Site Press Development Guide

Contents

Site Press – Installing Themes

What are Themes?

Site Press provides an easy way to update the look and feel of a website using *themes*. Themes are pre-packaged sets of files that, when installed, change the overall look and feel of a Site Press site. They can be used to change everything from the default color scheme to the entire design of a site. Themes may also be used to extend the default functionality of Site Press with the use of jQuery scripts and other plugins. While many themes have already been developed and packaged, developers may also create their own.

Choosing a Theme

To view a list of themes available for download, please visit the Site Press Themes Gallery at:

http://themes.SitePress.net

Installing an Existing Theme

Once you've browsed through the online catalog of themes and made a selection, you're ready to install your new theme! Download the selected theme to your local computer and be sure to remember its location. For the purposes of this demonstration, we'll use the Masonry Theme.

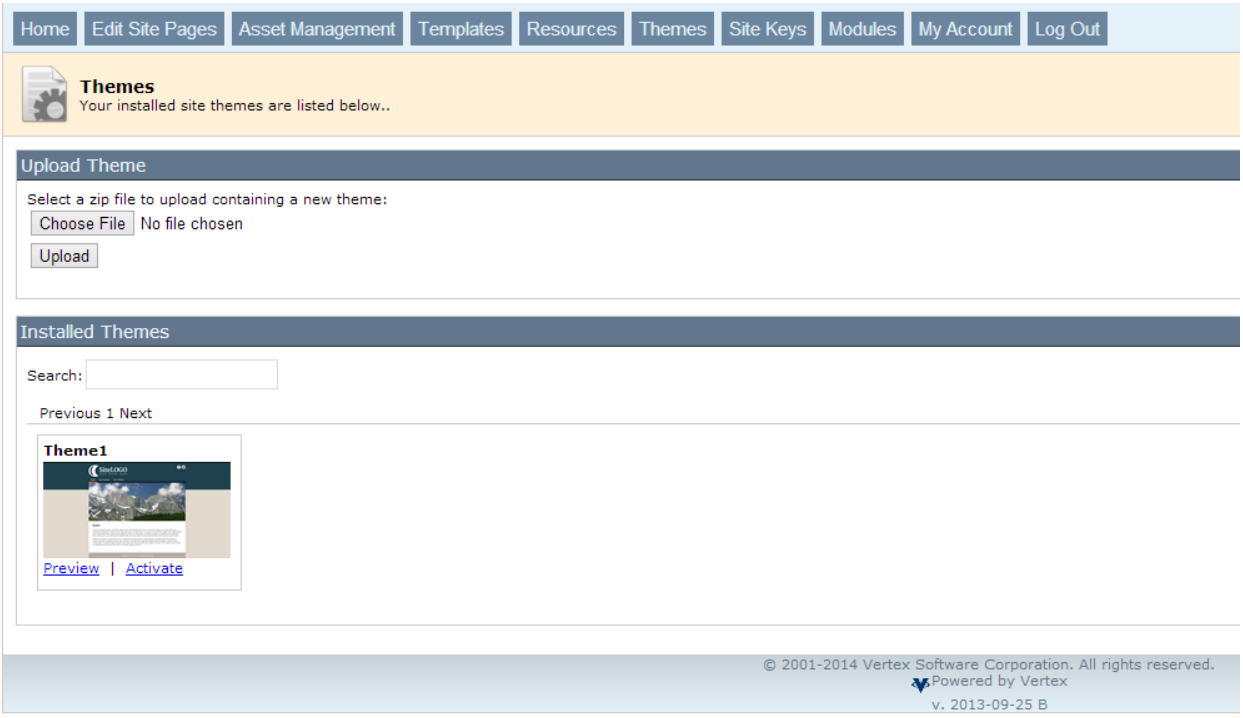

Figure 1. - Theme Administration screen prior to installing new theme.

1. Log into the Site Press Administration page for your website. (By default, the login page for Administration can be found by typing in your website address and then adding "/admin" on the end of the URL (Uniform Resource Locator). For example, "http://mywebsite.com/admin" would take you to the mywebsite.com administration page.)
2. Once logged in, click the "Edit Site" link for the website you would like to change the theme for.

3. Click the Themes tab.
4. Under the Upload Theme section, click the Choose File button.
5. Navigate to the downloaded theme zip file on your local computer and click to select it; then click the Open button.
   The name of the selected theme zip file will appear next to Choose File button, confirming that you have selected a theme.
6. Click the Upload button to upload the theme to your site.
7. Click the Activate link to install the theme on your site.
   The new theme appears in the Installed Themes panel at the bottom of the Themes page.
8. Navigate to the home page of your website to confirm that the new theme has been applied.

   You're done! That's all there is to it.

