BEA WebLogic Workshop™ Help

Version 8.1 SP4
December 2004

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Table of Contents

XML Map Tag Reference

Through XML maps, you can customize the way values in XML messages received and sent by your service's methods are mapped to the variables of your method's implementation. You use the following tags and attributes inside XML maps.

Topics Included in This Section

<xm:attribute> Tag

Specifies the variable to use when mapping an XML attribute value.

<xm:bind> Attribute

Declares a variable within a map and initializes to a variable of the declaration.

<xm:java−import> Tag

Specifies Java classes that should be imported to support variables in an XML map file.

<xm:map−file> Tag

Specifies that its contents constitute an XML map file.

<xm:multiple> Attribute

Specifies the data structure variable to which a specific repeating XML element should be mapped.

<xm:use> Tag

Specifies a function or XML map to invoke for processing XML.

<xm:value> Tag

Specifies the variable to use when mapping a specific XML element content.

<xm:xml−map> Tag

Specifies that its contents constitute an XML map; also provides the signature defining calls to this map.

Related Topics

Why Use XML Maps?

Deprecated. *XML Maps are deprecated as of the WebLogic Platform 8.1 release. For new code, use XQuery maps. For more information, see Introduction to XQuery Maps.*

<xm:attribute> Tag

Specifies the variable to use when mapping an XML attribute value.

Syntax

```
<xm:attribute
    name="attributeName"
    obj="[dataType ]attributeValueVariable[, …]"
>
```

Attributes

name

The name of the attribute that is being mapped.

obj

The portion of the Java declaration to which this attribute's value should be mapped, such as a parameter or "return".

Remarks

Use the <xm:attribute> tag to assign an attribute value from an XML message to a portion of your Java declaration. In the following example, max_records is an anticipated attribute of the <query> element:

```
/**
 * @common:operation


 * @jws:parameter-xml xml-map::
 * <query>
 *     <xm:attribute name="max_records" obj="resultSize"/>
 *     <element>
 *         <name>{criterionName}</name>
 *         <value>{criterionValue}</value>
 *     </element>
 * </query>
 * ::
 */
public void searchData(String criterionName, String criterionValue, int resultSize)
{
    System.out.println("The maximum number of records to return is " + resultSize);
}
```

As an alternative to using the <xm:attribute> tag, you can use curly braces shorthand. The following snippet is equivalent to the preceding example:

```
/**
 * @jws:parameter-xml xml-map::
 * <query max_records="{resultSize}">
 *     <element>
```

```
 *         <name>{criterionName}</name>
 *         <value>{criterionValue}</value>
 *     </element>
 * </query>
 * ::
 */
public void searchData(String criterionName, String criterionValue, int resultSize)
{
    System.out.println("The maximum number of records to return is " + resultSize);
}
```

Specify a data type when the variable is not declared with a type elsewhere in the map's context (such as elsewhere in the map, in Java code at the class level, or in the code for the method to which the map applies).

```
/**
 * @jws:parameter-xml xml-map::
 * <query max_records="{int resultSize}">
 *     <element>
 *         <name>{java.sql.String criterionName}</name>
 *         <value>{java.sql.String criterionValue}</value>
 *     </element>
 * </query>
 * ::
 */
public void searchData(String criterionName, String criterionValue, int resultSize)
{
    System.out.println("The maximum number of records to return is " + resultSize);
}
```

Note: The xm prefix and its URI are declared implicitly in any JWS file. However, you must declare the namespace prefix and URI to use the prefix in XMLMAP files.

Related Topics

How Do XML Maps Work?

Why Use XML Maps?

Deprecated. *XML Maps are deprecated as of the WebLogic Platform 8.1 release. For new code, use XQuery maps. For more information, see Introduction to XQuery Maps.*

<xm:bind> Attribute

Declares a variable within a map and initializes to a variable of the declaration.

Syntax

```
<elementName xm:bind="elementName variableNameVariable is dataStructure.arrayMember[,...]"
>
```

Attributes

bind

A statement declaring a variable and initializing it to a parameter variable.

Remarks

The <xm:bind> attribute is especially useful when you need to bind the XML value to a member of a data structure, while also assigning it a short variable name. The <xm:bind> attribute also specifies that the object corresponding to the variable you're binding should not be instantiated unless the tag is encountered in the instance document.

In the following example, <xm:bind> makes it possible to declare a new variable a of type Address (which must be a type available in the scope of the code, such as an internal class), then initialize the new variable to the address member of the customerData structure. Using <xm:bind> as an attribute of the <address> tag means that the <address> element's value will be mapped to the new variable. In addition, the new a variable can be used in the <street> and <zip> elements.

```
/**
 * @jws:parameter-xml xml-map::
 *  <customer>
 *    <name>{String customerData.name}</name>
 *    <address xm:bind="Address a is customerData.address">
 *      <street>{a.street}</street>
 *      <zip>{a.zip}</zip>
 *    </address>
 *  </customer>
 *  ::
 */
public void addCustomerData(MyStructure customerData)
{  System.out.println("Customer name is " + customerData.get("name"));  System.out.println("Cus
}
```

Note that the is operator is a reserved word in the context of the <xm:bind> attribute. This means that using variables or types called "is" in the value of the <xm:bind> attribute will generate an error.

