

Assignment 5

Roll no: 43144

Name: Ruchika Pande

Title: Any distributed application to consume the web service

Problem Statement:

To create a simple web service and write a calculator application to consume the web service.

Objectives:

- To study about web services

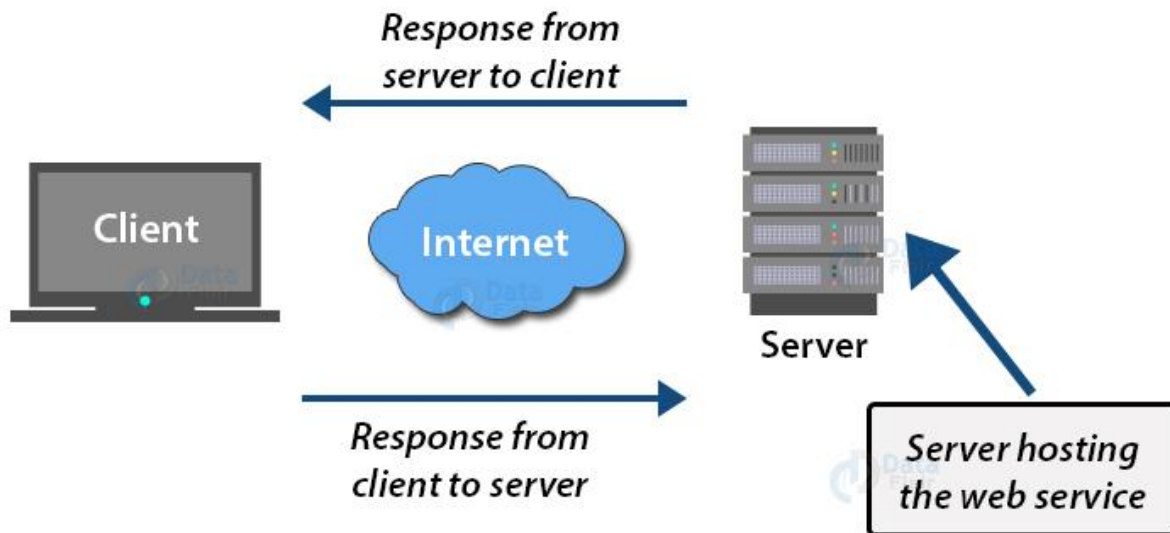
Theory:

Web Service :

Web service is a standardized medium to propagate communication between the client and server applications on the World Wide Web. A web service is a software module that is designed to perform a certain set of tasks.

- Web services in cloud computing can be searched for over the network and can also be invoked accordingly.
- When invoked, the web service would be able to provide the functionality to the client, which invokes that web service.

Working of web service :



The above diagram shows a very simplistic view of how a web service would actually work. The client would invoke a series of web service calls via requests to a server which would host the actual web service.

These requests are made through what is known as remote procedure calls. Remote Procedure Calls(RPC) are calls made to methods which are hosted by the relevant web service.

As an example, Amazon provides a web service that provides prices for products sold online via amazon.com. The front end or presentation layer can be in .Net or Java but either programming language would have the ability to communicate with the web service.

The main component of a web service design is the data which is transferred between the client and the server, and that is XML. XML (Extensible markup language) is a counterpart to HTML and easy to understand the intermediate language that is understood by many programming languages.

So when applications talk to each other, they actually talk in XML. This provides a common platform for applications developed in various programming languages to talk to each other.

Web services use something known as SOAP (Simple Object Access Protocol) for sending the XML data between applications. The data is sent over normal HTTP. The data which is sent from the web service to the application is called a SOAP message. The SOAP message is nothing but an XML document. Since the document is written in XML, the client application calling the web service can be written in any programming language.

Web Services Architecture :

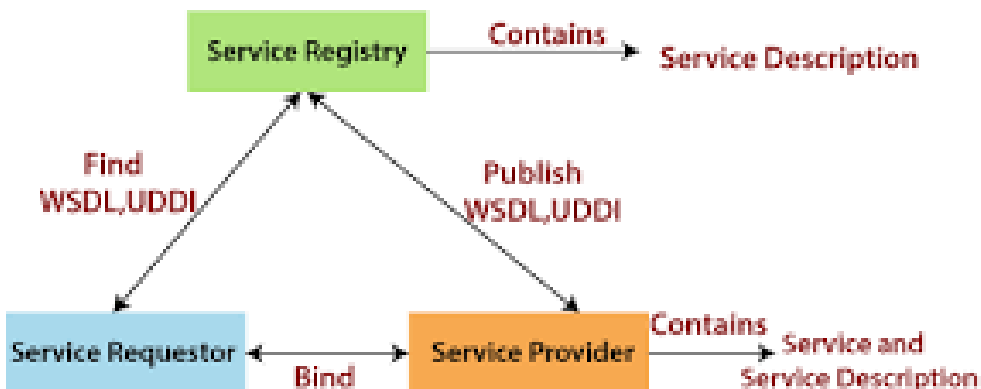
Every framework needs some sort of architecture to make sure the entire framework works as desired, similarly, in web services. The Web Services Architecture consists of three distinct roles as given below :

