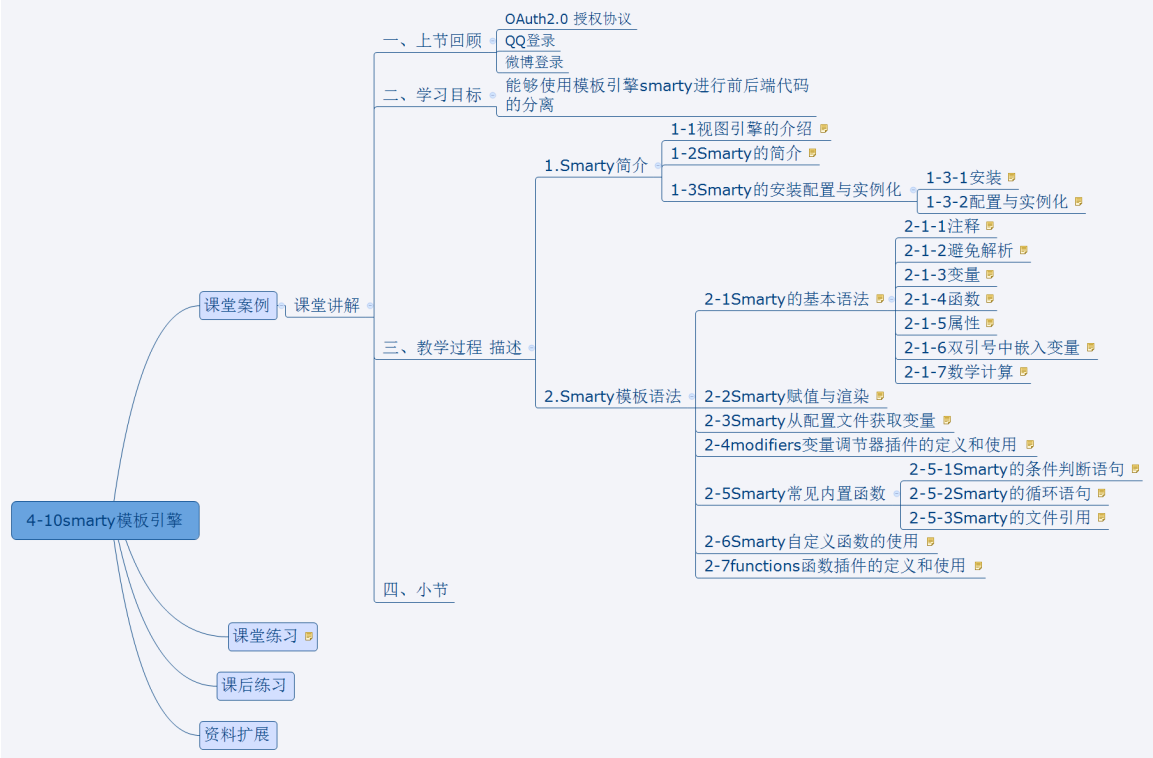
4-10smarty模板引擎



1. 课堂案例

课堂讲解

一、上节回顾

OAuth2.0 授权协议

QQ登录

微博登录

二、学习目标

能够使用模板引擎smarty进行前后端代码的分离

三、教学过程 描述

1.Smarty简介

1-1视图引擎的介绍

1. 什么是视图引擎

MVC中的V 就是视图层

2. 什么是好的视图引擎

1》基于该引擎开发出的模板要更贴近标准的HTML等

2》语法要简单易懂

3》良好的缓存机制

4》扩展性良好

5》网络资源多

3. PHP常见的模板引擎

1》Smarty

Smarty算是一种很老的PHP模板引擎了，它曾是我使用这门语言模板的最初选择。虽然它的更新已经不算频繁了，并且缺少新一代模板引擎所具有的部分特性，但是它仍然值得一看。

2》Twig

Twig是来自于Symfony的模板引擎，它非常易于安装和使用。它的操作有点像Mustache和liquid。

3》Haml

移植了同名的Ruby模板语言。注意，HAML使用的缩进模式（例如像Python）可能在最初会给你带来一定的困扰（而一旦你熟悉这种模式之后便会上瘾）。

4》Liquid

生成Shopify（以及原始的Ruby），Liquid是在限制用户权限的同时又可使其自定义页面服务风格的完美语言。此外，这个语言是跨平台的，并且相同的模板可在PHP和Ruby中交替使用。

5》Mustache

作为多种语言的模板，Mustache可以兼容所有能够想到语言的模板（例如，甚至包括bash）。

6》Plates

Plate受到Twig启发，重载了PHP的原生特性。如果你不想使用需要编译的模板语言，它可以为你大开方便之门。

1-2Smarty的简介

什么是Smarty？

Smarty是PHP的模板引擎，便于将应用程序逻辑与HTML / CSS进行分离。这意味着PHP代码是应用程序逻辑，并与演示文稿分离。

Smarty基于以下原因受到推动：

1. 前后端代码的分离 PHP后端，Smarty模板前端 它只做为PHP的补充

2. 快速开发，易于维护

3. 语法容易理解，不需要PHP知识

4. 免费，开源

关于PHP中的模板，基本上有两个思路

第一阵营宣称“PHP是模板引擎”。这种方法只需将PHP代码与HTML进行混合。虽然这种方法从纯脚本执行角度来看很快，但许多人会认为，PHP语法在与演示文稿混合时很混乱，难以维护。PHP适用于编程，但不适用于模板。

第二阵营表示，视图不应该是应用程序代码，而是使用简单的标签来指明应用程序内容的显示位置。这种方法与其他模板引擎（和其他编程语言）通用，Smarty所采用的方法也是如此。这个想法是将模板集中在视图，没有应用程序代码，并尽可能少的开销。

使用SMART模板分离有很多好处，如下：

语法简单，很像HTML

可移植性：由于Smarty模板与语言无关，因此可以使用不同的编译器轻松地编译成其他语言（如javascript），熟悉的语法也可以移植到其他编程语言中。

低学习成本：前端人员不需要需学习大量的PHP语法，只里需学习简单的smart语法

1-3Smarty的安装配置与实例化

1-3-1安装

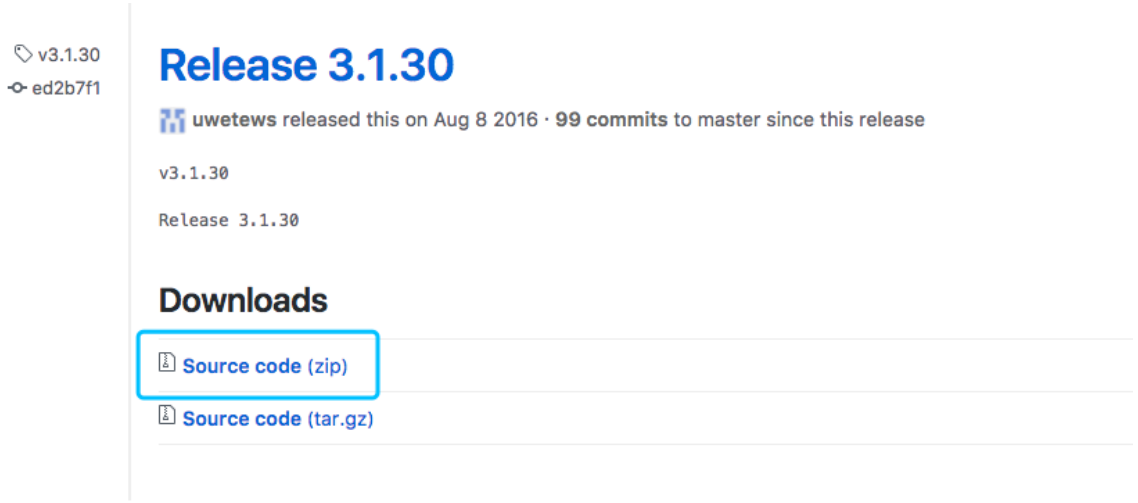
Smarty环境需求

Smarty需要运行PHP5.2或更高版本的服务器。

一、下载安装包:

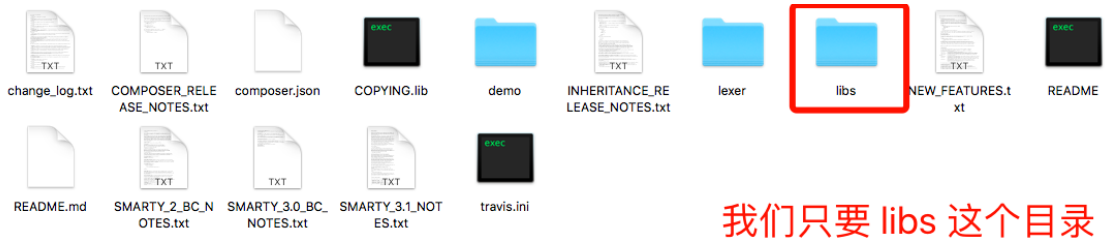
当前最新版本: 3.1.30 下载连接

https://github.com/smarty-php/smarty/releases/tag/v3.1.30



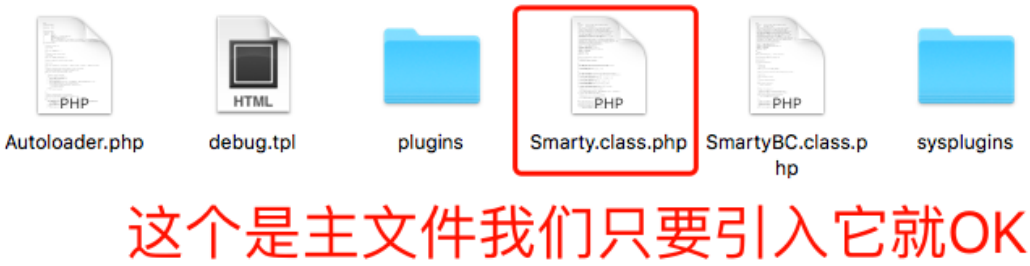
解压

解压后目录结构:



