CODE > PHP

# Introduction to the Smarty Templating Framework

by Joeri Rammelaere 14 Sep 2010

Difficulty: Beginner Length: Long Languages: English ▾

PHP Web Development Smarty

Smarty is a PHP-based templating engine/framework. It allows you to further separate your business logic from its visualization, by removing as much PHP code as possible away from your views. Some developers and frameworks prefer not to use a templating engine, others do prefer them to using plain PHP in your views. Both points of view can be argued, and in the end, it's mostly a matter of taste. Anyway, it's never a bad idea to try it out before deciding not to use it, and that's what this tutorial is about: trying out the Smarty Templating Framework.

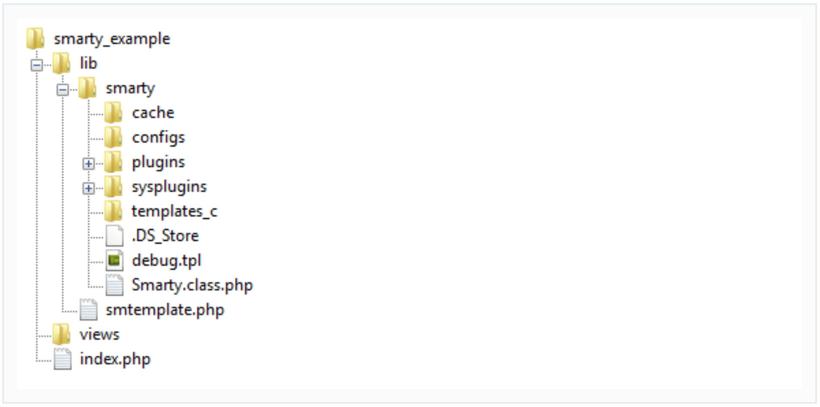
## Step 0: What To Expect

At the end of this tutorial, you'll have a basic idea of how Smarty works. You'll be able to load template files, pass variables to them, use a "layout" in which your other views are inserted, and write your own modifiers. This will all be accomplished using an additional wrapper class, which you can easily integrate in your existing projects.

## Step 1: Setting Up The Project

The project for this tutorial will have a very easy setup, since we're not developing a real application. Just create a project folder (mine is named "smarty_example") with an index.php file in it, and a directory called "lib" inside of it. Also, create a file named smtemplate.php in the "lib" folder. Next, create a "views" folder inside "smarty_example". This folder will contain our smarty template files.

Before you're able to use something, you have to install it. Thankfully, installing Smarty is extremely easy and requires almost no configuration. First of all, download Smarty and extract the archive. You can check out everything inside the archive, but we'll only need the "libs" folder for our application. Rename it to "smarty" and paste it inside the "lib" folder of our application. Smarty uses some additional folders, so create the "templates_c", "cache" and "configs" folders inside our "lib/smarty" folder. If you're not using Windows, you'll have to give 775 permissions on these folders to your webserver. Your directory tree should now look like this:


```
smarty_example
├─ lib
│  ├─ smarty
│  │  ├─ cache
│  │  ├─ configs
│  │  ├─ plugins
│  │  ├─ sysplugins
│  │  ├─ templates_c
│  │  ├─ .DS_Store
│  │  ├─ debug.tpl
│  │  └─ Smarty.class.php
│  └─ smtemplate.php
├─ views
└─ index.php
```

## Step 2: Creating The SMTemplate Class

Every programmer has his own idea about the ideal API. In order to adjust Smarty's API slightly, and allow us to add some additional functionality, we'll create a wrapper class called SMTemplate, which will take care of the smarty details for us. This approach has another advantage: if, at one moment in time, you should choose to use another template engine, you can create a wrapper for that engine, while retaining the SMTemplate interface, and thus without breaking the code that uses our SMTemplate class.

## Storing Your Configuration

Before coding the SMTemplate class functionality, we'll need a place to store some configuration details. You can do this in multiple ways, i.e. by defining config options as class constants, by defining them as constants in the smtemplate.php file, or by keeping them in a separate config file. I prefer the last option, so I'll create an smtemplate_config.php file. Smarty needs configuration for the template, compiled template, cache, and config directories. Later, we might also add SMTemplate specific options to our config file, but for now, this will do:

```php
/**
 * @file
 * Configuration file for the SMTemplate class
 */

$smtemplate_config =
    array(
        'template_dir' => 'views/',
        'compile_dir' => 'lib/smarty/templates_c/',
        'cache_dir' => 'lib/smarty/cache/',
        'configs_dir' => 'lib/smarty/configs/',
        );
```

## Building the SMTemplate Class

The SMTemplate class will load this config file, and pass the options to Smarty. Before we can pass the options, we'll need an object of class Smarty. Our SMTemplate class could extend the Smarty class, but I prefer to use a private instance variable to contain the Smarty object. So far, we have the following for our SMTemplate class:

```php
/**
 * @file
 * Wrapper for Smarty Template Engine
 */

require_once('smarty/Smarty.class.php');
require_once('smtemplate_config.php');

class SMTemplate{

    private $_smarty;

    function __construct(){
        $this->_smarty = new Smarty();

        global $smtemplate_config;
        $this->_smarty->template_dir = $smtemplate_config['template_dir'];
        $this->_smarty->compile_dir = $smtemplate_config['compile_dir'];
        $this->_smarty->cache_dir = $smtemplate_config['cache_dir'];
        $this->_smarty->configs_dir = $smtemplate_config['configs_dir'];
    }
}
```

