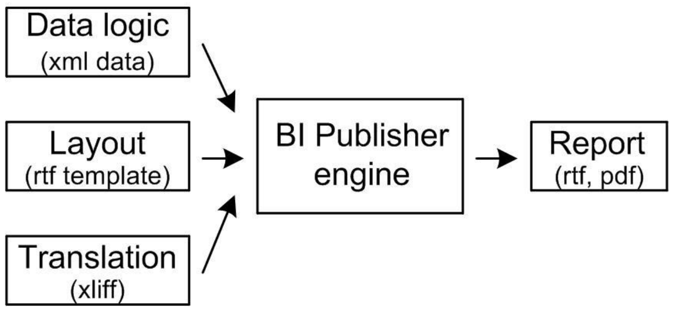
Oracle BI Publisher Templating: Power of Variables

By Alex Andreev, Oracle E-Business Suite Technical Consultant, ITFB Group

Oracle BI Publisher is a reporting solution which is delivered with Oracle Applications like Oracle Business Intelligence Enterprise Edition, Oracle E-Business Suite, Oracle Siebel. This article intends to get you started with template creating by using variables. It also explains the usage of some common scenarios.

BIP report creating

Oracle BI Publisher (BIP) separates the creation of data (Data logic) from the process of formatting it for different uses (Layout + Translation). The engine can format any well-formed XML data, allowing integration with any system that can generate XML, including Web Services or any data source available through. BIP can merge multiple data sources into a single output document.



Template design

BI Publisher report templates can be designed using the Microsoft Word, Microsoft Excel, Adobe Acrobat etc. It seems you have MS Office, so you can use Oracle BIP Desktop tool to create templates. In this article we are limited by RTF template with main focus on variables.

Why are variables the aim of this article?

The creating of report requires the comprehension of the source xml data and what we can have in the end (report). The optimal distribution of logic is an important aspect of development and maintain in the future. For example, you need to generate a report of differences from several sources. That is, the data model based on the source A and the source B. You need to get report data which is available only in source A and/or only in source B and/or in both of them at the same time. How can this logic be implemented? And what level: data logic level or template level or using the capabilities of the database?

Further in the article is given the information how it’s easy to realize by variables at the template level.

When and how can variables be used?

Some descriptions of variables are the following.

1. XSL variables or not updatable variables. You cannot change or modify values.

Cases to use:

1.1. If you use the same value multiple times it would be better to assign that value to a variable and use references to the variable instead.

It can be described like a formula: "`x = f(y)`", where "`f(y)`" is some simple operation based on a number or a string with the result which is also as simple type as a string and a number.

Syntax:

- to declare/set a variable value:

`<xsl:variable name="variable_name">` (XSL Syntax);

`<?variable:variable_name?>` (BI Publisher Tag);

as an example: `<xsl:variable name="x" select="10"/>` or `<?variable:x;10?>` is assignment of the value "`10`" to the "`x`" variable;

- to get a variable value:

the variable can then be referenced in the template as "`$x`".

Note: The only way in which a variable can be changed is by declaring it inside for-each loop, in which case its value will be updated for every iteration.

```
<xsl:for-each select="/foo/bar">

  <xsl:variable name="some-bar" select="."/>

  <xsl:value-of select="$some-bar"/>

</xsl:for-each>.
```

It's worth noting that xsl variable can store not only the number or the string but there is content inside of it too. The first case is commonly used in BIP templating. Unfortunately the second case is less known. That's why description of xsl variable is divided in two parts.

Cases to use:

1.2. If you want to pre-process some values based on some logic for next using.

It can be described like a formula: "`x = f(y)`", where "`y`" is another processed variable or some data subset, for instance, subset of source xml data and so on.

Syntax for set/get is like previous case.

As example:

```
<xsl:variable name="x">

  <xsl:choose>

    <xsl:when test="$y &gt; 5">

      <xsl:text>10</xsl:text>
    </xsl:when>

    <xsl:otherwise>

      <xsl:text>0</xsl:text>

    </xsl:otherwise>

  </xsl:choose>

</xsl:variable>
```

is assignment of the text value `"10"` or `"0"` to the `"x"` variable based on verification of another variable `"y"`: `"y > 5"`;

```
<xsl:variable name="x">

  <xsl:for-each select="ROWSET/ROW">

    <xsl:copy-of select="." />

  </xsl:for-each>

</xsl:variable>
```

is assignment of data subset which is identified by XPath `"ROWSET/ROW"` to the `"x"` variable.

2. Updatable variables. They are updatable during the template application. The variables use a set/get approach for assigning, updating, and retrieving values.

Cases to use:

for re-assignment of already existed variable, for example, by some conditions, or for storing some mediator values.

It can be described like a formula: `"x = f(x)"`, where `"f(x)"` is some post-processing operations based on already existed value. In general it’s about some calculation like cumulative sum, re-assignment of variable’s value.

Syntax:

- to declare/set a variable value:

`<?xdoxslt:set_variable($_XDOCTX, 'variable name', value)?>`;

as example:

`<?xdoxslt:set_variable($_XDOCTX, 'x', xdoxslt:get_variable($_XDOCTX, 'x') + 1)?>` is re-assignment of the value of `"x"` variable based on previous value of `"x"` plus 1. It’s like a formula: `"x = x + 1"`;

`<?xdoxslt:set_variable($_XDOCTX, 'x', xdoxslt:get_variable($_XDOCTX, 'x') * xdoxslt:get_variable($_XDOCTX, 'x'))?>` is re-assignment of the value of `"x"` variable based on previous value of `"x"` squared. It’s like a formula: `"x = x * x"`;

- to retrieve a variable value:

`<?xdoxslt:get_variable($_XDOCTX, 'variable name')?>`.

