****

**Name of the Student: Choudhari Arti Rajendra Roll no: 08**

**CLASS:- B.E.[I.T] Division: A Course:- 2015**

**Assignment No.5**

**COMPUTER LABORATORY-IX**

**Marks: /10**

**Date of Performance: Sign with Date:**

## ASSIGNMENT NO.5

**ProblemStatement:**

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

**Objective:**

1) The course aims to provide an understanding of the principles on which the distributed systems are based; their architecture, algorithms and how they meet the demands of Distributed applications.

2) The course covers the building blocks for a study related to the design and the implementation of distributed systems and applications.

**Outcomes:**

1) Demonstrate knowledge of the core concepts and techniques in distributed systems.

2) Learn how to apply principles of state-of-the-Art Distributed systems in practical application.

3) Design, build and test application programs on distributed systems.

**PEOs:2; POs: a,b,c,d,f,g,i, l, m ; PSOs: 1,2,3 and COs satisfied: 1, 2, 3.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Assignment No.** | **Assignment Title** | **Assignment Statement** | **Scenarios** | **Software Required** |
| 5 | Web-services (SOAP Based, RESTful) | To create a simple web service and write any distributed application to consume the web service. | This program will be run those conditions in which distributed services are requested by the client | 1) JDK version 8 mandatory  2) Netbeans IDE java(EE) version 8.2 with glassfish server.  Or eclipse java(EE) version (June 2017) (Oxygen)  with apache tomcat server version 9.0. |

## Tools /Environment:

Java Programming Environment, JDK 8, Netbeans IDE with GlassFish Server

## RelatedTheory:

## WebService:

A web service can be defined as a collection of open protocols and standards for exchanging information among systems orapplications.

A service can be treated as a web serviceif:

* + The service is discoverable through a simplelookup
  + It uses a standard XML format formessaging
  + It is available across internet/intranetnetworks.
  + It is a self-describing service through a simple XMLsyntax
  + The service is open to, and not tied to, any operating system/programminglanguage

## Types of Web Services:

There are two types of web services:

1. **SOAP**: 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 viceversa.
2. **REST**: 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 theirURIs.

## Web service architectures:

As part of a web service architecture, there exist three major roles.

**Service Provider** is the program that implements the service agreed for the web service and exposes the service over the internet/intranet for other applications to interact with.

**Service Requestor** is the program that interacts with the web service exposed by the Service Provider. It makes an invocation to the web service over the network to the Service Provider and exchanges information.

**Service Registry** acts as the directory to store references to the web services.

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

1. The client program that wants to interact with another application prepares its request content as a SOAPmessage.
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 therequest.
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 canunderstand.
4. The server program processes the request content as programmed and prepares the output as the response to the SOAPrequest.
5. 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 thisresponse.
6. 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 arequest.

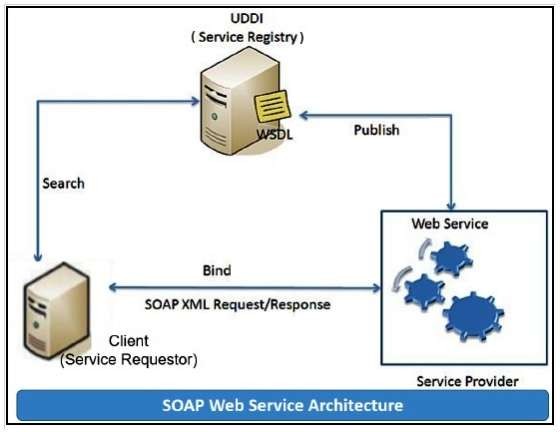
## SOAP web services:

**Simple Object Access Protocol** (**SOAP**) is an XML-based protocol for accessing web services. It is a W3C recommendation for communication between two applications, and it is a platform- and language-independent technology in integrated distributed applications.

While XML and HTTP together make the basic platform for web services, the following are the key components of standard SOAP web services:

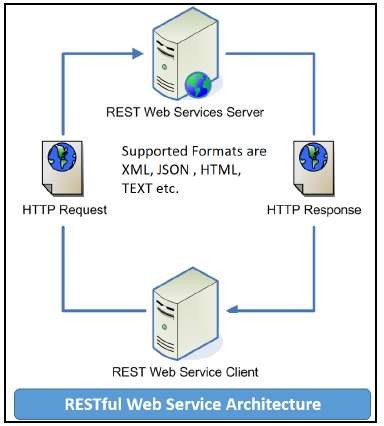
**Universal Description, Discovery, and Integration** (**UDDI**)*:* UDDI is an XMLbased framework for describing, discovering, and integrating web services. It acts as a directory of web service interfaces described in the WSDL language.

