Templates Creation

This document gives guidelines on templates creation using SpinPike™ Lite.

This document is designed for Web Developers to use when building a web site with the help of SpinPike™ Lite. It contains a description of the site building process as well as an example of how to create a template for the home page. If you have any questions that have not been covered in the given document, please contact our technical support service at support@savvybox.com.

Introduction

The templates define the pages layout of a SpinPike™ Lite -powered site. A template is an HTML page with incorporated dynamic blocks in it. These blocks are realized using Smarty (http://smarty.php.net). The Smarty-related documentation is available at http://smarty.php.net/docs.php.

The template files have a *tmpl* extension. The template creation process goes as follows:

1. When SpinPike™ Lite is installed some of the templates are generated automatically. These templates provide for the system correct operation. Besides, a special template directory tree is generated to store the templates. For example, the Content module is responsible for managing the site content. Correspondingly, a *Content directory* is created that stores a template for the site homepage named index.tmpl.
2. You can modify templates either through the administration interface or on the server using any available code editor (for example, HomeSite or Dreamwever).
3. To add a template when in the administration area, you need to browse through **Modules -> Templates**, select the module to which you want to add a template and press **Add Template**. Enter the name for the template (for example, Homepage), enter the file name and press **Apply**. After that the template can be modified.

The given document gives a detailed description of the template creation process. In brief, the process goes like this:

1. A static HTML template is created.
2. The template is either put on the server or its code is copied and pasted through the SpinPike™ Lite administration interface.
3. Finally, the dynamic blocks such as menu or content, etc., are embedded in the templates.

A template can contain pictures and have style sheets and JavaScript files linked to it. Because of the Smarty peculiarities, you cannot use such symbols as "**{**", "**}**" in templates. It is recommended that you link style sheets and JavaScript files externally, and not embed them in the template. However, you may use special command {literal}{/literal} for escaping **{** and **}** symbols.

Pictures and all external files are uploaded to the root directory. For instance, if the server root directory is named *html,* and the pictures are stored in the *images* folder, you need to upload the *images* folder into the *html* directory. As a result, the folders will be arranged as follows: *html/images/*. That is you follow the same procedure as if you were developing a common static site.

Below are described the basic elements used when creating templates. Those are constants, objects and functions.

Constants

Constants are used to create paths to files and URLs. The SpinPike™ Lite constants are called through the `$smarty` variable.

Example: Output of the site address:

```
{$smarty.const.DOMAIN}
```

Constants available:

- UID – unique user ID (is used for the system to identify the user; should be inserted in the static links, etc.)
- DOMAIN – site address (*http://www.example.com*)
- PREFIX- path to the SpinPike™ system root (*/home/www.example.com*)
- DOCUMENTROOT - path to the site pages (*/home/www.example.com/html*)
- TMP – temporary directory.
- TIMEZONE - site time zone
- SOFTWARENAME - name of the system (for internal use)
- AUTHTIMEOUT - system timeout in seconds

Variables

Variables are used to output keywords, description, site title, etc.

$Section

This object is the object of the current section. The object properties depend on the module currently used.

Properties available:

```
{$Section->id}  // section ID
{$Section->name}  // section name
{$Section->header}  // section headline
{$Section->sort}  // sorting index
{$Section->keywords}  // section keywords (META)
{$Section->description}  // section description (META)
{$Section->rel}  // parent section ID
{$Section->Template}  // section object “Template”
```

Objects

You can use objects mainly to create your own templates functions. Besides, they possess such properties that can be already called in the existing functions. If you need to output a particular property of the object and you are not sure, whether it is available and how it is named, please refer to this section.

Module

Designed for system modules management.

Properties:

- `id =>` identifier // number
- `name =>` name // string
- `type =>` type // string

Application Example:
Output of the module name:

```
{$Module->name}
```

Template

Designed for templates management.

Properties:

- `id =>` identifier // number
- `Version =>` Version object // see “Version” object for details
- `name =>` name // string
- `src =>` source file // string
- `Module =>` Module object // see “Module” object for details

Content

Designed for content sections management.

Properties:

- `id =>` identifier // number
- `sort =>` sorting index // number
- `rel =>` parent section identifier // number
- `Image =>` Image object // see “Image” object for details
- `Template =>` Template object // see “Template” object for details
- `name =>` name // string
- `header =>` header (HTTP META) // string

- date => date (typically Unix Timestamp) // number
- date_allow => allow subsections sorting by date // 1 or 0
- description => description (HTTP META) // string
- keywords => keywords (HTTP META) // string
- visibility => hidden section // 1 or 0
- groupid => group access ID (if set to 0 - free access)
- Module => Module object // see “Module” object for details
- href => hyperlink leading to a section // string

Application Example:
Output of a hyperlink leading to the section; section name; section creation date:

```
<a href=”{$Section->href}”>{$Section->name}</a><br>
{$Section->date|date_format}<br><br>
```

ContentItem

Designed for pages content management.

Properties:

- id => identifier // number
- name => name // string
- text => page content // string
- sort => sorting index // number
- rel => parent section ID (“Content” object)) // number

Application Example:
Output a link list at current section pages:

```
{a_section_content_items assign="items"}
{foreach from=$items item="it"}
   <a href="{$it->href}">{$it->name}</a><br>
{/foreach}
```

Image

Designed for images management.

Properties:

- id => identifier // number
- name => name // string
- width => width // number
- height => height // number
- src => name (name of the source file physically located on a hard disk) // string
- type => extension (extension of the source file physically located on a hard disk) // string

- Module => Module object // see “Module” object for details
- string => generated HTML string (<img src="/pict/83c677f175e7d024fda7a3858eaf30b3.jpg" width="1200" height="801" border="0">)

Application Example:
Output of an image:

```
<img src=”/pict/{$Image->src}.{$Image->type}” width=100
height=230>
```

Log

Designed for log files management.