Provider - The provider creates the web service and makes it available to client applications who want to use it.

Requestor - A requestor is nothing but the client application that needs to contact a web service. The client application can be a .Net, Java, or any other language based application which looks for some sort of functionality via a web service.

Broker - The broker is nothing but the application which provides access to the UDDI. The UDDI, as discussed in the earlier topic, enables the client application to locate the web service.

The diagram below showcases how the Service provider, the Service requestor and Service registry interact with each other.



Publish - A provider informs the broker (service registry) about the existence of the web service by using the broker's publish interface to make the service accessible to clients

Find - The requestor consults the broker to locate a published web service

Bind - With the information it gained from the broker(service registry) about the web service, the requestor is able to bind, or invoke, the web service.

Web service Characteristics:

Web services have the following special behavioral characteristics:

1. They are XML-Based - Web Services uses XML to represent the data at the representation and data transportation layers. Using XML eliminates any networking, operating system, or platform sort of dependency since XML is the common language understood by all.
2. Loosely Coupled – Loosely coupled means that the client and the web service are not bound to each other, which means that even if the web service changes over time, it should not change the way the client calls the web service. Adopting a loosely coupled architecture tends to make software systems more manageable and allows simpler integration between different systems.
3. Synchronous or Asynchronous functionality- Synchronicity refers to the binding of the client to the execution of the service. In synchronous operations, the client will actually wait for the web service to complete an operation. An example of this is probably a scenario wherein a database read and write operation are being performed. If data is read from one database and subsequently written to another, then the operations have to be done in a sequential manner. Asynchronous operations allow a client to invoke a service and then execute other functions in parallel. This is one of the common and probably the most preferred techniques for ensuring that other services are not stopped when a particular operation is being carried out.
4. Ability to support Remote Procedure Calls (RPCs) - Web services enable clients to invoke procedures, functions, and methods on remote objects using an XML-based protocol. Remote procedures expose input and output parameters that a web service must support.
5. Supports Document Exchange - One of the key benefits of XML is its generic way of representing not only data but also complex documents. These documents can be as simple as representing a current address, or they can be as complex as representing an entire book.

Types of web services:

There are two types of web services:

1. SOAP Web Services
2. REST Web Services

SOAP Web Services :

SOAP is an XML-based protocol. The biggest advantage of using the SOAP Web Service is its own security. SOAP stands for Simple Object Access Protocol.

SOAP provides an envelope to send web services messages over the Internet, using the HTTP protocol. The messages are generally in XML format.



In simple words, SOAP is a technique to send an XML request over the Internet using HTTP protocol (hitting a URL), and in return getting an XML response. Every application serving SOAP requests has a WSDL file. WSDL is an XML, and it stands for Web Service Description Language. WSDL describes all the methods available in the web service, along with the request and response types. It describes the contract between service and client.

SOAP was intended to be a way to do remote procedure calls to remote objects by sending XML over HTTP.

If we look at the current software industry, you will find that SOAP is being used in enterprise applications, generally in legacy code. Today the world is moving fast towards the RESTful Web Services.

