JBossWS

J2EE-5.0 compatible Web Services

Presented by
Thomas Diesler

March 8, 2005

© JBoss Inc. 2005

Agenda

- Goals
- What's new in WSDL-2.0
- What's new in SOAP-1.2
- What's new in JAXRPC-2.0
- Current Status
- JBossWS Architecture

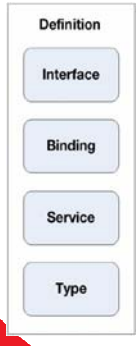
Goals

- JAXB based marshalling layer
- Support SOAP-1.2, WSDL-2.0, Basic Profile 1.1
- Ease of deployment using Java Annotations (JSR175)
- Transport layer based on Remoting
- Session Management
- Simplified Handlers

What's new in WSDL-2.0

- Port renamed to Endpoint
- PortType renamed to Interface
- Interface inheritance
- No more operation overloading

WSDL-2.0 Internals



Definition acts as (XML) container

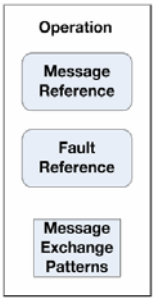
Interface describes WS conceptually

Binding declares transport protocol and message formats

Service associates Interfaces with Endpoints i.e. Bindings

Type declares the data types used in the WS

WSDL-2.0 Interface Operations



MEPs

- In-Only
- Robust In-Only
- In-Out
- In-Multi-Out
- Out-Only
- Robust Out-Only
- Out-In
- Asynchronous Out-In
- Out-Multi-In

What's new in SOAP-1.2

- Specification comes in three parts
  - ✓ Part 0 non-normative introduction
  - ✓ Part 1 describes the structure of SOAP messages, the SOAP processing model and a framework for binding SOAP to underlying protocols.
  - ✓ Part 2 describes optional add-ins to the core of SOAP including a data model and encoding, an RPC convention and a binding to HTTP.

7

SOAP-1.2 changes

- Infoset
  - ✓ In SOAP 1.2 it is left to the specification of a binding to an underlying protocol to specify the XML serialization used in underlying protocol data units.
- Trailers
  - ✓ SOAP 1.1 allows additional elements to follow the SOAP Body element, SOAP 1.2 disallows these
- Actors and Roles
  - ✓ SOAP 1.1 has the actor attribute. In SOAP 1.2 this attribute is renamed to role. The semantics of the attribute are unchanged.

8

SOAP-1.2 changes

- New Fault Codes
  - ✓ DTDNotSupported - generated when a received message contains a document type declaration
  - ✓ DataEncodingUnknown - generated when a received message uses an unrecognised value of the encoding-Style attribute
  - ✓ The SOAP 1.1 Client fault code is renamed Sender in SOAP 1.2
  - ✓ The SOAP 1.1 Server fault code is renamed Receiver in SOAP 1.2
- Hierarchical Fault Codes
  - ✓ SOAP 1.1 allows extension of fault codes using a dot notation, SOAP 1.2 disallows this and provides a more XML-like representation instead

9

SOAP-1.2 changes

- Misunderstood Header
  - ✓ SOAP 1.2 adds a new standard header for reporting additional information in MustUnderstand faults.
- Versioning Mechanism
  - ✓ A SOAP 1.2 node that receives a SOAP 1.1 message will respond with a SOAP 1.1 envelope containing a SOAP 1.1 VersionMismatch fault.
  - ✓ A SOAP 1.2 node that receives a message from any other version of SOAP (including future versions) will respond with a SOAP 1.2 envelope containing a SOAP 1.2 VersionMismatch fault and an Upgrade header with a list of the supported envelope versions.

10

SOAP-1.2 Processing Model

- SOAP 1.2 clarifies the processing model of SOAP 1.1

1. Determine the roles the node is to play. I.e. the values of the role attribute for which it will assume processing liability.
2. Identify mandatory headers targeted to the node. I.e. header marked with a mustUnderstand=true attribute and a role attribute with a value matching one of the roles determined in step 1.
3. Generate a MustUnderstand fault if one or more of the headers identified in step is not understood. I.e. the node does not contain software for processing the header.
4. Process headers and, if the node is the ultimate recipient, the body of the message.
5. If the node is acting as an intermediary remove headers targeted at the node and forward the message.

