GetActive

WSM Design Handbook

A Guide for Designers and Information Architects

Author	Revision	Date	Comments
Peter Genuardi	0.1	March 8, 2006	Initial Draft
Bruce Keilin	0.2	March 17, 2006	Added Definitions; added content from Kristin's WSM Training PPT; added original content.
Bruce Keilin	0.3	March 30, 2006	Draft Version presented to PS Team.

TABLE OF CONTENTS

1 Introduction

1.1 Goal of This Document

The goal of this document is to help designers produce information architecture, markup and cascading style sheets for sites built using GetActive's WebSite Management (WSM) content management tool.

This document will:

- describe the structure and relationship of pages and components in WSM;
- suggest optimal uses of WSM to achieve the implementation of a scalable, easy-to-manage website; and,
- provide examples of markup for elements over which designers have control.

1.2 Audience for This Document

This document has been written for designers who will be responsible for producing the look and feel of websites that will be implemented using WSM. It is also useful for information architects who are planning wrappers, content types, templates, categories and the rest of the pieces that make a complete WSM site.

It is assumed that designers will have a strong understanding of the general principles of website design – focusing on end user tasks, developing information architecture – as well as the technical expertise necessary for producing accessible HTML markup and CSS.

2 Definition of Terms

There are a number of GetActive-centric terms for concepts that you are likely already know about. A strong understanding of these elements will help you conceptualize information architecture and design schemes that will fit well into WSM. You should familiarize yourself with the terms listed in the glossary on page 37.

2.1 Website Management (WSM) Module

GetActive's Website Management Module is a Content Management System which enables non-technical managers to maintain a website of any scale that integrates seamlessly with membership, fundraising, messaging, advocacy, and other outreach initiatives.

WSM automates the process of creating, publishing and maintaining content for your website. WSM simplifies content production by empowering authors, editors, publishers and managers by enabling them to collaboratively create and edit content, manage the approval and archival processes, and perform other related tasks.

WSM consists of a system for defining the properties of your content types, dynamic forms with WYSIWYG editors for authors to input content, a content repository for storing content and a template-based presentation system for serving content in a consistent look-and-feel.

WSM provides seamless integration between GetActive's Fundraising, Advocacy and Membership Management modules and your WSM-powered website. WSM provides designers control over almost all of the code in every HTML page, and provides full support for CSS. WSM also supports Javascript, DHTML, Flash, IFRAMES and other web technologies.

Pages presented by WSM are built on Wrappers, Content Types, Display Templates and Components.

2.2 Wrappers

Wrappers are the HTML that defines the overall structure of a web page. A typical wrapper includes the main navigation, branding, side navigation and footer. The Wrapper includes an area for the web page body.

2.3 Content Types

A content type is a set of fields or properties that define an object that is managed by WSM. Content Types that are based on Web Page or Image may also contain one or more display templates.

2.4 Display Templates

A display template is HTML (and optionally embedded server-side logic) that defines the consistent formatting of content items. Single Display Templates render one content item, while List Display Templates render a list of content items.

2.5 Components

Components are reusable objects that perform specific functions and are rendered in HTML. Components can be included in Web Pages. Some components can be included in Wrappers.

3 Page Structure

WSM pages are built using wrappers and templates. Wrappers provide the overall structure surrounding the content on a Web Page. Wrappers usually contain the header, branding and navigation elements that are common to most pages of a website. Templates control the structure of the body of a Web Page. Templates may include components, some of which may themselves use templates.

The following diagram shows the relationship between Wrappers, Page Templates and Page Component Templates:

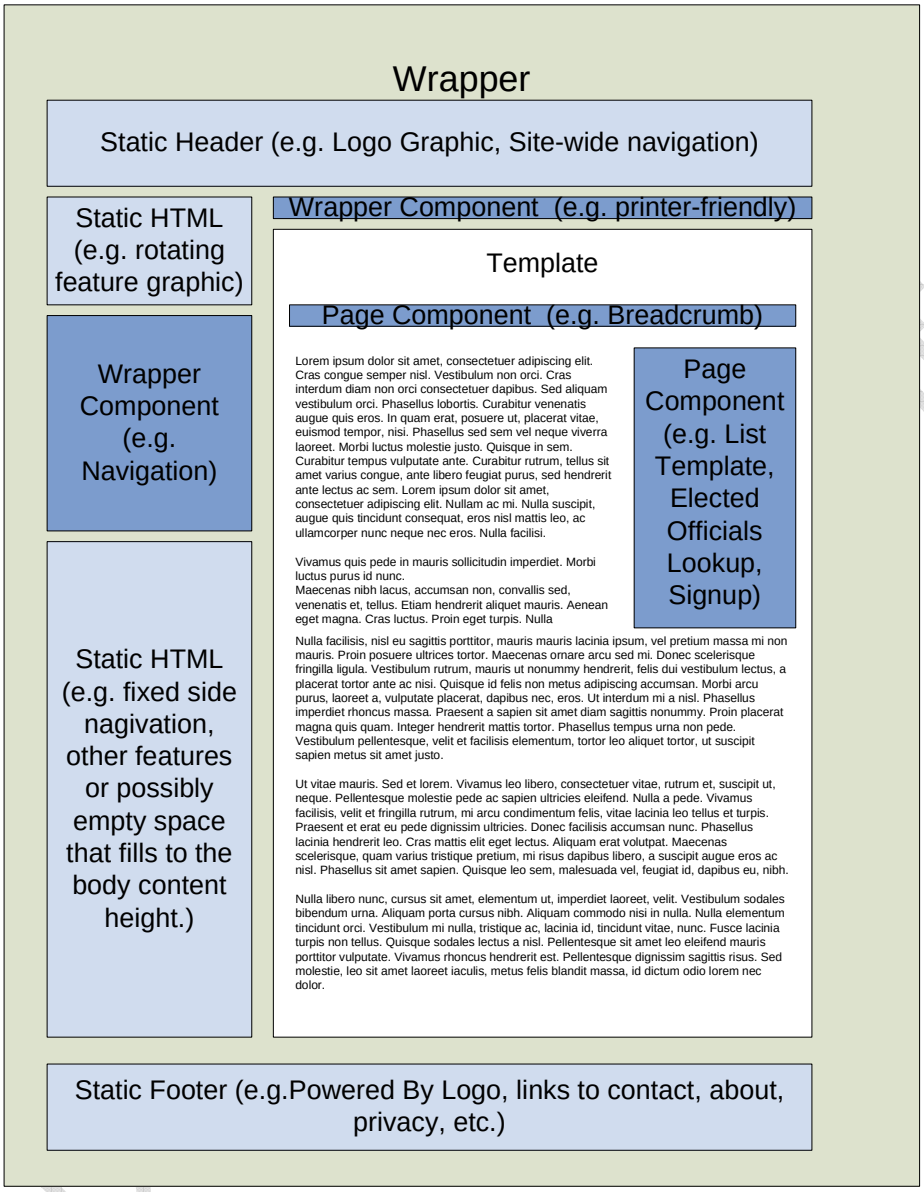


Figure 1. Page Structure

3.1 Wrappers

Wrappers can be shared between WSM and the other GetActive modules. They define the overall structure of pages. Typically, wrappers include header branding, site-wide navigation, sectional navigation (often found on the side of the pages), and the footer.

3.1.1 Default Wrappers

In WSM, a wrapper can be assigned to a folder as the default wrapper. Any page in that folder will use the default wrapper unless specifically configured to use a non-default wrapper. Folder can also choose to inherit the setting of their parent folder. The figure below illustrates the folder properties page where the default wrapper is configured.

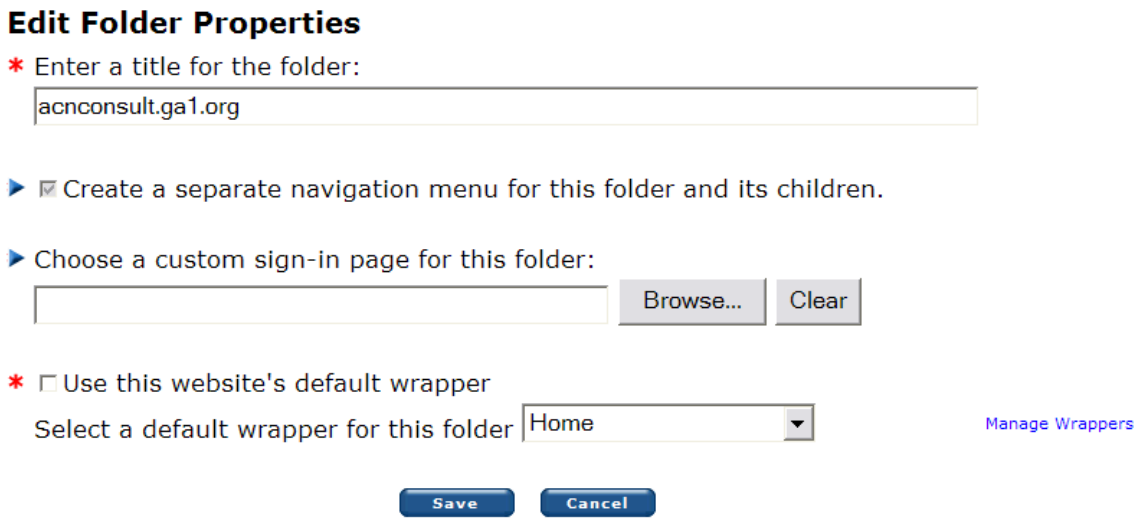


Figure 2. Setting the default wrapper for a folder

3.1.2 Wrapper Tags

There are three wrapper tags that belong in every wrapper:

1. %page_title% which dynamically inserts the title in the HEAD of the HTML document
2. %page_content% which pulls the content of the page body
3. %page_powered_by% which pulls the GetActive branding image or text into the bottom of the page.

