

Rajalakshmi Engineering College

Rajalakshmi Nagar, Thandalam, Chennai - 602 105
Department of Computer Science and Engineering



Web Services

Simple Applications

User Manual

(Windows Operating System)



Prepared by:

B.BHUVANESWARAN

Assistant Professor (SS) / CSE / REC

bhuvaneshwaran@rajalakshmi.edu.in

SOAP WEB SERVICE EXAMPLE IN JAVA USING ECLIPSE

Create hello world SOAP web service example in eclipse. Eclipse provides good API for creating web services. Eclipse will do all work for you-creating WSDL, stub, endpoints etc.

Steps for creating web services in eclipse:

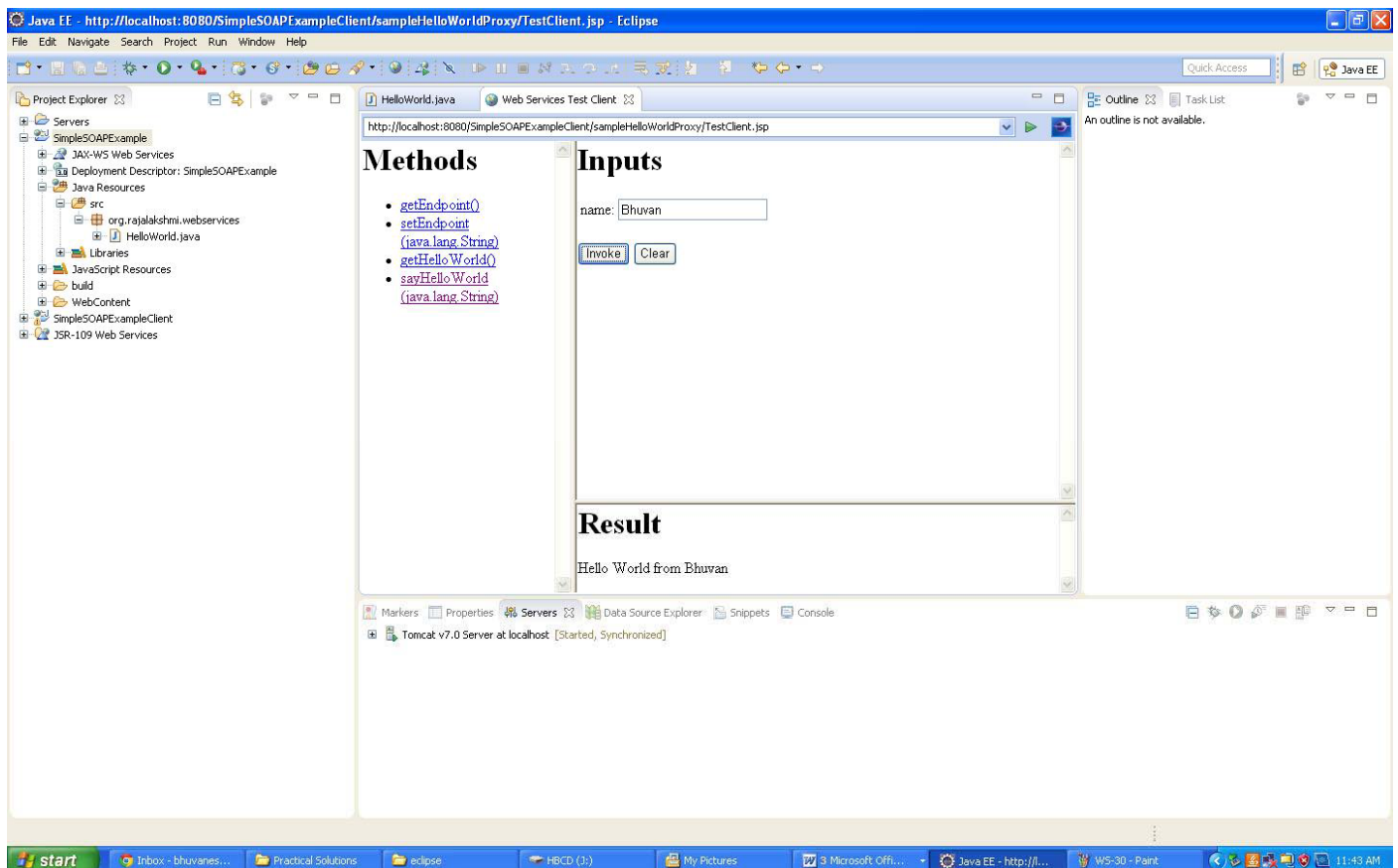
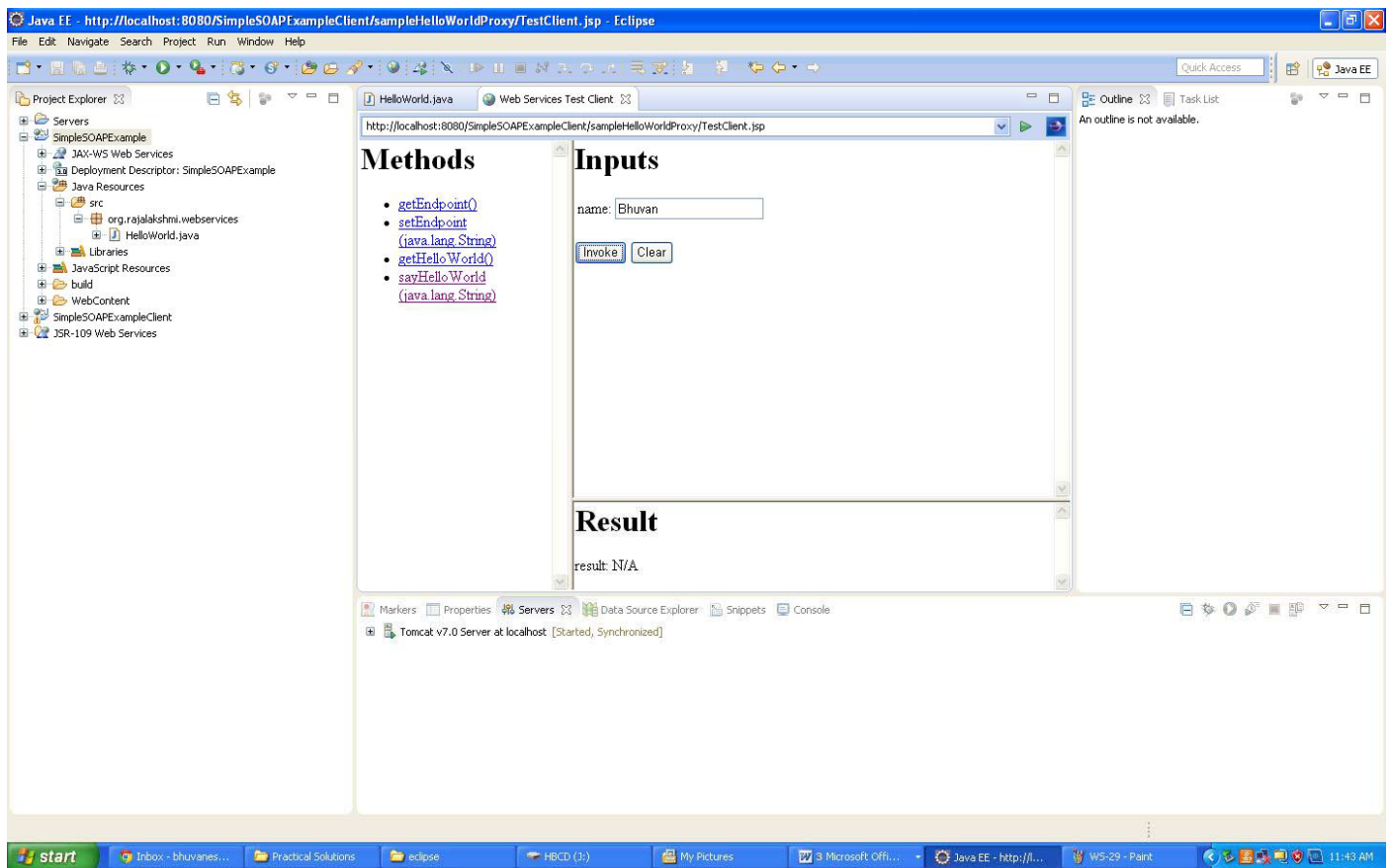
1. Open the eclipse IDE.
2. Switch to J2EE Perspective.
3. Select File → New → Dynamic Web Project.
4. Enter the Project and name as SimpleSOAPExample.
5. Click Next.
6. Click Next.
7. Click Finish.
8. Right click on Project → New → Package.
9. Enter the package name as org.rajalakshmi.webservices.
10. Click Finish.
11. Right click on Package → New → Class.
12. Enter the Java class name as HelloWorld.
13. Uncheck all the method stub and comments.
14. Click Finish.
15. Enter the following code:

```
package org.rajalakshmi.webservices;  
  
public class HelloWorld {  
    public String sayHelloWorld(String name)  
    {  
        return "Hello World from " + name;  
    }  
}
```