11

What's new in JAXRPC-2.0

- XML binding is delegated to JAXB-2.0
- Added support for WS-I BasicProfile 1.1
- Use of Java annotations (JSR 175, JSR 181)
- Improve support for document centric usage
- Simplify Handlers and allow handler/handler and handler/service endpoint collaboration
- Versioning and Evolution of Web Services
- SOAP encoding has been deprecated
- No support for Java prior to J2SE-1.5

12

JAXRPC-2.0 changes

- Client Dispatch interface provides support to work at the XML representation level
  - ✓ **Synchronous request response** (invoke methods) The method blocks until the remote operation completes and the results are returned.
  - ✓ **Asynchronous** request response (invokeAsync methods) The method returns immediately, any results are provided either through a callback or via a polling object.
  - ✓ **One-way** (invokeOneWay methods) The method is logically non-blocking, subject to the capabilities of the underlying protocol, no results are returned.

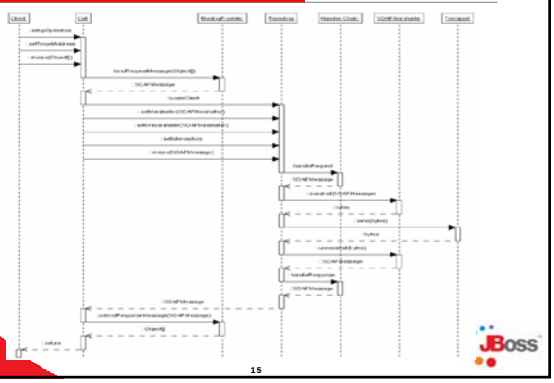
13

JAXRPC-2.0 changes

- Service Provider interface provides support to work at the XML representation level
  - ✓ Invoke is called with a logical request message, logical response message and an invocation context
- Handler Framework
  - ✓ The handler chain for a binding is constructed by merging the applicable per-service, per-port or per-protocol binding chains configured for the service instance.

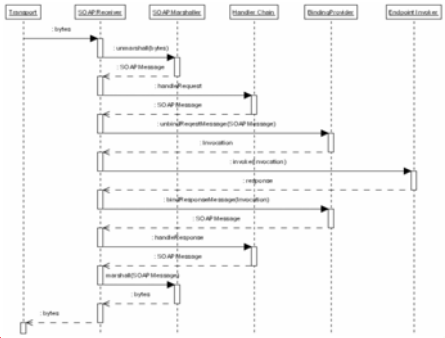
14

JBossWS Client



15

JBossWS Service



16

Binding Provider

- The central piece of SOAP message binding is the BindingProvider

```
/** An implementation of this interface transforms the Java call parameters to a SOAPMessage and vice versa.
 *
 * @author Thomas.Diesler@jboss.org
 * @since 16-Oct-2004
 */
public interface BindingProvider
{
   /** On the client side, generate the SOAPMessage from IN parameters. */
   SOAPMessage bindRequestMessage(OperationDesc opDesc, Object[] inParams, Map unboundHeaders)

   /** On the server side, extract the IN parameters from the SOAPMessage and populate an Invocation object */
   EndpointInvocation unbindRequestMessage(OperationDesc opDesc, SOAPMessage reqMessage)

   /** On the server side, generate the SOAPMessage from OUT parameters in the Invocation object. */
   SOAPMessage bindResponseMessage(OperationDesc opDesc, EndpointInvocation epInv)

   /** On the client side, extract the OUT parameters from the SOAPMessage and return them to the client. */
   void unbindResponseMessage(OperationDesc opDesc, SOAPMessage resMessage, Map outParams, Map unboundHeaders))
```

17

RPC vs. Document style

- RPC style services
  - ✓ is easy to dispatch because of the RPC element that names the endpoint operation
  - ✓ cannot be validated because there is no complex schema type that completely defines the payload
  - ✓ there is a disconnect in service description between the WSDL and type defining schema

```
<soap:Envelope>
   <soap:Header> +
      <header-element> *
   </soap:Header>
   <soap:Body>
      <rpc-wrapper-element>
         <rpc-parameter> *
      </rpc-wrapper-element>
   </soap:Body>
</soap:Envelope>
```

