



The Iam Lotus User Group

Creating PDF's using Apache FOP

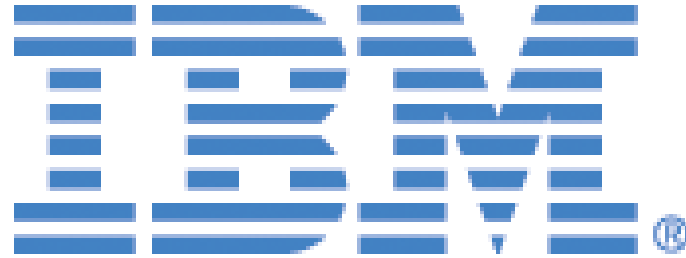
Paul T. Calhoun
NetNotes Solutions Unlimited, Inc

© 2013 by the individual speaker





The Iam Lotus User Group



IamLUG 2013 Sponsors

© 2013 by the individual speaker



Obligatory Introduction Slide

- **Have used Notes/Domino since R2**
- **Certified in Administration and Development on releases: R3-R8.5**
- **2013 IBM Champion**
- **Independent**
 - ♦ **Trainer**
 - ♦ **Administrator**
 - ♦ **Developer**
 - ♦ **Mentor**
 - ▶ **For the past 22 years running NetNotes Solutions Unlimited**

Obligatory Introduction Slide

- **Specializing in**
 - ♦ **Java**
 - ♦ **Web Services**
 - ♦ **XPages**
 - ♦ **J2EE (WebSphere)**
 - ♦ **Eclipse**
 - ♦ **Rational Developer**
 - ♦ **And....**

Obligatory Introduction Slide

- Raising Grand Kids



Agenda

- *What is APACHE FOP ?*
- **Setup and configure FOP for development and production**
- **The XML Source**
- **The StyleSheet**
- **The Code**
- **Summary – Q&A**

What is APACHE FOP

- **First APACHE**

- ♦ The Apache Software Foundation (www.apache.org) is an Open Source (Yes that means FREE) consortium of companies and developers who donate time and resources to developing tools (primarily Java based) that simplify many tasks that developers are faced with today
- ♦ It's a lot like OpenNTF, but for all developers
- ♦ IBM is a major contributor/supporter

What is APACHE FOP (cont)

- ♦ If you develop using Java code (and you should be !) than you owe it to yourself to spend some time reviewing the projects on this site
- ♦ Some I use all the time
 - ▶ Xerces
 - ▶ Xalan
 - ▶ XML Graphics (FOP)
 - ▶ POI
 - ▶ Commons

So what is FOP?

- **A sub-project of the XML Graphics project at APACHE**
- **FOP stands for Formatting Objects Processor**
 - ♦ **This is a print formatter driven by XSL FO (Formatted Objects)**
 - ♦ **An API that**
 - ▶ **Reads an FO Tree**
 - ▶ **Renders the resulting pages to a specified output**
 - ***PDF (Primary)***
 - ***PS,PCL***
 - ***AFP***
 - ***XML***
 - ***Print, AWT, PNG***

Um.. How about that last slide in English !!

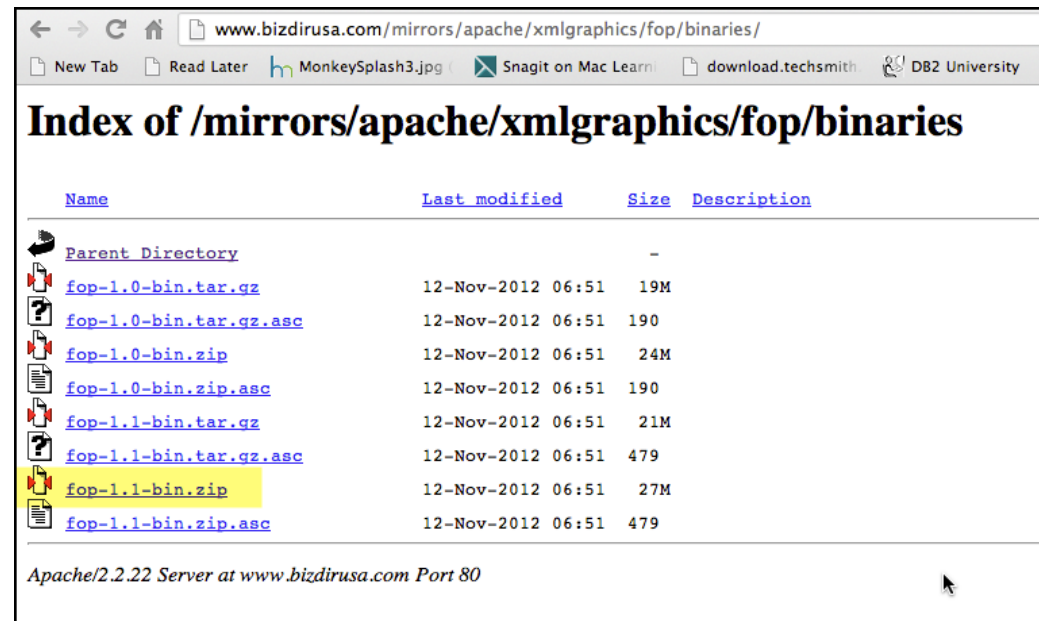
- **It's a Java based toolset that lets YOU (the developer) create solutions that allow your End Users (the Bain of all existence) to create PDF documents from Notes content for FREE !!!**
- **Yes FREE!!!**
 - ♦ **Like Free Beer !**
 - ♦ **Ok, Free is relative.**
 - ♦ **You are going to have to invest some time, but I'm going to give you a working framework that you can implement out of the box.**
 - ♦ **For Free!!! (Who doesn't like Free ?)**

Agenda










- **What is APACHE FOP ?**
- ***Setup and configure FOP for development and production***
- **The XML Source**
- **The StyleSheet**
- **The Code**
- **Summary – Q&A**

Download FOP from APACHE

- Download the Java toolset from APACHE
 - ♦ <http://xmlgraphics.apache.org/fop/download.html>
- For reference you can use these slides but the Quick Start Guide is a nice reference reminder of how to set up everything
 - ♦ <http://xmlgraphics.apache.org/fop/quickstartguide.html>



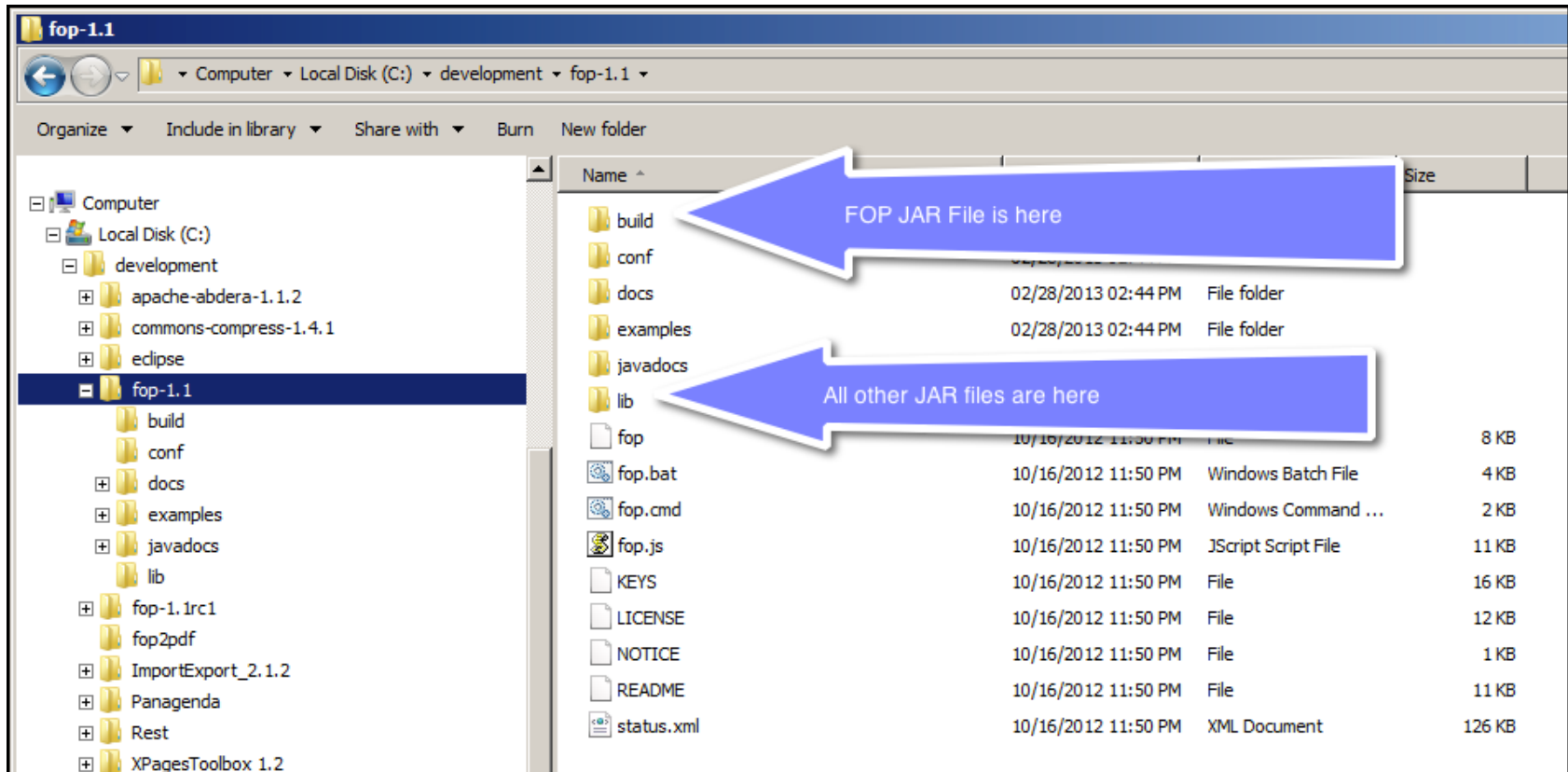
The screenshot shows a web browser window with the address bar displaying `www.bizdirusa.com/mirrors/apache/xmlgraphics/fop/binaries/`. The browser tabs include "New Tab", "Read Later", "MonkeySplash3.jpg", "Snagit on Mac Learn...", "download.techsmith", and "DB2 University". The main content area displays the title "Index of /mirrors/apache/xmlgraphics/fop/binaries" and a table of files.

