procesado. El orden en el cual son ejecutados multiples filtros del mismo tipo depende del orden en el que estos son registrados o leidos.

El directorio de `directory` puede ser una cadena que contenga una ruta o un arreglo que contenga multiples rutas. Para instalar un plugin, simplemente coloquelo en el directorio y el Smarty lo usara automáticamente.

## Nombres convencionales

Los archivos y funciones de Plugin deben seguir una convención de apariencia muy especifica a fin de que pueda ser localizada por el Smarty.

Los archivos de plugin deben ser nombrados de la siguiente forma:

*type.name.php*

Donde `type` es uno de los siguientes tipo de plugin:

- function
- modifier

- block
- compiler
- prefilter
- postfilter
- outputfilter
- resource
- insert

Y name seria un identificador valido (solo, letras, números, y underscores).

Algunos ejemplos: `function.html_select_date.php`, `resource.db.php`, `modifier.spacify.php`.

Las funciones de plugin dentro de los archivos de plugin deben ser nombradas de la siguiente forma:

```
smarty_type, _name
```

El significado de type and name son los mismo que los anteriores.

El Smarty mostrara mensajes de error apropiados si el archivo de plugins que es necesario no es encontrado, o si el archivo a la función de plugin esta nombrado inadecuadamente.

## Escribiendo Plugins

Los Plugins pueden ser leidos por el Smarty automáticamente del sistema de archivos o pueden ser registrados en tiempo de ejecución por medio de una de las funciones de API `register_*`. Estos también pueden ser usados con la función API `unregister_*`.

Para los plugins que son registrados en tiempo de ejecución, el nombre de la(s) función(es) de plugin no tiene que seguir la convención de apariencia.

Si un plugin depende de alguna función alimentada por otro plugin (como es el caso con algunos plugins incrustados con el Smarty), entonces la forma apropiada para leer el plugin necesario es esta:

```
<?php
require_once $smarty->_get_plugin_filepath('function', 'html_options');
?>
```

Como regla general, el objeto Smarty siempre es pasado a los plugins como ultimo parámetro (con dos excepciones: los modificadores no pasan el objeto de Smarty del todo y los blocks obtenidos son pasados `&$repeat` después el objeto de Smarty para manter compatibilidad con antiguas versiones de Smarty).

## Funciones de Template

```
void smarty_function_name($params, &$smarty);

array $params;
```

```
object &$smarty;
```

Todos los atributos pasados para las funciones de template a partir del template estan contenidas en *\$params* como un arreglo asociativo.

La salida(valor de retorno) de la función será substituida en el lugar de la etiqueta de la función en el template (la función *fetch*, por ejemplo). Alternativamente, la función puede simplemente ejecutar alguna otra tarea sin tener alguna salida (la función *assign*).

Si la función necesita pasar valores a algunas variables del template o utilizar alguna otra funcionalidad del Smarty, esta puede usar el objeto *\$smarty* alimentandolo para hacer eso.

Vea tambien: *register\_function()*, *unregister\_function()*.

### Example 16.1. Función de plugin con salida

```
<?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];
}
?>
```

que puede ser usada en el template de la siguiente forma:

```
Question: Will we ever have time travel?
Answer: {eightball}.
```

## Example 16.2. Función de plugin sin salida

```

<?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)
{
    if (empty($params['var'])) {
        $smarty->trigger_error("assign: missing 'var' parameter");
        return;
    }

    if (!in_array('value', array_keys($params))) {
        $smarty->trigger_error("assign: missing 'value' parameter");
        return;
    }

    $smarty->assign($params['var'], $params['value']);
}
?>

```

## Modificadores

Los modificadores son funciones que son aplicadas a una variable en el template antes de ser mostrada o usada en algun otro contexto. Los modificadores pueden ser encadenados conjuntamente.

```
mixed smarty_modifier_name($value, $param1);
```

```
mixed $value;
[mixed $param1, ...];
```

El primer parámetro en el modificador de plugin es el valor sobre el cual el modificador es precisa para funcionar. El resto de los parámetros pueden ser opcionales, dependiendo de cual tipo de operación va a ser ejecutada.

El modificador debe retornar el resultado de su procesamiento.

Vea Tambien register\_modifier(), unregister\_modifier().

