Smarty Templating Engine

Extension Key: **smarty**

Table of Contents

Introduction

Reader's note

In order to avoid confusion between the name of this extension (smarty) and the Smarty templating engine as such (Smarty), I will refer to the extension as 'EXT:smarty' and to the Smarty templating engine as 'Smarty' in this document.

What does it do?

EXT:smarty is an extension which integrates the Smarty templating engine into TYPO3. Included in EXT:smarty are connectors for the traditional pi_base extension development scenario as well as the new lib/div mvc scenario, a variety of customized Smarty tags to facilitate template development in TYPO3 and, of course, the Smarty library (currently at version 2.6.18).

Intended Audience

This document is for developers who create extensions using the TYPO3 framework. Basic knowledge of Smarty is a plus, but if you are unfamiliar with Smarty, you should be able to pick it up quickly. For further reference I recommend the following links:

smarty.php.net

Smarty For Template Designers

ZEND Developer Zone: PHP Templating with Smarty

Smarty Cheat Sheet Version 2.0

Smarty Crash Course

Why use Smarty for Extension Development?

If you're familiar with frontend extension development you'll invariably have come across the templating methods the TYPO3 framework offers for extension templating: getSubpart, substituteSubpart, substituteMarker, substituteMarkerArrayCached, substituteMarkerArray, substituteMarkerInObject, fillInMarkerArray. EXT:smarty is an which includes the Smarty Templating Engine as an alternative to these methods. EXT:smarty includes the following components:

– Smarty (currently at version 2.6.18)

– Connectors for the TYPO3 extension development framework (pi_base & lib/div)

– Custom Smarty tags and functions which implement some of the most common TYPO3 features

The key advantage of Smarty are:

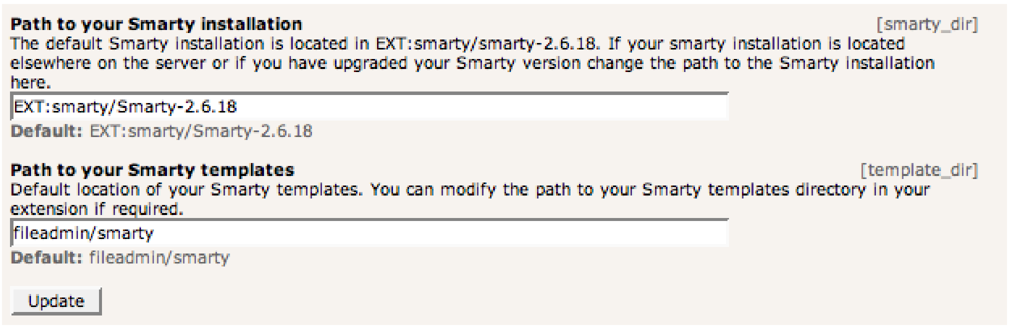
– Easy to use and well documented (no PHP knowledge required).

– Makes it easier to split extension development between developers and designers: Developers can focus on the application, designers can tweak the templates.

– Many advanced features (Variable Modifiers, Template Functions, Filters and Plugins)for handling HTML.

Installation

Install the extension with the extension manager. Once you have created the required directories, you should see a page asking you to define the location of your Smarty installation and the default location of your Smarty templates:



Path to your Smarty installation [smarty_dir]
The default Smarty installation is located in EXT:smarty/smarty-2.6.18. If your smarty installation is located elsewhere on the server or if you have upgraded your Smarty version change the path to the Smarty installation here.
EXT:smarty/Smarty-2.6.18
Default: EXT:smarty/Smarty-2.6.18

Path to your Smarty templates [template_dir]
Default location of your Smarty templates. You can modify the path to your Smarty templates directory in your extension if required.
fileadmin/smarty
Default: fileadmin/smarty

Path to your Smarty installation

The Complete Smarty package (currently version 2.6.18) is included in the extension. By default this is the Smarty installation the extension will use. If you'd prefer to use a different Smarty installation or if you have upgraded your Smarty installation to a newer version, simply change the path to your Smarty installation.

Path to your Smarty templates

This will define a default location for your Smarty templates. This is the path that Smarty will try to locate your Smarty templates in.