二、 将libs 目录复制到网站根目录(并改名为 Smarty), 比如我们当前环境根目录为 /4.10/app/,

最终目录结构为: /4.10/app/Smarty



1-3-2配置与实例化

Smarty使用一个叫做'SMARTY_DIR'的php常量作为它的系统库目录

原则上，我们只要能够引入 Smarty.class.php 文件成功，就无需配置其它目录，但基于环境因素，我们习惯上都手动配置目录，以免发生错误。同时，我们可能有很多项目都是加载同一个smarty类库，可以手动将不同的项目的smarty配置信息划分到不同目录下，这样更利于维护！

需要配置的目录有4个，默认目录名称如下：

1. tempalates, 【模板文件存放目录】
2. templates_c 【编译目录】此目录必须可写
3. configs 【配置文件目录】
4. cache 【缓存目录】此目录必须可写

比如我们当前的目录结构为：假设我们的网站根目录为：/4.10/app/

在根目录下新建一个index.php文件

内容如下：

```php
<?php
define('SMARTY_DIR',"./Smarty/");            //定义smarty 的根目录常量:  SMARTY_DIR, 必须以 / 结尾
include_once SMARTY_DIR."Smarty.class.php";  //加载smarty 类

$smarty               = new Smarty;          //实例化smarty
$smarty->template_dir = "./templates";       //配置模板目录
$smarty->compile_dir  = "./templates_c";     //配置编译目录
$smarty->config_dir   = "./configs";         //配置, 配置文件目录
$smarty->cache_dir    = "./cache";           //配置缓存目录
```

到此配置成功，但这样配置会有一个问题，每一个php文件需要使用smarty都要进行一次配置，显然不够灵活

因此我们这里介绍另外一种更灵活的配置方式，以OOP继承的方式配置

方法如下：如在根目录下，新建一个 smarty_common.php 的文件，文件内容如下：

```php
<?php
define('SMARTY_DIR',"./Smarty/");            //定义smarty 的根目录常量:  SMARTY_DIR, 必须以 / 结尾
include_once SMARTY_DIR."Smarty.class.php";  //加载smarty 类

class smarty_common extends Smarty{
    public function __construct(){
        parent::__construct();
        $this->setTemplateDir("./templates");    //配置模板目录
        $this->setCompileDir("./templates_c");   //配置编译目录
        $this->setConfigDir("./configs");        //配置, 配置文件目录
        $this->setCacheDir("./cache");           //配置缓存目录

        //$this->caching = Smarty::CACHING_LIFETIME_CURRENT; //开启缓存, 开发阶段不建议开启
        $this->assign('appname','留言板');
    }
}
```

这样个性后，index.php修改为如下，从如下代码可能看出，使用更简单，而且今后修改目录只要改smarty_common.php文件即可

```php
<?php
/*date_default_timezone_set("PRC");
define('SMARTY_DIR',"./Smarty/");            //定义smarty 的根目录常量:  SMARTY_DIR, 必须以 / 结尾
include_once SMARTY_DIR."Smarty.class.php";  //加载smarty 类

$smarty               = new Smarty;          //实例化smarty
$smarty->template_dir = "./templates";       //配置模板目录
$smarty->compile_dir  = "./templates_c";     //配置编译目录
$smarty->config_dir   = "./configs";         //配置, 配置文件目录
$smarty->cache_dir    = "./cache";           //配置缓存目录

$smarty->assign('name','echo');
$smarty->display('index.tpl');*/

include_once "./smarty_common.php";
$smarty = new smarty_common;

$smarty->assign('name','echo');
$smarty->display('index.tpl');
```

2.Smarty模板语法

2-1Smarty的基本语法

Smarty注释语句

Smarty变量

Smarty函数

Smarty属性

Smarty双引号中嵌入变量

Smarty数学计算

Smarty避免解析

2-1-1注释

{* 我是一个Smarty的注释, 显示输出时我不会存在 *}

<html>

<head>

<title>{$title}</title>

</head>

<body>

```
{* 另一个单行的注释例子 *}
<!-- HTML 注释会发送到浏览器 -->

{*
  Smarty的多行
  注释
  不会发送到浏览器
*}

{************************************************************
多行注释的说明栏
 @ author:         bg@example.com
 @ maintainer:     support@example.com
 @ para:           var that sets block style
 @ css:            the style output
************************************************************}

{* 头部文件包括LOGO和其他东西 *}
{include file='header.tpl'}



{* 开发说明: $includeFile是通过foo.php赋值的 *}
<!-- 显示 main content 块 -->
{include file=$includeFile}

{* 这里的 <select> 块是多余的 *}
{*
<select name="company">
  {html_options options=$vals selected=$selected_id}
</select>
*}

<!-- 变量被注释了 -->
{* $affiliate|upper *}

{* 注释不能嵌套 *}
{*
<select name="company">
```

```
  {* <option value="0">-- none -- </option> *}
  {html_options options=$vals selected=$selected_id}
</select>
*}


</body>
</html>
```

2-1-2避免解析

避免Smarty解析

有时候部分模板中的代码是不需要或者不希望被Smarty解析的，
比较典型的例子是嵌入在页面HTML中的Javascript或CSS代码。
问题通常发生在这些语言会经常使用{ 和 }，但{ 和 }也恰好是Smarty的定界符。
Note

避免被解析的一个良好方式，是分离你的Javascript/CSS代码到单独的文件中，
然后在HTML中引入它们。而且这样也有利于浏览器进行缓存。
当你需要嵌入Smarty的变量或者函数到Javascript/CSS中，请参考下面的方式：
Smarty模板中，当{ 和 }定界符两边都是空格的时候，将会被自动忽略解析。
此特性可以通过设置Smarty的成员变量 $auto_literal为false来关闭。
Example 3.8. 使用自动忽略解析的特性

```
<script>
  // 下面的定界符两边都是空格，所以可以被自动忽略解析
  function foobar {
   alert('foobar!');
  }
  // 下面需要手动忽略解析
  {literal}
        function bazzy {alert('foobar!');}
  {/literal}
</script>
```

{literal}..{/literal}可以让块中间的内容忽略Smarty的解析。 在需要使用定界符的时候，可以通过{ldelim}，{rdelim}标签，或者 {$smarty.ldelim},{$smarty.rdelim}的变量来使用。
Smarty的默认定界符{ 和 }可以整齐地界定一般内容的显示。
然而你可以通过修改Smarty的$left_delimiter 和 $right_delimiter 的变量值，设置更适合的定界符。

Note

修改定界符会影响到全部模板代码和解析，请确保在修改前已清除了全部的缓存和编译文件。

Example 3.9. 改变定界符的例子

```
<?php
//修改定界符
$smarty->left_delimiter = '<!--{';
$smarty->right_delimiter = '}-->';

$smarty->assign('foo', 'bar');
$smarty->assign('name', 'Albert');
$smarty->display('example.tpl');

?>
```

模板：

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

2-1-3变量

模板变量以美元符号$开头，由字母、数组和下划线组成

模板变量分为两种，一种是普通的变量，即通过assign();函数指定，一种是配置文件中的变量，两种变量引用 方式不一样，

普通变量

```
{$foo}       <-- 显示简单的变量 (非数组/对象)
{$foo[4]}    <-- 在0开始索引的数组中显示第五个元素
{$foo.bar}   <-- 显示"bar"下标指向的数组值，等同于PHP的$foo['bar']
{$foo.$bar}  <-- 显示以变量$bar值作为下标指向的数组值，等同于PHP的$foo[$bar]
{$foo->bar}  <-- 显示对象属性 "bar"
{$foo->bar()} <-- 显示对象成员方法"bar"的返回
```

{#foo#}　　<-- 显示变量配置文件内的变量"foo"