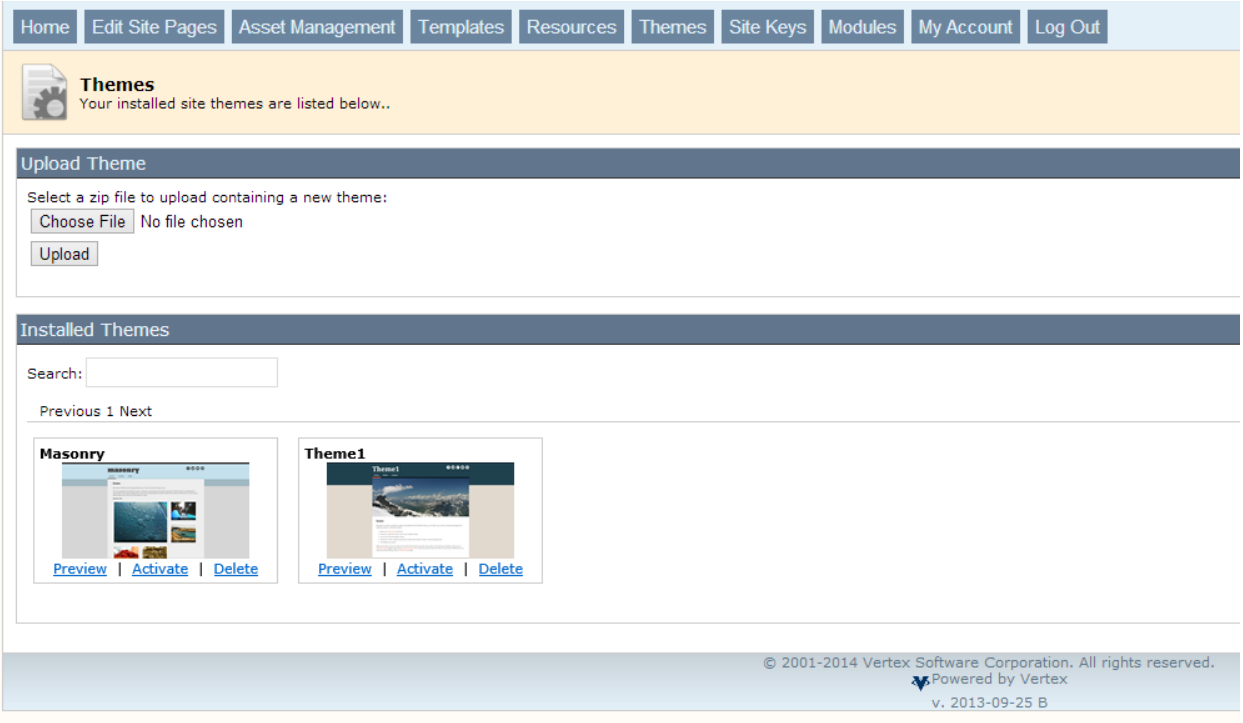

Figure 2. - Theme Administration screen after installing the *Masonry* theme.

Theme Development

Site Press provides professional developers and web-savvy users alike the ability to build custom themes for websites. To get started, download the Masonry theme from themes.SitePress.net so we can look at that for reference. The best way to start theme development is to understand the underlying structure of an existing theme. Site Press themes are packaged as compressed zip files. Download the zip file and extract its contents on your local machine. Browsing the contents of the extracted file reveals that themes adhere to the following structure:

Theme structure	
/assets	The assets folder. This folder should contain any images that will be used by the theme.
/css	The css folder should contain any cascading style sheets that should be included in the theme.
/eot	This folder should contain any Embedded Open Type (eot) files to be copied when the theme is installed.
/js	JavaScript and jQuery files should be placed in the js folder for inclusion in the theme.
/png	This folder should contain any portable network graphics (png) files that should be copied when the theme is installed.
/svg	This folder should contain any scalable vector graphics (svg) files that should be copied when the theme is installed. This includes SVG fonts as well.
/tpl	The tpl folder should contain the following: 1. Site Press templates stored as text files with the extension .txt. 2. Site Press page files stored in XML format with the extension .xml. 3. The theme__assets.xml file lists the assets that should be included when the theme is installed.
/ttf	This folder should contain any True Type Font (ttf) files to be copied when the theme is installed.
/woff	This folder should contain any Web Open Font Format (woff) files to be copied when the theme is installed.
preview.png	The preview image is used by Site Press to show end users a picture of what the theme will look like once installed on a Site Press instance. This file should be placed in the root level of your theme folder, NOT in the "png" sub-folder. (minimum dimensions - 1024x768)
theme.txt	The theme text file is used to describe the contents of theme. Developers use this file to set the theme's name, author, website url, and version number. Comments, which are displayed to end users, may also be placed into the file. Save this file in the root level of your theme folder.

NOTE: For more information on support of different webfont types, visit http://caniuse.com/#feat=fontface. There are many webfont services available online and you can create your own as well with services like http://www.fontsquirrel.com/tools/webfont-generator.

Now that you can see the underlying contents of the zip file, let's look at creating our own theme.

