

Muse CMS

Creating Article Types and Building Templates

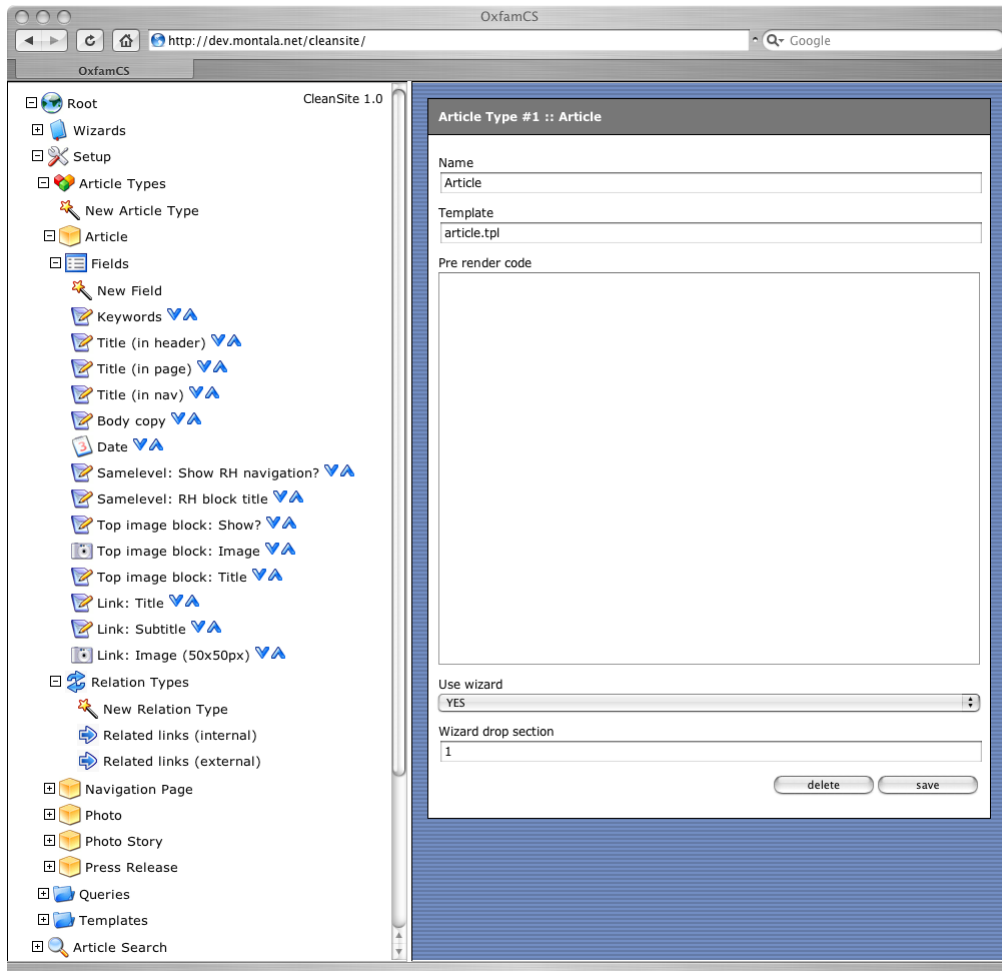


Contents

Creating a new article type.....	2
Templates.....	3
Article fields.....	3
Using a field within a template.....	6
Using Javascript in a template.....	6
Using an uploaded file within a template.....	6
Conditional blocks.....	6
Related Article Types (Relation Types).....	6
Navigation.....	8
Including other template files.....	8
Queries (advanced).....	9

Creating a new article type

Article Types are created under the “Setup” folder. The name you specify is internal only and will not be published to the website. Once the article type has been created, you will need to specify a Smarty template to use by clicking on the item and entering a value under “Template” in the right pane.



Additional options for the article type are:

- **Pre-Render Code** (advanced) – This is a place to enter PHP code that will be executed before the article is rendered. It may be used to feed extra data (possibly from a remote source) into a Smarty template. It has been provided to provide support for special cases and is not required for the vast majority of article types.
- **Include in Wizard** – If set to 'Yes' then this article type will appear in the 'Wizards' section of the tree and users will be able to create new articles of this type using the wizard functionality.

- **Wizard Drop Section** – The ID number of the section that the new articles will be placed in, when created using the wizard functionality. E.g. a 'press release' article type may have a suitable 'press release' section that new articles should be created within.

