Demystifying AXIS WSDD

Hermod Opstvedt

Chief Architect

DnB NOR ITU

Colorado Software Summit

Demystifying AXIS WSDD

- What is Axis?
  - An implementation of the SOAP ("Simple Object Access Protocol") as defined from w3c.
    - Simple stand-alone server.
    - Server which plugs into servlet engines such as Tomcat.
    - Extensive support for the *Web Service Description Language (WSDL)*.

Demystifying AXIS WSDD

- Axis
  - Deploying WebServices in Axis
    - Alt. 1 – As a .jws file

      This means putting your .java file in a folder of your choice under your webapp directory and renaming it .jws. Note that there is no sense of packages when doing this.
  - Lets test it out!
    - ✓ Deploy Sample1.java as Sample1.jws
    - ✓ Run Sample1Client.java to access it.

Demystifying AXIS WSDD

- Axis
  - Deploying WebServices
    - ✓ Then place your .java file there .
    - ✓ Then you can access it as you would normally expect.
      - See Sample1WS and Sample1Client.
  - To verify that it exists you can simply type in: http://localhost:8080/axis/Sample1WS.jws in your favorite browser. This will give you a reply:

There is a Web Service here

Click to see the WSDL

Demystifying AXIS WSDD

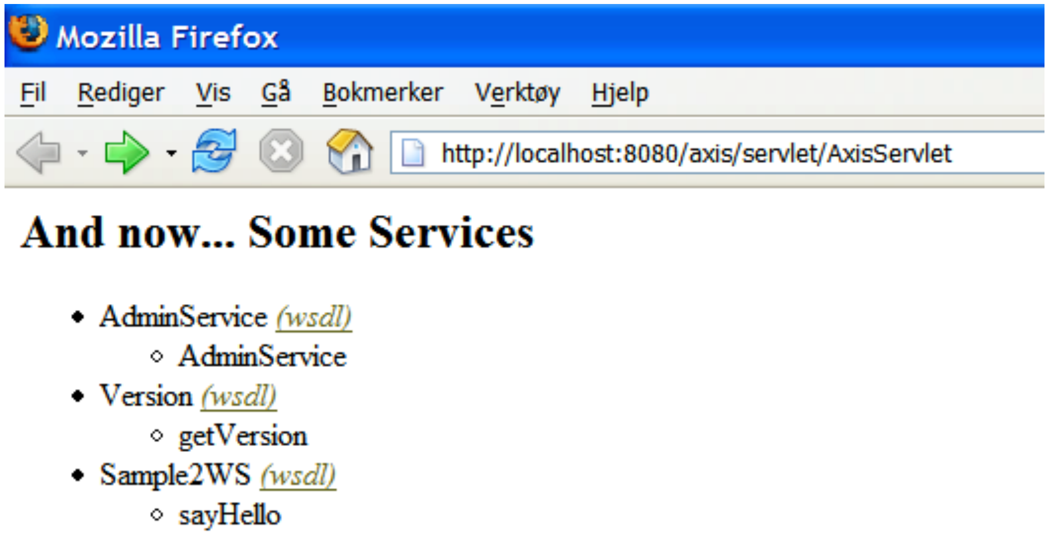
- Axis
  - Deploying WebServices
    - Alt. 2. Through a WSDD file
      - ✓ Package your class(es) in a .jar file and put them in the Axis lib directory.
      - ✓ Define your service in an xml file and name it .wsdd
      - ✓ This gives much greater flexibility and granularity with respect to the deployment process.
      - ✓ Through a .wsdd file you can describe exactly what you want to expose as a WebService and how it can be invoked.

Demystifying AXIS WSDD

- Axis
  - Verifying the deployment
    - To verify that you have (hopefully) succeeded in deploying your WebService, you can access the AxisServlet. You do this by entering for instance http://localhost:8080/axis/servlet/AxisServlet in your browser of choice.
    - This will give you a list of the deployed WebServices including some of Axis's own services.

Demystifying AXIS WSDD

- Axis
  - Verifying the deployment



http://localhost:8080/axis/servlet/AxisServlet

And now... Some Services

- AdminService (wsdl)
  - AdminService
- Version (wsdl)
  - getVersion
- Sample2WS (wsdl)
  - sayHello

Demystifying AXIS WSDD

- Axis
  - Verifying the deployment.
    - Axis stores all the definitions in a central configuration called server-config.wsdd. This file is placed in the WEB-INF directory of your Axis webapp.
  - In the next slides we will take a closer look at the .wsdd file and what goes into it.