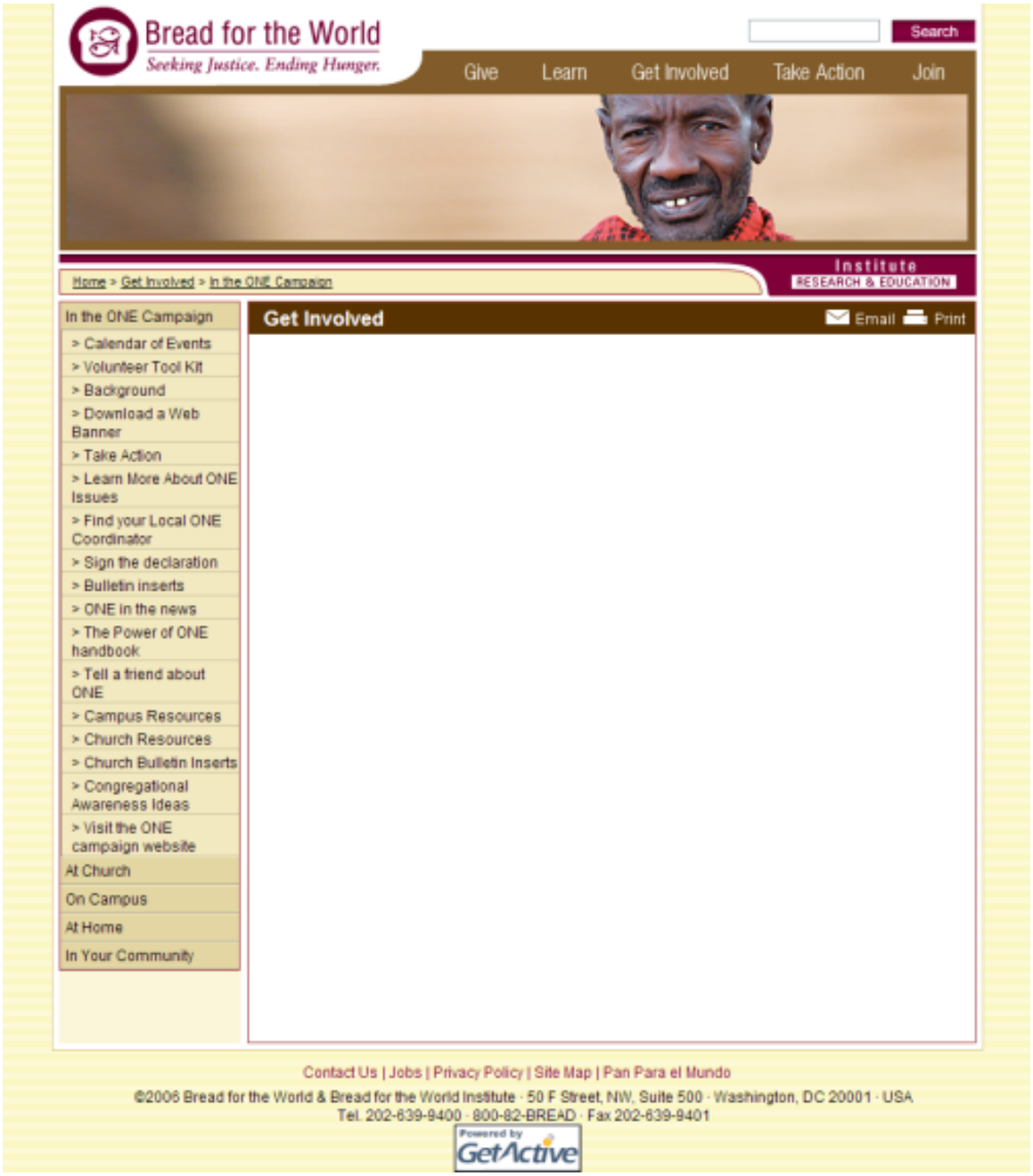
Wrappers can optionally include the following additional tags:

1. %page_search% which dynamically inserts a search form
2. %tellafriend_url% which inserts the URL of the tell-a-friend page
3. %center_url% which inserts the URL to the center
4. %sendtofriend% which inserts the URL to the send-to-friend page
5. %breadcrumb% which inserts the breadcrumb navigation

6. %sectionnav% which inserts the section navigation typically found on the left-hand side of Web Pages.

3.1.3 Example Wrapper

Here is an example of a web page with an empty body. Everything shown here is contained in the wrapper:



3.1.4 Wrapper Design

Since GA Wrappers are used to wrap WSM pages and non-WSM pages (e.g. advocacy and fundraising campaigns) and WSM pages and non-WSM pages are served from different URLs (e.g. www.getactive.com and online.getactive.com) , absolute references must be used for external assets such as images, CSS style sheets and Javascript. For the same reason, it considered a best practice to put as much as possible of the page design into the wrapper.

Oftentimes, it will be necessary to have multiple wrappers. In particular, the home page will often need a separate wrapper due to its unique design nature.

3.1.5 Search in Wrapper

The default HTML for a search box can be added by using the %page_search% tag or custom HTML for a search form can be embedded into the wrapper. The form action should be "search.jsp" or "/search.jsp" and the text input field for the query should be named "query". It is recommended to use the GET form method so that the search results page can be bookmarked and emailed.

Note: The search.jsp can be made to live in any folder. You could make the target of the search form be /search.jsp or search.jsp or any other existing folder path. The search page will use the default wrapper for the appropriate folder, so if the URL for the search page is /search.jsp, the default wrapper for the root folder will be used. This allows the designer to have the search results use the same look-and-feel as other pages in the same folder. This is particularly helpful when implementing search pages that perform a restricted search, such as a Press Release search page that looks like the Press Release section and a News search that looks like the News section.

3.1.6 Printer-friendly Wrapper

Sites that support printer-friendly pages will need to define a "printer-friendly wrapper." Typically, printer-friendly wrappers have minimal branding, the page body, and possibly some breadcrumb navigation.

3.2 Content Types

Content Types are the core elements of WSM. Content Types extend one of three basic types of managed content: Web Pages, Images, and Files. Web Pages are for content items that will get displayed in a browser. Images are for GIF, JPEG and PNG images that will be used in Web Pages. Files are for everything else including CSS, JS, Word, PPT, SWF and MP3.

A Content Type consists of a set of properties as defined by the Content Type data entry form, and one or more display templates. Content Type properties are the fields that store information about instances of the type. For example, a Press Release content type might have the following properties:

- Title
- URL

- Byline
- Release Date
- Contact E-mail address

There are some fields that all content types share. Those include:

- Title
- URL (also called filename)
- Author
- First Publication Date
- Last Publication Date (meaning “most recent”)
- Description (also used in search results)
- Body (Web Pages only)
- File (Images and Files only)

3.2.1 Developing a Site Content Architecture

The first step in developing a content architecture for a WSM project is to determine what content types are required, what properties they need and what display templates are necessary. For example, to build a site for Dinky Donuts, you might need the following content types:

1. Donuts: one instance per type of donut sold
2. Stores: one instance per store
3. Coupons: one per coupon

The Donuts content type will have the following properties:

Property	Example Value
Name	Jelly Donut
URL	Jelly-donut
isFIlled	True
isFrosted	False
Calories	834
Description	Cherry jelly-filled jubilee

Table 1. Donuts Properties

For display, a single display template for displaying information about one donut and a list display template for displaying information about a list of donuts are needed.

The Stores content type will have these properties:

Property	Example Value
Name	East Main Street
URL	East-main
Opens	6
Closes	23

Phone	555-1212
Address	123 East Main St. Anytown, USA[1]

Table 2. Stores Properties

Again, one single display template and one list display template will be required for stores.

Finally, the coupon content type might look like this:

Property	Example Value
Name	January Buy 2 dozen get a free coffee
URL	January-buy-24-get-coffee
Coupon File	/coupons/January-buy-24-get-coffee.pdf
Valid From	1/1/2006
Valid Until	1/31/2006

Table 3. Coupon Properties

As with the other content types, we need a single and a list display template for coupons.

3.2.2 Content Type Creation

Content Types are created using the "New Type" link in the content type area of the WSM management console.

