Soap Webservices

Agenda Day 1

Introduction to Webservices

WebServices terminology

Writing a Java WebService client – Stub generation & calling the service

Setting up eclipse / Java EE 7 SDK

Coding and deploying a webservice

Adding input arguments

Service First and Contract First Webservices

Understanding and customizing the WSDL

Hands on Lab

Motivation

- The ability to program the Web.

Web Service definition

A simple definition:

> “a Web Service is an application component accessible over open protocols”.

History

- Web services evolved from previous technologies that served the same purpose such as RPC, ORPC (DCOM, CORBA and JAVA RMI).
- Web Services were intended to solve three main problems:
  1. Interoperability
  2. Firewall traversal
  3. Complexity

Interoperability

- Earlier distributed systems suffered from interoperability issues because each vendor implemented its own on-wire format for distributed object messaging.
- Development of DCOM apps strictly bound to Windows Operating system.
- Development of RMI bound to Java programming language.

Firewall traversal

- Collaboration across corporations was an issue because distributed systems such as CORBA and DCOM used non-standard ports.
- Web Services use HTTP as a transport protocol and most of the firewalls allow access though port 80 (HTTP), leading to easier and dynamic collaboration.

Complexity

- Web Services is a developer-friendly service system.
- Most of the above-mentioned technologies such as RMI, COM, and CORBA involve a whole learning curve.
- New technologies and languages have to be learnt to implement these services.

Web Service definition revisited

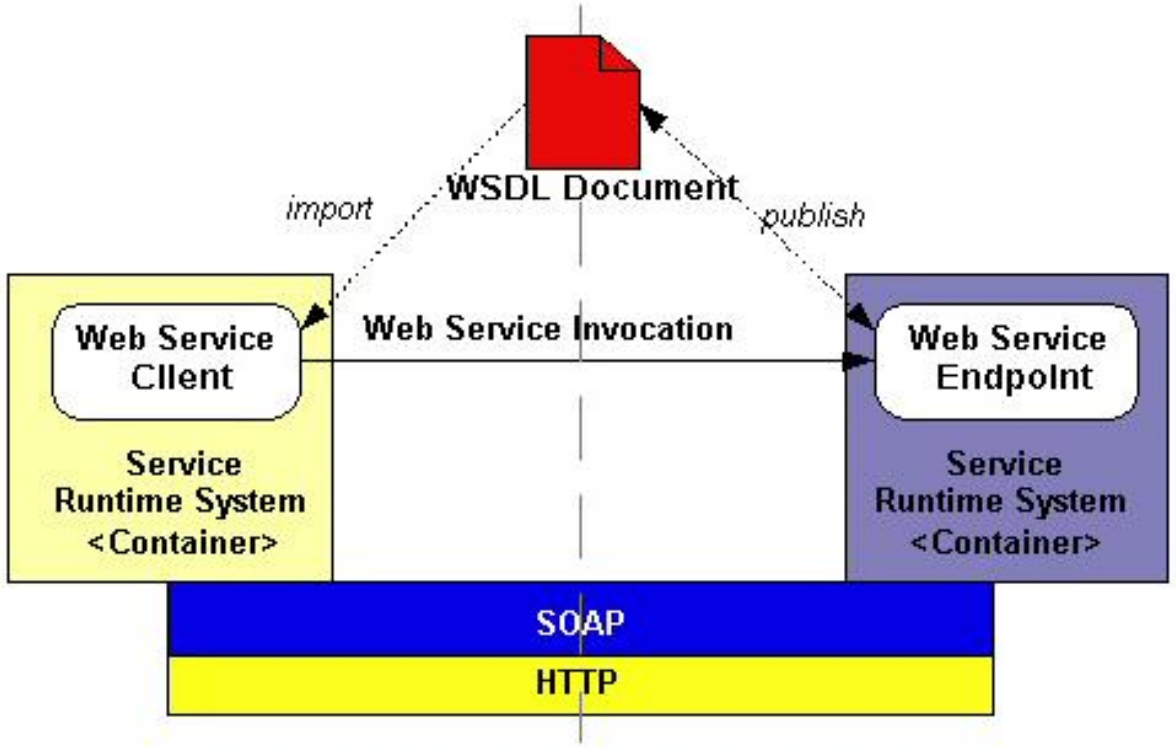
- A more precise definition:
  - an application component that:
    - Communicates via open protocols (HTTP, SMTP, etc.)
    - Processes XML messages framed using SOAP
    - Describes its messages using XML Schema
    - Provides an endpoint description using WSDL
    - Can be discovered using UDDI

Web Services Components

- **XML** – eXtensible Markup Language – A uniform data representation and exchange mechanism.
- **SOAP** – Simple Object Access Protocol – A standard way for communication.
- **UDDI** – Universal Description, Discovery and Integration specification – A mechanism to register and locate WS based application.
- **WSDL** – Web Services Description Language – A standard meta language to described the services offered.