16. Right click on project → New → Other → Web Services → Web service.
17. Click Next.

18. In service implementation text box, write fully qualified class name of above created class (HelloWorld.java) and move both above slider to maximum level (i.e. Test service and Test Client level) and click on Finish. You are done!! A new project named SimpleSOAPExampleClient will be created in your work space.
19. Click Next.
20. Click Next.
21. Click Start Server.
22. Click Next.
23. Click Next.
24. Click Next.
25. Click Finish.
26. After clicking start server, eclipse will open test web service API. With this test API, you can test your web service.

OUTPUT:



JAX-WS WEB SERVICE - CREATING AND PUBLISHING THE WEB SERVICE

In this example, we create a SOAP based web service for a simple Java Calculator class with operations 'add' and 'subtract'. We then create a web service client which consumes the web service and displays the result of the invoked web service.

Steps for creating web services in eclipse:

1. Open the eclipse IDE.
2. Switch to Java Perspective.
3. Select File → New → Java Project.
4. Enter the Project name as CalcWS.
5. Click Next.
6. Click Finish.
7. Right click on Project → New → Package.
8. Enter the Package name as org.rajalakshmi.webservices.calc.
9. Click Finish.
10. Right click on Package → New → Class.
11. Enter the Java class name as Calculator.
12. Uncheck all the method stub and comments.
13. Click Finish.
14. Enter the following code:

```
package org.rajalakshmi.webservices.calc;

import javax.jws.WebService;

@WebService
public class Calculator {

    public int add(int a, int b)
    {
        return (a + b);
    }

    public int sub(int a, int b)
    {
        return (a - b);
    }
}
```

15. The @WebService annotation at the beginning of the class definition tells the Java interpreter that we intend to publish ALL the methods of this class as a web service. If we want to publish only particular methods then we can use @WebMethod annotation before the method signature.
16. In this step we want to publish our class as a web service endpoint.
17. For that we use the static publish() method of the javax.xml.ws.Endpoint class to publish our “Calculator” class as a web service in the specified context root.
18. Right click on Package → New → Class.
19. Enter the Java class name as CalcEndpointPublisher.
20. Check the public static void main(String[] args) option.
21. Enter the following code:

```
package org.rajalakshmi.webservices.calc;  
  
import javax.xml.ws.Endpoint;  
  
public class CalcEndPointPublisher {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        Endpoint.publish("http://localhost:8080/CalcWS/Calculator",  
            new Calculator());  
    }  
}
```

22. Right Click on the code window → Run As → Java Application.
23. You may not get output in the Console.
24. To check whether our class is published as web service, open a browser and type the URL mentioned in the endpoint with a parameter ?wsdl appended.