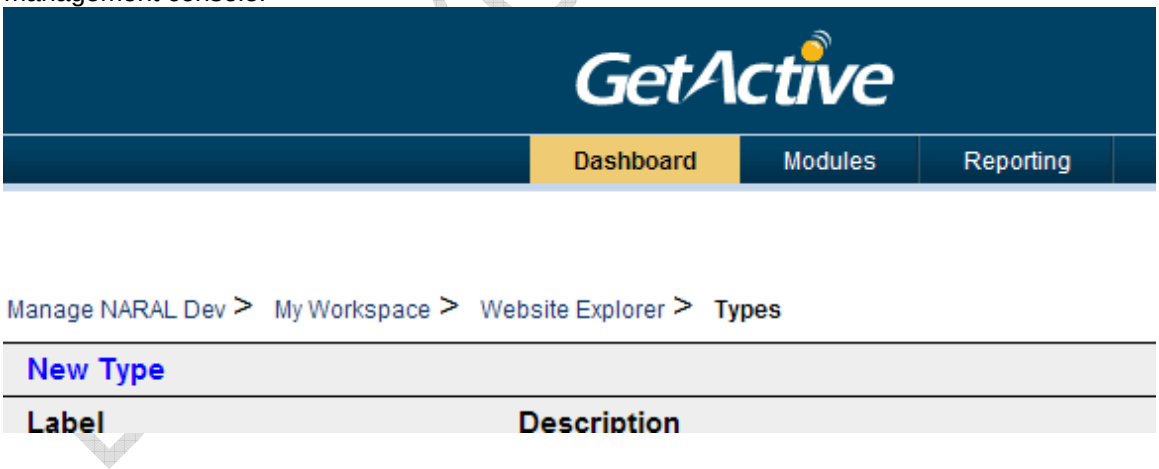


Figure 3. New type link

[1] In a real implementation, it would be best to use separate street, town, state and country fields.

Content types extend one of three basic types: Web page, Image and File. Web pages are HTML pages that can be viewed in a web browser. Images are GIF, JPEG and PNG images that can be used in Web pages. Files are everything else including PDFs, office documents, audio files, etc.

Each content type can choose which wizard steps to include. For Web pages, the possible steps are the following:

- Enter Item Properties (Required)
- Enter Body Content using HTML Editor
- Preview Published Item
- Schedule Publication and Expiration
- Submit for Review and Publication (Required)

The item properties step is where you edit the information about a particular item of the content type. The body content step is where you enter the body of the item using the WYSIWYG editor. You can put a smaller version of the editor on the properties step and not use the body step. The preview step allows authors to see the item as it will appear on the live site once it is published. Authors can schedule the publication and expiration dates and times on the schedule publication step. If you just want items to go live when they are approved, you can not include this step. The final step is where authors submit content to administrators and where administrators approve (or reject) submitted items.

3.2.2.1 Content Type Field Definition

Once you have created the content type, you need to define the extra properties of the content type. You can do this by clicking on the "Edit" link of the Fields section:

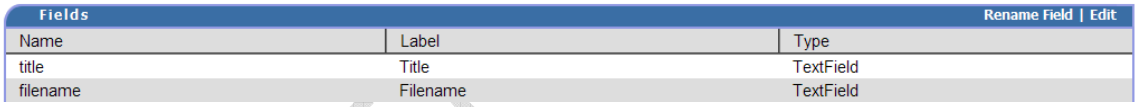


Figure 4. Content Type fields list

The fields or properties editor is a WSYIWYG editor where you manage the HTML form that is used by authors to create and edit items of the content type.

You can add the following kinds of properties to a type:

- Text Field
- Text Box
- Rich Text Box
- Menu[1]

[1] There are some magic widgets that you can use to provide a richer user-experience. Menu properties that are backed by related items can use a related item chooser or a related image chooser (if the items are images.) Menus that are backed by categories can use a category chooser.

To get a related item chooser, you need to wrap the <SELECT> tag with the following HTML:

```
<DIV class=related-item-chooser><SELECT title="Documentation" name=documentation validation="type:string" model="items;filterID=27275284"></SELECT></DIV>
```

- Radio button
- Check box
- File

Fields can be made required or optional (although the basic field title is required.)

Some of these property types can be further constrained. A text field can be configured as “text”, “integer”, “number”, “date”, or “email”. Menus can be single- or multi-select. The value options can be driven by a list of categories, a set of content items, a set of authors, a set of templates, or a set of hand-coded values. There are additional properties of each field type that can also be managed using the field’s administration dialog.

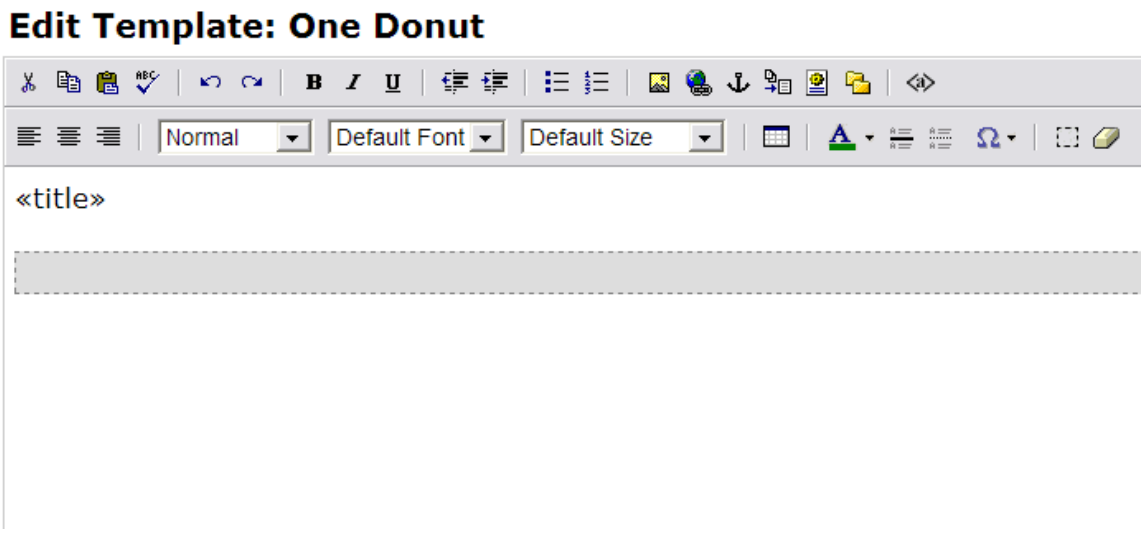
3.2.3 Display Templates

Content type items are rendered as HTML through the use of templates. Templates come in two flavors: single display templates and list display templates. Single display templates are used to render a single item, such as a press release, or a donut. List display templates, are used to show a list of items. This might be used to show a list of new press releases on the press section index page.

3.2.3.1 Single Display Templates

Single display templates control the layout of web pages that contain one content item. The single template is what is used to render the page content as specified by the %page_content% tag in the wrapper.

Continuing with the donut example, the single display template for a donut might look like this:



For related image choosers, use “image-chooser” as the class name for the DIV tag. For category choosers, the DIV class name should be “category-chooser”.

Figure 5. Donut single display template

Or, in HTML:

```
<div id="title"><span class="templateVariable"
contentEditable="false">«title»</SPAN></div>

<p class="templateBody"></p>
```

Creating the jelly donut is done by filling out the data input form:

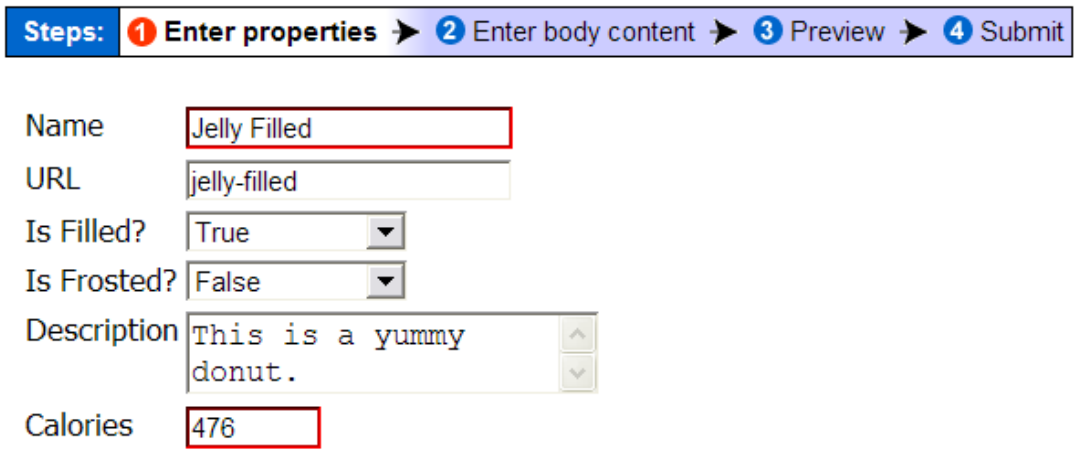


Figure 6. Creating the jelly donut

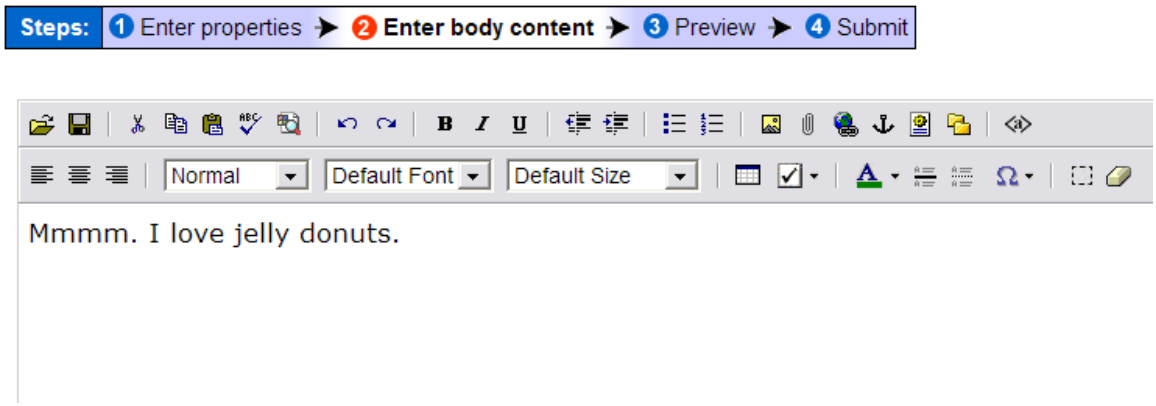


Figure 7. Jelly donut body entry

When the jelly donut is rendered in that single display template, the result looks like this:

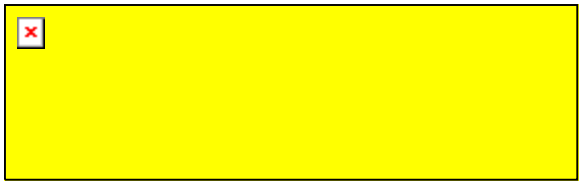


Figure 8. Jelly donut rendered in display template

3.2.3.2 List Templates

List templates are most commonly used with the List Component. A list template controls the layout of a collection of content items of one content type. A list of donut shops might look like this:

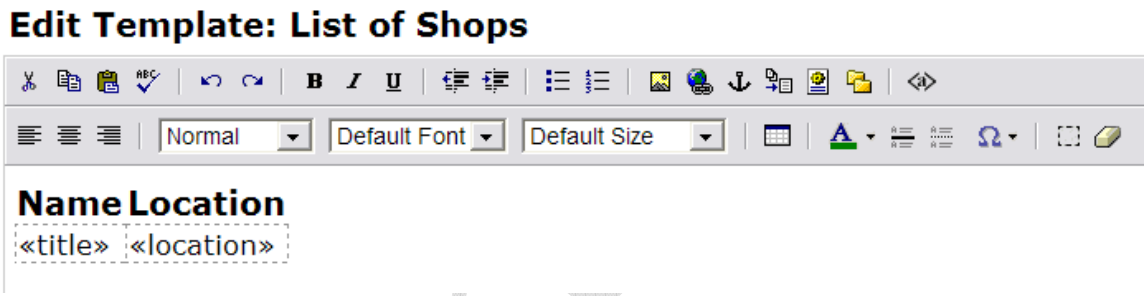


Figure 9. Donut shop list template

Here is the template in HTML:

```
<TABLE>
<THEAD>
 <TR>
  <TH>Name</TH>
  <TH>Location</TH>
 </TR>
</THEAD>
<TBODY>
 <TR class=templateRepeat>
  <TD><SPAN class=templateVariable contentEditable=false>«title»</SPAN>
  </TD>
  <TD><SPAN class=templateVariable contentEditable=false>«location»</SPAN>
  </TD>
 </TR>
</TBODY>
</TABLE>
```

This will look like this when rendered as HTML:

Name	Location
West Main Street	123 West Main St. Anytown, USA
East Main Street	123 East Main St. Anytown, USA

Figure 10. Screen shot of a list of donut shops

Note: For a full description of the WSM template system, see the online reference page at http://www.frontleaf.com/doc/types/display-templates.html or in Section 11.

When developing a site design, you should pay special attention to lists of content so that you can be certain that your design can be implemented in a list template.

4 Components

Components are useful to include dynamic features in web pages. For example, the donut page shown in Figure 8. Jelly donut rendered in display template, might want to include a list of shops on the page. Since this list would be present for any donut displayed, the list should be put into the single display template for the donut content type. If it was added to the body of the jelly donut page, it would not show up on the page that displayed the old fashioned donut.

Components can be added to Web Pages or Display Templates. In both cases, they are added from the WYSIWYG editor using the “Add Component” toolbar button as shown below:

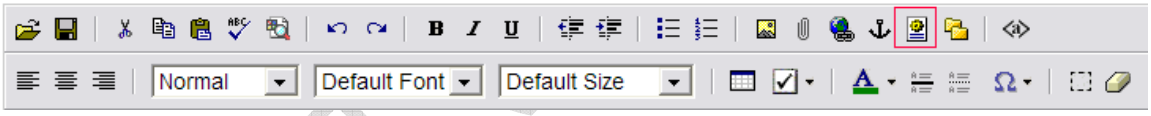


Figure 11. Editor Toolbar with Add Component tool highlighted

WSM supports a variety of components. Currently, the list includes the following:

- Breadcrumb
- E-card
- Elected Officials Lookup
- Filter Criteria
- Include
- List Template
- Login
- Navigation
- Printer-friendly
- Related Links
- Related Links with Display Template
- Sign-up

The style and layout of every component can be controlled by the component style and layout dialog as shown below:

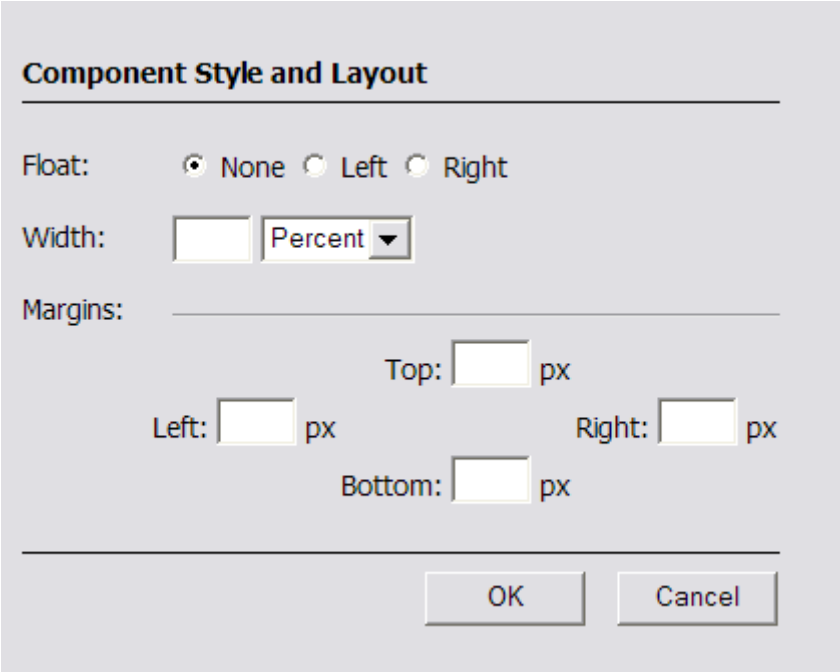


Figure 12. Component Style and Layout Dialog

This dialog controls the overall layout of the component in the web page or display template. Some components have their own administrative dialog boxes where various aspects of the component can be controlled. In many case, this includes the selection of a style sheet from the WSM style sheet manager. Components that support style sheet selection can be controlled at the instance level. That means there can be two differently styled instances of the same type of page component on the same page or in the same template.