{$smarty.config.foo} <-- 等同于{#foo#}

{$foo[bar]}　<-- 仅在循环的语法内可用，见{section}

{assign var=foo value='baa'}{$foo} <--　显示"baa", 见{assign}

更多合成变量的方式：

{$foo.bar.baz}

{$foo.$bar.$baz}

{$foo[4].baz}

{$foo[4].$baz}

{$foo.bar.baz[4]}

{$foo->bar($baz,2,$bar)} <-- 传递参数

{"foo"}　　<-- 静态值

{* 显示服务器的环境变量"SERVER_NAME" ($_SERVER['SERVER_NAME'])*}

{$smarty.server.SERVER_NAME}

数学运算和嵌入标签：

{$x+$y}　　// 显示x加y的和

{assign var=foo value=$x+$y}　　// 和用于赋值

{$foo[$x+3]}　　// 作为下标使用

{$foo={counter}+3}　　// 标签内的标签

{$foo="this is message {counter}"}　// 在双引号内的标签

定义数组：

{assign var=foo value=[1,2,3]}

{assign var=foo value=['y'=>'yellow','b'=>'blue']}

{assign var=foo value=[1,[9,8],3]}　// 可嵌套

缩写方式：

{$foo=$bar+2}

{$foo = strlen($bar)}　　// 函数赋值

{$foo = myfunct(($x+$y)*3)}　　// 函数参数

{$foo.bar=1}　　// 赋值给特定的数组元素

```
{$foo.bar.baz=1}
{$foo[]=1}                    // 附加到数组
```

Smarty 点号语法:

```
{$foo.a.b.c}       =>  $foo['a']['b']['c']
{$foo.a.$b.c}      =>  $foo['a'][$b]['c']       // 变量下标
{$foo.a.{$b+4}.c}  =>  $foo['a'][$b+4]['c']     // 表达式下标
{$foo.a.{$b.c}}    =>  $foo['a'][$b['c']]       // 嵌套下标
```

类似PHP的语法，另一种点号的语法:

```
{$foo[1]}           // 一般的
{$foo['bar']}
{$foo['bar'][1]}
{$foo[$x+$x]}       // 下标可以是各种表达式
{$foo[$bar[1]]}     // 嵌套下标
{$foo[section_name]}  // smarty {section} 存取, 非数组存取!
```

变量构造变量:

```
$foo                // 一般的变量
$foo_{$bar}         // 变量名包含了其他变量
$foo_{$x+$y}        // 变量名包含了表达式
$foo_{$bar}_buh_{$blar}  // 更复杂的
{$foo_{$x}}         // 如$x = 1，那么将显示$foo_1的值
```

对象链:

```
{$object->method1($x)->method2($y)}
```

PHP函数直接使用:

```
{time()}
```

Smarty保留变量

页面请求变量 如$_GET, $_POST, $_COOKIE, $_SERVER, $_ENV 和 $_SESSION 可以通过下面的方式来使用:

{$smarty.get.page}

{$smarty.post.name}

{$smarty.session.name}

2-1-4函数

每个Smarty的标签都可以是显示一个变量或者调用某种类型的函数。

调用和显示的方式是在定界符内包含了函数，和其属性，如:{funcname attr1="val1" attr2="val2"}.

函数语法

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

{include file="header.tpl"}
{insert file="banner_ads.tpl" title="My Site"}

{if $logged_in}
   Welcome, <span style="color:{#fontColor#}">{$name}!</span>
{else}
   hi, {$name}
{/if}

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

包括内置函数 和自定义函数 都是用同样的语法调用。

内置函数是工作在Smarty 内部的函数, 类似 {if}, {section}和 {strip}等等。

它们不需要进行修改或者改变。

自定义函数是通过插件定义的 额外的函数。

你可以任意修改自定义函数，或者创建一个新的函数。

{html_options}就是一个自定义函数的例子。

2-1-5属性

大多数函数都会使用属性来定义或者修改它们的行为。
Smarty函数中的属性比较像HTML语法中的属性。 静态值不需要引号引起来，但必须是纯字符串。
带或不带修饰器的变量都可以使用，而且也不需要引号，
甚至可以使用PHP函数的结果，插件结果和复杂的表达式。
一些属性要求布尔值(TRUE 或 FALSE)。 它们可以直接写成true 和 false。
如果属性没有被赋值，那么它会将true作为默认值。

函数属性语法

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

{include file="header.tpl" nocache} // 等同于 nocache=true

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

{include file=$includeFile}

{include file=#includeFile# title="My Title"}

{assign var=foo value={counter}} // 插件结果

{assign var=foo value=substr($bar,2,5)} // PHP函数结果

{assign var=foo value=$bar|strlen} // 使用修饰器

{assign var=foo value=$buh+$bar|strlen} // 复杂的表达式

{html_select_date display_days=true}

{mailto address="smarty@example.com"}

<select name="company_id">
  {html_options options=$companies selected=$company_id}
</select>
```

2-1-6双引号中嵌入变量

双引号中嵌入变量

Smarty可以识别出在双引号中嵌套的变量值，这些变量名称必须只包括　　字母、数字和下划线。参见命名规则。

另外，带有其他字符的，如点号(.)或者 $object->reference形式的变量，必须用`单引号`括起来。

Smarty3中允许在双引号中嵌入Smarty的标签并运行。

如果你需要在双引号的变量上使用修饰器、插件或者PHP函数等，这是非常有用的。

```
{func var="test $foo test"}            // 识别变量 $foo
{func var="test $foo_bar test"}        // 识别变量 $foo_bar
{func var="test `$foo[0]` test"}       // 识别变量 $foo[0]
{func var="test `$foo[bar]` test"}     // 识别变量 $foo[bar]
{func var="test $foo.bar test"}        // 识别变量 $foo (不是 $foo.bar)
{func var="test `$foo.bar` test"}      // 识别变量 $foo.bar
{func var="test `$foo.bar` test"|escape} // 引号外的修饰器!
{func var="test {$foo|escape} test"}   // 引号内的修饰器!
{func var="test {time()} test"}        // PHP函数结果
{func var="test {counter} test"}       // 插件的结果
{func var="variable foo is {if !$foo}not {/if} defined"} // Smarty区块函数
```

```
{* 将使用$tpl_name的值 *}
{include file="subdir/$tpl_name.tpl"}

{* 不使用$tpl_name的值 *}
{include file='subdir/$tpl_name.tpl'} // 必须用双引号才能用变量值!

{* 当变量包含了点号“.”，必须用单引号括起来 *}
{cycle values="one,two,`$smarty.config.myval`"}
```

{* 当变量包含了点号“.”，必须用单引号括起来 *}
{include file="`$module.contact`.tpl"}

{* 点号后面跟着变量 *}
{include file="`$module.$view`.tpl"}

2-1-7数学计算

变量值内可以直接进行数学计算

{$foo+1}

{$foo*$bar}

{* 更复杂的例子 *}

{$foo->bar-$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`"}

2-2Smarty赋值与渲染

Smarty 通过 $smarty->assign() 方法进行赋值，

Smarty 通过 $smarty->display() 方法进行渲染，

Smarty 通过 $smarty->fetch() 方法跟display() 方法类似，只是fetch()只是返回渲染后的代码，并不会展示到浏览哭，一般于用渲染再处理数据的情况

2-3Smarty从配置文件获取变量

从配置文件中读取变量，一般用于全局变量的定义

从配置文件获取的变量，可以通过 并号引用起来访问如#hash_marks#，
或者通过Smarty变量$smarty.config来访问。
后者在使用其他属性或者是访问别的变量值时比较有用，如$smarty.config.$foo。

比如，在configs 目录下有个配置文件：foo.conf 内容如下：

```
pageTitle = "This is mine"
bodyBgColor = '#eeeeee'
tableBorderSize = 3
tableBgColor = "#bbbbbb"
rowBgColor = "#cccccc"
```

模板中如何使用foo.conf 中定义的变量呢？

```
{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>
```

2-4modifiers变量调节器插件的定义和使用

smarty 系统自带了很多变量调节器

变量修饰器可以用于变量， 自定义函数或者字符串。 使用修饰器，需要在变量的后面加上
|(竖线)并且跟着修饰器名称。 修饰器可能还会有附加的参数以便达到效果。
参数会跟着修饰器名称，用:(冒号)分开。 同时，默认全部PHP函数都可以作为修饰器来使用
(不止下面的)，而且修饰器可以被 联合使用。

备注：

| 号前的返回值是函数的第1个变量，:第2个参数:第3个参数: 第n个参数

常用的修饰器

1.capitalize（字符串首字母转为大写）

capitalize

使变量内容里的每个单词的第一个字母大写。 与PHP函数的 `ucwords()` 相似。

参数顺序	类型	必选参数	默认值	说明
1	boolean	No	`FALSE`	带数字的单词是否也头字母大写。
2	boolean	No	`FALSE`	设置单词内其他字母是否小写，如"aAa" 变成 "Aaa"。

Example 5.2. capitalize

```
<?php

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

?>
```

模板是:

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

输出:

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

2.cat（连接字符串）

cat

连接多个变量。

参数顺序	类型	必选参数	默认值	说明
1	string	No	空	需要连接的变量

Example 5.3. cat

```
<?php

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

?>
```

模板是:

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

输出:

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

3.count_characters（计算变量字符数）

count_characters

计算变量内容里有多少个字符。

参数顺序	类型	必选参数	默认值	说明
1	boolean	No	`FALSE`	计算总数时是否包括空格字符。

Example 5.4. count_characters

```
<?php

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

?>
```

模板:

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

输出:

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

4.count_paragraphs（计算变量段落数）

count_paragraphs

计算变量内容有多少个段落。

Example 5.5. count_paragraphs

```
<?php

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

?>
```

模板是:

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

输出:

```
War Dims Hope for Peace. Child's Death Ruins Couple's Holiday.

