Web Service development with Apache Axis2 and ODE

SYSTEMATIC THOUGHT LEADERSHIP FOR INNOVATIVE BUSINESS

Benjamin Schmeling
SAP Research

Talk Outline

- Apache Axis2
  - Overview and Introduction
  - Tutorial
    - Installation & Administration
    - Bottom up Web Service development and deployment
    - Client Stub generation
    - Security enforcement
- Apache ODE
  - Overview and Introduction to BPEL
  - Tutorial
    - Step by step BPEL process development with Visual BPEL designer
    - BPEL deployment
    - Invocation of the BPEL process

Axis2 Overview

- Open-source Web Service Engine from Apache completely written in Java
  - There is also a version written in C Axis2/C
- Follow-up to the popular Axis Project with improvements in
  - Efficiency
  - Flexibility
  - Modularity
  - XML Orientation
- Axis2 was introduced during August 2004 Summit in Colombo, Sri Lanka

Axis2 Technologies

- SOAP 1.1, SOAP 1.2, REST
  - MTOM, XOP, Soap with Attachments
- WSDL 1.1, WSDL 2.0
- Transport protocols
  - HTTP, SMTP, JMS, TCP, FTP...
- SAAJ 1.1
- WS-Policy
- WS-Addressing
- Modules for
  - WS-ReliableMessaging 1.0 and 1.1
  - WS-Security 1.0, 1.1

Pluggable Modules

- Sandesha
  - WS-ReliableMessaging 1.0, 1.1
  - WS-MakeConnection 1.0
  - WS-RM Policy

- Rampart
  - WS-Security 1.0, 1.1
  - WS-Secure Conversation
  - WS-Security Policy 1.1, 1.2
  - WS-Trust

- Kandula
  - WS-Coordination
  - WS-AtomicTransaction
  - WS-BusinessActivity

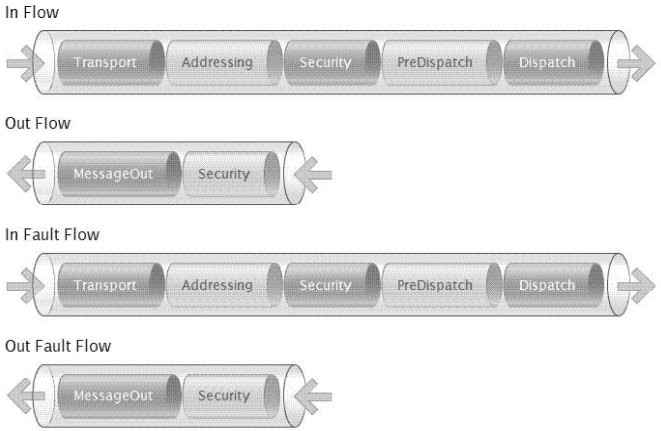
Service Development Approaches

- Bottom-up
  - Java Beans
  - EJB3
  - Spring
  - JAX-WS
  - Creating from Scratch
- Top-down
  - Skeleton generation from WSDL with WSDL2Java
  - JAX-WS skeleton generation from WSDL with wsimport
- Databindings
  - Axis Data Binding (ADB)
  - XML Beans
  - JiBX
  - JAXB 2.0

Handlers & Phases

- **Handler**
  - Stateless message interceptor with read and write access to the SOAP messages
  - Modular way to address non-functional concerns
- **Phase**
  - Supports dynamic ordering of handlers
  - Can be defined as a logical collection of handlers
  - Multiple phases define a flow

Flows



ODE Features

- WS-BPEL 2.0 and BPEL4WS 1.1 support
- Core based on Axis2
- Supports JBI (Java Business Integration) standard
- BPEL Extensions provided
  - Stateful exchange protocol
  - Atomic Scopes (WS-AT)

BPEL Activity Overview

- Relevant BPEL Concepts for this tutorial
  - Variables
  - Partner Links
  - Partner Link Types
  - Activities
- Messaging Activities
  - Receive
  - Reply
  - Invoke
- Activities to manipulate data
  - Assign
- Structured Activities
  - Scope
  - Sequence
  - Flow

BPEL
Messaging Activities: Receive and Reply


```
<bpel:receive name="receive"
  partnerLink="client"
  portType="tns:TestBPEL"
  operation="process"
  variable="input"
  createInstance="yes"/>
```

```
<bpel:reply name="reply"
    partnerLink="client"
    portType="tns:TestBPEL"
    operation="process"
    variable="output" />
```