## Rendering templates

As you can see, our class is still pretty pathetic, as it can't render anything. We'll solve this issue by adding a render function, which loads a template and displays it.

```php
function render($template){
    $this->_smarty->display($template . '.tpl');
}
```

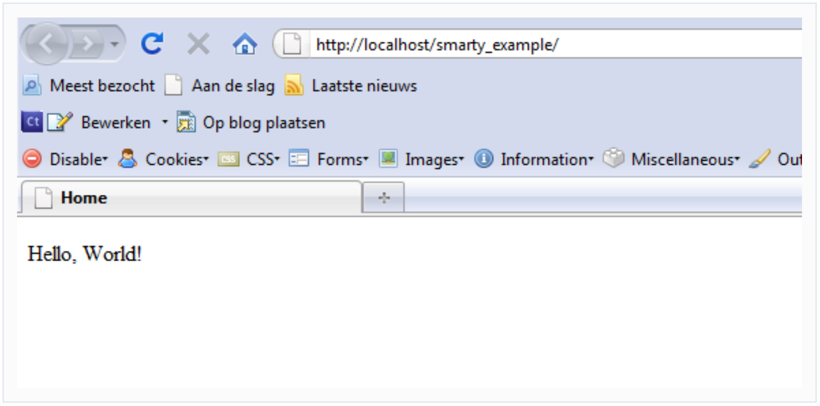
In order to render anything, we'll need to create a template file, and then call the render function from our index.php file. The template file will be pretty basic, containing a simple html page. Name it "home.tpl", and place it inside our "views" directory.

file will be pretty basic, containing a simple html page. Name it "home.tpl", and place it inside our "views" directory.

```
<html>
    <head>
        <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
        <title>Home</title>
        <link rel="stylesheet" href="/css/master.css" type="text/css" media="screen" title="no title" charset="utf-8" />
    </head>
    <body>
        <p>Hello, World!</p>
    </body>
</html>
```

Now, all that is left is to create an SMTemplate object and render 'home'. Open up index.php, add the following lines of code, and navigate there in your browser.

```
require_once('lib/smtemplate.php');

$tpl = new SMTemplate();
$tpl->render('home');
```



```
Hello, World!
```

## Step 3: Assigning and Formatting Variables

If we couldn't render anything dynamically, Smarty would be pretty useless. Luckily, we can assign variables to our smarty class, and display those in our template. We can also use some Smarty functions (well, modifiers actually) to format them the right way.

### Passing an Array of Variables

Though Smarty supports the assignment of variables, our SMTemplate doesn't (yet). We'll provide the CodeIgniter-style of assignment, where you pass an array to the render function. You can adapt SMTemplate to support other methods as well; for example, assigning them to the object and then using __set to store them in an array is also a clean way. For this tutorial though, passing an array will do. Before assigning the variables, we'll edit our template to something a little more dynamic. Saying hello to the world is customary for programmers, but not very useful, so let's use a variable to determine who we're hello-ing. Secondly, we'll add today's date to the message. Variables can be displayed by wrapping them in curly brackets.

```
<body>
    <p>Hello, {$receiver}! It's {$date} today!</p>
</body>
```

If you refresh the page, you'll see that the variables haven't been filled in, since we didn't set them. Setting variables can be done using smarty->assign, so let's assign them. The render function will now take an optional data array as a second argument.

```
function render($template, $data = array()){
    foreach($data as $key => $value){
        $this->_smarty->assign($key, $value);
    }
    $this->_smarty->display($template . '.tpl');
    $this->_smarty->display($template . '.tpl');
}
```

It still won't work, because we don't pass in an array when calling our render function. We can easily do this, by altering a few lines in our index.php file.

```
$data = array(
    'receiver' => 'JR',
    'date' => time(),
    );

$tpl = new SMTemplate();
$tpl->render('home', $data);
```

If you refresh now, the page will say something like "Hello, JR! It's 1282810169 today!". Of course, this date isn't really what we had in mind. It needs to be formatted, which brings us to the next section.

### Using Modifiers to Format Variables

Smarty isn't just a template engine that searches and replaces variables. It's also a powerful framework, that allows you to save time by using things like modifiers, functions, and blocks. If we wish to format our date, for example, we can use the date_format modifier. To apply a modifier to a variable, simply put a pipe character and the modifier name behind it, followed by the optional arguments which are separated by colons. The date_format modifier takes a string argument, which represents the format the date will take, and an optional default date, which we won't need. The following code will display the date as "day (in decimals) Month".

```
<body>
    <p>Hello, {$receiver}! It's {$date|date_format:"%d %B"} today!</p>
</body>
```

This should now give something of the form "Hello, JR! It's 26 August today!" Now, maybe we want to make sure our receiver is uppercased. We can achieve that by using the upper modifier.

```
<body>
    <p>Hello, {$receiver|upper}! It's {$date|date_format:"%d %B"} today!</p>
</body>
```

Now, if I alter index.php to pass 'jr' instead of 'JR', the template will still show 'JR'. Easy, isn't it? Next, we'll include our templates in a default "layout".