One of the most commonly used components is the List Template component. The administrative dialog box for a list template component is shown below:

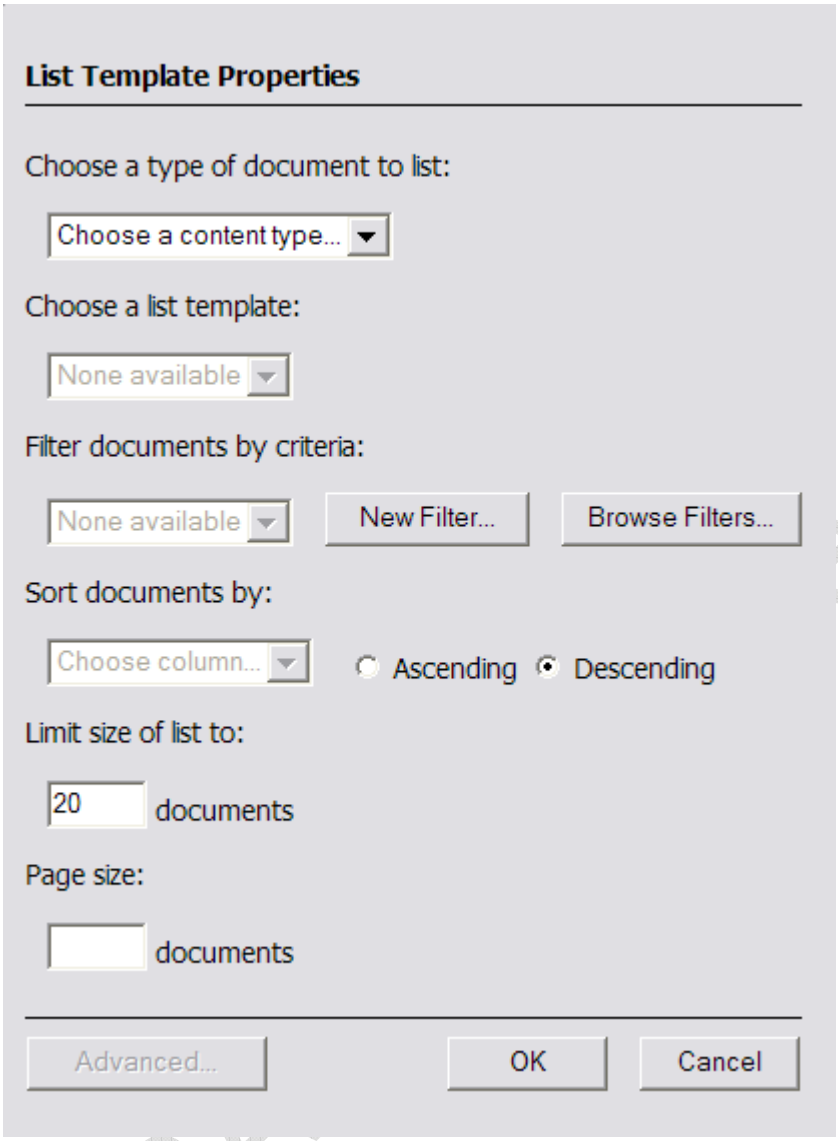


Figure 13. List Template Administrative Dialog

After adding the list template component to the single display template, the HTML looks like this:

```
<div id="title"><span class="templateVariable"
contentEditable="false">«filename»</SPAN></div>

<p class="templateBody"></p>

<DIV class=templateComponent id=templatelist-27579472 contentEditable=false
style="DISPLAY: inline; float:right"></DIV>
```

With the addition of the list of donut shops, the jelly donut page now looks like this:

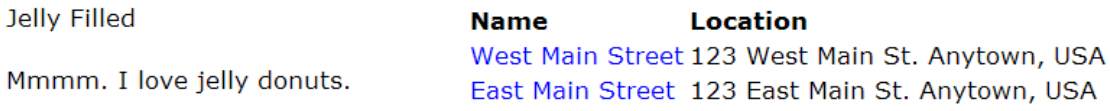


Figure 14. Screen shot of a jelly donut with a list of donut shops

5 Filters

A filter is a collection of rules that are used to select a group of content items. Filter rules can be applied to the following content item characteristics:

- Content Type
- Publication Date
- Folder
- Related Items
- Category

For example, to collect a group of web pages in the Consultants folder that have been published in the past two weeks, configure a filter as shown:

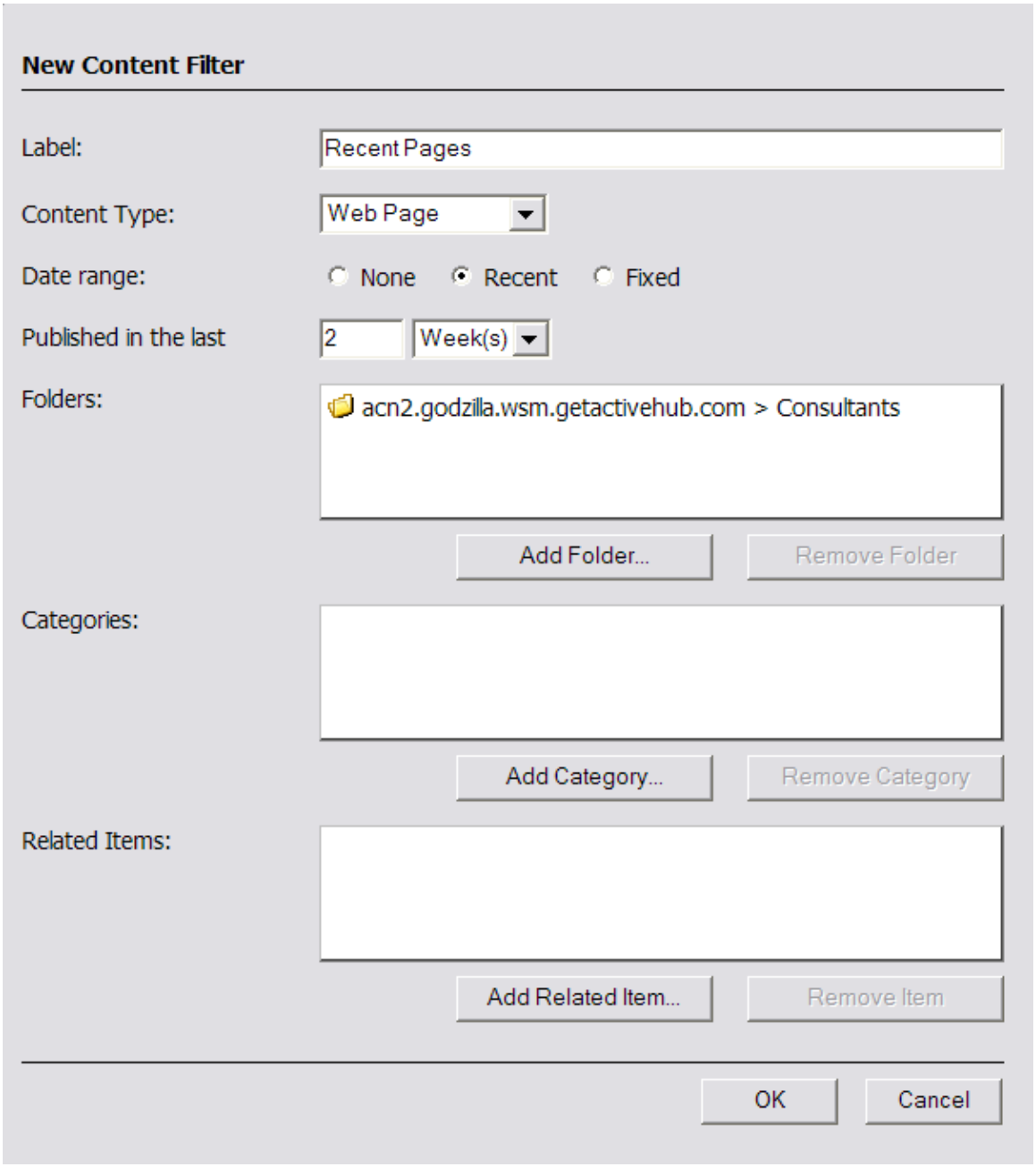


Figure 15. Content Filter of Recent Pages

6 CSS and Style Sheets

Style sheets may be used in three ways in WSM powered sites:

1. external CSS files managed by WSM, referenced by the HTML in the HEAD section in a wrapper, or
2. embedded inline in a STYLE block in the HEAD of a wrapper, or
3. dynamic embedded inline styles created using WSM's native Style Sheet Manager which controls the styling of page components (including the body component.)

6.1 Web Page Structure

There are many elements that can be styled in Web Pages generated by WSM. The body will be enclosed in the following HTML structure:

```
<table id="contentTable" cellpadding="0" cellspacing="0" width="100%">
 <tbody>
  <tr>
   <td id="contentArea">
    <table style="width: 100%;" cellpadding="0" cellspacing="0">
     <tbody>
      <tr>
       <td valign="top">
         <div id="contentDiv">

<!-- Page Body will be inserted here -->

         </div>
       </td>
      </tr>
     </tbody>
    </table>
   </td>
  </tr>
 </tbody>
</table>
```

Given this structure, styles can be defined that apply to contentTable, contentArea and/or contentDiv. Additionally, the WSM components can be styled as defined in Appendix A: WSM Component Styles.

