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) *I* CSE *I* REC [bhuvaneswaran@rajalakshmi.edu.in](mailto:bhuvaneswaran@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 Æ New Æ 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 Æ New Æ Package.

9. Enter the package name as org.rajalakshmi.webservices.

10. Click Finish.

11. Right click on Package Æ New Æ 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 Æ New Æ Other Æ Web Services Æ Web service.

17. Click Next.

**BHUVANESWARAN B / AP (SS) / CSE / REC - 2**

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.

**BHUVANESWARAN B / AP (SS) / CSE / REC - 3**

**OUTPUT:**

O Java [[ http.//localhost.BOBOIS1mpleSOAP[Xc1mpleChent/sampleHelloWorldProxyfTestChent.jsp[chpse [~@~](mailto:~@~)



i=:..... ~;:\_·\_0.::...·......:~:.....·.....:~~·-0.::...·.....:~;;,,.;;:....::::....·......:~~-~l-"\_,;;,\_\_.;...,.:..:.,\_

..:......:....

..."."""'.'.•.. ,..·1 ~ (!i\_,.,au]

(o Project Explorer &l 8 ii, ;. v c:,, e (!) HeloWorki.}!va g Web Services Test Client ~ TMkList



lilldrSefvers

Gil~SinpleSOAPEx~

*~ tJ* lAX·WSWeb5eMces

r1J'il;,Oepbyme,to..ao«•"\_...,..

S~JavaResoi.rces

l\Iethods Inputs

*v ""* 11

AnOU:lineisnotavaW:i6e.

ta '9src

13 Elf oro.rajMakshmi.webserwes

Ct] I!) --·!av•

a\ Lb'arles

fil•Java~Resoi.rces

r.l 11& buld

fi Cil,WebCON.ert

r.l::. SimploSOAPEx..,;.o«,:

*Qfl* JSR.·109 Web SeNices

• getEndpoi:ntQ

• setEndpoi:nt

(1ava lang String)

• getHelloW orldQ sayHelloWorld (1ava lang Stnng)



name:~ ]

Markers Properties ~ Servers ~ Data 5ou'ce Explorer ' 5nppets C COl'lsole liJ 5, Tomcat v7.0 Server at loc~st (st,,,ted1Syndw-Ol'IZed]



O Java f ( http.//localhost.BOBOIS1mplcSOAP(XdmplcChcnt/s.unplctlclloWorldProxyfTcstCllcnjslp Icllpsc [[)@rgi](mailto:@rgi)

File Edit ~~e Search Proje,ct M W'ndow .....

~·O·~·

(CJ Project Explorer~

lil(d,Servers

Gil *'t1* SirnpleSOAPExatll)le

g WebSorvices TestCNent: ~

.."..'.°..'c.lfloc\_....,-.\_\_,\_\_seo. \_o APE\_.,.,., l•\_•\_•ip1e,1\_..•.\_.t.•1\_o\_-w\_-,·"""\_'-ox-=r

~·e·"'



======---'v ilo- 11 Anoutlinelsnotav~ .

*~*lil'i*!*b*I*OeJpAbXy·WmSe,W<«DJSe<efav<ic>es<o,,

Gil~JavaResol..l'ces

13"'src

Gil fB oro,ra~.wiebs«w:es

fil [lJ HeloWor1d.java

lil a\Lharies [ilmlJava5crttResou'ces [IJ II& buld

Iii til,WebCortent lil ~ SirpeSOAPEx~

lil *(a* JSR·109WebSer.ices

l\Iethods

• getEndpointQ

• setEndpoint

(java.lang.String)

• getHelloWorldQ sayHelloWorld (javalang.String)

Inputs

name: IBhlNan ~



!Result

Hello World from Bhuvan

Mairket-s Properties ffl5efvers ~ ~aSo.lceE..:pber ~ OConsoll

a

Ci! Tomcat v7.0 Server at loc-,t,ost (St~ted, Syndyoo:ed)



**BHUVANESWARAN B / AP (SS) / CSE / REC - 4**

**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 Æ New Æ Java Project.

4. Enter the Project name as CalcWS.

5. Click Next.

6. Click Finish.

7. Right click on Project Æ New Æ Package.

8. Enter the Package name as org.rajalakshmi.webservices.calc.

9. Click Finish.

10. Right click on Package Æ New Æ 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](mailto:@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](mailto:@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 m[ethods then we can use @WebMethod](mailto:@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 Æ New Æ 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",](http://localhost:8080/CalcWS/Calculator)