## Step 4: Working With a Layout

Before we alter our SMTemplate class to enable layouts, we'll create a layout first. Create a new directory named "layouts" inside our "smarty_example" folder and move home.tpl there. Rename it to 'page.tpl'. We'll remove our previous 'hello world' content, and put two horizontal lines in. Our content will be placed in between these lines.

```
<html>
    <head>
        <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
        <title>Home</title>
        <link rel="stylesheet" href="/css/master.css" type="text/css" media="screen" title="no title" charset="utf-8" />
    </head>
    <body>
        <hr />
        <hr />
    </body>
</html>
```

Of course, this won't cut it, since Smarty won't know where to insert our content. There is more than one way to get content from another template inside of our layout, and I'll use Smarty's fetch function. This function returns our template as text, instead of another template inside of our layout, and I'll use Smarty's fetch function. This function returns our template as text, instead of displaying it. This means we can fetch the template, and then assign it to a variable for use within our template! This variable's name is yours to choose. I prefix my special variables with __, to distinguish them from the other variables I use. I'll call this one 'content', since we're assigning our page content to it.

```
<body>
    <hr />
        {$__content}
    <hr />
</body>
```

This concludes our layout, so let's create some templates to use as content. I'll create a 'hello' template, which will contain a standard 'hello world' line, and a 'lipsum' template, which holds some Lorem Ipsum text. Don't forget to give these templates a .tpl extension.

```
<p>Hello, World!</p>
```

```
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean aliquet dignissim diam at vulputate. Aenean nec ligula a
```

Adapting our SMTemplate class to use a layout is also extremely easy. We'll first set up a configuration option for the layouts directory, like we did for our views.

```
/**
 * @file
 * Configuration file for the SMTemplate class
 */

$smtemplate_config =
    array(
        'layouts_dir' => 'layouts/',
        'template_dir' => 'views/',
        'compile_dir' => 'lib/smarty/templates_c/',
        'cache_dir' => 'lib/smarty/cache/',
        'configs_dir' => 'lib/smarty/configs/',
        );
```

Next, we'll change our render function. We'll supply the layout as an optional third parameter, and let it default to 'page'. Then, we'll fetch the requested template, assign it to the $__content variable, and display our layout.

```
function render($template, $data = array(), $layout = 'page'){
    foreach($data as $key => $value){
        $this->_smarty->assign($key, $value);
    }
    $content = $this->_smarty->fetch($template . '.tpl');
    $this->_smarty->assign('__content', $content);
    $this->_smarty->display($layout . '.tpl');
}
```

There are a couple of things to consider, regarding this code. First of all, we haven't told Smarty where to find our layouts yet. We can do that by adding a template dir, but this approach means we can't give our layouts the same name as our templates - Smarty wouldn't know which one to pick. We could solve this by giving our layouts a different extension, or by setting and resetting our template directory inside our render function, or by using more advanced Smarty functions. For now, we'll just settle with the constraint that layouts and views can't have the same name. We can add our layouts directory using the addTemplateDir() function.

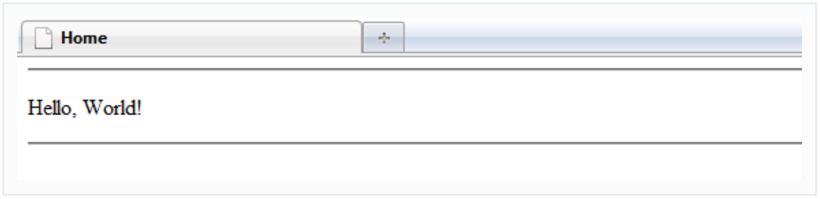
```
function __construct(){
    $this->_smarty = new Smarty();

    global $smtemplate_config;
    $this->_smarty->template_dir = $smtemplate_config['template_dir'];
    $this->_smarty->addTemplateDir($smtemplate_config['layouts_dir']);  // <- new line
    $this->_smarty->compile_dir = $smtemplate_config['compile_dir'];

    $this->_smarty->cache_dir = $smtemplate_config['cache_dir'];
    $this->_smarty->configs_dir = $smtemplate_config['configs_dir'];
}
```

```
}
```

Let's check it out by changing our index.php file again.

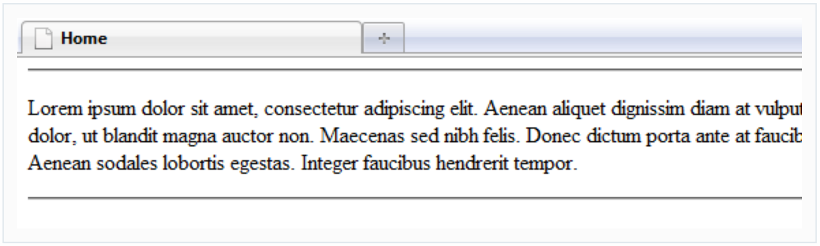
```
require_once('lib/smtemplate.php');

$tpl = new SMTemplate();
$tpl->render('hello');
```

It works!



Hello, World!

And if we change it to render 'lipsum', it works as well:



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean aliquet dignissim diam at vulput
dolor, ut blandit magna auctor non. Maecenas sed nibh felis. Donec dictum porta ante at faucib
Aenean sodales lobortis egestas. Integer faucibus hendrerit tempor.

## Step 5: Creating Your Own Modifiers

As the final part of this tutorial, I'll introduce one of Smarty's more advanced features, that make it more than a simple templating engine. Smarty contains a number of standard functions and modifiers, but it's also extremely easy to create your own. Let's have a look at the modifier we used to format our date:

```
{$date|date_format:"%d %B"}
```

> *If you want a custom modifier, all you need to do is write a PHP function.*

This will actually result in a call to the function smarty_modifier_date_format(), with $date and our format string as arguments. This function will return a string, and this string will be displayed. So if you want a custom modifier, all you need to do is write a PHP function. As an example, we'll write a modifier called 'weirdcase', which will uppercase all consonants and lowercase all vowels, i.e. 'Lorem Ipsum' becomes 'LoReM IPSuM'. To do this, create a file called 'modifier.weirdcase.php' in the 'lib/smarty/plugins' folder. Our modifier will take only one argument, the string that needs to be altered.

