HowToTableOfContents

Overview

This HOWTO describes how to implement a simple table of contents for a page. For long, properly structured HTML pages you will have many sections to the page with each section headed by an H2 element. This technique, when the page is retrieved, finds the H2 elements on a page, generates anchors for each element and contructs a list of these anchors that can be placed at the top of the page (or anywhere else).

Step 1: Generate the anchors

The anchors on our H2 tags will be dynamically generated when the page is retrieved. We are going to have Lenya add anchors using the content of the H2 element so that `<h2>Subsection 1</h2>` becomes `<h2><a name="Subsection%201">Subsection 1</a></h2>`.

To do this we must modify how Lenya process our XML page into XHTML content. This is done in the transformation of `xhtml2xhtml.xsl`. We will add an XSL template to handle all the H2 elements.

```
<!-- Add named anchors to H2 elements for our Table of Contents -->
 <xsl:template match="xhtml:h2">
   <xsl:copy>
     <xsl:apply-templates select="@*"/>
     <xsl:if test="$rendertype != 'edit'">
     <a name="{.}">
     <xsl:apply-templates select="node()"/></a>
     </xsl:if>
   </xsl:copy>
</xsl:template>
```

When the transformation reaches an H2 element, this template is called. As long as we're not editing the page, we wrap the contents of the H2 element in a named anchor using the contents of the H2 element. (Note: If you're in the habit of putting anything besides text in your heading tags, you'll have to work out a diffrent way to name the anchors.)

Step 2: Create the Table of Contents

We'll create another XSL template for the table of contents in the presentation XSLT `page2xhtml-xhtml.xsl`. We'll scan through the document for H2 elements and add each to an unordered list. At the same time, we'll link each list item to the appropriate heading. We know the appropriate link to use because it is the content of the H2 itself. Because a table of contents with one item is fairly useless, we only create the list if there are more than one H2 elements.

```
<xsl:template name="toc">
  <xsl:if test="count(//xhtml:div[@id='body']/xhtml:h2) &gt; 1">
        <div id="toc">
     Table of Contents
         <ul id="toclist">
           <xsl:for-each select="//xhtml:div[@id='body']/xhtml:h2">
                 <li><a href="#{.}"><xsl:value-of select="." /></a></li>
           </xsl:for-each>
         </ul>
        </div>
  </xsl:if>
</xsl:template>
```

This can easily be modified to generate an ordered list or DIVs or whatever you prefer. We must then remember to actually display the list by calling this template in our XSLT, e.g:

```

<body>
        <xsl:call-template name="toc" />
        <xsl:apply-templates select="xhtml:div[@id = 'body']"/>
</body>
```

Step 3: Getting Clever

A table of contents, especially a big one, will take up a lot of room at the top of a page where it is most useful. This can be bothersome to users if they have to scroll very far to get to the beginning of the page. A couple modifications and a little Javascript can enhance the usability of the table of contents.

The following Javascript function will alternately show or hide our table of contents when called. (Taken from Showing and Hiding a DIV using CSS and Javascript)

```
function toggleLayer(whichLayer)
{
        if (document.getElementById)
        {
                // this is the way the standards work
                var style2 = document.getElementById(whichLayer).style;
                style2.display = style2.display? "":"block";
        }
        else if (document.all)
        {
                // this is the way old msie versions work
                var style2 = document.all[whichLayer].style;
                style2.display = style2.display? "":"block";
        }
        else if (document.layers)
        {
                // this is the way nn4 works
                var style2 = document.layers[whichLayer].style;
                style2.display = style2.display? "":"block";
        }
}
```

In order for this to work properly you must style the list to be hidden at first. So add the following to your CSS:

```
#toclist { display: none; }
```

Finally add a link that calls this Javascript function, passing it the id of the list.

```
<xsl:template name="toc">
  <xsl:if test="count(//xhtml:div[@id='body']/xhtml:h2) &gt; 1">
        <div id="toc">
        <div id="toclink"><a href="javascript:toggleToc('toclist');">Table of Contents</a></div>
        <ul id="toclist">
        <xsl:for-each select="//xhtml:div[@id='body']/xhtml:h2">
                <li><a href="#{.}"><xsl:value-of select="." /></a></li>
        </xsl:for-each>
        </ul>
        </div>
  </xsl:if>
</xsl:template>
```