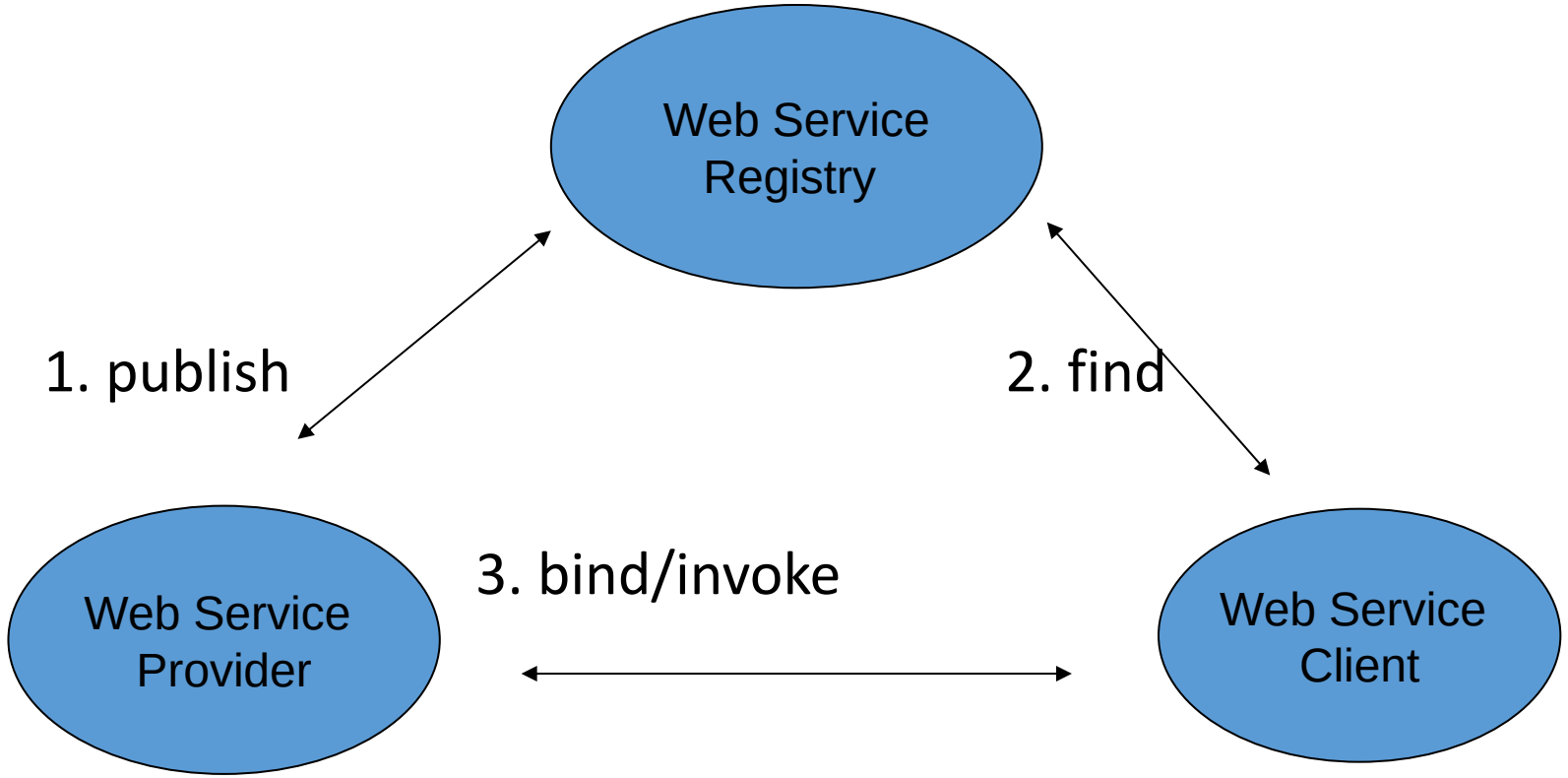
The Web Service Model

- The Web Services architecture is based upon the interactions between three roles:
  - Service provider
  - Service registry
  - Service requestor
- The interactions involve the:
  - Publish operations
  - Find operation
  - Bind operations.



