



Customizing HIP: A Beginner's Guide to Editing XSLT Stylesheets

Phil Feilmeyer
Hennepin County Library
CODI/HUG - November, 2003
<http://www.hclib.org/catalog>



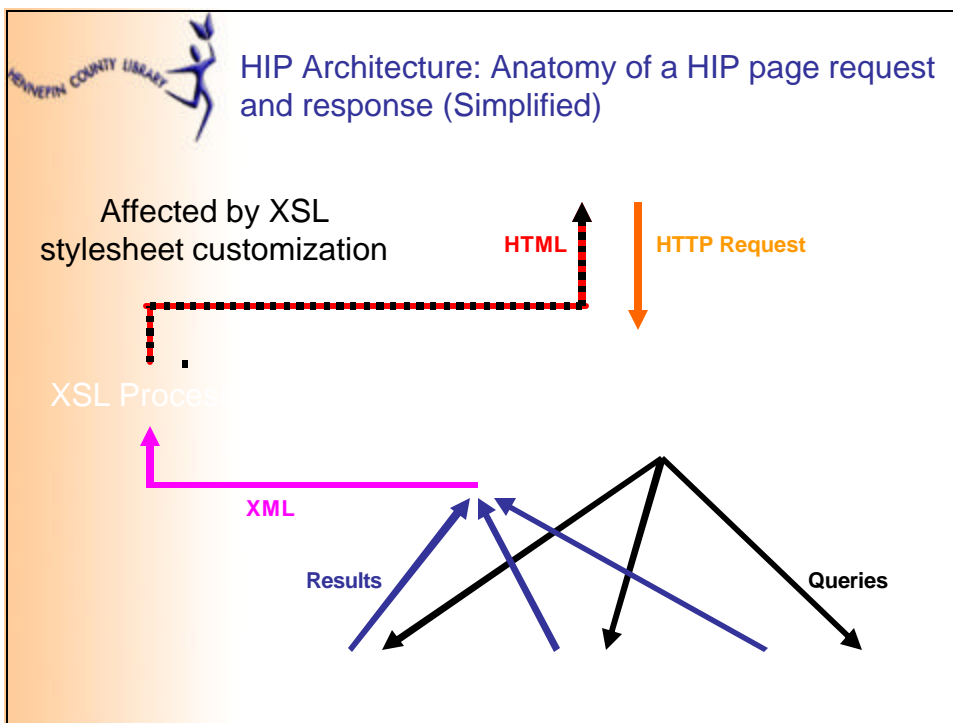
Scope


- Introduction to customizing the HIP user interface by way of the XSLT Stylesheets
- XSLT = Extensible Stylesheet Language for Transformations
 - Tag-based programming language
 - Transforms XML documents into something else (in this case, HTML)



XSLT Sample


```
<xsl:template name="footer">
  <center>
  <table>
    <xsl:attribute name="class">
      <xsl:value-of select="$css_table_background"/>
    </xsl:attribute>
    <xsl:attribute name="valign">bottom</xsl:attribute>
    <xsl:attribute name="width">100%</xsl:attribute>
    <xsl:if test="not(boolean(/searchresponse/sendingemail))">
      <tr>
        <td>
          <xsl:attribute name="nowrap">true</xsl:attribute>
          <xsl:attribute name="align">center</xsl:attribute>
          <a>
            <xsl:attribute name="class">
              <xsl:value-of select="$css_global_anchor"/>
            </xsl:attribute>
            <xsl:attribute name="href">http://www.hclib.org</xsl:attribute>
            <xsl:text>home</xsl:text>
          </a>
        </td>
      </tr>
    </xsl:if>
  </table>
</center>
</xsl:template>
```






Customizing at Hennepin County Library

- HIP 2.03.01 (production) and 2.1 (development) running on Windows 2000
- Dynix ILS
- HCL is a single library with 27 branches
 - Each branch has a HIP profile, but they are essentially identical
 - Decision-making aspects of customization are centralized
- Experience with HTML and ColdFusion
 - ColdFusion is a web application platform that uses templates and a tag-based programming language




Customizing at Hennepin County Library

- Resources available for customization effort
 - Development server
 - Extra ILS licenses
 - Staff with experience in web application development
- Started with cosmetic changes
 - Reduction of white space
 - Removal of extraneous information
 - Color conformity with the library's web pages