**Web Services Description Language** (**WSDL**)*:* WSDL is an XML document containing information about web services, such as the method name, method parameters, and how to invoke the service. WSDL is part of the UDDI registry. It acts as an interface between applications that want to interact based on web services. The following diagram shows the interaction between the UDDI, Service Provider, and service consumer in SOAP webservices:

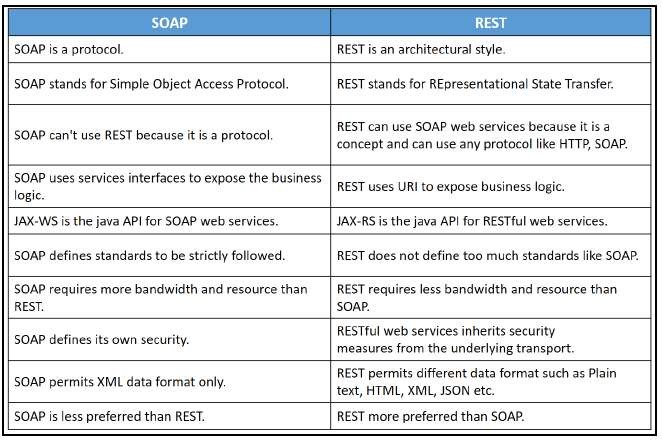


## RESTful web services

**REST** stands for **Representational State Transfer**. RESTful web services are considered a performance-efficient alternative to the SOAP web services. REST is an architectural style, not a protocol. Refer to the following diagram:



While both SOAP and RESTful support efficient web service development, the difference between these two technologies can be checked out in the following table :

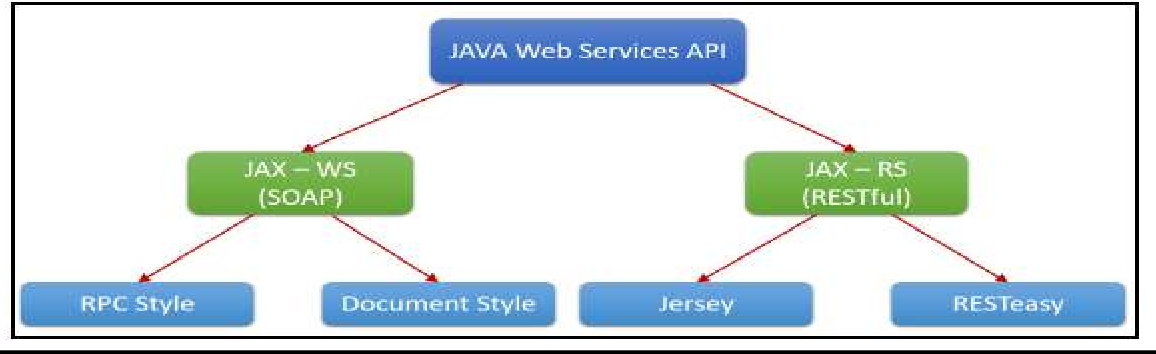


## Designing thesolution:

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 clientapplication.
2. **JAX-RS**: Java 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 withthem.



**Students are required to implement both i.e. using SOAP and RESTful APIs.**

**Implementing thesolution:**

1. **Creating a web serviceCalculatorWSApplication:**
   * Create New Project forCalculatorWSApplication.
   * Create a packageorg.calculator
   * Create class CalculatorWS.
   * Right-click on the CalculatorWSand create New Web Service.
   * IDE starts the glassfish server, builds the application and deploys the application on server.

## Consuming theWebservice:

* + Create a project with anCalculatorClient
  + Create packageorg.calculator.client;
  + add java class CalculatorWS.java, addresponse.java, add.java, CalculatorWSService.java andObjectFactory.java

## Creating servlet in webapplication

* + Create new jsp page for creating userinterface.

**Compiling and Executing thesolution:**

Right Click on the Project and Choose Run.

## Conclusion:

This assignment, described the Web services approach to the Service Oriented Architecture concept. Also, described the Java APIs for programming Web services and demonstrated examples of their use by providing detailed step-by-step examples of how to program Web services in Java.

Code:-

// CalculatorApplication.java

package com.calculator.service;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class CalculatorApplication {

public static void main(String[] args) {

SpringApplication.run(CalculatorApplication.class, args);

}

}

// CalculatorService.java

**package**com.calculator.service;

**import**org.springframework.web.bind.annotation.GetMapping;

**import**org.springframework.web.bind.annotation.PathVariable;

**import**org.springframework.web.bind.annotation.RestController;

@RestController

**publicclass**CalculatorService {

@GetMapping("/add/{x}/{y}")

**publicint** add(@PathVariable**int**x,@PathVariable**int**y) {

**return**x+y;

}

@GetMapping("/sub/{x}/{y}")

**publicint** sub(@PathVariable**int**x,@PathVariable**int**y) {

**return**x-y;

}

@GetMapping("/mul/{x}/{y}")

**publicint**mul(@PathVariable**int**x,@PathVariable**int**y) {

**return**x\*y;

}

@GetMapping("/div/{x}/{y}")