The Web Service Model (cont)

The Web Services model follows the *publish*, *find*, and *bind* paradigm.



XML

- XML stands for E**X**tensible **M**arkup **L**anguage.
- XML is a **markup language** much like HTML.
- XML was designed to **describe data**.
- XML tags are not predefined. You must **define your own tags.**
- The prefect choice for enabling cross-platform data communication in Web Services.

XML vs HTML

An HTML example:

```
<html>
<body>
    <h2>John Doe</h2>
    <p>2 Backroads Lane<br>
        New York<br>
        045935435<br>
        john.doe@gmail.com<br>
        </p>
</body>
</html>
```

XML vs HTML

- This will be displayed as:

- HTML specifies how the document is to be displayed, and not what information is contained in the document.
- Hard for machine to extract the embedded information. Relatively easy for human.

XML vs HTML

- Now look at the following:

```
<?xml version=1.0?>
<contact>
  <name>John Doe</name>
  <address>2 Backroads Lane</address>
  <country>New York</country>
  <phone>045935435</phone>
  <email>john.doe@gmail.com</email>
</contact>
```

- In this case:
  - The information contained is being marked, but not for displaying.
  - Readable by both human and machines.

SOAP

- SOAP originally stood for "Simple Object Access Protocol" .
- Web Services expose useful functionality to Web users through a standard Web protocol called SOAP.
- Soap is an XML vocabulary standard to enable programs on separate computers to interact across any network. SOAP is a simple markup language for describing messages between applications.
- Soap uses mainly HTTP as a transport protocol. That is, HTTP message contains a SOAP message as its payload section.

SOAP Characteristics

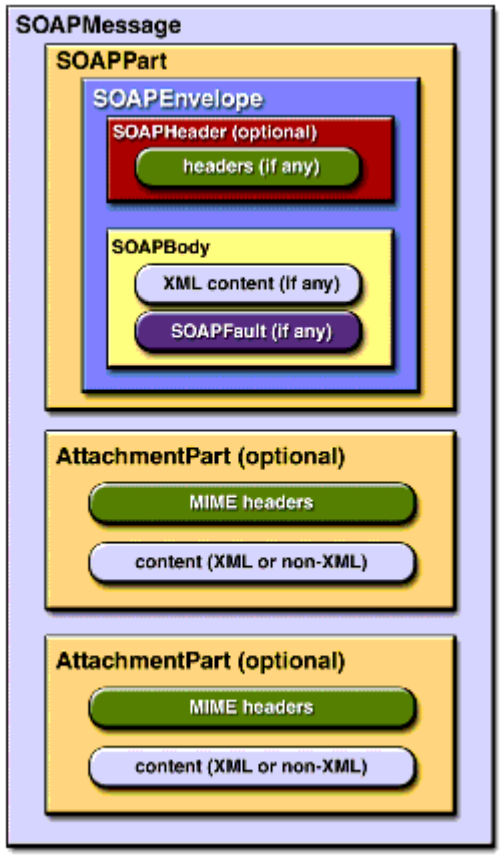
- SOAP has three major characteristics:
  - Extensibility – security and WS-routing are among the extensions under development.
  - Neutrality - SOAP can be used over any transport protocol such as HTTP, SMTP or even TCP.
  - Independent - SOAP allows for any programming model .

SOAP Building Blocks

A SOAP message is an ordinary XML document containing the following elements:

- A required Envelope element that identifies the XML document as a SOAP message.
- An optional Header element that contains header information.
- A required Body element that contains call and response information.
- An optional Fault element that provides information about errors that occurred while processing the message.

Soap Message



SOAP Request

```
POST /InStock HTTP/1.1
Host: www.stock.org
Content-Type: application/soap+xml; charset=utf-8 Content-Length: 150

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle=http://www.w3.org/2001/12/soap-encoding”>

    <soap:Body xmlns:m="http://www.stock.org/stock">
            <m:GetStockPrice>
                <m:StockName>IBM</m:StockName>
            </m:GetStockPrice>
    </soap:Body>
</soap:Envelope>
```

SOAP Response

```
HTTP/1.1 200 OK
Content-Type: application/soap; charset=utf-8
Content-Length: 126

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

    <soap:Body xmlns:m="http://www.stock.org/stock">
        <m:GetStockPriceResponse>
            <m:Price>34.5</m:Price>
        </m:GetStockPriceResponse>
    </soap:Body>
</soap:Envelope>
```

WSDL

- WSDL stands for Web Services Description Language.
- WSDL is an XML vocabulary for describing Web services. It allows developers to describe Web Services and their capabilities, in a standard manner.
- WSDL specifies what a request message must contain and what the response message will look like in unambiguous notation. In other words, it is a contract between the XML Web service and the client who wishes to use this service.
- In addition to describing message contents, WSDL defines where the service is available and what communications protocol is used to talk to the service.