Demystifying AXIS WSDD

- WSDD
  - WebService Deployment Descriptor
  - Used to
    - Deploy WebServices on Apache Axis
    - Describes the service
      - ✓ Methods that can be invoked
      - ✓ Input parameters (types)
      - ✓ Return parameters (types)
      - ✓ Handlerchain
      - ✓ Handler

Demystifying AXIS WSDD

- WSDD
  - Is an XML document
  - Outermost (root) element
    - Deployment
    - Has as parameters some namespace definitions.
    - Typically
      - ✓ xmlns="http://xml.apache.org/axis/wsdd/"
      - ✓ xmlns:java="http://xml.apache.org/axis/wsdd/providers/java"

Demystifying AXIS WSDD

- Deployment
  - First child is documentation
    - Use this to document the service(s) that you are deploying
  - Next child is globalConfiguration
    - This element describes global configuration of the Axis Engine.
      - ✓ global request
      - ✓ global response
      - ✓ global fault
      - ✓ global transport flows

Demystifying AXIS WSDD

- Deployment
  - Next child element can be typeMapping
    - describes the mapping between XML and a programming language specific object.
  - or next child element can be chain
    - describes a collection of handlers invoked sequentially as a single unit.
  - or next child element can be handler
    - describes the deployment of an individual handler component.

Demystifying AXIS WSDD

- Deployment
  - or next child element can be transport.
    - describes the request, response, and fault flows for a given transport mechanism.
  - or next child element can be service
    - describes the deployment of a Web Service.
  - Each of these elements have properties and child elements. We will look more closely at each as we touch them later on.

Demystifying AXIS WSDD

- WSDD – Let’s start simple.
  - In this example we will look at deploying a very simple service that just returns a string.
  - The Service:

```
package no.dnbnor.css2005.ws.samples;

/**
 * @author <a href="mailto:hermod.opstvedt@dnbnor.no">Hermod Opstvedt, DnB NOR</a>
 * @version 1.0
 */
public class Sample2WS {


		public String sayHello()
		{
				return "Hello CSS2005";
		}

}
```

Demystifying AXIS WSDD

- To deploy this service we need to create a .wsdd file describing it. sample2.wsdd:

```
<deployment xmlns="http://xml.apache.org/axis/wsdd/"
            xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
 <service name="Sample2WS" provider="java:RPC">
 <parameter name="className" value="no.dnbnor.css2005.ws.samples.Sample2WS"/>
 <parameter name="allowedMethods" value="*"/>
 </service>
</deployment>
```

Demystifying AXIS WSDD

- If we look closer at it, you see that it has a root element:

```
<deployment xmlns="http://xml.apache.org/axis/wsdd/"
  xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
```

- This also declares some namespaces that we will use.
- First child element is the service element:

```
<service name="Sample2WS" provider="java:RPC">
```

  - Name: This uniquely identifies our service

Demystifying AXIS WSDD

- Provider:
  - java:RPC
    - ✓ Synchronous calls
  - java:MSG
    - ✓ Asynchronous calls
- Within the service element we have several child elements:

Demystifying AXIS WSDD

- Parameter:
  - This is used to specify name/value attributes

```
<parameter name="className" value="no.dnbnor.css2005.ws.samples.Sample2WS"/>
 <parameter name="allowedMethods" value="*"/>
```

  - Here we declare a parameter with a name attribute with value "className" which identifies our class through the "value" attribute.
  - Next we declare a parameter with a "name" attribute with value "allowedMethods" with a "value" attribute with value "*" which tells Axis which method(s) can be invoked on our service. In this case all (*).

Demystifying AXIS WSDD

- Operation:

```
<operation name="sayHello" qname="oNS:sayHello" xmlns:oNS="http://samples.ws.css2005.dnbnor.no/"
   returnQName="response" />
```

- Here we specify in detail the operation that is to be invoked:
  - name: the name of the operation
  - qname: XML namespace identifier of the operation
  - xmlns: the XML namespace
  - returnQName: XML namespace identifier of the return

Demystifying AXIS WSDD

- To deploy this WebService on Axis we run the Axis AdminClient. To do this you need to set up a classpath containing all the .jar files that are in the Axis lib directory.
- Then invoke (with Axis running):

```
java org.apache.axis.client.AdminClient <name>.wsdd
```

- This will then deploy our WebService

Demystifying AXIS WSDD

- In our case this will be sample2.wsdd
- Before we can deploy it with the AdminClient we need to package our WebService class file in a .jar file and deploy it in the Axis lib directory. This way Axis will find it during deployment of our .wsdd file.