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 fop-1.0-bin.tar.gz	12-Nov-2012 06:51	19M	
 fop-1.0-bin.tar.gz.asc	12-Nov-2012 06:51	190	
 fop-1.0-bin.zip	12-Nov-2012 06:51	24M	
 fop-1.0-bin.zip.asc	12-Nov-2012 06:51	190	
 fop-1.1-bin.tar.gz	12-Nov-2012 06:51	21M	
 fop-1.1-bin.tar.gz.asc	12-Nov-2012 06:51	479	
 fop-1.1-bin.zip	12-Nov-2012 06:51	27M	
 fop-1.1-bin.zip.asc	12-Nov-2012 06:51	479	

Apache/2.2.22 Server at www.bizdirusa.com Port 80

Expand the ZIP file

- Expand the Zip file to a directory that you have access to from you Notes Designer Client and Development(Yes I said DEVELOPMENT) Domino Server.



Get the JAR files

- **The following jar files will be needed for development and production**
 - ◆ **From the build folder**
 - ▶ **FOP.jar**
 - ◆ **From the lib folder (Depending on the version the “xxx” will be different)**
 - ▶ **avalon-framework-xxx.jar**
 - ▶ **batick-all-xxx.jar**
 - ▶ **commons-io-xxx.jar**
 - ▶ **xml-apis-ext-xxx.jar**
 - ▶ **xmlgraphic-commons-xxx.jar**

Development Setup

- **You have two choices when configuring an application to use third party Java libraries**
 - ◆ **Put all the JARS in the NSF**
 - ◆ **Put the JARS on Host File System**

Put all the JARS in the NSF

- **Pros**

- ♦ Makes the application more portable.
- ♦ Can be “deployed” to the test server and production by replication

- **Cons**

- ♦ JARS are only accessible by code in the containing NSF
- ♦ If many NSF's use this solution then maintenance can become difficult

Put the JARS on Host File System

- **Pros**

- ◆ Easier to maintain code/upgrade code for all applications that use it

- **Cons**

- ◆ Harder to deploy (insert snarky Evil Admin comment here)

Development Setup

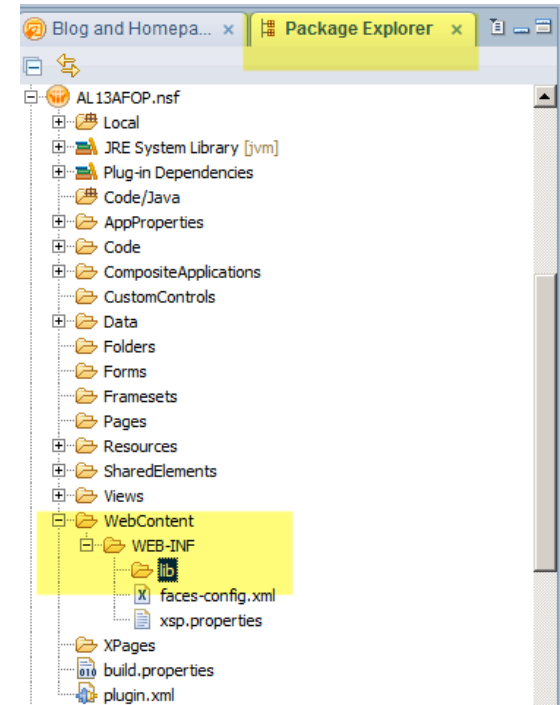
- I prefer to deploy the JARS to the Host File system
 - ◆ (Insert self referencing Evil Admin Comment here)
 - ◆ If the code that CALLS the classes from the JARS is
 - ▶ A Java Agent, Servlet, Java Class or Java Code Element (available in 8.5.3 and above)
 - *Deploy the JARS to the <installDir>/jvm/lib/ext folder*
 - ▶ SSJS Code from an XPage
 - *Deploy the JARS to the <installDir>/xsp folder*
 - ◆ The JARS will have to be deployed to the Notes Designer Client AND the development/production servers !!
 - ◆ If the Client or Server is running then they will have to be restarted in order for your code to recognize that they are there.
 - ▶ **This is the step you will forget. Just say'in.**

Development Setup

- **The JARS can also be added to the NSF container when developing XPages.**
 - ♦ **This is the option I'm using so you have a self contained demonstration/example system you can play with locally or on a TEST Server.**

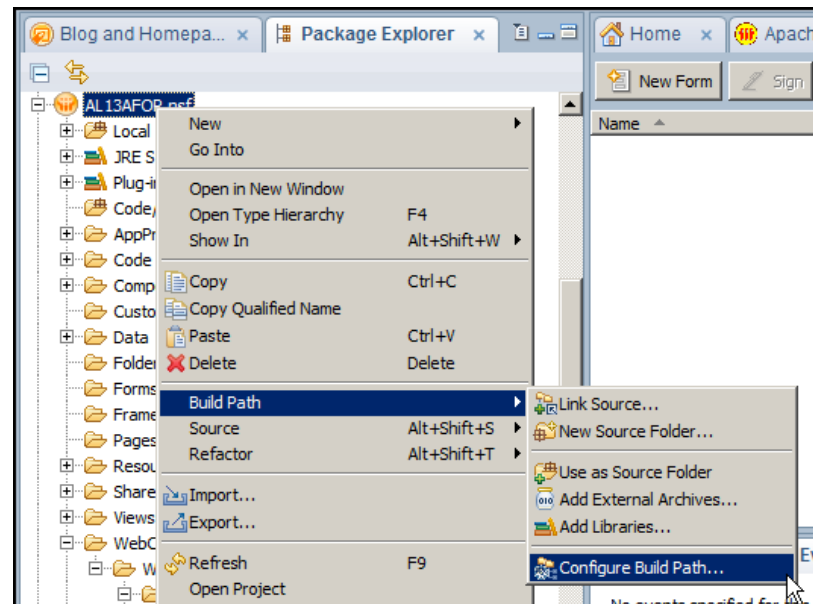
Development Setup

- In 8.5.3 and below
 - ◆ In the Application that will contain the XPage Code
 - ▶ From the XPages perspective switch to the Package Explorer View
 - ▶ Expand the WebContent/WEB-INF folder
 - *Create a folder named “lib”*
 - ▶ Import (you can also drag and drop) the JARS to the lib folder



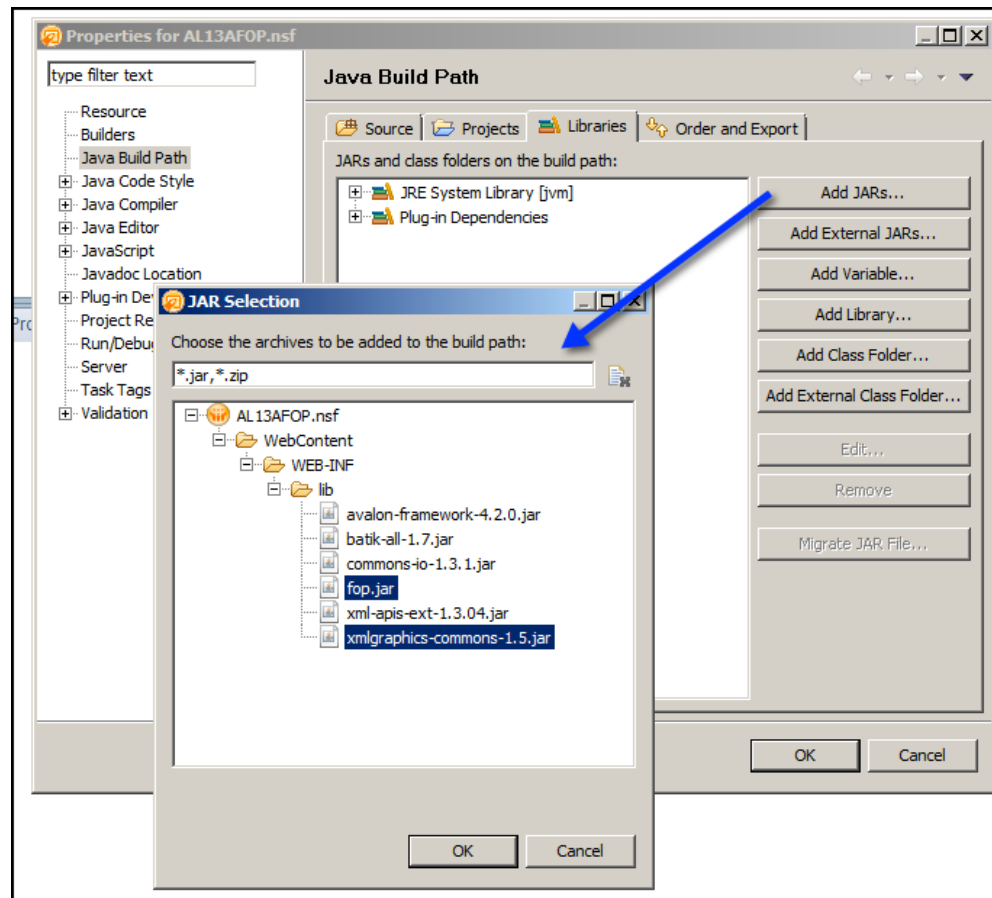
Add to the Build Path

- In order for your code to find the classes in the JAR files some will need to be added to the Build Path so the code can be “compiled”
- In the package explorer view, right click on the project (nsf) and choose **Build Path > Configure Build Path**