Note: The xm prefix and its URI are declared implicitly in any JWS file. However, you must declare the namespace prefix and URI to use the prefix in XMLMAP files.

Related Topics

Deprecated. *XML Maps are deprecated as of the WebLogic Platform 8.1 release. For new code, use XQuery maps. For more information, see Introduction to XQuery Maps*.

<xm:java−import> Tag

Specifies Java classes that should be imported to support variables in an XML map file.

Syntax

```
<xm:java−import
    class="fullyQualifiedClassName"
/>
```

Attributes

class

The fully−qualified name of the class.

Remarks

Use the <xm:java−import> tag in an XMLMAP file just as you would use the import directive in a JAVA or JWS file. As with the Java import directive in Java code, you use <xm:java−import> to specify classes and packages needed for type context in XML maps. For example, to use Date as a short type name in XML maps, first import the Date type as follows:

```
<xm:java−import class="java.sql.Date"/>
```

You may use multiple <xm:java−import> tags, but all must occur immediately following the <xm:map−file> tag and before any XML maps.

Note: The xm prefix and its URI are declared implicitly in any JWS file. However, you must declare the namespace prefix and URI to use the prefix in XMLMAP files.

Related Topics

Creating Reusable XML Maps

Deprecated. *XML Maps are deprecated as of the WebLogic Platform 8.1 release. For new code, use XQuery maps. For more information, see Introduction to XQuery Maps.*

<xm:map−file> Tag

Specifies that its contents constitute an XML map file.

Syntax

```
<xm:map−file
    xmlns:xm="http:bea.com/jws/xmlmap"
>
```

Attributes

xmlns:xm

URI defining the namespace for tags with an xm prefix.

Remarks

The <xm:map−file> tag is used only in map files—files with an .xmlmap extension that contain XML maps. A map file must begin and end with <xm:map−file> tags.

Note: The xm prefix and its URI are declared implicitly in any JWS file. However, you must declare the namespace prefix and URI to use the prefix in XMLMAP files. You may change the prefix by declaring another prefix.

Related Topics

Creating Reusable XML Maps

Deprecated. *XML Maps are deprecated as of the WebLogic Platform 8.1 release. For new code, use XQuery maps. For more information, see Introduction to XQuery Maps*.

<xm:multiple> Attribute

Specifies the data structure variable to which a specific repeating XML element should be mapped.

Syntax

```
<elementName
    xm:multiple="[dataType ]arrayMemberVariable in [dataType ]arrayVariable[, …]"
>
```

Attributes

xm:multiple

A statement expressing the mapping between an element's value and a method variable, and the variable's role as a member of an array.

Remarks

Use the <xm:multiple> tag to capture the values of repeating XML elements. In the following XML example, the <part> element (and its children) repeats three times. Because this repetition resembles a list, you can capture the repeating values by mapping them to a data structure.

```
<order>
    <part>
        <partID>19573</partID>
        <partQuantity>1</partQuantity>
    </part>
    <part>
        <partID>28912</partID>
        <partQuantity>1</partQuantity>
    </part>
    <part>
        <partID>39485</partID>
        <partQuantity>57</partQuantity>
    </part>
</order>
```

The <xm:multiple> attribute assumes that your Java code has declared collections to store the values of each repeating element. The syntax of the <xm:multiple> attribute's value essentially assigns each value of the repeating element to a place in the collection, which your code can then inspect iteratively.

Note: The in operator is a reserved word in the context of the <xm:multiple> attribute. This means that using variables or types called "in" in the value of the <xm:multiple> attribute will generate an error.

The following example is designed to operate on an incoming XML message like the preceding example. In this example, the <xm:multiple> attribute specifies that the contents of the <patID> and <numberOfItems> elements should be added as members of the serialNumber and quantity arrays.

```
/**
 * @common:operation
 * @jws:parameter-xml xml-map::
```

```
 * <placeOrder>
 * <order>
 *      <part xm:multiple="String serial in serialNumber, int quant in quantity">
 *         <partID>{serial}</partID>
 *         <numberOfItems>{quant}</numberOfItems>
 *      </part>
 * </order>
 * </placeOrder>
 *
 * ::
 */
public void placeOrder(String[] serialNumber, int[] quantity)
{
      for (int i = 0; i < serialNumber.length; i++)
      {
        System.out.println("Ordered " + quantity[i] + " of part " + serialNumber[i]);
      }
}
```

Note: The xm prefix and its URI are declared implicitly in any JWS file. However, you must declare the namespace prefix and URI to use the prefix in XMLMAP files. You may change the prefix by declaring another prefix.

Related Topics

Handling Repeating XML Values with <xm:multiple>

Deprecated. *XML Maps are deprecated as of the WebLogic Platform 8.1 release. For new code, use XQuery maps. For more information, see Introduction to XQuery Maps.*

<xm:use> Tag

Specifies a function or XML map to invoke for processing XML.