**new** Calculator());

}

}

22. Right Click on the code window Æ Run As Æ 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](http://localhost:8080/CalcWS/Calculator)

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:**

l

C [') loca rost 8080/QllcWS/Qllculator?wsdl



This XMLfile does not appear to have any style informationassociated with it. The document tree is shown below

T< '--

Publi!lhCd by JAX-VSRI at <http://jax-vs.dev.java.net>. RI's version is JAX-VSRI 2.2.1:-bOl.

e c ,

Generatl!d by JAX-VSRI at. [http://jax-vs.dev.java.nl!:t.RI'](http://jax-vs.dev.java.nl!:t.RI)s veeaaen r e JAX-VSRI 2.2.4-bOl.

-->

T <det init ions xmln!I: vsu•"hct.p :/ /docs. ceare-cpen. oro/ W!l!l/200"1:/0l/oasls-200401-wss-wssecur ity-util ity-1. 0. X!ld" xml~: vsp•"ht.t.p: *I I* vww. v3. org/ns/ws-pol lcy"

xmln:!1: wspl 2•"http: //schemes. xmlsoap .oro/vs/2001/09/policy" xmlns: wsam•"http: *I* /vvv. v3. oro/200?/0S/addi:e:,.slng/metadata" xmln!I: soap•"http: / /!lc:hernas. xmlsoap. oro/vsdl/soap/" xmlna: tns•-;;http: / /calc. vebservices. rajalakshmi. org/,. xmln!I: xsd•"http: *I I* vvv. v3. org/ 2001/XMLScheme." xmlns•"http: *I* /scheme.!I. xmlsoap. org/vsdl/"

targetNa.mespace•"http: //calc. veeeecvreee . rajalak!lhmi. erg/" name•"CalculatorService">

•<type!I>

• cxad: !lchema>

<x!ld.:impoz::-t name!lpace•"http ://eale. eeeeeevreee . z::-ajalakshrni. org/" !IChemaLoeat ion•"http: // loealhOst :8080/CaleWS/Caleulatoz::-?xsd•l *"I>*

</x!ld:schema>

</types>

•<message name•"add">

<paz::-tname•"paz::-arnetez::-s" element•'"tns:add"/>

</message>

• <message na.me•"addResponse">

<paz::-tname•"parameters" element•'"tns:add.Resporu,e"/>

</message>

•<message name•"!lub">

<part name•'"paraJMters'" element•'"tns: sub'"*I>*

</message>

•<message name•"subResporu,e'">

<part name•"parame:ters" element•'"tns:subRespOn!le"/>

</messsge>

•<portType narroe•NCt1.lculatoc-">

e ccpe z at.Lcn name•"add">

<input vsam:.let ion•"http: *I* /cale. veeeecvrcee , rajslakshrni. org/Calculstor/ add.Request" message•"trus: add'"/>

<output vsam: .lction•"ht;tp: //calc. vebservices. rsjslakshrni. erg/Calculator/ actctResponse" message•"tns: adct.Resporuse" />

</operation>

e ccpe r at.Lon name•"sub">

<input vsam: .let ion•"http: *I* /calc. veeeecvrcee , rajalaks.hmi. org/Calculator/subRequest" message•"tns: sub'"/>

<output vsam: .lction•"http: //calc. vebservices. rajalakshrni. erg/Calculator/ subResponse" message•"tns: subResponse"/>

</operation>

*<I* port Type>

• <binding name•"CalculatorPortBinding" type•"tns: Calculator">

<soap :binding transport•'"http: *I* /schemas. xmlsoap. erg/soap/http" style•"docwnent" />

e cc pexe t.t on name•"add">

<soe.p: operation soap.let ion•""/>

•<input>

<soap:body use•"literal"/>

</ input>

• <output>

<soap:body use•"literal"/>

</output>

*,/start* ~qlocahost808Q/Cal:W•• '4iMrP"=h.-es "'3webSerV1Ces·2015- r~SOAP·MoosdtWOtd rgSOAP~EC·WS·fi'O, •• 'OJava-Clll,cWS/srcJor .. rlt' Jt4·Pall"lt ~/'v ...... lli'I~ ... 12:03F'M