Add JARS to the build path

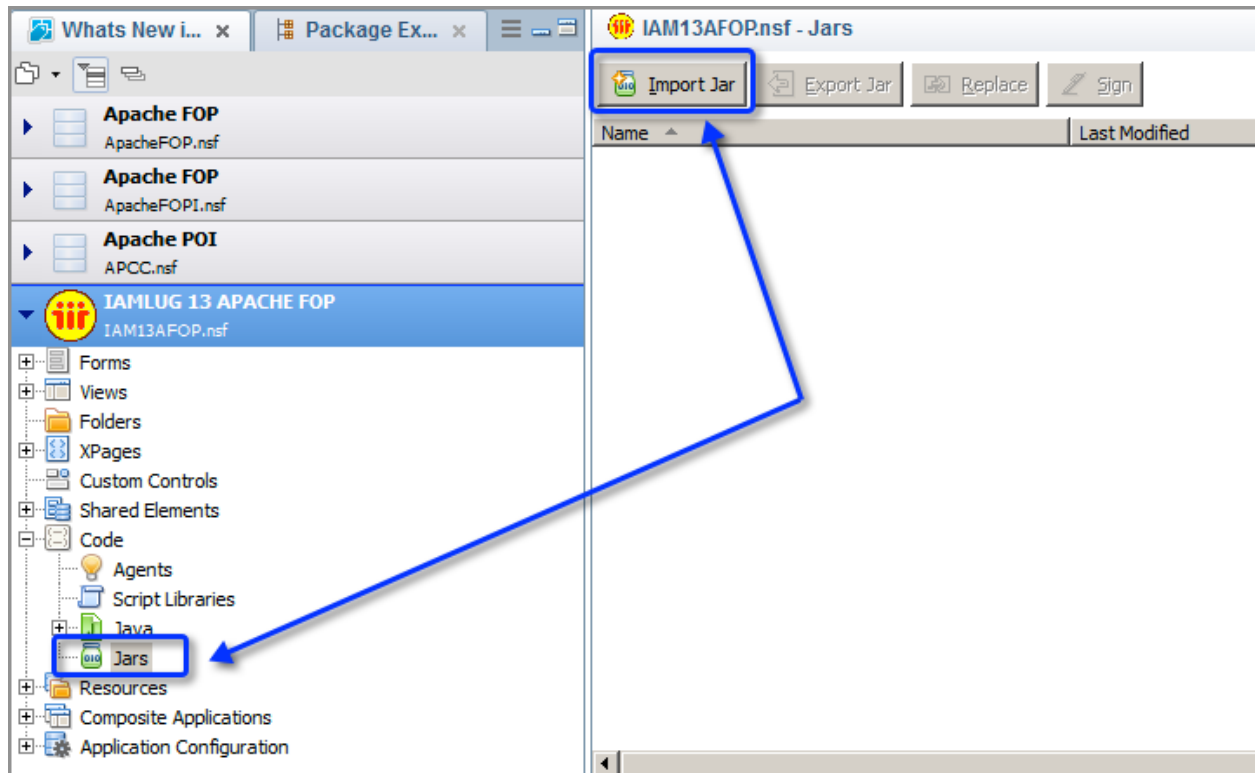
- Add the following JAR files to the build path
 - ◆ FOP.jar
 - ◆ xmlgraphic-commons-xxx.jar



Development Setup

- In 9.0

- ◆ Import the Jars to the new “JAR” design element
- ◆ This will automatically add it to the class/build path
 - ▶ No other configuration is required

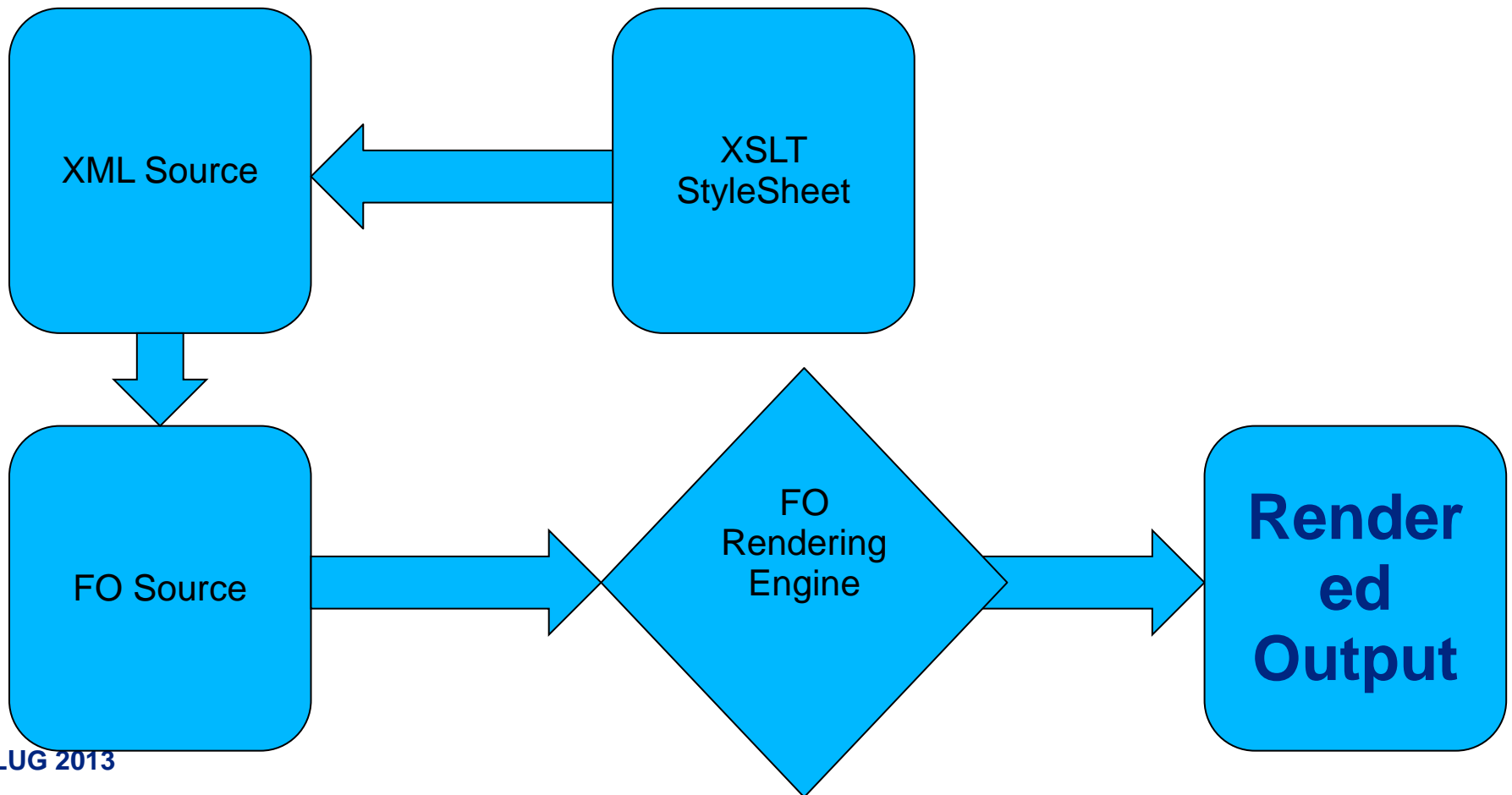


Agenda

- **What is APACHE FOP ?**
- **Setup and configure FOP for development and production**
- *The XML Source*
- **The StyleSheet**
- **The Code**
- **Summary – Q&A**

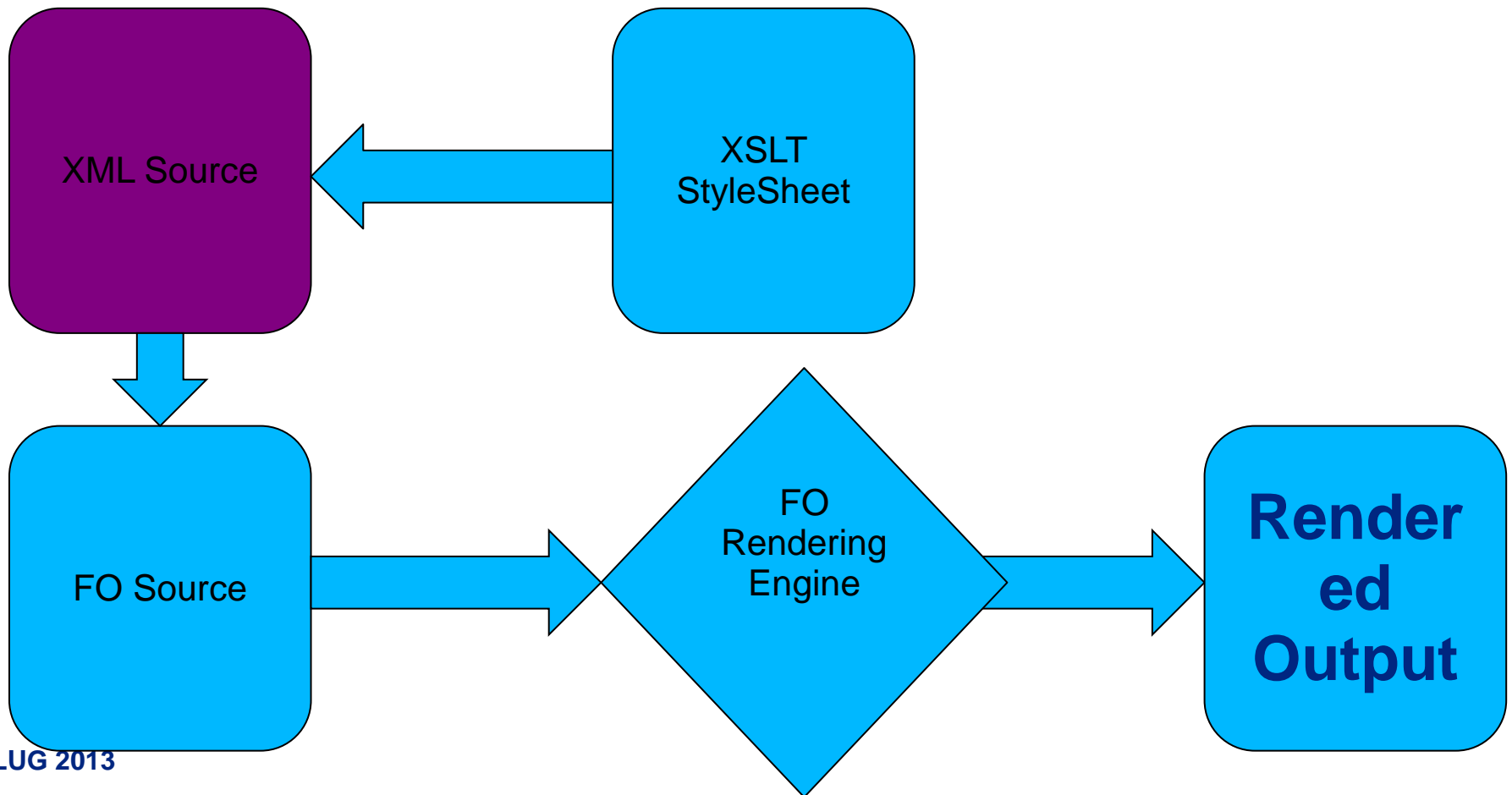
The Process

- The great thing about FOP is it is a highly duplicable design pattern



The Process

- It all starts with an XML Document (can be from disk or in memory)