When expanding the Article Type in the tree using the 'plus' icon, there are two folders underneath the article type, “Fields” and “Relation Types”. Fields are the variables that are fed directly to the template and can include items such as text boxes, files and options. Relation types define the ways in which we can relate other articles to articles of this type. There may be several different relation types for a given article type, for example a news article may have “related internal links”, “related external links” and “related photos”.

Templates

The template itself can be created and edited under the “Templates” folder also under “Setup”. Typically there will be one template per article type, however a template can call additional templates (using the Smarty ‘include’ tag) where common functionality is required across several article types, for example a site header and footer or a related links block.

The templates should have the extension “.tpl”. These are Smarty templates and further documentation can be found on the Smarty website at <http://www.smarty.net>. The crash course is very useful and covers most of the functions required; this has been included as an appendix at the end of this document.

Article fields

Article fields are created by clicking “New Field” under the “Fields” subfolder. Once a field has been created, the options available are as follows:

Field #7 :: Link: Image (50x50px)

Name

Field type

Help text

Include in wizard

Use as article name

Update database column

Template variable name

Upload path

Jpeg width

Jpeg height

Copy/scale image from field...

- **Name** – The internal name only, this is displayed within the CleanSite system only and is not published.
- **Field Type** – This defines how the field is presented to the user entering the data. Options are: single line text, medium text box, large text box, file upload, checkbox and date. Checkboxes are used for conditional blocks within the template as explained below.
- **Help Text** – Text entered here will appear above the field and can be used to offer guidance regarding the correct use of the field.
- **Include in Wizard** – If set to 'Yes' then the field will appear in the wizard for this article type. This should therefore be set to 'Yes' for any required fields.

- **Use as Article Name**– If set to 'Yes' then the article name in the CMS will be updated to the value of this field when the user saves the field. So for example you may want a 'title' field to automatically update the article name. It is wise to have one field set to this value if using the wizard functionality.
- **Update Database Column** (advanced) – The name of the database column in the articles table to update when this field is updated. This means some of the article properties (such as 'show in navigation') can be mapped to fields.
- **Template Variable Name** – The name used in the template. This must follow PHP variable naming rules, i.e. it should not contain any spaces and must only contain the characters 0-9, a-z, A-Z and the underscore character (_). It's best to stick to lower case to avoid confusion.
- **Upload Path** – This is only used if the field type is 'file upload'. This defines a path on the system, starting from the 'assets' folder, in which the uploaded file will be stored. The character '?' will be replaced with the article number. So in the above example: the upload path is set to 'articles/link_?.jpg'. A file uploaded to this field within article number 23 would be uploaded to the path 'assets/articles/link_23.jpg'. This is defined here so the content editors don't need to worry about file paths when uploading; they simply see that the file has been uploaded and attached to the article. Any name the file had prior to the upload is discarded. The upload path is essentially a template for the new path and filename.

Note that all uploaded files are automatically copied as part of the publishing process, that is - if they have been altered since the last publish. This is the same way that HTML publishing works – only files that have changed will be transferred. Because of this, there should be no need for content editors to manually transfer files between dev and live servers.

- **JPEG Width / Height** – This is only used if the field type is 'file upload' and you wish the user to upload a JPEG image. If a width and height is set then the image will be automatically scaled to these dimensions. If you are using ResourceSpace integration then these dimensions will be used to allow the selection of a suitable area of the selected resource image.
- **Copy / Scale Image From Field** – This is only used if the field type is 'file upload' and is used in conjunction with the above width/height options. A field can be selected that is used as the source field. When a JPEG file is uploaded to that field, then a suitably rescaled copy is uploaded to this field too. So for example it is possible to have a main image at 150x100px which, when a file is uploaded, will automatically generate the thumbnail image at 50x60px. Several 'children' fields can be selected in this way so a main image could automatically generate several other sizes of images for other fields.

Using a field within a template

Within the template code, a field is simply referred to using the name as entered under "Template Variable Name" in the field setup above. Curly braces and a dollar sign are used to differentiate between a Smarty variable and normal HTML.

```
<p><span class="heading">{$title}</span></p>
<p>&nbsp;</p>
{$body}
```

In the above example, {\$title} and {\$body} will be replaced with the field data entered for the title and body fields respectively.

Using Javascript in a template

The curly braces used within Javascript can confuse the template parser, so an additional tag can be used to temporarily turn off template parsing for a given block of HTML. Place a {literal} tag before the block of code and a {/literal} closing tag after the block. Smarty will ignore any curly braces within this block.

Using an uploaded file within a template

Referring to an uploaded file is much the same as for normal field data as above. The URL of the file will be replaced with the matching tag instead of any text. So an image uploaded to a field with the template variable name set to 'image' would be displayed simply using . The URL of the uploaded image is automatically entered.