6.2 Default Body Style

If a site does not define a default WSM style sheet for the body component using the WSM Style Manager, then WSM will automatically add the following styling to every Web Page:

```
<style>
  p { font-family:Verdana; } #contentDiv { margin:16px; }
</style>
```

7 Information Architecture Checklist

Describe the steps an IA must go through to build a WSM-based site (with minimal PS support)

- Define Content Types
- Define CT relationships
- Define/Create Categories
- Create CTs
- Defining Wrappers
- Defining Templates
- Defining special requirements

8 Site Design Checklist

Describe the steps a designer must go through to build a WSM-based site (with support from an IA, who may be from PS)

- Create wrappers
- Create templates
- Create folder structure
- Build pages

9 Appendix A: WSM Component Styles

9.1 Advocacy Campaign:

```
<table>
 <thead>
  <tr>
  <th colspan=”2” class=”Title”>Dynamic Title</th>
  </tr>
 </thead>
 <tbody>
<tr>
 <td colspan=”2”><p class=”PromptText”>Dynamic Prompt</p></td>
</tr>
<!-- repeat for each campaign -->
<tr>
 <td><a><img></a></td>
 <td><p class=”Name”><a>Campaign Name</a></td>
</tr>
<!-- end repeat -->
<!-- optional link -->
<tr>
 <td> </td>
 <td><p class=”ViewAll floatright”><a>View all campaigns</a></td>
</tr>
 </tbody>
</table>
```

9.2 Breadcrumb:

```
<span class=”breadcrumbComponent”>
 Dynamic folder navigation path
 <span class=”breadcrumbseparator”>dynamic separator</span>  
 <span class=”breadcrumb breadcrumbitem”>dynamic page title</span>
</span>
```

9.3 Ecard:

```
<span class=”ecard”>
<form>
<table class=”ecardTable”>
 <tr class=”ecardRow”>
  <td class=”ecardField”><span id=”ecardNameLabel">Recipient Name:</span></td>
  <td class="ecardValue"><span id="ecardNameValue"><input
name="name"></span></td>
 </tr>
 <tr class="ecardRow">
  <td class="ecardField"><span id="ecardSubjectLabel">Email Subject:</span></td>
  <td class="ecardValue"><span id="ecardSubjectValue"><input
name="subject"></span></td>
 </tr>
 <tr class="ecardRow">
  <td class="ecardField"><span id="ecardEmailLabel">Recipient Email
Address:</span></td>
```

```
  <td class="ecardValue"><span id="ecardEmailValue"><input
name="email"></span></td>
 </tr>
 <tr class="ecardRow">
  <td class="ecardField"><span id="ecardFromLabel">Your Email
Address:</span></td>
  <td class="ecardValue"><span id="ecardFromValue"><input name="from"
value="<%=email%>"></span></td>
 </tr>
 <tr class="ecardRow">
  <td class="ecardField"><span id="ecardNoteLabel">Personal Message to
Recipient:</span></td>
  <td class="ecardValue"><span id="ecardNoteValue"><textarea
name="note"></textarea></span></td>
 </tr>
 <tr class="ecardRow">
  <td class="ecardButton" colspan="2">
    <input type="submit" id="previewButton" onclick="return ecardDoPreview();"
value="Preview">
    <input type="submit" id="sendButton" onclick="return ecardDoSend();"
value="Send">
  </td>
 </tr>
</table>
</form>
</span>
```

9.4 Elected Officials Lookup:

```
<table>
 <thead>
  <tr>
  <th colspan=”2” class=”Title”>Dynamic Title</th>
  </tr>
 </thead>
 <tbody>
<tr>
 <td colspan=”2”><p class=”PromptText”>Dynamic Prompt</p></td>
</tr>
<tr>
 <td colspan="2">
   <form name="eo_lookup_<%=pageletID%>" method="post" action="<%=lookupURL%>">
    <input name="zip_search_p" type="hidden" value="1" />
    <input name="zip_five" class="PromptText" type="text" size="5" maxlength="5"
/>&#8212;
    <input name="zip_four" class="PromptText" type="text" size="4" maxlength="4"
/>
    <input name="go" class="PromptText" type="submit"
value="<%=buttonText%>"
<input name="submit" type="image">
    </form>
   </td>
  </tr>
</tbody>
</table>
```

9.5 Filter Criteria:

```
<form action="<%=url%>" style="display:inline">
<table class="filterCriteria">
<tr>
  <td class="filterLabel"><%=column.getLabel()%></td>
<td class="filterSelector">
<!-- loop for each option -->
    <select name="optionName">
        <option>Option Label
</select>
<!-- end loop for each option -->
</td>
 </tr>
<!-- if a date field: -->
<tr>
  <td class="filterLabel">First published between:</td>
  <td class="filterSelector">

<input type="hidden" name="startDate">

    <select name="startMonth">
    <% for (int i = 0; i < months.length; i++) { %>
    <option value="<%=i + 1%>"><%=months[i]%>
    <% } %>
    </select>
    <select name="startYear">
    <option value="2004">2004
    <option value="2005">2005
    <option value="2006">2006
    <option value="2007">2007
    <option value="2008">2008
    </select>

    and

<input type="hidden" name="endDate">

    <select name="endMonth">
    <% for (int i = 0; i < months.length; i++) { %>
 <option <%=(i == 11) ? "selected" : ""%> value="<%=i + 1%>"><%=months[i]%>
    <% } %>
    </select>
    <select name="endYear">
    <option value="2004">2004
    <option value="2005">2005
    <option value="2006" selected>2006
    <option value="2007">2007
    <option value="2008">2008
    </select>
  </td>
</tr>
<!-- end if -->
<tr>
  <td class="filterLabel"> </td>
  <td class="filterSelector">
    <input type="submit" value="Submit">
  </td>
</tr>

</table>
```

```
</form>
</table>
```

9.6 Fundraising Campaigns:

```
<table>
 <thead>
  <tr>
  <th colspan=”2” class=”Title”>Dynamic Title</th>
  </tr>
 </thead>
 <tbody>
<tr>
 <td colspan=”2”><p class=”PromptText”>Dynamic Prompt</p></td>
</tr>
<!-- repeat for each campaign -->
<tr>
  <td></td>
  <td><p class="Name"><a href="<%=campaignURL%>"><%=campaignTitle%></a></p>
      <p class="Expiration">Donate before: DynamicExpirationDate </p></td>
 </tr>
<!-- end repeat -->
<!-- optional link -->
<tr>
 <td> </td>
 <td><p class=”ViewAll floatright”><a>View all campaigns</a></td>
</tr>
 </tbody>
</table>
```

9.7 Include:

```
<span class=”includeDocument”>
DynamicPageHere
</span>
```

9.8 List Template:

```
N/A
```

9.9 Login:

```
 <form name="signup_<%=pageletID%>" method="post" action="<%=loginURL%>">
 <input name="domain" type="hidden" value="<%=domain.getCode()%>" />
 <input name="return_url" type="hidden" value="<%=postLoginURL%>" />

  <table>
    <thead>
      <tr>
        <th colspan="2" class="Title">DynamicTitle</th>
      </tr>
    </thead>
    <tbody>
    <tr>
```

```
      <td colspan="2">
        <p class="PromptText"><%=prompt%></p>
      </td>
    </tr>
    <tr>
      <td><p class="Name">E-mail:</p></td>
      <td nowrap="nowrap">
        <input name="email" class="Name" type="text" size="emailBoxSize"
maxlength="100" />
      </td>
    </tr>
    <tr>
      <td><p class="Name">Password:</p></td>
      <td>
        <input name="password" class="Name" type="password" size="15"
maxlength="50" />
        <input name="go" class="Name" type="submit" value="<%=buttonText%>" />
        <input name="submit" type="image" border="0" alt="altText%>">
      </td>
    </tr>
    <tr>
      <td colspan="2">
        <label class="RememberMe">
          <input name="remember_me_p" class="Name" type="checkbox" value="t" />
          Remember Me?
        </label>
      </td>
    </tr>
    <tr>
      <td colspan="2">
        <p class="Link">&raquo; <a
href="<%=signupURL%>"><%=signupLinkText%></a></p>
        <p class="Link">&raquo; <a>Dynamic Forget Password Text</a></p>
      </td>
    </tr>
    <tr>
      <td colspan="2">
        <p class="Name">Dynamic Logged In Text</p>
      </td>
    </tr>
    <tr>
      <td> </td>
      <td><p class="PromptText">&raquo; <a class="Name"
href="<%=smpURL%>">Go to your subscription management page</a></p>
      </td>
    </tr>
    <tr>
      <td> </td>
      <td><p class="PromptText">&raquo; <a class="Name">Logout</a></p>
      </td>
    </tr>
  </tbody>
 </table>
</form>
```

9.10 Navigation:

```
<div id=”navigation-ID” class=”navigationMenu”>
```

```
<!-- some combination of the following -->
<div class=”navigationItemLabel”>Top-level Dynamic Label Text 1</div>
<div class=”navigationItem”>Top-level Dynamic Link Text 2</div>
<div class=”navigationItem”>Top-level Dynamic Link Text 3</div>
 <div class=”navigationSubMenu”>
  <div class=”navigationItem”>second-level Dynamic Link Text 1</div>
 </div>
</div>
```

9.11 Pagination Control:

```
<table class=”paginator”>
 <tr>
<td>Items <b>1st - N-th</b> of TOTAL   </td>
<td><a>Previous</a></td>
<td><a>1</a>
.
.
.
<td><a>N</a></td>
<td><a>Next</a></td>
 </tr>
</table>
```