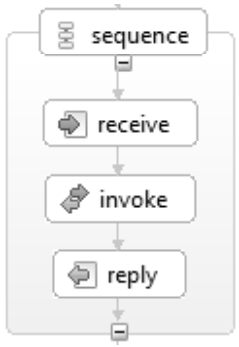

BPEL
Messaging Activities: Invoke

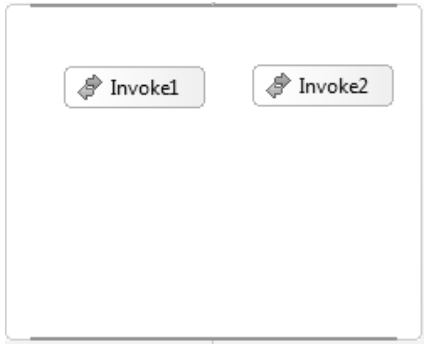

```
<bpel:invoke name="SearchFlight"
  partnerLink="FlightPartner"
  operation="searchBestFlight"
  portType="myns:MyFlightReservationServicePortType"
  inputVariable="searchBestFlightRequest"
  outputVariable="searchBestFlightResponse">

</bpel:invoke>
```

BPEL
Basic Activities: Assign


```
<bpel:assign validate="no" name="assign">
  <bpel:copy
      <bpel:from part="payload" variable="fromVar">
            <bpel:query>departureDate</bpel:query>
      </bpel:from>
      <bpel:to part="parameters" variable="toVar">
            <bpel:query>departureDate</bpel:query>
      </bpel:to>
   </bpel:copy>
  </bpel:assign>
```


BPEL
Structured Activities: Sequence






```
<bpel:sequence name="sequence">
      <bpel:receive/>
      <bpel:invoke />
      <bpel:reply  />
</bpel:sequence>
```

```
<bpel:flow name=„flow">
      <bpel:invoke />
      <bpel:invoke />
</bpel:flow>
```


Tutorials and Links

http://ws.apache.org/axis2/

http://ws.apache.org/sandesha/sandesha2/

http://ws.apache.org/rampart/

http://ws.apache.org/kandula/2/

http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html

http://ode.apache.org/

http://ode.apache.org/links.html

Thank you for your attention

Thank you for your attention!

Axis Installation

- Prerequisites
  - Servlet Container, e.g. Apache Tomcat
  - Apache Axis2 binary distribution
- Deploy the axis2.war file
- Go to http://localhost:8080/axis2/ and validate your installation
- In the webapps folder of Tomcat the axis2 folder contains
  - WEB-INF/conf/axis2.xml is the main configuration file of Axis (activate hotdeployment)
  - WEB-INF/services containing all deployed service archives
  - WEB-INF/modules containing the additional modules you have installed
  - WEB-INF/classes & WEB-INF/lib you can add additional classes/jars
- Module Installation
  - Copy the <module_name>.mar to the modules and required libraries to lib folder

Axis Administration

The Apache Software Foundation
http://www.apache.org/

Tools
- Upload Service

System Components
- Available Services
- Available Service Groups
- Available Modules
- Globally Engaged Modules
- Available Phases

Execution Chains
- Global Chains
- Operation Specific Chains

Engage Module
- For all Services
- For a Service Group
- For a Service
- For an Operation

Services
- Deactivate Service
- Activate Service
- Edit Parameters

Contexts
- View Hierarchy

Back | Log out

Available Modules

- **ping** : module description not found
- **kandula-outflow** : module description not found
- **soapmonitor** : module description not found
- **rahas** : This module is used to STS enable a service where it adds the RequestSecurityToken operation to a service that the module is engaged to
- **mtompolicy** : This is the MTOM policy module. It is engaged when we have MTOM policy assertion.
- **addressing** : This is the WS-Addressing implementation on Axis2, supporting the WS-Addressing 1.0 Recommendation, as well as the Submission version (2004/08).
- **rampart** : This module provides the WS-Security and WS-SecureConversation functionalities for Axis2, based on Apache WSS4J, Apache XML-Security and Apache Rahas implementations.
- **kandula-inflow** : module description not found
- **metadataExchange** : module description not found
- **sandesha2** : This module implements WS-ReliableMessaging for Axis2. This implements both the WSRM submitted spec and up to the version CD4 of the new WSRM 1.1 spec.
- **metadataExchange** : module description not found
- **jaxws** : This is Axis2 implementation of JAX-WS
- **script** : module description not found