### Example 16.3. Plugin modificador simple

Este plugin básicamente es un alias de una función incorporada en PHP. Este no tiene ningun parámetro adicional.

```
<?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. Plugin modificador mas complejo

```

<?php
/*
 * Smarty plugin
 * -----
 * File:      modifier.truncate.php
 * Type:      modifier
 * Name:      truncate
 * Purpose:   Truncate a string to a certain length if necessary,
 *           optionally splitting in the middle of a word, and
 *           appending the $etc string.
 * -----
 */
function smarty_modifier_truncate($string, $length = 80, $etc = '...',
                                  $break_words = false)
{
    if ($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;
}
?>

```

## Block Functions

```

void smarty_block_name($params, $content, &$smarty, &$repeat);

array $params;
mixed $content;
object &$smarty;
boolean &$repeat;

```

Las funciones de bloque son funciones de forma: {func} .. {/func}. En otras palabras, estas encapsulan un bloque del template y operan el contenido de este bloque. Las funciones de bloque toman precedencia sobre las funciones habituales con el mismo nombre, es decir, usted no puede tener ambas, las funciones habituales {func} y las funciones de bloque {func} .. {/func}.

Por default la implementación de su función es llamada dos veces por el Smarty: una vez por la etiqueta de apertura, y la otra por la etiqueta de cierre (vea &\$repeat abajo para ver como hacer cambios a esto).

Solo la etiqueta de apertura de la función de bloque puede tener atributos. Todos los atributos pasados a las funciones de template estan contenidos en `$params` como un arreglo asociativo. Usted puede acceder a cualquiera de estos valores directamente, e.g. `$params['start']`. Los atributos de la etiqueta de apertura son también accesibles a su función cuando se procesa la etiqueta de cierre.

El valor de la variable `$content` depende de que si su función es llamada por la etiqueta de cierre o de apertura. En caso de que la etiqueta sea de apertura, este será `null`, si la etiqueta es de cierre el valor será del contenido del bloque del template. Se debe observar que el bloque del template ya a sido procesado por el Smarty, asi todo lo que usted recibirá es la salida del template, no el template original.

El parámetro `&$repeat` es pasado por referencia para la función de implementación y proporciona la posibilidad de controlar cuantas veces será mostrado el bloque. Por default `$repeat` es `true` en la primera llamada de la block-function (etiqueta de apertura del bloque) y `false` en todas las llamadas subsecuentes a la función de boque (etiqueta de cierre del boque). Cada vez que es implementada la función retorna con el `&$repeat` siendo `true`, el contenido entre `{func} .. {/func}` es evaluado y es implementado a la función es llamada nuevamente con el nuevo contenido del bloque en el parámetro `$content`.

Si usted tiene funciones de bloque anidadas, es posible descubrir cual es el padre de la función de bloque accedando la variable `$smarty->_tag_stack`. Solo hacer un `var_dump()` sobre ella y la estructura estara visible.

Vea tambien: `register_block()`, `unregister_block()`.

### Example 16.5. Función de bloque

```
<?php
/*
 * Smarty plugin
 * -----
 * File:      block.translate.php
 * Type:      block
 * Name:      translate
 * Purpose:   translate a block of text
 * -----
 */
function smarty_block_translate($params, $content, &$smarty, &$repeat)
{
    if (isset($content)) {
        $lang = $params['lang'];
        // do some intelligent translation thing here with $content
        return $translation;
    }
}
?>
```

## Funciones Compiladoras

Las funciones compiladoras solo son llamadas durante la compilación del template. Estas son útiles para inyectar código PHP o contenido estático time-sensitive dentro del template. Si existen ambas, una función compiladora y una función habitual registrada bajo el mismo nombre, la función compiladora tiene precedencia.



```
mixed smarty_compiler_name($tag_arg, &$smarty);

string $tag_arg;
object &$smarty;
```

En las funciones compiladoras son pasados dos parámetros: la etiqueta string del argumento de la etiqueta - básicamente, todo a partir del nombre de la función hasta el delimitador del cierre, y el objeto del Smarty. Es supuesto que retorna el código PHP para ser inyectado dentro del template compilado.

Vea también `register_compiler_function()`, `unregister_compiler_function()`.

### Example 16.6. Función compiladora simple

```
<?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:M
}
?>
```

Esta función puede ser llamada en un template de la siguiente forma:

```
{* esta función es ejecutada solamente en tiempo de compilación *}
{tplheader}
```

El código PHP resultante en el template compilado sería algo así:

```
<?php
echo 'index.tpl compiled at 2002-02-20 20:02';
?>
```

## Prefiltros/Postfiltros

Los Plugins Prefilter y postfilter son muy similares en concepto; donde ellos difieren es en la ejecución -- más precisamente en el tiempo sus ejecuciones.