Customizing at Hennepin County Library

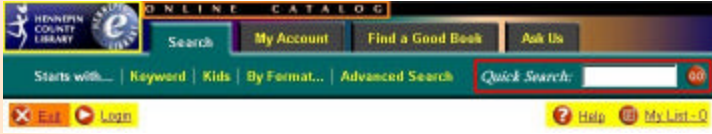
- Breakthrough moment: realizing there are two parts to every HIP page:
 - Functional elements
 - links, input boxes, submit buttons, JavaScript
 - Display elements
- As long as the functional elements are left intact, you can do as you like with the display elements
- First major customization: [toolbar2.xsl](#)




Header example (toolbar2.xsl)



The original header features a yellow navigation bar with links for 'My List - 0', 'Login', and 'Help'. Below it is a dark green search bar with options for 'Starts with...', 'Keyword', 'Kids', 'By Format...', and 'Advanced Search'.




The customized header, titled 'ONLINE CATALOG', features a dark blue navigation bar with links for 'Search', 'My Account', 'Find a Good Book', and 'Ask Us'. Below it is a dark green search bar with a 'Quick Search' input field and a search button. The 'Exit' and 'Login' links are now located at the bottom left, and 'Help' and 'My List - 0' are at the bottom right.




Benefits of Customizing

- Shape the catalog to better meet the specific needs of your library
 - Developers need to create a single product that libraries of any type can use
- Override Dynix’s development decisions
 - especially with regard to layout
- Generates interest in the catalog from staff and the public
 - Sense of ownership when a suggestion is incorporated
- Satisfying




Risks of Customizing

- Not supported, help from Dynix is billable
 - If you’re careful, the risk is small
- Upgrades/patches are more work and take longer
 - Potentially a lot more work and a lot longer
 - Not too bad if a good upgrade plan is developed
 - 2.0 to 2.01 40 hours
 - 2.02 to 2.03, 2.03 to 2.03.01 1 hour
 - 2.03.01 to 2.1 4 hours
- A major change in system architecture could have serious consequences
 - A close call: Dynix patches required for 2.0x upgrades were backward compatible




Prerequisites to Customizing

- A solid working knowledge of HTML
 - Primarily working backwards from the final product
- HTML tables
 - HIP uses tables extensively as the framework for each page
 - Table cells and rows are used to display variable sets of information
 - Tabs in the tabset
 - Items in the item display
 - Subject headings in the sidebar
- Basic understanding of how Cascading Stylesheets (CSS) work




Prerequisites to Customizing

- Access and rights to the XSLT stylesheets
- Development HIP
 - Separate installation of HIP for development purposes
 - Develop, test and tweak customizations without bothering users of the production system
 - Very easy to copy pre-tested stylesheets to the production HIP
 - Additional ILS licenses
 - Dynix: uses at least one Dynix/UniVerse/Unix license
 - Horizon: ??
 - On a separate server
 - Desktop PC with 1GB of RAM (or less?)
 - On the production server
 - Runs on different ports or a different IP address than production HIP, more complex setup
 - Resource issues?




Managing Upgrades with Customized Stylesheets

- During upgrades, stylesheets may be replaced with new versions
 - Customizations are NOT retained
- Before the upgrade, make sure you have backup copies of all the stylesheets that have been customized *along with the original versions*
- After the upgrade, compare the new version of the stylesheet to the old *original uncustomized version*.
 - Date/time modified
 - Size




Managing Upgrades with Customized Stylesheets

- If there are no differences between the old original version and the new version, your customized version needs no changes
- If there are differences between the old and new stylesheets, find out the nature of the changes
 - Side by side comparison
 - If the changes are relatively minor, incorporate them into your old customized stylesheet
 - If the change is relatively major compared to your customizations, incorporate your customizations into the new version of the stylesheet
- Upgrade documentation has been pretty good about listing the stylesheets that will be replaced during the upgrade



The XSL Processor

- Stylesheets are organized into six functional areas of HIP
- Each of the functional areas has a **root** stylesheet
 - [searchresponse.xml](#) – search, view results, login
 - [patronpersonalresposne.xml](#) – my account
 - [placerequest.xml](#) – hold placement
 - [mylistresponse.xml](#) – my list, manage lists
 - [sendbooklist.xml](#) – send a booklist via email
 - [sessioninfo.xml](#) – ???




