UNIT V

Web Services**:** JAX-RPC-Concepts-Writing a Java Web Service-Writing a Java Web Service Client-Describing Web Services: WSDL- Representing Data Types: XML Schema-communicating Object Data: SOAP Related Technologies-Software Installation-Storing Java Objects as Files-Databases and Java Servlets

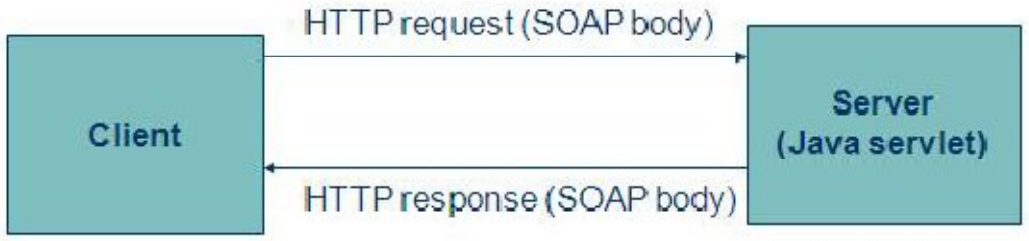
Web Application vs Web Services

- A web application uses Web technologies to provide functionality to an end user
- A web service uses Web technologies to provide functionality to another software application

Web Services

Web services are open standard based Web applications that interact with other web applications for the purpose of exchanging data.

XML is used to encode all communications to a web service. For example, a client invokes a web service by sending an XML message, and then waits for a corresponding XML response. Because all communication is in XML, web services are platform neutral and language neutral. Java can talk with Perl; Windows applications can talk with UNIX applications. Web services conceptually are just specialized web applications:



Body of web services request and response will be SOAP message which defines a protocol for message exchange between applications.

Popular **examples** for Web Services

1. Currency Converter
2. Temperature conversion
3. Weather Forecast system
4. Credit card validation system

Standard web services technologies:

- SOAP
- WDSL
- XML Schema

Higher-level API's such as JAX-RPC and others are often used to automatically generate web services client and server communication software. Microsoft .NET framework is one popular alternative to JAX-RPC

JAX-RPC

It stands for Java API for XML-based RPC. JAX-RPC is a technology for building web services and clients that use *remote procedure calls* (RPC) and XML. Often used in a distributed client-server model, an RPC mechanism enables clients to execute procedures on other systems.

With JAX-RPC, clients and web services have a big advantage: the platform independence of the Java programming language. In addition, JAX-RPC is not restrictive: a JAX-RPC client can access a web service that is not running on the Java platform, and vice versa. This flexibility is possible because JAX-RPC uses technologies defined by the World Wide Web Consortium (W3C) such as:

- SOAP defines message format
- WSDL describes the various services offered by web service application as xml file
- XML Schema – defines data types used in WSDL doc.

Writing Web Service using JAX-RPC (Currency Converter)

These are the basic steps for creating the web service and client:

1. Create an interface
2. Code the implementation class.
3. Use wscompile to generate wsdl doc.
4. Create deployment descriptor file
5. Package the files into a WAR file.
6. Deploy the WAR file.
7. Create a client file to access the web service
8. Use wscompile to generate and compile the web service artifacts needed to connect to the service.
9. Compile the client class.
10. Run the client.

Example for Creating Web service software

To write web service using JAX-RPC, we need JWSDP(Java Web Server Development Package). This package has wscompile and wsdeploy tools. wscompile is used to convert java file into wsdl file whereas wsdeploy is used to package our web service.

To Create a web server file

Create a directory in the name of CurrencyConverter and create a sub director myCurCon under WEB-INF/Src folder

Step 1: Create a service end point interface

Rules for creating service end point interface are as follows:

1. The interface must extend java.rmi.Remote interface
2. Every method in this interface must throw java.rmi.Remote exception
3. Every method return type and its parameter data type in this interface must be java primitive data types
4. The interface must not contain any public static final declarations.

```
package myCurCon;
public class ExchangeValues
{
public docuble dollars;
public double euros;
publci doubel yens;
}
package myCurCon;
public interface CurCon extends java.rmi.Remote
{
public ExchangeValues fromDollars(double dollars) throws java.rmi.RemoteException;
public ExchangeValues fromEuros(double dollars) throws java.rmi.RemoteException;
public ExchangeValues fromYens(double dollars) throws java.rmi.RemoteException;
}
```

