# An Introduction to Web Content Publisher Authoring Templates

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

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

## Inside a JSP Template
The raw materials the JSP can work with include parameters, helper classes, and commands.

## 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 the template is expected to work with. Note the *id* parameter is not present in the Add template.

## Helper Class
A helper class provided by WCP allows your JSP to access resources. This helper class is named *com.ibm.wcm.GetGenericResource*. 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 provide 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 and setter methods of the resource. For example, you can include the following lines in your JSP to get the value of the NAME attribute of the resource and include the value in an input field.

```
 <INPUT style="width:200px;" TYPE="text" NAME="NAME" VALUE="<%=
resObj.getNAME() %>" SIZE="25" MAXLENGTH="50"/>
```

## Commands

WCP is structured in such a way that all commands such as add a resource and update a resource are processes by a single command servlet. This command servlet is initialized as part of WCP's startup. The name of the command servlet is */wps/wcp/command*. The command servlet accepts the commands listed in the commands.properties file. The commands of interest to customer-defined templates include *addItem* and *updateItem*. By setting the value of the *command* attribute to one of these values, you tell WCP which command to execute when you POST your form.

For example, the following lines tell WCP to update a resource:

```
<FORM ACTION="/wps/wcp/command" METHOD="POST">
<INPUT TYPE="hidden" NAME="command" VALUE="updateItem"/>
```

In addition to this attribute, you also need to pass the beanName and any attributes collected by the form when you POST as illustrated in the following fragment:

```
<INPUT TYPE="hidden" NAME="beanName" VALUE="WCMSample.Product"/>
INPUT style="width:200px;" TYPE="text" NAME="PRODUCTNUMBER" VALUE=""
SIZE="25"
MAXLENGTH="6"/>
```

## Inside an XSL Template

An alternative to JSP author forms is XSL author 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.

The mission of the XSL author template is to parse this stream and produce a web page that allows the author to edit, show, or preview a resource.

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 attribute values, as in the following example:

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

If this is an update form, at some point in the output stream, you must build a form that will be POSTed to the command servlet in order to accomplish the *updateItem* action (note that *addItem* XSL templates are not allowed). This is the same command servlet described in the JSP section. Here is an example:

```
    <form action="/wps/wcp/command" " method="post">
     <input type="hidden" name="command" value="updateItem"/>
     <input type="hidden" name="beanName" value="WCMSample.Product"/>
    <xsl:apply-templates select="properties" />
    </form>
```

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 as in the following example:

```
   <xsl:for-each select="property">
```

```
    <xsl:if test="@name='QUANTITY_OVERSTOCK'">
     <tr>
      <td valign="top" align="right">Overstock:</td>
      <td valign="top"><xsl:element name="input">
         <xsl:attribute name="type">text</xsl:attribute>
         <xsl:attribute name="name">QUANTITY_OVERSTOCK</xsl:attribute>
         <xsl:attribute name="size">10</xsl:attribute>
         <xsl:attribute name="maxlength">10</xsl:attribute>
         <xsl:attribute name="value"><xsl:value-of select="."
/></xsl:attribute>
        </xsl:element>
      </td>
     </tr>
    </xsl:if>
   </xsl:for-each>
```

If this is an updateItem form, then there must be a way to trigger the action. This can be done with a button built like this:

```
      <input type="submit" name="formAction" value="Save"/>
```

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.