Conditional blocks

The 'checkbox' field type can be used to toggle certain areas of the page on or off, so for example we could decide if we wanted to display a related links title using this code:

```
{if $related_internal}
<table border="0" cellpadding="5" cellspacing="0" width="214">
  <tbody><tr>
    <td class="boxoliveheading" align="left" bgcolor="#cccc99" valign="top">Related links</td>
  </tr>
</tbody></table>
{/if}
```

'related_internal' is the template variable name of our checkbox field. The HTML in between the {if} and the {/if} is that which will be toggled on and off as the field value is changed.

Related Article Types (Relation Types)

These work in a similar fashion to fields, however a list of article 'results' is accessible within the template rather than a single block of text. In the admin system the template variable name is set in a similar fashion to the fields:

Relation Type #1 :: Related links (internal)

Name

Template variable name

Limit to articles of type (optional)

Include in wizard

Reverse relation (optional)

Fields are as follows:

- **Name** – The internal name only, this is displayed within the CleanSite system only and is not published.
- **Template Variable Name** – The name used in the template. This must follow PHP variable naming rules, i.e. it should not contain any spaces and must only contain the characters 0-9, a-z, A-Z and the underscore character (_). It's best to stick to lower case to avoid confusion.
- **Limit to Articles of Type** – If specified, then only the articles of the type selected will be available when selecting related articles for this relation. For example you could set the 'Related Press Releases' relation so only articles of type 'Press Release' could be selected.
- **Include in Wizard** – If set to 'Yes' then the relationship will appear in the wizard for this article type.
- **Reverse Relation** – If selected, the relationship chosen will also be updated 'in reverse' when this relationship is updated. For example, if an article of type 'News Story' has a 'Related Photos' relationship, and the 'Photos' article type has a 'Related News' relationship, you may want the system to automatically add related news for a given photo to a photo's related news section, and hence the relationship is two directional rather than one.

Within the template we can then loop through related articles for this relation type as follows:

```

{section name=nav loop=$related_internal}
<p><a href="{ $related_internal[nav].url}"></a></p>
{/section}

```

The {section} and closing {/section} tags denote the block of HTML which we would like to be repeated, once for every related article. The 'name' of the section is the name we use when referring to the repeated elements. For related articles, all the article field data is available; it's possible to refer to any article parameter from related articles from within the repeating block. In this case we are extracting and displaying the 'url', 'linkimage' and 'linktitle' fields from the related articles.

Note that all the above techniques can be used together and nested inside one another, so a conditional block can exist within a related links block and vice versa.

Navigation

All articles exist within a hierarchical tree structure. That is, every article exists within a section, and that section may exist within another section, all the way down to the site root. The various levels of navigation are available to the template as effectively related links blocks. The 'root' level of the tree will be passed to the template using the variable name 'navigation0', the next level will be 'navigation1', and so on. If the section is related to an article (as all sections should be), then all article data will also be passed to the template, just as with the related types blocks above.

Additionally, articles within the article's current section (the article's siblings) will be passed to the article with the variable name 'samelevel'.

An example:

```
{section name=nav loop=$navigation[1]}  
<p><a href="{ $navigation[1][nav].url}">{ $navigation[1][nav].name}</a></p>  
{/section}
```

This displays all sections at navigation level 1 (one down from the root). The URL and name of each section's associated article is extracted and displayed.

For both 'samelevel' and 'navigation' variable blocks, an additional parameter is passed to the template – 'current'. This is set to true if the current article is within this section (or one of the children sections) and hence may need to be visually represented to the user as the current section, perhaps using bold text.

Including other template files

Other templates can be included within a template using the {include} tag. This allows, for example, a global header and footer to be created which will remove duplication within templates. To include a file named 'header.tpl' use the syntax {include file='header.tpl'}. All fields, navigation elements, relation type blocks and so on will be available from within the included template just as if it was a part of the main template.

Queries (advanced)

Under "Setup", the "Query" folder allows custom SQL queries to be created that produce specific lists of articles. This is like a custom version of the 'samelevel' block discussed above. The query can be looped through in the template just as for 'samelevel' and relationship types.

The query must return an 'article' column which lets the system know which article is being referred to and should be used within the template loop.

The "Specific to article type" option is optional and means the query will only be executed for articles of that type. For example if (as below) you have a 'recent press releases' query, and it is only needed for the home page, it makes sense to set this field to 'Home Page' only to improve publishing time.

Query #1 :: recent_press_releases

Name and template variable

Specific to article type (optional)

Query

```
select ref article from article where article_type=7 and show_in_navigation=1 order by date desc limit 10
```