The XML

- **The XML can be**
 - ♦ **A Static Document**
 - ♦ **The output from ?ReadViewEntries**
 - ▶ **Appended to the end of a Domino View URL**
 - ♦ **The output from generateXML**
 - ▶ **Method of the Notes Document class**
 - ♦ **The results of running an XAgent**
 - ♦ **The results of running an Agent**
 - ♦ **The results of running a Web Service**

The XML

- The source of the XML is not as important as the **FORMAT** and **CONSISTANCY** of the XML
 - ◆ The XML must be
 - ▶ **Well formed**
 - ▶ **Optionally valid**
 - ◆ **Make sure that if you *do not* control the source that you have an SLA with the source provider that includes them providing the XML Schema AND changes to the XML Schema in enough advance that you have time to test it.**

The XML

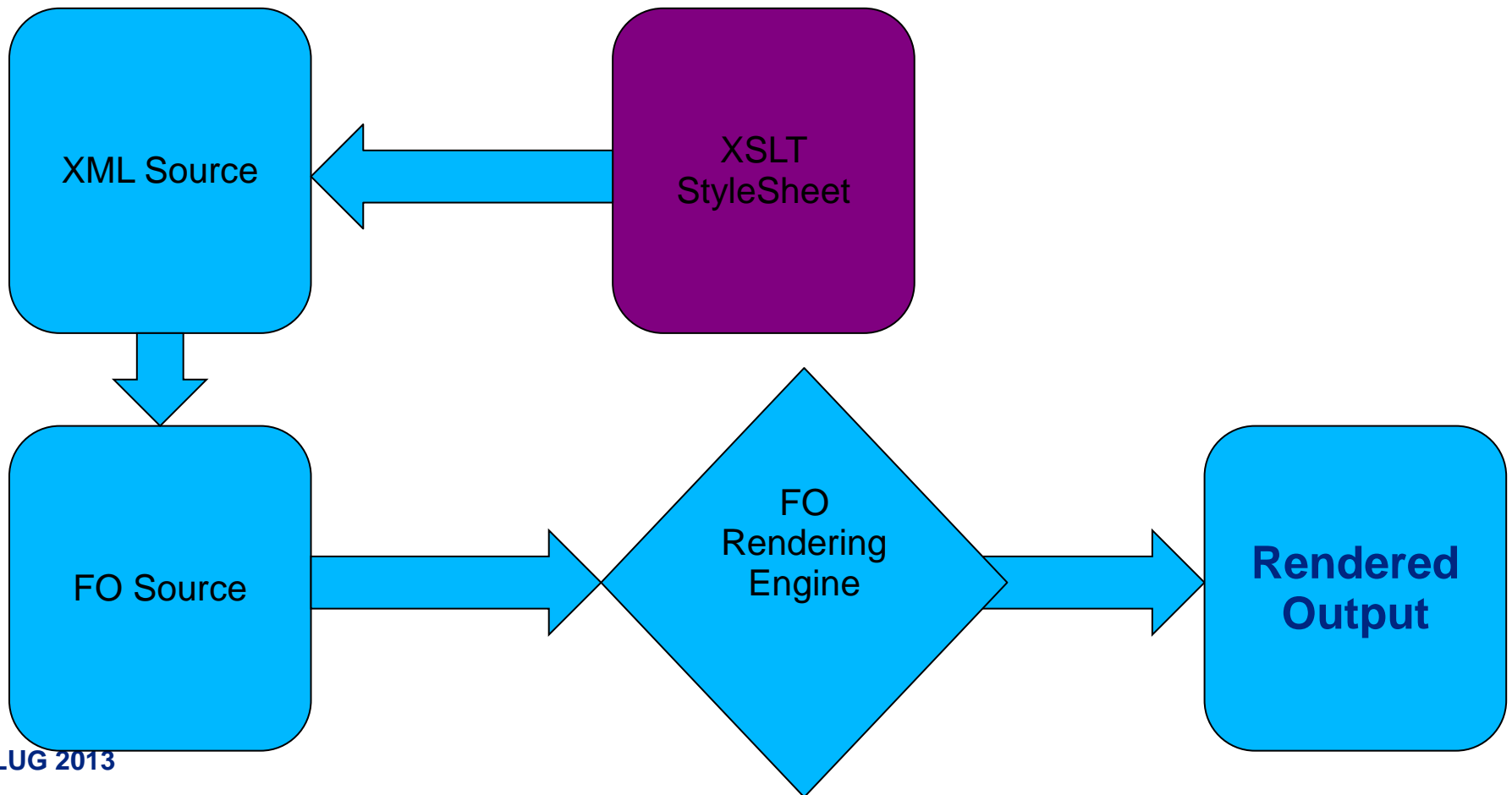
- **Using the ?ReadViewEntries option is HIGHLY duplicable**
 - ♦ Once you have the stylesheet (and I'm providing that to you) that transforms the source XML to the FO XML then ANY view source can be passed to the code to produce a PDF of the view
- **The other options require an XSLT stylesheet that is specific to transforming the specified XML to FO XML.**
 - ♦ This is not as flexible, but once the base stylesheet is created it can be stored in a notes document that is editable by a non-developer
 - ▶ This option allows changes to colors, fonts etc without developer intervention and re-compiling/re-deploying the code

Agenda

- **What is APACHE FOP ?**
- **Setup and configure FOP for development and production**
- **The XML Source**
- ***The StyleSheet***
- **The Code**
- **Summary – Q&A**

The Stylesheet

- The stylesheet that is used is an XSLT document (written in XML) that uses the FOP tags from the tag library

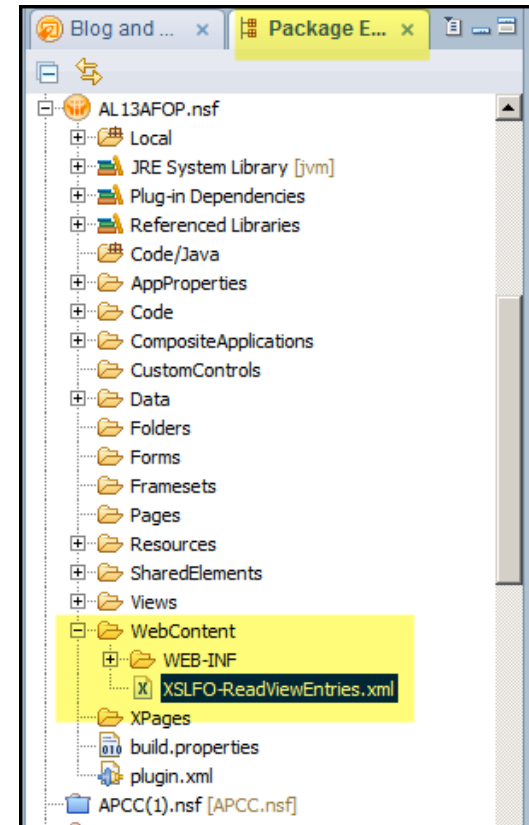


The XSL FO Stylesheet

- **This is by far the most challenging part of this solution**
 - ◆ Stylesheet creators must be able to create and edit XSLT stylesheets (Doh!)
- **The Tags used in the style sheet are not documented at the FOP site**
 - ◆ The good news is they ARE documented at the w3 schools site
 - ▶ <http://www.w3schools.com/xslfo/default.asp>
 - ◆ The other good news is they are documented in the form of a tutorial !!
 - ◆ You can follow the tutorial to examine how to create a base XSL FO style sheet
 - ▶ You can also copy/paste this example code directly to your XSL FO style sheet to get started !!