`http://localhost:8080/CalcWS/Calculator?wsdl`

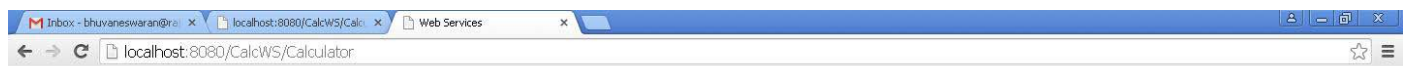
25. When you run the application, the Java SE 6 platform has a small web application server that will publish the web service at the address

`http://localhost:8080/CalcWS/Calculator`

while the JVM is running. If you see a large amount of XML that describes the functionality behind the web service, then the deployment is successful.

OUTPUT:

```
<!--
  Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.2.4-b01.
-->
<!--
  Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.2.4-b01.
-->
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:wspl_2="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://calc.webservices.rajalakshmi.org/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/"
  targetNamespace="http://calc.webservices.rajalakshmi.org/" name="CalculatorService">
  <types>
    <xsd:schema>
      <xsd:import namespace="http://calc.webservices.rajalakshmi.org/" schemaLocation="http://localhost:8080/CalcWS/Calculator?xsd=1"/>
    </xsd:schema>
  </types>
  <message name="add">
    <part name="parameters" element="tns:add"/>
  </message>
  <message name="addResponse">
    <part name="parameters" element="tns:addResponse"/>
  </message>
  <message name="sub">
    <part name="parameters" element="tns:sub"/>
  </message>
  <message name="subResponse">
    <part name="parameters" element="tns:subResponse"/>
  </message>
  <portType name="Calculator">
    <operation name="add">
      <input wsam:Action="http://calc.webservices.rajalakshmi.org/Calculator/addRequest" message="tns:add"/>
      <output wsam:Action="http://calc.webservices.rajalakshmi.org/Calculator/addResponse" message="tns:addResponse"/>
    </operation>
    <operation name="sub">
      <input wsam:Action="http://calc.webservices.rajalakshmi.org/Calculator/subRequest" message="tns:sub"/>
      <output wsam:Action="http://calc.webservices.rajalakshmi.org/Calculator/subResponse" message="tns:subResponse"/>
    </operation>
  </portType>
  <binding name="CalculatorPortBinding" type="tns:Calculator">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <operation name="add">
      <soap:operation soapAction=""/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  </binding>
</definitions>
```



Web Services

Endpoint	Information
Service Name: (http://calc.webservices.rajalakshmi.org/) CalculatorService	Address: http://localhost:8080/CalcWS/Calculator
Port Name: (http://calc.webservices.rajalakshmi.org/) CalculatorPort	WSDL: http://localhost:8080/CalcWS/Calculator?wsdl
	Implementation class: org.rajalakshmi.webservices.calc.Calculator

CREATING AND CONSUMING A WEB SERVICE CLIENT

1. Having published the web service, we now create a client which communicates with the service and displays the result.
2. Select File → New → Java Project.
3. Enter the Project name as CalcWSCClient.
4. Click Next.
5. Click Finish.
6. JAX-WS provides a tool called “wsimport” for generating the artifacts required for creating and consuming a web service. “wsimport” takes a wsdl file as input.
7. From the project folder “CalcWSCClient” in command prompt or terminal, issue the following command:

```
wsimport -s . http://localhost:8080/CalcWS/Calculator?wsdl
```

```
C:\>cd CS2358
```

```
C:\CS2358>cd CalcWSCClient
```

```
C:\CS2358\CalcWSCClient>cd src
```

```
C:\CS2358\CalcWSCClient\src>wsimport -s . http://localhost:8080/CalcWS/Calculator?wsdl
```

```
parsing WSDL...
```

```
Generating code...
```

```
Compiling code...
```

```
C:\CS2358\CalcWSCClient\src>
```

(Note: Your Web Service needs to be running)

8. After refreshing your Eclipse workspace you should see the generated files.
9. Let us now create a Java class with main method to run this client.
10. Right click on project → New → Package.
11. Enter the package name as org.rajalakshmi.webservices.calc.client.
12. Click Finish.
13. Right click on package → New → Class.
14. Enter the Java class name as CalcClient.