Man is Fatally Slain. Death Causes Loneliness, Feeling of Isolation.
2
```

5.count_sentences（计算变量句字数量）

count_sentences

计算变量内容有多少个句子。 每个句子必须以点号、问号或者感叹号结尾。(.?!)

Example 5.6. count_sentences

```
<?php

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

?>
```

模板是:

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

输出:

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

6.count_words（计算变量单词数量）

count_words

用于计算变量内容有多少个单词。

Example 5.7. count_words

```
<?php

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

?>
```

模板是：

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

输出：

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

7.date_format（日期格式化）

date_format

将日期和时间格式化成 `strftime()` 的格式。 时间可以是unix的 时间戳，DateTime 对象，mysql时间戳，或者月日年格式的字符串，与 PHP函数 `strtotime()` 类似。 模板设计者可以对 `date_format` 的格式有完全的控制。 如果传递到 `date_format` 的时间为空， 而第二个参数传递了值，那么第二个参数将作为需要格式化的时间。

参数顺序	类型	必选参数	默认值	说明
1	string	No	%b %e, %Y	输出时间的格式定义
2	string	No	n/a	如果输入为空，则作为默认时间。

Example 5.8. date_format

```
<?php

$config['date'] = '%I:%M %p';
$config['time'] = '%H:%M:%S';
$smarty->assign('config', $config);
$smarty->assign('yesterday', strtotime('-1 day'));

?>
```

模板使用了 `$smarty.now` 取得当前的时间：

```
{$smarty.now|date_format}
{$smarty.now|date_format:"%D"}
{$smarty.now|date_format:$config.date}
{$yesterday|date_format}
{$yesterday|date_format:"%A, %B %e, %Y"}
{$yesterday|date_format:$config.time}
```

输出是：

```
Jan 1, 2022
01/01/22
02:33 pm
Dec 31, 2021
Monday, December 1, 2021
14:33:00
```

date_format支持格式：

- %a - 当前区域星期几的简写
- %A - 当前区域星期几的全称
- %b - 当前区域月份的简写
- %B - 当前区域月份的全称
- %c - 当前区域首选的日期时间表达
- %C - 世纪值（年份除以 100 后取整，范围从 00 到 99）
- %d - 月份中的第几天，十进制数字（范围从 01 到 31）
- %D - 和 %m/%d/%y 一样
- %e - 月份中的第几天，十进制数字，一位的数字前会加上一个空格（范围从 ' 1' 到 '31'）
- %g - 和 %G 一样，但是没有世纪
- %G - 4 位数的年份
- %h - 和 %b 一样
- %H - 24 小时制的十进制小时数（范围从 00 到 23）
- %I - 12 小时制的十进制小时数（范围从 00 到 12）
- %j - 年份中的第几天，十进制数（范围从 001 到 366）
- %k - 小时，24 小时格式，没有前导零
- %l - 小时，12 小时格式，没有前导零
- %m - 十进制月份（范围从 01 到 12）
- %M - 十进制分钟数
- %n - 换行符
- %p - 根据给定的时间值为 `am' 或 `pm'，或者当前区域设置中的相应字符串
- %r - 用 a.m. 和 p.m. 符号的时间
- %R - 24 小时符号的时间
- %S - 十进制秒数
- %t - 制表符

8

- %T - 当前时间，和 %H:%M:%S 一样
- %u - 星期几的十进制数表达 [1,7]，1 表示星期一
- %U - 本年的第几周，从第一周的第一个星期天作为第一天开始
- %V - 本年第几周的 ISO 8601:1988 格式，范围从 01 到 53，第 1 周是本年第一个至少还有 4 天的星期，星期一作为每周的第一天。（用 %G 或者 %g 作为指定时间戳相应周数的年份组成。）
- %w - 星期中的第几天，星期天为 0
- %W - 本年的第几周数，从第一周的第一个星期一作为第一天开始
- %x - 当前区域首选的时间表示法，不包括时间
- %X - 当前区域首选的时间表示法，不包括日期
- %y - 没有世纪数的十进制年份（范围从 00 到 99）
- %Y - 包括世纪数的十进制年份
- %Z - 时区名或缩写
- %% - 文字上的 `%' 字符

8.default （变量默认值)

default

为变量设置默认值。 当变量是unset或者empty的字符串时，默认值将显示。 必须要有一个参数。

参数顺序	类型	必选参数	默认值	说明
1	string	No	*控制*	当变量为空时输出的值

Example 5.9. default

```
<?php

$smarty->assign('articleTitle', 'Dealers Will Hear Car Talk at Noon.');
$smarty->assign('email', '');

?>
```

模板:

```
{$articleTitle|default:'no title'}
{$myTitle|default:'no title'}
{$email|default:'No email address available'}
```

输出:

```
Dealers Will Hear Car Talk at Noon.
no title
No email address available
```

9.escape（变量转码）

escape

`escape` 可用于将变量编码或转换成 `html` , `url` , `单引号` , `十六进制` , `十六进制实体` , `javascript` 和 `电邮地址` 。默认是: `html` 。

参数顺序	类型	必选参数	允许取值	默认值	说明
1	string	No	`html` , `htmlall` , `url` , `urlpathinfo` , `单引号` , `十六进制` , `十六进制实体` , `javascript` , `电邮地址`	`html`	这是escape转换后的格式
2	string	No	`ISO-8859-1` , `UTF-8` ,和其他 `htmlentities()` 支持的字符集	`UTF-8`	传递给htmlentities()的字符集类型
3	boolean	No	`FALSE`	`TRUE`	两次转换实体, & 到 & (仅在 `html` 和 `htmlall` 使用)

10.from_charset

11.indent（变量缩进）

indent

缩进每一行的字符串，默认是缩进4个空格。 可选的参数可以设置缩进的空格数量。 可选的第二个参数设置缩进使用的字符，如用 `"\t"` 来代替空格缩进。

参数顺序	类型	必选参数	默认值	说明
1	integer	No	4	设置缩进多少空格
2	string	No	一个空格	设置用什么字符来进行缩进

12.lower（变量转小字字母）

lower

将变量值转成小写字母。 等同于PHP的 `strtolower()` 函数。

Example 5.13. lower

```
<?php

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

?>
```

模板:

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

输出:

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

13.nl2br （\n 转成
）

nl2br

将变量值中的 `"\n"` 回车全部转换成HTML的 `<br />` 。 等同于PHP的 `nl2br()` 函数。

Example 5.14. nl2br

```
<?php

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

模板:

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

输出:

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

14.regex_replace (正则替换)

regex_replace

用正则表达式搜索和替换变量值。 使用PHP的 `preg_replace()` 函数进行。

温馨提示:

虽然Smarty提供了较方便的正则表达式修饰器， 但通常更好的方式是通过自定义函数或自定义修饰器在PHP端进行正则匹配替换。 正则表达式是程序应用代码，不是显示的逻辑代码。

Parameters

参数顺序	类型	必选参数	默认值	说明
1	string	Yes	*n/a*	正则表达式
2	string	Yes	*n/a*	替换的字符

Example 5.15. regex_replace

```
<?php

$smarty->assign('articleTitle', "Infertility unlikely to\nbe passed on, experts say.");

?>
```

模板:

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

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

输出:

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

15.replace （字符串替换）

replace

对变量进行简单的搜索和替换。 等同于PHP函数的 `str_replace()` 。

参数顺序	类型	必选参数	默认值	说明
1	string	Yes	*n/a*	需要搜索并替换的字符
2	string	Yes	*n/a*	替换用的字符

Example 5.16. replace

```
<?php

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

?>
```

模板:

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

输出:

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

16.spacify （变量中插入字符默认插入空格）

17.string_format（字符串格式化）

string_format

格式化字符串，如浮点数等。 使用 `sprintf()` 的PHP函数来进行。

参数顺序	类型	必选参数	默认值	说明
1	string	Yes	*n/a*	指定哪种格式 (sprintf)

Example 5.18. string_format

```
<?php

$smarty->assign('number', 23.5787446);

?>
```

模板:

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

输出:

```
23.5787446
23.58
24
```

18.strip （转换**空格**）

strip

转换连续空格，回车和tab到单个空格或是指定字符串。

说明

如果你希望转换模板文字内的空格，使用内置的 `{strip}` 函数。

Example 5.19. strip

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

模板：

```
{$articleTitle}
{$articleTitle|strip}
{$articleTitle|strip:' '}
```

输出：

```
Grandmother of
eight makes        hole in one.
Grandmother of eight makes hole in one.
Grandmother of eight makes hole in one.
```

19.strip_tags（去除HTML标记，保留标记内的字符）

strip_tags

去除标记等任何包含在 `<` 和 `>中间的字符。` .

