Resource Transformations using Web Content Publisher Generation Templates

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

Abstract

Web Content Publisher (WCP) provides the ability to generate static content from structured resources. For example, you can have a structured resource type of Press Release and generate static html pages from instances of that type. Such generation is done using generation templates. Generation templates can also be used to generate instances of other resource types, such as *Toys* or *News*. This document describes how to create generation templates.

Designing a Resource Generation Template

The first step in designing a resource generation template is to choose the source and target resource types for the transformation. For example, you could decide to use one of WCP's built-in types, *Channel items*, to generate one of WCP's sample types, *News*.

After choosing the source and target resource types, the next step is to decide how the attributes of the source type map to the target type. Continuing the *Channel items* and *News* example, you could decide on the following mappings:

Source Resource Type	Source Attribute	Target Resource Type	Target Attribute
Rsschannelitem	Title	News	Title
		News	Confidential
		News	ForSite
		News	ForLocation
		News	ForDepartment
		News	CategoryName
Rsschannelitem	Link	News	Content
Rsschannelitem	ChannelTitle		
Rsschannelitem	Content		
Rsschannelitem	Category		

Writing a Resource Generation Template

Resource generation templates are written using XSL. The XSL document you write parses an XML document that contains the attributes for the source type and generates an XML document of the target type. WCP creates an instance of the source XML document from the resource instance and passes it to the XSL processor along with your stylesheet. After the target XML document is created, WCP uses it to instantiate a resource of the target type. Both the source and target XML documents are transient and are not stored by WCP once the generation is complete.

In order to write an XSL style sheet, you need to know something about the source and target XML documents. The easiest way to learn about these documents is to export a WCP project that contains instances of each. The resulting documents have the extension .wcp. Exporting a project containing instances of *Channel items* and *News* results in files such as those shown below:

The source type

```
<?xml version="1.0" encoding="UTF-8"?>
<com.ibm.wcm.resources.Rsschannelitem>
 <description>Channel Item</description>
 <displayName>Channel Item</displayName>
 <properties resourceId="IBM: We won`t seek patent plan royalties">
 <property name="CHANNELTITLE" type="java.lang.String">Moreover - IBM
news</property>
 <property name="LINK"
type="java.lang.String">http://c.moreover.com/click/here.pl?r36359337</
property>
 <property name="CONTENT" type="java.lang.String">No content stored for
this channel</property>
 <property name="CATEGORY" type="[Ljava.lang.String;" array="true">
 </property>
 <property name="DESCRIPTION" type="java.lang.String">Interactive Week
Apr 19 2002 7:00AM ET</property>
 <property name="TITLE" type="java.lang.String">IBM: We won`t seek
patent plan royalties</property>
 </properties>
</com.ibm.wcm.resources.Rsschannelitem>
```

The target type

```
<?xml version="1.0" encoding="UTF-8"?>
<WCMSample.CompanyNews>
 <description>News</description>
 <displayName>News</displayName>
 <properties resourceId="IBM: We won`t seek patent plan royalties">
 <property name="CONTENT"
type="java.lang.String">http://c.moreover.com/click/here.pl?r36359337</
property>
 <property name="CONFIDENTIAL" type="java.lang.String"></property>
 <property name="FORSITE" type="java.lang.String"></property>
 <property name="CATEGORYNAME" type="[Ljava.lang.String;" array="true">
 </property>
 <property name="FORLOCATION" type="java.lang.String"></property>
 <property name="TITLE" type="java.lang.String">IBM: We won`t seek
patent plan royalties</property>
 <property name="FORDEPT" type="java.lang.String"></property>
 </properties>
</WCMSample.CompanyNews>
```

This provides all the information you need to write an XSL stylesheet that implements the attribute mappings.

