JAXB

Introduction

JAXB is the default data binding for CXF. If you don't specify one of the other data bindings in your Spring configuration or through the API, you will get JAXB. Releases of CXF since 2.3.x have used the JDK7 default of JAXB 2.2, however Maven users running on JDK 6 will need to use the Java endorsed override mechanism to use JAXB 2.2 instead of JAXB 2.1.

JAXB uses Java annotation combined with files found on the classpath to build the mapping between XML and Java. JAXB supports both code-first and schema-first programming. The schema-first support the ability to create a client proxy, dynamically, at runtime. See the CXF DynamicClientFactory class.

CXF uses the JAXB reference implementation. To learn more about annotating your classes or how to generate beans from a schema, please read the JAXB user's guide.

JAXB versus JAX-WS (or other front-ends)

There are some pitfalls in the interaction between the front end and the data binding. If you need detailed control over the XML that travels on the wire, you may want to avoid the 'wrapped' alternative, and stick with 'bare'. When you use the wrapped parameter style or the RPC binding, the front ends construct more or less elaborate XML representations for your operations. You have less control over those constructs than you do over JAXB's mappings. In particular, developers with detailed requirements to control the XML Schema 'elementFormDefault' or the use or non-use of XML namespace prefixes often become frustrated because the JAXB annotations for these options don't effect mappings that are purely the work of the front-end. The safest course is to use Document/Literal/Bare.

Configuring JAXB

CXF allows you to configure JAXB in two ways.

JAXB Properties

JAXB allows the application to specify two sets of properties that modify its behavior: *context* properties and *marshaller* properties. CXF allows applications to add to these properties. **Take care.** In some cases, CXF sets these properties for its own use.

You can add items to both of these property sets via the JAXBDataBinding class. The 'contextProperties' and 'marshallerProperties' *properties* (in the Spring sense) of JAXBDataBinding each store a Map<String, Object>. Whatever you put in the map, CXF will pass along to JAXB. See the JAXB documentation for details.

Example of Configuring a Context Property

```
<jaxws:server id="bookServer"
    serviceClass="org.myorg.mytypes.AnonymousComplexTypeImpl"
    address="http://localhost:8080/act"
    bus="cxf">
    <jaxws:invoker>
      <bean class="org.apache.cxf.service.invoker.BeanInvoker">
        <constructor-arg>
          <bean class="org.myorg.mytypes.AnonymousComplexTypeImpl"/>
        </constructor-arg>
      </bean>
    </jaxws:invoker>
    <jaxws:dataBinding>
      <bean class="org.apache.cxf.jaxb.JAXBDataBinding">
        <property name="contextProperties">
          <map>
            <entry>
              <key><value>com.sun.xml.bind.defaultNamespaceRemap</value></key>
              <value>uri:ultima:thule</value>
            </entry>
          </map>
        </property>
      </bean>
    </jaxws:dataBinding>
</jaxws:server>
```

Activating JAXB Validation of SOAP requests and responses

Please see the FAQ.

Namespace Prefix Management

The JAXB reference implementation allows the application to provide an object that in turn maps namespace URI's to prefixes. You can create such an object and supply it via the marshaller properties. However, CXF provides an easier process. The namespaceMap property of the JAXBDataBinding accepts a Map<String, String>. Think of it as a map from namespace URI to namespace prefix. If you load up this map, CXF will set up the necessary marshaller property for you.

Example of Configuring a Namespace Mapping

```
<jaxws:server id="bookServer"
    serviceClass="org.myorg.mytypes.AnonymousComplexTypeImpl"
    address="http://localhost:8080/act"
    bus="cxf">
    <jaxws:invoker>
      <bean class="org.apache.cxf.service.invoker.BeanInvoker">
        <constructor-arg>
          <bean class="org.myorg.mytypes.AnonymousComplexTypeImpl"/>
        </constructor-arg>
      </bean>
    </jaxws:invoker>
    <jaxws:dataBinding>
      <bean class="org.apache.cxf.jaxb.JAXBDataBinding">
        <property name="namespaceMap">
          <map>
            <entry>
              <key><value>http://cxf.apache.org/anonymous_complex_type/</value></key>
              <value>BeepBeep</value>
            </entry>
          </map>
        </property>
      </bean>
    </jaxws:dataBinding>
</jaxws:server>
```