Pure XML

XML Binding Namespace

The extensions used to describe XML format bindings are defined in the namespace http://cxf.apache.org/bindings/xformat. CXF tools use the prefix `xformat` to represent the XML binding extensions. Add the following line to your contracts:

```
xmlns:xformat="http://cxf.apache.org/bindings/xformat"
```

Editing

To map an interface to a pure XML payload format do the following:

1. Add the namespace declaration to include the extensions defining the XML binding.
2. Add a standard WSDL `binding` element to your contract to hold the XML binding, give the binding a unique name, and specify the name of the WSDL `portType` element that represents the interface being bound.
3. Add an `xformat:binding` child element to the `binding` element to identify that the messages are being handled as pure XML documents without SOAP envelopes.
4. Optionally, set the `xformat:binding` element's `rootNode` attribute to a valid QName.
5. For each operation defined in the bound interface, add a standard WSDL `operation` element to hold the binding information for the operation's messages.
6. For each operation added to the binding, add the `input`, `output`, and `fault` children elements to represent the messages used by the operation. These elements correspond to the messages defined in the interface definition of the logical operation.
7. Optionally add an `xformat:body` element with a valid `rootNode` attribute to the added `input`, `output`, and `fault` elements to override the value of `rootNode` set at the binding level.
   If any of your messages have no parts, for example the output message for an operation that returns void, you must set the `rootNode` attribute for the message to ensure that the message written on the wire is a valid, but empty, XML document.

XML Messages on the Wire

When you specify that an interface's messages are to be passed as XML documents, without a SOAP envelope, you must take care to ensure that your messages form valid XML documents when they are written on the wire. You also need to ensure that non-CXF participants that receive the XML documents understand the messages generated by CXF.

A simple way to solve both problems is to use the optional `rootNode` attribute on either the global `xformat:binding` element or on the individual message's `xformat:body` elements. The `rootNode` attribute specifies the QName for the element that serves as the root node for the XML document generated by CXF. When the `rootNode` attribute is not set, CXF uses the root element of the message part as the root element when using doc style messages, or an element using the message part name as the root element when using rpc style messages.

For example, if the `rootNode` attribute is not set the message defined below would generate an XML document with the root element `lineNumber`.

```
<type ...>
  ...
  <element name="operatorID" type="xsd:int"/>
  ...
</types>
<message name="operator">
  <part name="lineNumber" element="ns1:operatorID"/>
</message>
```

For messages with one part, CXF will always generate a valid XML document even if the `rootNode` attribute is not set. However, the message below would generate an invalid XML document.

```
<types>
  ...
  <element name="pairName" type="xsd:string"/>
  <element name="entryNum" type="xsd:int"/>
  ...
</types>

<message name="matildas">
  <part name="dancing" element="ns1:pairName"/>
  <part name="number" element="ns1:entryNum"/>
</message>
```

Without the `rootNode` attribute specified in the XML binding, CXF will generate an XML document similar to the one below for the message defined above. The generated XML document is invalid because it has two root elements: `pairName` and `entryNum`.

```
<pairName>
  Fred&Linda
</pairName>
<entryNum>
  123
</entryNum>
```

If you set the `rootNode` attribute, as shown below CXF will wrap the elements in the specified root element. In this example, the `rootNode` attribute is defined for the entire binding and specifies that the root element will be named entrants.

```
<portType name="danceParty">
  <operation name="register">
    <input message="tns:matildas" name="contestant"/>
  </operation>
</portType>

<binding name="matildaXMLBinding" type="tns:dancingMatildas">
  <xmlformat:binding rootNode="entrants"/>
  <operation name="register">
    <input name="contestant"/>
    <output name="entered"/>
</binding>
```

An XML document generated from the input message would be similar to the one shown below. Notice that the XML document now only has one root element.

```
<entrants>
  <pairName>
    Fred&Linda
  <entryNum>
    123
  </entryNum>
</entrants>
```

Overriding the Binding's `rootNode` Attribute Setting

You can also set the `rootNode` attribute for each individual message, or override the global setting for a particular message, by using the `xformat:body` element inside of the message binding. For example, if you wanted the output message to have a different root element from the input message, you could override the binding's root element as shown below.

```
<binding name="matildaXMLBinding" type="tns:dancingMatildas">
  <xmlformat:binding rootNode="entrants"/>
  <operation name="register">
    <input name="contestant"/>
    <output name="entered">
      <xformat:body rootNode="entryStatus"/>
    </output>
  </operation>
</binding>
```