Standard Customizations In JAX-RPC 2.0

Roberto Chinnici

Senior Staff Engineer

Sun Microsystems, Inc.

Sun microsystems

We make the net work.

Agenda

- Overview/introduction
- WSDL⇨Java mapping
- Java⇨WSDL mapping
- Runtime use of annotations
- Portability and inclusion in J2SE 6.0
- Conclusions

Introduction

Customizations in JAX-RPC 1.1

- Lots of mapping options, but no standard representation thereof
- Examples: do-not-unwrap, use -DataHandler-only, rename-an-operation, ...
- Hard for users, tools, implementors
- Hard to ensure complete TCK coverage
- Stark contrast with JAXB

Proposal for JAX-RPC 2.0

- Standard customizations for both WSDL⇨Java and Java⇨WSDL mappings
- Alignment with JAXB 2.0 (they are one technology to users)
- Use annotations to simplify customization
- Exploit annotations at runtime

WSDL⇨Java

Binding Declarations

- Patterned after JAXB 1.0/2.0
- Standard declaration language
- May appear
  - Embedded in a WSDL document
  - In a separate file
- Aligned and integrated with JAXB's binding declaration language
- Support required by JAX-RPC tools

Declaration Container

```
<jaxrpc:bindings
     wsdlLocation="xs:anyURL"?
     node="xs:string"?
     version="xs:string"?>
   ...binding declarations go here...
</jaxrpc:bindings>
```

- @wsdlLocation points to a WSDL document
- @node is an XPath expression

Declarations

Customizations

- **`jaxrpc:package (name/javadoc)`**
- **`jaxrpc:class (name/javadoc)`**
- **`jaxrpc:method (name/javadoc)`**
- **`jaxrpc:parameter (name)`**

Flags (boolean, inherit value from outer scope)

- **`jaxrpc:enableWrapperStyle`**
- **`jaxrpc:enableAsynchronousMapping`**
- **`jaxrpc:enableAdditionalSOAPHeaderMapping`**
- **`jaxrpc:enableMIMEContent`**

Example 1 - embedded

```
<wsdl:definitions targetNamespace="http://example.org/foo"
                  xmlns:tns="http://example.org/foo"
                  xmlns:stns="http://example.org/bar">
  ...
  <wsdl:portType name="StockQuoteUpdater">
    <wsdl:operation name="setLastTradePrice">
      <wsdl:input message="tns:setLastTradePrice"/>
      <wsdl:output message="tns:setLastTradePriceResponse"/>
      <jaxrpc:bindings>
        <jaxrpc:method name="updatePrice"/>
      </jaxrpc:bindings>
    </wsdl:operation>
    <jaxrpc:bindings>
      <jaxrpc:enableAsynchronousMapping>
        true
      </jaxrpc:enableAsynchronousMapping>
    </jaxrpc:bindings>
  </wsdl:portType>
</wsdl:definitions>
```

Example 1 - standalone

```
<jaxrpc:bindings wsdlLocation="http://example.org/foo.wsdl">
  <jaxrpc:package name="com.acme.foo"/>
  <jaxrpc:bindings node="wsdl:portType[@name='StockQuoteUpdater']">
    <jaxrpc:enableAsynchronousMapping>
      true
    </jaxrpc:enableAsynchronousMapping>
    <jaxrpc:bindings
            node="wsdl:operation[@name='setLastTradePrice']">
      <jaxrpc:method name="updatePrice"/>
    </jaxrpc:bindings>
  </jaxrpc:bindings>
</jaxrpc:bindings>
```

Using JAXB 2.0 Declarations

- Put them in a schema embedded or imported by the WSDL document
- Or insert them in an external binding file, targeting them at a <xs:schema/> element
- JAX-RPC tool will pass the customizations on to the JAXB tool

Example 2 - embedded

```
<wsdl:definitions targetNamespace="http://example.org/foo"
                  xmlns:tns="http://example.org/foo"
                  xmlns:stns="http://example.org/bar">

  <wsdl:types>
    <xs:schema targetNamespace="http://example.org/bar">
      <jaxb:bindings>
        some JAXB binding declarations...
      </jaxb:bindings>
      ...
    </xs:schema>
  </wsdl:types>

  ...
</wsdl:definitions>
```