Demystifying AXIS WSDD

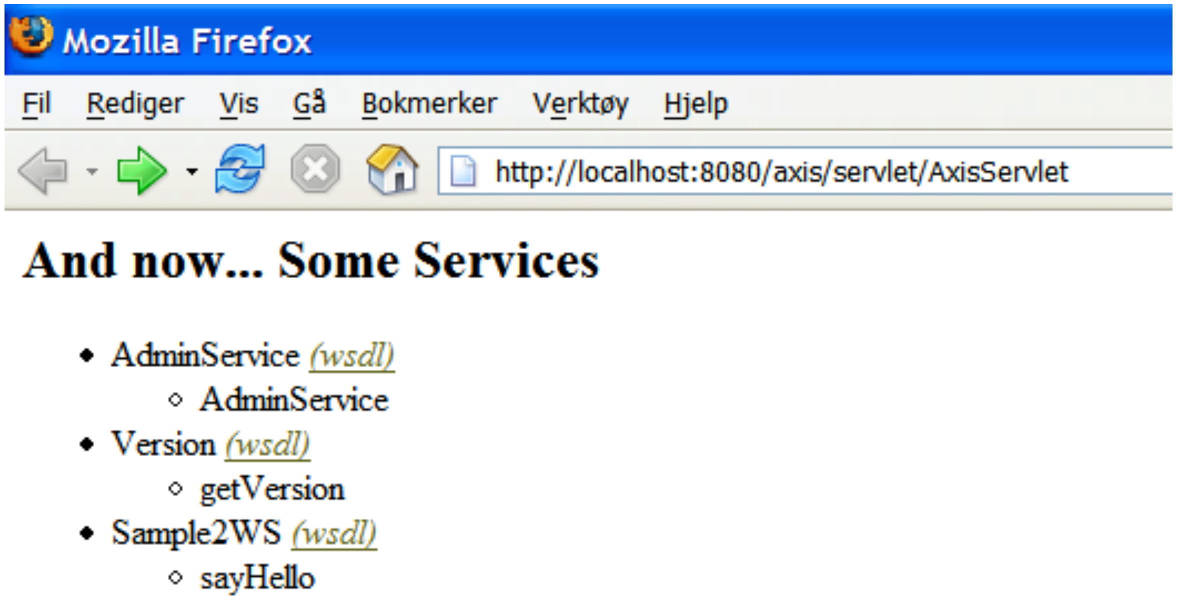
- After invoking the AdminClient it will reply:

```
Processing file c:\CSS2005\WSDD\WSSamples\wsdd\sample2.wsdd
<Admin>Done processing</Admin>
```

- Invoking the AxisServlet will then verify to us that we have been successful.

Demystifying AXIS WSDD

- Output from AxisServlet:

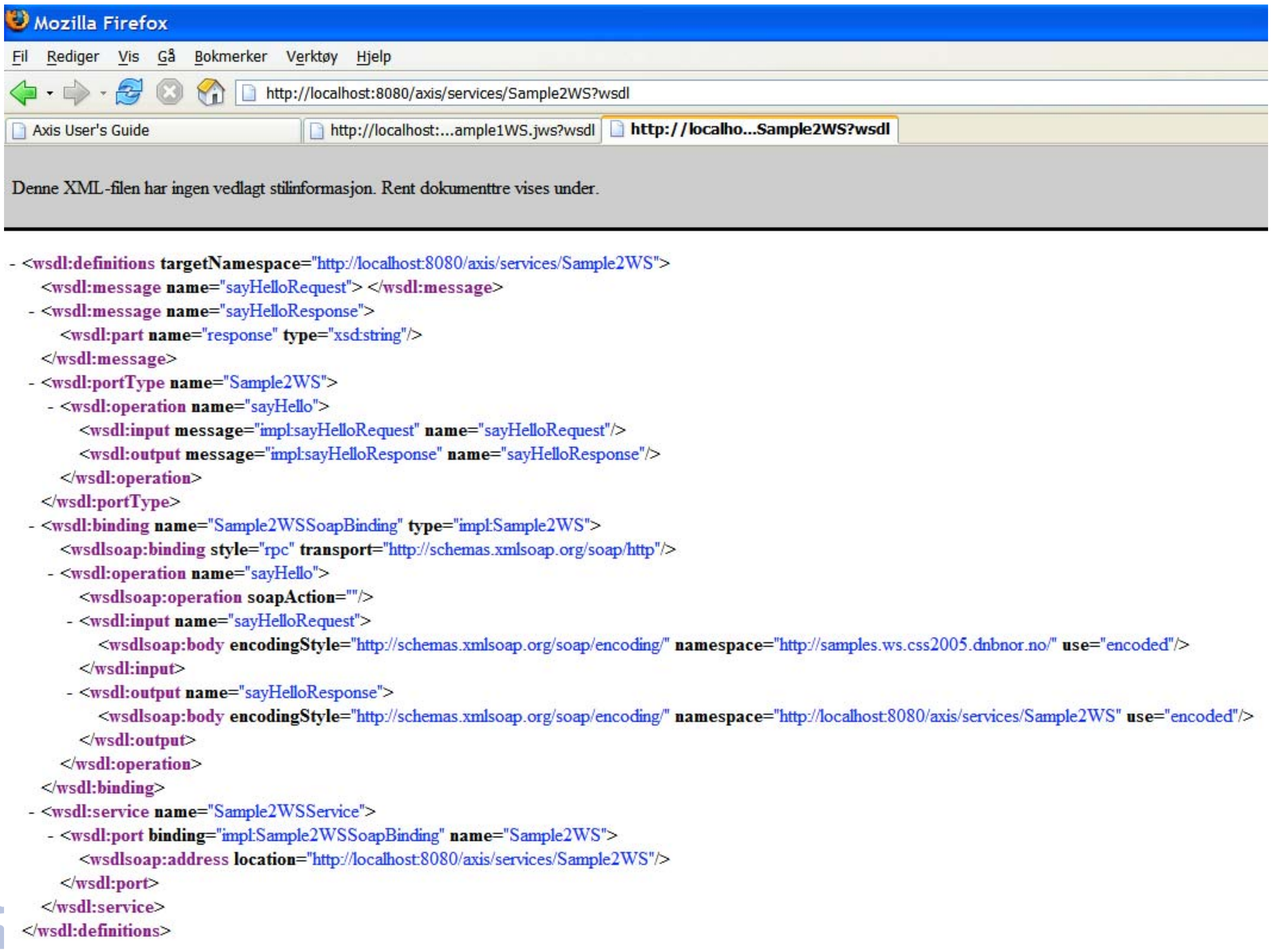

Mozilla Firefox
Fil Rediger Vis Gå Bokmerker Verktøy Hjelp
http://localhost:8080/axis/servlet/AxisServlet

And now... Some Services

- AdminService *(wsdl)*
  - AdminService
- Version *(wsdl)*
  - getVersion
- Sample2WS *(wsdl)*
  - sayHello

- And clicking on the wsdl link of Sample2

Demystifying AXIS WSDD


Mozilla Firefox
Fil Rediger Vis Gå Bokmerker Verktøy Hjelp
http://localhost:8080/axis/services/Sample2WS?wsdl
Axis User's Guide
http://localhost:...ample1WS.jws?wsdl
http://localho...Sample2WS?wsdl

Denne XML-filen har ingen vedlagt stilinformasjon. Rent dokumenttre vises under.

```
- <wsdl:definitions targetNamespace="http://localhost:8080/axis/services/Sample2WS">
    <wsdl:message name="sayHelloRequest"> </wsdl:message>
  - <wsdl:message name="sayHelloResponse">
      <wsdl:part name="response" type="xsd:string"/>
    </wsdl:message>
  - <wsdl:portType name="Sample2WS">
    - <wsdl:operation name="sayHello">
        <wsdl:input message="impl:sayHelloRequest" name="sayHelloRequest"/>
        <wsdl:output message="impl:sayHelloResponse" name="sayHelloResponse"/>
      </wsdl:operation>
    </wsdl:portType>
  - <wsdl:binding name="Sample2WSSoapBinding" type="impl:Sample2WS">
      <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    - <wsdl:operation name="sayHello">
        <wsdlsoap:operation soapAction=""/>
      - <wsdl:input name="sayHelloRequest">
          <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://samples.ws.css2005.dnbnor.no/" use="encoded"/>
        </wsdl:input>
      - <wsdl:output name="sayHelloResponse">
          <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://localhost:8080/axis/services/Sample2WS" use="encoded"/>
        </wsdl:output>
      </wsdl:operation>
    </wsdl:binding>
  - <wsdl:service name="Sample2WSService">
    - <wsdl:port binding="impl:Sample2WSSoapBinding" name="Sample2WS">
        <wsdlsoap:address location="http://localhost:8080/axis/services/Sample2WS"/>
      </wsdl:port>
    </wsdl:service>
  </wsdl:definitions>
```

Demystifying AXIS WSDD

- You can also verify it by looking into the server-config.wsdd file.
- So now we have successfully deployed our first WebService, although very simple, and verified that it is there.
- Next up is actually trying to call it.
- To do that we need a WebService client.

