Introduction to Apache Axis2: Next Generation Web Services

Asst. Prof. Dr. Kanda Runapongsa
(krunapon@kku.ac.th)
Department of Computer Engineering
Khon Kaen University

Agenda
- What and Why Apache Axis2?
- Understanding AXIOM
- Learning Axis2 Basics
- Understanding the Deployment Model
- Axis2 Installation
- Writing a Service and Deploying using POJO (Plain Old Java Object)

What is Apache Axis2?
- Apache Axis2 is the core engine for Web services
- It is a complete re-design and re-write of widely used Apache Axis
What is Apache Axis?

- Apache Axis is an implementation of the SOAP
  - An envelope that defines a framework for describing what is in a message and how to process it
  - A set of encoding rules for expressing instances of application-defined data types
  - A convention for representing remote procedure calls and responses.

Why Apache Axis2?

- More flexible, efficient, and configurable in comparison to Axis1.x
- Supports SOAP 1.1 and SOAP 1.2 as well as REST style of Web services
- Support the easy addition of plug-in “modules”
  - WS-ReliableMessaging
  - WS-Coordination and WS-AtomicTransaction
  - WS-Security
  - WS-Addressing

Key Features of Apache Axis2 (1/4)

- Speed: much faster than the old version
  - Use its own object model and StAX
- Low memory foot print
- AXIOM: extensible, highly performance, and developer convenient
  - Light-weight object model
Key Features of Apache Axis2 (2/4)

- **Hot deployment**
  - New services can be added without having to shut down the server

- **Asynchronous Web services**
  - Supports asynchronous Web services invocation using non-blocking clients and transports

- **MEPs support**
  - Support Message Exchange Patterns (MEPs)

Key Features of Apache Axis2 (3/4)

- **Transport framework**
  - A clean and simple abstraction for integrating and using Transports (over various protocols such as SMTP and FTP)

- **WSDL support**
  - Axis2 supports WSDL 1.1 and 2.0 which allows you to easily build stubs to access remote services

Key Features of Apache Axis2 (4/4)

- **Add-ons**
  - Have been incorporated with WSS4J for security, Sandesha for reliable messaging, Kandula for coordination, atomic transaction, and business activity

- **Compositions and Extensibility**
  - Modules and phases improve support for comosability and extensibility
Supported Specifications
- SOAP 1.1 and 1.2
- Message Transmission Optimization Mechanism (MTOM), XML Optimized Packaging (XOP) and SOAP with Attachments
- WSDL 1.1, including both SOAP and HTTP bindings
- WS-Addressing
- WS-Policy
- SAAJ 1.1

Supported Transports and Data Bindings
- Supported Transports
  - HyperText Transfer Protocol (HTTP)
  - Simple Mail Transfer Protocol (SMTP)
  - Java Message Service (JMS)
  - Transmission Control Protocol (TCP)
- Supported Data Bindings
  - Axis Data Binding (ADB)
  - XMLBeans
  - JibX
  - JaxMe

Tools Included in Axis2 version 1.1
- Axis2 Web Application (Web App)
- WSDL2WS
  - Eclipse plug in / Maven2 WSDL2Code Plug in
- Service Archive Wizard
  - Eclipse plug in / Maven2 AAR Plug -in
- Java2WSDL
  - Maven 2 Java2WSDL Plug-in
Extension Modules

- Apache Rampart: Supporting WS-Security (and soon WS-Policy)
- Apache Sandesha2: Supporting WS-Reliable Messaging
- Apache Axis2 comes built in with a module that supports WS-Addressing

Agenda

- What and Why Apache Axis2?
- Understanding AXIOM
- Learning Axis2 Basics
- Understanding the Deployment Model
- Axis2 Installation
- Writing a Service and Deploying
- Writing a Module and Deploying

AXIOM (AXIS Object Model)

- Objects are created “on demand” using a pull model
- Allows direct access to the underlying pull stream with or without building the tree
- Allows the event based navigation of the OM tree
- Support for storing binary data
Agenda

- What and Why Apache Axis2?
- Understanding AXIOM
- Learning Axis2 Basics
- Understanding the Deployment Model
- Axis2 Installation
- Writing a Service and Deploying
- Writing a Module and Deploying

Message Processing Stages

- There are three main stages
  - Transport Receiver
    - Transport related processing
  - Dispatching
    - Finding service and operation
  - Message Receiver
    - Last handler of the chain

Contexts and Descriptions Hierarchy

- Descriptors keep static information
  - Information extracted from deployment descriptors
- Contexts keep runtime information
- This information needs to be in various scope
- Good to keep contexts and descriptions separate
Agenda

- What and Why Apache Axis2?
- Understanding AXIOM
- Learning Axis2 Basics
- Understanding the Deployment Model
- Axis2 Installation
- Writing a Service and Deploying
- Writing a Module and Deploying