Service Development and Deployment

- Eclipse 3.4 for Java EE developers http://www.eclipse.org/downloads/packages/release/ganymede/sr2
- Implement your service as Java class
- Create the META-INF/services.xml configuration
  - Use *org.apache.axis2.rpc.receivers.RPCMessageReceiver*
  - Add the *ServiceClass* parameter pointing to your implementation class
  - Documentation at http://ws.apache.org/axis2/1_5/axis2config.html
- Export the project as jar file and rename it to .aar
- Go to the Administration (user: admin, password: axis2) and navigate to the Upload section
- Upload your axis archive
- You can directly call your service via
  - http://localhost:8080/axis2/services/<ServiceName>/<OperationName>?param1=x&param2=y
- The SOAP messages can be seen using the SOAP Monitor (must have been engaged for the service)
  - Add the applet classes to the axis2 folder
  - Insert the applet into the WEB-INF/web.xml
  - The location of the monitor is http://localhost:8080/axis2/SOAPMonitor
  - Documentation at http://ws.apache.org/axis2/1_5/soapmonitor-module.html

Client Stub Generation

- Create a new Java Project and
  - Add the Axis2 libraries to the project class path using a user library
  - Run the class org.apache.axis2.wsdl.WSDL2Java with arguments
    -uri http://localhost:8080/axis2/services/<servicename> -S generated-src -or
- In order to engage modules for your stub
  - Create a new folder, e.g. named axis2_client
  - Create a subfolder modules containing .mar files of the required modules
  - Copy the axis2.xml config file to the axis2_client folder and rename to client_axis2.xml
  - Pass the configuration context object pointing to your axis2 client config to the stub constructor

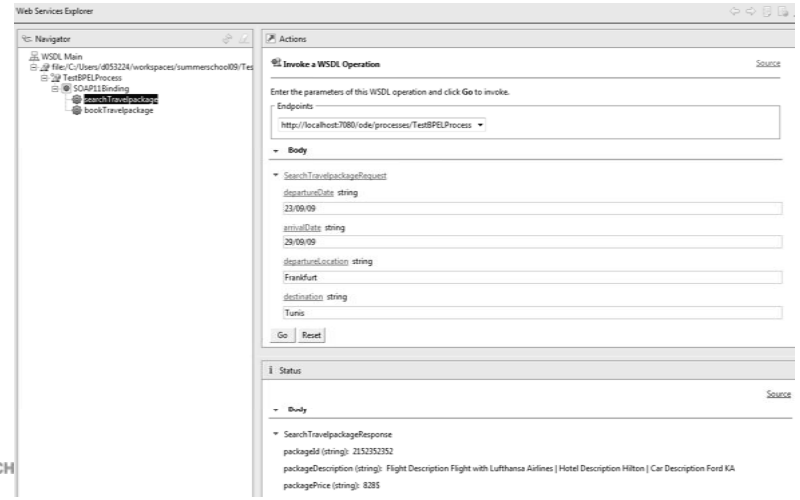
```
String axis2_xml = CLIENT_REPO_PATH + File.separator +"client_axis2.xml";

ConfigurationContext configContext =
   ConfigurationContextFactory.createConfigurationContextFromFileSystem(CLIENT_RE
   PO_PATH,axis2_xml);

stub._getServiceClient().engageModule("addressing");
```

ODE and Visual Designer

- Copy the ode.war from the ODE distribution to your Tomcat webapps folder
- Install the BPEL Visual Designer from the Eclipse update site:
  - http://download.eclipse.org/technology/bpel/update-site/
- Create a new BPEL project.
  - Add the BPEL Facet to the project Configuration->Modify
- Create a new BPEL file
  - Use synchronous template
  - Add the name and namespace
- Modify the BPEL process with the editor and the corresponding WSDL interface
- Add a new ODE server runtime in the server view
- Restart eclipse and add your project to the server

ODE and Visual Designer

- Click on the WSDL of the BPEL file and right-click
  - Select Web Services -> Test with Web Services Explorer



SAP RESEARCH