Step 2: Create a class to implement this interface

```
public class CurConImpl implements CurCon
{
public ExchangeValues fromDollars(Double Dollars) thorws java.rmi.RemoteException
{
ExchangeValues ev=new ExchangeValues();
ev.dollars=dollars;
ev.euros=dollars*100;
ev.yens=dollars*200;
return ev;
}
}
```

Step 3: Compile using javac to create respective classes for ExchangeValues.java,

CurCon.java and CurConImpl.java
Step 4: Use wscompile to create WSDL doc. Inputs to wscompile are supplied via the following configuration file called config.xml file.

```
<? xml version="1.0" ?>
<configuration xmlns="url" >
<service name="HistoricCurrencyConverter" targetNameSpace="url"
typeNameSpace="url" packageName="myCurCon">
<interface name="muCurCon.CurCon" />
</service>
</configuration>
```

- Save this file as config.xml and use the following command to create wsdl doc.

wscompile –define – d WEB-INF –classpath WEB-INF/classes – model WEB-INF/model.xml.gz
config.xml

-d specifies the directory to receive the generated wsdl doc. which will be named as Historic Currencyconverter.wsdl

Step 5: Create another configuration file called jaxrpc-ri.xml which is used as deployment descriptor file as shown below

```
<? xml version="1.0" ?>
<webServices xmlns="url" targetNameSpace="url"
typeNameSpace="url"
urlPatternBase="/converter" >
<endpoint name="CurrConverter"
displayName="Currency Converter"
description="Converts b/w dollars, yens and euros"
interface="myCurCon.curCon"
model="/WEB-INF/model.xm.gz"
implementation="myCurCon.CurConImpl"/>
<enpointMapping endPointName="CurrConverter"
urlPattern="/Currency" />
</webServices>
```

SOAP:-

SOAP is a simple XML-based protocol that allows applications to exchange information over HTTP.

WSDL:-

- Web Services Description Language is the standard format for describing a web service in XML format.
- WSDL definition describes how to access a web service and what operations it will perform.
- WSDL is often used in combination with SOAP and XML Schema to provide web services over the Internet. A client program connecting to a web service can read the WSDL to determine what functions are available on the server. Then the client can then use SOAP to actually call one of the functions listed in the WSDL.

The WSDL Document Structure:

The main structure of a WSDL document looks like this:
<definitions>
<types>
definition of types.....
</types>
www.vidyarthiplus.com **2013**
<message>
definition of a message....
</message>
<portType>
<operation>
definition of a operation.....
</operation>
</portType>
<binding>
definition of a binding....
</binding>
<service>
definition of a service....
</service>
</definitions>

A WSDL document can also contain other elements, like extension elements and a service element that makes it possible to group together the definitions of several web services in one single WSDL document.

A WSDL document describes a web service using these major elements:

Element Defines
<types> The data types used by the web service
<message> The messages used by the web service
<portType> The operations performed by the web service
<binding> The communication protocols used by the web service
A WSDL document can also contain other elements, like extension elements, and
a service element that makes it possible to group together the definitions of
several web services in one single WSDL document.
Note: The second set of lines of the WSDL document is optional
First lines of the WSDL document

XML Schema:-

The purpose of an XML Schema is to define the legal building blocks of an XML document. It is used to represent data types of an XML document. It is an alternative to DTD (Document Type Definition). XML Schema defines elements, attributes and vales of elements and attributes.

An XML Schema:

- defines elements that can appear in a document
- defines attributes that can appear in a document
- defines which elements are child elements
- defines the order of child elements
- defines the number of child elements
- defines whether an element is empty or can include text
- defines data types for elements and attributes
- defines default and fixed values for elements and attributes

XML Schemas Data Types
One of the greatest strength of XML Schemas is the support for data types. Using XML schema it is easier to describe allowable document content; it is easier to validate the correctness of data; it is easier to convert data between different data types. Built in data types supported by XML schema XML Schema has a lot of built-in data types.

The most common types are:

- xs:string
- xs:decimal
- xs:integer
- xs:boolean
- xs:date
- xs:time

XML schema defines in the namespace *http://www.w3.org/2001/XMLschema* that contains built in data types. There are two classes of XML schema data types: Complex type is a data type is represented using markup.

