HowToBodySwitching

Using the body tag for styling your page

One of the useful techniques that has surfaced over the last year or so has been assigning an `id` attribute to your body tag so that you can use one CSS file for stylings across multiple sections of your site. For example, if you have a site that has tabs at the top of your site leading users to each major section of your site, you may want to highlight that tab in some way when the user is in that section of the site.

You could approach the styling for this in a few ways. One method is to simply create a separate CSS file with all the special stylings needed for the highlighted tab, then include it after all of your standard CSS files, putting the cascade in Cascading Style Sheets to work. Another method is to use a separate CSS file for each subsection of your site, one for each tab in our example.

Or, in order to minimize the amount of CSS you have and worry about what is overriding what in your CSS file, you could create an ID for each section and make that an attribute to your body tag. Then, you could create CSS that changes the design of the tab based on the ID of the `body` tag.

Example

So, say for example that your tabs at the top of every page on your site has the following HTML markup:

```
<ul id="tabs">
  <li id="tabs-home"><a href="/index.html">Home</a></li>
  <li id="tabs-products"><a href="/products.html">Products</a></li>
  <li id="tabs-services"><a href="/services.html">Services</a></li>
  <li id="tabs-shop"><a href="/shop.html">Shop!</a></li>
</ul>
```

So, when you click on the Products link, and you click on any other links within the Products section, you want that Products tab to have a different color. So, you could assign the body tag with an `id`:

```
<html>
  <head>
    <title>Products</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  </head>
  <body id="products">

    ... content and navigation for page to go here ...

  </body>
</html>
```

The next step would be to create the CSS to make this happen. Here's a simple example to show that the color of the text is changed when in the appropriate section:

```
ul#tabs li {
  color: #000;
}

body#products ul#tabs li#tabs-products,
body#home ul#tabs li#tabs-home,
body#services ul#tabs li#tabs-services,
body#shop ul#tabs li#tabs-shop {
  color: #f00;
}
```

See how when the `id` for the body combined with the `id` for the list item match that they take on a different color? So now this stylesheet works across multiple sections of your site in only a matter of a few lines of CSS.

Doing this for Lenya

So, the whole point here is to show how this can be done in Lenya. We'll break this down into 2 parts, the Live area and the Authoring area.

The Live Area

I'll use this area first because it's the easiest to demonstrate. Using the basic XSL files in your publication's `xslt` directory, you can assign the `id` attribute quite easily. In our case, let's use the document ID for each top-level page as our `id` for the `body` tag. Here's what the XSL file would look like:

```
   ... initial stuff in XSL file up here ...

<xsl:template match="cmsbody">
  <html>
    <head>
      <title><xsl:value-of select="//lenya:meta/dc:title"/></title>
      <meta http-equiv="content-type" content="text/html; charset=utf-8" />

      <link rel="stylesheet" type="text/css" media="screen, projection" href="{$root}/css/level2.css" />

    </head>

    <body>
      <xsl:attribute name="id">
        <xsl:choose>
          <xsl:when test="substring-before(substring($documentid,2),'/') = ''">
            <xsl:value-of select="substring($documentid,2)"/>
          </xsl:when>
          <xsl:otherwise>
            <xsl:value-of select="substring-before(substring($documentid,2),'/')"/>
          </xsl:otherwise>
        </xsl:choose>
      </xsl:attribute>

    ... rest of file below ...
```

So, right after the body tag, I created an attribute with a name of `id`. I do a quick if/else statement with `<xsl:choose>` to check if the following string comes up empty:

```
  substring-before(substring($documentid,2),'/') = ''
```

Let's work our way out. The initial `substring($documentid,2)` simply takes the document ID of the page in question, and strips out the front character, which would simply be a slash (if this were the home page of your publication, then the document ID would be `/index`). The `substring-before` says only using everything from the result of the initial `substring` that appears before the first `/`. So, this works great for pages further down in your site's tree.

For example, if you are on the page with a document ID of `/products/coffee/premium`, then the inner substring would produce: `products/coffee/premium` and the outer substring-before would produce: `products`. Perfect! We can use that for our ID.

Going back to the if/else thing we have going on, if the result is empty (meaning that we are on one of the top level pages - there was no slash at the end so all the text was removed), then we just use the regular `substring` to capture the document ID. Publish your pages with your new-found knowledge and watch the `id` attribute get added automatically.

The Authoring Area

The authoring area is a little different because you can't control the body tag from your publication's `xslt` directory. So, you need to do something a little more tricky. Go to your publication's `lenya/xslt/` directory and create a new folder called `menu`. You want to copy the file `$LENYA_HOME/lenya/xslt/menu/menu2xslt.xsl` to the `menu` directory you just created. That file controls the layout of your site in the Authoring area.

Now, open up that file you copied over and add in the same stuff we did for the `id` attribute in the live area:

```
... initial stuff in XSL file up here ...

 <body>
   <xsl:attribute name="id">
     <xsl:choose>
       <xsl:when test="substring-before(substring($documentid,2),'/') = ''">
         <xsl:value-of select="substring($documentid,2)"/>
       </xsl:when>
       <xsl:otherwise>
         <xsl:value-of select="substring-before(substring($documentid,2),'/')"/>
       </xsl:otherwise>
     </xsl:choose>
   </xsl:attribute>

 ... rest of file below ...
```

Don't get excited. You aren't done quite yet. You'll need to add the following near the top of the file too:

```
<xsl:param name="documentid"/>
```

Just put this underneath the others that look the same. Of course, you can't just add this parameter without it being assigned someplace, so that's the last step. Go to `$LENYA_HOME/` and edit the file global-sitemap.xmap. Look for a section that looks like the following, adding in the line for `$documentid`:

```
        <map:otherwise>
          <map:transform src="{fallback:xslt/menu/menu2xslt.xsl}">
            <map:parameter name="contextprefix" value="{request:contextPath}"/>
            <map:parameter name="publicationid" value="{1}"/>
            <map:parameter name="area" value="{2}"/>
            <map:parameter name="documenturl" value="{page-envelope:document-url}"/>
            <map:parameter name="documentid" value="{page-envelope:document-id}"/>
          </map:transform>
        </map:otherwise>
```

Save and exit. If your sitemaps are cached, you'll want to restart Lenya for the changes to take place. Why does adding the copy of menu2xslt.xsl in your publication work? Well, the answer is shown above. See where it uses the notation `fallback:`? That simply means that it will check your publication first if you have something there, and if you don't, it will just use the Lenya-wide default.

Now all you need is to make sure your CSS files are up to snuff and you're on your way.