```
string smarty_prefilter_name($source, &$smarty);

string $source;
object &$smarty;
```

Los Prefilters son usados para procesar el fuente del template inmediatamente antes de la compilación. El primer parámetro de la función del prefilter es el fuente del template, posiblemente modificado por algunos otros prefilters. El Plugin es supuesto que retorne el fuente modificado. Observe que este código no es salvado en ningun lugar, este es solo usado para la compilación.

```
string smarty_postfilter_name($compiled, &$smarty);

string $compiled;
object &$smarty;
```

Los Postfilters son usados para procesar la salida compilada del template (el código PHP) inmediatamente después de que la compilacion es terminada pero antes de que el template compilado sea salvado en el sistema de archivos. El primer parámetro para la función postfilter es el código del template compilado, posiblemente modificado por otros postfilters. El plugin se supone retornara la versión modificada de este código.

### 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("$1")', $source);
}
?>
```

### Example 16.8. postfilter plugin

```
<?php
/*
 * Smarty plugin
 * -----
 * File:      postfilter.post01.php
 * Type:      postfilter
 * Name:      post01
 * Purpose:   Output code that lists all current template vars.
 * -----
 */
function smarty_postfilter_post01($compiled, &$smarty)
{
    $compiled = "<pre>\n<?php print_r(\${this->get_template_vars()}); ?>\n</pre>" .
    return $compiled;
}
?>
```

## Filtros de Salida

Los Filtros de salida operan en la salida del template, después que el template es cargado y ejecutado, pero antes que la salida sea mostrada.

```
string smarty_outputfilter_name($template_output, &$smarty);

string $template_output;
object &$smarty;
```

El primer parámetro de la función de filtro de salida es la salida del template que necesita ser procesada, y el segundo parámetro es la instancia del Smarty invocando el plugin. El plugin debe hacer el procesamiento y retornar los resultados.

## Example 16.9. plugin de filtro de salida

```
<?php
/*
 * Smarty plugin
 * -----
 * File:      outputfilter.protect_email.php
 * Type:      outputfilter
 * Name:      protect_email
 * Purpose:   Converts @ sign in email addresses to %40 as
 *            a simple protection against spambots
 * -----
 */
function smarty_outputfilter_protect_email($output, &$smarty)
{
    return preg_replace('!(\S+)@([a-zA-Z0-9\.\-]+\.[a-zA-Z]{2,3}|[0-9]{1,3})!',
        '$1%40$2', $output);
}
?>
```

## Fuentes

Las fuentes de los plugins son como una forma generica de suministrar código fuente de template o componentes de script PHP al Smarty. Algunos ejemplos de fuentes: base de datos, LDAP, memoria compartida, sockets, etc.

Existe un total de 4 funciones que necesitan estar registradas para cada tipo de fuente. Cada función recibirá el fuente requerido como primer parámetro y el objeto de Smarty como ultimo parámetro. El resto de los parámetros dependen de la función.

```
bool smarty_resource_name_source($rsrc_name, &$source, &$smarty);

string $rsrc_name;
string &$source;
object &$smarty;

bool smarty_resource_name_timestamp($rsrc_name, &$timestamp, &$smarty);

string $rsrc_name;
int &$timestamp;
object &$smarty;

bool smarty_resource_name_secure($rsrc_name, &$smarty);

string $rsrc_name;
object &$smarty;

bool smarty_resource_name_trusted($rsrc_name, &$smarty);

string $rsrc_name;
```

```
object &$smarty;
```

La primera función debe devolver el recurso. Su segundo parámetro es una variable pasada por referencia donde el resultado debe ser almacenado. La función debe retornar `true` si esta pudo recuperar satisfactoriamente el recurso y en caso contrario retornara `false`.

La segunda función debe devolver la ultima modificación del recurso requerido (como un timestamp Unix). El segundo parámetro es una variable pasada por referencia donde el timestamp sera almacenado. La función debe retornar `true` si el timestamp pudo ser determinado satisfactoriamente, y en caso contrario retornara `false`.

La tercera función debe retornar `true` o `false`, dependiendo si el recurso requerido es seguro o no. Esta función es usada solo para recursos de template pero esta debe ser definida.

La cuarta función debe retornar `true` o `false`, dependiendo si el recurso requerido es seguro o no. Esta función es usada solo para componetes de script de PHP solicitado por las etiquetas **include\_php** o **insert** con el atributo `src`. Sin embargo, este debe ser definido para los recurso del template.