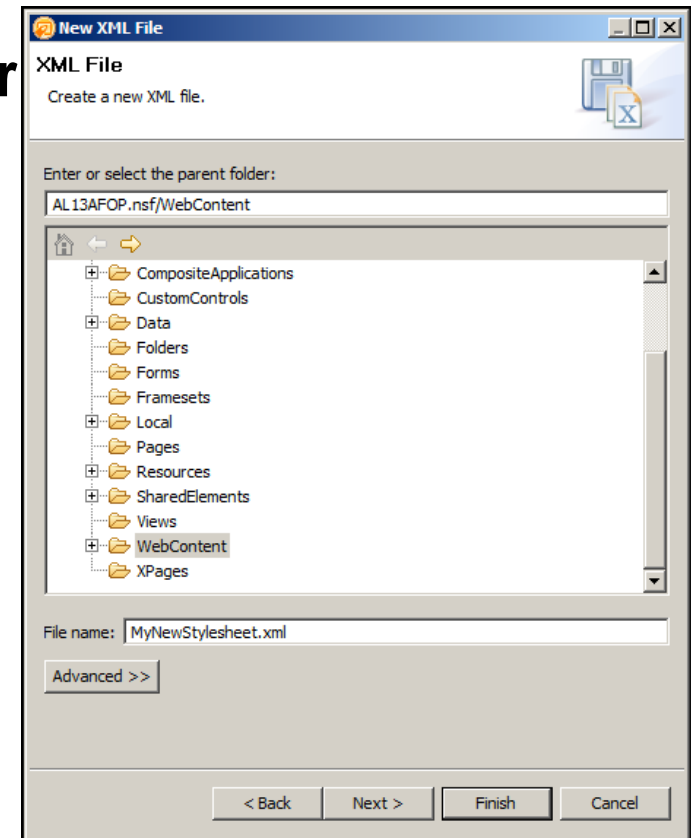
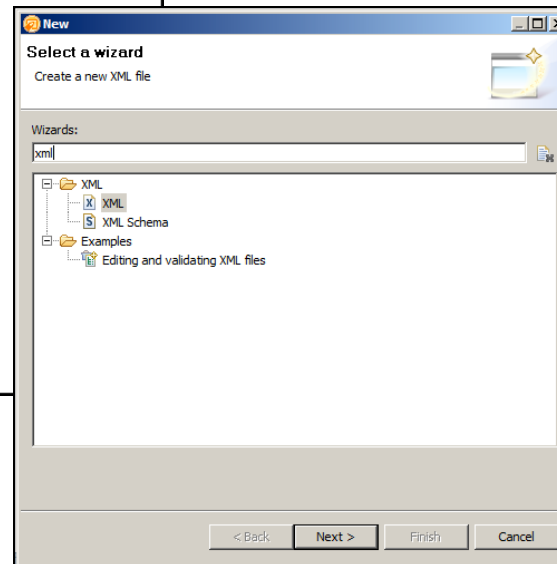
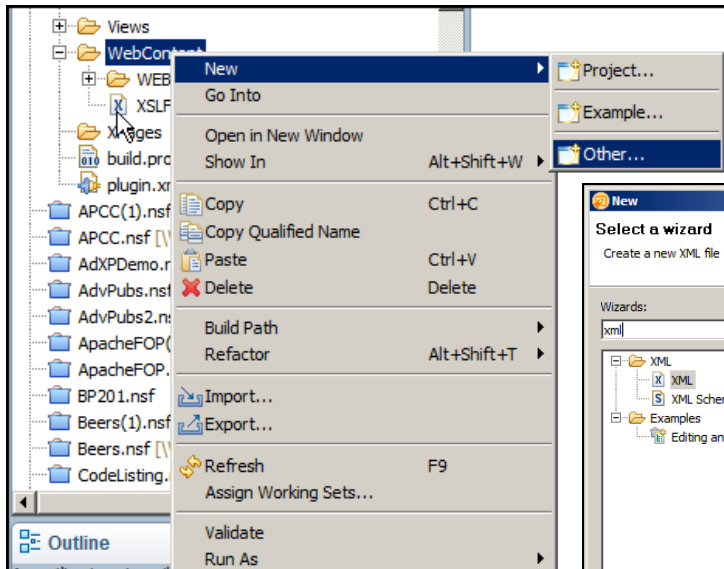
XML Editor in DDE

- I find it easier to create/edit my XSL FO style sheets in a proper XML editor
 - ♦ There are many out there and available, but I use the one in Eclipse
 - ♦ Wait, Eclipse you say !! Isn't Domino Designer BUILT on top of Eclipse?
 - ▶ Yes, yes it is.
 - ▶ You can use the XML Editor in DDE !!!!
- Switch to the Package Explorer View
- Create a new Static Web Project OR create an new XML file in one of your existing NSF projects
- I put them in the NSF in the WebContent folder



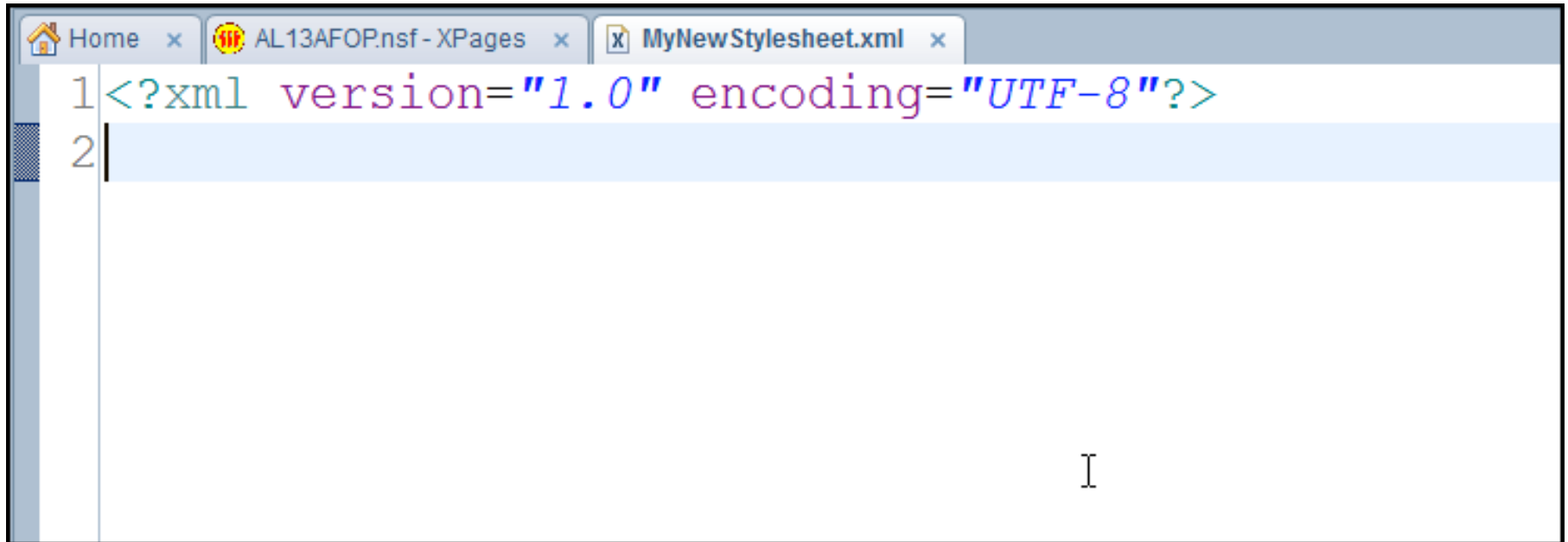
Create an XSLT Stylesheet in DDE

- Right click on the WebContent folder
 - ◆ Select New | Other
 - ◆ Filter on XML
 - ◆ Select XML and click Next
 - ◆ Add new file to the WebContent folder



XML Document open in DDE

- The new XML document will open in DDE



The screenshot shows a web browser window with three tabs: 'Home', 'AL13AFOP.nsf - XPages', and 'MyNewStylesheet.xml'. The active tab displays an XML document in DDE mode. The document content is as follows:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2
```

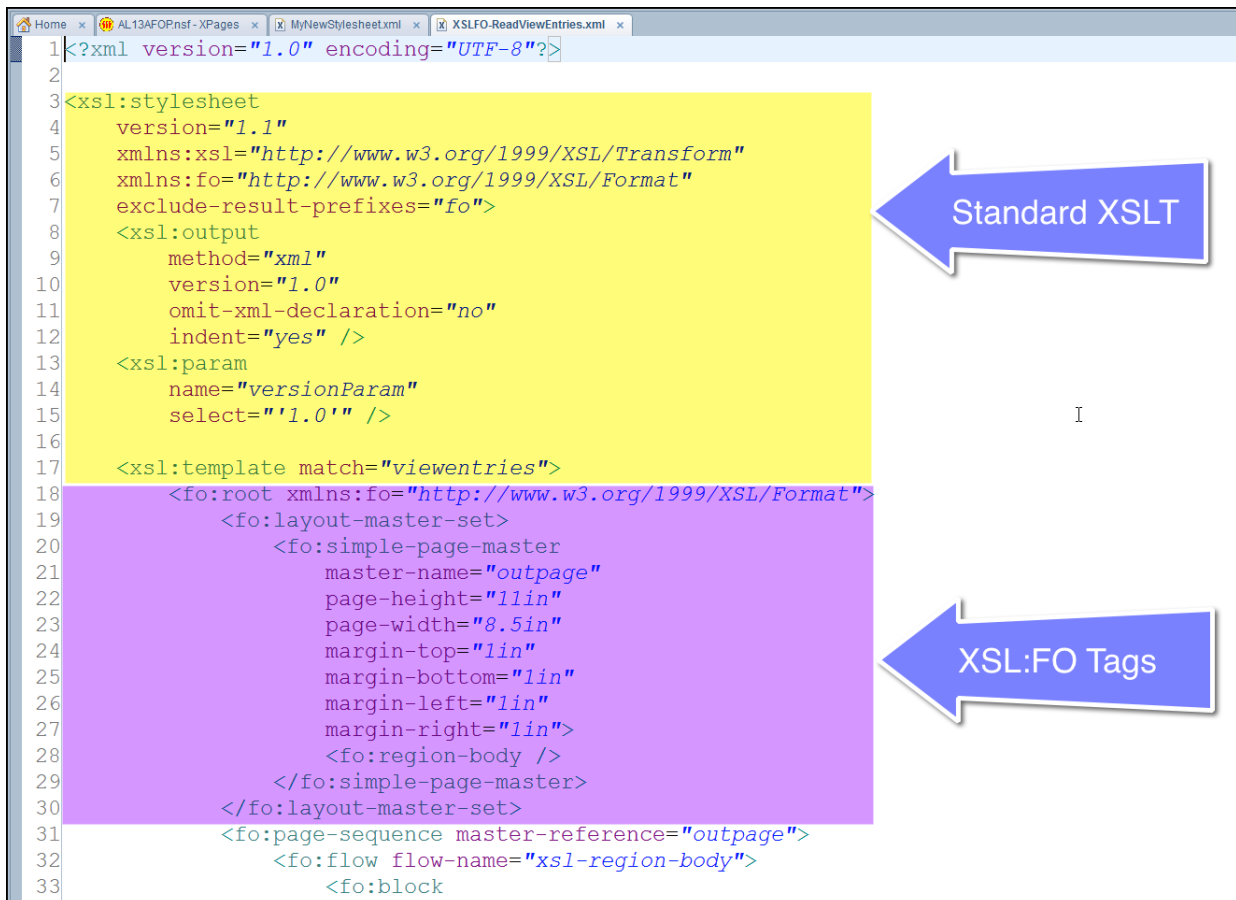
The text is color-coded: '<?' is green, 'xml' is purple, 'version=' is purple, '1.0' is purple, 'encoding=' is purple, and 'UTF-8' is blue. A vertical cursor is positioned at the end of the second line.

Or you can use Eclipse

- **Download the version that has the Web Tools Plugin (WTP)**
- **I usually download the one that supports J2EE development**
- **The benefit of this option is there is an XSLT editor included not just an XML editor**
- **Also you can train “power-users” to use eclipse to build, edit, maintain XSLT stylesheets for the purpose of maintaining their own output without the need for them to have designer**
 - ♦ **This might sound difficult, but it is significantly easier to train both developers and power users on XSLT than it is Java !!**

The XSLT

- The Stylesheet is made up of a combination of XSLT and XSL:FO tags



The image shows a screenshot of an XML editor with several tabs open. The active tab displays XML code. The code is divided into two sections: a yellow highlighted section for XSLT and a purple highlighted section for XSL:FO tags. Blue arrows point from the text labels to their respective code sections.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <xsl:stylesheet
4   version="1.1"
5   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
6   xmlns:fo="http://www.w3.org/1999/XSL/Format"
7   exclude-result-prefixes="fo">
8   <xsl:output
9     method="xml"
10    version="1.0"
11    omit-xml-declaration="no"
12    indent="yes" />
13   <xsl:param
14     name="versionParam"
15     select="'1.0'" />
16
17   <xsl:template match="viewentries">
18     <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
19       <fo:layout-master-set>
20         <fo:simple-page-master
21           master-name="outpage"
22           page-height="11in"
23           page-width="8.5in"
24           margin-top="1in"
25           margin-bottom="1in"
26           margin-left="1in"
27           margin-right="1in">
28           <fo:region-body />
29         </fo:simple-page-master>
30       </fo:layout-master-set>
31       <fo:page-sequence master-reference="outpage">
32         <fo:flow flow-name="xsl-region-body">
33           <fo:block
```