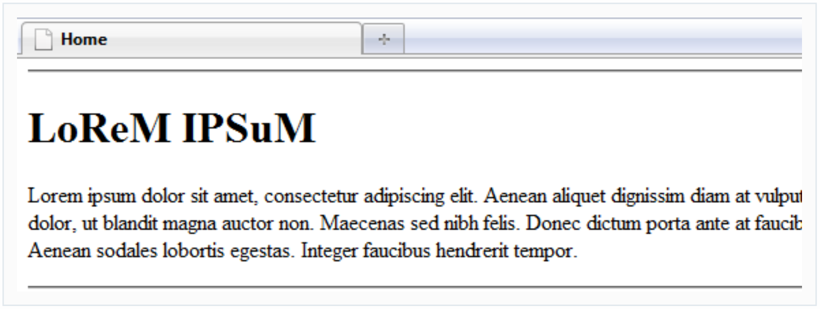
```
/**
 * Smarty weirdcase modifier plugin
 *
 * Type:     modifier
 * Name:     weirdcase
 * Purpose:  turn consonants into uppercase and vowels into lowercase
 * @param string
 * @return string
 */

function smarty_modifier_weirdcase($string){


}
```

We can get our result by defining an array 'vowels', turning our string into an array and then traversing it, and checking whether each character is in our vowels array. If it is, we lowercase it, otherwise, we uppercase it. The modified characters are then appended to a result variable.

```
function smarty_modifier_weirdcase($string){
    $str_array = str_split($string);
    $result = '';
    $vowels = array('a', 'e', 'i', 'o', 'u');

    foreach ($str_array as $char){
        if (in_array($vowels, $char)) $result .= strtolower($char);
        else $result .= strtoupper($char);
    }

    return $result;
}
```

This should do the trick, so let's check it out. Edit the 'lipsum.tpl' template and add an h1 containing our weirdcased 'Lorem Ipsum' to it.

```
<h1>{'Lorem Ipsum'|weirdcase}</h1>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean aliquet dignissim diam at vulputate. Aenean nec ligula a
```



LoReM IPSuM

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean aliquet dignissim diam at vulput
dolor, ut blandit magna auctor non. Maecenas sed nibh felis. Donec dictum porta ante at faucib
Aenean sodales lobortis egestas. Integer faucibus hendrerit tempor.

## Step 6: Conclusion

Although there is a lot more to Smarty than I could fit within this tutorial, hopefully this should provide you with a basic knowledge of how to work with it. You essentially already know everything you *need* to know. You should also be able to determine whether you like the idea of using this templating framework or not by now. The more advanced topics, such as filters and blocks, are useful, however, you'll still do fine without them. You can find documentation on the more advanced features at the Smarty website. Thanks for reading!

Joeri Rammelaere

I'm a 19 year old Computer Science student at the University of Antwerp, with a broad knowledge of programming language. My main experience is with the C++ and Oberon-2 languages, though I also know other languages, such as Ruby and Haskell. I'm also very interested in both webdesign and webdevelopment.

**Mina Kolta** • 5 years ago
This is All just fine and really good hands-on tutorial except :
1. smarty_modifier_weirdcase() function
the in_array parameters should be

in_array($char,$vowels);

instead of

in_array($vowels,$char);

2.in Building the SMTemplate Class section we should use proper methods for setting smarty variables

$this->_smarty->setTemplateDir($smtemplate_config['template_dir']);
$this->_smarty->setCompileDir($smtemplate_config['compile_dir']);
$this->_smarty->addTemplateDir($smtemplate_config['layouts_dir']);
$this->_smarty->setCacheDir($smtemplate_config['cache_dir']);
$this->_smarty->setConfigDir($smtemplate_config['configs_dir']);

Great Post
regards,
minakolta
5 • Reply • Share ›

**Adam** • 6 years ago
In configuration file, the Configs Dir variable should be 'config_dir' not 'configs_dir'.
5 • Reply • Share ›

**Vinicius** ➜ Adam • 6 years ago
Thanks Joeri Rammelaere AND thanks ADAM!
1 • Reply • Share ›

**Oscar** • 3 years ago
Hi:

Very good tuto, wonder, you could share the code with us on some server ..?

Thanks in advance.
2 • Reply • Share ›

**Adamito** • 5 years ago
Hello. I recently moved a website from one web server to another and all navigation and styling is working, but for some reason all of the content templates aren't showing up on the site. Has anyone run into this before? Is a path wrong somewhere, or permissions?
1 • Reply • Share ›

**Rupesh Gardi** • 5 years ago
thanks a lot . . . nice tut
1 • Reply • Share ›

**new here** • 7 years ago
I have done the everything this tutorial says, but I keep getting this when I go to localhost:

require_once('lib/smtemplate.php'); $tpl = new SMTemplate(); $tpl->render('home.tpl');

I checked the places and the structure of the files.