Using variables by samples

Sample 1

Problem: It’s necessary to get cumulative total.

Sample XML data:

```
<ROWSET>
  <ROW>
    <VAL>1</VAL>
    <COL>10</COL>
  </ROW>
  <ROW>
    <VAL>2</VAL>
    <COL>10</COL>
  </ROW>
  <ROW>
    <VAL>3</VAL>
    <COL>10</COL>
  </ROW>
</ROWSET>
```

RTF Template:

Decl

Val	**Col**	**Res**
F<?VAL?>	<?COL?>SetVar	GetVar E

Where fields (gray) are:

Decl	<?xdoxslt: set_variable($_XDOCTX, 'LVar',number(0))?>
F	<?for-each:ROW?>
SetVar	<?xdoxslt:set_variable($_XDOCTX, 'LVar', xdoxslt:get_variable($_XDOCTX, 'LVar') + COL)?>
GetVar	<?xdoxslt: get_variable($_XDOCTX, 'LVar')?>
E	<?end for-each?>

Result:

Val	**Col**	**Res**
1	10	10
2	10	20
3	10	30

Sample 2

Problem: The result data set is formed from two sources A and B. According to this, it’s necessary to generate a report of the differences of elements.

Sample XML data:

```
<ROWSETS>
  <ROWSET_A>
    <ROW>
      <FIELD>User A</FIELD>
    </ROW>
    <ROW>
      <FIELD>User R</FIELD>
    </ROW>
    <ROW>
      <FIELD>User S</FIELD>
    </ROW>
  </ROWSET_A>
  <ROWSET_B>
    <ROW>
      <FIELD>User A</FIELD>
    </ROW>
    <ROW>
      <FIELD>User R</FIELD>
    </ROW>
    <ROW>
      <FIELD>User Z</FIELD>
    </ROW>
  </ROWSET_B>
</ROWSETS>
```

RTF Template:

Decl_Set_A Decl_Set_B
Only in ROWSET_A

#	**Field value**
F1<?position()?>	<?FIELD?>E1

Only in ROWSET_B

#	**Field value**
F2<?position()?>	<?FIELD?>E2

In both ROWSETs

#	**Field value**
F3<?position()?>	<?FIELD?>E3

Where fields (gray) are:

Decl_Set_A	<xsl:variable name="Set_A" select="/ROWSETS/ROWSET_A/ROW/FIELD" />
Decl_Set_B	<xsl:variable name="Set_B" select="/ROWSETS/ROWSET_B/ROW/FIELD" />
F1	<?for-each: ROWSET_A/ROW[not(FIELD=$Set_B)]?>
F2	<?for-each: ROWSET_B/ROW[not(FIELD=$Set_A)]?>
F3	<?for-each: ROWSET_A/ROW[FIELD=$Set_B]?>

E1, E2, E3	<?end for-each?>

Result:

Only in ROWSET_A

#	**Field value**
1	User S

Only in ROWSET_B

#	**Field value**
1	User Z

In both ROWSETs

#	**Field value**
1	User A
2	User R

Sample 3

Problem: It's necessary to remove duplicate values. If xml data is ordered, then you can use construction like "`ROW[position()=1 or ./preceding-sibling::ROW[1]/FIELD_FOR_CHECK!= FIELD_FOR_CHECK]`".

But if xml data is mixed, this construction will not work. If you apply the sort operation after for-each, the sort will take place on resulted set after above checking up. How to get presorted list before?

Sample XML data:

```
<ROWSET>
  <ROW>
    <POS>42</POS>
    <FIELD>SCOTT</FIELD>
  </ROW>
  <ROW>
    <POS>44</POS>
    <FIELD>Adam</FIELD>
  </ROW>
  <ROW>
    <POS>46</POS>
    <FIELD>SCOTT</FIELD>
  </ROW>
</ROWSET>
```

RTF Template:

Decl

Pos	**Field**
F <?POS?>	<?FIELD?> E

Where fields (gray) are:

Decl	<xsl:variable name="sortedSet"><xsl:for-each select="ROWSET/ROW"><xsl:sort select="FIELD"

	/><xsl:copy-of select="." /> </xsl:for-each> </xsl:variable>
F	<?for-each:$sortedSet/ROW[position()=1 or ./preceding-sibling::ROW[1]/FIELD!=FIELD]?>
E	<?end for-each?>

Result:

Pos	Field
44	Adam
42	SCOTT

Sample 4

Problem: as in sample 3 but removing duplicate is by variable itself.

RTF Template:

```
Decl
```

Pos	Field
F <?POS?>	<?FIELD?> E

Where fields (gray) are:

Decl	<xsl:variable name="sortedSet" select="//ROW[not(./FIELD=following::ROW/FIELD)]" />
F	<?for-each:$sortedSet?>
E	<?end for-each?>

Result:

Pos	Field
44	Adam
46	SCOTT

Summary

Thus, considered a number of examples will help the readers to take a broader look at the possibilities of using variables and to improve knowledge in templating. It can be also used as a good addition to the main documentation.

About the Author

Alex Andreev, Oracle E-Business Suite Technical Consultant, ITFB Group

Alex, as a part of ITFB Group, helps companies get the best value from their investments in a wide range of technology stacks. In ITFB Group, he focuses on Oracle E-Business Suite, Business Intelligence and SOA technologies.