**Assignment No 5**

* A Web service, is a method of **communication between two applications**

or electronic devices **over the World Wide Web** (WWW).A web service can be defined as a **collection of open protocols and standards for exchanging information among systems or applications.**

* A Web service is an abstract notion that must be implemented by a concrete agent. The agent is the concrete piece of software or hardware that sends and receives messages, while the service is the resource characterized by the abstract set of functionality that is provided.

There are two types of web services:

* **SOAP-Based Web Services:**
* SOAP stands for Simple Object Access Protocol. SOAP is an XML based industry standard protocol for designing and developing web services. Since it’s XML based, it’s platform and language independent. So, our server can be based on JAVA and client can be on .NET, PHP etc. and vice versa.
* **RESTful Web Services-**
* REST (Representational State Transfer ) is an architectural style for developing web services. It’s getting popularity recently because it has small learning curve when compared to SOAP. Resources are core concepts of Restful web services and they are uniquely identified by their URIs.
* **REST** stands for **Representational State Transfe**r. RESTful web services are considered a performance-efficient alternative to the SOAP web services. REST is an architectural style, not a protocol.

SOAP vs REST:

**Steps involved in basic SOAP web service operational behaviour:**

The following are the steps involved in a basic SOAP web service operational behaviour:

1. The client program that wants to interact with another application prepares its request content as a SOAP message.
2. Then, the client program sends this SOAP message to the server web service as an HTTP POST request with the content passed as the body of the request.
3. The web service plays a crucial role in this step by understanding the SOAP request and

converting it into a set of instructions that the server program can understand.

1. The server program processes the request content as programmed and prepares the output as the response to the SOAP request.
2. Then, the web service takes this response content as a SOAP message and reverts to the SOAP HTTP request invoked by the client program with this response.
3. The client program web service reads the SOAP response message to receive the

outcome of the server program for the request content it sent as a request.

**Designing the solution:**

Java provides it’s own API to create both SOAP as well as RESTful web services.

1. **JAX-WS:** JAX-WS stands for Java API for XML Web Services. JAX-WS is XML based Java API to build web services server and client application.
2. **JAX-RS: J**ava API for RESTful Web Services (JAX-RS) is the Java API for creating REST web services. JAX-RS uses annotations to simplify the development and deployment of web services.

Both of these APIs are part of standard JDK installation, so we don’t need to add any jars to work with them.

**Web service implementation:**

* **Step 1: Choosing a container :** You can either deploy your web service in a

web container.

* Choose **File** > **New Project** (Ctrl-Shift-N on Linux and Windows).
* Select **Web Application** from the Java **Web** category.
* Name the project CalculatorWSApplication.
* Select a location for the project. Click **Next**.
* Select the server **[Glassfish / Tomcat]** and **Java EE** version and click **Finish.**
* **Step 2: Creating a Web Service from a Java Class :**
* Right-click the CalculatorWSApplication node and **choose New** >

**Web Service**.

* Name the web service CalculatorWS and type org.me.calculator

in Package.

* Keep “**Create Web Service from Scratch**” check box selected.
* If you are creating a Java EE project on GlassFish, select “**Implement Web Service as a Stateless Session Bean”**.
* Click **Finish**. The Projects window displays the structure of the new web service and the source code is shown in the editor area.
* **Step 3: Adding an Operation to the Web Service :**
* Find the web service's node in the Projects window. Right-click that node. A

context menu opens.

* Click **Add Operation** in either the visual designer or the context menu. The Add Operation dialog opens.
* In the upper part of the **Add Operation dialog box**, type **add** in **Name** and type **int**

in the **Return Type** drop-down list.

* In the lower part of the Add Operation dialog box, click **Add** and create a parameter of type **int** named **i**. Click **Add** again and create a parameter of type **int** called **j**.

Remove the **default hello operation**, either by deleting the hello() method in the **source code** or by selecting the hello operation in the **visual designer** and clicking **Remove Operation**.

In the editor, extend the skeleton **add operation.**

* **Step 5: Consuming the Web Service :** Once the web service is deployed, you need to **create a client** to make use of the web service's **add** method. Here, you can create three types of clients : a Java class in a Java SE application, a servlet, and a JSP page in a web application.

**Client 1: Java Class in Java SE Application**

* Choose **File** > **New Project**.
* Select **Java Application** from the **Java** category.
* Name the project CalculatorWS\_Client\_Application.
* Keep “**Create Main Class selected**” and accept all other default settings. Click

**Finish**.

* Right-click the CalculatorWS\_Client\_Application node and choose

**New** > **Web Service Client**. The New Web Service Client wizard opens.

* Select “Project as the **WSDL…** source. Click on **Browse** button. Browse to the CalculatorWS web service in the **CalculatorWSApplication** project. When you have selected the web service, click **OK**.
* The Projects window displays the new **web service client**, with a node for

the **add** method that is created:

* Double-click your **main class** so that it opens in the Source Editor. Drag the

**add** node below the main() method.

* In the main() method body, **replace the TODO comment** with code that

initializes values for i and j, calls add(), and prints the result.

Right-click the project node and choose **Run.**