参数顺序	类型	必选参数	默认值	说明
1	bool	No	`TRUE`	设置是否将标签替换成' '或者 ''

Example 5.20. 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>."
               );

?>
```

模板:

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

输出:

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

20.to_charset

21.truncate （截取字符串到指定长度）

truncate

截取字符串到指定长度，默认长度是80. 第二个参数可选，指定了截取后代替显示的字符。 截取后的字符长度是截取规定的长度加上第二个参数的字符长度。 默认 `truncate` 会尝试按单词进行截取。 如果你希望按字符截取（单词可能会被截断），需要设置第三个参数 `TRUE` 。

参数顺序	类型	必选参数	默认值	说明
1	integer	No	80	截取的长度
2	string	No	...	截取后替代显示的字符，该字符长度会被计算到截取长度内。
3	boolean	No	`FALSE`	是否按单词截取 `FALSE` ，或是按字符截取 `TRUE` 。
4	boolean	No	`FALSE`	当字符截取的长度刚好等于字符本身长度时，是否截取。 `FALSE` 也会截取。 `TRUE` 是不会截取。 注意如果设置为 `TRUE` ，单词的边界会被忽略。

Example 5.21. truncate

```
<?php
$smarty->assign('articleTitle', 'Two Sisters Reunite after Eighteen Years at Checkout
Counter.');
?>
```

模板:

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

输出:

```
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...
Two Sisters Re..ckout Counter.
```

22.unescape （解码 escape 反操作）

unescape

`unescape` 可以解码 `entity` , `html` 和 `htmlall` 等的编码。 它与escape 修饰器的效果刚好相反。

参数顺序	类型	必选参数	允许取值	默认值	说明
1	string	No	`html` , `htmlall` , `entity` ,	`html`	解码的类型
2	string	No	`ISO-8859-1` , `UTF-8` ,或者任何 `htmlentities()` 可以支持的字符集。	`UTF-8`	传递给 html_entity_decode() 、 htmlspecialchars_decode() 或 mb_convert_encoding()的字符集

Example 5.22. escape

```
<?php

$smarty->assign('articleTitle',
                "Germans use &quot;&Uuml;mlauts&quot; and pay in &euro;uro"
                );

?>
```

`unescape` 例子

```
{$articleTitle}
Germans use &quot;&Uuml;mlauts&quot; and pay in &euro;uro

{$articleTitle|unescape:"html"}
Germans use "&Uuml;mlauts" and pay in &euro;uro

{$articleTitle|unescape:"htmlall"}
Germans use "Ümlauts" and pay in €uro
```

23.upper （字符串转大写）

upper

将变量值转成大写字母。 等同于PHP的 `strtoupper()` 函数。

Example 5.23. upper

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

模板:

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

输出:

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

24.wordwrap （强制换行）

wordwrap

限制一行字符的长度（自动换行），默认是80个字符长度。 可选的第二个参数，可自定义换行字符，默认换行字符是 `"\n"` 。默认情况下，是根据单词来换行的，也就是按英文语法的 自动换行 。 如果你希望按照字符来换行（边界的单词将拆开），那么可以设置 可选的第三个参数为 `TRUE` ，效果与PHP函数 `wordwrap()` 一样。

参数顺序	类型	必选参数	默认值	说明
1	integer	No	80	限定一行的长度。
2	string	No	\n	换行符号
3	boolean	No	`FALSE`	设置按单词换行(`FALSE`)，或者按字符换行(`TRUE`)。

Example 5.24. wordwrap

```
<?php

$smarty->assign('articleTitle',
                "Blind woman gets new kidney from dad she hasn't seen in years."
               );

?>
```

模板:

```
{$articleTitle}

{$articleTitle|wordwrap:30}

{$articleTitle|wordwrap:20}

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

{$articleTitle|wordwrap:26:"\n":true}
```

输出:

```
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 kidn
ey from dad she hasn't se
en in years.
```

修饰器例子

```
{* 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="smarty@example.com"}

{* using  php's str_repeat *}
{"="|str_repeat:80}

{* php's count *}
{$myArray|@count}

{* this will uppercase and truncate the whole array *}
<select name="name_id">
{html_options output=$my_array|upper|truncate:20}
</select>
```

2-5Smarty常见内置函数

2-5-1Smarty的条件判断语句

{if},{elseif},{else}

下面是可用的运算符列表，使用中都会放到元素的中间并且用空格分隔。 注意列表中[方括号]的是可选的，而且还会列出对应PHP的表达式。

运算符	别名	语法示例	含义	对应PHP语法
==	eq	$a eq $b	等于	==
!=	ne, neq	$a neq $b	不等于	!=
>	gt	$a gt $b	大于	>
<	lt	$a lt $b	小于	<
>=	gte, ge	$a ge $b	大于等于	>=
<=	lte, le	$a le $b	小于等于	<=
===		$a === 0	绝对等于	===
!	not	not $a	非 (一元运算)	!
%	mod	$a mod $b	取模	%
is [not] div by		$a is not div by 4	取模为0	$a % $b == 0
is [not] even		$a is not even	[非] 取模为0 (一元运算)	$a % 2 == 0
is [not] even by		$a is not even by $b	水平分组 [非] 平均	($a / $b) % 2 == 0
is [not] odd		$a is not odd	[非] 奇数 (一元运算)	$a % 2 != 0
is [not] odd by		$a is not odd by $b	[非] 奇数分组	($a / $b) % 2 != 0

实例:

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

{* an example with "or" logic *}
{if $name eq 'Fred' or $name eq 'Wilma'}
  ...
{/if}

{* same as above *}
{if $name == 'Fred' || $name == 'Wilma'}
  ...
{/if}



{* parenthesis are allowed *}
{if ( $amount < 0 or $amount > 1000 ) and $volume >= #minVolAmt#}
  ...
{/if}



{* you can also embed php function calls *}
{if count($var) gt 0}
  ...
{/if}

{* check for array. *}
{if is_array($foo) }
  .....
{/if}

{* check for not null. *}
{if isset($foo) }
  .....
{/if}
```

```
{* test if values are even or odd *}
{if $var is even}
  ...
{/if}
{if $var is odd}
  ...
{/if}
{if $var is not odd}
  ...
{/if}



{* test if var is divisible by 4 *}
{if $var is div by 4}
  ...
{/if}



{*
 test if var is even, grouped by two. i.e.,
 0=even, 1=even, 2=odd, 3=odd, 4=even, 5=even, etc.
*}
{if $var is even by 2}
  ...
{/if}

{* 0=even, 1=even, 2=even, 3=odd, 4=odd, 5=odd, etc. *}
{if $var is even by 3}
  ...
{/if}

{if isset($name) && $name == 'Blog'}
    {* do something *}
{elseif $name == $foo}
   {* do something *}
{/if}

{if is_array($foo) && count($foo) > 0}
   {* do a foreach loop *}
{/if}
```

2-5-2Smarty的循环语句

1. {for},{forelse}

   {for}{forelse}用于创建一个简单的循环。下面的几种方式都是支持的:

   {for $var=$start to $end}步长1的简单循环。

   {for $var=$start to $end step $step}指定步长的循环。

   {forelse}在循环不能遍历的时候执行。

2. {foreach},{foreachelse}

   @index

   @iteration

   @first

   @last

   @show

   @total

   {break}

   {continue}

   {foreach}用于循环数组。

{foreach}的语法比{section}循环要更简单和清晰，并且可以使用非数字下标的数组。

   {foreach $arrayvar as $itemvar}

   {foreach $arrayvar as $keyvar=>$itemvar}

3. {section},{sectionelse}

{section}可以循环遍历 连续数字索引的数组, 区别于{foreach} 可以循环任意关联数组.

每个{section}标签都必须有一个匹配的{/section}关闭标签。

温馨提示:

{foreach} 可以做到任何{section}做到的功能,

而且更简单和有更清晰的语法。一般更推荐使用{foreach}语法.

4, {while}

   {while $foo > 0}

      {$foo--}

   {/while}

2-5-3Smarty的文件引用

{include}

属性:

参数名称	类型	必选参数	默认值	说明
file	string	Yes	*n/a*	包含载入的文件名
assign	string	No	*n/a*	将包含的文件内容赋值给变量
cache_lifetime	integer	No	*n/a*	单独开启被包含模板的缓存时间
compile_id	string/integer	No	*n/a*	单独设置被包含模板的编译ID
cache_id	string/integer	No	*n/a*	单独设置被包含模板的缓存ID
scope	string	No	*n/a*	定义被包含模板的赋值变量作用范围: 'parent','root' 或 'global'
[var ...]	[var type]	No	*n/a*	传递到包含模板的变量

可选标记:

名称	说明
nocache	关闭包含模板的缓存
caching	打开包含模板的缓存
inline	设置成true时，在编译时把包含模板的内容也合并到当前模板的编译文件中。

例子

简单 {include} **例子**

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

{* body of template goes here, the $tpl_name variable
   is replaced with a value eg 'contact.tpl'
*}
{include file="$tpl_name.tpl"}

{* using shortform file attribute *}
```

```
{include 'page_footer.tpl'}
</body>
</html>
```

{include} 传递变量

```
{include 'links.tpl' title='Newest links' links=$link_array}
{* body of template goes here *}
{include 'footer.tpl' foo='bar'}
```

包含了下面的 links.tpl 模板

```
<div id="box">
<h3>{$title}{/h3>
<ul>
{foreach from=$links item=l}
.. do stuff  ...
</foreach}
</ul>
</div>
```

{include} 作用范围示例

在包含的模板内赋值的变量，在包含模板内可见。

```
{include 'sub_template.tpl' scope=parent}
...
{* display variables assigned in sub_template *}
{$foo}<br>
{$bar}<br>
...
```

包含了下面的 sub_template.tpl 模板

```
...
{assign var=foo value='something'}
{assign var=bar value='value'}
...
```

{include} 关闭缓存

包含模板将不被缓存