The WSDL Document Structure

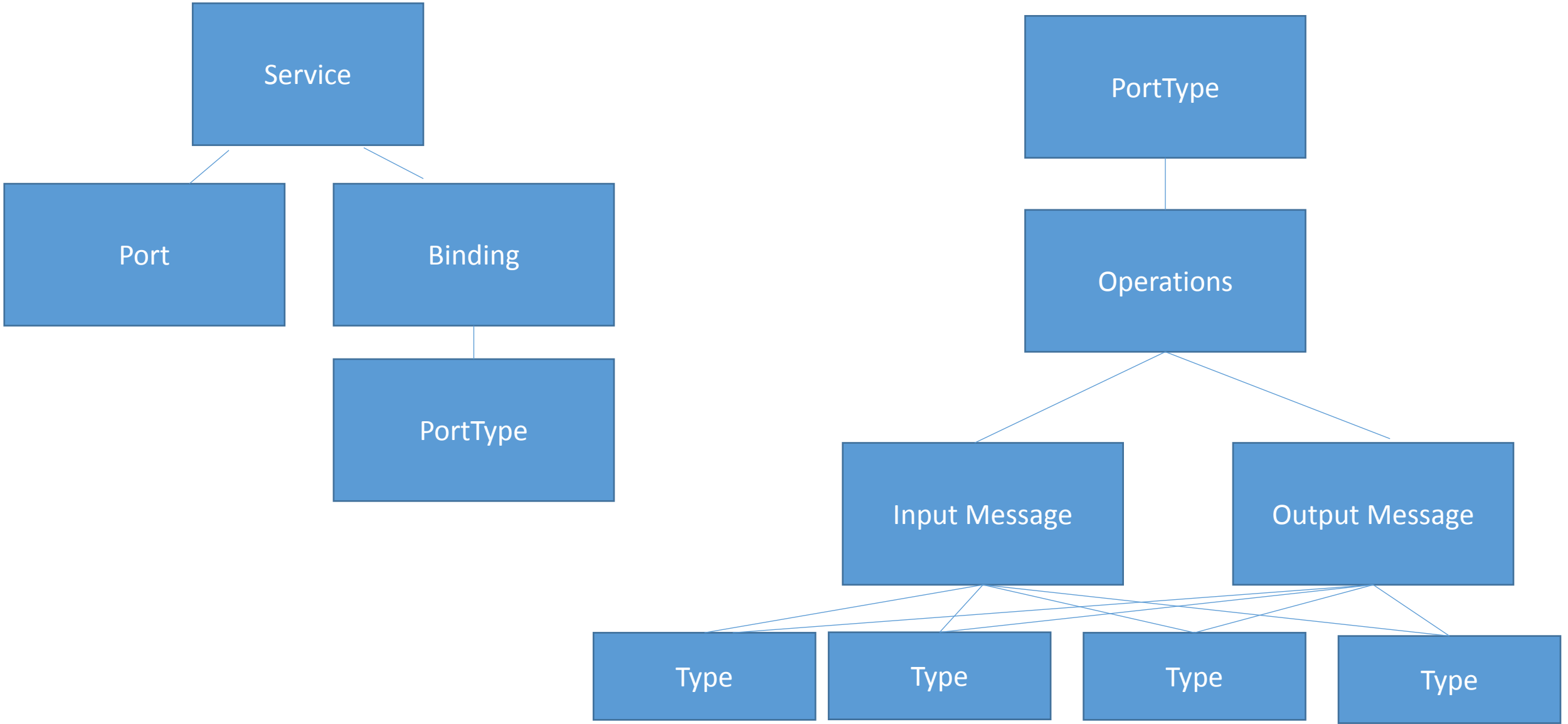
- A WSDL document is just a simple XML document.
- It defines a web service using these major elements:
  - **port type** - The operations performed by the web service.
  - **message -** The messages used by the web service.
  - **types -** The data types used by the web service.
  - **binding -** The communication protocols used by the web service.