'Mlrbox·bh.lv.-.es:w. .... xf )locahost:8080/Cllt;WS/C.ek xJ .....,WebServlces )( ( ~ '!.'-Id\ X

C o localrost vffi/CalcWS/Calculator

Web Services

|  |  |
| --- | --- |
| IEndpoint | µntonnataon |
| Service Name: {httpl/calc.webservicesrajalak.shmt.org/)CalculatorService  Port Name (httpl/calc.webservices rajalak.shmi.org/)CalculatorPort  I | Address· httpJAocalhost:8080/CalcWSICalculator WSDL: httplJlocalh.ost:8080/CalcWS/Calcu!ator?wsdl lmplementabon class· org.rajalakshmiwebservices.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 Æ New Æ Java Project.

3. Enter the Project name as CalcWSClient.

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 “CalcWSClient” in command prompt or terminal, issue the following command:

wsim[port -s . http://localhost:8080/CalcWS/Calculator](http://localhost:8080/CalcWS/Calculator)?wsdl

C:\>cd CS2358

C:\CS2358>cd CalcWSClient

C:\CS2358\CalcWSClient>cd src

[C:\CS2358\CalcWSClient\src>wsimport -s . http://localhost:8080/CalcWS/Calculator](http://localhost:8080/CalcWS/Calculator)

?wsdl

parsing WSDL... Generating code... Compiling code...

C:\CS2358\CalcWSClient\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 Æ New Æ Package.

11. Enter the package name as org.rajalakshmi.webservices.calc.client.

12. Click Finish.

13. Right click on package Æ New Æ 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 Æ Run As Æ Java Application.

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

10 + 20 = 30

10 - 20 = -10

**BHUVANESWARAN B / AP (SS) / CSE / REC - 9**

**OUTPUT:**

10 + 20 = 30

10 - 20 = -10

**BHUVANESWARAN B / AP (SS) / CSE / REC - 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 Æ New Æ Java Project.

4. Enter the Project name as WSDB.

5. Click Next.

6. Click Finish.

7. Right click on Project Æ New Æ Package.

8. Enter the Package name as org.rajalakshmi.webservices.database.

9. Click Finish.

10. Right click on Package Æ New Æ 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.jws.WebService;

[@WebService](mailto:@WebService)

**public class** Student {

// **TODO** Auto-generated method stub

**public** String DispStud (String rollNo)

{

Connection conn = **null**; PreparedStatement ps = **null**; ResultSet rs = **null**;

**BHUVANESWARAN B / AP (SS) / CSE / REC - 11**

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())

{

}

**else**

{

}

sName = rs.getString(2);

sName = "Not Found...";

rs.close(); ps.close(); conn.close(); **return** sName;

}

**catch**(Exception e)

{

System.***out***.println(e);

**return** "Error..." ;

}

}

}

15. The [@WebService](mailto:@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 m[ethods then we can use @WebMethod](mailto:@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 Æ New Æ 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());](http://localhost:8090/WSDB/Student)

}

}

**BHUVANESWARAN B / AP (SS) / CSE / REC - 12**

22. Right Click on the code window Æ Run As Æ 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](http://localhost:8090/WSDB/Student)

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:**

l l

C o oca rost 80'Xl/WSDB/Student?wsd:



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

'I'<,

Publi:!!lhed by JAX-VS RI at [http://jax-vs.dev. java](http://jax-vs.dev.java). net. RI' !I version is JAX-VSRI 2 .2. i-bOl.

•< ,

Generatl!d by JAX-VSRI at <http://jax-vs.dev.java.nl!!:t>. RI'!I vecercn 1!1 JJ..X-WRSI 2.2.4-bOl.