The best way to explain how to write a resource generation template is to view a complete example and explain what each XSL section does.
Here is a complete example of a style sheet that takes an instance of the resource type *Channel item* to generate an instance of the type *News*

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

   <xsl:template match="com.ibm.wcm.resources.Rsschannelitem">
    <WCMSample.CompanyNews>
      <description>Company News</description>
      <displayName>News</displayName>
        <xsl:apply-templates select="properties" />
    </WCMSample.CompanyNews>
  </xsl:template>

 <xsl:template match="properties">
 <xsl:element name="properties">
  <xsl:attribute name="resourceId"><xsl:value-of select="@resourceId"
/></xsl:attribute>
  <xsl:apply-templates select="property" />
 </xsl:element>
 </xsl:template>

 <xsl:template match="property">
 <xsl:if test="@name='TITLE'">
  <xsl:element name="property">
 <xsl:attribute name="name">TITLE</xsl:attribute>
 <xsl:attribute name="type">java.lang.String</xsl:attribute>
 <xsl:value-of select="." />
  </xsl:element>
 </xsl:if>
 </xsl:template>

 <xsl:template match="property">
 <xsl:if test="@name='LINK'">
  <xsl:element name="property">
 <xsl:attribute name="name">CONTENT</xsl:attribute>
 <xsl:attribute name="type">java.lang.String</xsl:attribute>
 <xsl:value-of select="." />
  </xsl:element>
 </xsl:if>
 </xsl:template>

</xsl:stylesheet>
```

Now let's look at each part of the stylesheet.

A. This section provides the XSL processor the information it needs to parse the file and is the same regardless of what generation template you write.

```
<?xml version="1.0"?>
```

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

B. This section tells the XSL processor to generate the header node of the target XML document whenever it finds an element in the incoming XML document with a match on the node <sourceresourcetype>. In this case, whenever the XSL processor finds a string matching *com.ibm.wcm.resources.Rsschannelitem*, the template writes out the header node of the XML document for the *WCMSample.CompanyNews* instance. The template also directs the processor to apply the *properties* part of the stylesheet.

```
  <xsl:template match="com.ibm.wcm.resources.Rsschannelitem">
    <WCMSample.CompanyNews>
      <description>Company News</description>
      <displayName>News</displayName>
        <xsl:apply-templates select="properties" />
    </WCMSample.CompanyNews>
  </xsl:template>
```

C. This section tells the processor to look for the node <properties>, to write a node for the *resourceId,* then apply the *property* part of the stylesheet. Note that the resource id of the source resource is the same as the resource id of the target.