Theme.txt

The **THEME.TXT** file is used to tell the underlying Site Press instance about the theme. Among other things, it indicates what the name of the theme is, who the author is, and the theme version number. It also contains information about the theme's website and any comments the author wishes to include. The **THEME.TXT** file should be included at the root level of the theme zip package and is required to have the same structure as the example below:

Sample Theme.txt File

```
1  // NAME OF THEME – DEVELOPER COMMENTS
2  NAME=MYNEWTHEME
3  AUTHOR=JOE DEVELOPER
4  AUTHORWEBSITE=WWW.SITEPRESS.NET
5  VERSION=1.0
6  COMMENTS=THESE COMMENTS APPEAR WHEN PREVIEWING A THEME.
```

In the sample above, line one contains a comment. This comment allows developer to leave comments for end users that might install the theme. Comments might include further instructions for the user once the theme has been installed and activated on a website. Since developers may include jQuery scripts or other third party components, this section is helpful for providing additional information. Line two of the file indicates the name of the theme. This is the name that will appear in the Theme Admin section of Site Press. In this example, the name of the theme would be *MyNewTheme*. It is important to note that theme names must be unique. If two theme names have the same name, the most recently installed theme will be installed over a theme having the same name. On line 3, the name of the author is included. In this example, the author name is set to *Joe Developer*. Each author may include a link to their website on line 4. Themes may also contain version numbers (line 5). The version number is used by the developer to make enhancements to a theme while still supporting older versions of the same theme. On line 6, the Comments directive tells Site Press to include a description of the theme. This description appears in the Themes Admin section of Site Press.

Preview.png

The preview file is used by Site Press to display an example of how a website will appear once the new theme is installed. The preview file should be created by developers and included at the root level of the theme zip package. In order to adhere to the style guidelines of Site Press its dimensions should, at a minimum, be 1024x768. The file must be named **PREVIEW.PNG** in order to be rendered correctly by Site Press.

Site Press Folder Structure – Sub Folders

Each new website that is created in Site Press has a unique folder on the host file system. A site named *MyWebSite*, for example, would have a corresponding folder created for it on the host file system. To allow for multiple sites to have the same name, Site Press generates a unique folder on the file system for each site starting with the site's name. The name is then appended with random characters for uniqueness. For the purposes of this example, the website *MyWebSite* might have a file system name of *MyWebSite_3FD1B*.

Each website is placed in the Site Press templates folder. To fully understand what is happening here, we must evaluate the structure of Site Press. If the Site Press instance were installed in the folder ***D:\Site Press3***, the full path to the website location would be ***D:\Site Press3\templates\MyWebSite_3FD1B***. This is the same location that theme files are extracted to. Any sub folders included in a theme's zip package are extracted and placed in this location.

NOTE: All sub folders are not treated equally by Site Press.

Template Files – TPL

Template files are raw text files containing Site Press3 templates. One approach to the development of a theme would be to construct templates first in the Site Press interface, then copy the templates to raw text files. To create dynamic content for your templates (such as user-editable content areas, page title, metadata, etc.), use Site Press Tags. Although many template files might be included in a theme, every theme must include the following:

Required Template Files	
theme__home.txt	The template file containing the HTML and layout for the default or home page of a website.
theme__one_column.txt	The template file containing the HTML and layout for pages that use one column.
theme__two_column.txt	The template file containing the HTML and layout for pages that use two columns.
theme_head.txt	The template file defining the HTML Head css includes, script includes, definitions, and layout.
theme_head_home.txt	The specialized template file for the default or home page defining the HTML Head css includes, script includes, definitions, and layout.
theme_footer.txt	The template file containing the HTML and layout for the footer section of a Site Press Page.

NOTE: The main themes contain two underscores (__) in their file names. This notation is used so that, when displayed in the Site Press Template Administration tool, themes associated directly with pages appear first in the list. The word “theme” in these txt files is not a placeholder for your theme name – use the literal word “theme” when naming these files.

Images (Assets) in Themes

In order for a theme to import images into a Site Press instance, it must be included in the **THEME__ASSETS.XML** file. This is a text based XML file with a relatively simple schema. In Site Press3 Images are referred to as Assets. An Image is one particular type of Asset. Other Assets include: JavaScript, CSS, and Include files. Each Asset must be referenced in the XML file in order to be successfully imported into Site Press. The following is an example of a basic **THEME__ASSETS.XML** file:

Sample theme_assets.xml File

```
1  <ArrayOfImportAsset xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
2      <ImportAsset>
3          <Description>Cool picture that I took</Description>
4          <DisplayName>photo1</DisplayName>
5          <FileSystemName>photo1.jpg</FileSystemName>
6          <Extension>.jpg</Extension>
7          <Groups>
8              <ImportGroup>
9                  <Name>Banners</Name>
10             </ImportGroup>
11             <ImportGroup>
12                 <Name>Cropped</Name>
13             </ImportGroup>
14         </Groups>
15     </ImportAsset>
16 </ArrayOfImportAsset>
```

On line 1 of the XML file, a declaration is made opening the **ArrayOfImportAsset** tag. This is the root container which is required in order to import assets. Each image (Asset) is referenced by using the **ImportAsset** tag. Each **ImportAsset** node must contain the following children: **DisplayName**, **FileSystemName**, and **Extension**. **DisplayName** is the name that will be rendered along with an image. This is also used as the default HTML *ALT* tag when rendered by Site Press. In the example above, the **DisplayName** is set to *Photo1* on line 4. The **FileSystemName** directs the importer to the physical file included in the theme zip file. Line 5 directs the importer to include *photo1.jpg*. In addition, all imported assets must register their extension by using the **Extension** tag. In the example above, the **Extension** is set to *.jpg*.