NOT experienced with this, does anybody know what is wrong?
1 • Reply • Share ›

**Raymond Ho** • 7 years ago
Why am I receiving this error: Call to undefined method SMTemplate::render() in C:\wamp\www\smarty_example\index.php on line 5
1 • Reply • Share ›

**Adalberto Reyes** • 2 years ago
Really good tutorial.

Thanks
Reply • Share ›

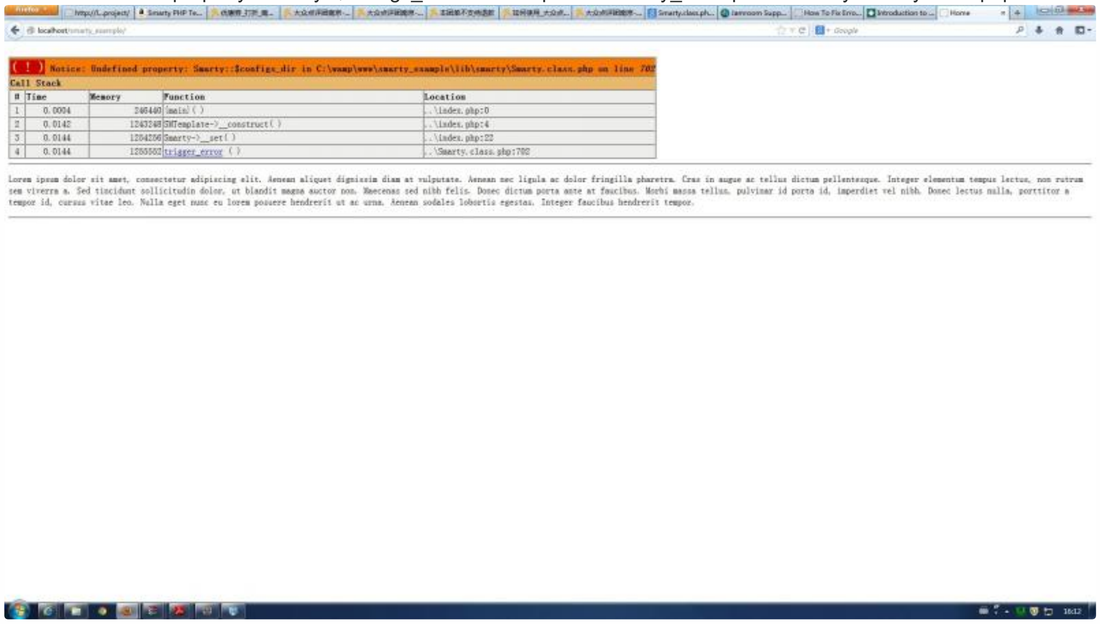
**Md. Zubaer Ahammed** • 2 years ago
Great tutorial. But it should be $config_dir rather than $configs_dir. $configs_dir isn't defined in Smarty.class.php, So it will give Undefined property: Smarty::$configs_dir error if you use $configs_dir. Except this everything is cool.
• Reply • Share ›

**Vasulu Saya** • 3 years ago
Nice tutorial for those who are new to smarty.
• Reply • Share ›

**Chandra** • 4 years ago
Thanks for such a nice tutorial, I have tried 2-3 time to setup smarty as per my requirement , but failed. This tutorial it so brilliantly composed that it answered all my questions till i finished the tutorial.
• Reply • Share ›

**Niyigena John Peter** • 4 years ago
why do i get this error?

Notice: Undefined property: Smarty::$configs_dir in C:\wamp\www\smarty_example\lib\smarty\Smarty.class.php on line 702



```
( ! ) Notice: Undefined property: Smarty::$configs_dir in C:\wamp\www\smarty_example\lib\smarty\Smarty.class.php on line 702
Call Stack
#  Time    Memory   Function                   Location
1  0.0004   246440  {main}( )                  ..\index.php:0
2  0.0142  1243248  SMTemplate->__construct( ) ..\index.php:4
3  0.0144  1254256  Smarty->__set( )           ..\index.php:22
4  0.0144  1255552  trigger_error ( )          ..\Smarty.class.php:702
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean aliquet dignissim diam at vulputate. Aenean nec ligula ac dolor fringilla pharetra. Cras in augue ac tellus dictum pellentesque. Integer elementum tempus lectus, non rutrum sem viverra a. Sed tincidunt sollicitudin dolor, ut blandit magna auctor non. Maecenas sed nibh felis. Donec dictum porta ante at faucibus. Morbi massa tellus, pulvinar id porta id, imperdiet vel nibh. Donec lectus nulla, porttitor a tempor id, cursus vitae leo. Nulla eget nunc eu lorem posuere hendrerit ut ac urna. Aenean sodales lobortis egestas. Integer faucibus hendrerit tempor.

• Reply • Share ›

**abhijit jagtap** Niyigena John Peter • a year ago
Configs Dir variable should be 'config_dir' not 'configs_dir'. Thanks @adam
• Reply • Share ›

**niko** • 5 years ago
I like this tutorial. More you can find on Be a better smarty programmer
• Reply • Share ›

**one call houston tx** • 5 years ago
Hello! I just wish to offer you a big thumbs up for the great information you have
got here on this post. I am returning to your site for more soon.
• Reply • Share ›

**Muhammad Ahmed** • 5 years ago
wooow sounds good
great tutorial I did it without any mistakes
thank you so much
• Reply • Share ›

**junnydc** • 6 years ago
I was looking for a simple shopping cart with smarty as framework. Hope you can help.
• Reply • Share ›

**Douglas** • 6 years ago
everything works fine until i introduce the modifiers. i'm getting an error saying unrecognized tag $date|date_format

Reply • Share ›

**Nanda** ➦ Douglas • 5 years ago

I think you may need to add libraries .!!!

Reply • Share ›

**Tiru** • 6 years ago

Thank you very much for this tutorial. It is short, concise, brief and perfect!! I know 0% about smarty other than knowing it was just template engine. After this tutorial though, I am feeling like I can do a lot with smarty. I might not use it as most of the tasks can be performed using zend easily - but I like the way you provide it.
Thanks again, keep the good job.

Reply • Share ›

**Pankaj** • 6 years ago

This is very good article. Thanks real help.
One other good article - Smarty Template Engine using PHP

http://onlinewebapplication...

Reply • Share ›

**Julius** • 6 years ago

Great tutorial...it works just fine there was just one tiny mistake in the weirdcase modifier the "in_array($vowels, $char)) $result .= strtolower($char); "should look like in_array($char, $vowels)) $result .= strtolower($char);

otherwise great thx

Reply • Share ›

**Thai** • 7 years ago

I use Smarty in my everyday use and I have to tell you that it's does what it says. I like how it allows you to separate the front end process from back end stuff. I'm surprised that I don't see more tutorials on Smarty template anywhere.

Reply • Share ›

**Web Design Sheffield** • 7 years ago

Great tut, but when I navigate to http://localhost/smarty nothing renders has the index page just shows a blank page.

Where should this piece of code go:

```
function render($template){
$this->_smarty->display($template . '.tpl');
}
```

Would the above code go below :

```
/**
* @file
* Wrapper for Smarty Template Engine
*/

require_once('smarty/Smarty.class.php');
require_once('smtemplate_config.php');

class SMTemplate{

private $_smarty;

function __construct(){
$this->_smarty = new Smarty();

global $smtemplate_config;
$this->_smarty->template_dir = $smtemplate_config['template_dir'];
$this->_smarty->compile_dir = $smtemplate_config['compile_dir'];
$this->_smarty->cache_dir = $smtemplate_config['cache_dir'];
$this->_smarty->configs_dir = $smtemplate_config['configs_dir'];
}
}
```

in the smtemplate.php file?

Reply • Share ›

**Bhagu** ➦ Web Design Sheffield • 7 years ago

I have same question too!? thx for posting.

Reply • Share ›

**pixibug** • 7 years ago
I use smarty 2 years, and this template engine integrates seamlessly with a MVC architecture (no slow down, cache system, easy to create your own plugins and modifier). And most importantly, when you work with a team of html graphic designer (creating view), smarty is easily accessible and the code is more clear than php IMO.
Good tutorial.
Reply • Share ›

**Satya Prakash** • 7 years ago
For another smarty template tutorial you can visit the post - http://www.satya-weblog.com...
Reply • Share ›

**mdennisa** • 7 years ago
this is a nice tutorial, but...
I prefer to use codeigniter.

and I have no idea why some peoples still use smarty template for codeigniter --"
Reply • Share ›

**Adam Leonard** • 7 years ago
Smarty is not something I suggest learning. It only adds a layer of complexity, it doesn't solve anything. If you are looking for any sort of templating checking out HAML, it's beautiful.
Reply • Share ›

**gabi** • 7 years ago
hi. i get: Fatal error: Call to undefined method Smarty::addTemplateDir() in /home/www/smarty_example/lib/smtemplate.php on line 18. i searched on the smarty deocumentation and i didn't found this method
Reply • Share ›

**Linh** → gabi • 7 years ago
You must get Smarty version 3.0xx to use that method.
Reply • Share ›

**thomasw** • 7 years ago
The author forgot one important thing: escape your HTML output (modifier "escape"). Or you will die with XSS ;-)

I use smarty to create HTML and PDF documents - yes, PDF, with my own meta language processor! A good way to separate PDF document formatting from code.

To everyone who thinks smarty is a performance killer: Your "compiled" template will be just executed (included) PHP code - nothing else.

Also I think the following code is easier to maintain:

**{$blah|escape}**

or

You can create your own modifiers, e.g. for escaping to UTF-8. With your own functions and modifiers you can also easily create a "higher" level of HTML / web application design.
Reply • Share ›

**Valerij Primachenko** → thomasw • 7 years ago
Compiling of the templates is a nice feature, but i don't think it's just as fast as hand written (and hand optimized) code. Its like compiled C++ (please c++ programmer don't hit me for this comparison) code against pure Assembler for desktop applications.
Reply • Share ›

**thomasw** → thomasw • 7 years ago
OK, now with correct escaping for the samples ;-):

<strong>{$blah|escape}</strong>

or

<strong><?php echo htmlentities($blah); ?></strong>
Reply • Share ›

**nosmartyplease** • 7 years ago
I will use Smarty templating when hell freezes over. Smarty is the enemy of scaling and performance websites.
Reply • Share ›

**Ionut** ➦ nosmartyplease • 7 years ago
I will not agree with that, my company uses Smarty for really big content websites, over 150k unique visitor/day and is working awesome.
^ | v • Reply • Share ›

**Mack** • 7 years ago
To me, template engines arise to the occasion when you have to encode a piece of software, but still allow the customer some form of customization. That is the only time I found smarty useful.
^ | v • Reply • Share ›

**martin** • 7 years ago
smarty is great, I use it mostly for developing typo3 extensions.
^ | v • Reply • Share ›

**Bargok** • 7 years ago
Smarty is great..and this tutorial doesn't show why..shame.
I don't like the wrapper, and things such as globals really make me a sad panda.
^ | v • Reply • Share ›

**aditia** • 7 years ago
owh now I see there are still a lot smarty users, cause since I knowing php framework I rarely using smarty
^ | v • Reply • Share ›

**Arun** • 7 years ago
We use PHPTAL, its way easier than smarty and gets jo done. Perfect for webdesigners like me, who already are profficient in HTML, but dont know PHP.
^ | v • Reply • Share ›

**kemy** ➦ Arun • 7 years ago
agree
^ | v • Reply • Share ›

**waro** • 7 years ago
I use Smarty since 2007 and my designer loves it.
Smarty also has good forum and very helpful members.
^ | v • Reply • Share ›

**Alex** • 7 years ago
I've used Smarty v2 in the past but much prefer Dwoo which is built for php5, similar syntax / functionality just faster. Smarty v3 looks a bit more promising but I wouldn't be using it for a commercial project just yet.
^ | v • Reply • Share ›

**Cristian** ➦ Alex • 7 years ago
Same here, Dwoo really rocks!!
^ | v • Reply • Share ›

**Koren Berman** • 7 years ago
Good intro. More people should definitely be using templating frameworks like Smarty. I was underwhelmed by WP 3.0 when I learned that they had still not implemented anything like that for the templates. Tumblr and EE do it, why does WP lag behind on this? not sure.
^ | v • Reply • Share ›

**Springmann** ➦ Koren Berman • 7 years ago
You can do it by yourself, using any template framework you want (including Smarty). Anyway, the way that WP works is awesome and much more interesting and flexible than Drupal or Joomla, for example.