Properties:

- id => identifier // number
- string => notification string // string

Template functions

The templates functions are used for creating dynamic blocks in the site templates. They are a Smarty extension (plug-ins). If you have strong knowledge of Smarty, you will be able to create your own functions. At the moment there are three classes of functions:

- functions to work with site versions (outputting navigation by versions, version name, etc.)
- functions to work with site sections (outputting menus, subsections lists, news, etc.)
- functions to work with the site content (outputting section content)

Site sections functions

Content is the site sections management object. The properties of the section are described in the specification to the given object.

a_section

This function allows to get a site section property with the assigned ID.

Attribute	Type	Mandatory	Default	Description
get	string	no	id	section property to be displayed
id	number	yes	n/a	section ID
date_format	string	yes, if the property value is “Date”	n/a	date format see the documentation provided by Smarty

Example: Output of the section date with the format “day/month/year” and id=64:

```
{a_section id=64 get="date" date_format="%d.%m.%Y"}
```

a_sections

This function allows to access site sections.

Attribute	Type	Mandatory	Default	Description
id	number	no	current section ID	parent section ID for returned sections
assign	string	yes	n/a	name of the sections array
visibility	number	no	0	hidden sections output

Example: Output of three hyperlinks leading from the section with id=64 to the subsections:

```
{a_sections id=64 assign="news" limit=3 module="content"}
```

```
{foreach from=$news item="it"}
   <a href="{$it->href}">{$it->name}</a><br>
{/foreach}
```

a_section_trail

This function outputs the breadcrumb trail navigation string.

Attribute	Type	Mandatory	Default	Description
current	string	no	no	**outputs the name of the current section in the navigation trail**
link	string	no	no	makes the navigation elements appear as links
startpage	string	no	no	name of the homepage
divider	string	no	|	navigation elements separator

Example: Output of the breadcrumb trail navigation, beginning with the Homepage and separated by "»", the elements of which are hyperlinks:

```
{a_section_trail current="yes" startpage="Homepage"
divider="&raquo;"}
```

a_sitemap

This function outputs the site map as a sections tree.

Attribute	Type	Mandatory	Default	Description
totalstr	string	no	no	message to be displayed in a branch when the number of subsections exceeds the limit designated in the "limit" attribute

Example: Output of the site map with the limit in the branch set to 3:

```
{a_sitemap limit=3 totalstr="... sections in total"}
```

a_sections_tab

This function outputs the tab navigation by sections as [1-10] [11-20] [21-30].

Attribute	Type	Mandatory	Default	Description
count	number	yes	n/a	number of sections in a navigation group
rdiv	string	no	n/a	**right-hand separator** do not use **{** and **}** as a separator
ldiv	string	no	n/a	left-hand separator
visibility	number	no	0	output of hidden sections

Example: Output of the tab navigation with the sections grouped by 10 and separated by []:

```
{a_sections_tab count=10 rdiv="]" ldiv="["}
```

a_sections_level

This function enables accessing the required subsections of the current or any other designated section. Using this function you can create menus of different nesting levels.

Attribute	Type	Mandatory	Default	Description
id	number	no	n/a	identifier of parent section for returned sections
assign	string	yes	n/a	name of the sections array
level	number	yes	n/a	subsections level

Example: Output of the second level menu using subsections within the section with id=2:

```
{a_sections_level id=2 assign="news" level=1}
{foreach from=$news item="it"}
   <a href="{$it->href}">{$it->name}</a><br>
{/foreach}
```

a_sections_repmenu

This function outputs a replication menu that looks as follows Company | Services| Contacts

Attribute	Type	Mandatory	Default	Description
id	number	no	n/a	parent section identifier
divider	string	no	|	navigation elements separator

Example: Output of the replication menu using first-level sections and forward slash separator (/):

```
{a_sections_repmenu divider="/"}
```

a_sections_count

This function either outputs or assigns a variable the number of subsections within the current or any other designated section.

Attribute	Type	Mandatory	Default	Description
id	number	no	n/a	section identifier
assign	string	no	n/a	variable name to be assigned to the function output

Example:

```
{a_sections_count assign="count"}
```

a_sections_rand

This function allows accessing random subsections within the current or any other designated section. Using this function you can create an elementary internal banner rotator.

Attribute	Type	Mandatory	Default	Description
id	number	no	n/a	parent section identifier
limit	number	yes	n/a	maximum number of returned sections
assign	string	no	n/a	variable name to be assigned to the function output

Example: Output of a random picture with a hyperlink leading to the section to which the picture is attached:

```
{a_sections_rand limit=1 assign="ad"}
{foreach from=$ad item="i"}
  <a href="{$i->href}">{$i->Image->string}</a>
{/foreach}
```

Site content module functions

ContentItem is a site version management object. Content properties are described in the specification for the given object.a_section_content

This function outputs the content of the current page or section.

Attribute	Type	Mandatory	Default	Description
id	number	no	n/a	section identifier
get	string	no	text	page property

Example: Output of the current page name:

```
{a_section_content get="name"}
```

a_section_content_items

This function allows to access the current section pages.

Attribute	Type	Mandatory	Default	Description
assign	string	yes	n/a	variable name to be assigned to the function output

Example: Output of the list of hyperlinks leading to the current section pages:

```
{a_section_content_items assign="items"}
{foreach from=$items item="it"}
   <a href="{$it->href}">{$it->name}</a><br>
{/foreach}
```

* If you have a question that has not been touched upon in the given document, please contact our technical support service at support@savvybox.com.