The XSL Processor

- Root stylesheets are associated with their supporting stylesheets through two configuration files
 - Located up one level from the stylesheets in the main “XSLProcessor” folder
 - [xsltransformer.xml](#)
 - [xsltransformer.cfg](#)




Snippet from xsltransformer.xml

```
...  
<Root>  
  <Name>./xsl/placerequest.xsl</Name>  
  <Includes>  
    <Include type="error">error.xsl</Include>  
    <Include type="toolbar">toolbar2.xsl</Include>  
    <Include type="requestoptions">placerequestoptions.xsl</Include>  
    <Include type="footer">generalfooter.xsl</Include>  
    <Include type="timer">expiretimer.xsl</Include>  
    <Include type="bookings">bookings.xsl</Include>  
    <Include type="image">image.xsl</Include>  
    <Include type="features">onoff.xsl</Include>  
    <Include type="attributes">other.xsl</Include>  
    <Include type="string">string.xsl</Include>  
  </Includes>  
</Root>  
...
```




XSLT Editing Tips and Tricks

- Be sure to make a backup copy of a file before you edit
 - Be comfortable with your backup plan
- Use a plain text editor or an editor designed for XSLT
 - MS Word or Word Pad might save the file in a non-ASCII format
- If possible, break up the customization into small edits that can be tested individually along the way
- XSL Processor must be stopped and restarted for changes to take affect
 - Horizon Application Server (JBoss) does not need to be restarted




XSLT Editing Tips and Tricks

- Newly edited stylesheets won't be tested until they're called by HIP
 - Example: changes to **placerequest.xsl** won't be tested until a hold is placed
- Run the XSL Processor as an application rather than as a service when you are testing an edited stylesheet
 - More complete error messages
 - Fast and easy to stop and start



XSLT Editing Tips and Tricks

- HIP stylesheets contain both XSL and HTML tags
 - XSL tags are labeled as such:
 - `<xsl:attribute>` `</xsl:attribute>`
 - `<xsl:if>` `</xsl:if>`
 - HTML tags look like bare HTML tags
 - attributes are applied with the `<xsl:attribute>` tag



XSLT Editing Tips and Tricks


The following XSLT:

```
<img>  
  <xsl:attribute name="src">/images/go.gif</xsl:attribute>  
  <xsl:attribute name="border">0</xsl:attribute>  
</img>
```

Generates this HTML:


```

```



XSLT Editing Tips and Tricks

- XSL Processor is picky when it comes to tags:
 - Tags are case sensitive
 - Bad: `<TD> ... </td>`
 - Good: `<td> ... </td>`
 - End tags are required
 - Bad: `
`
 - Good: `
</br>`
 - Empty tags can use the end marker to close:
 - Good: `<p/>`
`<xsl:value-of select="text"/>`




XSLT Editing Tips and Tricks

- Comment tags
 - Add comments to the stylesheet using standard HTML comment tags


```
<!-- pf added series title -->
```
 - Use comment tags to deactivate Dynix code

```
<!-- pf commented out unnecessary stuff
<xsl:if test="boolean(normalize-space(explanation))">
  <a>
    <xsl:attribute name="class">
      <xsl:value-of select="$css_normal_black_font1"/>
    </xsl:attribute>
    <xsl:value-of select="explanation"/>
  </a>
</xsl:if>
-->
```
 - NOTE: A comment used to deactivate code cannot include any comment tags



XSLT Editing Tips and Tricks

- Following the XSL trail from the HTML source to the corresponding spot in the stylesheets can be tricky
- Never a direct route
 - Root and support stylesheets
 - Variables
 - Templates
 - Cascading Style Sheets
 - Flow control tags




XSLT Editing Tips and Tricks

- Root and support stylesheets
 - The XSL responsible for the HTML on any given page comes from multiple stylesheets
 - Find out which root file by adding “&GetXML=true” to the end of the URL
 - Case sensitive
 - Remove “#focus” from the end of the URL
 - Example:

```
http://catalog.hclib.org/ipac20/ipac.jsp?profile=rd#focus
```



```
http://catalog.hclib.org/ipac20/ipac.jsp?profile=rd&GetXML=true
```