```
{include 'sub_template.tpl' nocache}
```

{include} 单独的缓存时间

下面例子包含模板将单独设置缓存时间500秒。

```
{include 'sub_template.tpl' cache_lifetime=500}
```

{include}开启缓存

下面的例子包含模板的缓存将独立于全局模板缓存设置之外。

```
{include 'sub_template.tpl' caching}
```

{include} 和赋值变量

下面的例子将nav.tpl的内容赋值给了$navbar 变量, 该变量将页面的头部和底部显示。

```
<body>
  {include 'nav.tpl' assign=navbar}
  {include 'header.tpl' title='Smarty is cool'}
    {$navbar}
    {* body of template goes here *}
    {$navbar}
  {include 'footer.tpl'}
</body>
```

2-6Smarty自定义函数的使用

以下是Smarty 原生给我们提供的自定义函数, 可以直接使用

{counter}　计数器函数

{counter}

`{counter}` 用于显示一个计数器。 `{counter}` 可以记住foreach循环的次数。 你可以设置计数器的数值、步长、计算方向，和是否每次显示数值。 你可以同时使用多个不同名的计数器。 如果没有指定计数器名称，那么"default"将是默认的名称。

如果你指定了 *assign* 属性，那么 `{counter}` 的输出将被赋值给模板变量。

参数名称	类型	必选参数	默认值	说明
name	string	No	*default*	计数器的名称
start	number	No	*1*	开始计数的数值
skip	number	No	*1*	步长，也就是计算间隔
direction	string	No	*up*	计算的方向，(递增/递减)
print	boolean	No	*TRUE*	是否每次都显示计数器的值
assign	string	No	*n/a*	赋值的变量名

Example 8.1. {counter}

```
{* initialize the count *}
{counter start=0 skip=2}<br />
{counter}<br />
{counter}<br />
{counter}<br />
```

输出:

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

{cycle} 交替循环 通常 用于表格中 隔行换色

{cycle}

`{cycle}` 用于交替循环一系列值。 例如它可以轻易做到： 在表格中各行交替显示两种或多种颜色， 或者交替循环数组。

参数名称	类型	必选参数	默认值	说明
name	string	No	*default*	交替循环的名称
values	mixed	Yes	*N/A*	交替遍历的值，可以是用逗号分隔的字符串列表（注意限定符号），也可以是数组。
print	boolean	No	`TRUE`	是否每次都显示该值
advance	boolean	No	`TRUE`	是否递进到下一个值
delimiter	string	No	,	values属性使用的限定符号
assign	string	No	*n/a*	赋值的变量名
reset	boolean	No	`FALSE`	交替循环将重置回到最前面的值，而不会递进。

- 你可以在模板内使用多个不同 `name` 属性的 `{cycle}` 。
- 设置 `print` 为 `FALSE` 可以让当前值不显示。 在你希望可以静默地跳过一些值的时候很有用。
- `advance` 属性用来重复一个值。当设置成 `FALSE` 下次执行 `{cycle}` 将输出同一个值。
- 如果你设置了 `assign` 属性，那么 `{cycle}` 的输出将会被赋值给变量。

Example 8.2. {cycle}

```
{section name=rows loop=$data}
<tr class="{cycle values="odd,even"}">
   <td>{$data[rows]}</td>
</tr>
{/section}
```

输出：

```
<tr class="odd">
   <td>1</td>
</tr>
<tr class="even">
   <td>2</td>
</tr>
<tr class="odd">
   <td>3</td>
</tr>
```

{fetch}　获取内容

{fetch}

`{fetch}` 用于获取文件内容、HTTP或者FTP内容，以便输出。

- 如果提供的文件名是以 `http://` 开头，那么网站的内容将被获取并显示。

  温馨提示:

  该函数不支持直接的HTTP访问，请确认网页的地址是以斜杠结尾。
- 如果文件名是以 `ftp://` 开头，那么文件会被从FTP中下载并显示。
- 本地文件而言，可以使用绝对路径的文件，或者相对于当前执行的PHP脚本的路径文件均可。

  温馨提示:

  如果开启了安全机制，那么从本地获取文件内容， `{fetch}` 函数仅能读取在 `$secure_dir` 安全目录下的文件。 参见安全机制。
- 如果你提供了 `assign` 属性， `{fetch}` 函数的输出将不会显示，而是赋值给模板变量。

参数名称	类型	必选参数	默认值	说明
file	string	Yes	*n/a*	获取的文件名、HTTP或者FTP网址
assign	string	No	*n/a*	用于赋值的变量名

Example 8.5. {fetch} 例子

```
{* 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'}
{* as above but with variables *}
{fetch file="ftp://`$user`:`$password`@`$server`/`$path`"}

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

{html_checkboxes} 创建HTML复选框

{html_checkboxes}

`{html_checkboxes}` 是一个 自定义函数，用于创建HTML的多选框组和提供数据。 请注意默认选中的情况。

参数名称	类型	必选参数	默认值	说明
name	string	No	*checkbox*	多选框的名称
values	array	必选，除非使用options属性	*n/a*	多选框的值数据
output	array	必选，除非使用options属性	*n/a*	多选框的显示数据
selected	string/array	No	*empty*	选中的项（一个或多个）
options	associative array	必须，除非使用values 和 output属性	*n/a*	多选框的值-显示的数组
separator	string	No	*empty*	字符串中分隔每项的字符
assign	string	No	*empty*	将多选框标签赋值到数组，而不是输出
labels	boolean	No	`TRUE`	在输出中增加<label>标签
label_ids	boolean	No	`FALSE`	给<label> 和 <input> 设置ID属性
escape	boolean	No	`TRUE`	将输出中的/转换(值总是会被转换)

- 必须赋值的属性是 `values` 和 `output`，除非使用 `options` 来代替。
- 全部的输出标签都遵循XHTML规则。
- 任何不在上面列表中的键值对属性，都会被输出到<input>标签中作为属性和值。

Example 8.6. {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);

?>
```

模板是:

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

或者PHP代码:

```
<?php

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

?>
```

模板是：

```
{html_checkboxes name='id' options=$cust_checkboxes
   selected=$customer_id separator='<br />'}
```

上面两个例子的输出：

```
<label><input type="checkbox" name="id[]" value="1000" />Joe Schmoe</label><br />
<label><input type="checkbox" name="id[]" value="1001" checked="checked" />Jack
Smith</label>
<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 8.7. 数据库例子(例如 PEAR 或 ADODB):

```
<?php

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

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

?>
```

输出：

```
{html_checkboxes name='contact_type_id' options=$contact_types
        selected=$contact.contact_type_id separator='<br />'}
```

{html_image} 创建html图片

{html_image}

`{html_image}` 用于生成HTML的 `<img>` 标签的 自定义函数。 如果没有提供 *`height`* 和 *`width`* 参数， 此函数会自动从图片文件计算出来。

参数名称	类型	必选参数	默认值	说明
file	string	Yes	*n/a*	图片名称/路径
height	string	No	*真实图片高度*	图片显示高度
width	string	No	*真实图片宽度*	图片显示宽度
basedir	string	no	*网站根目录*	相对路径的起始目录
alt	string	no	""	图片的说明内容
href	string	no	*n/a*	图片上的链接地址
path_prefix	string	no	*n/a*	显示路径的前缀

- *`basedir`* 设置了相对图片路径的起始目录。如果没有提供， 则使用网站根目录 `$_ENV['DOCUMENT_ROOT']` 。 如果开启了安全限制，图片文件必须放置在 *`$secure_dir`* 目录。 参见安全机制来了解更多。
- *`href`* 设置图片上超链接地址。如果链接地址有提供，那么 图片标签的外围会加上 `<a href="超链接地址"><a>` 的标签。
- *`path_prefix`* 可选的，设置显示路径时的前缀字符串。 当你需要给图片设置不同的服务器名时，这会很有用。
- 其他不在上面列表中的键值对参数，会直接在输出的 `<img>` 标签中显示成 名称=值 的属性。

技术说明

`{html_image}` 需要每次都读取硬盘上的图片，并且计算图片长宽。 除非你开启了缓存，否则一般建议是避免使用 `{html_image}` ，用回HTML的IMG标签，会有更高的性能。

Example 8.8. {html_image} 例子

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

输出:

```
<img src="pumpkin.jpg" alt="" width="44" height="68" />
<img src="/path/from/docroot/pumpkin.jpg" alt="" width="44" height="68" />
<img src="../path/relative/to/currdir/pumpkin.jpg" alt="" width="44" height="68" />
```

{html_options} 创建HTML下拉列表

{html_options}

`{html_options}` 是一个自定义函数， 可以使用提供的数据，生成HTML的 `<select><option>` 标签，还可以设置选中项等属性。

参数名称	类型	必选参数	默认值	说明
values	array	Yes, 除非使用 options 属性	*n/a*	下拉框值的数组
output	array	Yes, 除非使用 options 属性	*n/a*	下拉框显示的数组
selected	string/array	No	*empty*	选中的项
options	数组	Yes, 除非使用 values 和 output	*n/a*	键值对的数组，用于下拉框
name	string	No	*empty*	select组的名称