-->

• <def lnl tions xmlns: vsu•"http: //docs. ceara-cpen. org/ vss/200i/01/oas re-aoo so i -eee-eeeeeue 1 ty-uti lity-1. 0. xsd" xmlns: vsp•"http: / /vvv. v3. org/ns/ws-pol1cy"

xrnlm,: wspl 2•"http: //schemas. xmlsoap .org/vs/200-1/09/ policy" xmlns: wsam•"http: *I I* wwv. v3. org/ 2007/05/addz:-essino/me:ut.data" xmlns: soap•"http: / /:!!IC:hernas. xmlsoap. org/vsdl/soap/"

xmln:!!1: tns•-;;http: *I* /database. vebservices. rajalabhmi. erg/" xmlns: xsd•''http: / /wwv. w3 . org/2001/ XllLSehema" x.mlrus•"http: *I* /schemas. x.mlsoap. org/wsdl/" targeeName:space•"http: //datab"e. vebserviees. rajalak:shmi. erg/" name:•"StudentServiee">

•<types>

• <xsd: schema>

<x.sd.: import namespaee•"http ://database. vebserviees. rajalakshmi. erg/" seheme.Loeation•"http: //local host: 8090/VSDB/Student ?xsd•l "/>

</xsd:sehema>

</types>

•<message name•"Disp5c.ud">

<part name•"paramet.ers" element.•"tns: DispStud"/>

</message>

•<message name•"Dil!lpSt.udResponse">

<part name•"parameeers" element•"tns: DispSt.udResponse"/>

</message>

•<port.Type name•"St.udent.">

•<operat.ion name•"D18pStud">

<input. vsam: let. ion•"ht.t.p: / /dat.abase. eeeeeevreee . rajalakshmi. org/St.udent/ Dil!lpSt.udReque!lt." messl!lQe•"t.ns: DispStud"/>

ccuepue vsam: lct.ion•"ht.t.p: //database. vebserviees. rajalakshmi. org/St.udent/DispSt.udResponse" messaoe•"tns: DispStudResponse" />

</operation>

</port.Type>

• <binding name•"St.udentPoC"t.81nd1ng" t.ype•"tns: St.udent">

<soap :binding transport•"http: //schemas. xmlsoap. oro/soap/http" style•"document *"I>*

v cope r e t aon name•"DispStud">

<soap: operation soap.lct1on•"" /> T<input>

<soap :body use•" literal"/>

*<I* input>

• <output>

<soap:body use•"literal"/>

</output>

</operation>

</binding>

e ceecv rce name•"StudentService">

•<port name•"StudentPort" b1nd1ng•"tn!!:StudentPortBinding">

<eoep : e.ddress loeation•"http:// loealhost:B090/VSDB/Student"/>

</port>

</service>

</definitions>

,/Start, r ~ WS-Oe• Maosoft WOid r ~ WebS.w:es-2015 ·... *r* O Java EE· WSDefsrcf *r:,* lccebcst 8090(WSDS ~-'J~ ~ 8 47PM

J x

D WebS.'lices \ **eneeee**

*e* D localrost0090/WSDB/Student

Web Services

|  |  |
| --- | --- |
| IEndpoint | ~fonnataon |
| Service Name: [{http://database.websernces](http://database.websernces) raJalakshmLor21} StudentService  Port Name [{http://database.webservices](http://database.webservices) rajalak.shmi.or2/)StudentPort | Address httplnocalhost8090/WSDB/Stud<nt WSDL hnp/Jlocaihost:8090/WSDB/Student?wsdl Implementation class: org.raJalak.shmi.webservices.database.Student |
|  |  |

~Sfa\_,.rff *r~ws-re* MICfosoftWotd r~Web5erwes·2015-... rolavaEE•WSOB(src/. QWebSerVKes *GC1f:4 f<\_..•~i:-\~* l48PM

**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 Æ New Æ 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:

wsim[port -s . http://localhost:8090/WSDB/Student?wsdl](http://localhost:8090/WSDB/Student)

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...](http://localhost:8090/WSDB/Student)

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 Æ New Æ 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 Æ New Æ 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">](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 Æ Run As Æ Run on Server.