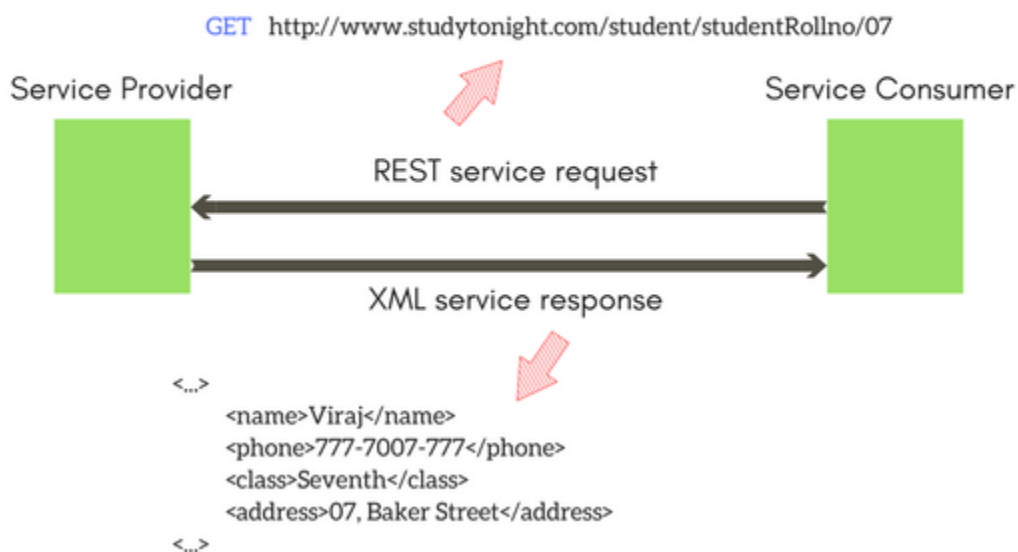
REST Web Services:

The REST stands for Representational State Transfer. REST is not a set of standards or rules, rather it is a style of software architecture. The applications which follow this architecture are referred to as RESTful

Unlike SOAP which targets the *actions*, REST concerns more on the resources. REST locates the resources by using URL and it depends on the type of transport protocol(with HTTP - GET, POST, PUT, DELETE,...) for the actions to be performed on the resources. The REST service locates the resource based on the URL and performs the action based on the transport action verb. It is more of architectural style and conventions based.

For Example: in a RESTful architecture, this URL `http://{serverAddress}/students/studentRollno/07` can be used to:

- To get student information by sending a REST call of GET type, and the service will return information of student with roll no as 07
- The same service can also be used to update the student data, by sending in the new values as Form data in a PUT request.



Difference between REST and SOAP:

Here are some of the basic differences between the two types of web services:

REST	SOAP
------	------

REST is a style of software architecture.	SOAP is a protocol or a set of standards.
REST stands for Representational State Transfer	SOAP stands for Simple Object Access Protocol
REST can use SOAP because it is a concept and can use any protocol like HTTP, SOAP etc.	SOAP cannot use REST because it itself is a protocol.
REST uses URI to expose business logic. But as REST works on the basis of type of HTTP request, hence same URI can work for more than a single type of operation.	SOAP uses the service interface to expose business logic.
REST does not define too many standards. REST is cool!	SOAP defines standards to be strictly followed.
REST inherits security measures from the underlying transport protocols.	SOAP defines its own security layer.
REST accepts different data formats like, Plain Text, HTML, JSON, XML etc.	SOAP only works with XML format.

Implementation:

1. Creating a web service CalculatorWSApplication:

Create a New Project for CalculatorWSApplication.

Create a package org.calculator

Create class CalculatorWS.

Right-click on the CalculatorWS and create New Web Service. IDE starts the glassfish server, builds the application and deploys the application on the server.

2. Consuming the Webservice:

Create a project with an CalculatorClient

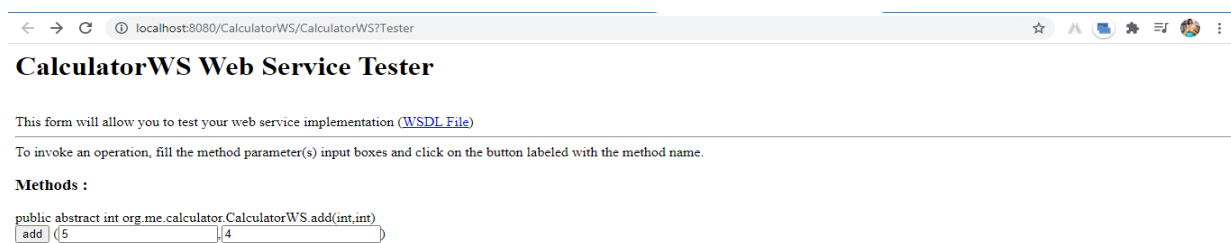
Create package org.calculator.client; add java class CalculatorWS.java, addresponse.java, add.java, CalculatorWSService.java and ObjectFactory.java

3. Creating servlet in web application:

Create new jsp page for creating user interface.

