



Customizing HIP: Advanced XSLT Stylesheet Editing

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



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 26 branches
 - Each branch has a HIP profile, but they are essentially identical
 - Decision-making aspects of customization are centralized



Customization at Hennepin County Library


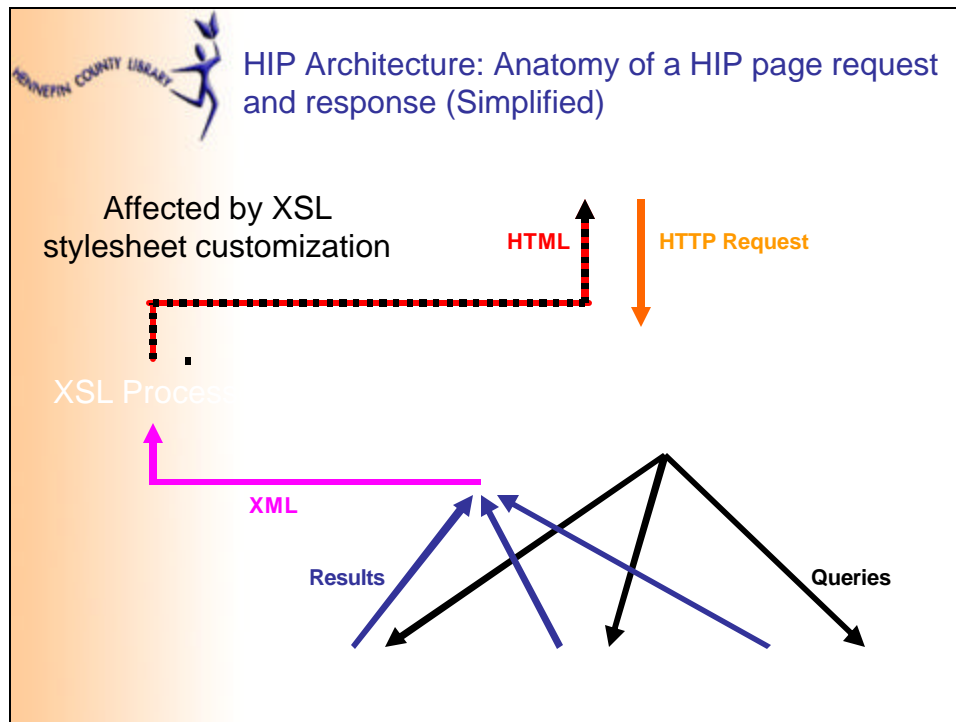
- Breakthrough moment: realizing there are two parts to every HIP page:
 - Functional elements
 - links, input boxes, submit buttons, JavaScript
 - Display elements
- As long as the functional elements are left intact, you can do as you like with the display elements
- First major customization: **toolbar2.xsl**



Header example (toolbar2.xsl)




The screenshot displays two examples of the library's header. The top example shows a toolbar with buttons for 'My List - 0', 'Login', and 'Help'. The bottom example shows a toolbar with buttons for 'Search', 'My Account', 'Find a Good Book', and 'Ask Us', along with a 'Quick Search' input field.



Viewing XML data

- To see the XML behind most HIP pages, add “&GetXML=true” to the URL
 - Case sensitive
 - Remove “#focus” from the end of the URL first
 - Example:
 - <http://catalog.hclib.org/ipac20/ipac.jsp?profile=rd#focus>
 - <http://catalog.hclib.org/ipac20/ipac.jsp?profile=rd&GetXML=true>
 - You are re-submitting the HIP request, so you may be re-renewing or re-reserving a title
 - Occasionally it doesn't work
 - address bar only contains base URL




XML Path Language (XPath)

- Syntax for referencing elements of an XML document
 - Similar to UNIX and DOS filesystem addressing

```
<?xml version="1.0" encoding="UTF-8"
standalone="yes" ?>
<searchresponse>
  <version>2.03.01</version>
  <session>VT677734S9875.51</session>
  <profile>elibrary</profile>
  <expirepage>300</expirepage>
  ...
</searchresponse>
```
 - XPath reference to the user's profile:
`/searchresponse/profile`