Syntax

```
<xm:use
    call="fileName([dataType ]elementValueVariable)"
/>
```

Attributes

call

A call to an XML map or script function that is external to the service source code.

Remarks

Use the <xm:use> tag when you have an XML map in a map file or ECMAScript functions in a JSX file and you want to invoke the map or script from within your service source code. The value of the call attribute specifies the map or function name, along with parameters to pass to the map or function. The names of parameters in parenthesis must match the names of items in the Java declaration that you are mapping.

fileName can refer to one of the following three types of files:

- An XML map contained in a file with an .xmlmap extension.
- An ECMAScript file with a .jsx extension.

  Using XML programming extensions available with WebLogic Server, you can manipulate XML with ECMAScript. Note that the .jsx file must contain two functions: mymapToXML and mymapFromXML.

Note: The xm prefix and its URI are declared implicitly in any JWS file. However, you must declare the namespace prefix and URI to use the prefix in XMLMAP files. You may change the prefix by declaring another prefix.

Related Topics

Creating Reusable XML Maps

Handling XML with ECMAScript Extensions

Deprecated. *XML Maps are deprecated as of the WebLogic Platform 8.1 release. For new code, use XQuery maps. For more information, see Introduction to XQuery Maps.*

<xm:value> Tag

Specifies the variable to use when mapping a specific XML element content.

Syntax

```
<xm:value
    obj="[dataType ]elementValueVariable[, ...]"
/>
```

Attributes

obj

The portion of the Java declaration to which this element content should be mapped.

Remarks

Use the <xm:value> tag to assign element content from an XML message to a portion of your Java declaration. The following example maps criterionName and criterionValue to the <name> and <value> elements, respectively.

```
/**
 * @common:operation
 * @jws:parameter-xml xml-map::
 * <query>
 *     <xm:attribute name="max_records" obj="resultSize"/>
 *     <element>
 *         <name><xm:value obj="criterionName"/></name>
 *         <value><xm:value obj="criterionValue"/></value>
 *     </element>
 * </query>
 * ::
 */
public void searchData(String criterionName, String criterionValue, int resultSize)
{
    System.out.println("The maximum number of records to return is " + resultSize);
}
```

As an alternative to using the <xm:value> tag, you can use curly braces shorthand. The following snippet is equivalent to the preceding example:

```
/**
 * @common:operation
 * @jws:parameter-xml xml-map::
 * <query max_records="{resultSize}">
 *     <element>
 *         <name>{criterionName}</name>
 *         <value>{criterionValue}</value>
 *     </element>
 * </query>
 * ::
 */
public void searchData(String criterionName, String criterionValue, int resultSize)
```

```
{
    System.out.println("The maximum number of records to return is " + resultSize);
}
```

Specify a data type when the variable is not declared with a type elsewhere in the map's context (such as elsewhere in the map, in Java code at the class level, or in the code for the method to which the map applies).

```
/**
 * @jws:parameter-xml xml-map::
 * <query max_records="{resultSize}">
 *     <element>
 *         <name>{java.sql.String criterionName}</name>
 *         <value>{java.sql.String criterionValue}</value>
 *     </element>
 * </query>
 * ::
 */
public void searchData(String criterionName, String criterionValue, int resultSize)
{
    System.out.println("The maximum number of records to return is " + resultSize);
}
```

Note: The xm prefix and its URI are declared implicitly in any JWS file. However, you must declare the namespace prefix and URI to use the prefix in XMLMAP files. You may change the prefix by declaring another prefix.

Related Topics

How Do XML Maps Work?

Why Use XML Maps?

Deprecated. *XML Maps are deprecated as of the WebLogic Platform 8.1 release. For new code, use XQuery maps. For more information, see Introduction to XQuery Maps.*

<xm:xml−map> Tag

Specifies that its contents constitute an XML map; also provides the signature defining calls to this map.

Syntax

```
<xm:xml−map
    signature="mapSignature"
>
```

Attributes

signature

The signature of map enclosed in <xm:xml−map> tags.

Remarks

The <xm:xml−map> tag is used only in map files — files with an .xmlmap extension that contain XML maps. You must enclose every map in a map file between <xm:xml−map> tags. The signature attribute specifies name of the XML map and parameters expected for a call to the enclosed map. For example, consider the following map:

```
<!−− Defined in a file called "BookMaps.xmlmap" −−>
<xm:xml−map signature="getInventory(String partID)">
    <checkInventory>
        <partID>{partID}</partID>
    </checkInventory>
</xm:xml−map>
```

A call to the map in this example might look something like the following.

```
/**
 * @common:operation
 * @jws:parameter−xml xml−map::
 * <checkInventory>
 *     {BookMaps.getInventoryString(String ISBN)}
 * </checkInventory>
 * ::
 *
 */
public String checkInventory(String ISBN)
{
    return "You checked for copies of " + ISBN + ".";
}
```

Note: The xm prefix and its URI are declared implicitly in any JWS file. However, you must declare the namespace prefix and URI to use the prefix in XMLMAP files. You typically do this in the <xm:map−file> tag.

Related Topics

Creating Reusable XML Maps

<xm:map−file> Tag