Assets in Site Press also have the option of belonging to **Groups**. Placing assets into groups allows for certain plugins or themes to filter sets of images based upon a group name. For the purposes of this example, the asset with **DisplayName** Photo1 has been placed into two Groups: Banners and Cropped. In order to add an asset to a group, the **ImportGroup** tag is used (Line 8). There is no limit to the amount of **Groups** that can be assigned to an **ImportAsset**.

Packaging a Theme

Once a new theme has been created, it is ready to be packaged. Site Press requires that themes be packaged in zip format. Verify that the theme contains a valid theme.txt file, a preview image, and all subfolders that will be included in the theme. Verify that each subfolder contains the appropriate files to be included: (assets, css, eot, js, png, svg, tpl, etc.). A sample folder structure can be seen in the image below:

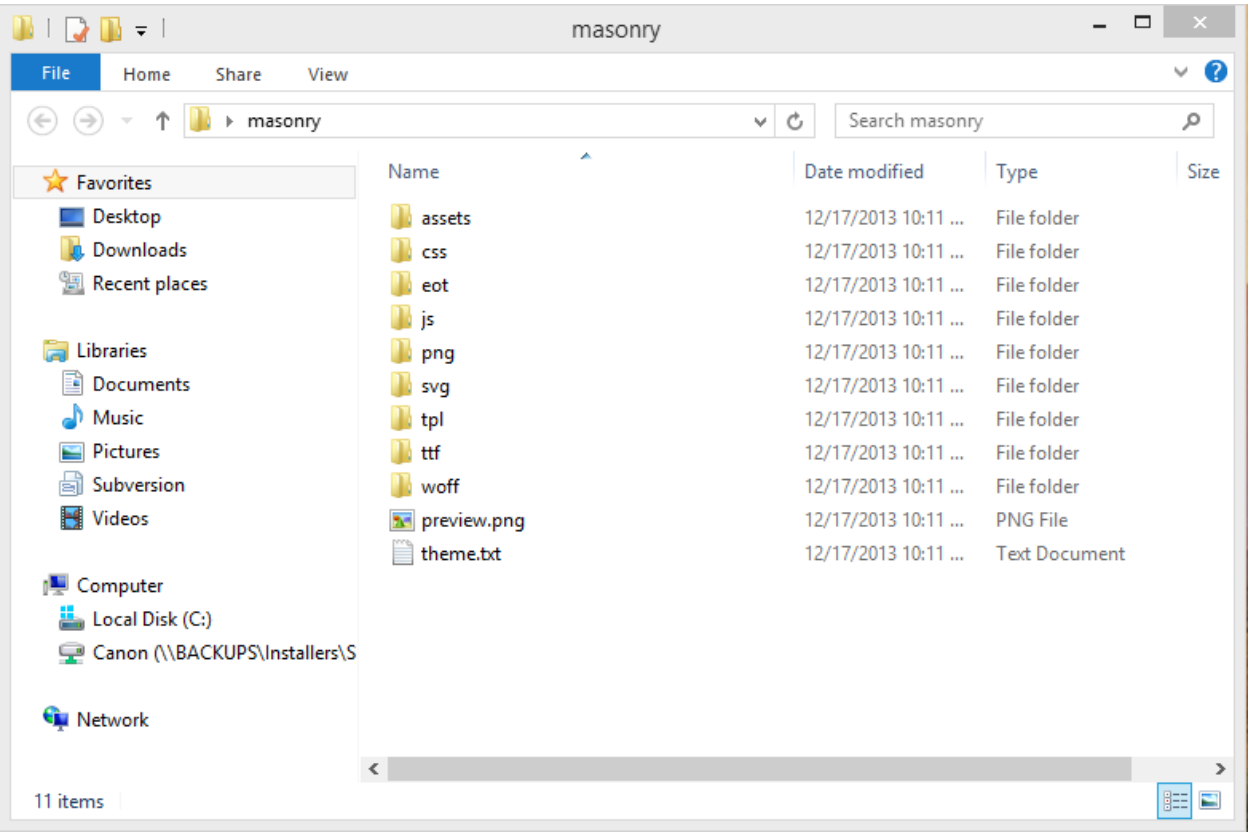

Figure 3. – View of the file system structure of the Masonry Theme.

In Windows, select all files in the folder, right click and select *Send to*. Select *Compressed (zip) folder* to create a zip file containing the theme files.