Demystifying AXIS WSDD

```
package no.dnbnor.css2005.wsdd.samples;
import java.net.URL;
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import org.apache.axis.encoding.XMLType;
public class Sample2Client {
        // Set the endpoint adress
        private static final String endpoint = "http://localhost:8080/axis/services/Sample2WS";
        public static void main(String[] args) throws Exception{
                // Create the service
                Service service = new Service();
                // Create the call
                Call call = (Call) service.createCall();
                // set the endpoint
                call.setTargetEndpointAddress(new URL(endpoint));
```

Demystifying AXIS WSDD

```
                    // set the method (operation)
                    call.setOperationName("sayHello");
                    // set the return type (Use standard WS-I type)
                    call.setReturnType(XMLType.XSD_STRING);
                    String retval = (String) call.invoke(new Object[] {});
                    System.out.println("Return value: " + retval);
          }
}
```

- So let's try it – Start up Axis, and set up the same classpath as for the AdminClient, and invoke the client:

```
java no.dnbnor.css2005.wsdd.samples.Sample2Client
Return value: Hello CSS2005
```

Demystifying AXIS WSDD

- Next step – Add an input parameter to a new service
  - Now we create a new service with a method "greetMe" which will take a String input parameter. So we add a parameter element to the operation element of the wsdd file.

```
 <operation name="greetMe" qname="oNS:greetMe"
xmlns:oNS="http://samples.ws.css2005.dnbnor.no/" returnQName="returnval" >

      <parameter name="name" type="tns:string" xmlns:tns="http://www.w3.org/2001/XMLSchema"/>

  </operation>
```

Demystifying AXIS WSDD

- parameter:
  - name: the name of the parameter, in this case name (our name).
  - type: the type of the input parameter, in this case tns:string
  - xmlns:tns: the namespace definition

Demystifying AXIS WSDD

- Sample3WS:

```
public String greetMe(String name)
{
        return "Greetings: " + name;
}
```

- Sample3Client

```
// Create the service
Service service = new Service();
// Create the call
Call call = (Call) service.createCall();
// set the endpoint
call.setTargetEndpointAddress(new URL(endpoint));
// set the method (operation)
call.setOperationName("greetMe");
// set the input parameter (Use standard WS-I type)
call.addParameter("name", XMLType.XSD_STRING, javax.xml.rpc.ParameterMode.IN);
// set the return type (Use standard WS-I type)
call.setReturnType(XMLType.XSD_STRING);
String retval = (String) call.invoke(new Object[] { "Hermod" });
System.out.println("Return value: " + retval);
```

Demystifying AXIS WSDD

- Package jar file, export and deploy it, verify deployment and test it!
- Test for sample3 yields:

```
Return value: Greetings: CSS2005
```

- Let’s move one step further and add a second parameter which is of type integer

```
<operation name="greetMe" qname="oNS:greetMe"
  xmlns:oNS="http://samples.ws.css2005.dnbnor.no/" returnQName="returnval" >
    <parameter name="name" type="tns:string"
  xmlns:tns="http://www.w3.org/2001/XMLSchema"/>
    <parameter name="year" type="tns:int" xmlns:tns="http://www.w3.org/2001/XMLSchema"/>
</operation>
```

Demystifying AXIS WSDD

- The new service:

```
public String greetMe(String name, Integer year)
{
    return "Greetings: " + name + " for " + year.toString();
}
```

- And in the client we add:

```
call.addParameter("year", XMLType.XSD_INTEGER, javax.xml.rpc.ParameterMode.IN);
```

- After our previous parameter definition

Demystifying AXIS WSDD

- Package jar file, export and deploy it, verify deployment and test it!
- Test for sample4 yields:

```
Return value: Greetings: CSS for 2005
```

- So far so good. But hold on – Axis is supposed to be SMART with respect to deployment. Is all that we have done necessary?