Configuration

You can configure EXT:smarty in your extension's TypoScript setup or you can opt for a more direct approach and set Smarty class variables in your extension class (For a complete reference of the available Smarty class variables please refer to the online documentation at smarty.php.net). The following example illustrates 3 different approaches to defining the Smarty class variable 'template_dir' (the path to your Smarty templates):

(i) Use your extension's TypoScript setup:

```
plugin.tx_myextension_pi1.smarty.template_dir = EXT:my_extension/templates
```

(ii) Use the method 'setSmartyVar':

```
$mySmartyInstance->setSmartyVar('template_dir','EXT:my_extension/templates');
```

(iii) Set the Smarty class variable directly:

```
$mySmartyInstance->template_dir = t3lib_div::getFileAbsFileName('EXT:my_extension/templates');
```

TypoScript Configuration

EXT:smarty will interpret any TypoScript settings of the property 'smarty' from your extension's TypoScript as Smarty configuration values. i.e. you can define the Smarty configuration for your extension in the extension's TypoScript setup file. The following example from the TypoScript setup file of an extension 'my_extension' sets the Smarty template directory and turns debugging on:

```
plugin.my_extension {
	//
	smarty {
		template_dir =  EXT:my_extension/templates
		debugging = true
	}
}
```

Direct Configuration

If you prefer you can set Smarty class variables directly in your extension class. If you are familiar with Smarty you will probably feel most comfortable with this approach:

```
$mySmartyInstance->SmartyVariable = value;
```

Alternatively you can use the method 'setSmartyVar'. The setSmartyVar method has the advantage that it perfoms basic sanity checks on the variable and the value you pass (for example, it will resolve TYPO3 path references if the variable requires a path). The syntax is:

```
$mySmartyInstance->setSmartyVar(SmartyVariable, value);
```

Default Configuration

Important: You should always include the static TypoScript template for smarty, in particular if you want to use the TYPO3 plugins. The default settings are:

plugin.smarty.

Property:	Description:	Default:
debugging	Enable/disable the smarty debug console.	false
error_reporting	php error_reporting level inside of display() and fetch()	E_ALL
debugging_ctrl	Alternate ways to enable debugging. NONE means no alternate methods are allowed. URL means when the keyword SMARTY_DEBUG is found in the QUERY_STRING	NONE
debug_tpl	Location of the debug template	EXT:smarty/debug/smarty_debug.tpl
compile_check	Smarty tests to see if the current template has changed (different time stamp) since the last time it was compiled.**This should be set to true for development and debugging purposes.**	true
force_compile	Makes Smarty regenerate the template each time the script is called. Can be set to true for development and debugging purposes.	false
caching	Tells Smarty whether or not to cache the output of the templates. **Use of Smarty's caching mechanism inside of TYPO3 is experimental.**	false
plugins_dir	Location of the Typo3 Smarty plugins, which provide FE functions such as localization, creating TYPO3 links etc.	EXT:smarty/typo3_plugins
cache_dir	Default location of the Smarty cache files	typo3temp/smarty_cache
compile_dir	Default location of the compiled Smarty template files	typo3temp/smarty_compile
autoload_filters.pre.0 autoload_filters.pre.1	Array of prefilters that Smarty will autoload. These include the 'conditions' prefilter (enables using TYPO3 condition statements inside of Smarty if statements.) and the 'dots' prefilter (required for various TYPO3 Smarty plugins)	conditions dots

Developers Manual

Basic usage of Smarty in your extension can be summarized in three steps

Step 1: Create an instance of Smarty

```
// Create a new instance of Smarty
$mySmartyInstance = tx_smarty::smarty();
```

Step 2: Pass your data to Smarty

```
// Pass a variable to Smarty
$text = 'Smarty is a template language and a very useful tool for designers and programmers.'
$mySmartyInstance->assign('info', $text);

// Pass an array to Smarty
$myArray = array('Apple','Banana','Mango','Strawberry');
$mySmartyInstance->assign('fruit', $myArray);
```