Standard XSLT

XSL:FO Tags

Structure of the FO Tags

- The XSL:FO tags are all about the layout of the “printed” page
- XSL:FO tags always start with “root”
 - ♦ Followed by a “layout master”
 - ▶ Followed by a “page master”
 - ▶ Then a series of page sequences that contain
 - *Flows*
 - *Blocks*

```
<xsl:template match="viewentries">
  <fo:root>
    <fo:layout-master-set>
      <fo:simple-page-master
        master-name="outpage"
        page-height="11in"
        page-width="8.5in"
        margin-top="1in"
        margin-bottom="1in"
        margin-left="1in"
        margin-right="1in">
        <fo:region-body />
      </fo:simple-page-master>
    </fo:layout-master-set>

    <fo:page-sequence master-reference="outpage">
      <fo:flow flow-name="xsl-region-body">
        <fo:block
          font-size="16pt"
          font-weight="bold"
          space-after="5mm">
            View data from ?{readViewEntries}
          </fo:block>
          <fo:block
            font-size="12pt"
            space-after="5mm">
            <xsl:value-of select="$versionParam" />
          </fo:block>
          <fo:block font-size="10pt">
            <fo:table
              table-layout="fixed"
              width="100%"
              border-collapse="separate">
              <fo:table-column column-width="2.25in" />
              <fo:table-column column-width="1in" />
              <fo:table-column column-width="1in" />
              <fo:table-column column-width="3in" />
              <fo:table-header>
                <fo:table-row>
                  <fo:table-cell>
                    <fo:block
                      font-weight="bold"
                      text-align="left"
                    />
                  />
                />
              />
            />
          />
        />
      />
    />
  />
</xsl:template>
```

The Layout and Page Master

- These tags define the output page
 - ♦ Height
 - ♦ Width
 - ♦ Margins
 - ♦ etc


```
<xsl:template match="viewentries">
  <fo:root>
    <fo:layout-master-set>
      <fo:simple-page-master
        master-name="outpage"
        page-height="11in"
        page-width="8.5in"
        margin-top="1in"
        margin-bottom="1in"
        margin-left="1in"
        margin-right="1in">
        <fo:region-body />
      </fo:simple-page-master>
    </fo:layout-master-set>
  </fo:root>
</xsl:template>
```

The Page Sequence tag

- References the page master set tag to get its output constraints

```
<xsl:template match="viewentries">
  <fo:root>
    <fo:layout-master-set>
      <fo:simple-page-master
        master-name="outpage"
        page-height="11in"
        page-width="8.5in"
        margin-top="1in"
        margin-bottom="1in"
        margin-left="1in"
        margin-right="1in">
        <fo:region-body />
      </fo:simple-page-master>
    </fo:layout-master-set>

    <fo:page-sequence master-reference="outpage">
      <fo:flow flow-name="xsl-region-body">
        <fo:block
          font-size="16pt"
          font-weight="bold"
          space-after="5mm">
            View data from ?{ReadViewEntries}
          </fo:block>
        </fo:block>
      </fo:flow>
    </fo:page-sequence>
  </fo:root>
</xsl:template>
```



The Page Content

- The Page content is output using a series of “flows” and “blocks”
- A “flow” contains a series of “blocks”
- A “block” is roughly equivalent to a paragraph on the page
- Blocks can contain other constraining tags like the “table” tag

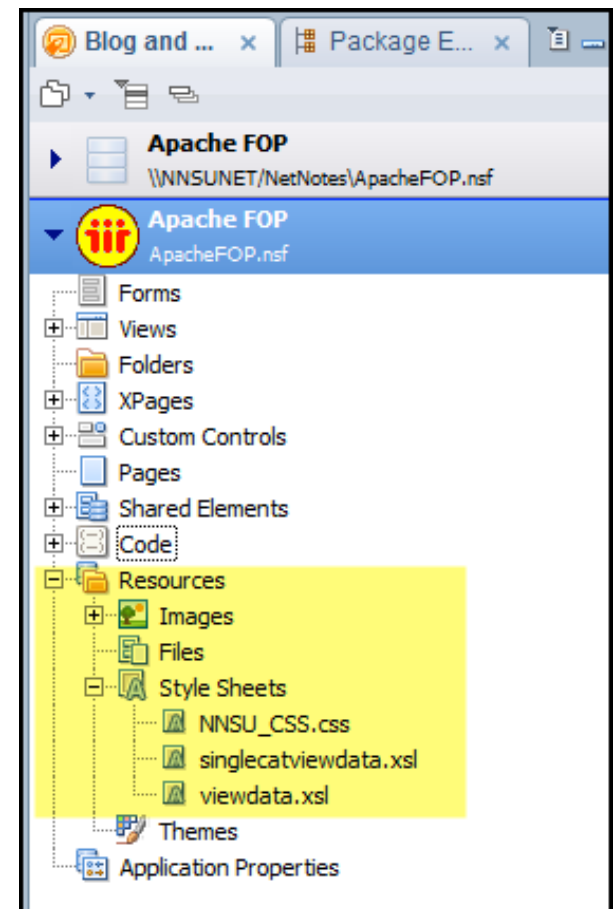
```
<fo:page-sequence master-reference="outpage">
  <fo:flow flow-name="xsl-region-body">
    <fo:block
      font-size="16pt"
      font-weight="bold"
      space-after="5mm">
      View data from ?{ReadViewEntries}
    </fo:block>
    <fo:block
      font-size="12pt"
      space-after="5mm">
      <xsl:value-of select="$versionParam" />
    </fo:block>
    <fo:block font-size="10pt">
      <fo:table
        table-layout="fixed"
        width="100%"
        border-collapse="separate">
        <fo:table-column column-width="2.25in" />
        <fo:table-column column-width="1in" />
        <fo:table-column column-width="1in" />
        <fo:table-column column-width="3in" />
        <fo:table-header>
          <fo:table-row>
            <fo:table-cell>
              <fo:block
                font-weight="bold"
                text-align="left">
```

Where are the stylesheets Stored

- **After the stylesheet is created there are two options**
 - ◆ **Save the stylesheet as a design resource (A stylesheet)**
 - ◆ **Save the stylesheet in a document that is accessible from the notes client**
 - ▶ **This allows editing/maintenance of the stylesheets without the need of a designer client**
- **The option you choose will depend upon level of expertise of the folks you have back at your house**

