

Hannon Hill Corporation

800.407.3540 | www.hannonhill.com | info@hannonhill.com

Makers of the Award-Winning Cascade Server content management software

Muenchian Grouping in Cascade Server

By Ross Williams

Monday, July 28th, 2008 at 3:00pm

As Services Trainer for Hannon Hill, clients often ask me how to group a list of articles by category. XSLT 1.0, the version of the XML-based language used in Cascade Server, doesn't have easy-to-use grouping capabilities, but we can take advantage of other features in this version that will allow us to create groups. This technique is called Muenchian Grouping, after Steve Muench at Sun Microsystems.

Here's how to do it:

First, we will assume that you are going to group by a dynamic metadata field named "Category." The parts of an index block that we're concerned with matching look like this (a simplified version of a real index block):

```
<system-index-block>  <system-page>      <name>sample_page</name>
<display-name>Sample</display-name>      <title>Sample Page</title>      ...
<dynamic-metadata>      <name>Category</name>      <value>Apples</value>
  </dynamic-metadata>      ...  </system-page>  <system-page>
<name>alternate_page</name>      <display-name>Alternate</display-name>
  <title>Alternate Sample Page</title>      ...  <dynamic-metadata>
<name>Category</name>      <value>Oranges</value>      </dynamic-metadata>
  ...  </system-page>  ... (More pages like the above) </system-index-block>
```

Assume there are many more pages and more categories than above; there are no reasonable limits to the number of pages you can sort or categories you can group by. The next thing we need is an XSLT stylesheet to match this index block. This first sample will list the pages in folder order. For simplicity's sake, we won't worry about subfolders or providing links to the pages; we'll just show the pages' Display Name in a list:

```
<?xml version="1.0" encoding="UTF-8" ?> <xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"> <xsl:template
match="/system-index-block"> <ul> <xsl:apply-templates select="//system-page"/>
</ul> </xsl:template> <xsl:template match="system-page"> <li><xsl:value-of
select="display-name"/></li> </xsl:template> </xsl:stylesheet>
```

Now we need some way to figure out, of all the pages in the index block, which categories are present. XSLT 1.0 provides the xsl:key element that does just that, so we're going to add one just after the xsl:stylesheet element in the above example. I'll just show the xsl:key element below, but at the end of this post I'll show the completed stylesheet:

```
<xsl:key name="pages-by-category" match="system-page"
use="dynamic-metadata[name = 'Category']/value" />
```

After this, we need to be able to find all the pages in a particular category. This can be done with the XPath function `key`. An example:

```
key('pages-by-category', 'Oranges')
```

But what we really need is a way to create a heading for each category, and *then* list all the pages in that category. In Muenchian Grouping, we'll know that we should add a category heading when we've run out of pages in one category, and we're moving onto the first page in the next category. "What?" you may be saying. We're going to sort all the pages by category, and then display a header before the first page in each category. We'll do that by adding this to the stylesheet:

```
<xsl:for-each select="//system-page[generate-id() =
generate-id(key('pages-by-category', dynamic-metadata[name =
'Category']/value)[1])]"> <xsl:sort select="dynamic-metadata[name = 'Category']/value"
/> <h1><xsl:value-of select="dynamic-metadata[name = 'Category']/value" /></h1> ...
```

Feel free to ask questions in the comments, but what that code literally says is, "for each page in the total list of pages (sorted by category), check if it is the same page as the first page with the same category in the key named 'pages-by-category'. Then display a heading with the category name." If that explanation ties your brain in knots as much as it tied my fingers, don't worry; you'll just need to follow this pattern when you create your own grouping.

After finding the first page in the category and displaying a heading, we need to display a list of pages in that category, sorted by Display Name:

```
<xsl:for-each select="key('pages-by-category', dynamic-metadata[name =
'Category']/value)"> <xsl:sort select="display-name" /> <xsl:apply-templates select="." />
</xsl:for-each>
```

The key ('pages-by-category', ...) should look familiar. It's like the key ('pages-by-category', 'Oranges') I showed above, except this time it gives a list of all the pages in the current category (from the previous for-each element, the one with `generate-id()` in it). Then, the `apply-templates` element passes each page to the `system-page` template, which will be just like what I showed in the no-grouping, original stylesheet. That's it. Those are all the techniques for grouping pages (or any other XML element). The complete stylesheet is below, though you'll probably need to change the text anywhere it says "Category" or "display-name," to customize it for your specific implementation. Ask any questions in the comments!

```
<?xml version="1.0" encoding="UTF-8" ?> <xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"> <xsl:key
name="pages-by-category" match="system-page" use="dynamic-metadata[name
= 'Category']/value" /> <xsl:template match="/system-index-block"> <xsl:for-each
select="//system-page[generate-id() = generate-id(key('pages-by-category',
dynamic-metadata[name = 'Category']/value)[1])]"> <xsl:sort
select="dynamic-metadata[name = 'Category']/value" /> <h1><xsl:value-of
select="dynamic-metadata[name = 'Category']/value" /></h1> <ul> <xsl:for-each
select="key('pages-by-category', dynamic-metadata[name = 'Category']/value)">
<xsl:sort select="display-name" /> <xsl:apply-templates select="." /> </xsl:for-each>
</ul> </xsl:for-each> </xsl:template> <xsl:template match="system-page">
<li><xsl:value-of select="display-name"/></li> </xsl:template> </xsl:stylesheet>
```

Category

- Resources

© 2001-2009 Hannon Hill Corporation. All rights reserved.