Demystifying AXIS WSDD

- No, not really. We could have written the deployment descriptor without specifying the operation with input and output parameters. Let's try it.
- In sample4a we simply use:

```
<service name="Sample4WS" provider="java:RPC">
 <parameter name="className" value="no.dnbnor.css2005.ws.samples.Sample4WS"/>
 <parameter name="allowedMethods" value="*"/>
</service>
```

- Let's deploy sample4 again using this new wsdd file, and test it again. Also let's look for any difference in the wsdl.

Demystifying AXIS WSDD

- There is no difference. So why bother?
  - Greater control. Especially when things get complicated, and also: Axis is only so smart.
- Let's try something new. This time we will stick with letting Axis do the work, but instead of returning something simple we will define a class of our own.

Demystifying AXIS WSDD

- Sample 5.
  - In this sample we will return a person class that has three attributes:
    - Name – String
    - Sex – String
    - Age – Integer

Demystifying AXIS WSDD

- So now our service looks like this:

```
public class Sample5WS {

        public Person getPerson()
        {
           // Create a new Person and return it
           return new Person("John Doe", "Male", new Integer(30));
        }

}
```

Demystifying AXIS WSDD

- Package, export and deploy it.
- Look at the wsdl:
  - Notice that Axis has added our Person entity :

```
<wsdl:types>
    <schema targetNamespace="http://samples.ws.css2005.dnbnor.no"
    xmlns="http://www.w3.org/2001/XMLSchema">
    <import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
       <complexType name="Person">
               <sequence>
                  <element name="age" nillable="true" type="xsd:int" />
                  <element name="name" nillable="true‟ type="xsd:string" />
                  <element name="sex" nillable="true‟ type="xsd:string" />
               </sequence>
       </complexType>
    </schema>
</wsdl:types>
```

Demystifying AXIS WSDD

- Let’s test it – Sample5Client.java

```
.....
 // Create the service
        Service service = new Service();
        // Create the call
        Call call = (Call) service.createCall();
        // set the endpoint
        call.setTargetEndpointAddress(new URL(endpoint));
        // set the method (operation)
        call.setOperationName("getPerson");
        Person retval = (Person) call.invoke(new Object[] {});
        System.out.println("Return value:\nPersons name: " + retval.getName()+"\nPersons sex: " +
retval.getSex()+"\nPersons age: "+retval.getAge());
......
```

Demystifying AXIS WSDD

- What ?

Demystifying AXIS WSDD

- Let's look in the log:

```
AxisFault
 faultCode: {http://schemas.xmlsoap.org/soap/envelope/}Server.userException
 faultSubcode:
 faultString: java.io.IOException: No serializer found for class
    no.dnbnor.css2005.ws.samples.Person in registry
    org.apache.axis.encoding.TypeMappingImpl@1148603
 faultActor:
 faultNode:
 faultDetail:
    {http://xml.apache.org/axis/}stackTrace: java.io.IOException: No serializer found for
    class no.dnbnor.css2005.ws.samples.Person in registry
```

Demystifying AXIS WSDD

- No serializer; not so smart after all then.
- Let’s define our person in the wsdd using typeMapping – Sample5a.wsdd.

```
…
 <typeMapping
   xmlns:ns="http://samples.ws.cs2005.dnbnor.no"
   qname="ns:Person"
   languageSpecificType="java:no.dnbnor.css2005.ws.samples.Person"
   serializer="org.apache.axis.encoding.ser.BeanSerializerFactory"
   deserializer="org.apache.axis.encoding.ser.BeanDeserializerFactory"
   encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  />
…
```

Demystifying AXIS WSDD

- typeMapping
  - xmlns:ns: The namespace, normally reflects package structure of our java class.
  - qname: The xml identifier within the namespace
  - languageSpecificType: The fully qualified name of our java class.
  - serializer: The serializer used to convert our object into an xml representation. You can write your own implementation or use the Axis one.

Demystifying AXIS WSDD

- deserializer: The deserializer used to convert an xml representation into a java object.
- encodingStyle: How the object should be encoded in xml.

Demystifying AXIS WSDD

- Deploy, verify and test again.
- This time no error from server, but the client…..?
  - org.xml.sax.SAXException: Deserializing parameter 'getPersonReturn': could not find deserializer for type {http://samples.ws.cs2005.dnbnor.no}Person

Demystifying AXIS WSDD

- We need to to tell the Axis runtime how to deserialize our bean in our client:

```
 …
call.registerTypeMapping(Person.class, new QName("http://samples.ws.css2005.dnbnor.no",
   "Person"),   BeanSerializerFactory.class, BeanDeserializerFactory.class);
…
```

- Lets rerun the client Sample5a.java

```
Return value:
Persons name: John Doe
Persons sex: Male
Persons age: 30
```

- Success!