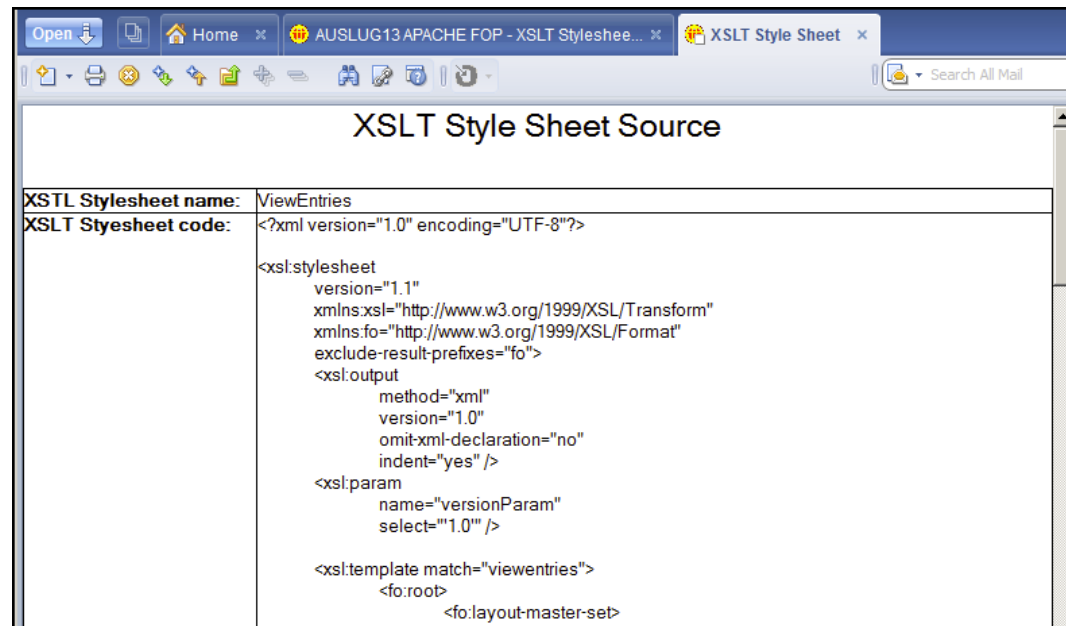
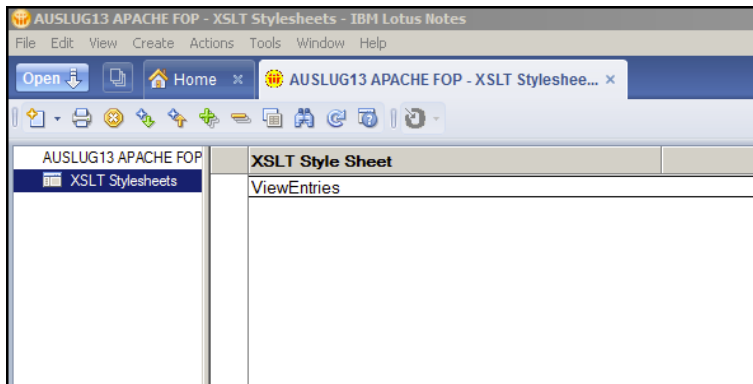
Storing Stylesheets in Designer

- The XSLT Sheets can be stored in DDE in the Style Sheets folder contained in the resources folder



Alternately Store in documents

- You can create a
 - ♦ Form
 - ♦ View
- to store Notes documents that contain the XSLT stylesheets
- Your code will “lookup” the stylesheet when applying it to the XML source



Agenda

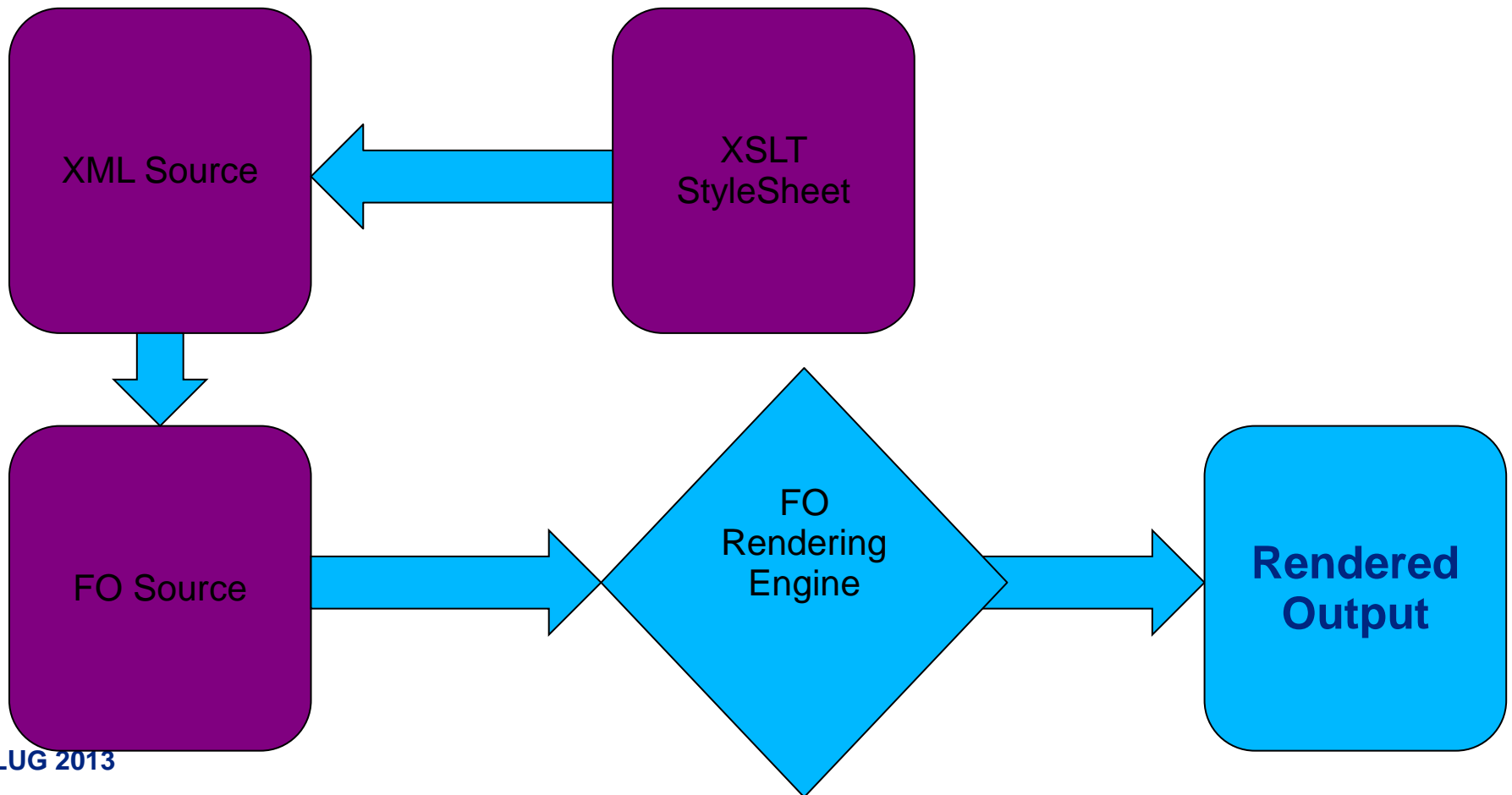
- **What is APACHE FOP ?**
- **Setup and configure FOP for development and production**
- **The XML Source**
- **The StyleSheet**
- ***The Code***
- **Summary – Q&A**

The Code

- **Now that you have the XML Source and the XSLT stylesheet you are ready to write some code !!! (Yea !!)**
- **APACHE FOP is a JAVA Library**
 - ♦ **Yes this means you have to code the solution using Java !!**
- **Can be coded as**
 - ♦ **Java Agents**
 - ♦ **Java Code elements**
 - ♦ **Coded in SSJS (like an XAgent)**

Creating the FO Source

- The code will take the XSLT stylesheet, apply it to the XML source and produce the XSL:FO that is used by the rendering engine to produce the PDF



The Code

- The following is a simple example with very few moving parts
- Contains hard coded URL's (NOT A BEST PRACTICE)

```
package com.nnsu.util.fop;

import java.io.OutputStream;

import javax.xml.transform.*;

import org.apache.fop.apps.*;

public class DominoXMLFO2PDF {
    public static void getPDF(OutputStream pdfout) {
        try {

            //Create xml and xslt source urls
            Source xmlSource, xsltSource;
            String xml, xslt;
            xml = "http://localhost:8080/sample.nsf/Main?ReadViewEntries&count=999&ResortAscending=2";
            xslt = "http://localhost:8080/IAM13AFOP.nsf/ViewEntries.xsl";
            //Create Stream Sources from the URL's
            xmlSource = new StreamSource(xml);
            xsltSource = new StreamSource(xslt);

            // configure fopFactory as desired
            final FopFactory fopFactory = FopFactory.newInstance();
            // configure foUserAgent as desired
            FOUUserAgent foUserAgent = fopFactory.newFOUserAgent();
```

**XML
Source**

**XSLT
Source**

The Code

- The following code takes advantage of a process called “pipelining” that uses the in memory FOP document rather than having to write the transformation results to disk and process those results

```
// Setup output

// Construct fop with desired output format
Fop fop = fopFactory.newFop(MimeConstants.MIME_PDF, foUserAgent, pdfout);

// Setup XSLT
TransformerFactory factory = TransformerFactory.newInstance();
Transformer transformer = factory.newTransformer(xsltSource);

// Set the value of a <param> in the stylesheet
transformer.setParameter("versionParam", "Company List");

// Setup input for XSLT transformation
Source src = xmlSource;

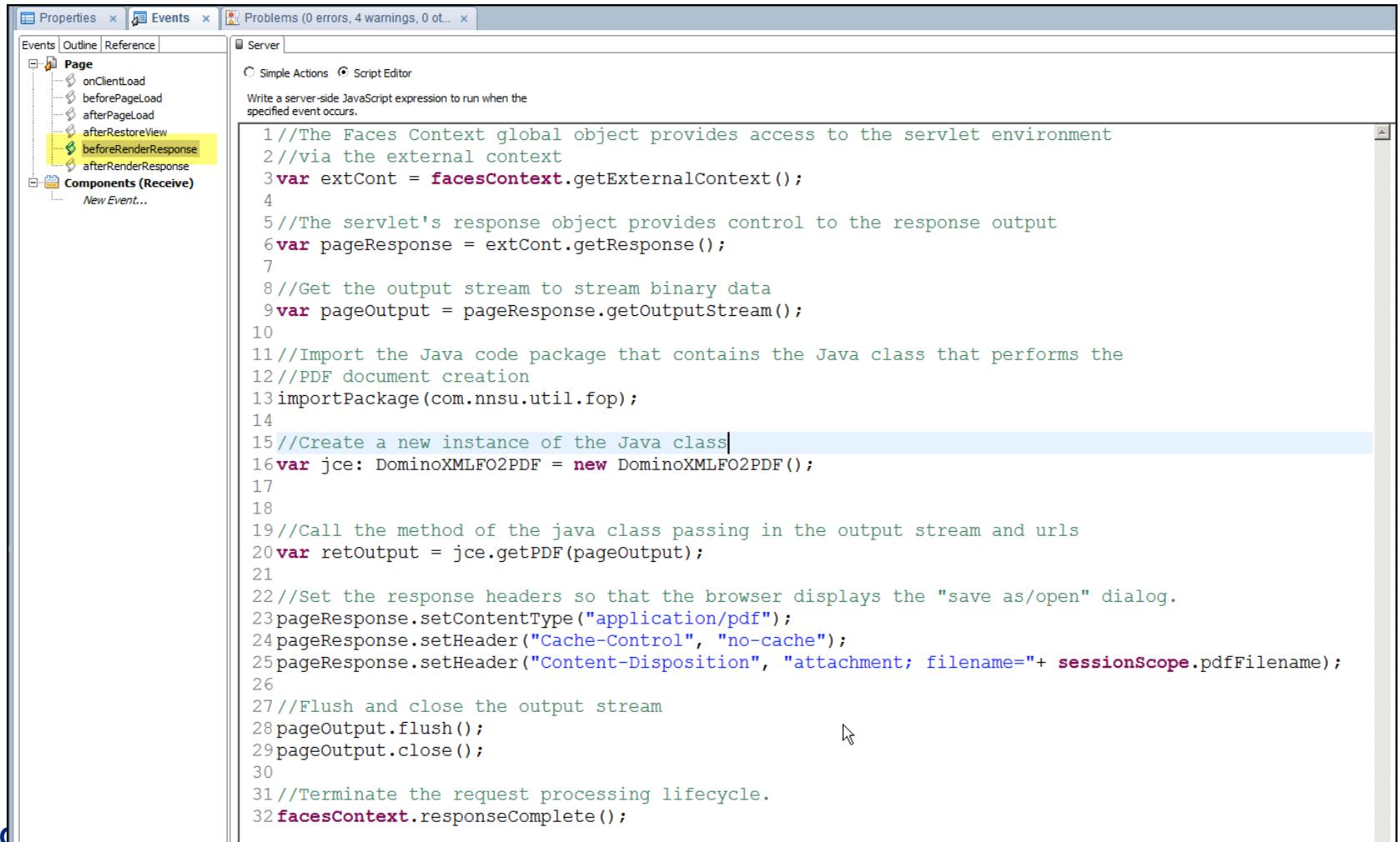
// Resulting SAX events (the generated FO) must be piped through to FOP
Result res = new SAXResult(fop.getDefaultHandler());

// Start XSLT transformation and FOP processing
transformer.transform(src, res);
```

