XSLT Converter Implementation

Overview

The Xslt plugin, makes use of the xalan XSLT processor, in order to translate an XML document to and from a specified XML/xhtml format. The plugin, makes use of user created XSLT stylesheets, which the XSLT processor reads, in order to determine what the translated document will contain. Because this plugin is designed to either produce an xml/xhtml result from a StarWriter document, or to create a StarWriter document given an xml/xhtml document as input, this means that the plugin requires two XSLT stylesheets. The urls of the stylesheets to be used by the XSLT converter, are specified in the Plugin Configuration file.

Using the XSLT Converter

The XSLT converter can be invoked in a similar manner to all other plugins, by invoking the test Driver program as follows:

```
java org.openoffice.xmerge.test.Driver -from <MIMETYPE> -to <MIMETYPE> doc
```

Example 1: Invoking the test driver

At the moment, only one plugin that uses the XSLT converter implementation exists and it is contained in the `htmlsoff.jar`. This jar file contains the following files.

htmlsoff.jar Contents	**Description**
META-INF/converter.xml	Contains info that will be read into a `ConverterInfo` object.
sofftohtml.xsl	This is an XLST stylesheet for converting from StarWriter to html.
Htmltosoff.xsl	This is an XSLT stylesheet for converting from html to StarWriter XML .

Table 1: JAR File Contents

The Plugin Configuration file – converter.xml - file is very important, as it contains information that will be read into the `ConverterInfo` object, and in turn tell the Driver the capabilities of the plugin. The following is a sample Plugin Configuration file:

```
<?xml version="1.0"?>
<!--<!DOCTYPE converters SYSTEM "converter.dtd">-->
<converters>
    <converter type="staroffice/sxw" version="1.0">
        <converter-display-name>
            XSLT Transformation
        </converter-display-name>
        <converter-description>
           Converter which performs xslt transformations
        </converter-description>
        <converter-vendor>OpenOffice.org</converter-vendor>
        <converter-class-impl>
                 org.openoffice.xmerge..xslt.PluginFactoryImpl
         </converter-class-impl>
         <converter-xslt-serialize>
              sofftohtml.xsl
         </converter-xslt-serialize>
         <converter-xslt-deserialize>
              htmltosoff.xsl
         </converter-xslt-deserialize>
        <converter-target type="text/html" />
    </converter>
</converters>
```

Example 2: Sample Plugin Configuration File (converter.xml)

This Plugin Configuration file specifies that when a user wants to convert from one of MIMETYPE specified by the `converter-type` tag to another MIMETYPE specified by the `converter-target` tag or vice versa, the `org.openoffice.xmerge.convert.xml.xslt.PluginFactoryImpl` should be used for the conversion.

The following examples demonstrate the use of the XSLT plugin to perform conversion.

```
java org.openoffice.xmerge.test.Driver -from staroffice/sxw -to text/html somedocument.sxw
```

Example 3: Using the XSLT plugin toconvert from Office to HTML

```
java org.openoffice.xmerge.test.Driver -from text/html -to staroffice/sxw somedocument.html
```

Example 4: Using the XSLT plugin to convert from HTML to Office

The *`converter-xslt-serialize`* and *`converter-xslt-deserialize`* tags, specify the urls of the XSL stylesheets to be used in the conversion.

If the file specified is in the form of a URL (e.g.: `file:///somedir/someotherdir/sofftohtml.xsl` or `http://www.getxslfilesfromhere.com/sofftohtml.xsl`) the XSLT is loaded from the URL like any other resource. If however the file location is not a URL (e.g.: `sofftohtml.xsl` or `somedir/someotherdir/sofftohtml.xsl`), then the plugin assumes that the path specified, is located within the jar file.

The `DocType` tag may be uncommented, if a developer wishes to carry out validation on a Plugin Configuration file, to ensure it conforms to the `converter.dtd.`

Creating a new XSLT plugin

To create a new plugin, the following steps are required:

1. Create stylesheets that describe a particular translation
2. Create a plugin configuration file called `converter.xml` file that contains the correct information.
3. Package the stylesheets and the `converter.xml` file into a jar, ensuring that all of the files are in the correct location, as specified in the file.
4. Also, the `converter.xml` file itself must be contained within the `META-INF/` directory in the jar.
5. Update the `org.openoffice.xmerge.test.ConverterInfoList.properties` file, to include the location of your newly created plugin
6. Run the test Driver program to test the new plugin.