Practical Setup and Implementation of XML Publisher Templates

Session #30733
Tuesday, March 20, 2012 (10:45 AM – 11:45 AM)
Agenda/Contents

• Overview
• Your presenter
• Deloitte
• What are we talking about?
• The basics – code examples
• More advanced – code examples
• Best practices
• Summary
Overview

• Not about 3C communications
  – Campus Solutions Letter configuration
  – Communication Generation (CommGen)

• Not about setting up XML Publisher
  – Data Sources
  – Report Definitions

• Alliance 2010: Communications Configuration and Communication Generation – Common Questions and Helpful Hints
Overview

• All about XML Publisher RTF templates

• Concepts common to 3C communication templates, custom report templates, transcript templates, etc.

• Coding examples

• Based on my own experience creating complex XMLP Templates
Your Presenter

• Michael Eastwood
  – A senior specialist with Deloitte.
  – Based out of Halifax, NS.
  – Eleven years experience in higher education.
  – Cover campus community, recruiting & admissions, student records, and academic advisement modules of PeopleSoft Campus Solutions.
  – Has acted as a functional lead for upgrades and implementations of both versions 8.9 and 9.0.
  – Fourth time presenting at Alliance.
• Deloitte Touche Tohmatsu is an organization of member firms around the world devoted to client service excellence. Member firms offer services in audit, tax, consulting and financial advisory.

• Consultants from Deloitte member firms help organizations plan, grow and structure their business to address key issues such as strategy, technology and change management.

• Consulting services include:
  • Enterprise Applications
  • Human Capital
  • Technology Integration
  • Strategy and Operations
  • Regulations (Enterprise Risk Services)
  • Security and Privacy (Enterprise Risk Services)

• Deloitte has an established and growing Higher Education Practice in North America

• Deloitte has developed proprietary Total Campus™ Tools and Methodologies specific to Higher Education
What are we talking about?
What are XMLP Templates?

- **XML** - stands for eXtensible Markup Language
  - designed to carry data, not to display data
  - does not really DO anything

- **XSL** - stands for eXtensible Stylesheet Language
  - Style sheets for XML
  - Style sheets describe how documents are presented on screens and in print
What are XMLP Templates?

- XML is the format of the data
- XSL displays the data on the screen
- XMLP templates are RTF documents marked up with BI Publisher’s simplified XSL
What is XSL?

- **eXtensible Stylesheet Language**

- Consists of three parts
  - XSLT – eXtensible Stylesheet Language Transformations
  - XSL-FO – eXtensible Stylesheet Language Formatting Objects
  - XPath – a language for navigating attributes and elements in XML documents
What does this mean in XMLP?

- BI Publisher’s RTF Template Parser converts RTF documents to XSL-FO

- You can add data fields and other markup to templates using BI Publisher's simplified tags for XSL

- BI Publisher also supports pure XSL elements in a template

- In fact, you can include *any* XSL element, many FO elements, and a set of SQL expressions extended by BI Publisher
What does this mean in XMLP?

• If you use BI Publisher’s simplified tags, you are one step removed from real XSL

• Pro: Easier to understand

• Con: Harder to find solutions to more complicated requirements
The Basics

Coding Examples
The Basics

• BI Publisher supports two methods for creating RTF templates:
  1. Basic RTF Method
  2. Form Field Method

• To use XSL or XSL-FO code rather than the simplified syntax, you must use the form field method
The Basics

- The examples that follow are all from RTF templates created in MS Word using the BI Publisher add-in

- All use the simplified syntax and form field method

- The code shown is contained within form fields in the actual templates, unless otherwise noted
Sample XML Data

```xml
<SCCquery_DS_ADMA_PROGRAMS numrows="2" queryname="DS_ADMA_PROGRAMS">
  <row_DS_ADMA_PROGRAMS rownumber="1">
    <APPL_PROG_NBR>0</APPL_PROG_NBR>
    <ACAD_CAREER>UGRD</ACAD_CAREER>
    <ACAD_PROG>BA</ACAD_PROG>
    <PROG_DESCR>Bachelor of Art</PROG_DESCR>
    <ACAD_PLAN>BABIOL</ACAD_PLAN>
    <PLAN_DESCR>Biology (BA)</PLAN_DESCR>
    <PROG_STATUS>CN</PROG_STATUS>
    <PROG_ACTION>DENY</PROG_ACTION>
    <ACTION_DESCR>Denied</ACTION_DESCR>
    <ADMIT_TERM>1118</ADMIT_TERM>
    <TERM_DESCR>Fall 2011</TERM_DESCR>
  </row_DS_ADMA_PROGRAMS>
  <row_DS_ADMA_PROGRAMS rownumber="2">
    <APPL_PROG_NBR>1</APPL_PROG_NBR>
    <ACAD_CAREER>UGRD</ACAD_CAREER>
    <ACAD_PROG>BA</ACAD_PROG>
    <PROG_DESCR>Bachelor of Art</PROG_DESCR>
  </row_DS_ADMA_PROGRAMS>
</SCCquery_DS_ADMA_PROGRAMS>
```
For-Each Statement

- Allows you to repeat the same text or conditional test for each row, a.k.a., a loop (XSLT)

```xml
<?for-each:row_DS_ADMA_PROGRAMS?>
<?sort:APPL_PROG_NBR;'ascending';data-type='text'?>
<?end for-each?>
```
If Statement