 Simple type is a data type whose values are represented in XML doc by character data.

XML markup such as <types> is known as XML schema that conforms to W3C defined XML schema vocabulary which defines all or part of the vocabulary for another XML document. The <schema> is the root element for any XML schema document. The child elements of schema define the data types.

Example for writing a simple XML Schema

Step 1: Write a xsd file in which the desired structure of the XML document is defined and named it as StudentSchema.xsd.

```
<?xml version=”1.0”?>
<xs:schema xmlns:xs=” http://www.w3.org/2001/XMLSchma”>
<xs:element name=”student” >
<xs:complexType>
<xs:sequence>
<xs:element name=”name” value=”xs:string”
/>
<xs:element name=”regno” value=”xs:string” />
<xs:element name=”dept” value=”xs:string” />
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

The xs qualifier used to identify the schema elements and its types. The xs:schema is the root element. It takes the attributes xmlns:xs which has the value, “http://www.w3.org/2001/XMLSchema. This declaration indicates that the document follows the rule of XMLSchema defined by W3 in year 2001.

The xs:element is used to define the xml element. The Student element is complexType which has three child elements:name, regno and dept. All these elements are of simple type string.
Step 2: Write a XML document and reference the xsd file. Named it as
myXML.xml

```
<?xml version=”1.0”?>
<student xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
xsi:schemaLocation=”StudentSchema.xsd”>
<name> Raj </name>
<regno>3212654556</regno>
<dept>CSE</dept>
```

Various xml validation tools are available which can be used to validate xml and its schema document.

Advantages

- Supports data types
- Supports namespaces
- XML schemas support a set of data types, similar to the ones used in most common programming languages
- Provides the ability to define custom data types
- Easy to describe allowable content in the document
- Easy to validate the correctness of data
- Object oriented approach like inheritance and encapsulation can be used in creating the document
- Easier to convert data between different data types
- Easy to define data formats
- Easy to define restrictions on data
- It is written in xml so any xml editor can be used to edit xml schema Xml Parser can be used to parse the schema file
- XML schemas are more powerful than DTDs. Everything that can be

defined by the DTD can also be defined by schemas, but not vice versa.

Disadvantages:-

- Complex to design and learn
- Maintaining XML schema for large XML document slows down the processing of XML document

DTD for the above xml file instead of XML schema The purpose of a DTD (Document Type Definition) is to define the legal building blocks of an XML document.

A DTD defines the document structure with a list of legal elements and attributes.

Step 1: Create a DTD. Name it as student.dtd

<!ELEMENT student(name, regno, dept)>
<!ELEMENT name(#PCDATA)>
<!ELEMENT regno(#PCDATA)>
<!ELEMENT dept(#PCDATA)>
!ELEMENT student defines that the note element contains three elements: "name.
regno, dept"
PCDATA means parsed character data which is the text found between the start
tag and the end tag of an XML element.
Step 2: Write the XML document and reference the DTD file in it. Name it as
myXML.xml
<?xml version=”1.0”?>
<!DOCTYPE student SYSTEM student.dtd>
<student>
<name> Raj </name>
<regno>3212654556</regno>
<dept>CSE</dept>
</student>
!DOCTYPE student defines that the root element of this document is student and
links the myXML.xml file with the student.dtd file.

SOAP:-

SOAP is an acronym for Simple Object Access Protocol. SOAP defines a protocol for message exchange between applications. SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.

 SOAP describes envelope and message formats, and has a basic request/response

 SOAP is a communication protocol

- SOAP is for communication between applications

 SOAP is a format for sending messages

 SOAP communicates via Internet

 SOAP is platform independent

 SOAP is language independent

 SOAP is based on XML

 SOAP is simple and extensible

 SOAP is a W3C recommendation

- SOAP Building Blocks

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

 An Envelope element that identifies the XML document as a SOAP message

 A Header element that contains header information

 A Body element that contains call and response information

 A Fault element containing errors and status information

Structure of SOAP Message

The following code fragment gives an outline of the SOAP message

// code for HTTP binding

HTTP binding

// used to directs the message

Envelop

// helps the server to identify the message as SOAP

Header

// Optional. It adds features to SOAP message such as authentication, message route etc.