15. Check the public static void main(String[] args) option.

16. Enter the following code:

```
package org.rajalakshmi.webservices.calc.client;

import org.rajalakshmi.webservices.calc.Calculator;
import org.rajalakshmi.webservices.calc.CalculatorService;

public class CalcClient {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int a = 10;
        int b = 20;
        CalculatorService calcService = new CalculatorService();
        Calculator calc = calcService.getCalculatorPort();
        System.out.println(a + " + " + b + " = " + calc.add(a, b));
        System.out.println(a + " - " + b + " = " + calc.sub(a, b));
    }
}
```

17. Right Click on the code window → Run As → Java Application.

18. You will get the following output in the console:

```
10 + 20 = 30
10 - 20 = -10
```

OUTPUT:

```
10 + 20 = 30  
10 - 20 = -10
```

JAX-WS WEB SERVICE - CREATING AND PUBLISHING THE WEB SERVICE - DB

In this example, we create a SOAP based web service for a simple Student class with operations DispStud. We then create a web service client which consumes the web service and displays the result of the invoked web service.

Steps for creating web services in eclipse:

1. Open the eclipse IDE.
2. Switch to Java Perspective.
3. Select File → New → Java Project.
4. Enter the Project name as WSDB.
5. Click Next.
6. Click Finish.
7. Right click on Project → New → Package.
8. Enter the Package name as org.rajalakshmi.webservices.database.
9. Click Finish.
10. Right click on Package → New → Class.
11. Enter the Java class name as Student.
12. Uncheck all the method stub and comments.
13. Click Finish.
14. Enter the following code:

```
package org.rajalakshmi.webservices.database;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

import javax.xml.ws.WebService;

@WebService
public class Student {
    // TODO Auto-generated method stub
    public String DispStud (String rollNo)
    {
        Connection conn = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
```

```

String sName = null;
try
{
    Class.forName("com.mysql.jdbc.Driver");
    String URL = "jdbc:mysql://localhost:3306/rec";
    conn = DriverManager.getConnection(URL, "root", "admin");
    ps = conn.prepareStatement("select * from student where rollno = ?");
    ps.setString(1, rollNo);
    rs = ps.executeQuery();
    if(rs.next())
    {
        sName = rs.getString(2);
    }
    else
    {
        sName = "Not Found...";
    }
    rs.close();
    ps.close();
    conn.close();
    return sName;
}
catch(Exception e)
{
    System.out.println(e);
    return "Error..." ;
}
}
}

```

15. The @WebService annotation at the beginning of the class definition tells the Java interpreter that we intend to publish ALL the methods of this class as a web service. If we want to publish only particular methods then we can use @WebMethod annotation before the method signature.

16. In this step we want to publish our class as a web service endpoint.

17. For that we use the static publish() method of the javax.xml.ws.Endpoint class to publish our “Student” class as a web service in the specified context root.

18. Right click on Package → New → Class.

19. Enter the Java class name as StudentEndpointPublisher.

20. Check the public static void main(String[] args) option.

21. Enter the following code:

```

package org.rajalakshmi.webservices.database;
import javax.xml.ws.Endpoint;

public class StudEndPointPublisher {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Endpoint.publish("http://localhost:8090/WSDB/Student", new Student());
    }
}

```

22. Right Click on the code window → Run As → Java Application.

23. You may not get output in the Console.

24. To check whether our class is published as web service, open a browser and type the URL mentioned in the endpoint with a parameter ?wsdl appended.

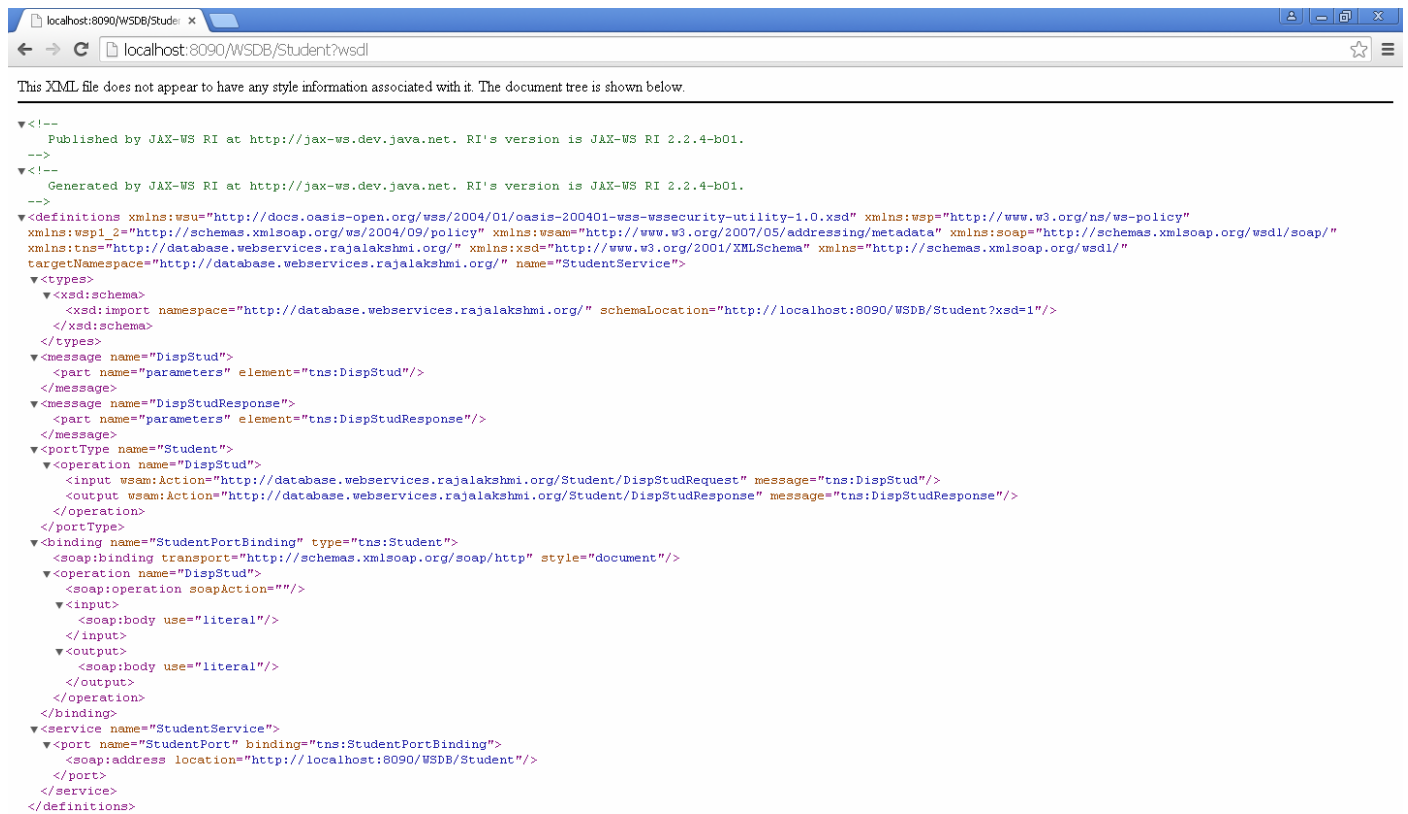
`http://localhost:8090/WSDB/Student?wsdl`