18

RPC vs. Document style

- Document style services
  - ✓ Can be validated because the payload must be defined by a single complex type in XML schema
  - ✓ Marshalling layer can utilize the schema definition
  - ✓ Business parties agree on exchanging complex documents, no need to know RPC details
  - ✓ Framework may unwrap the payload

```
<soap:Envelope>
   <soap:Header> +
      <header-element> *
   </soap:Header>
   <soap:Body>
      <single-document-element>
   </soap:Body>
</soap:Envelope>
```

19

SAAJ view

- The SAAJ API offers a DOM based object view of the SOAP envelope
- SOAPHeaderElement, SOAPBodyElement extend SOAPElement, which implements org.w3c.dom.Element
- JBossWS chops the message into fragments

```
<soap:Envelope>
  <soap:Header>
    <tns:SimpleHeader>123456</tns:SimpleHeader>
    <tns:ComplexHeader>
      <name>Tom</name>
      <age>3</age>
    </tns:ComplexHeader>
  </soap:Header>
  <soap:Body>
    <tns:PurchaseOrder>
      <item>Ferrari</item>
      <address>
        <street>Wall Street 102</street>
        <city>New York</city>
      </address>
    </tns:PurchaseOrder>
  </soap:Body>
</soap:Envelope>
```

```
<tns:SimpleHeader>123456</tns:SimpleHeader>
```

```
<tns:ComplexHeader>
  <name>Tom</name>
  <age>3</age>
</tns:ComplexHeader>
```

```
<tns:PurchaseOrder>
  <item>Ferrari</item>
  <address>
    <street>Wall Street 102</street>
    <city>New York</city>
  </address>
</tns:PurchaseOrder>
```

20

XML Infoset vs. Java Objects

- The SOAPContentElement marshals/unmarshals the XML fragments on demand

```
/**
 * A SOAPElement that gives access to its content as XML fragment or Java object.
 *
 * The SOAPContentElement has three content representations, which may exist in parallel.
 * The getter and setter of the content properties perform the conversions.
 * It is the responsibility of this objects to keep the representations in sync.
 *
 * +----------+         +-------------+          +-------------+
 * | Object   | <-----> | XMLFragment |  <-----> | DOMTree     |
 * +----------+         +-------------+          +-------------+
 *
 * The idea is, that jaxrpc handlers can work with both the object and the DOM view of this SOAPElement.
 * Note, that state transitions may be expensive.
 *
 * @author Thomas.Diesler@jboss.org
 * @since 13-Dec-2004
 */
public class SOAPContentElement extends SOAPElementImpl
```

21

JBossWS status

- JAXRPC Client using DII
- JAXRPC Client configured from WSDL
- WS4EE Client deployment
- WS4EE service deployment JSE
- WS4EE service deployment EJB
- Client/Server side handler chains
- Attachment support
- INOUT parameters
- Bound and unbound headers
- Support for all JAXRPC primitive types
- RPC style web services
- Document style web services
- JAXB support for complex types
- Pass JBoss WS4EE test suite (> 200 tests)
- Pass Sun compatibility test suite (>2200 tests)

22

Beyond WS4EE

- WS-Security describes enhancements to SOAP messaging to provide quality of protection through message integrity, message confidentiality, and single message authentication.
- WS-Addressing provides transport-neutral mechanisms to address Web services and messages. Specifically, this specification defines XML elements to identify Web service endpoints and to secure end-to-end endpoint identification in messages.
- WS-Transaction defines mechanisms for transactional interoperability between Web services domains and provide a means to compose transactional qualities of service into Web services applications.

23

JBossWS links

- TODO, issue tracking, road map
  http://jira.jboss.com/jira/browse/JBWS
- Docs, Tutorials & Specs
  http://www.jboss.org/wiki/Wiki.jsp?page=JBossWS
- Design Discussions
  http://www.jboss.org/index.html?module=bb&op=viewforum&f=174
- User Questions
  http://www.jboss.org/index.html?module=bb&op=viewforum&f=200

24

The team

- Project Lead
  thomas.diesler@jboss.com
- JBossWS Tools
  anil.saldhana@jboss.com
- JAXB marshalling layer
  alexey.loubyansky@jboss.com
- Core Service & Security
  jason.greene@jboss.com

25