^ | v • Reply • Share ›

**exped** • 7 years ago
Forget about Smarty. It has so many annoying "features": no method chaining, JS/CSS unfriendly, slow, poor API).
Just switch to Twig Template Engine. Far more better. Supports template inheritance (page layout belongs to view layer, not controller), is JS/CSS friendly, relatively fast (comparing to raw PHP), easy to extend.
^ | v • Reply • Share ›

**Steve** • 7 years ago
Never hurts to learn something new, but I would never put all your eggs into the "Smarty" basket. Before the advent of great PHP frameworks it had it's uses, but now...

Go with any framework and write some real MVC code :P

Go with any framework and write some real MVC code !!
• Reply • Share ›

**Aleksandar Babic** • 7 years ago
Smarty is good and have some nice features like caching. It is not difficult to learn.

But I don't see any advantage to create and use SMTemplate class on this way. If you ever want to migrate to some other template system, you will have other, much greater problems.
• Reply • Share ›

**IT Village** • 7 years ago
I am big fan of nettuts+. great post. can I share it on my new website?
• Reply • Share ›

**Jaspal Singh** • 7 years ago
Excellent tutorial for beginners.
Thanks for sharing.
• Reply • Share ›

**Nilson** • 7 years ago
Great.

I would like to learn more about Cache.
• Reply • Share ›

**Dave k** • 7 years ago
Was happy to see this, I've wanted to look in to smarty for a while. Be good to see a follow up tutorial
• Reply • Share ›

**went_to_france** • 7 years ago
'global'? really? why not pass that config array into the constructor?
• Reply • Share ›

**Steve Maggs** • 7 years ago
From my limited exposure to Smarty it seemed the biggest problem for a commercial website was caching, which caused lots of problems and meant that to push through changes made using the company's CMS the Smarty cache had to be manually cleared frequently.

As a templating engine it's one of the clearest but I'm with the others here who try to avoid adding that extra un-necessary layer of code whenever possible.
• Reply • Share ›

**Philip Wallage** • 7 years ago
MODx satisfies all my needs! Smarty has always been on my to-do list but I just havent found any motivation to kick-start it.
• Reply • Share ›

**Ben McRae** • 7 years ago
I used Smarty2 commercially for almost 2 years and it proved very successful. I wouldnt worry about the time a smarty template takes to compile when you have issues with DB access/queries to worry about.

Im looking forward to see whats changed in version 3! Im sure this will be a great tutorial :)
• Reply • Share ›

**Abdullah Al Mamun** • 7 years ago
Very Basic ignoring some factors like caching.
I hope some follow-up tutorials is coming next.
Thanks. :-)
• Reply • Share ›

**vestimir** • 7 years ago
IMO Smarty is great for teaching you how to separate the html from the php, I used it for almost 5 years, and now when I have this culture of separation I am back to PHP templates and I am happy. With simple one file class, that does not use regex parsing for the templates, everything gets instant faster.
• Reply • Share ›

**Bratu Sebastian** • 7 years ago
Smarty is complicated, but you might need it in a certain project. I had to learn it before, when I had a client with an existing store.

It was awful. But I get the point with Smarty. It's just not my thing, because I write my own {tag} parser or use CodeIgniter's, which is great!

• Reply • Share ›

**Forbs** • 7 years ago
Great tutorial,thanks!
• Reply • Share ›

**Waldo** • 7 years ago
I can agree with the posts that Smarty is not really necessary just for the template-thing, but Smarty gives more then just seperating your PHP-code from your HTML.
I think that the caching-solution is also a good add-on for your project. I haven't worked with another caching solution but Smarty, so maybe there are other solutions for that, but template + caching makes it a nice addon for my PHP-projects..
• Reply • Share ›

**Codeforest** • 7 years ago
I think that learning yet another syntax is a little bit overkill. You can achieve the same thing with plain PHP.

But still, I like this tutorial as it gives me an inside look to Smarty without me having to learn it.
• Reply • Share ›

**Sergio** • 7 years ago
Is this project still alive?
• Reply • Share ›

**Frank** Sergio • 7 years ago
haha :)

i stopped using smarty when they started supporting chained method calling from objects
• Reply • Share ›

**Vijay Padhariya** • 7 years ago
Dude.

Really interesting Article though, I was using Smarty before a year ago, but i avoid using it, as it costs Slowing down my site a little bit, some of my clients told me that it causes this problem so i refused to use it.

you see as we use this engine, php's process become double.

1. its parses HTML data into PHP
2. again PHP parser parses PHP Template into HTML to give HTML view to client side.

so its good not to use, better use inbuilt templating serviced provided by good frameworks like codeignator, symfony, magento etc etc.

Thanks,
Vijay Padhariya
• Reply • Share ›

**SAPONO** • 7 years ago
waiting the part 2 of this introduction......
• Reply • Share ›

**kankuro** • 7 years ago
It doesn't matter if you use that framework template or not.... as long as you feel comfortable for it, then go and if not then don't use it.... find a way that you feel comfortable on using those various apps.

nettuts provide us a great tutorial, if we like it or not.... long live nettuts.... the best.
• Reply • Share ›

**Adeel Ejaz** • 7 years ago
Great tutorial but I've moved on from Smarty last year. It was mainly because of the braces and javascript. Now I use twig template engine and haven't looked back since. Its definitely worth a try. Miles ahead of Smarty :)