Output:

SOAP Test Client:



← → ↻ ⓘ localhost:8080/CalculatorWS/CalculatorWS7Tester ☆ 🔍 📄 ⚙️ 👤 ⋮

CalculatorWS Web Service Tester

This form will allow you to test your web service implementation ([WSDL File](#))

To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

Methods :

```
public abstract int org.me.calculator.CalculatorWS.add(int,int)
```

add (5) 4

Activate Windows
Go to PC settings to activate Windows.



← → ↻ localhost:8080/CalculatorWS/CalculatorWS?Tester

add Method invocation

Method parameter(s)

Type	Value
int	5
int	4

Method returned

int : "9"

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:add xmlns:ns2="http://calculator.me.org/">
      <i>5</i>
      <j>4</j>
    </ns2:add>
  </S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
```

Windows taskbar: 12:03 PM 24/5/21

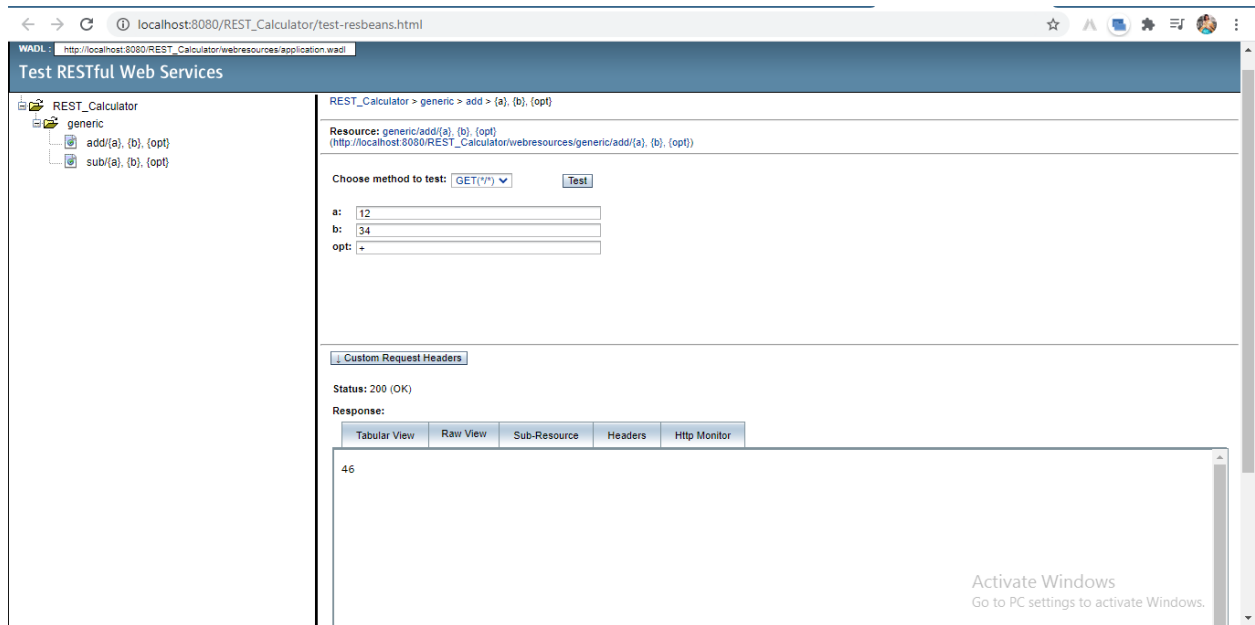
Output X

JB Database Process X GlassFish Server 4.1 X Shreyas Pande - C:\Users\Shreyas Pande X Retriever Output X CalculatorWS_Client_Application (run)

```
ant -f "C:\Users\Shreyas Pande\Documents\NetBeansProjects\CalculatorWS_Client_Application" -Dnb.internal.action=run
init:
Deleting: C:\Users\Shreyas Pande\Documents\NetBeansProjects\CalculatorWS_Client_Application\build\build-jar.properties
deps-jar:
Updating property file: C:\Users\Shreyas Pande\Documents\NetBeansProjects\CalculatorWS_Client_Application\build\build-jar.properties
wsimport-init:
files are up to date
wsimport-client-generate:
Compiling 1 source file to C:\Users\Shreyas Pande\Documents\NetBeansProjects\CalculatorWS_Client_Application\build\classes
compile:
run:
Addition of 10 and 20 is 30
BUILD SUCCESSFUL (total time: 1 second)
```

Windows taskbar: 12:03 PM 24/5/21

REST Test Client:



Conclusion:

In this assignment, we learned about the Web services approach to the Service Oriented Architecture concept. We used JAVA APIs for programming web services and learned how to program web services in JAVA.