An Introduction to Web Content Publisher Generation Templates

By Gregory Melahn (melahn@us.ibm.com)
Software Engineer, IBM Corp.
July 2002

Abstract
Web Content Publisher (WCP) uses templates to generate content from resources. Templates can be written using JSP™ or XSL syntax. This document provides an overview of the structure of these templates.

Types of Generation Templates
There are two types of generation templates, detail and summary. Detail templates work with individual resources. Summary templates work with lists of resources. Both detail and summary templates can be written in JSP or XSL syntax, and produce content that is captured by WCP. The generated content can then be reviewed, previewed, and published to a web site.

Inside a JSP Template
The raw materials your generation JSP can work with include parameters and some helper classes.

Parameters
There are several parameters that are passed to the JSP, such as *beanName* and *id*. The *beanName* parameter contains the fully qualified classname of the resource the template is expected to work with. For example, a template handling instances of the WCP sample resource type Product would be passed the *beanName* of WCMSample.Product. The *id* parameter contains the resource id of the resource that the template is expected to work with.

Helper Classes
There are two helper classes provided by WCP that allow your JSP to access resources. One helper class is named *com.ibm.wcm.GetGenericResource*. The other is named *com.ibm.wcm.GetGenericResourceList.*

com.ibm.wcm.GetGenericResource is used to get an instance of a resource.
Using the getResource method of an instance of this class instantiates an attribute named resObj of the *beanName* type. The following two lines of code illustrate this access. The first line actually retrieves the resource. The second line provides a reference to resObj.

```
<jsp:useBean class="com.ibm.wcm.GetGenericResource" id="getResource"
type="com.ibm.wcm.GetGenericResource"><%
getResource.getResource(request); %></jsp:useBean>
<jsp:useBean id="resObj" type="WCMSample.Product" scope="request"/>
```

Once you have a reference to this resObj in the JSP, you can access the getter methods of the resource to discover the attributes. For example, if you have a valid resObj reference, you can include the following lines in your JSP to get the value of the NAME attribute of the resource.

```
  <P>The name is <%= resObj.getNAME() %></P>
```

com.ibm.wcm.GetGenericResourceList is used to get lists of resources. This helper class is most useful in summary generation templates though it could be used elsewhere. (There is a separate white paper on this subject titled Getting resources in authoring templates in Web Content Publisher.) Including the following code fragment provides you with an instance of this class properly initialized with the HTTP request. The HTTP contains information that allows WCP to know what project, activity, and locale your template is running in.

```
<jsp:useBean class="com.ibm.wcm.GetGenericResourceList"
id="getResourceList" type="com.ibm.wcm.GetGenericResourceList"><%
getResourceList.setRequest(request); %></jsp:useBean>
```

Having an instance of *com.ibm.wcm.GetGenericResourceList* allows you to use a useful method called getResourceList(). This method returns an enumeration of resources, as illustrated in the code fragment below:

```
<% java.util.Enumeration enumObjs = getResourceList.getResourceList();
%>
```

Having an enumeration, now you can iterate over that enumeration to process the resources in the folder the summary template is attached to. Once you have the resource, you can use the getter methods to access the attributes of the resource to do things like generate HTML. This is illustrated in the code fragment below:

```
       <%
        while( enumObjs.hasMoreElements() )
        {
         WCMSample.Product product =
(WCMSample.Product)enumObjs.nextElement();
             // do something here with the product
        }
       %>
```

Note that casting the resource returned from the enumeration is necessary.

Inside an XSL Template

Alternatives to JSP author forms are XSL generation forms. Unlike JSPs, which are passed attributes, XSL author templates are passed an XML stream. The XML stream passed to the style sheet adheres to WCP's export format. This simple serialization of a resource is best described through the example below:

```
<?xml version="1.0" encoding="UTF-8"?>
<WCMSample.Product>
 <description>Toys</description>
 <displayName>Toys</displayName>
 <properties resourceId="FT0100">
  <property name="QUANTITY_SOLD"
type="java.lang.Integer">34562</property>
  <property name="STAGE" type="java.lang.String">Available</property>
  <property name="DESCRIPTION" type="java.lang.String">Large play
station with many compartments for future trips to Mars. Installs on
the ground. Base adapts to unpredictable surface conditions. Ages 4-12.
Includes laser tag set!</property>
  <property name="RETAILPRICE"
type="java.math.BigDecimal">0.00</property>
  <property name="WHOLESALEPRICE"
type="java.math.BigDecimal">0.00</property>
  <property name="PRODUCTNUMBER"
type="java.lang.String">FT0100</property>
  <property name="IMAGEURL"
type="java.lang.String">/WCMSample/products/marsBase.jpg</property>
  <property name="SITE" type="java.lang.String">Raleigh</property>
  <property name="NAME" type="java.lang.String">Mars Play
Station</property>
  <property name="QUANTITY_OVERSTOCK"
type="java.lang.Integer">0</property>
 </properties>
</WCMSample.Product>
```

You can easily see the XML format for a given resource type by exporting a resource and looking at the resulting file.