- 必要的属性是 `values` 和 `output` ，除非你使用组合的 `options` 来代替。
- 除非提供了可选属性 `name` ， 才会创建 `<select></select>` 标签， 不然，只会生成 `<option>` 列表。
- 如果设置的值是数组，会当作HTML的 `<optgroup>` ，并且显示该下拉组。 `<optgroup>` 是支持递归的。
- 其他不在上面列表中的键值对参数，会直接在输出的 `<select>` 标签中显示成 名称=值 的属性。 如果可选参数 `name` 没有设置，那么它们将被忽略。
- 全部的输出都符合XHTML的。

Example 8.9. 使用 `options` 属性

```
<?php
$smarty->assign('myOptions', array(
                                1800 => 'Joe Schmoe',
                                9904 => 'Jack Smith',
                                2003 => 'Charlie Brown')
                                );
$smarty->assign('mySelect', 9904);
?>
```

下面模板将生成一个下拉列表。 注意 `name` 提供了值，所以会生成 `<select>` 标签。

```
{html_options name=foo options=$myOptions selected=$mySelect}
```

输出:

```
<select name="foo">
<option value="1800">Joe Schmoe</option>
<option value="9904" selected="selected">Jack Smith</option>
<option value="2003">Charlie Brown</option>
</select>
```

Example 8.10. 分开赋值 `values` 和 `ouptut`

```
<?php
$smarty->assign('cust_ids', array(56,92,13));
$smarty->assign('cust_names', array(
                              'Joe Schmoe',
                              'Jane Johnson',
                              'Charlie Brown'));
$smarty->assign('customer_id', 92);
?>
```

上面的两个数组，将会如下输出HTML (注意这里有使用了PHP的 `count()` 函数作为修饰器来计算size值).

```
<select name="customer_id" size="{$cust_names|@count}">
   {html_options values=$cust_ids output=$cust_names selected=$customer_id}
</select>
```

输出:

```
<select name="customer_id" size="3">
    <option value="56">Joe Schmoe</option>
    <option value="92" selected="selected">Jane Johnson</option>
    <option value="13">Charlie Brown</option>
</select>
```

Example 8.11. 数据库例子(如 ADODB 或 PEAR)

```
<?php

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

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

?>
```

下面是模板，注意使用了 `truncate` 修饰器。

```
<select name="type_id">
    <option value='null'>-- none --</option>
    {html_options options=$contact_types|truncate:20 selected=$contact.type_id}
</select>
```

Example 8.12. <optgroup> 下拉组

```
<?php
$arr['Sport'] = array(6 => 'Golf', 9 => 'Cricket',7 => 'Swim');
$arr['Rest']  = array(3 => 'Sauna',1 => 'Massage');
$smarty->assign('lookups', $arr);
$smarty->assign('fav', 7);
?>
```

而模板里:

```
{html_options name=foo options=$lookups selected=$fav}
```

输出:

```
<select name="foo">
<optgroup label="Sport">
<option value="6">Golf</option>
<option value="9">Cricket</option>
<option value="7" selected="selected">Swim</option>
</optgroup>
<optgroup label="Rest">
<option value="3">Sauna</option>
<option value="1">Massage</option>
</optgroup>
</select>
```

{html_radios} 创建HTML单选框

{html_radios}

`{html_radios}` 是一个 自定义函数，用于创建HTML的单选框组和提供数据。 请注意默认选中的情况。

参数名称	类型	必选参数	默认值	说明
name	string	No	*radio*	单选框的名称
values	array	必选，除非使用options属性	*n/a*	单选框的值数据
output	array	必选，除非使用options属性	*n/a*	单选框的显示数据
selected	string	No	*empty*	选中的项
options	数组	必须，除非使用values 和 output属性	*n/a*	单选框的值-显示的数组
separator	string	No	*empty*	字符串中分隔每项的字符
assign	string	No	*empty*	将单选框标签赋值到数组，而不是输出
labels	boolean	No	`TRUE`	在输出中增加<label>标签
label_ids	boolean	No	`FALSE`	给<label> 和 <input> 设置ID属性
escape	boolean	No	`TRUE`	将输出中的/转换(值总是会被转换)

- 必须赋值的属性是 `values` 和 `output` ，除非使用 `options` 来代替。
- 全部的输出标签都遵循XHTML规则。
- 任何不在上面列表中的键值对属性，都会被输出到 `<input>` 标签中作为属性和值。

Example 8.13. {html_radios}第一个例子

```
<?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);

?>
```

模板:

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

Example 8.14. {html_radios}第二个例子

```
<?php

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

?>
```

模板:

```
{html_radios name='id' options=$cust_radios
     selected=$customer_id separator='<br />'}
```

上面的两个例子都输出:

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

Example 8.15. {html_radios} 数据库例子(如 PEAR 或 ADODB):

```
<?php

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

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

?>
```

输出:

```
{html_radios name='contact_type_id' options=$contact_types
     selected=$contact.contact_type_id separator='<br />'}
```

{html_select_date} 创建HTML 日期选择下拉框

{html_select_date}

`{html_select_date}` 是一个 自定义函数，用于创建一个选择日期的下拉框。 它可以显示任何或者全部的年、月、日。 任何不在上面列表中的键值对属性，都会被输出到 `<select>` 标签中作为属性和值。

参数名称	类型	必选参数	默认值	说明
prefix	string	No	Date_	下拉框名称的前缀
time	时间戳, DateTime, mysql时间戳或任何 `strtotime()` 能支持的字符串，或者是数组(当设置了field_array)	No	当前 时间戳	默认选中的日期。如果提供了数组，那么field_array和prefix属性将单独作用在每个数组元素上， 包括年月日。
start_year	string	No	当前年份	下拉框开始显示的年份，可以设置一个年份数字或者默认当前年份(+/- N)
end_year	string	No	same as start_year	下拉框结束显示的年份，可以设置一个年份的数字或者默认当前年份(+/- N)
display_days	boolean	No	`TRUE`	是否显示日期
display_months	boolean	No	`TRUE`	是否显示月份
display_years	boolean	No	`TRUE`	是否显示年份
month_format	array	No	null	月份显示的字符串的数组. 如 array(1 => 'Jan', ..., 12 => 'Dec')
month_names	string	No	%B	月份显示的格式 (strftime)
day_format	string	No	%02d	日期显示的格式 (sprintf)
day_value_format	string	No	%d	日期值显示的格式 (sprintf)
year_as_text	boolean	No	`FALSE`	是否将年份显示为文字
reverse_years	boolean	No	`FALSE`	是否按倒序显示年份
field_array	string	No	null	如果设置了field_array值，则下拉框的值发送的PHP时，将会是 值[Day], 值[Year], 值[Month]的格式。

day_size	string	No	null	附加日期select标签的size属性
month_size	string	No	null	附加月份select标签的size属性
year_size	string	No	null	附加年份select标签的size属性
all_extra	string	No	null	附加给全部select/input标签附加的属性
day_extra	string	No	null	附加给日期select/input标签附加的属性
month_extra	string	No	null	附加给月份select/input标签附加的属性
year_extra	string	No	null	附加给年份select/input标签附加的属性
all_id	string	No	null	全部select/input标签的ID值
day_id	string	No	null	日期select/input标签的ID值
month_id	string	No	null	月份select/input标签的ID值
year_id	string	No	null	年份select/input标签的ID值
field_order	string	No	MDY	显示各下拉框的顺序
field_separator	string	No	\n	显示在各字段之间间隔的字符串

month_value_format	string	No	%m	月份值的显示格式（按 strftime()）默认是 %m
all_empty	string	No	null	该属性可以在每个下拉框的第一行显示文字，并以“”作为它的值。 在需要让下拉框的第一行显示“请选择” 的情况下比较有用。
year_empty	string	No	null	该属性可以在年份下拉框的第一行显示文字，并以“”作为它的值。 在需要让年份下拉框的第一行显示“请选择年份” 的情况下比较有用。 注意你可以使用如“-MM-DD”的值，作为时间属性来显示没有选中的年份。
month_empty	string	No	null	该属性可以在月份下拉框的第一行显示文字，并以“”作为它的值。 注意你可以使用如“YYYY--DD”的值，作为时间属性来显示没有选中的月份。
day_empty	string	No	null	该属性可以在日期下拉框的第一行显示文字，并以“”作为它的值。 注意你可以使用如“YYYY-MM-”的值，作为时间属性来显示没有选中的日期。

温馨提示:

在日期技巧文章中，介绍了 较有用的php函数来将 `{html_select_date}` 值转换成时间戳。

Example 8.16. {html_select_date}

模板代码

```
{html_select_date}
```

输出

```
<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="2006" selected="selected">2006</option>
</select>
```

Example 8.17. {html_select_date} 第二个地理

```
{* start and end year can be relative to current year *}
{html_select_date prefix='StartDate' time=$time start_year='-5'
   end_year='+1' display_days=false}
```

如果当前是 2000年 则输出:

```
<select name="StartDateMonth">
<option value="1">January</option>
<option value="2">February</option>
.... snipped ....
<option value="11">November</option>
<option value="12" selected="selected">December</option>
</select>
<select name="StartDateYear">
<option value="1995">1995</option>
.... snipped ....
<option value="1999">1999</option>
<option value="2000" selected="selected">2000</option>
<option value="2001">2001</option>
</select>
```

