**WEBSERVICES:**

**These are built on top of open standards such as TCP/IP, HTML, JAVA, XML. Webservice is an xml based information exchange system that use the internet for app – app interaction.**

1. There are many framworks like glassfish, jersey and amany more.
2. Steps to create web service:

Step1: add tomcat and run server

Step 2: create dynamic web page

Step3: create package and class and write any method in that

Example:

**package** webserviceexample1;

**public** **class** HelloWorld {

**public** **float** addValue(**float** value){

**return** (value + 10);

}

**public** **float** subtractValue(**float** value){

**return** (value - 10);

}

}

Step 4: click webrvice - > createwebservice and click 2 checkboes and click finish it gives 2 projects and it will ron server will display In a exploerer window.

**3)what is WSDL**(wbservice description language)

: is a xml notation to describe webservice , it tells the client how to compose webservice request and describes the interface provided by the webservice provider:

**HOW TO COMPOSE WEBSERVICE:**

First create proxy class and use it.

**ELEMENTS IN WSDL :**

There are many 3 types of elements in wedl:

1) types

2) operations

3) binding.

There are many other elements also available will combined together to develop separate document.

<definition>

<datatypes>

<messages></messages>

</datatypes>

<porttypre>

<operation>

</porttype>

<binding/>

<service/>

Including <documentation> a nd <import>

**TARGETNAMESPACE:**

It declares the namespace for other xml and xsd document to refer to this schema, the target prefix in this case refers to the same namespace and you would use it within this schema definition to reference other element attribute, type and etc..also defined in this same schema definition.

For example : if we have a party the guest list is targetnamespace and guest wearing a name tag is the namespace, to identify a guest we can use guest list instead of seeing tag on them. If v want to search a person in the party can easily identify by seeing guest list.

SOAP: is a webservice protocol which relies exclusively on xml to provide messaging services. SOAP is designed to support expansion, so it has all sorts of other acronyms and abbreviations associated with it, such as WS-Addressing, WS-Policy, WS-Security, WS-Federation, WS-ReliableMessaging, WS-Coordination, WS-AtomicTransaction, and WS-RemotePortlets. In fact, you can find a whole laundry list of these standards on [Web Services Standards](http://www.w3.org/Submission/).

If u use ws-public nonned of using ws-security.

XML is used to make a request and receive response in soap can become extremely complex, in some lang u need to buils those req manually which becomes problems becoz soap is intolerant of rrors.

**Wsdl** file present in soap which helps in description how to req wbsrvice.

Soap is inbuilt with error handling if there is any error in req the the response contains response that u can fix the error. U can use smtp req instead of http req.

**DISADV**: working WITH SOAP IN JAVASCRIPT MEANS writing a ton of code to perform extremely simple tasks becoz u must create req xml structure absolutely everytime.

**REST**: it provides a lighter weight alternative. Instead of using xml to make a req, rest relies on a simple url in many cases. In some situation u must provide additional information in a special way, but most web service using rest rely on obtaining the needed information using url approach, it can use 4 diff http(get, post, delete and put) to perform task.

Unlike the soap it doesnot use xml instead it use diff formats to get responsive u find rest-based web service that output the data in CSV, JSON AND RSS.

As an example of working with REST, you could create a URL for [Weather Underground](http://www.wunderground.com/weather/api). The [API’s documentation page](http://www.wunderground.com/weather/api/d/docs) shows an example URL of http://api.wunderground.com/api/Your\_Key/conditions/q/CA/San\_Francisco.json. The information you receive in return is a JSON formatted document containing the weather for San Francisco. You can use your browser to interact with the Web service, which makes it a lot easier to create the right URL and verify the output you need to parse with your application.

JAXB: java architecture for xml binding: it maps java code and xml representatives, it have 2 features like it marshel the java objects into xml , inverse like unmarshel back xml to java objects.

CUSTOME CLASSES IN WEBSERVICE:

1)Create server side class:

package com.server;

public class CompanyService{

/\*\*

\* @return company object

\*/

public Company getCompanyData(){

Company company = new Company();

String [] employeeNames = new String[] { "Balaji", "LSMS" , "LLQ" };

company.companyID = 2311;

company.companyName = "Test";

company.employeeNames = employeeNames;

return company;

}

}

2)Deploy the web service using admin client:

<deployment xmlns="http://xml.apache.org/axis/wsdd/"

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

<service name="CompanyRepository" provider="java:RPC">

<parameter name="className" value="com.server.CompanyService"/>

<parameter name="allowedMethods" value="getCompanyData"/>

<parameter name="wsdlTargetNamespace" value="CompanyService"/>

<beanMapping qname="myNS:Company"

xmlns:myNS="urn:CompanyService"

languageSpecificType="com.server.Company"/>

</service>

</deployment>

3)Chaeck the web service wsdl

4)Create and run the client to accesss the web service