Example 2 - standalone

```
<jaxrpc:bindings wsdlLocation="http://example.org/foo.wsdl">
  <jaxrpc:bindings node=
     "wsdl:types/xs:schema[targetNamespace='http://example.org/']">
    <jaxb:bindings>
      some JAXB binding declarations...
    </jaxb:bindings>
  </jaxrpc:bindings>
  ...
</jaxrpc:bindings>
```

Java ⇨ WSDL

Web Services Annotations

- Defined by JSR-181
- Used to annotate a hand-written SEI or service implementation class (eg an EJB component)
- Markers (`extends Remote, throws RemoteException`) become optional
- Some fixes needed to align with JAX-RPC 2.0
- Annotation processor generates a WSDL document based on the annotations

Example 3 - SEI

```
@WebService(
   targetNamespace="http://example.org")
public interface StockQuoteUpdater {
   @WebMethod
   void setLastTradePrice(
         @WebParam(name="tickerSymbol")
         String ticker,
         float lastTradePrice);
}
```

Allowed Annotations

- Existing JSR-181 annotations:
  - `@WebService`
  - `@WebMethod`
  - `@WebParam`
  - `@WebResult`
  - `@OneWay`
  - `@SOAPBinding`
- Potential new (JSR-181.next) annotations:
  - `@WebServiceClient`
  - `@WebEndpoint`
  - `@WebFault`
- All JAXB 2.0 annotations

Generating Annotated Classes

- WSDL⇨Java tool can generate annotated classes!
- Consistent view for developers
  - No mapping files needed any more
  - Possible to “import” an interface from an existing service then evolve it in Java
- Consistent view for implementors
  - Single annotation-driven engine for WSDL⇨Java and Java⇨WSDL

Annotations @Runtime

Annotations for Static Info

- All static (type- and signature-related) information can be described with annotations:
  - SEI to WSDL portType/interface
  - Service interface to WSDL service
  - Java to XML schema type mapping
- Marshalling/dispatching engine relies on annotations (with defaults) for all static information

Example 4 - client

```
@WebServiceClient(name="StockQuoteService",
                  targetNamespace="http://example.org",
                  wsdlLocation="...")
public interface MyService extends Service {
  @WebEndpoint(name="stockQuotePort")
  StockQuote getStockQuote();
}

@WebService(name="StockQuoteService")
public StockQuote {
  @WebMethod(name="getStockQuote")
  double getQuote(@WebParam(name="tickerSymbol") String ticker);
}

MyService service = ServiceFactory.loadService(MyService.class);
StockQuote sq = service.getStockQuote();
// or StockQuote sq = service.getPort(StockQuote.class);
double d = sq.getQuote("GOOG");
```

Acquiring Binding Info

- Binding information is more volatile than signatures
  - It changes at deployment time
  - Potentially hundreds of bindings defined
- We don't want people to program to bindings anyway
- Don't put it in annotations – rather, look it up in the WSDL

Inclusion in J2SE 6.0 (Brainstorming)

JAX-RPC 2.0 and J2SE 6.0

- Client and server support
- JMX 2.0 is a big customer
- Annotation-based runtime for both JAX-RPC 2.0 and JAXB 2.0 makes portability easy!
  - No stubs, no ties, no serializers, nothing!
- SOAP/HTTP binding required
- Binding pluggability not a goal for 6.0
  - But we'll keep an eye on JBI and its Binding Components

What's Missing?

- APIs to publish an endpoint
- As easy as

```
@WebService
public class MyService { ... }

MyService impl = new MyService();
Server.publish(impl);
```

- Alternatively, use the Provider interface

What's Missing?

- APIs to specify endpoint address

```
Server.publish(locationURL, impl);
```

- APIs to configure the bindings
  - But we have them on the client... or you could use a WSDL (even partial)
- Lightweight HTTP server functionality
  - Something we can ask J2SE to provide!

Concluding Remarks

Benefits, benefits, BENEFITS!

- Users have a simple, unified model that works across all containers and environments
- JAX-RPC implementors have better test coverage and a unified runtime
- Inclusion in J2SE becomes a lot less painful, with fewer new APIs

Q & A

Roberto Chinnici

roberto.chinnici@sun.com

Sun microsystems

We make the net work.