Avoiding duplicate effort with post-preprocessing in the DITA Open Toolkit

Presented by: Tim Grantham
Senior Publishing Consultant
tgrantham@tengwarsystems.com

What is post-preprocessing?

- Transformation code that runs after the standard OT preprocessing, but before the output-type specific code (i.e. PDF, HTML)
- Uses the `depend.preprocess.post` extension point
- Runs every time you publish, no matter what the transformation type is.

Why use post-preprocessing?

Default <dl> styling (PDF)

Example <dt> Element	This is an example of a <dd> element.
Example <dt> Element	This is an example of a <dd> element.

Preferred <dl> styling (PDF and HTML)

- **Example <dt> Element**: This is an example of a <dd> element.

  Some more of the <dd> element.
- **Another Example <dt> Element**: This is another example of a <dd> element.

  Some more of the <dd> element.

```
<xsl:template match="dl">
    <ul class="- topic/ul ">
        <xsl:copy-of select="@*[name() != 'class']"/>
        <xsl:for-each select="dlentry">
            <li class="- topic/li ">
                <xsl:copy-of select="@*[name() != 'class']"/>
                <b class="+ topic/ph hi-d/b ">
                    <xsl:for-each select="dt">
                        <xsl:apply-templates/>
                        <xsl:if test="position() != last()">
                            <xsl:text> </xsl:text>
                        </xsl:if>
                    </xsl:for-each>
                </b> &amp;#8212; <xsl:apply-templates select="dd[1]/node()"/>
                <xsl:for-each select="dd[position() > 1]/node()">
                    <xsl:choose>
                        <xsl:when test="local-name() = ''">
                            <p class="- topic/p ">
                                <xsl:apply-templates select="."/>
                            </p>
                        </xsl:when>
                        <xsl:otherwise>
                            <xsl:apply-templates select="."/>
                        </xsl:otherwise>
                    </xsl:choose>
                </xsl:for-each>
            </li>
        </xsl:for-each>
    </ul>
</xsl:template>
```

Why use post-preprocessing?

- Convert content to use default styling of other elements
- Eliminate duplicate code across transformation types
- Clean up topics and maps
- Restructure content

How to use post-preprocessing?

- Create the following:
  - The plugin folder
  - The plugin definition file (plugin.xml)
  - The ANT build file for the post-preprocessing target
  - The post-preprocessing XSLT file

```
<project basedir="." default="pcdl2ul" name="pcdl2ulProject">
 <target name="pcdl2ul" description="Convert DITA definition lists to unordered lists">
  <condition property="dita.preprocess.reloadstylesheet.pcdl2ul" value="${dita.preprocess.reloadstylesheet}">
   <not><isset property="dita.preprocess.reloadstylesheet.pcdl2ul"></isset></not>
  </condition>
  <xslt taskname="pcdl2ul" basedir="${dita.temp.dir}" destdir="${dita.temp.dir}" includesfile="${dita.temp.dir}/${fullditatopicfile}" classpathref="dost.class.path"
reloadstylesheet="${dita.preprocess.reloadstylesheet.pcdl2ul}" style="${dita.dir}/plugins/com.tengwarsystems.pc.dl2ul/xsl/pcdl2ul.xsl">
   <mapper type="glob" from="*" to="*.pcdl2ul"></mapper>
   <xmlcatalog refid="dita.catalog"></xmlcatalog>
  </xslt>
  <move todir="${dita.temp.dir}">
   <fileset dir="${dita.temp.dir}" includes="**/*.pcdl2ul"></fileset>
   <mapper type="glob" from="*.pcdl2ul" to="*"></mapper>
  </move>
 </target>
</project>
```

Things to watch out for

- Remember that you are working with normalized topics and maps, not standard source ones.
- If you are converting content from one element to another, remember to change the @class attribute value as well as the element.
- Do any of the other attributes on the source element have to be changed? Or others added?