9.12 Poll Results:

```
<div class="tableArea"L
 <div class="tableDiv">
   <table class="resultsTable">
    <thead>
    <tr>
     <th colspan="2" class="resultsTitle">Dynamic Title</th>
     <th class="resultsHeaderCell2">Dynamic Header</th>
     <th class="resultsHeaderCell2">Dynamic Header</th>
    </tr>
</thead>
<!-- repeat for each field in poll -->
    <thead>
    <tr>
     <th colspan="4" class="resultsHeaderCell">Label</th>
    </tr>
    </thead>
    <tbody>
    <tr>
     <td class="labelCell">Label</td>
     <td class="percentBars">
      <div class="resultBar-N" style="width:dynamicWidth”/>
     </td>
     <td class="numberFormat">N%</td>
     <td class="numberFormat"># of votes</td>
    </tr>
<!-- end repeat for each field in poll -->
    <tr>
     <td colspan="4" class="respondantTotal">Dynamic “Total” Label Dynamic
Count</td>
    </tr>
    </tbody>
```

```
   </table>
 </div>
</div>
```

9.13 Printer Friendly:

```
<a class="printerFriendlyLink">Dynamic Label</a>
```

Or

```
<a><img class="printerFriendlyImage">" border="0"/></a>
```

9.14 Related Links:

```
<span class="relatedLinks">
<table cellspacing=0 class="relatedLinksBoundingTable" cellpadding="0">
  <thead>
    <tr>
      <th colspan="2" class="relatedLinksTitle">Dynamic Title</th>
    </tr>
  </thead>
<tbody>
<tr>
  <td colspan="2">
   <table class="relatedLinksRows" cellpadding="0" cellspacing="0">
<!-- Repeat -->
     <tr><td class="relatedLinksRow"><p class="relatedLinksLink"><a> Dynamic
Title</a></p></td></tr>
   </table>
  </td>
</tr>
</tbody>
<tfoot class="relatedLinksFooter">
<tr>
<td colspan="2">
  <table style="width:100%" cellpadding="0" cellspacing="0">
    <tr>
      <td style="width:50%;text-align:left;border:0 none;padding:0">
       <a>Alerts</a> </td>
      <td style="width:50%;text-align:right;border:0 none;padding:0">
        
           <a>Archive</a>
      </td>
    </tr>
    </table>
  </td>
</tr>
</tfoot>
</table>
</span>
```

9.15 Send Page to Friend:

```
<span class="sendToFriend">
<a>Dynamic Title</a>
OR
<a><img></a>
</span>
```

9.16 Sign-Up:

```
  <table>
    <thead>
      <tr>
        <th colspan="2" class="Title">Dynamic Title</th>
```

```
      </tr>
    </thead>
    <tbody>
      <tr>
        <td colspan="2">
          <form name="signup_ID">
            <label class="Name">E-mail:
            <input name="email" class="Name" type="text" size="emailBoxSize"
maxlength="100" />
            </label>
            <input name="go" class="Name" type="submit" value="Dynamic
buttonText" />
            <input name="submit" type="image" border="0" />
          </form>
        </td>
      </tr>
      <tr>
        <td> </td>
        <td><p class="AlreadyMember">&raquo; <a class="Name">Dynamic Login
Text</a></p></td>
      </tr>
      <tr>
        <td colspan="2">
          <p class="Name">Dynamic Logged in Text</p>
        </td>
      </tr>
      <tr>
        <td> </td>
        <td><p class="AlreadyMember">&raquo; <a class="Name">Go to your
subscription management page</a></p></td>
      </tr>
      <tr>
        <td> </td>
        <td><p class="PromptText">&raquo; <a
class="Name">Logout</a></p></td>
      </tr>
    </tbody>
  </table>
```

10 Glossary

Term	Definition
Base Property	A property of a Content Type that WSM manages for all Content Types; examples include: Title, Body, Description, First Publication Date, Last Publication Date, Author, etc.
Breadcrumb Navigation	Set of links that presented to users on a Web Page to provide context for the current page as well as access to the ancestor folders
Category	A structure for organizing, typing, and representing a dimension of a Content Type
Category Chooser	A Data Entry Form widget that supports selection of a Category
Checkbox	A form field of type "checkbox"
Component	A reusable object that performs a specific function and which can be included in a Web Page
Content Item	instantiation of a Content Type with stored values for the Content Type's properties
Content Repository	The collection of all stored content in WSM
Content Type	set of fields and associated templates that define one type of managed content
Data Entry Form	HTML Form used for setting/editing the properties of a Content Item
Display Template	HTML and embedded logic that controls the consistent formatting and display of structured content
E-Card	Application that supports end-user configuration of an email message that contains an image
Extended Property	A property of a Content Type that was created by a Website Manager or Publisher; examples might include: byline for a Press Release, location for an office, etc.
File	basic type of content; usually downloaded by users and view in a desktop application e.g. Adobe Acrobat, Microsoft Word.
Filter	A method of isolating a selection of items from a longer list.
Filter Criteria	A set of folder, category, content type, related item and publication date settings that specify a subset of all Content Items in the Content Repository
Folder	Container of Content Type Instances

Form	A set of data entry fields on a web page that are processed on the server
Form Notification Recipient	WSM administrator configured to receive an email notification each time a particular form is submitted.
Gallery Mode	Windows explorer mode that shows folder contents as set of thumbnail icons with a configurable hover menu
Gallery View	See Gallery Mode
HTML Editor	What-You-See-Is-What-You-Get (WYSIWYG) editor that supports the editing HTML without specific knowledge. HTML Editors are used for editing Web Page body content, rich text properties of Content Type Instances, Data Entry forms, and Display Templates. HTML Editors can have different toolbars depending on context and administrative configuration choices.
HTML Mode	The normal view in an HTML Editor
Image	basic type of content; GIF, JPEG or PNG image that can be displayed in a browser alone or as part of a Web Page
Include	Component that includes a reusable chuck of HTML
List Display Template	Display template that iterates over a list of Content Items
List Mode	Windows explorer mode that shows folder contents as a list
List View	See List Mode
Members Only Folder	A folder that contains Content Items that can only be viewed by authenticated Members of a website
Navigation	Visual elements of a web page that answer the questions: Where am I? Where can I go? How will I get there? How can I get back to where I once was?
Page Body	The property of a Web Page that contains the main HTML typically used when displaying Content Items
Pagination	The separating of large lists of items into distinct numbered web pages with inter-page navigation
Permission	the ability of the users affected to view or make changes
Poll	An HTML form that consists of only category-backed Select Lists; also has form/results.jsp set as the finish page
Preview	View a Web Page as it will appear live on the Web Site once it has been published

Property	A field of a Content Type; either a Base Property or an Extended Property
Radio Button	A form field of type "radio"
Related Image Chooser	A Data Entry Form widget that supports selection of a related Image
Related Item Chooser	A Data Entry Form widget that supports selection of a related Content Item
Repeat Block	List Display Template
Role	a named list or group of priviledges that are collected together and granted to users.
Scheduled Expiration	The future date and time at which a live Content Item will be automatically expired
Scheduled Publication	The future date and time at which a Content Item will be automatically published (if the item is approved by that time)
Section Navigation	Set of links that presented to users in a section of a website to provide context for the current page as well as access to other sections
Select List	A form field of type "select"
Single Display Template	Display template that renders a single Content Item
Slideshow	Application that supports the display of a set of images
Source Mode	An alternate view in an HTML Editor that shows the HTML code instead of the rendered HTML. This is useful for performing advanced HTML markup that is not supported by the UI tools in the HTML Editor
Status	The current state of a Content Item; one of Authoring In Progress, Submitted for Approval, Approved for Publication, Live
Stylesheet	set of rules (called patterns) modifying the default appearance of HTML content
Stylesheet Manager	The WSM tool for managing component-level Style Sheets
Template	see Display Template
Template Editor	HTML Editor that is used to edit Display Templates
Template Field	A Content Type Property available for use in a Display Template
Text Area	A form field of type "textarea"
Text Field	A form field of type "text"

Toolbar	Set of icons that represent tools, menu items and actions; see Website Explorer and HTML Editor
Web Page	basic type of content; rendered as HTML in a browser
Website Author	Role that allows a user to create and edit Content Items, but not to publish them
Website Explorer	Tree-based graphical browser of the Content Repository; contains a toolbar that allows a user to add new content items; copy/move/delete existing content items; manage navigation, folder properties, permissions; and toggle the view between List and Gallery modes
Website Manager	Role that allows a user to perform all WSM tasks in a given folder
Website Publisher	Same as Website Manager plus enables a user to receive notifications whenever a Content Item is submitted for approval
Workflow	a collection of steps that defines the paths that can be taken to publish an item
Workflow Instance	a set of ordered Workflow Tasks
Workflow Task	one step in a series of steps that must be completed for a Content Item in a Workflow to get published
Workspace	The main working area of WSM; contains links to website explorer, categories, workflows, components, content types, wrappers and tools for each Web Site that the user can work on; also contains list of tasks and recently accessed documents
Wrapper	HTML that defines the overall structure of a Web Page; typically includes main navigation, side navigation and footer; "wraps" body
Wrapper Components	Components that can be referenced in a Wrapper: Breadcrumb Navigation, Site Navigation, Send to Friend and Printer Friendly are the four wrapper components
WSM Stylesheet	A component-level stylesheet that can be used to apply formating to one or more instances of a Component
WYSIWYG Editor	See HTML Editor

11 Display Template Reference

11.1 Overview

A display template provides a customizable view of a content item's properties. The template defines a layout into which the system dynamically inserts content item properties. This helps ensure consistency across all pages of a web site.