**publicint** div(@PathVariable**int**x,@PathVariable**int**y) {

**return**x/y;

}

}

//OUTPUT

. \_\_\_\_ \_ \_\_ \_ \_

/\\ / \_\_\_'\_ \_\_ \_ \_(\_)\_ \_\_ \_\_ \_ \ \ \ \

( ( )\\_\_\_ | '\_ | '\_| | '\_ \/ \_` | \ \ \ \

\\/ \_\_\_)| |\_)| | | | | || (\_| | ) ) ) )

' |\_\_\_\_| .\_\_|\_| |\_|\_| |\_\\_\_, | / / / /

=========|\_|==============|\_\_\_/=/\_/\_/\_/

\_[32m :: Spring Boot :: \_[39m \_[2m (v2.4.5)\_[0;39m

\_[2m2021-05-19 12:42:08.236\_[0;39m \_[32m INFO\_[0;39m \_[35m9104\_[0;39m \_[2m---\_[0;39m \_[2m[ main]\_[0;39m \_[36mc.c.service.CalculatorApplication \_[0;39m \_[2m:\_[0;39m Starting CalculatorApplication using Java 15.0.2 on ADMINRG-IN6CGJK with PID 9104 (C:\Users\Bhushan\Documents\workspace-spring-tool-suite-4-4.10.0.RELEASE\Calculator\target\classes started by Bhushan in C:\Users\Bhushan\Documents\workspace-spring-tool-suite-4-4.10.0.RELEASE\Calculator)

\_[2m2021-05-19 12:42:08.239\_[0;39m \_[32m INFO\_[0;39m \_[35m9104\_[0;39m \_[2m---\_[0;39m \_[2m[ main]\_[0;39m \_[36mc.c.service.CalculatorApplication \_[0;39m \_[2m:\_[0;39m No active profile set, falling back to default profiles: default

\_[2m2021-05-19 12:42:09.338\_[0;39m \_[32m INFO\_[0;39m \_[35m9104\_[0;39m \_[2m---\_[0;39m \_[2m[ main]\_[0;39m \_[36mo.s.b.w.embedded.tomcat.TomcatWebServer \_[0;39m \_[2m:\_[0;39m Tomcat initialized with port(s): 8080 (http)

\_[2m2021-05-19 12:42:09.348\_[0;39m \_[32m INFO\_[0;39m \_[35m9104\_[0;39m \_[2m---\_[0;39m \_[2m[ main]\_[0;39m \_[36mo.apache.catalina.core.StandardService \_[0;39m \_[2m:\_[0;39m Starting service [Tomcat]

\_[2m2021-05-19 12:42:09.349\_[0;39m \_[32m INFO\_[0;39m \_[35m9104\_[0;39m \_[2m---\_[0;39m \_[2m[ main]\_[0;39m \_[36morg.apache.catalina.core.StandardEngine \_[0;39m \_[2m:\_[0;39m Starting Servlet engine: [Apache Tomcat/9.0.45]

\_[2m2021-05-19 12:42:09.413\_[0;39m \_[32m INFO\_[0;39m \_[35m9104\_[0;39m \_[2m---\_[0;39m \_[2m[ main]\_[0;39m \_[36mo.a.c.c.C.[Tomcat].[localhost].[/] \_[0;39m \_[2m:\_[0;39m Initializing Spring embedded WebApplicationContext

\_[2m2021-05-19 12:42:09.414\_[0;39m \_[32m INFO\_[0;39m \_[35m9104\_[0;39m \_[2m---\_[0;39m \_[2m[ main]\_[0;39m \_[36mw.s.c.ServletWebServerApplicationContext\_[0;39m \_[2m:\_[0;39m Root WebApplicationContext: initialization completed in 1109 ms

\_[2m2021-05-19 12:42:09.609\_[0;39m \_[32m INFO\_[0;39m \_[35m9104\_[0;39m \_[2m---\_[0;39m \_[2m[ main]\_[0;39m \_[36mo.s.s.concurrent.ThreadPoolTaskExecutor \_[0;39m \_[2m:\_[0;39m Initializing ExecutorService 'applicationTaskExecutor'

\_[2m2021-05-19 12:42:09.839\_[0;39m \_[32m INFO\_[0;39m \_[35m9104\_[0;39m \_[2m---\_[0;39m \_[2m[ main]\_[0;39m \_[36mo.s.b.w.embedded.tomcat.TomcatWebServer \_[0;39m \_[2m:\_[0;39m Tomcat started on port(s): 8080 (http) with context path ''

\_[2m2021-05-19 12:42:09.851\_[0;39m \_[32m INFO\_[0;39m \_[35m9104\_[0;39m \_[2m---\_[0;39m \_[2m[ main]\_[0;39m \_[36mc.c.service.CalculatorApplication \_[0;39m \_[2m:\_[0;39m Started CalculatorApplication in 2.043 seconds (JVM running for 3.029)

<http://localhost:8080/add/10/2>

12

<http://localhost:8080/sub/10/2>

8

<http://localhost:8080/mul/10/2>

20

<http://localhost:8080/div/10/2>

5