**EXPERIMENT 2.**

**Problem Statement: Design a Web service using Simple Object Access Protocol (SOAP).**

**Description:** SOAP is an acronym for Simple Object Access Protocol. It is an XML-based messaging protocol for exchanging information among computers. SOAP is an application of the XML specification.

**Points to Note:**

SOAP is a communication protocol designed to communicate via Internet.

SOAP can extend HTTP for XML messaging.

SOAP provides data transport for Web services.

SOAP can exchange complete documents or call a remote procedure.

SOAP can be used for broadcasting a message.

SOAP is platform- and language-independent.

SOAP is the XML way of defining what information is sent and how.

SOAP enables client applications to easily connect to remote services and invoke remote methods.

Although SOAP can be used in a variety of messaging systems and can be delivered via a variety of transport protocols, the initial focus of SOAP is remote procedure calls transported via HTTP.

Other frameworks including CORBA, DCOM, and Java RMI provide similar functionality to SOAP, but SOAP messages are written entirely in XML and are therefore uniquely platform- and language-independent.

**Steps to create a web service:**

**1. Creating the web service**

Now let’s create our web service class. The web service method returns a MD5-hahsed value of an input string. Using the annotations @WebService for the class and @WebMethod for the service method.

With help of the annotations, the web service class looks like just a normal Java class.

Type the following command to compile the web service class (suppose the current directory is parent of the directory structure for the package: vce.webservices.server):

**javac -d . MD5WebService.java**

**2. Creating the server program:**

The JAX-WS implementation will create necessary infrastructure to start the server using some default configuration. And once started, the server is ready to receive client’s requests.

Type the following command to compile the server class:

**javac -d . WebServiceServer.java**

Start the server program using the following command:

**java vce.webservices.server.WebServiceServer**

We should see the server started and is waiting for client’s requests at the command prompt.

**3. Open a web browser and type the following URI into its address bar:**

[**http://localhost:9898/md5WebService?wsdl**](http://localhost:9898/md5WebService?wsdl)

The browser will receive an XML document that describes the web service

**4. Creating the client program:**

Before writing code for the client program, we have to generate some metadata code for the web service, by using the wsimporttool. This tool imports metadata about a web service provided by a URI (URI stands for **Uniform Resource Identifier**, and it's the official name for those things you see all the time on the Web that begin ' http: ' or ' mailto: ', for example http://www.w3.org/ , which is the URI for the home page of the World Wide Web) and generates Java source files required for a web service client. Syntax of the wsimport command is as follows:

**wsimport [options] <WSDL\_URI>**

Where **options**: specifies some options when generating the client code. You can type only wsimport in the command prompt to see a list of options.

**WSDL\_URI**: specifies a URI that describes the web service.

*A “WSDL” (Web Services Description Language) document describes a web service. It specifies the location of the service, and the methods of the service.*

Open another command prompt and change the current directory to the parent directory of vce\webservices. Type the following command:

**wsimport -keep -p vce.webservices.client http://localhost:9898/md5WebService?wsdl**

Based on the information obtained from the web service, the wsimport tool generates the following classes (both .java and .class files) and put them under package **net.codejava.webservices.client**:

package-info.java

ObjectFactory.java

MD5WebServiceService.java

MD5WebService.java

HashStringResponse.java

HashString.java

5. This client program invokes the web service method hashString() and passes “admin” as an argument, and it will display the result received from web service server to the console. Type the following command to compile the web service client class:

**javac -d . WebServiceClient.java**

And type this command to run the client program:

**java vce.webservices.client.WebServiceClient**

The client connects to the server, invokes the remote method and receives the result.

**Source Code:**

**//MD5WebService.java**

package vce.webservices.server;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import javax.jws.WebMethod;

import javax.jws.WebService;

@WebService

public class MD5WebService {

@WebMethod

public String hashString(String input) {

try {

MessageDigestmsgDigest = MessageDigest.getInstance("MD5");

byte[] inputBytes = input.getBytes();

byte[] hashedBytes = msgDigest.digest(inputBytes);

StringBuffer sb = new StringBuffer();

for (int i = 0; i<hashedBytes.length; i++) {

sb.append(Integer.toString((hashedBytes[i] & 0xff) + 0x100, 16).substring(1));

}

return sb.toString();

} catch (NoSuchAlgorithmException ex) {

ex.printStackTrace();

return "";

}

}

}

**//WebServiceServer.java**

package vce.webservices.server;

import javax.xml.ws.Endpoint;

public class WebServiceServer {

/\*\*

\* Starts a simple server to deploy the web service.

\*/

public static void main(String[] args) {

String bindingURI = "http://localhost:9898/md5WebService";

MD5WebService webService = new MD5WebService();

Endpoint.publish(bindingURI, webService);

System.out.println("Server started at: " + bindingURI);

}

}

**//WebServiceClient.java**

package vce.webservices.client;

public class WebServiceClient {

/\*\* \* Starts the web service client. \*/

public static void main(String[] args) {

MD5WebServiceService client = new MD5WebServiceService();

MD5WebService md5Webservice = client.getMD5WebServicePort();

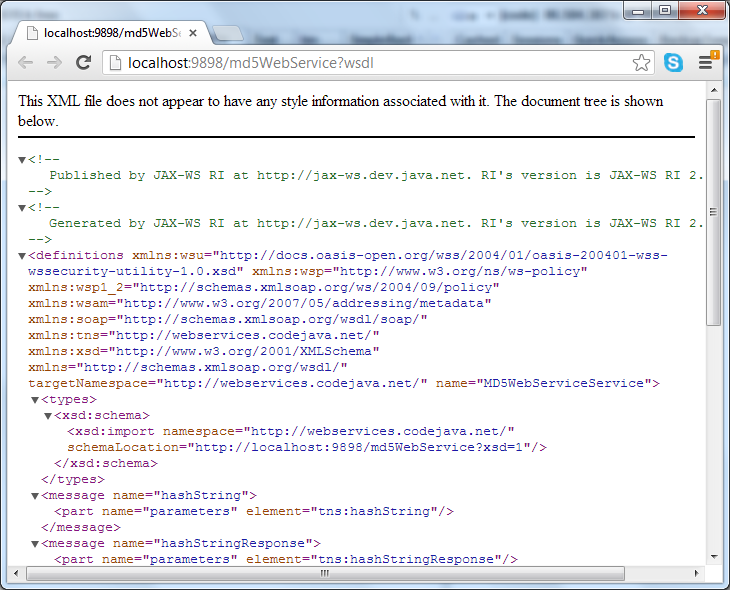
String hash = md5Webservice.hashString("hyderabad");

System.out.println("MD5 hash string: " + hash);

}

}

**Input / Output:**

****