Vea también `register_resource()`, `unregister_resource()`.

**Example 16.10. Plugin resource (recurso)**

```

/*
 * Smarty plugin
 * -----
 * File:      resource.db.php
 * Type:      resource
 * Name:      db
 * Purpose:   Fetches templates from a database
 * -----
 */
function smarty_resource_db_source($tpl_name, &$tpl_source, &$smarty)
{
    // 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_timestamp, &$smarty)
{
    // 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
}
?>

```

## Inserts

Los Plugins Insert son usados para implementar funciones que son invocadas por las etiquetas **insert** en el template.

```
string smarty_insert_name($params, &$smarty);
```

```
array $params;  
object &$smarty;
```

El primer parámetro de la función es un arreglo asociativo de atributos pasados al insert.

La función insert debe retornar el resultado que ira a sustituir el lugar de la etiqueta **insert** en el template.

### Example 16.11. insert plugin

```
<?php  
/*  
 * Smarty plugin  
 * -----  
 * File:      insert.time.php  
 * Type:      time  
 * Name:      time  
 * Purpose:   Inserts current date/time according to format  
 * -----  
 */  
function smarty_insert_time($params, &$smarty)  
{  
    if (empty($params['format'])) {  
        $smarty->trigger_error("insert time: missing 'format' parameter");  
        return;  
    }  
  
    $datetime = strftime($params['format']);  
    return $datetime;  
}  
?>
```

---

# Part IV. Appendixes

---



---

## Table of Contents

17. Localización de Errores .....	222
Errores Smarty/PHP .....	222
18. Consejos y Trucos .....	224
Manipulación de Variables Vacías .....	224
Manipulación del valor default de una variable .....	224
Pasando la variable titulo a la cabecera del template .....	225
Fechas .....	226
WAP/WML .....	227
Templates con Componetes .....	229
Ofuscando direcciones de E-mail .....	230
19. Fuentes .....	231
20. ERRORES .....	232

---

# Chapter 17. Localización de Errores

## Errores Smarty/PHP

El Smarty puede obtener muchos errores tales como, atributos de etiquetas perdidos o nombres de variables mal formadas. Si este ocurre, Usted vera un error similar al siguiente:

### Example 17.1. Errores de Smarty

```
Warning: Smarty: [in index.tpl line 4]: syntax error: unknown tag - '%blah'  
      in /path/to/smarty/Smarty.class.php on line 1041
```

```
Fatal error: Smarty: [in index.tpl line 28]: syntax error: missing section name  
      in /path/to/smarty/Smarty.class.php on line 1041
```

Smarty te muestra el nombre del template, el número de la línea y el error. Después de esto, el error consiste en el número de la línea de la clase Smarty donde ocurrió el error.

Existen ciertos errores que el Smarty no puede entender, tales como un etiqueta de cierre errado. Estos tipos de erros normalmente termina en una interpretacion de error del tiempo de compilacion de PHP.

### Example 17.2. Errores de analisis gramatical de PHP

```
Parse error: parse error in /path/to/smarty/templates_c/index.tpl.php on line 75
```

Cuando usted encuentra un error de analisis de PHP, el número de la línea de error corresponde al script PHP compilado, no al template en si. Normalmente usted puede en el template localizar el error de sinxis. Algunas cosas que usted puede buscar: falta de cierre de etiquetas para `{if}{/if}` o `{section}{/section}`, o sintaxis de la lógica dentro de una etiqueta `{if}`. Si usted no encuentra el error, usted tendra que abrir el archivo PHP compilado y dirigirse al número de línea mostrado, donde el correspondiente error esta en el template.

### Example 17.3. Otros errores comunes

•

```
Warning: Smarty error: unable to read resource: "index.tpl" in...  
or  
Warning: Smarty error: unable to read resource: "site.conf" in...
```

- El `$template_dir` no existe o es incorrecto, o el archivo `index.tpl` no esta en la carpeta `templates/`
- La función `{config_load}` esta dentro del template (o `config_load()` habia sido llamado) y cualquiera de los dos `$config_dir` es incorrecto, no exista o `site.conf` no existe en el directorio.

•

```
Fatal error: Smarty error: the $compile_dir 'templates_c' does not exist,  
or is not a directory...
```

Cualquiera de las dos el `$compile_dir` es asignado incorrectamente, el directorio no existe, o `templates_c` es un archivo y no un directorio.