• Reply • Share ›

**Luke** • 7 years ago
I have tried to use smarty before but couldn't get my head around it. It seems kind of pointless to me. I never bothered to dig much into it though because it didn't seem like it was even worth using. If I was creating an app I planned on redistributing I might use it but for my own projects I don't need to add this overhead.
• Reply • Share ›

**DJ** • 7 years ago

You know, as long as we are 'just trying it out' before we decide whether to like it or not, as well written as this is, it would make a great screencast so we could actually *see* it work from someone who had already gone to the trouble of downloading and installing it before we decided whether or not to invest in the effort and band-width. There are some things better seen than explained, this may be one of them. How about making this a screencast?

^ | ˅ • Reply • Share ›

**Ryan** • 7 years ago

Is this suppose to be a CMS or something?

^ | ˅ • Reply • Share ›

**Djinn** • 7 years ago

Oh great! thanks ... can you do more tutorials about smarty? i'm really interested in this. Thank you

^ | ˅ • Reply • Share ›

**Joe S** • 7 years ago

I've been wanting to learn how to use Smarty more as I've had a couple projects that needed it. It would be nice if this could be a series. :D

^ | ˅ • Reply • Share ›

**MidnightLightning** • 7 years ago

Using a "Layout" as described here bypasses the Smarty caching system. When I first started using Smarty, I did something similar to this (using a "main.tpl" layout file that used {include $content} to render the page; no custom class needed). Problem with this method of including the content from another template is if Smarty caching is used, every site uses the one layout template, so all gets cached as the same page, creating an illusion that all pages of the site look the same. A better solution that I work with now is to have a "header.tpl" and "footer.tpl" file that are {include}d in every content template. That way the content template gets cached, not the layout template.

And, any reason you went off and made your own class object, rather than extending the Smarty class object with your added functionality?

^ | ˅ • Reply • Share ›

**Stefan** • 7 years ago

I don't understand the use of this. Smarty just adds another layer of complexity.
PHP itself was made to included within HTML files, etc.

You just have to be disciplined enough to strictly divide your business logic and views.

^ | ˅ • Reply • Share ›

**Thad Bloom** • 7 years ago

Thanks,

I do a lot of work with CS-Cart at work which makes use of Smarty. I sometimes find myself guessing when modifying the code so hopefully going over this thoroughly will help save me some frustration :P

^ | ˅ • Reply • Share ›

**zoran** • 7 years ago

Smarty sucks, i agree with Luca, but some projects require you to use Smarty even if you don't like it. Whatever it is, it's good to know it, so thank you for the tutorial.

^ | ˅ • Reply • Share ›

**bl_nk** • 7 years ago

A lot of controversy surrounds using a template engine around a templating language (which is what PHP was created to do). Lots of experts advise against it, but the main road to expertise(sp?) is clear code and smarty's is one of the clearest ones. PHP is tolerable with shorthand, but short tags are a no-no, so my PHP templates border on unreadable no matter how much I try to beautify it (and I usually do, which takes time).

I'm using smarty3 for my projects now and do it almost exactly as this excellent article suggests (nettuts has risen in quality by the way!).

^ | ˅ • Reply • Share ›

**bl_nk** ➦ bl_nk • 7 years ago

The comment system stripped from the
>PHP is tolerable with shorthand
sentence.

^ | ˅ • Reply • Share ›

**Luca** • 7 years ago

smarty pretty sucks IMHO

^ | ˅ • Reply • Share ›

**Flatline** ➦ Luca • 7 years ago

Couldn't agree more with both of you (Luca and Ryan Allen)

^ | ˅ • Reply • Share ›

**Raitis Stengrevics** ➜ Flatline • 7 years ago

Imho, Ryan's right... =P

• Reply • Share ›

**Adam C** ➜ Luca • 7 years ago

Why?
IMO, Smarty is great. It makes your templates file much easier to read (i.e., no more in the html) and it also has caching system, which makes your page load faster.

• Reply • Share ›

**Ryan Allen** ➜ Adam C • 7 years ago

PHP itself is a templating language, the fact that you have to wrap your code in tags indicates this.

By using smarty you give away the ability to do things in straight up PHP, requires you to learn yet another syntax for doing simple things like if statements, and the reason why they have template cache is because Smarty is slow compared to straight up PHP.

I think it's an abstraction that is too confusing for beginners, and in practice I don't see any massive added benefit from using it. A better guideline would be "set up all your variables, get stuff out of the database, figure out access and stuff in one script, then require another php script that is mostly html and makes use of these plain old variables".

That way you get the separation of concerns (fetching data, checking access versus putting it all together in HTML), without the overhead of yet-another-library that is slower and doesn't give you any extra functionality!

All you need in my opinion is something like PHP ADO and a couple of guidelines like I've mentioned above and you can write easy to understand code.

• Reply • Share ›

**Julien Tant** • 7 years ago

Great tutorial, i use smarty at work and it's pretty cool.

• Reply • Share ›

**mnowluck** • 7 years ago

thanks a lot for this..

• Reply • Share ›