Step 3: Render your Smarty template

```
// Process the template and return the parsed HTML
$content = $smarty->display('mySmartyTemplate.html');
return $content;
```

EXT:smarty API reference

Function	Comments

`tx_smarty::smarty`	**Create a new instance of Smary**
	
	Factory API for creating instances of Smarty. Invoking tx_smarty::smarty returns a new instance of Smarty:
	
	`// Create an instance of Smarty`
	`$mySmartyInstance = tx_smarty::smarty();`
	
	Optionally pass an array of configuration values (Smarty class variable => value) with the method to configure the Smarty instance:
	`// Create an instance of Smarty with alternative configuration values`
	`$altSmartyConf = array(`
	`      'template_dir' => 'fileadmin/templates/smarty',`
	`      'debugging' => true,`
	`)`
	`$mySmartyInstance = tx_smarty::smarty($altSmartyConf);`

`setSmartyVar`	**Set Smarty class variables**
	
	Method for setting Smarty configuration variables. Performs basic sanity checks when setting Smarty class variables.
	
	Examples:
	
	`// Change the template directory`
	`$mySmartyInstance->setSmartyVar('template_dir','EXT_my_ext/templates');`
	
	`// Enable caching`
	`$mySmartyInstance->setSmartyVar('caching',true);`

`setPathToLanguageFile`	**Define the default location of your language file**
	
	Defines the location of the default language file. EXT:smarty will try to locate the default language file of your extension so you shouldn't need this method. If however you would like to use a different language file you should use this method:
	
	`// Define location of language file`
	`$mySmartyInstance->setPathToLanguageFile('fileadmin/lang/locallang.xml');`

Special

– EXT:smarty modifies the standard Smarty 'display' method to return the rendered template to your extension instead of directly to the screen.

Debugging Smarty Templates

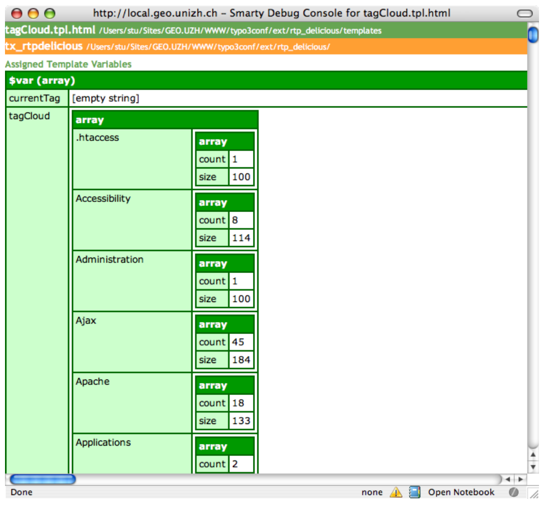
EXT:smarty provides a customized version of the Smarty debug console. To invoke the debug console add the debug tag to your template or configure debugging = true for your Smarty instance:

(i) Add the following tag to your Smarty template:

```
{debug}
```

(ii) Add the following TypoScript to your extension setup (where “tx_myextension_pi1” is your extension key):

```
plugin.tx_myextension_pi1.smarty.debugging = true
```



```
http://local.geo.unizh.ch – Smarty Debug Console for tagCloud.tpl.html
tagCloud.tpl.html /Users/stu/Sites/GEO.UZH/WWW/typo3conf/ext/rtp_delicious/templates
tx_rtpdelicious /Users/stu/Sites/GEO.UZH/WWW/typo3conf/ext/rtp_delicious/
Assigned Template Variables
$var (array)
currentTag   [empty string]
tagCloud     array
             .htaccess        array
                              count 1
                              size  100
             Accessibility    array
                              count 8
                              size  114
             Administration   array
                              count 1
                              size  100
             Ajax             array
                              count 45
                              size  184
             Apache           array
                              count 18
                              size  133
             Applications     array
                              count 2
Done                                              none   Open Notebook
```

TYPO3 Plugins

Smarty has a plugin architecture which can be used to customize it's functionality (for example, by adding your own Smarty tags). EXT:smarty comes with a variety of additional functions and modifiers specifically geared towards using Smarty as a templating engine for extension development inside TYPO3.