Demystifying AXIS WSDD

- Sample 6.
- Lets go one step further:
  - Add a new type to Person: Address
    - Streetaddress
    - Zipcode
    - Nation
    - AdressType (Home, Office, *etc*.)

Demystifying AXIS WSDD

- Create a new Person2 that extends Person
- Add Array of Adress to it.
- Create Sample6WS
- Package, export, deploy and verify.
  - Note that Axis has added a typeMapping for the array of address on its own:

```
<complexType name="ArrayOf_tns1_Adress">
    <complexContent>
    <restriction base="soapenc:Array">
<attribute ref="soapenc:arrayType" wsdl:arrayType="tns1:Address[]"/>
</restriction>
</complexContent>
</complexType>
```

Demystifying AXIS WSDD

- Let’s test it with Sample6Client – Using the new Person2 and the new Address

```
…
Person2 retval = (Person2) call.invoke(new Object[] {});
System.out.println("Return value:\nPersons name: " + retval.getName() + "\nPersons sex: " + retval.getSex()
      + "\nPersons age: " + retval.getAge());
System.out.println("Addresses:");
for(int i=0; i< retval.getAdresses().length;i++)
{
   System.out.println("Address type: " + retval.getAdresses()[i].getAddressType());
   System.out.println("Strretaddress: " + retval.getAdresses()[i].getStreetAdress());
   System.out.println("Zipcode: " + retval.getAdresses()[i].getZipCode());
   System.out.println("Nation: " + retval.getAdresses()[i].getNation());
}
```

Demystifying AXIS WSDD

- Hey, what happened!?
  - When things start to get a little more complicated we need to augment the client side a bit.
  - In this case the client could not deserialize the response. So we need to add some more information to the client runtime.
  - We need to register the typeMapping with the client.

Demystifying AXIS WSDD

- Sample 6a.
  - Here we register the typeMappings before we do the call:

```
…
call.registerTypeMapping(Person2.class, new QName("http://samples.ws.css2005.dnbnor.no", "Person2"),
            BeanSerializerFactory.class, BeanDeserializerFactory.class);

call.registerTypeMapping(Person.class, new QName("http://samples.ws.css2005.dnbnor.no", "Person"),
            BeanSerializerFactory.class, BeanDeserializerFactory.class);

call.registerTypeMapping(Address.class, new QName("http://samples.ws.css2005.dnbnor.no", "Address"),
            BeanSerializerFactory.class, BeanDeserializerFactory.class);
..
```

Demystifying AXIS WSDD

- Let’s test it:

```
Return value:
Persons name: John Doe
Persons sex: Male
Persons age: 30
Addresses:
Address type: Home
Streetaddress: Elm street 24
Zipcode: 12345
Nation: USA
Address type: Office
Strretaddress: Wall Street 1098
Zipcode: 11111
Nation: USA
```

- So now it worked ok.

Demystifying AXIS WSDD

- allowedMethods
  - Let's take a look at the allowedMethods part of the service element.
    - Add in a couple of more methods, and specify exactly which of these should be available to the public.
  - Sample 7.
    - Add method sayHello and myPrivateMethod
    - Specify only getPerson2 and sayHello in the allowedMethods attribute of the .wsdd file.

Demystifying AXIS WSDD

```
<service name="Sample7WS" provider="java:RPC">
  <parameter name="className" value="no.dnbnor.css2005.ws.samples.Sample7WS"/>
  <parameter name="allowedMethods" value="getPerson sayHello"/>
…
```

- Package, export and deploy it
- Let’s test it – Sample7Client

```
   ….
   // This method should work ok
   call.setOperationName("sayHello");
   String retval2=(String) call.invoke(new Object[]{new String("CSS2005")});
   System.out.println("sayHello returned: " + retval2);
   // This method should not work
   call.setOperationName("myPrivateMethod");
   String retval3=(String) call.invoke(new Object[]{new String("CSS2005")});
   System.out.println("myPrivateMethod returned: " + retval2);
```

