

Szablony tematyczne i Smarty

Spis treści

- Szablony tematyczne i Smarty
 - Podstawowe zasady
 - Ograniczniki
 - Komentarze
 - Zmienne
 - Tryby warunkowe
 - Ptle
 - Wczanie pliku
 - Funkcja debugowania
 - Zaawansowane zastosowania
 - Zrzuty
 - Przypisanie zmiennych
 - Zmienna \$smarty
 - Literówki
 - Funkcje
 - Wtyczki
 - Smarty niemile widziane z PrestaShop

Szablony tematyczne i Smarty

Smarty jest silnikiem szablonu PHP który jest wytrzymały, szybki, łatwy w nauce i użyciu, i ma czysty i kompaktowy składnię do projektowania dedykowanego. Pomaga zbudować znacznie prostszy kod HTML i działa poprzez gromadzenie, a następnie buforowanie każdej strony.

Możesz się dowiedzieć więcej wchodząc na oficjalną stronę: <http://www.smarty.net/>

Podstawowe zasady

Ograniczniki

Ograniczniki umożliwiają silnikowi szablonu "rozpoznać" Smarty zwraca się wewnątrz szablonu. Ograniczniki wykorzystywane przez Smarty są w nawiasach klamrowych. The delimiters used by Smarty: `{smarty_call}`



Jeli trzeba użyć rzeczywistych nawiasów klamrowych w kodzie szablonu Smarty posiada specjalne połączenie dla Ciebie: `{redelim}` na lewej nawias klamrowy (`{`), i `{rdelim}` dla prawej nawias klamrowy (`}`).

Komentarze

Jak w przypadku każdego języka programowania lub skryptów, można umieścić komentarze w kodzie szablonu, a silnik szablonu nie będzie ich analizować.

Komentarze używaj regularnych ograniczników, wraz z otwieraniem i zamykaniem * znaków.

```
{* One-line comment. *}

{*
Multiple
lines
comment.
*}

{* Comment for Smarty {$code} *}
```

Zmienne

Tak jak w PHP, zmienne są reprezentowane przez znak dolara, po którym następuje nazwa zmiennej. Umieszczenie zmiennej bezpośrednio w ograniczniku Smarty oznacza, że chcesz wyświetlić zmienną tak jaką jest.

Na przykład, `{$foo}` Smarty jest odpowiednikiem PHP `echo $foo;`

Mona również stosować modyfikatory do zmiennych:

```
{* Displaying the content of the variable in lowercase. *}
{$foo|lower}

{* Displaying the content of the variable after replacing one word with another. *}
{$foo|replace:'bar':'baz'}

{* Example of "chaining" modifiers. *}
{$foo|lower|capitalize|truncate:10:'...'}
```

Możesz uzyskać pełną listę dostępnych modyfikatorów na oficjalnej stronie internetowej: <http://www.smarty.net/docs/en/language.modifiers.tpl>

Tryby warunkowe

Smarty ma warunkowy system `if / else / elseif`:

```
{if $coffee == 'good'}
  {* Happy coder *}
{elseif $coffee == 'very good'}
  {* Very happy coder *}
{else}
  {* Grumpy coder *}
{/if}
```

Możesz się dowiedzieć więcej o różnych trybach warunkowych na oficjalnej stronie: <http://www.smarty.net/docs2/en/language.function.if.tpl>

Ptle

Smarty obsługuje dwie ptle: `section` i `foreach`. Można użyć iteratorów, a nawet trybu warunkowego `foreachelse`:

```

<?php
    $items_list = array( 23 => array('no' => 2456, 'label' => 'Salad'), 96 => array('no' => 4889, 'label' =>
    'Cream' ) );
    $smarty->assign('items', $items_list);
?>
<ul>
{foreach from=$items key=myId item=i}
    <li><a href="item.php?id={$myId}">{$i.no}: {$i.label}</li>
{/foreach}
</ul>

```

Możesz dowiedzieć się więcej o każdej pętli na oficjalnych stronach internetowych:

- <http://www.smarty.net/docs/en/language.function.section.tpl>
- <http://www.smarty.net/docs/en/language.function.foreach.tpl>

Wczytanie pliku

Można także doczytać plik szablonu do innego używając funkcji `include`, `extends` or `block`. Dzięki dziedziczeniu, wczytanie pliku może mieć wpływ na wiele szablonów naraz.