•

```
Fatal error: Smarty error: unable to write to $compile_dir '....
```

El `$compile_dir` no puede ser reescrito por el servidor web. Vea a fondo la pagina de permisos del instalación de smarty.

•

```
Fatal error: Smarty error: the $cache_dir 'cache' does not exist,  
or is not a directory. in /..
```

Esto significa que `$caching` es habilitado y cualquiera de los dos; el `$cache_dir` es asignado incorrectamente, o el directorio no existe, o `cache` es un archivo y no un directorio.

•

```
Fatal error: Smarty error: unable to write to $cache_dir '/...
```

Esto significa que `$caching` es habilitado y el `$cache_dir` no puede ser rescrito por el web server. Ver ampliamente la pagina de permisos de la instalacion de smarty.

Vea también `debugging`, `$error_reporting` y `trigger_error()`.

---

# Chapter 18. Consejos y Trucos

## Manipulación de Variables Vacías

Cuando usted en algunas ocasiones quiere imprimir un valor que usted define a una variable vacía en vez de imprimir nada, tal como imprimir "&nbsp;" a fin de que el plano del fondo de la tabla funcione correctamente. Muchos usarían una sentencia `{if}` para manejar esto, mas existe otra forma con Smarty, usando el modificador de la variable *default*.

### Example 18.1. Imprimiendo &nbsp; cuando una variable esta vacia

```
{* the long way *}

{if $title eq ""}
    &nbsp;
{else}
    {$title}
{/if}

{* the short way *}

{$title|default:"&nbsp;"}
```

Ver también `default` y `Default Variable Handling`.

## Manipulación del valor default de una variable

Si una variable es usada frecuentemente en sus templates, aplicando el modificador `default` toda vez que este es mencionado puede evitar un bit desagradable. Usted puede remediar esto con la atribución de un valor por default a la variable con la función `{assign}`.

### Example 18.2. Atribuyendo el valor por default a una variable en el template

```
{* ponga esto en algun lugar en la parte de arriba de su template *}
{assign var="title" value=$title|default:"no title"}

{* Si el $titulo estaba vacio, este ahora tendra el valor "sin titulo" cuando
usted lo exhiba *}
{$title}
```

Vea también `default` y `Blank Variable Handling`.

# Pasando la variable titulo a la cabecera del template

Cuando la mayoría de sus templates usan los mismo encabezados y los mismos pies de pagina, es común dividirlos uno en cada template y entonces incluirlos {include}. Que pasara si el encabezado necesita tener un titulo diferente, dependiendo de que pagina estas viniendo? usted puede pasar el titulo en el encabezado cuando este es incluido.

## Example 18.3. Pasando la variable titulo al encabezado del template

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>
```

Cuando la pagina principal es mostrada, el titulo de la "Página Principal" es pasado al template `header.tpl`, y será posteriormente usado como el titulo. Cuando la pagina de archivo es mostrada, el titulo sera "Archivos". Observelo en el ejemplo de archivo, nosotros estamos usando una variable del archivo `archives_page.conf` en vez de una variable codificada rigida. Tambien note que "BC news" es mostrada si la variable `$titulo` no esta definida, usando el modificador de la variable default.

## Fechas

Como una regla basica, siempre pase fechas al Smarty como timestamps. Esto permite al diseñador de template utilizar `date_format` para el control completo sobre el formato de fechas, y también facilita la comparación de fechas si es necesario.

### Note

En el Smarty 1.4.0, usted puede pasar fechas al Smarty como timestamps unix,mysql, o cualquier otra fecha interpretable por `strtotime()` [<http://php.net/strtotime>].

### Example 18.4. Usando `date_format`

```
{ $startDate | date_format }
```

Esta es la salida:

```
Jan 4, 2001
```

```
{ $startDate | date_format : "%Y/%m/%d" }
```

Esta es la Salida:

```
2001/01/04
```

```
{ if $date1 < $date2 }  
  ...  
{ /if }
```

Cuando usa {html\_select\_date} en un template, el programador normalmente va a querer convertir la salida de un formulario de vuelta al formato timestamp. Aquí esta una función para ayudar con esto.

### Example 18.5. Convirtiendo elementos en forma de fecha de vuelta a un timestamp

```
<?php
// esto asume que la forma de sus elementos son nombradas como
// startDate_Day, startDate_Month, startDate_Year

$startDate = makeTimeStamp($startDate_Year, $startDate_Month, $startDate_Day);