XSLT Editing Tips and Tricks

- Variables
 - HTML to be customized

```
Send suggestions to
```

```
<a class="normalBlackFont1"
```

```
  href="mailto:d.nackos@epixtech.com">
```


```
  d.nackos@epixtech.com
```

```
</a>
```
 - Assigned to a variable

```
<xsl:variable name="emailsuggestions">
```

```
  d.nackos@epixtech.com
```


```
</xsl:variable>
```



XSLT Editing Tips and Tricks

- Variables
 - Variable called by the code to be customized

```
<!-- pf c-o dynix code
Send suggestions to
<a>
  <xsl:attribute name="class">
    <xsl:value-of select="$css_normal_black_font1"/>
  </xsl:attribute>
  <xsl:attribute name="href">
    mailto:<xsl:value-of select="$emailsuggestions"/>
  </xsl:attribute>
  <xsl:value-of select="$emailsuggestions"/>
</a>
-->
```




XSLT Editing Tips and Tricks

- Templates
 - Allows XSLT code to be packaged and reused
 - Define a template

```
<xsl:template name="footer">
  <center>
    ...
  </center>
</xsl:template>
```

- Call up a template

```
...
<xsl:call-template name="footer"/>
</form>
</body>
</html>
```



XSLT Editing Tips and Tricks


Another template example

```
<!-- put the current session ID in to the URL -->
<xsl:template name="addSessionToURL">
  <xsl:choose>
    <!-- if sending an email get new session -->
    <xsl:when test="/searchresponse/sendingemail = $true">
      <xsl:text>session=new</xsl:text>
    </xsl:when>
    <!-- if not sending an email preserve session ID -->
    <xsl:otherwise>
      <xsl:text>session=</xsl:text>
      <xsl:value-of select="/searchresponse/session"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

Output of `<xsl:call-template name="addSessionToURL"/>`

As part of an email: **session=new**


All other situations: **session=106W6250V8X87.3594**



XSLT Editing Tips and Tricks

- Cascading stylesheets
 - In HIP, defines the style elements for fonts and buttons
 - Colors
 - Size
 - Text decoration: bold, underline, etc.
 - Assigned in the XSLT by the 'class' attribute

```
<a>
  <xsl:attribute name="class">
    <xsl:value-of select="$css_normal_black_font1"/>
  </xsl:attribute>
  <b><xsl:value-of select="$full_bib_call_num"/></b>
</a>
```




XSLT Editing Tips and Tricks

- Flow control tags
 - `<xsl:for-each ... > </xsl:for-each>`
 - Repeat the code within the tag for each item in the dataset

```
<xsl:for-each select="//blockdata/block">
  <xsl:call-template name="blocktable"/>
</xsl:for-each>
```
 - `<xsl:if ... > </xsl:if>`
 - Only use the code within the tag if the condition is true

```
<xsl:if test="boolean(//itemsout)">
  ...
</xsl:if>
```




XSLT Editing Tips and Tricks

- Flow control tags (continued)

```
<xsl:choose>
  <xsl:when ... > </xsl:when>
  <xsl:otherwise> </xsl:otherwise>
</xsl:choose>
```


 - Choose which code to use from multiple alternatives

```
<xsl:choose>
  <xsl:when test="//profile = 'elibrary'">
    call your local library
  </xsl:when>
  <xsl:otherwise>
    ask for assistance at the info desk
  </xsl:otherwise>
</xsl:choose>
```




Editors

- Stylesheet Editor
 - XMLSpy
 - Xselerator
 - FrontPage 2003
- Text Editor
 - Notepad
 - UltraEdit
 - Find/Replace across multiple files
 - Line number indicator, "Go to line" utility
 - Compare files



Other Resources

- *XSLT*, Doug Tidwell
- *iPac User Interface Customization Guide*, from the Dynix support website
 - <http://customer.dynix.com/archive/pdf/?support/ipac/docu/202/iPacCustom.pdf>
- iPac Listserv and archive
 - <http://lists.tbtc.org/mailman/listinfo/ipac/>
- iPac Zone, compiled by James Day
 - <http://www.scpl.lib.fl.us/ipaczone/>
 - Contributions from HIP administrators/customizers
- Google



Customizing HIP: A Beginner's
Guide to Editing XSLT Stylesheets

Phil Feilmeyer
Hennepin County Library

pfeilmeyer@hclib.org
<http://www.hclib.org/catalog>

Handouts and Powerpoint:
<http://www.hclib.org/extranet>

CODI/HUG - November, 2003