Metoda przypisania używa dwóch obowiązkowych argumentów:

- `file`: Uwzględnia nazwę pliku szablonu.
- `assign`: Nazwa zmiennej wyjścia zostanie przypisana.

Oto kilka przykładów:

```

{* Including a file. *}
{include file='steps.tpl'}

{* Placing the included content in a variable, then display the variable. *}
{include file='text.tpl' assign='MyText'}
{$MyText}

```

Demonstracja dziedziczenia z `extends` i `block`:

```

{* Filename: parent.tpl *}
<html>
<head>
<title>{block name="title"}Default Title{/block}</title>
...
{block name="title"}New Title{/block}
</head>
</html>

{* Filename: child.tpl *}
{extends file='parent.tpl'}
{block name='title'}New Page Title{/block}

```

Ten ostatni przykład używa funkcji `block`, który jest przeznaczony do określenia nazwy powierzchni rządu matrycy do matrycy dziedziczenia.

Można się dowiedzieć więcej na temat dziedziczenia szablonów i każdej funkcji umieszczenia na swojej stronie na oficjalnej stronie:

- <http://www.smarty.net/docs/en/advanced.features.template.inheritance.tpl>
- <http://www.smarty.net/docs/en/language.function.include.tpl>
- <http://www.smarty.net/docs/en/language.function.extends.tpl>
- <http://www.smarty.net/docs/en/plugins.block.functions.tpl>

Funkcja debugowania

Kompletny zestaw procesów wewnętrznych z szablonu Smarty mog by wywietlane, gdy wywietlana jest strona.

Podczas rozwoju tematycznego, mona to zrobi dla kadego adowania strony poprzez edycje pliku `/config/smarty.config.inc.php` i edycje `$smarty->debugging` wartoci:

```
$smarty->debugging = true;
```

Gdy motyw jest na stronie produkcyjnej, mona wczy funkcje debugowania poprzez dodanie dyrektywy `{DEBUG}` w pliku szablonu.

Mona take zarzdz funkcj debugowania bezporednio z PrestaShop w menu "Paramenty Zaawansowane" na sronie "Konservacja" zmie opcje "Konsola Debugownia" do swoich upodoba,

Moesz dowiedzie si wiecej o tej funkcji debug na oficjalnej stronie: <http://www.smarty.net/docs/en/language.function.debug.tpl>

Zaawansowane zastosowania

Zrzuty

Funkcja `capture` umoliwia pobieranie danych wyjciowych szablonu bez wywietlania go. Na przykad: `{capture name="myCapture"} ... {/capture}`

W celu korzystania z takich materiaów naley przywoa `$smarty` super zmienna: `$smarty.capture.myCapture`

Nie zapomnij sprawdzi chwytu przed uciem go:

```
{if $smarty.capture.<name>} or {if isset($smarty.capture.<name>)}
```

Funkcja przechwytywania ma moliwo automatycznego przypisania zmiennej:

```
{capture name='myCapture' assign='myVar'}
...
{/capture}
{* Then, in order to use the content, call the $myVar variable. *}
```

Moesz dowiedzie si wiecej o funkcji `capture` na oficjalnej stronie: <http://www.smarty.net/docs/en/language.function.capture.tpl>

Przypisanie zmiennych

Moliwe jest przypisanie zmiennej do pliku szablonu (widok) uywajc funkcj `assign` :

```
{assign var='myVar2' value='My value'}


{* Will display "My value". *}
{$myVar2}
```

Możesz dowiedzieć się więcej o funkcji `assign` na oficjalnej stronie: <http://www.smarty.net/docs/en/language.function.assign.tpl>