On a Mac, select all files in the folder, right click and select the option that begins with "*Compress...*" to create a zip file containing the theme files.

Site Press Tags

Site Press Tags are a way for developers and end users to insert dynamic content into Pages. The following is a list of tags and their functionality currently recognized by Site Press.

{{CONTENTAREA}}

Defines an area within a template where content may be placed. (Only valid inside a template)

usage: {{CONTENTAREA:content}}

This creates a content area with the name *content*. This will appear in the "Select Content Area to Edit" DropDownList in the Site Press Page Editor for a Page using the Template in which this CONTENTAREA is defined.

{{CURRENT_YEAR}}

Renders the current calendar year to the output.

usage: {{CURRENT_YEAR}}

This tag, when used in a Template, will be replaced with the Current Year. At the time this document was written it would be replaced with *2013*.

{{DESCRIPTION}}

The {{DESCRIPTION}} Site Press Tag renders the page description as saved in the page properties using the HTML Meta Tag "description". Make sure to include this in your theme <head> template so that theme users have the option of adding unique descriptions to each of their website pages. If a Page Description is added by end users, Site Press renders it in the page output. This is then picked up by search engines, increasing the effectiveness of SEO (Search Engine Optimization). Unlike other popular Content Management Systems, this functionality is built into the core Site Press system, so you don't have to waste time looking for or writing specialized plugins.

usage: {{DESCRIPTION}}

This tag causes Site Press to look up the Description property for a Site Press Page and renders it in place of the Tag. This is typically used as Meta Information about a page. When the Description Tag is used on a Page that has the Page Description of "*This is an example description about this specific page*", the following text will be rendered in the output:

<meta name="description" content="This is an example description about this specific page">

{{INCLUDE_CSS}}

The {{INCLUDE_CSS}} Site Press Tag renders code for the inclusion of a CSS file given a relative path. This type of Site Press Tag should go in the <head> element of the theme's template. Although providing a relative path in the markup, the final output will be rendered as a full link (see example below).

usage: {{INCLUDE_CSS:*relative path starting from the site root-level folder*}}

Example:
{{INCLUDE_CSS:css\site.css}}

Output:
<link rel="stylesheet" href="http://SitePress.net/templates/yourwebsite/css/site.css">

{{INCLUDE_IMAGE}}

The {{INCLUDE_IMAGE}} Site Press Tag renders an image specified by a relative path. For example, a template might be created with an image as the default header banner. If a theme user then chose to replace the default banner image, they would simply upload a new image with the same file name, clicking yes when asked to overwrite the current image. This provides a mechanism to show users what might look good for a given theme.

usage: {{INCLUDE_IMAGE:*relative path starting from the site root-level folder*}}

Example:
{{INCLUDE_IMAGE:assetsheader-banner**.jpg}}**

Output:
<img src="http://SitePress.net/templates/yourwebsite/assets/header-banner**.jpg">**

{{INCLUDE_INC}}

The {{INCLUDE_INC}} Site Press Tag renders the contents of an include file. This becomes useful when a snippet of code needs to be reused in several templates. Using {{INCLUDE_INC}} allows the snippet of code to be edited just once and then have the code updated in all instances of the theme.

An .INC file supports HTML, CSS, JS, etc., but cannot process other Site Press Tags. If other Site Press Tags (e.g., {{PATHNAV}} or {{TITLE}} tags) are required, a template should be used instead along with the corresponding {{INCLUDE_TEMPLATE}} Tag.

A practical example of using the {{INCLUDE_INC}} Tag might be for a small business website. The website has multiple templates, but an update is required to include an advertisement with a photo and link. The {{INCLUDE_INC}} tag can be placed in all of the templates that should show the advertisement.

If there is no content in the referenced .INC file (i.e., the file is blank), Site Press does not add any output when the Tag is encountered. However, once content is added to the .INC file, Site Press will render its contents. This makes it relatively easy to highlight a special promotion on a website, as in the example, without having to update in more than one place.

To create an include file, navigate to the Resources tab in the Site Press Admin pages. Select “Add New Resource.” In the “Create Blank Resource” section, select “.inc” from the file type drop-down and type a name for the file. The “Create File” button will create a new, blank file. To edit the new include file, select it from the resource list and click edit.

usage: {{INCLUDE_INC:*relative path starting from the site root-level folder*}}

Example:
{{INCLUDE_INC:inc/sidebar.inc}}

Output:

Any code contained in the “sidebar.inc” file will be rendered in full form where referenced in a template.

{{INCLUDE_JS}}

The {{INCLUDE_JS}} Site Press Tag renders code for the inclusion of a JavaScript/jQuery file given a relative path.

usage: {{INCLUDE_JS:*relative path starting from the site root-level folder*}}

Example:
{{INCLUDE_JS:js/script.js}}

Output:

```
<script type="text/javascript" src="http://www.SitePress.net/templates/sitename/js/script.js">
</script>
```

{{INCLUDE_TEMPLATE}}

The {{INCLUDE_TEMPLATE}} Site Press Tag renders the contents of a template given the template name. Templates may be referenced inside of other templates.