24. Enter the student roll no.

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

**BHUVANESWARAN B / AP (SS) / CSE / REC - 16**

**OUTPUT:**

['}S<oden<Detok

- !. - di x

+- C [') localrost 8080JW$08Client/StuJent.html

[201201001 '~

*,J start* ~WS-te-HICtosoftWord QwebSenolces-2015· OJ.avaEE·http/{'10<.al *"I* tu,:J...r,t[.-11,I ••• ~ 111- • 850PM

['}S<«Jent

- ;.. - 61 x

+- C D localrost~JW~S~DB~C~lie~nt/~St~u:O~is~p.j~sp~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-

Anm

*<Ci \_*

:: start ~WHe Moosoftw«d ~Web5erwes·2015· QJavaEE http/fr.Jul *"I* t.J0:1,,.rit *»c.. ·"}~* r~ ,'< 1111- • 850PM

**BHUVANESWARAN B / AP (SS) / CSE / REC - 17**

**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)

**BHUVANESWARAN B / AP (SS) / CSE / REC - 18**

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 Æ New Æ Java Project.

4. Enter the Project name as AirlineServer.

5. Click Next.

6. Click Finish.

7. Right click on Project Æ New Æ Package.

8. Enter the Package name as org.rajalakshmi.webservices.database.

9. Click Finish.

10. Right click on Package Æ New Æ 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;

**BHUVANESWARAN B / AP (SS) / CSE / REC - 19**

[@WebService](mailto:@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) + "</th>"; str = str + "<td>" + rs.getString(2) + "</th>"; str = str + "<td>" + rs.getString(3) + "</th>"; str = str + "<td>" + rs.getString(4) + "</th>"; str = str + "<td>" + rs.getString(5) + "</th>"; str = str + "<td>" + rs.getString(6) + "</th>"; str = str + "<td>" + rs.getString(7) + "</th>"; str = str + "</tr>";

}

str = str + "</table>";

rs.close(); ps.close(); conn.close(); **return** str;

}

**catch**(Exception e)

{

**return** (e.toString()) ;

}

}

}

15. The [@WebService](mailto:@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 m[ethods then we can use @WebMethod](mailto:@WebMethod) annotation before the method signature.

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

**BHUVANESWARAN B / AP (SS) / CSE / REC - 20**

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 Æ New Æ 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",](http://localhost:8090/AirlineServer/Availability)

**new** Availability());

}

}

22. Right Click on the code window Æ Run As Æ 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](http://localhost:8090/AirlineServer/Availability)

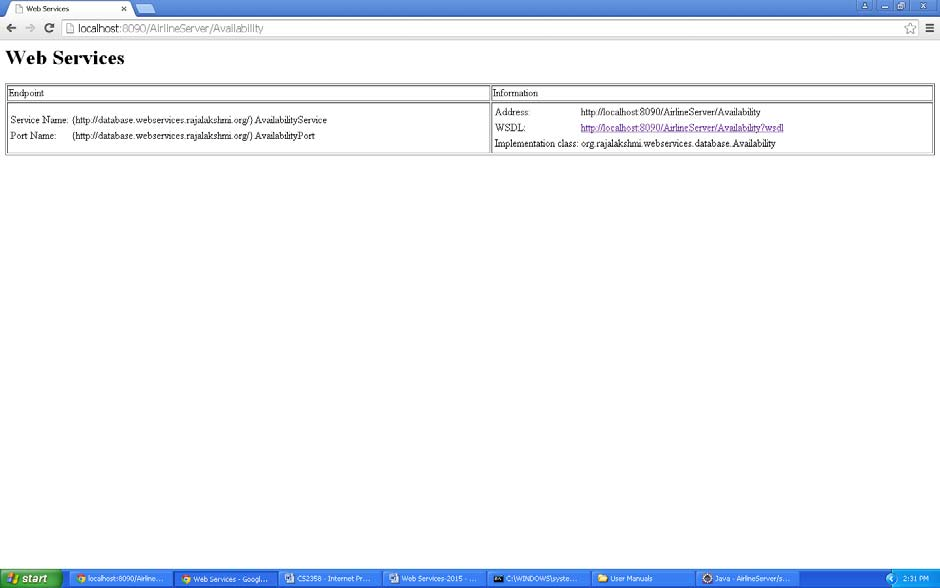
25. When you run the application, the Java SE 6 platform has a small web application server that will publish the web service at the address