```
 <xsl:template match="properties">
 <xsl:element name="properties">
  <xsl:attribute name="resourceId"><xsl:value-of select="@resourceId"
/></xsl:attribute>
  <xsl:apply-templates select="property" />
 </xsl:element>
 </xsl:template>
```

D. This section is repeated for each attribute and implements the attribute mapping by generating a property node in the target XML document for a source node. You would have one of these sections for every mapping rule. In this particular fragment, the TITLE attribute of *Channel item* is being mapped to the TITLE attribute of *News*.

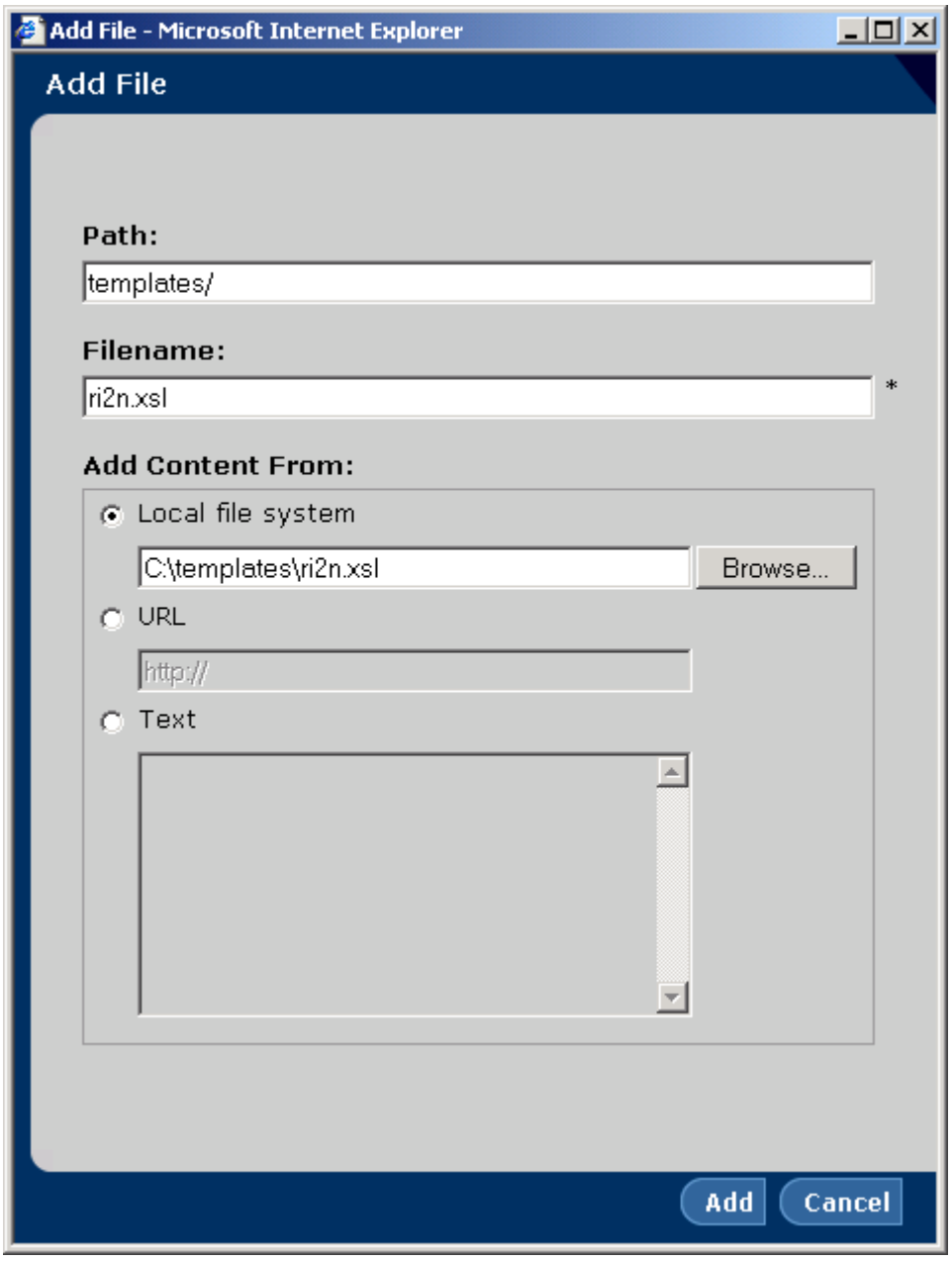
```
 <xsl:template match="property">
 <xsl:if test="@name='TITLE'">
  <xsl:element name="property">
 <xsl:attribute name="name">TITLE</xsl:attribute>
 <xsl:attribute name="type">java.lang.String</xsl:attribute>
 <xsl:value-of select="." />
  </xsl:element>
 </xsl:if>
</xsl:template>
```

E. This section closes the stylesheet.

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

Adding the Template to the WCP Project

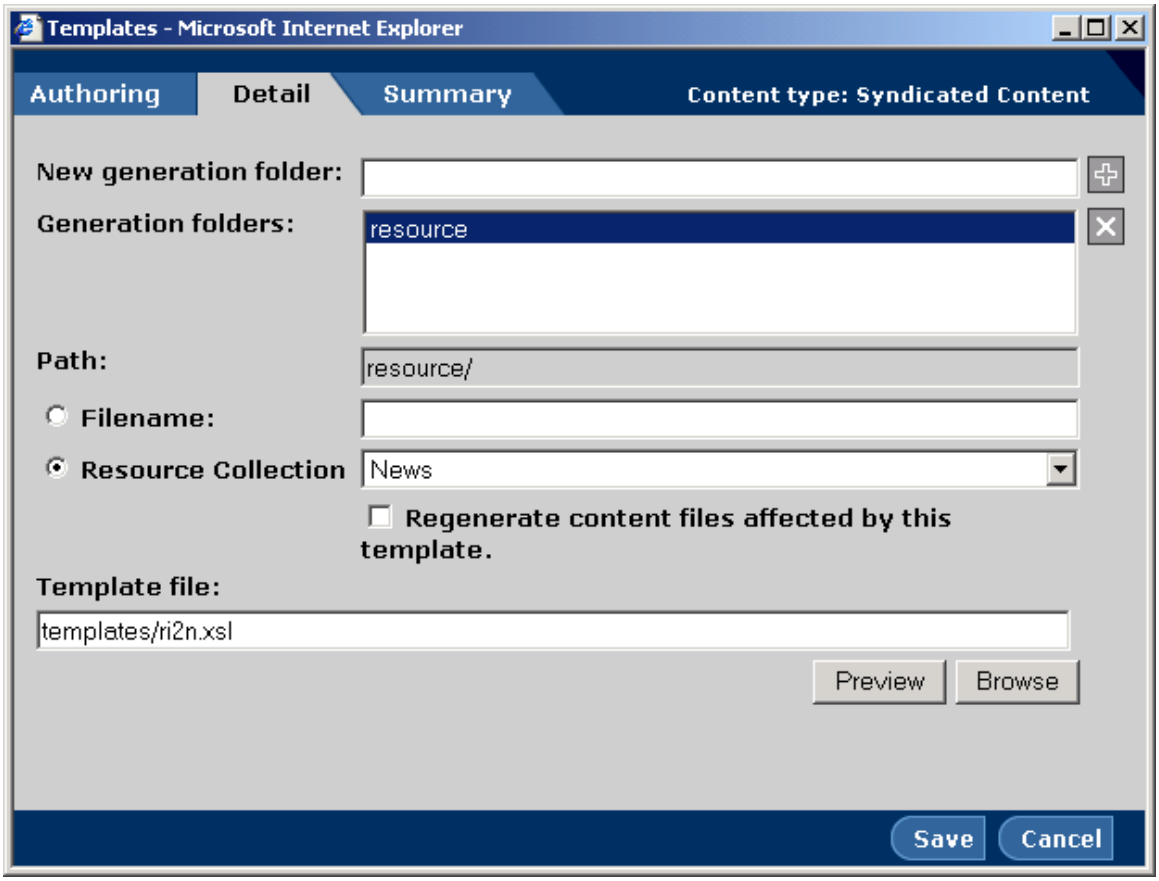
Having completed writing the stylesheet, you then add it to the WCP project by clicking **Add File**  .



Activating a Resource Generation Template

Once the template is in the WCP project, it can be associated with a folder of the source resource collection. Select the source folder (in this case Syndicated content folder), and

click **Templates.** Then, select **Resource Collection** and choose the name of the target resource collection from the drop down list. Specify the name of local file containing the template you wrote. Specify a generation folder of *resource*. Note the folder does not need to be named resource, but naming the folder resource makes it clear this is a resource generation instead of a static file generation folder. Click **Save**.



Generating Resources

Having made this association, every time you edit a resource in the source folder, a resource will be generated in a folder of the same name in the target resource collection. You can also force the generation of all resources by selecting the **Regenerate content files affected by this template** checkbox in the templates dialog.

Note that resource transformations can be chained together. In the example discussed in this paper, if you had a generation template associated with *News,* this template would be used to generate static html (or even another resource) whenever a News instance was created. Therefore, changing one resource could trigger the creation of a number of other resources or static pages. Circular references are not allowed.

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.