The {{INCLUDE_TEMPLATE}} Tag might be used for the purpose of creating a common page header. When referenced inside another template, the Site Press Content Engine will process every Site Press Tag included in the template. In other words, if the header template were included in the body template using the INCLUDE_TEMPLATE Tag, all contents of the header template would be processed and then included in the output of the body template. Although the {INCLUDE_INC}} appears similar in functionality, its contents are not processed through the Content Engine.

Another example might be a unique page requiring its own template since it uses a different layout or different content areas, but keeps the same header and footer as the rest of the website. A “unique-page” template might be created and then reference the “Header” and “Footer” templates in the same manner as the “Default-Page” template.

usage: {{INCLUDE_TEMPLATE:*template name*}}

Example:
{{INCLUDE_TEMPLATE:header}}

Output:
Any code in the referenced “header” template will be processed by the Site Press Content Engine and included where referenced in another template.

{{INCLUDE_TEXT}}

The {{INCLUDE_TEXT}} Site Press Tag includes the contents of a text file and renders its output. This is a “utility” tag that provides an easy way to include raw text in a particular part of a theme. As an added feature, this prevents HTML from being included and breaking the template. This feature also prevents inline styling from interfering with existing style sheets (or clashing with a theme).

usage: {{INCLUDE_TEXT:*relative path starting from the site root-level folder*}}

Example:
{{INCLUDE_TEXT:assets/example.txt}}

Output:
The raw text contents of “example.txt” will be included in the template.

{{KEYWORDS}}

The {{KEYWORDS}} Site Press Tag renders keywords saved in Page Properties using the HTML Meta Tag “keywords”. This should be included in the theme’s <head> template to provide theme users the option of adding unique keywords to each of their website pages. Using Page Keywords effectively increases the SEO for pages and the overall site.

usage: {{KEYWORDS}}

This tag causes Site Press to look up the Page Keywords property for a Site Press Page and render them. This is typically used as supplemental Meta Information about a page.
If a user adds "one, two, three, four, five, six" to the Page Keywords field in the properties for a Page, the following will be rendered in the output for that Page:

<meta name="keywords" content="one, two, three, four, five, six">

{{META_NOINDEX}}

The {{META_NOINDEX}} Site Press Tag provides end-users the ability to indicate to search engine robots whether a page should be crawled and indexed or not. This can be done on an individual Page basis. The {{META_NOINDEX}} tag should be placed in the <head> element.

The Robots Meta tag will not be included on a given Page unless the user de-selects the "Include in external search engine (i.e. Google) listings" option in the Page Options section of Page Properties. Essentially, this creates a toggle switch that allows end-users to choose whether they want a Page to be crawled by search engine robots. The {{META_NOINDEX}} Site Press Tag should generally be included in the <head> element of all themes.

usage: {{META_NOINDEX}}

If a webpage has the "Include in external search engine (i.e. Google) listings" option in Page Properties selected, nothing will be shown here – even if the template contains the {{META_NOINDEX}} tag.

If a webpage has the "Include in external search engine (i.e. Google) listings" option in Page Properties de-selected, then the following output will be shown:

<meta name="robots" content="noindex">

{{NAV}}

The {{NAV}} Site Press Tag generates navigation items in a list-based, hierarchal structure. There are several versions of the {{NAV}} tag as well as parameters that can be included (for examples, see below). Individual pages can be removed from the website navigation by de-selecting the "Is In Navigation" Page Option in the Page Properties.

{{NAV}} Tag types:
{{NAV:Auto}}
{{NAV:CHILDREN}}
{{NAV:TopLevel}}
{{NAV:FULLLIST}}
{{NAV:SUBNAV}}

{{NAV}} Tag parameters:
Depth
StartLevel

Usage:
The {{NAV:FULLIST}} Tag always begins with a website's root-level pages. Different output may be generated by adding the ***Depth*** parameter. ***Depth*** determines how many overall levels of navigation should be rendered. When no ***Depth*** parameter is set, {{NAV:FULLLIST}} renders two levels of navigation.

The {{NAV:SUBNAV}} tag can generate different output by adding the ***Depth*** and/or ***StartLevel*** parameters.

To indicate sub-navigation should start with the root-level pages of a website, set the ***StartLevel*** value equal to "1". If sub-navigation should start with third level pages of a website, set the ***StartLevel*** value equal to "3", and so on.

The depth of sub-navigation may be set as well. However, there is an important distinction from the {{NAV:FULLLIST}} Tag. The Depth parameter for {{NAV:SUBNAV}} determines how many *additional* levels of navigation to render after ***StartLevel***.

For example, if you the sub-navigation were to start at the second level pages of a website and show three additional levels of descendants, the following would be used: {{NAV:SUBNAV: Depth="3" StartLevel="2"}}. (More examples below.)

NOTE: The {{NAV:SUBNAV}} Tag requires at least one parameter or it will throw an error.