25. When you run the application, the Java SE 6 platform has a small web application server that will publish the web service at the address

`http://localhost:8090/WSDB/Student`

while the JVM is running. If you see a large amount of XML that describes the functionality behind the web service, then the deployment is successful.

OUTPUT:



The screenshot shows a web browser window with the address bar displaying `localhost:8090/WSDB/Student?wsdl`. The page content displays the WSDL (Web Services Description Language) for a service named `StudentService`. The XML document includes a header with version information, a `definitions` block with various namespaces, and a `portType` named `Student` with an operation `DispStud`. The `binding` and `service` elements are also present, defining the endpoint and the service name.

```
<!--
  Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.2.4-b01.
-->
<!--
  Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.2.4-b01.
-->
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:wsp="http://www.w3.org/ns/ws-policy"
xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://database.webservices.rajalakshmi.org/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://database.webservices.rajalakshmi.org/" name="StudentService">
  <types>
    <xsd:schema>
      <xsd:import namespace="http://database.webservices.rajalakshmi.org/" schemaLocation="http://localhost:8090/WSDB/Student?xsd=1"/>
    </xsd:schema>
  </types>
  <message name="DispStud">
    <part name="parameters" element="tns:DispStud"/>
  </message>
  <message name="DispStudResponse">
    <part name="parameters" element="tns:DispStudResponse"/>
  </message>
  <portType name="Student">
    <operation name="DispStud">
      <input wsam:Action="http://database.webservices.rajalakshmi.org/Student/DispStudRequest" message="tns:DispStud"/>
      <output wsam:Action="http://database.webservices.rajalakshmi.org/Student/DispStudResponse" message="tns:DispStudResponse"/>
    </operation>
  </portType>
  <binding name="StudentPortBinding" type="tns:Student">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <operation name="DispStud">
      <soap:operation soapAction="">
        <input>
          <soap:body use="literal"/>
        </input>
        <output>
          <soap:body use="literal"/>
        </output>
      </operation>
    </binding>
  </service name="StudentService">
    <port name="StudentPort" binding="tns:StudentPortBinding">
      <soap:address location="http://localhost:8090/WSDB/Student"/>
    </port>
  </service>
</definitions>
```



Web Services

Endpoint	Information
Service Name: (http://database.webservices.rajalakshmi.org/) StudentService	Address: http://localhost:8090/WSDB/Student
Port Name: (http://database.webservices.rajalakshmi.org/) StudentPort	WSDL: http://localhost:8090/WSDB/Student?wsdl
	Implementation class: org.rajalakshmi.webservices.database.Student



CREATING AND CONSUMING A WEB SERVICE CLIENT

1. Having published the web service, we now create a client which communicates with the service and displays the result.
2. Select File → New → Dynamic Web Project.
3. Enter the Project name as WSDBClient.
4. Click Next.
5. Click Next.
6. Click Finish.
7. JAX-WS provides a tool called “wsimport” for generating the artifacts required for creating and consuming a web service. “wsimport” takes a wsdl file as input.
8. From the project folder “WSDBClient” in command prompt or terminal, issue the following command:

```
wsimport -s . http://localhost:8090/WSDB/Student?wsdl
```

```
C:\>cd CS2358
```

```
C:\CS2358>cd WSDBClient
```

```
C:\CS2358\WSDBClient>cd src
```

```
C:\CS2358\WSDBClient\src>wsimport -s . http://localhost:8090/WSDB/Student?wsdl
```

```
parsing WSDL...
```

```
Generating code...
```

```
Compiling code...
```

```
C:\CS2358\WSDBClient\src>
```

(Note: Your Web Service needs to be running)

9. After refreshing your Eclipse workspace you should see the generated files.
10. Let us now create a HTML file to get the student roll no.
11. Right click on project → New → HTML File.
12. Enter the HTML file name as Student.
13. Click Next.
14. Click Finish.
15. Enter the following code:


```

<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Student Details</title>
</head>
<body>
<form method="post" action = "StudDisp.jsp">
<input type="text" name="rollno">
<input type="submit" value="Get">
</form>
</body>
</html>