For summary templates, the XML format is the same, except the properties section is repeated for each resource in the folder. For example, a list of two resources would look like the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<WCMSample.Product>
 <description>Toys</description>
 <displayName>Toys</displayName>
 <properties resourceId="FT0100">
  <property name="QUANTITY_SOLD"
type="java.lang.Integer">34562</property>
  <property name="STAGE" type="java.lang.String">Available</property>
  <property name="DESCRIPTION" type="java.lang.String">Large play
station with many compartments for future trips to Mars. Installs on
the ground. Base adapts to unpredictable surface conditions. Ages 4-12.
Includes laser tag set!</property>
```

```
  <property name="RETAILPRICE"
type="java.math.BigDecimal">0.00</property>
  <property name="WHOLESALEPRICE"
type="java.math.BigDecimal">0.00</property>
  <property name="PRODUCTNUMBER"
type="java.lang.String">FT0100</property>
  <property name="IMAGEURL"
type="java.lang.String">/WCMSample/products/marsBase.jpg</property>
  <property name="SITE" type="java.lang.String">Raleigh</property>
  <property name="NAME" type="java.lang.String">Mars Play
Station</property>
  <property name="QUANTITY_OVERSTOCK"
type="java.lang.Integer">0</property>
 </properties>
 <properties resourceId="FT0200">
  <property name="QUANTITY_SOLD"
type="java.lang.Integer">142</property>
  <property name="STAGE" type="java.lang.String">Future</property>
  <property name="DESCRIPTION" type="java.lang.String">Simple implement
such as a broom or umbrella. Used to jam and mentally become a rock
star. Stimulates the imagination. Encourages visualization.</property>
  <property name="RETAILPRICE"
type="java.math.BigDecimal">0.00</property>
  <property name="WHOLESALEPRICE"
type="java.math.BigDecimal">0.00</property>
  <property name="PRODUCTNUMBER"
type="java.lang.String">FT0200</property>
  <property name="IMAGEURL"
type="java.lang.String">/WCMSample/products/airGuitar.jpg</property>
  <property name="SITE" type="java.lang.String">Seattle</property>
  <property name="NAME" type="java.lang.String">Air Guitar</property>
  <property name="QUANTITY_OVERSTOCK"
type="java.lang.Integer">1</property>
 </properties>
</WCMSample.Product>
```

The mission of the XSL generation template is to parse this stream and produce a piece of content that WCP then stores.

The first thing the XSL template must do is include some housekeeping tags that tell the XSL processor how to process the remainder of the style sheet. This is done using the following lines:

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
```

The next thing to do is match the node that indicates what type of resource is being processed. The value of this node is the fully qualified *className* of the resource, as in the following example:

```
 <xsl:template match="WCMSample.Product">
```

Having matched this node, you can start to build the resulting output stream that will include the attribute values, as in the following example:

```
  <html>
   <title><xsl:value-of select="displayName" /></title>
   <body>
    <center>
    <p><xsl:value-of select="description" /></p>
```

The next step usually entails including a section that matches the properties node, enumerates the property values, and does things to construct a table entry. In this section, put an iterator for each property and output some content, as in the following example:

```
   <td valign="top">
    <xsl:for-each select="property">
     <xsl:if test="@name='PRODUCTNUMBER'">
      <xsl:value-of select="." />
     </xsl:if>
    </xsl:for-each>
   </td>
```

Finally, don't forget to close the style sheet with a balancing tag.

```
</xsl:stylesheet>
```

Trademarks

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

ActiveX, Microsoft, Windows, Windows NT®, and the Windows logo are trademarks or registered trademarks of Microsoft Corporation in the United States, or other countries, or both.

UNIX is a registered trademark of The Open Group.

Linux is a registered trademark of Linus Torvalds.

Other company, product, and service names, which may be denoted by a double asterisk(**), may be trademarks or service marks of others.

Notices

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION AND ANY ASSOCIATED CODE "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR CONDITIONS OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

© Copyright International Business Machines Corporation 2002. All rights reserved. US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.