What’s the Fuss with Deployment

- Axis 1.x deployment requires you to
  - Modify the XML files
  - Call the admin client
  - Add to the classpath
  - Restart the server
- For a beginner, a bit of headache 😞

New Deployment Model

- Archive based deployment
  - Bundle all together and drop in
- Directory based deployment (similar structure as archive)
- Hot deployment 😊
- An archive file can contain
  - Class files
  - Third party libraries
  - Any other resources required by the service
Axis2 Service
- Can be deployed as an archive (.aar) file or as a directory with all necessary resources
- Service configurations are given by the `services.xml` which contains
  - ServiceClass parameter
  - Namespaces
  - Expose transports
  - Operation
  - Modules to be engaged
  - Module configurations

Agenda
- What and Why Apache Axis2?
- Understanding AXIOM
- Learning Axis2 Basics
- Understanding the Deployment Model
- Axis2 Installation
- Writing a Service and Deploying
- Writing a Module and Deploying

Axis2 Installation
- Download and install Java SDK
- Download and deploy a servlet container such as Apache Tomcat
- Download distribution from [http://ws.apache.org/axis2](http://ws.apache.org/axis2)
- Copy `axis2.war` to `<TOMCAT>/webapps` folder
- Start Tomcat server
- Start URL at [http://localhost:8080/axis2](http://localhost:8080/axis2)
Download and Install Java SDK

- Download and install JavaSE SDK v1.5 update 9 which can be downloaded from
  - http://gear.kku.ac.th/~krunapon/xmlws/Tools/JDK 1.5.0.09 with NetBeans Bundle
- Note that JavaSE 6 has compatibility problems with Apache Ant tool
- Set %JAVA_HOME% environment variable to the directory at where JavaSE SDK 1.5 is located

Download a Servlet Container (Tomcat)

- In this case, we use Tomcat 5.5.20 which can be downloaded from
  - http://tomcat.apache.org/Download Tomcat 5.x
  - http://gear.kku.ac.th/~krunapon/xmlws/Tools/Tomcat 5.5.20
- Unzip apache-tomcat-5.5.20.zip

Deploy and Start Tomcat Server

- Set %CATALINA_HOME% environment variable to the Tomcat directory
- Start Tomcat server by running startup.bat which is at %CATALINA_HOME%/bin
Open the Homepage of Tomcat

http://localhost:8080

Download Apache Axis2 (1/2)
- Download Apache Axis2
  http://ws.apache.org/axis2/download.cgi
- Choose version 1.1.1

Download Apache Axis2 (2/2)
- Choose Standard Binary Distribution

- Extract axis2-1.1.1.zip
- Set %AXIS2_HOME% to the axis2 directory
Install Apache Axis2 Overview

- Obtain axis2.war which can use one of these two methods
  - Download axis2.war from http://ws.apache.org/axis2/download/1_1_1/download.cgi
  - Create axis2.war by using Apache Ant which can be downloaded from http://ant.apache.org/bindownload.cgi
- Copy axis2.war to %CATALINA_HOME%/webapps directory

Download axis2.war

- In Distribution Name column, choose WAR (Web Archive Distribution)
- In Download column, choose zip

Creating axis2.war using Apache Ant

- Download Apache Ant which can be downloaded from http://ant.apache.org/bindownload.cgi
- Set %PATH% environment variable to %ANT_HOME%/bin
- Run ant create.war at directory %AXIS2_HOME%/webapp
  - “ant” command needs the input source file “build.xml”
Running Apache Ant

Copy axis2.war

http://localhost:8080/axis2
List Available Services

http://localhost:8080/axis2/services/listServices

Axis2 Happiness Page

http://localhost:8080/axis2/axis2-web/HappyAxis.jsp
http://localhost:8080/axis2

Axis2 Administration Console
http://localhost:8080/axis2/axis2-admin/

Axis2 Web Admin Module
Axis2 Services Context

- The Directory Structure of axis2.war
  - axis2-web
  - META-INF
  - WEB-INF
    - classes
      - conf
        - axis2.xml
    - lib
    - modules
    - services
      - services.list
      - version.aar
    - web.xml

Axis2 Services Structure

- Services can be deployed as *.aar files
- The *.aar can be built using ant command in the directory that has build.xml

Deploy Sample Web Service: StockQuoteService

- Go to %AXIS2_HOME%/samples/quickstart which has the structure as follows
  - resources
    - META-INF
    - Services.xml
  - src
    - Java files
    - build.xml
- Create StockQuoteService.aar by running command “ant generate.service”
Create StockQuoteService.aar

Deploy StockQuoteService

Deployed StockQuoteService
WSDL of StockQuoteService

http://localhost:8080/axis2/rest/StockQuoteService/getPrice?symbol=IBM

This XML file does not appear to have any style information associated with it. The document tree is shown below:

```xml
<ns:getPriceResponse>
  <ns:return>42.0</ns:return>
</ns:getPriceResponse>
```