```

16. Let us now create a JSP file to run this client.

17. Right click on project → New → JSP File.

18. Enter the JSP file name as StudDisp.

19. Click Next.

20. Click Finish.

21. Enter the following code:

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Student</title>
</head>
<body>
<%
org.rajalakshmi.webservices.database.StudentService studService = new
    org.rajalakshmi.webservices.database.StudentService();
org.rajalakshmi.webservices.database.Student student = studService.getStudentPort();
out.println(student.dispStud(request.getParameter("rollno")));
%>
</body>
</html>

```

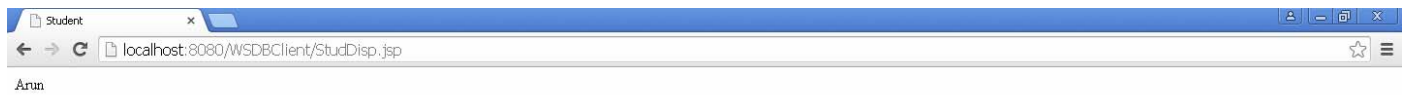
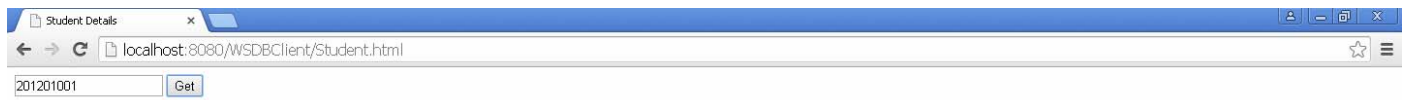
22. Select the Student.html file.

23. Right Click on the code window → Run As → Run on Server.

24. Enter the student roll no.

25. You will get the output as the student name.

OUTPUT:



JAX-WS WEB SERVICE - CREATING AND PUBLISHING THE WEB SERVICE - DB

Develop a web service for an airline management and implement the following scenario using database.

(a) Check ticket availability.

(b) Check air services through travel agent.

(c) Search a passenger whether he / she travelled in a particular date or not.

Database

```
mysql> create database airline;
Query OK, 1 row affected (0.01 sec)
```

```
mysql> use airline;
Database changed
```

```
mysql> create table airways(
-> flightno varchar(6),
-> airline varchar(20),
-> source varchar(20),
-> destination varchar(20),
-> departure varchar(5),
-> arrival varchar(5),
-> ticketsavail integer);
Query OK, 0 rows affected (0.08 sec)
```

```
mysql> desc airways;
```

Field	Type	Null	Key	Default	Extra
flightno	varchar(6)	YES		NULL	
airline	varchar(20)	YES		NULL	
source	varchar(20)	YES		NULL	
destination	varchar(20)	YES		NULL	
departure	varchar(5)	YES		NULL	
arrival	varchar(5)	YES		NULL	
ticketsavail	int(11)	YES		NULL	

```
7 rows in set (0.02 sec)
```

```
mysql> insert into airways values
-> ('AI-263', 'Air India', 'Chennai', 'Bengaluru',
-> '06.00', '06.55', 100);
Query OK, 1 row affected (0.02 sec)
```

```
mysql> insert into airways values
-> ('6E-271', 'Indigo', 'Chennai', 'Bengaluru',
-> '09.10', '10.00', 50);
Query OK, 1 row affected (0.01 sec)
```

```
mysql> insert into airways values
-> ('9W-464', 'Jet Airways', 'Chennai', 'Mumbai',
-> '05.50', '07.40', 25);
Query OK, 1 row affected (0.01 sec)
```

```
mysql> insert into airways values
-> ('SG-402', 'Spicejet', 'Chennai', 'Mumbai',
-> '05.50', '07.35', 10);
Query OK, 1 row affected (0.02 sec)
```

Steps for creating web services in eclipse:

1. Open the eclipse IDE.
2. Switch to Java Perspective.
3. Select File → New → Java Project.
4. Enter the Project name as AirlineServer.
5. Click Next.
6. Click Finish.
7. Right click on Project → New → Package.
8. Enter the Package name as org.rajalakshmi.webservices.database.
9. Click Finish.
10. Right click on Package → New → Class.
11. Enter the Java class name as Availability.
12. Uncheck all the method stub and comments.
13. Click Finish.
14. Enter the following code:

```
package org.rajalakshmi.webservices.database;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

import javax.jws.WebService;
```

```

@WebService
public class Availability {
    public String dispAvail (String source, String destination)
    {
        Connection conn = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        String str = "";
        try
        {
            Class.forName("com.mysql.jdbc.Driver");
            String URL = "jdbc:mysql://localhost:3306/airline";
            conn = DriverManager.getConnection(URL, "root", "admin");
            ps = conn.prepareStatement("select * from airways where source = ?
            and destination = ?");
            ps.setString(1, source);
            ps.setString(2, destination);
            rs = ps.executeQuery();
            str = str + "<table border = 1>";
            str = str + "<tr>";
            str = str + "<th>Flight No</th>";
            str = str + "<th>Airways</th>";
            str = str + "<th>Source</th>";
            str = str + "<th>Destination</th>";
            str = str + "<th>Departure</th>";
            str = str + "<th>Arrival</th>";
            str = str + "<th>Tickets Available</th>";
            str = str + "</tr>";
            while(rs.next())
            {
                str = str + "<tr>";
                str = str + "<td>" + rs.getString(1) + "</td>";
                str = str + "<td>" + rs.getString(2) + "</td>";
                str = str + "<td>" + rs.getString(3) + "</td>";
                str = str + "<td>" + rs.getString(4) + "</td>";
                str = str + "<td>" + rs.getString(5) + "</td>";
                str = str + "<td>" + rs.getString(6) + "</td>";
                str = str + "<td>" + rs.getString(7) + "</td>";
                str = str + "</tr>";
            }
            str = str + "</table>";
            rs.close();
            ps.close();
            conn.close();
            return str;
        }
        catch(Exception e)
        {
            return (e.toString()) ;
        }
    }
}

```

15. The @WebService annotation at the beginning of the class definition tells the Java interpreter that we intend to publish ALL the methods of this class as a web service. If we want to publish only particular methods then we can use @WebMethod annotation before the method signature.

16. In this step we want to publish our class as a web service endpoint.

17. For that we use the static publish() method of the javax.xml.ws.Endpoint class to publish our “Availability” class as a web service in the specified context root.

18. Right click on Package → New → Class.

19. Enter the Java class name as AvailabilityEndpointPublisher.

20. Check the public static void main(String[] args) option.

21. Enter the following code:

```
package org.rajalakshmi.webservices.database;

import javax.xml.ws.Endpoint;

public class AvailabilityEndPointPublisher {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Endpoint.publish("http://localhost:8090/AirlineServer/Availability",
            new Availability());
    }

}
```

22. Right Click on the code window → Run As → Java Application.

23. You may not get output in the Console.

24. To check whether our class is published as web service, open a browser and type the URL mentioned in the endpoint with a parameter ?wsdl appended.

`http://localhost:8090/AirlineServer/Availability?wsdl`

25. When you run the application, the Java SE 6 platform has a small web application server that will publish the web service at the address

`http://localhost:8090/AirlineServer/Availability`

while the JVM is running. If you see a large amount of XML that describes the functionality behind the web service, then the deployment is successful.

OUTPUT:

Web Services	
Endpoint	Information
Service Name: (http://database.webservices.rajalakshmi.org/) AvailabilityService	Address: http://localhost:8090/AirlineServer/Availability
Port Name: (http://database.webservices.rajalakshmi.org/) AvailabilityPort	WSDL: http://localhost:8090/AirlineServer/Availability?wsdl
	Implementation class: org.rajalakshmi.webservices.database.Availability



localhost:8090/AirlineServer/Availability?wsdl

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.2.4-b01. -->
<!-- Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.2.4-b01. -->
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:wsp="http://www.w3.org/ns/ws-policy"
xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:soap="http://schemas.xmlsoap.org/soap/"
xmlns:tns="http://database.webservices.rajalakshmi.org/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/soap/"
targetNamespace="http://database.webservices.rajalakshmi.org/" name="AvailabilityService">
  <types>
    <xsd:schema>
      <xsd:import namespace="http://database.webservices.rajalakshmi.org/" schemaLocation="http://localhost:8090/AirlineServer/Availability?xsd=1"/>
    </xsd:schema>
  </types>
  <message name="dispAvail">
    <part name="parameters" element="tns:dispAvail"/>
  </message>
  <message name="dispAvailResponse">
    <part name="parameters" element="tns:dispAvailResponse"/>
  </message>
  <portType name="Availability">
    <operation name="dispAvail">
      <input wsam:Action="http://database.webservices.rajalakshmi.org/Availability/dispAvailRequest" message="tns:dispAvail"/>
      <output wsam:Action="http://database.webservices.rajalakshmi.org/Availability/dispAvailResponse" message="tns:dispAvailResponse"/>
    </operation>
  </portType>
  <binding name="AvailabilityPortBinding" type="tns:Availability">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <operation name="dispAvail">
      <soap:operation soapAction="">
        <input>
          <soap:body use="literal"/>
        </input>
        <output>
          <soap:body use="literal"/>
        </output>
      </operation>
    </binding>
  </binding>
  <service name="AvailabilityService">
    <port name="AvailabilityPort" binding="tns:AvailabilityPortBinding">
      <soap:address location="http://localhost:8090/AirlineServer/Availability"/>
    </port>
  </service>
</definitions>
```

start localhost:8090/Airline... localhost:8090/Airline... CS2358 - Internet Pr... Web Services 2015 C:\WINDOWS\sys... User Manuals Java - AirlineServer.js... 2:31 PM

CREATING AND CONSUMING A WEB SERVICE CLIENT

1. Having published the web service, we now create a client which communicates with the service and displays the result.
2. Select File → New → Dynamic Web Project.
3. Enter the Project name as AirlineClient.
4. Click Next.
5. Click Next.
6. Click Finish.
7. JAX-WS provides a tool called “wsimport” for generating the artifacts required for creating and consuming a web service. “wsimport” takes a wsdl file as input.
8. From the project folder “WSDClient” in command prompt or terminal, issue the following command:

```
wsimport -s . http://localhost:8090/AirlineServer/Availability?wsdl
```

```
C:\>cd CS2358
```

```
C:\CS2358>cd AirlineClient
```

```
C:\CS2358\AirlineClient>cd src
```

```
C:\CS2358\AirlineClient\src>wsimport -s .  
http://localhost:8090/AirlineServer/Availability?wsdl
```

```
parsing WSDL...
```

```
Generating code...
```

```
Compiling code...
```

```
C:\CS2358\AirlineClient\src>
```

(Note: Your Web Service needs to be running)

9. After refreshing your Eclipse workspace you should see the generated files.
10. Let us now create a HTML file to get the student roll no.
11. Right click on project → New → HTML File.
12. Enter the HTML file name as Availability.
13. Click Next.
14. Click Finish.