`<xsl:value-of select="/searchresponse/profile"> = elibrary`




XML Path Language (XPath)

- Absolute path starts with a 'slash'
`/patronpersonalresponse/patroninfo/name/full`
- Use double-slash to left truncate
`//profile`
`//session`
- Include position in brackets for 'multivalue fields'
`/searchresponse/summary/searchresults/results/row[3]/isbn`
- No initial 'slash' for a relative path
`TITLE/data/text`

Context is usually provided by `<xsl:for-each>` tag

```
<xsl:for-each select="//searchresults/results/row">
  ... <xsl:value-of select="TITLE/data/text"/> ...
</xsl:for-each>
```




Benefits of looking at the XML

- If a patron is logged in, some of their personal data is part of the XML on every HIP page

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<searchresponse>
  ...
  <security>
    <auth>true</auth>
    <name>FEILMEYER, PHILIP A</name>
    <patronid>831256</patronid>
    <address>12601 RIDGEDALE DR, MINNETONKA, MN 55305</address>
    <emailaddress>pfeilmeyer@hclib.org</emailaddress>
    <phone>952-847-8768</phone>
  </security>
  ...
</searchresponse>
```

- The email address in XPath
[/searchresponse/security/emailaddress](#)
[//security/emailaddress](#)



Benefits of looking at the XML

- Customized XSLT to automatically include the patron's email address in the input box whenever an email option appears

```
<input>
  <xsl:attribute name="class">
    <xsl:value-of select="$css_text_input_box"/>
  </xsl:attribute>
  <xsl:attribute name="type">text</xsl:attribute>
  <xsl:attribute name="size">40</xsl:attribute>
  <xsl:attribute name="name">emailaddress</xsl:attribute>
  <xsl:attribute name="maxlength">100</xsl:attribute>
  <xsl:attribute name="value">
    <xsl:if test="string-length(//security/emailaddress) > 0">
      <xsl:value-of select="//security/emailaddress"/>
    </xsl:if>
  </xsl:attribute>
</input>
```

 **Benefits of looking at the XML**

- Used the same information when confirming a hold placement to check if a patron is using email notification

Your request has been successfully placed


Subterranean jungle.
by *Ramones (Musical group)*

You will be notified when this title is available.
The pickup location for this request will be: **Ridgedale**



Make the switch to email notification now!


[Make the Switch](#) [Return to Searching](#) [Logout and Return](#)

 **Another editing example**


- Make something conditional based on profile
 - Provide more targeted information


```
<xsl:if test="not(//profile = 'elibrary')">
  <tr>
    <td>
      To protect your personal information,close this
      browser window before you leave the workstation.
    </td>
  </tr>
</xsl:if>
```
 - Another example

```
<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>
```


 **Advanced Customizing – incorporating custom data elements**

- Bring in data from the ILS for a customization, but don't want it treated as other display additions
 - Format icon in the summary display




 **Advanced Customizing – incorporating custom data elements**

- Add the new data element to the appropriate display using the Admin Tool



Add New Element		
Reorder Data Elements		
	Label	Field Location/Dictionary
1	<input type="text" value="Title"/>	TITLE
2	<input type="text" value="Author"/>	AUTHOR
3	<input type="text" value="Publisher"/>	PUBLISHER
4	<input type="text" value="Pub Date"/>	PUBDATE
5	<input type="text" value="Call No."/>	CALL
6	<input type="text" value="URL"/>	MEDUSA
7	<input type="text" value="Edition"/>	L-IPAC_EDITION_SUMMARY Delete
8	<input type="text" value="Format Icon"/>	L-IPAC_FORMAT2 Delete




Advanced Customizing – incorporating custom data elements

–Check the XML to find out how the bibliographic data is structured

Part 1: the Label