<http://localhost:8090/AirlineServer/Availability>

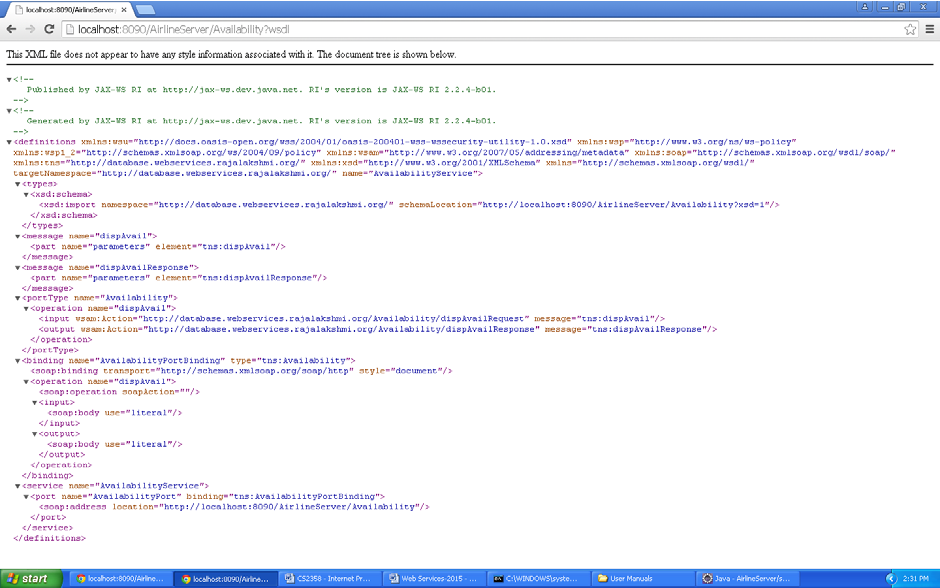
while the JVM is running. If you see a large amount of XML that describes the functionality behind the web service, then the deployment is successful.

**BHUVANESWARAN B / AP (SS) / CSE / REC - 21**

**OUTPUT:**



**BHUVANESWARAN B / AP (SS) / CSE / REC - 22**



**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 Æ New Æ 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 “WSDBClient” in command prompt or terminal, issue the following command:

wsim[port -s . http://localhost:8090/AirlineServer/Availability](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](http://localhost:8090/AirlineServer/Availability)

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 Æ New Æ HTML File.

12. Enter the HTML file name as Availability.

13. Click Next.

14. Click Finish.

**BHUVANESWARAN B / AP (SS) / CSE / REC - 23**

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 Æ New Æ JSP File.

18. Enter the JSP file name as Availability.

19. Click Next.

20. Click Finish.

21. Enter the following code:

[<%@page](mailto:%@page) import=*"org.rajalakshmi.webservices.database.AvailabilityService"*%>

[<%@page](mailto:%@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">](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 Æ Run As Æ Run on Server.

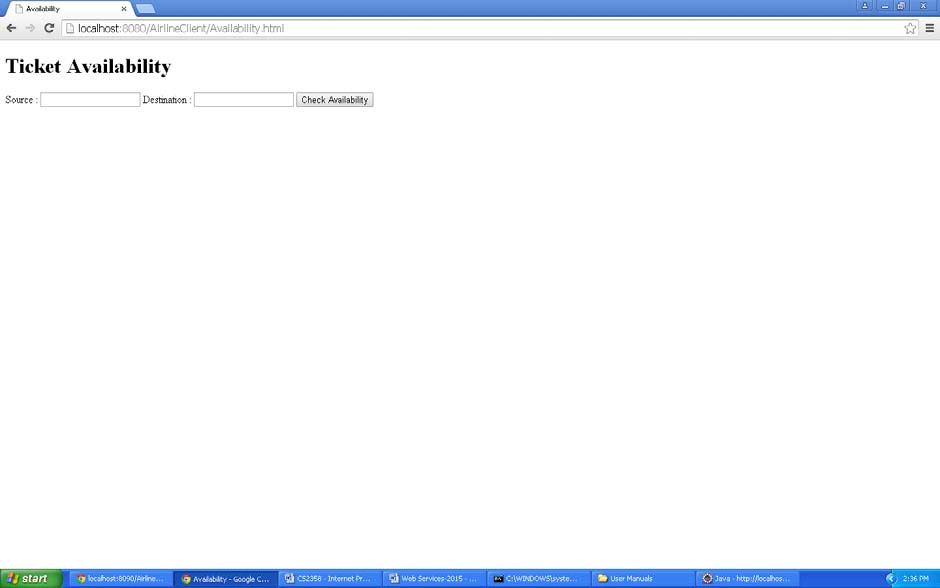
**BHUVANESWARAN B / AP (SS) / CSE / REC - 24**