15. Enter the following code:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Availability</title>
</head>
<body>
<h1>Ticket Availability</h1>
<form method="post" action = "Availability.jsp">
Source :
<input type="text" name="source">
Destination :
<input type="text" name="destination">
<input type="submit" value="Check Availability">
</form>
</body>
</html>
```

16. Let us now create a JSP file to run this client.

17. Right click on project → New → JSP File.

18. Enter the JSP file name as Availability.

19. Click Next.

20. Click Finish.

21. Enter the following code:

```
<%@page import="org.rajalakshmi.webservices.database.AvailabilityService"%>
<%@page import="org.rajalakshmi.webservices.database.Availability"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Availability</title>
</head>
<body>
<%
    AvailabilityService availabilityService = new AvailabilityService();
    Availability availability = availabilityService.getAvailabilityPort();
    String result = availability.dispAvail(request.getParameter("source"),
        request.getParameter("destination"));
    out.println(result);
%>
</body>
</html>
```

22. Select the Availability.html file.

23. Right Click on the code window → Run As → Run on Server.

24. Enter the source and destination city.

25. You will get the output as table with availability status.

OUTPUT:



Ticket Availability

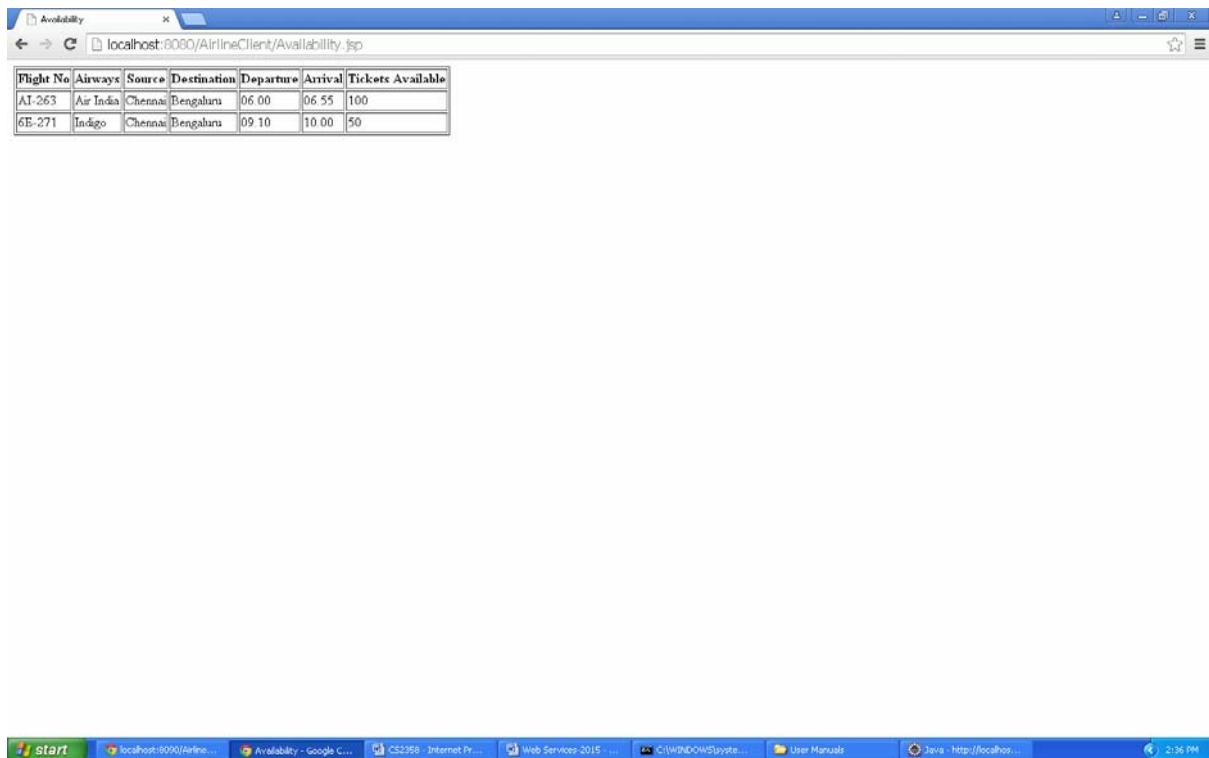
Source: Destination:



Ticket Availability

Source: Destination:





Flight No	Airways	Source	Destination	Departure	Arrival	Tickets Available
AI-263	Air India	Chennai	Bengaluru	06:00	06:55	100
6E-271	Indigo	Chennai	Bengaluru	09:10	10:00	50

EXERCISES

1. Take the case of a Travel Website and the two web services airline and travel agent. A normal operation between a user and the travel site would look like:
 1. User logs onto website
 2. Searches for flights on a particular day
 3. Selects the flight most suited
 4. Enters passenger data for the flight
 5. Makes a payment on a banking gateway
 6. On successful payment gets a confirmed PNR number.
 Develop the required program for invoking the web service.
2. Consider a case where we have two web services: an airline service and a travel agent. The travel agent is searching for an airline. Implement this scenario using web services and database. Document the functional requirements you are considering.
3. Hamen's Automobiles provides a facility through which customers can reserve date and time slots in advance and send their cars to service. Implement this scenario using web services and database. Document the functional requirements you are considering.



RAJALAKSHMI

ENGINEERING COLLEGE

Website : www.rajalakshmi.org

Elearning : www.rajalakshmiengg.com

***Give a man a fish and you feed him for a day.
Teach him how to fish and you feed him for a lifetime.***