```
<searchresponse>  
  <summary>  
    <searchresults>  
      <header>  
        <col>  
          <label>Edition</label>  
        </col>  
        <col>  
          <label>Format Icon</label>  
        </col>  
      </header>  
    </searchresults>  
  </summary>  
</searchresponse>
```

`/searchresponse/summary/searchresults/header/col[n]/label`




Advanced Customizing – incorporating custom data elements

Part 2: the Data

```
<searchresponse>  
  <summary>  
    <searchresults>  
      <results>  
        <row>  
          <cell>  
            <data>  
              <text>Unabridged ed.</text>  
            </data>  
          </cell>  
          <cell>  
            <data>  
              <text>AC.gif</text>  
            </data>  
          </cell>  
          ...  
        </row>  
      </results>  
    </searchresults>  
  </summary>  
</searchresponse>
```


`/searchresponse/summary/searchresults/results/row[n]/cell[n]/data/text`



Advanced Customizing – incorporating custom data elements

–The default `summary.xsl` stylesheet takes any additional elements and just appends them as new rows in the table


```
<xsl:for-each select="cell">
  <xsl:variable name="pos">
    <xsl:value-of select="position()"/>
  </xsl:variable>
  <xsl:choose>
    <xsl:when test="boolean(normalize-space(data/text))">
      [create a row with label and data]
    </xsl:when>
    <xsl:otherwise>
      [don't do anything]
    </xsl:otherwise>
  </xsl:choose>
</xsl:for-each>
```



Advanced Customizing – incorporating custom data elements

- Identify the position of the label of your new element using `<xsl:variable>` at the uppermost level in the template


```
<xsl:variable name="format_pos">
  <xsl:for-each select="//searchresults/header/col">
    <xsl:if test="label = 'Format Icon'">
      <xsl:value-of select="position()"/>
    </xsl:if>
  </xsl:for-each>
</xsl:variable>
```



Advanced Customizing – incorporating custom data elements from HIP

- Insert your custom code in the appropriate spot


```
<td>
  <xsl:attribute name="rowspan">2</xsl:attribute>
  <xsl:attribute name="valign">top</xsl:attribute>
  <xsl:attribute name="width">50</xsl:attribute>
  <img>
    <xsl:attribute name="src">
      <xsl:value-of select="$images_path"/>
      <xsl:value-of select="cell[number($format_pos)]/data/text"/>
    </xsl:attribute>
    <xsl:attribute name="width">50</xsl:attribute>
    <xsl:attribute name="height">60</xsl:attribute>
  </img>
</td>
```



Advanced Customizing – incorporating custom data elements


- Modify the default code to exempt this data from generic display

```
<xsl:for-each select="cell">
  <xsl:variable name="pos">
    <xsl:value-of select="position()"/>
  </xsl:variable>
  <xsl:choose>
    <xsl:when test="boolean(normalize-space(data/text)) and
      ($pos != $format_pos)">
      [create a row with label and data]
    </xsl:when>
    <xsl:otherwise>
      [don't do anything]
    </xsl:otherwise>
  </xsl:choose>
</xsl:for-each>
```




Advanced Customizing – Modifying text coming from the ILS

```
<xsl:for-each select="//message/reason">  
  <xsl:if test="contains(., 'Old PIN value entered does  
    not match current PIN value')">  
    <a>  
      <xsl:attribute name="class">  
        <xsl:value-of select="$css_bold_black_font1"/>  
      </xsl:attribute>  
      NOTE: If you have been using the last four digits of  
        your phone number as a PIN, leave the "Old PIN" box  
        empty when you change your PIN  
    </a>  
  </xsl:if>  
</xsl:for-each>
```




Advanced Customizing – Other possibilities

- Add JavaScript to a page
 - Calculate number of days until due date
 - “Validate” email address
- Connect dynamically to another web application
 - Update patron information
 - Due date calendar
 - Find a Good Book tab



Upgrades with Customized Stylesheets – The Hennepin Method


- Management of customized stylesheets
 - Make copies of the original XSL Processor configuration files for the current version
`xsltransformer.xml` (as `original_20301_xsltransformer.xml`)
`xsltransformer.cfg` (as `original_20301_xsltransformer.cfg`)
 - Identify stylesheets that have customizations using unique filenames
 - Whenever a customization is made to an original stylesheet (no matter how small), give the edited file a new name and leave the unedited original in the XSL folder
 - Example: `generalfooter.xsl` becomes `hcl_generalfooter_20301.xsl` but the original unedited `generalfooter.xsl` remains in the XSL folder



Upgrades with Customized Stylesheets – The Hennepin Method

–Update any and all references to the customized files in the XSL Processor config files `xsltransformer.xml` and `xsltransformer.cfg`