Call Stock Quote Service


Update Stock Quote Price

Call Stock Quote Service after Update


This XML file does not appear to have any style information associated with it. The document tree is shown below:

```
<ns:getPriceResponse>
  <ns:return>100.0</ns:return>
</ns:getPriceResponse>
```

Create CalculatorService Overview

- Create directory CalculatorService
  - Edit Java source file
  - Edit services.xml
  - Edit build.xml
- Use ant to create CalculatorService.aar
- Copy CalculatorService.aar to %CATALINAME_HOME%/webapps/axis2/WEB-INF/services

Create CalculatorService

- Create directory CalculatorService
  - (May copy from StockQuoteService)
    - resources
      - META-INF
        - Services.xml
    - src
      - service
        - pojo
          - CalculatorService.java
      - build.xml
      - README.txt
Edit Java Source File (1/2)

package service.pojo;
public class CalculatorService {
    public int add(int a, int b) {
        return a+b;
    }
    public int sub(int a, int b) {
        return a-b;
    }
}

Edit Java Source File (2/2)

public int mult(int a, int b) {
    return a*b;
}

public int div(int a, int b) {
    if (b != 0)
        return a/b;
    else
        return 0;
}

Edit services.xml

<service name="CalculatorService" scope="application" targetNamespace="http://quickstart.samples/">
    <description>
        Calculator Service
    </description>
    <messageReceivers>
        <messageReceiver mep="http://www.w3.org/2004/08/wsdl/in-only"
            class="org.apache.axis2.rpc.receivers.RPCMessageReceiver"/>
        <messageReceiver mep="http://www.w3.org/2004/08/wsdl/in-out"
            class="org.apache.axis2.rpc.receivers.RPCMessageReceiver"/>
    </messageReceivers>
    <schema schemaNamespace="http://quickstart.samples/xsd"/>
    <parameter name="ServiceClass">service.pojo.CalculatorService</parameter>
</service>
Edit build.xml (1/3)

```xml
<project basedir="." default="generate.service">
  <property environment="env"/>
  <property name="AXIS2_HOME" value="${env.AXIS2_HOME}"/>
  <property name="build.dir" value="build"/>
  <path id="axis2.classpath">
    <fileset dir="${AXIS2_HOME}/lib">
      <include name="*.jar"/>
    </fileset>
  </path>
  <target name="compile.service">
    <mkdir dir="${build.dir}"/>
    <mkdir dir="${build.dir}/classes"/>
    <!--First let's compile the classes-->
    <javac debug="on"
      fork="true"
      destdir="${build.dir}/classes"
      srcdir="${basedir}/src"
      classpathref="axis2.classpath">
    </javac>
  </target>
</project>
```

Edit build.xml (2/3)

```xml
<target name="generate.wsdl" depends="compile.service">
  <taskdef name="java2wsdl"
    classname="org.apache.ws.java2wsdl.Java2WSDLTask"
    classpathref="axis2.classpath"/>
  <java2wsdl className="service.pojo.CalculatorService"
    outputLocation="${build.dir}"
    targetNamespace="http://quickstart.samples/"
    schemaTargetNamespace="http://quickstart.samples/xsd">
    <classpath>
      <pathelement path="${axis2.classpath}"/>
      <pathelement location="${build.dir}/classes"/>
    </classpath>
  </java2wsdl>
</target>
```

Edit build.xml (3/3)

```xml
<target name="generate.service" depends="compile.service">
  <!--aar them up-->
  <copy toDir="${build.dir}/classes" failonerror="false">
    <fileset dir="${basedir}/resources">
      <include name="**/*.xml"/>
    </fileset>
  </copy>
  <jar destfile="${build.dir}/CalculatorService.aar">
    <fileset excludes="**/Test.class" dir="${build.dir}/classes"/>
  </jar>
  <target>
  </target>
  <target name="clean">
    <delete dir="${build.dir}"/>
  </target>
</project>
```
Use ant to create CalculatorService.aar

- Go to directory CalculatorService
- Run command “ant generate.service”

Now we have CalculatorService.aar in directory CalculatorService\build

Deploy CalculatorService

- Copy CalculatorService.aar from CalculatorService\build to %CATALINA_HOME%/webapps/axis2/WEB-INF/services

Deployed CalculatorService
WSDL of CalculatorService

Call Calculator Service (Add Operation)


This XML file does not appear to have any style information associated with it. The document tree is shown below.

```xml
<ns:add Response>
    <ns:return>5</ns:return>
</ns:add Response>
```

References

- Deepal Jayasinghe and Ruchith Fernando, "Building Enterprise Applications with Axis2"
- Chathura Herath and Eran Chinthaka, "Axis 2 Tutorial"