{html_select_time} 创建HTML选择日间

{html_select_time}

`{html_select_time}` 是一个 自定义函数，用于创建一个选择时间的下拉框。 它可以显示任何或全部的小时、分钟、秒和上下午。

`time` 属性可以是多种格式。它可以是时间戳，一个格式为 `YYYYMMDDHHMMSS` 的字符串，或者是PHP函数 `strtotime()` 能支持的字符串。

参数名称	类型	必选参数	默认值	说明
prefix	string	No	Time_	下拉框名称的前缀
time	时间戳, DateTime, mysql时间戳或任何 `strtotime()` 能支持的字符串，或者是数组(当设置了field_array)	No	当前时间戳	默认选中的时间。如果提供了数组，那么field_array和prefix属性将单独作用在每个数组元素上， 包括小时、分钟、秒数和上下午。
display_hours	boolean	No	`TRUE`	是否显示小时数
display_minutes	boolean	No	`TRUE`	是否显示分钟数
display_seconds	boolean	No	`TRUE`	是否显示秒数
display_meridian	boolean	No	`TRUE`	是否显示上下午(am/pm)
use_24_hours	boolean	No	`TRUE`	是否使用24小时格式的时间
minute_interval	integer	No	1	分钟下拉框的时间间隔
second_interval	integer	No	1	秒数下拉框的时间间隔
hour_format	string	No	%02d	小时的格式(sprintf)
hour_value_format	string	No	%20d	小时值的格式(sprintf)
minute_format	string	No	%02d	分钟的格式(sprintf)
minute_value_format	string	No	%20d	分钟值的格式(sprintf)
second_format	string	No	%02d	秒数的格式(sprintf)
second_value_format	string	No	%20d	秒数值的格式(sprintf)
field_array	string	No	n/a	显示值数组的名称
all_extra	string	No	null	附加给select/input标签附加的属性

hour_extra	string	No	null	附加给小时下拉框select/input标签的属性
minute_extra	string	No	null	附加给分钟下拉框select/input标签的属性
second_extra	string	No	null	附加给秒数下拉框select/input标签的属性
meridian_extra	string	No	null	附加给上下午下拉框select/input标签的属性
field_separator	string	No	\n	显示在各字段之间间隔的字符串
option_separator	string	No	\n	显示在各选项之间的字符串
all_id	string	No	null	全部select/input标签的ID值
hour_id	string	No	null	小时下拉框select/input标签的ID值
minute_id	string	No	null	分钟下拉框select/input标签的ID值
second_id	string	No	null	秒数下拉框select/input标签的ID值
meridian_id	string	No	null	上下午下拉框select/input标签的ID值
all_empty	string	No	null	该属性可以在每个下拉框的第一行显示文字，并以""作为它的值。 在需要让下拉框的第一行显示"请选择" 的情况下比较有用。
hour_empty	string	No	null	该属性可以在小时下拉框的第一行显示文字，并以""作为它的值。 在需要让小时下拉框的第一行显示"请选择小时"的情况下比较有用。
minute_empty	string	No	null	该属性可以在分钟下拉框的第一行显示文字，并以""作为它的值。 在需要让分钟下拉框的第一行显示"请选择分钟"的情况下比较有用。
second_empty	string	No	null	该属性可以在秒数下拉框的第一行显示文字，并以""作为它的值。 在需要让秒数下拉框的第一行显示"请选择秒数"的情况下比较有用。
meridian_empty	string	No	null	该属性可以在上下午下拉框的第一行显示文字，并以""作为它的值。 在需要让上下午下拉框的第一行显示"请选择上午或下午" 的情况下比较有用。

Example 8.18. {html_select_time}

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

当早上9点20分23秒的时候，模板将显示：

```
<select name="Time_Hour">
<option value="00">00</option>
<option value="01">01</option>
... snipped ....
<option value="08">08</option>
<option value="09" selected>09</option>
<option value="10">10</option>
... snipped ....
<option value="22">22</option>
<option value="23">23</option>
</select>
<select name="Time_Minute">
<option value="00">00</option>
<option value="01">01</option>
... snipped ....
<option value="19">19</option>
<option value="20" selected>20</option>
<option value="21">21</option>
... snipped ....
<option value="58">58</option>
<option value="59">59</option>
</select>
<select name="Time_Second">
<option value="00">00</option>
<option value="01">01</option>
... snipped ....
<option value="22">22</option>
<option value="23" selected>23</option>
<option value="24">24</option>
... snipped ....
<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>
```

{html_table} 创建HTML表格

{html_table}

`{html_table}` 是一个 自定义函数，可使用数组形式的数据来创建一个HTML的 `<table>` .

参数名称	类型	必选参数	默认值	说明
loop	array	Yes	*n/a*	循环赋值的数组
cols	mixed	No	*3*	表格的列数，或者是逗号分隔的列头文字列表，或是列头文字的数组。 如果cols属性为空，但设置了rows，将以rows数量和显示元素的总数进行计算得出列数， 以便每列能显示全部的元素。 如果rows和cols都设置了，那么cols会忽略默认值3. 如果设置cols为一个列表或数组，那么列数将取决于列表或数组的元素个数。
rows	integer	No	*empty*	表格的行数。如果设置为空，但设置了cols，那么将以cols数量和显示元素的总数进行计算得出行数， 以便每行能显示全部的元素。
inner	string	No	*cols*	显示元素的循环方向。 *cols*意味着元素将按"一列一列"地显示。 而*rows*意味着元素将"一行一行"地显示。
caption	string	No	*empty*	表格中 `<caption>` 属性值

table_attr	string	No	*border="1"*	`<table>` 标签的属性
th_attr	string	No	*empty*	`<th>` 标签的属性 (循环)
tr_attr	string	No	*empty*	`<tr>` 标签的属性 (循环)
td_attr	string	No	*empty*	`<td>` 标签的属性 (循环)
trailpad	string	No	* *	在最后行空单元格中填充的字符(如果有的话)
hdir	string	No	*right*	每行显示的方向。可以设置： *right* (从左到右), 和 *left* (从右到左)
vdir	string	No	*down*	每列显示的方向。可以设置： *down* (上到下), *up* (下到上)

- `cols` 属性决定表格可以显示多少列。
- `table_attr` , `tr_attr` 和 `td_attr` 的值决定了 `<table>` , `<tr>` 和 `<td>` 标签的数量。
- 如果 `tr_attr` 或者 `td_attr` 是一个数组， 那么它们的值将被循环交替使用。
- `trailpad` 是在最后行空单元格中填充的字符(如果有的话)。

Example 8.19. {html_table}

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

例子演示如何从PHP赋值到模板并且显示表格。下面是各种输出：

```
{**** Example One ****}
{html_table loop=$data}

<table border="1">
<tbody>
<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>
</tbody>
</table>


{**** Example Two ****}
{html_table loop=$data cols=4 table_attr='border="0"'}

<table border="0">
<tbody>
<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> </td><td> </td><td> </td></tr>
</tbody>
</table>


{**** Example Three ****}
{html_table loop=$data cols="first,second,third,fourth" tr_attr=$tr}

<table border="1">
<thead>
<tr>
<th>first</th><th>second</th><th>third</th><th>fourth</th>
</tr>
</thead>
<tbody>
<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> </td><td> </td><td> </td></tr>
</tbody>
</table>
```

2-7functions函数插件的定义和使用

1. 如何在模板中使用插件

使用smarty 提供的成员方法：registerPlugin() 注册插件

registerPlugin() — 注册插件

说明

```
void registerPlugin(string type,
                    string name,
                    mixed callback,
                    bool cacheable,
                    mixed cache_attrs);
```

该函数将以插件的形式来注册函数或者方法。 参数如下:

- *type* defines the type of the plugin. Valid values are "function", "block", "compiler" and "modifier".
- *name* defines the name of the plugin.
- *callback* defines the PHP callback. it can be either:
  - A string containing the function *name*
  - An array of the form `array(&$object, $method)` with `&$object` being a reference to an object and `$method` being a string containing the method-name
  - An array of the form `array($class, $method)` with `$class` being the class name and `$method` being a method of the class.
- 大多数情况下 *cacheable* 和 *cache_attrs* 可被省略。 参见缓存能力设置它们的值。

如:PHP 代码文中

```
<?php
$smarty->registerPlugin("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());
}
?>
```

模板中:

{date_now}

{* 或定义时间格式 *}
{date_now format="%Y/%m/%d"}

2. 如何注册成功全局插件

在 Smarty/plugin 目录下创建一个php文件，命名方式为：function[block][modifier].插件名称.php

function插件：如我们创建一个 function.test_a.php 的插件文件，内容如下：

```
<?php
function smarty_function_test_a($params){
        echo $params['name'];
}
```

模板中可以直接使用：如 {test_a name='veiol'}

输出结构为：veiol

block 插件：如我创建一个 block.test_b.php 的插件文件，内容如下：

```
<?php
function smarty_block_test_b($params,$content){
        echo $params['name].$content;  //就参数与内容连接输出
}
```

模板中可以直接使用：如 {test_b name="echo}my name is {/test_b}

输出结果为：my name is echo

四、小节

2. 课堂练习

3. 课后练习

4. 资料扩展