Example 1:
{{NAV:Auto}}

Output 1:
List-based structure of published pages on the website.

Example 2:
{{NAV:TopLevel}}

Output 2:
List-based structure of pages located at the root-level of the website.

Example 3:
{{NAV:FULLLIST}}

Output 3:
List-based, hierarchal structure of pages starting with root-level pages and showing one level of descendants.

Example 4:
{{NAV:FULLLIST: Depth="4"}}

Output 4:
List-based, hierarchal structure of pages starting with root-level pages and showing three levels of descendants.

Example 5:
{{NAV:SUBNAV: Depth="2"}}

Output 5:
List-based, hierarchal structure of pages starting with root-level pages and showing two levels of descendants.

Example 6:
{{NAV:SUBNAV: Depth="3" StartLevel="2"}}

Output 6:
List-based, hierarchal structure of pages starting with 2nd-level pages and showing three levels of descendants.

Example 7:
{{NAV:SUBNAV: StartLevel="3"}}

Output 7:
List-based, hierarchal structure of pages starting with 3rd-level pages and showing one level of descendants.

{{PATHNAV}}

The {{PATHNAV}} Site Press Tag generates breadcrumb style navigation items that help website visitors easily determine where they are in a website and the path back to the home page.

usage: {{PATHNAV}}

Example:
{{PATHNAV}}

Output:
Breadcrumb style navigation items, starting with the current page's root-level ancestor and followed by all ancestors down to an unlinked reference to the current page.

{{PATH_TO_RESOURCE}}

The {{PATH_TO_RESOURCE}} Site Press Tag adds the relative path to a Site Press resource to the output. This output will provide a path that is relative to the hosting site. See example below. To include full paths to files, use the corresponding Site Press Tag – {{INCLUDE_CSS}}, {{INCLUDE_JS}}, etc.

usage: {{PATH_TO_RESOURCE}}

Example:

<link rel="stylesheet" href="{{PATH_TO_RESOURCE:css\site.css}}">

Output:
<link rel="stylesheet" href="/templates/yourwebsite/css/site.css">

{{PAGE_ID}}

The {{PAGE_ID}} Site Press Tag renders the Page ID for a Site Press Page. The PAGE_ID tag renders the internal Site Press Page ID for the currently executing Page. This is often helpful for developers who construct CSS classes for unique Pages. Page IDs are unique long integer values. Using the {{PAGE_ID}} Tag on an internal Page with ID 777 would render the string value “777” to the output of a Page.

usage: {{PAGE_ID}}

Example:
<body class="page_{{PAGE_ID}}">

Output:
<body class="page_112401">

{{PAGE_WINDOW_TITLE}}

The {{PAGE_WINDOW_TITLE}} Site Press Tag renders the "Page Window Title" as defined in the Page Properties.

usage: {{PAGE_WINDOW_TITLE}}

Example for Page with Page Window Title set to “All About My Dog, Trixie”:
<title>{{PAGE_WINDOW_TITLE}}</title>

Output:
<title>All About My Dog, Trixie</title>

{{PARENT_TITLE}}

The {{PARENT_TITLE}} Site Press Tag renders the Page Title of the parent Page as defined in the parent’s Page Properties. An example of this might be for category-based sections of a website.

usage: {{PARENT_TITLE}}

Example for a child of a page with the Page Title of “Nature”:
Topic – {{PARENT_TITLE}}

Output:
Topic – Nature

{{PARENT_WINDOW_TITLE}}

The {{PARENT_WINDOW_TITLE}} Site Press Tag renders the "Page Window Title" of the parent Page as defined in the parent's Page Properties.

usage: {{PARENT_WINDOW_TITLE}}

Example for a child of a page with the Page Window Title of "Articles on Nature":
See more in {{PARENT_WINDOW_TITLE}}.

Output:
See more in Articles on Nature.

{{REDIRECT}}

The {{REDIRECT}} Site Press Tag stops the current execution of a page and redirects to a different Site Press Page in the same domain or site. This is a 302 redirect, which is a "temporary redirect" for the page. This type of redirect comes in handy if you need to perform maintenance on a webpage and want to divert users to another page in the meantime. To learn about other types of redirects and when to use them, check out http://moz.com/learn/seo/redirection.

usage: {{REDIRECT: *relative path starting from the site root-level folder*}}

Example:
{{REDIRECT:/contact}}

Output:
Site Press stops the current execution of the page a user was attempting to access and redirects them to the contact page. (For example, if the website were "mywebsite.com" the example above would redirect the user to "mywebsite.com/contact"

{{SITE_CURRENT_HOST}}

The {{SITE_CURRENT_HOST}} renders the current host name and port for the request.

usage: {{SITE_CURRENT_HOST}}

The SITE_CURRENT_HOST tag renders the host name for the currently executing Site Press Site. Using the above tag in a Template or Page with a Site Host of "www.SitePress.net" would render the following value:

www.SitePress.net

{{SITE_ID}}

Retrieve the SITE ID for the Site Press Page and render it.

usage: {{SITE_ID}}

The SITE_ID tag renders the internal Site Press Site ID. This is often helpful for developers who use the Site ID for scripts. Site IDs are unique long integer values. Using the above tag in a Template or Page with a Site ID of 15 would render the string value "15" to the output of a Page.