Here we use “pipelining” to transform the XML and take the in memory XSL:FO and process it by the FOP engine.

The Code

- The “XAgent” that calls the Java Code



The screenshot shows a web IDE interface with a left-hand navigation pane and a main code editor. The navigation pane shows a tree view with 'Page' and 'Components (Receive)' folders. Under 'Page', several events are listed, with 'beforeRenderResponse' highlighted in yellow. The main editor displays a Java code snippet for a server-side JavaScript expression. The code is as follows:

```
1 //The Faces Context global object provides access to the servlet environment
2 //via the external context
3 var extCont = facesContext.getExternalContext();
4
5 //The servlet's response object provides control to the response output
6 var pageResponse = extCont.getResponse();
7
8 //Get the output stream to stream binary data
9 var pageOutput = pageResponse.getOutputStream();
10
11 //Import the Java code package that contains the Java class that performs the
12 //PDF document creation
13 importPackage(com.nnsu.util.fop);
14
15 //Create a new instance of the Java class
16 var jce: DominoXMLFO2PDF = new DominoXMLFO2PDF();
17
18
19 //Call the method of the java class passing in the output stream and urls
20 var retOutput = jce.getPDF(pageOutput);
21
22 //Set the response headers so that the browser displays the "save as/open" dialog.
23 pageResponse.setContentType("application/pdf");
24 pageResponse.setHeader("Cache-Control", "no-cache");
25 pageResponse.setHeader("Content-Disposition", "attachment; filename="+ sessionScope.pdfFilename);
26
27 //Flush and close the output stream
28 pageOutput.flush();
29 pageOutput.close();
30
31 //Terminate the request processing lifecycle.
32 facesContext.responseComplete();
```

The XPage that calls the XAgent

- In the onClick event of a button the XAgent is “executed” to produce the PDF output

The screenshot displays the IBM Design Center interface for configuring an XPage. The top browser window shows the URL 'FOPReports - XPage'. The main design area is titled 'APACHE FOP Reports' and contains the text: 'Click the Stream PDF via FOP button below to generate the Customer Report'. Below this text is a text input field labeled 'PDF File Name:' with the value 'pdfFilename'. A button labeled 'Stream PDF via FOP' is highlighted with a yellow box. The bottom panel shows the 'Events' tab for the 'onClick' event, with the following JavaScript code:

```
1//Call the Stream PDF XAgent
2context.redirectToPage("StreamFOPPDF.xsp");
```

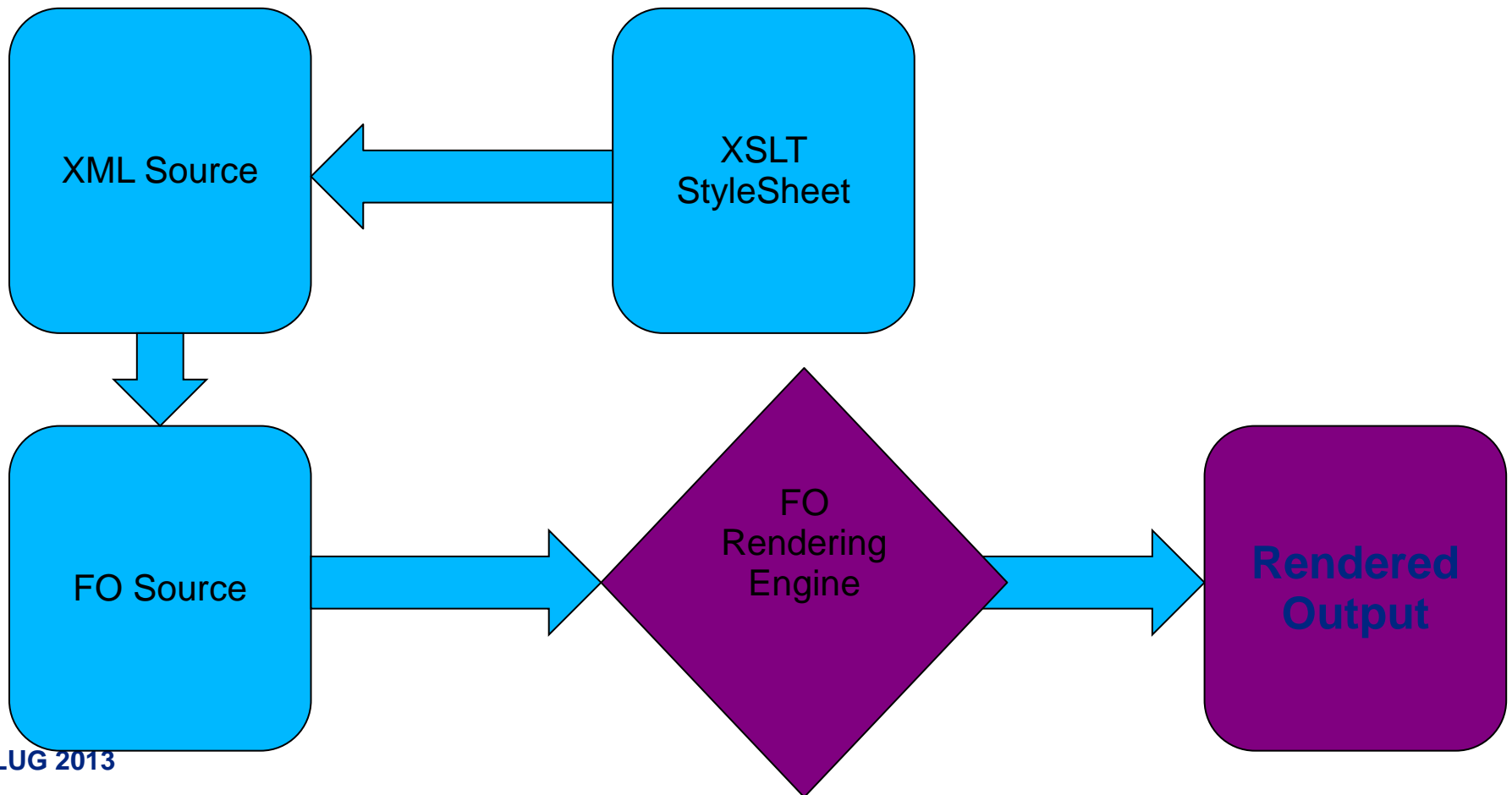
The right side of the bottom panel shows 'Server Options' with the following settings:

- Full Update
- Partial Update
- ID:
- No Update
- No Submission
- Do not validate or update data
- Process data without validation
- Set partial execution mode

Buttons for 'Select Element...', 'Edit Event Parameters...', and 'Edit Event Parameters...' are also visible.

The Complete Process

- That finishes the design pattern



Variations of the Pattern

- Obviously hard coding the URLs in the Java code is NOT a best practice
- Report “Framework” can be created
 - ◆ This allows folks to choose different XML/XSLT sources for report output
- XSLT Code can be...
 - ◆ Stored as Design Elements
 - ◆ Stored in Documents
- If the applications that are the XML/XSLT Sources do not allow anonymous access then you will have to authenticate
- Let’s take a look

Agenda

- **What is APACHE FOP ?**
- **Setup and configure FOP for development and production**
- **The XML Source**
- **The StyleSheet**
- **The Code**
- ***Summary – Q&A***

Resources

- **Apache FOP site**

- ♦ <http://xmlgraphics.apache.org/fop/>

- **W3 Schools**

- ♦ **XSLT**

- ▶ <http://www.w3schools.com/xsl/default.asp>

- ♦ **XSLFO**

- ▶ <http://www.w3schools.com/xslfo/default.asp>

- ♦ **XML**

- ▶ <http://www.w3schools.com/xml/default.asp>

Q and A

- **Thanks !!!**
- **Don't forget to fill out those evals !!!**

- **This is where you ask your questions !!!!!**

- **This is how to get ahold of me**
- **Email: pcalhoun@nnsu.com**
- **Twitter: ptcalhoun**