24. Enter the source and destination city.

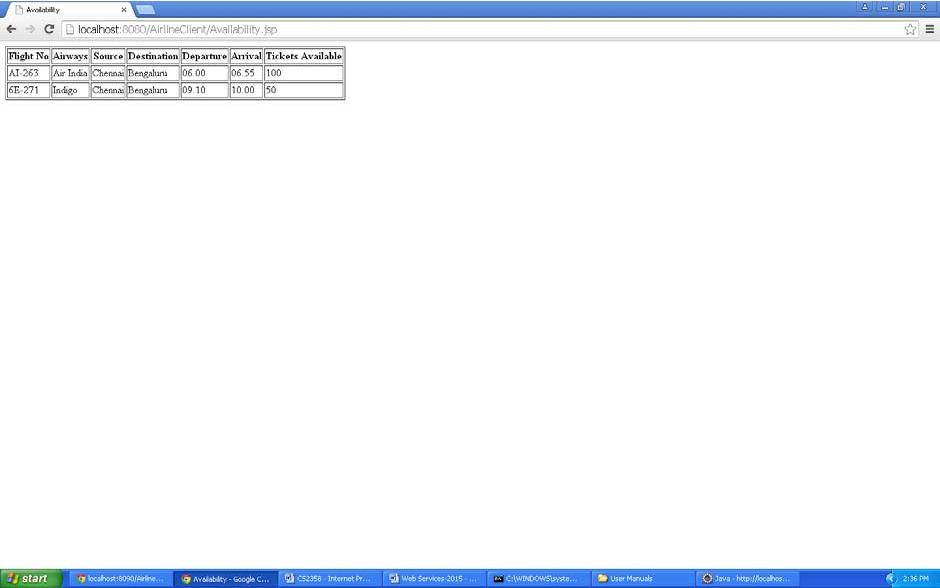
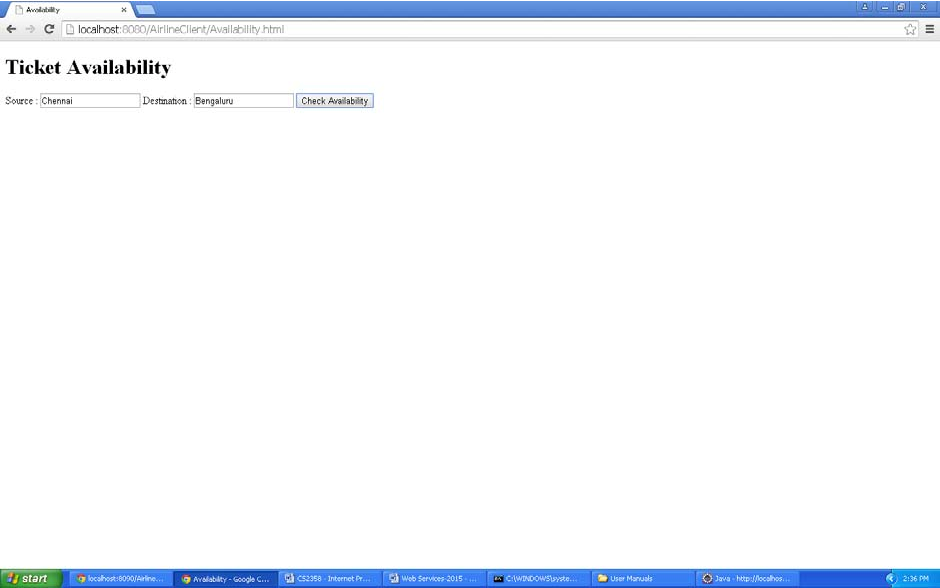
25. You will get the output as table with availability status.

**BHUVANESWARAN B / AP (SS) / CSE / REC - 25**

**OUTPUT:**



**BHUVANESWARAN B / AP (SS) / CSE / REC - 26**



**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.

**BHUVANESWARAN B / AP (SS) / CSE / REC - 27**



**BHUVANESWARAN B / AP (SS) / CSE / REC - 28**