Demystifying AXIS WSDD

- And it worked as expected
  - getPerson executed ok
  - sayHello executed ok
  - myPrivateMethod threw an exception:

```
AxisFault
 faultCode: {http://schemas.xmlsoap.org/soap/envelope/}Server.generalException
 faultSubcode:
 faultString: No such operation &apos;myPrivateMethod&apos;
 faultActor:
 faultNode:
 faultDetail:
{http://xml.apache.org/axis/}stackTrace: AxisFault
 faultCode: {http://schemas.xmlsoap.org/soap/envelope/}Server.generalException
 faultSubcode:
 faultString: No such operation &amp;apos;myPrivateMethod&amp;apos;
 faultActor:
 faultNode:
 faultDetail:
```

Demystifying AXIS WSDD

- loadOnStartup

```
<parameter name="loadOnStartup" value="true"/>
```

  - This is just like the load on startup parameter that is found in the web.xml file for making sure that servlets are loaded when the container starts.

Demystifying AXIS WSDD

- scope

```
<parameter name="scope" value="request"/>
```

- This tells how the client and service interact, just like with normal servlet conversation
  - request – all calls are request based
  - session – an new session is created at first call, and a cookie is returned along with the response. The client has to be aware of this and send the cookie back on the next call.
  - application – all interaction information is application wide, meaning shared by all callers.

Demystifying AXIS WSDD

- requestFlow & responseFlow, handler & handlerChain

```
<requestFlow>
      <handler type="java:MyHandler"/> <handler type="no.dnbnor.css2005.ws.samples.MyHandler">
                <parameter name="aname" value="avalue"/>
      </handler>
</requestFlow>
<responseFlow>
      <handler type="java:MyHandler"/> <handler type="no.dnbnor.css2005.ws.samples.MyHandler"/>
</responseFlow>
```

  - requestFlow & responseFlow may be thought of as filters in a regular servlet environment.
  - Handlers are services that get invoked from the Axis runtime.

Demystifying AXIS WSDD

- The Service Handler
  - Implements javax.xml.rpc.handler.Handler
    - public boolean handleRequest(MessageContext arg);
    - public boolean handleResponse(MessageContext arg);
    - public boolean handleFault(MessageContext arg);
    - public void init(HandlerInfo arg);
    - public void destroy();
    - public QName[] getHeaders()

Demystifying AXIS WSDD

- Handlers defined in the requestFlow get called prior to invoking the method on the webservice
- Handlers defined in the responseFlow get called after the method call to the webservice, but prior to returning the response to the client.
- Handlers can have <parameter> child elements.

Demystifying AXIS WSDD

- handlerChains

```
….
<requestFlow>
      <handlerChain>
                <handler type="java:MyHandler"/> <handler type="no.dnbnor.css2005.ws.samples.MyHandler"/>
                <handler type="java:OtherHandler"/> <handler type="no.dnbnor.css2005.ws.samples.OtherHandler"/>
      </handlerChain>
</requestFlow>
<responseFlow>
      <handlerChain>
                <handler type=" java:MyHandler "/> <handler type="no.dnbnor.css2005.ws.samples.MyHandler "/>
                <handler type=" java:OtherHandler "/> <handler type="no.dnbnor.css2005.ws.samples.OtherHandler "/>
      </handlerChain>
</responseFlow>
```

  - This is used to group a sequence of handlers together

Demystifying AXIS WSDD

- Handlers are used for things like authentication, encryption & signing, *etc*.

Demystifying AXIS WSDD

- Standard mappings from WSDL to Java

WSDL Type	Java type
xsd:base64Binary	byte[]
xsd:boolean	boolean
xsd:byte	byte
xsd:dateTime	java.util.Calendar
xsd:decimal	java.math.BigDecimal
xsd:double	double
xsd:float	float
xsd:hexBinary	byte[]
xsd:int	int
xsd:integer	java.math.BigInteger
xsd:long	long
xsd:QName	javax.xml.namespace.QName
xsd:short	short
xsd:string	java.lang.String

Demystifying AXIS WSDD

- Interoperability
  - Be careful about what types you declare
  - Base Java types along with their java.lang types are considered to be safe to use, mapping them as shown in the previous table.
  - Also arrays of these
  - User defined classes are also safe as long as their attributes are as above.

Demystifying AXIS WSDD

- WSDDHelper
  - Generates .wsdd files given a service
  - Must have available in classpath all referenced classes.
  - A demonstration.

Demystifying AXIS WSDD

- References
  - http://ws.apache.org/axis/
  - http://www.w3.org/TR/wsdl.html#_Toc492291084
  - http://www.ws-i.org/

Demystifying AXIS WSDD

- Questions