{translate}

Description

The translate plugin will enable you to localize your templates. Any text you place between the translate tags will be translated to the current language based on your localization file (i.e. locallang.xml). For this to work Smarty needs to know the location of your language file and the label it should reference in order to translate the text. If you are using the traditional extension framework (pi_base) the plugin will look for the language file in the pi directory of your current extension. If you are using the lib/div framework the plugin will get the current language file from your configuration (pathToLanguageFile). If however your language file is located elsewhere or you would like to use a different localization file than the default for your extension you need to define the location of your localization file:

Define the default language file In your extension's TypoScript configuration:

```
plugin.tx_myextension_pi1.smarty.pathToLanguageFile = path/to/my/localization.xml
```

Alternatively you can define which language file to use from within the translate tag (note that you can also use this feature to reference different language files inside a single template, so it's probably still a good idea to define a default language file):

```
{translate label=”LLL:path/to/my/localization.xml:the.element.label”}Some text that will be
translated{/translate}
```

In your extension class use the method setPathToLanguageFile:

```
$smarty->setPathToLanguageFile('path/to/my/localization.xml');
```

Usage

Parameter:	Description:	Default:
label	Reference to the element label and optionally the language file to use.	
alt	Provide an alternative text output if there is no translation available for the marker	
hsc	Passes the output through htmlspecialchars	false

Examples

(i) Translate text to the current language referencing the element with the label “the.element.label” in your language file:

```
{translate label=”the.element.label”}Some text to translate.{/translate}
```

(ii) Same as (i), but if the parameter “label” is missing the text between the tags is interpreted as the element label:

```
{translate}the.element.label{/translate}
```

(iii) Translate text to the current language referencing the element with the label “the.element.label” in the language file “EXT:my_ext/lang/locallang.xml”:

```
{translate label=”LLL:EXT:my_ext/lang/locallang.xml:the.element.label”}Some text to
translate.{/translate}
```

{link}

Description

The link tag will create links in your template using the TYPO3 typolink function. This will result in links that are compatible with TYPO3 features such as realurl and localization etc. Furthermore the link tag can be used to resolve page ids and aliases.

Usage

Parameter:	Description:	Default:
setup	A reference to a typoscript object in your TYPO3 template, which defines the configuration of the link, for example 'lib.myTypolink'	
->typolink	Any property that is valid for the typolink function for example 'parameter', 'target', 'wrap' etc.	

Examples

(i) Create a link to the page id 59:

```
{link parameter=”59”}About us{/link}
```

(ii) Create a link to an external URL which opens in a separate window and has the class “external”:

```
{link parameter=”http://www.google.com _blank external”}Search the web{/link}
```

(iii) Create an email link:

```
{link parameter=”info@rtpartner.ch”}Tell us what you think!{/link}
```

(iv) Create a file download:

```
{link parameter=”fileadmin/files/specs.pdf”}Further reading{/link}
```

(v) Create a link to the current page which has the class “print”, is wrapped in a div and has the parameter &print=1 added to the URL (Note that the keyword “_self” references the current page):

```
{link parameter=”_self - print” wrap=”<div>|</div>” additionalParams=”&print=1”}Print{/link}
```

(vi) Alternatively you can reference an existing typolink configuration from your template setup using the parameter “setup”:

```
{link setup=”lib.myLink”}Print{/link}
```

Where lib.myLink is:

```
lib.myLink {
      parameter = 59
      AtagParams = class=”board”
}
```

(vii) If you are using the “setup” parameter to define your link configuration you can override any setting from the link parameters. The following example will create a link to page id 60 (and not to page id 59 as defined in “lib.myLink”):

```
{link setup=”lib.myLink” parameter=”60”}Print{/link}
```

Where lib.myLink is:

```
lib.myLink {
      parameter = 59
      AtagParams = class=”board”
}
```

Special

– Note the use of the keyword '_self' in example (v). You can use this keyword to create a reference to the current page.

– If undefined the link function will use the text between the tags to create the title tag for the link.