Zmienna `$smarty`

`$smarty` jest to tak zwana super zmienna. To sprawia, że można pobrać kilka przydatnych informacji:

- capture wartości: `$smarty.capture.myVariable`
- wartości GET: `{ $smarty.get.<name> }`
- wartości POST: `{ $smarty.post.<name> }`
- aktualny timestamp: `{ $smarty.now }`, lub niestandardowego formatowania `{ $smarty.now | date_format: '%d-%m-%Y %H:%M:%S' }`
- stałe PHP: `{ $smarty.const.<constant name> }`

 w Smarty 2, `$smarty` mogłyby być stosowane `foreach`:

Gdy pętla jest zdefiniowana jako: `{foreach from=$myarray key="mykey" item="myitem" }`
...można wykonać wezwanie `$smarty.foreach.name.property`

Od PrestaShop 1.5 zaleca się polegać tylko na składni Smarty 3 z szablonu PrestaShop, w związku z tym `$smarty.foreach.varName.property` wezwanie call musi być zastąpione przez `$varName@property` równoważne wezwanie.

Możesz dowiedzieć się więcej o zmiennej `$smarty` na oficjalnej stronie: <http://www.smarty.net/docs/en/language.variables.smarty.tpl>

Literówki

Etykieta `literal` umożliwia blokowi danych do wykorzystania jej dosłownie, bez Smarty próbuje ją interpretować:

```
{literal}
<script type="text/javascript">
  function myFunction()
  {
    ...
  }
</script>
{/literal}
```

Możesz dowiedzieć się więcej na temat funkcji dosłownej na oficjalnej stronie: <http://www.smarty.net/docs/en/language.function.literal.tpl>

Funkcje

Funkcje Smarty nie używają \$ prefix, podobnie jak zmienne: `{debug}`, `{rdelim}`, `{ldelim}`, ...

Mogą one zaakceptować argumenty: `{include file='<name of the file>' }`

Struktura wywołania funkcji jest więc: `{nameOfTheFunction arg1='...' arg2='...' }`

Nie można użyć modyfikatorów na funkcjach, na przykład `{nameOfTheFunction arg1='...' |lower}` nie będzie działać zgodnie z oczekiwaniami. .

Możesz dowiedzieć się więcej o funkcjach Smarty na oficjalnej stronie:

- <http://www.smarty.net/docs/en/language.syntax.functions.tpl>
- <http://www.smarty.net/docs/en/language.builtin.functions.tpl>
- <http://www.smarty.net/docs/en/language.custom.functions.tpl>

Wtyczki

Wtyczki Smarty umożliwiają łatwe rozszerzenie standardowego zachowania. Są one zapisane w PHP. Na przykład, PrestaShop ma wtyczki Smarty, które tworzą specyficzne funkcje obsługi tłumaczeń: `{l}`

Po pierwsze w temacie:

```
{l s='Hello dear viewer'}
```

i w module (nawet jeśli jest nadpisany!) And in a module (even if overridden!):

```
{l s='Hello dear view' mod='myModule'}
```

Możesz dowiedzieć się więcej o wtyczkach Smarty na oficjalnej stronie: <http://www.smarty.net/docs/en/plugins.tpl>

Smarty niemile widziane z PrestaShop

Kilka rzeczy, które trzeba wystrzegać się przy użyciu Smarty:

- Nie rób stałego bezpośredniego połączenia PrestaShop. Bardziej szczegółowo nawet nie używaj `{$smarty.const._DB_PASSWD_}`, z oczywistych powodów..
- Nie zastępuj przypisanych zmiennych PrestaShop.
- Nie twórz niepotrzebnie kodu trudnego do odczytania. Na przykład powstrzymuj się od jakiegokolwiek zawołań `include` od dołączonego już od wewnątrz pliku.
- Nie rób bezpośredniego połączenia PHP. Na przykład nie należy używać `{php} // PHP code {/php}`