- Used to perform a conditional test against the data (XSLT)

```xml
<?if:PROG_STATUS = 'WT'?>
Insert text and markup code here.
<?end if?>
```

- Inlines

```xml
<?if@inlines:PROG_ACTION = 'DENY'?>Insert text and markup code here.<?end if?>
```
Choose Statement

- Allows you to express multiple conditional tests (XSLT)

```xml
<?choose:?>
<?when:PROG_STATUS='AD'?>
Insert text and markup code here.
<?end when?>
<?when:PROG_STATUS='WT'?>
Insert text and markup code here.
<?end when?>
<?otherwise:?>
Insert text and markup code here.
<?end otherwise?>
<?end choose?>
```
More Advanced

Coding Examples
Specifying Attributes and Elements

• Select an attributes (XPath)
  – the @ symbol is used to select the value of an attribute

<?SCCquery_DS_ADMA_PROGRAMS/@numrows?>

• Reference the value of an element in a specific node (XPath)
  – square brackets hold predicates, which reference specific nodes

<?row_DS_ADMA_PROGRAMS[1]/PLAN_DESCR?>
Specifying Attributes and Elements

• Example of this used in an if statement
  – compare PROG_ACTION element of the first row_DS_ADMA_PROGRAMS element with the second

```xml
<?if:row_DS_ADMA_PROGRAMS[1]/PROG_ACTION = row_DS_ADMA_PROGRAMS[2]/PROG_ACTION?>
Insert text and markup code here.
<?end if?>
```
Sub-Templates

- Useful for reducing the amount of repeated text/code within a single template or across many templates

- Include a sub-template

```xml
<?import:psxmlp://ADM_LETTERS_STMPLT?>
```

- Include a sub-template for testing

```xml
<?import:file:///c:/temp/ADM_LETTERS_STMPLT.rtf?>
```
Sub-Templates

• Start a template

<!--This is used as the header for all emails-->
<?template:Email_Header?>
Insert text and markup code here.
<?end template?>

• Call a portion of the sub-template

<?call:Email_Header?><?end call?>
Sub-Template Parameters

• Template that accepts a parameter

<!--This is used to display the program status-->
<?template:Program_Status?><?param:rownum;0?>
Insert text and markup code here.
<?end template?>

• Pass a parameter

<?call@inlines:Program_Status?><?with-
param:rownum;1?><?end call?>
Extended SQL and XSL Functions

- BI Publisher has an extended set of SQL and XSL functions

- The syntax for extended SQL functions is

  `<?xdofx:expression?>`

- The syntax for extended XSL functions is

  `<?xdoxslt:expression?>`
Extended SQL and XSL Functions

• Cannot mix xdofx and xdoxsLt in the same context

• Listed in the Oracle BI Publisher – Report Designer’s Guide starting on page 8-2

• Example, if-then-else statement (extended SQL)

```xml
<?xdofx:if PROG_STATUS = 'WT' then result1
else result2 end if?>
```
Updateable Variables

• Set an updateable variable (extended XSL)

  <?xdoxslt:set_variable($_XDOCTX, 'AdmComm', 'variable value goes here')?>

• Get an updateable variable (extended XSL)

  <?xdoxslt:get_variable($_XDOCTX, 'AdmComm')?>
Best Practices
RTF Template Best Practices

Use Form Fields

• Keep the template clean
• Supported by the Template Builder Field Browser
• Can be colored or hidden to help understand the structure
• Caveat: MS Word header and footer do not allow form fields

(Oracle Business Intelligence Publisher Best Practices)
RTF Template Best Practices

Don’t overcomplicate your template
• Keep it easy to understand, debug and maintain
• In general better to have different business documents in different templates
• Try to limit the logic in templates to simple if or loop statements
• Use sub templates to simplify documents if necessary
• Many calculations are better performed in the data model

(Oracle Business Intelligence Publisher Best Practices)
Interesting Side Note

Data Model Design Guidelines

Element Naming

• Avoid re-using the same name for element names; it can cause confusion in template design
  – For example, **CITY** under **SHIPPING_ADDRESS** and **BILLING_ADDRESS**. But in many cases you are better off using different names, e.g., **S_CITY** and **B_CITY**

*(Oracle Business Intelligence Publisher Best Practices)*
Reference Material

- w3schools.com
  - http://www.w3schools.com

- Oracle BI Publisher - Report Designer's Guide
  - http://docs.oracle.com/cd/E12844_01/doc/nav/portal_booklist.htm

- Oracle Business Intelligence Publisher Best Practices
Summary

- XSL is the style sheet language for XML
- BI Publisher uses a simplified version of XSL
- Basics: for-each, if, choose
- Advanced: attributes, sub-templates, updateable variables
- Use form fields where you can
- Don’t overcomplicate things, unless it’s necessary
Questions?
Contact

• Michael Eastwood
  Senior Specialist, Consulting
  Deloitte
  E-mail: meastwood@deloitte.ca
This presentation and all Alliance 2012 presentations are available for download from the Conference site at www.heug.org www.psuonline.org www.federalusersnetwork.com

Presentations from previous meetings are also available