{url}

Description

The url tag works exactly like the link tag, but it will return only the formatted url (as opposed to the complete link tag). The parameters for the url tag are the same as for the link tag. Usage of the tag varies in that there is no closing tag:

```
Copy this URL {url parameter=”59”} to your browser navigation toolbar.
```

For further details refer to the description of the link tag above.

{format}

Description
Formats a block of text (typically text from an RTE formatted content element) according to your default parseFunc configuration.

Usage

Parameter:	Description:	Default:
setup	Reference to the parseFunc configuration used to format the text.	lib.parseFunc_RTE
->parseFunc	Any property that is valid for the parseFunc function for example 'allowTags', 'denyTags' etc.	

Examples
(i) Parse a block of text according to the rules defined in lib.parseFunc_RTE:

```
{format}

Enthusiastically cultivate go forward process improvements without client-centered results.
Collaboratively incentivize <LINK 23>extensive growth</LINK> strategies whereas extensible imperatives.
Assertively morph innovative e-business via wireless supply chains.

Enthusiastically formulate stand-alone core competencies via backend convergence.

{/format}
```

(ii) Same as (i), but use an alternate parsing configuration:

```
{format setup="lib.parseFunc"}

Enthusiastically cultivate go forward process improvements without client-centered results.
Collaboratively incentivize <LINK 23>extensive growth</LINK> strategies whereas extensible imperatives.
Assertively morph innovative e-business via wireless supply chains.

Enthusiastically formulate stand-alone core competencies via backend convergence.

{/translate}
```

{data}

Description
Returns a value from the current TYPO3 data array. Could for example be used to return the current page title or a value from the register.

Usage

Parameter:	Description:	Default:
source	Reference to the data array defined as type:pointer. For a list of available type/pointer combinations refer to the getText documentation	lib.parseFunc_RTE

Examples
(i) Get the current page title:

```
{data source="page:title"}
```

(ii) Get the current time formatted dd-mm-yy:

```
{data source="date:d-m-y"}
```

(iii) get the value of the header of record with uid 234 from table tt_content:

```
{data source="DB:tt_content:234:header"}
```

{image}

Description
The image tag will render an image using the TYPO3 IMAGE function.

Usage

Parameter:	Description:	Default:
setup	A reference to a typoscript object in your TYPO3 template, which defines the configuration of the link, for example 'lib.myTypolink'	
->IMAGE	Any property that is valid for the IMAGE function for example 'file', 'params', 'border' etc.	

Examples

(i) Render the image in fileadmin/files/myPicture.jpg:

```
{image file=”fileadmin/files/myPicture.jpg”}
```

(ii) Same as (i) but set the width of the image to 100 pixels:

```
{image file=”fileadmin/files/myPicture.jpg” file.width="100"}
```

(iii) Render an image using a configuration defined in your TypoScript setup:

```
{image setup=”lib.myImage”}
```

Where lib.myLink is:

```
lib.myImage = IMAGE
      file = GIFBUILDER
      file {
            XY = 170, [10.h]
            10  = IMAGE
            10.file = fileadmin/images/background.gif
            20  = IMAGE
            20.file = fileadmin/images/picture.jpg
            20.file.width = 150
            20.offset = 10,10
      }
}
```

(iv) The same as (iii), but change the parameters of lib.myImage:

```
{image setup=”lib.myImage” file.20.file=”fileadmin/images/another_picture.jpg”}
```

Known problems

None so far...

Changelog

May, 2006:

– Initial Release

October, 2007:

– Updated to the current version of Smarty (2.6.18)

– Added integration for both the pi_base and the lib/div scenario (in particular with regard to the translation plugin getLL and LLL and to the setting of the correct templates dir)

– Revised all of the plugins in the directory typo3_plugins. I have also added some new plugins and removed some which were either broken or redundant. Note: I believe the framework for the typo3_plugins is much improved now, but as a consequence the backwards compatibility of a few of the plugins is *broken*. So please be careful.

– Added a customized debug console which should work inside the Smarty fetch command and provides debug information relevant to the current Smarty template and TYPO3 extension. The debug console can be activated by setting $smarty->debugging = true or adding the tag {debug} in your template.

November, 2007:

– Moved to extension key 'smarty'