WSDL Document

```
<message name="GetStockPriceRequest">
    <part name="stock" type="xs:string"/>
</message>
<message name="GetStockPriceResponse">
    <part name="value" type="xs:string"/>
</message>

<portType name="StocksRates">
    <operation name="GetStockPrice">
        <input message="GetStockPriceRequest"/>
        <output message="GetStockPriceResponse"/>
    </operation>
</portType>
```

Understanding WSDL



UDDI

- UDDI stands for Universal Description, Discovery and Integration.
- UDDI is a directory for storing information about web services , like yellow pages.
- UDDI is a directory of web service interfaces described by WSDL.

Environment Set up contd.

- Verify JDK Installation
- Go to command prompt and type java -version
- verify java is in the Path, ensure Java is 1.7 or more.
- Verify Java_HOME is set
- echo %JAVA_HOME%
- The path to bin folder of the Java inst dir will be displayed

Create a java first webservice

- Create a Java class and annotate with @WebService.(Part of javax.jws.*)
- Create the methods that will be invoked and annotate with @WebMethod. Annotating a method is optional if class is annotated.
- Publish and start the JEE server
- Using the WSDL test the service using SoapUI

How to create a webservice and access from a Java client

- **Code the implementation class.**
- **Compile the implementation class.**
- **Package the files into a WAR file.**
- **Deploy the WAR file.**
- **Code the client class.**
- **Use a wsimport task (comes with jdk) to generate and compile the web service artifacts needed to connect to the service.**

wsimport –keep <link to wsdl>

- **Compile the client class.**
- **Run the client.**

Contract first webservice

- Design the contract and create the xsd file
- From xsd file, generate java bindings through jaxb’s xjc tool

Command : xjc abc.xsd

- Implement the endpoint
- Test the endpoint

Webservice approaches

- 2 approach
- ContractFirst – lock the wsdl and create the java classes
- ServiceFirst – create java classes and generate wsdl

Customizing a WSDL – through annotations

- @WebService
  - name : name of the portType, default is class name. override by providing a name, changes in class name does not lead to change in wsdl
  - Service : alters the path to the wsdl
  - Default is protocol://hostname:port/appname/servicename?wsdl
  - targetNamespace : override the default which is reversed package name
- @WebMethod
  - exclude - excludes from being exposed as a webservice
  - action - becomes soapaction attribute
  - operationName - methodName --> operation element

Customizing a WSDL contd.

- @SOAPBinding
  - Binding has an impact on the type element.
  - use the style attribute : document,rpc
  - when the style is changed to be rpc, the wsdl tht emerges does not have a type section, instead there is a message section that has the types within.
  - Message has the part inside does not refer to anything external. This may seem more cleaner, but we prefer document so that the types schema can be validated.

- @WebParam(name="lookupInput")
  - inputargument name arg0 can be overriden by using the argument annotation
- @WebResult(name="lookupOutput")
  - The result return type can be annotated as a method annotation

SEI

- An alternate way of marking a webservice is to place the annotations on an interface instead of a class
- The class tht implements this interface will hv to be annotated with @WebService(EndPointInterface=“packagename.interfacename")
- Benefits - impl class can hv additional methods that need not be exposed and the interface on the java side freezes the contract

Marshalling with JAXB

The JAXB API is the standard solution provided by the JDK for Java XML data binding:

- Java classes are bound to XML types, elements, and attributes through Java annotations.

JAXB: Annotations

Although JAXB can bind almost any Java data object with little or no annotations, annotations are typically desirable, for example:

- They can tell JAXB whether to unmarshal a field into an attribute or an element.
- They can inform JAXB of ID fields, element order, and other schema constraints.
- They can be used to identify or customize schema types, element names, attribute names, element wrapping, etc.

JAXB: Common Annotations

JAXB defines *many* annotations to customize Java XML data binding. Here are just a few:

- @XmlRootElement
- @XmlElement
- @XmlAttribute
- @XmlElementWrapper
- @XmlTransient

JAXB: Rules and Conventions

Some general rules about JAXB annotations:

- Concrete classes must have a public default no-arg constructor.
- Properties that reference interfaces must be annotated with one or more **@XmlElementRef** annotations that identify the possible concrete types.
- Annotations may be placed on the **fields** or on the **setters** but not on both.
- By convention, annotating fields is preferable for simple POJOs.
- Properties not bound to XML values must be annotated with **@XmlTransient**.

Conclusion

- The standard Java APIs can be used to model your data for use by web services.
- The JDK, JAX WS API, and the marshallers with the Java Stack provide code generation and configuration utilities to make it easier to consume third-party web services.

Lab