```
<Root>
  <Name>./xsl/mylistresponse.xsl</Name>
  <Includes>
    <Include type="timer">expiretimer.xsl</Include>
    <Include type="features">onoff.xsl</Include>
    <Include type="image">image.xsl</Include>
    <Include type="attributes">hcl_other_20301.xsl</Include>
    <Include type="mylist">hcl_mylist_20301.xsl</Include>
    <Include type="string">hcl_string_20301.xsl</Include>
    <Include type="error">error.xsl</Include>
    <Include type="toolbar">hcl_toolbar2_20301.xsl</Include>
    <Include type="footer">hcl_generalfooter_20301.xsl</Include>
  </Includes>
</Root>
```



Upgrades with Customized Stylesheets – The Hennepin Method


–Update any and all references to the customized files in the XSL Processor config files **xsltransformer.xml** and **xsltransformer.cfg**

- The **root** files cannot be renamed, so if you customize them you'll need to keep track “manually”
 - HIP refers to the root file by the filename in the XML?
 - Keep copies of the unedited originals around with an identifying filename
- If you use a ‘Find and Replace’ tool to update the files, watch out for similarly named stylesheets
 - items.xsl**
 - summaryitems.xsl**




Upgrades with Customized Stylesheets – The Hennepin Method

- Upgrade procedure
 - Use for upgrades, hotfixes, service packs, patches
 - Before the upgrade, copy the XSLProcessor folder to an available location (e.g. HIP server, workstation, or network drive)
 - Rename the current working versions of the XSL Processor config files:
 - **xsltransformer.xml** becomes **xsltransformer_20301.xml**
 - Move the original config files back to their original filename
 - **original_xsltransformer_20301.xml** becomes **xsltransformer.xml**




Upgrades with Customized Stylesheets – The Hennepin Method

- Upgrade procedure (continued)
 - If you have customized **root** files
 - Rename the current customized versions using an identifiable filename
 - Put the original unedited versions back to their original filename
 - Run the upgrade.
 - After it's finished, you should have a default installation of the new version of the XSL Processor



Upgrades with Customized Stylesheets – The Hennepin Method


- After the upgrade
 - For all of the stylesheets that you've customized, compare the new version with the old unedited version
 - File size
 - Date/time modified
 - If there are no changes in default versions, your customized version can be renamed to the current version
 - **hcl_generalfooter_20301.xsl** becomes **hcl_generalfooter_21.xsl**



Upgrades with Customized Stylesheets – The Hennepin Method

–If the new version of the stylesheet IS different than the old unedited version, use a “Compare Files” utility to find out the nature of the changes

- If the changes are relatively minor, incorporate them into the old customized stylesheet and rename it to the current version
- If the changes are major compared to your customization, recreate your customizations in the new stylesheet and rename it to the current version
- In either case, make sure that an original copy of the new default stylesheet is left in the XSL folder
- Don’t forget the **root** files if they’re customized



Upgrades with Customized Stylesheets – The Hennepin Method


–Update the XSL Processor configuration files

- Make backup copies of the new config files
- Edit the new files, updating any and all references to the customized stylesheets

–Upgrade documentation has been pretty good about listing the stylesheets that are being replaced

–Perl program to initially compare files, copy to separate folders, and generate a report

• 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



Customizing HIP: Advanced XSLT
Stylesheet Editing

Phil Feilmeyer
Hennepin County Library

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

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

CODI/HUG - November, 2003