There are two basic types of display templates:

1. **Single** display templates are dedicated to one content item only. A simple display template for a single content item might include a title, publication date and the body content of a page.

2. **List** display templates contain a block of markup that repeats for each one of a set of selected content items, as defined by an associated filter. A simple list template would display the title of each item as a hyperlink to the full page, along with a short description or teaser.

   List templates are commonly used on index pages to provide an overview of a particular section of a web site. They are also used in callouts and sidebars to provide links to related content.

You write a display template using standard HTML along with a few custom elements to flag the location of content properties as well as repeating and conditional blocks of markup code. This document outlines the use of those elements.

11.2 Namespaces

Custom elements are declared in a separate namespace from standard HTML elements. The letter `t` is the default prefix for this namespace. When editing a template in a browser, the `HTML` element should declare the custom namespace:

```
<html xmlns:t>
```

The template parser API supports specifying an alternate namespace prefix if necessary.

11.3 Properties

To insert a dynamic content property in a display template, you use the `value` element with an `ID` attribute corresponding to the property name:

```
<h1><t:value id="title">My Content Item</t:value></h1>
```

The contents of the `value` element can be any sample text you wish to enter as a placeholder while designing the template. This text is replaced with an actual property value when a page is published using the template.

The `value` element itself does not appear in the output HTML. You can wrap the element in a SPAN or other markup if you wish to add styling using the `class` or `style` attributes.

Note: For legacy purposes the system also recognizes the following syntax:

```
<h1><span class="templateVariable">«title»</span></h1>
```

This syntax was replaced to provide more flexiblity with respect to styling the output text, and to avoid the need to always wrap dynamic text in a SPAN element.

11.4 Properties in Attributes

It is not possible to use the `value` tag to insert a dynamic property in an element attribute, such as the `href` of a hyperlink.

In this case you can use an alternate syntax to specify a dynamic property:

```
Click <a href="${url}">here</a>.
```

You can also use this syntax elsewhere in a display template, but it discouraged because it may confuse visual template editing tools.

11.5 Request Parameters

The request parameters are available in a display template:

```
<h1><span class="templateVariable">«param.query»</span></h1>
```

11.6 Base Properties

The following content properties are available in a display template:

itemID The unique system ID of the document.

title The title of the document as displayed in the page header, in sitewide search results and in other links to the document.

description The description of the document as displayed in sitewide search results.

owner The full name of the primary author of the document.

filename The URL name of the item, such as *`mypage.html`*.

body The body text of an HTML document.

url The full URL path of the item, such as */myfolder/mysubfolder/mypage.html*.

folderID A breadcrumb trail to the folder containing the document.

fileIcon A 16x16 pixel icon corresponding to the document type, such as web page, Microsoft Word document or PDF document.

firstPublicationDate The date when the document was first published.

lastPublicationDate The date when the document last published or re-published.

publicationDate The date when the document is next scheduled for publication.

modifiedDate The date when the draft version of the document was last modified. In most cases this property is only relevant for authors and administrators.

modifiedBy The name of the last user to modify the document.

creationDate The date when the document was first created.

createdBy The name of the user who created the document.

type A descriptive label for the type of the document.

size The size of the document body.

Other available properties are determined by the custom type of the item.

11.7 Body Content

The body content of an item can be inserted into a display template in one of two ways:

1. As a standard content property using the `value` element as described above.
2. Using the special `templateBody` class name to mark placeholder text for replacement with actual body text.

Here is an example of placing content using the `templateBody` class name:

```
<p class="templateBody">Lorem ipsum dolor sit amet, consectetuer
adipiscing elit. Praesent molestie pretium purus. Praesent id
nibh. Pellentesque vel arcu. Vivamus non nisl. Nullam interdum justo
quis sem. Donec turpis tellus, blandit dignissim, egestas nec,
vulputate sit amet, eros.</p>
```

This syntax is intended to provide more flexibility in formatting parts of body content, as well as in the placement of callouts. For example, you might want to display the first paragraph of a template in a larger font, or insert an ad banner between the second and third paragraphs. Note that the system does not actually offer this functionality yet but it is intended for a future release.

11.8 Format Attributes

For date properties, you can use the special `format` attribute to specify the format of the date value when the system prints it on the published page:

```
<p>
  Published on:
  <t:value id="publicationDate" format="3">Jan 1, 2005</t:value>
</p>
```

Valid options for the `format` attribute for date values are:

0 10 Nov 04 03:50 PM PST

1 Nov 10, 2004 03:50 PM PST

2 10 Nov 2004 03:50:14 PM PST

3 11/10/04

4 10 Nov 2004

5 November 10, 2004

6 11/10/2004

Other formatting attributes will become available in future releases as necessary.

11.9 Conditional Content

If many cases you want to include parts of a template based on the value of a content property. For example, you may only want to include the description paragraph if it is not empty. You can conditionally include part of a template using the `if` element:

```
<t:if test="description != null">
<p>
  <b>Description:</b>
  <t:value id="description">Description goes here.</t:value>
</p>
</t:if>
```

The system will only print this paragraph if the description property for the current item is not null.

At the very least, the following tests are available:

1. Test whether a text or date value is equal or not equal to `null`
2. Test whether a number value is less than, equal to or greater than a constant number

More complicated expressions may be not be supported in this release.

11.10 Related Items

Unlike a text, date or number property, the value of a related items property is a set of zero or more other content items. To display this property you typically need to format a set of item links in a table or list element.

You can format a list of related items using the `list` element:

```
<p>
  <t:list id=relateditems">
    <a href="${url}"><t:value id="title">Title Goes Here</t:value></a>
    <t:if test="index < length"><br></t:if>
  </t:list>
</p>
```

The output for this paragraph would look like this:

```
<p>
  <a href="/myfolder/page-one.html">Page One</a><br>
  <a href="/myfolder/page-two.html">Page Two</a>
</p>
```

The text inside the paragraph is repeated once for each item in the `relateditems` set. Inside the `list`, property names refer to the related item, rather than to the parent item associated with the page as a whole.

This example also uses the implicit `index` and `length` properties to check whether the last item has been output, and prints a BR tag if not. This properties are always available in the context of a `list` element.

All lists are associated with a `length` property that can be tested for conditional display. To expand on the previous example:

```
<p>
  <t:if test="relateditems.length > 0">
    <t:list id="relateditems">
      <a href="${url}"><t:value id="title">Title Goes Here</t:value></a>
      <t:if test="index < length"><br></t:if>
    </t:list>
  </t:if>
  <t:else>
    <em>No related items.</em>
  </t:else>
</p>
```

Note: Related items were previously displayed using the `variableRepeat` class. If feasible, I would like to update any templates that use this syntax and do away with it in this release.

11.11 Categories

Like a related items value, a categories value also may consist of zero or more individual categories. You also use a `list` element to tailor how the categories are displayed:

```
<p>
  <t:if test="mycategories.length > 0">
    <t:list id="mycategories">
      <a                    href="category.jsp?categoryID=${id}"><t:value
id="label">Category label goes here</t:value></a>
      <t:if test="index < length"><br></t:if>
    </t:list>
  </t:if>
  <t:else>
    <em>No categories.</em>
  </t:else>
</p>
```

Three properties are available for each category:

id A unique system ID for the category, suitable for passing as a request parameter.

name An author-recognizable keyword or code for the category, such as `apac`.

label A descriptive label of the category, such as `Asia-Pacific Region`

11.12 List Templates

The context for a list template is a set of items rather than a single item. A list template should contain a single `list` element without an `id` attribute. For example:

```
<h1>Latest Press Releases</h1>
<hr>
<t:list>
  <p>
    <b><a href="${url}"><t:value id="title">PR Title</t:value></a></b>
    <br>
    <t:value     id="description">Description     of     the     press
release.</t:value>
  </p>
</t:list>
```

Note that you can put any static markup you want above and below the `list` element.

Note: For legacy purposes the system also recognizes the following syntax:

```
<DIV class=templateRepeat>
  <P>
    <SPAN class=templateVariable format="link:false">«title»</SPAN>
  </P>
</DIV>
```

This syntax was replaced to provide a more general solution for repeating any part of a template, not just a `DIV` element.

11.13 Dynamic Components

Both display templates and body content can include dynamic components using the `include` element:

```
<t:include type="linkset" id="69210">
  Mockup linkset goes here.
</t:include>
```

Note: For legacy purposes the system also recognizes the following syntax:

```
<DIV class=templateComponent id=linkset-69210>
  Mockup linkset goes here.
</DIV>
```

This syntax was replaced to avoid the need to always place an include inside a `DIV` element.