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);
}
?>
```

Vea también {html\_select\_date}, {html\_select\_time}, date\_format y \$smarty.now,

## WAP/WML

Los templates WAP/WML requieren de un encabezado de Content-Type [<http://php.net/header>] de PHP para ser pasado junto con el template. La forma mas fácil de hacer esto seria escribir una función de manera habitual que imprima el encabezado. Si usted esta usando el sistema de cache, este no funcionara, entonces nosotros haremos esto usando una etiqueta de {insert} (recuerde que las etiquetas insert no son "cacheadas!"). Asegurarse que no exista ninguna salida al navegador antes del template, de otro modo el encabezado fallara.

**Example 18.6. Usando insert para escribir un encabezado WML Content-Type**

```

<?php

// Asegurarse que el apache esta configurado para las extenciones .wml !
// ponga esta función en algun lugar de su aplicación, o en Smarty.addons.php
function insert_header($params)
{
    // this function expects $content argument
    if (empty($params['content'])) {
        return;
    }
    header($params['content']);
    return;
}

?>

```

Su template de Smarty *debe* comenzar con la etiqueta insert, como en el ejemplo:

```

{insert name=header content="Content-Type: text/vnd.wap.wml"}

<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/wml11.dtd">

<!-- 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>

```



# Plantillas con Componentes

Tradicionalmente, programar plantillas en sus aplicaciones es de la siguiente forma: Primero, usted acumula sus variables dentro de su aplicación PHP, (talvez como requisiciones de una base de datos). Entonces, usted instancia su objeto Smarty assign(), atribuye valores a las variables y muestra el template display(). Por ejemplo nosotros tenemos un registrador de existencias en nuestro template. Nosotros recolectaremos los datos de las existencias en nuestra aplicación, entonces damos valor a estas variables en el template y lo mostramos. Ahora esto sería genial si usted adicionara este registrador de almacenamiento (stock ticker) a cualquier aplicación simplemente incluyendolo en el template, y no preocuparse acerca de como ir a traer los datos al frente?

Usted puede escribir este plugin haciendo que traiga un contenido y asignarlo a la variable del template.

## Example 18.7. Plantillas con Componentes

function.load\_ticker.php - deja el archivo en \$plugins directory

```
<?php

// deja el archivo "function.load_ticker.php" en el directorio de plugins

// configura nuestra función para traer los datos almacenados
function fetch_ticker($symbol)
{
    // ponga la lógica aquí que traera $ticker_name
    // y $ticker_price de algun recurso
    return $ticker_info;
}

function smarty_function_load_ticker($params, &$smarty)
{
    // llama la función
    $ticker_info = fetch_ticker($params['symbol']);

    // asigna las variables al template
    $smarty->assign($params['assign'], $ticker_info);
}
?>
```

index.tpl

```
{load_ticker symbol="YHOO" assign="ticker"}

Stock Name: {$ticker.name} Stock Price: {$ticker.price}
```

Vea también {include\_php}, {include} y {php}.

## Ofuscando direcciones de E-mail

Usted desea saber como su direccion de E-mail consigue entrar en tantas listas de e-mail de spam? Una direccion unica spammers recolecta direcciones de E-mail y de paginas web. Para ayudar a combatir este problema, usted puede hacer que su direccion de E-mail aparesca en javascript mostrado en el codigo HTML, este mismo aparecera y funcionara correctamente en el navegador. Esto se puede hacer con el plugin {mailto}.

### **Example 18.8. Ejemplo de ofuscamiento de una direccion de E-mail**

```
{* in index.tpl *}
```

```
Send inquiries to
```

```
{mailto address=$EmailAddress encode="javascript" subject="Hello"}
```

### **Nota Técnica**

Este metodo no es 100% a pueba de fallas. Un spammer podría crear un programa para recolectar el e-mail y para decodificar estos valores, mas no es muy común.

Vea también escape y {mailto}.

---

# Chapter 19. Fuentes

La pagina principal del Smarty está localizada en <http://www.smarty.net/>. Usted puede ingresar a la lista de email enviando un e-mail a [ismarty-discussion-subscribe@googlegroups.com](mailto:ismarty-discussion-subscribe@googlegroups.com). El archivo de la lista de e-mail puede ser visto en <http://groups.google.com/group/smarty-discussion>.

---

# Chapter 20. ERRORES

Revise el archivo de BUGS que viene con la ultima distribución del Smarty, o Revise el website.