{{SITE_KEY}}

Render the contents of a user defined Site Press variable known as a Site Key.

usage: {{SITE_KEY:sm-Facebook}}

Each Site Press Site contains a collection of user defined variables known as Site Keys. Using the SITE_KEY tag along with the name of the Site Key to be used, renders the value of the property. In the example above, the "sm-Facebook" key name contains a value of "www.facebook.com". The output of the example above will be:

www.facebook.com

{{SITE_TITLE}}

Render the title of the current site as it is set in Site Press Site Properties.

usage: <title>{{SITE_TITLE}}</title>

This tag causes Site Press to look up the Global Window Title property for a Site Press Site and renders it in place of the SITE_TITLE tag. For a Site with a Global Window Title of "My Site Name", when used in conjunction with the HTML Title tag, would be:

<title>My Site Name</title>

{{TITLE}}

Render the title of the Site Press Page as set in its properties.

usage: {{TITLE}}

This tag causes Site Press to look up the Title property for a Site Press Site and renders it in place of the TITLE tag. For a Page with a Title of "My Page", when used in conjunction with the HTML Title tag, would be:

<title>My Page</title>

{{WEBPART}}

Render a Site Press3 .NET WebPart and its controls as part of the Site Press Page.

usage: {{WEBPART:WebPartName}}

WebParts are an advanced feature of Site Press3. For more information on usage, please refer to the WebParts section of the documentation.

Site Press Plugins - WebParts

Site Press has the ability to include functionality and interactivity in small reusable pieces called Plugins. While these add ons are known to end users as Plugins, developers refer to them as WebParts. From here on out in the developer guide they will be referred to by their technical name, WebParts.

Site Press WebParts are built from Microsoft ASP.NET UserControls. This means that they are highly extensible and have the full power of the .NET Framework at their disposal. However, this also means that they are very easy to create. Creating a Site Press WebPart requires little more than starting with a standard .Net user control, inheriting a base class, and adding some required methods. The rest is up to you.

WebParts may be simple or complicated, some examples of WebPart functionality are:

- Contact Forms
- Include other .NET controls (i.e., AJAX ToolKit)
- List database records using a .NET Repeater Control
- Render a page property

CMSWebPart Base Class

All Site Press WebParts inherit the CMSWebPart class. It is through the inheriting of this base class that Site Press is able to inspect WebParts for various properties and methods to determine how to render it.

The CMSWebPart base class itself inherits from System.Web.UI.UserControl, but also adds some additional methods and properties:

Property: TagName

The TagName of the WebPart is important. This directly translates to the markup that will be created for the individual WebPart. It should be unique from other WebParts in a given site. Two WebParts with the same name cannot be installed into the same site.

Property: DisplayName

This property is a placeholder for future functionality.

Property: Description

This property is a placeholder for future functionality.

Property: CommandName

This property is a placeholder for future functionality.

Property: CommandArgument:

This property is a placeholder for future functionality.

Method: AssignRenderProperties()

This method must be supplied in a WebPart as it is called by Site Press at runtime to determine what the WebParts specific requirements are. This is also where the name of the WebPart is set. An example implementation of AssignRenderProperties is provided below:

```
// AssignRenderProperties
 public override void AssignRenderProperties()
 {
     this.TagName = "SP3_ContactUs";


     RenderProperties.Add("EmailToAddresses", new CMSWebPartProperty()
     {
         Name = "Email To Addresses",
         PropertyType = typeof(string),
         Value = "",
         IsRequired = true
     });
 }
```

Method: IsHostedInCmsPage()

The IsHostedInCmsPage helper method is provided through the base class. It is used to determine if the WebPart is currently hosted in a Site Press page. This is primarily used during development when the WebPart is being developed in a wrapper aspx page. Used in this context, some CMS-only features are disabled.

Method: HasNoParameters()

Some WebParts use URL segments past the "host page" as parameters. This helper method (which checks IsHostedInCmsPage internally) indicates whether such parameters are available.

CMSPage and PageContent Access

Through a WebPart, developers have access to the Site Press PageContent class. Note that the IsHostedInCmsPage() method should be called first. There is a wealth of information that may be obtained from PageContent. In order to retrieve the properties of this object, cast the Page the control is on to a CMSPage. Through the newly created object, the "page" property is accessible:

```
var pageContent = ((Site Press3.Classes.CMSPage)this.Page).page;
```

Developers now have access to the Entity Framework Page model for the currently executing Page. Through this object, properties such as PageTitle, PageDescription, PageUrl, PageKeywords, and many others are accessible.

Note: While it may appear that the Entity Framework navigation properties are available as well, the Page's Context has gone out of scope by this time and is no longer accessible.