

Velammal College of Engineering and Technology, Madurai-09.

Department of Computer Science and Engineering



Regulation 2013

B.E CSE - IV YEAR - VII SEMESTER

**CS6712 – Grid and Cloud Computing Lab
Manual**



ANNA UNIVERSITY, CHENNAI-600025

Table of Contents

Sl.No.	Description	Page No.
1.	Objective and outcome of the course	3
2.	Syllabus of the course	4
3.	Pre requisite of the course	5
4.	Requirements	6
5.	Instructions for Assessment	7
6.	Ex.No.1 : Calculator Web service	8
7.	Ex.No.2 : OGSA Compliant Web Service	14
8.	Ex.No.3 : Grid Service using Apache	36
9.	Ex.No.4 : Applications on Java or C/C++ Grid APIs	48
10.	Ex.No.5 : Secure Web Service	52
11.	Ex.No.6 : Running Virtual Machine of Different Configuration	68
12.	Ex.No.7: Virtual Machine to Virtual Box	71
13.	Ex.No.8 : Installing C Compiler on Virtual Machine	74
14.	Ex.No.9 : Virtual Machine Migration	77
15.	Ex.No.10 : Installing Storage Controller	82
16.	Ex.No.11 : Hadoop- WordCount using Map and Reduce	85
17	Ex.No.12: Mounting a Node in Hadoop using FUSE	92

Objective and Outcome of the Course

The student should be made to:

- Be exposed to tool kits for grid and cloud environment.
- Be familiar with developing web services/Applications in grid framework
- Learn to run virtual machines of different configuration.
- Learn to use Hadoop

Upon successful completion of this course, students will be able to:

- C01:** Develop web services and portlet application [K3]
- C02:** Build applications using REST API [K3]
- C03:** Develop application with security mechanisms [K3]
- C04:** Demonstrate the virtualization concepts in cloud environment [K3]
- C05:** Implement a single node cluster environment and map reduce concept in Hadoop framework [K3]

Syllabus

GRID COMPUTING LAB:

1. Develop a new Web Service for Calculator.
2. Develop new OGSA-compliant Web Service.
3. Using Apache Axis develop a Grid Service.
4. Develop applications using Java or C/C++ Grid APIs
5. Develop secured applications using basic security mechanisms available in Globus Toolkit.
6. Develop a Grid portal, where user can submit a job and get the result. Implement it with and without GRAM concept.

CLOUD COMPUTING LAB:

1. Find procedure to run the virtual machine of different configuration. Check how many virtual machines can be utilized at particular time.
2. Find procedure to attach virtual block to the virtual machine and check whether it holds the data even after the release of the virtual machine.
3. Install a C compiler in the virtual machine and execute a sample program.
4. Show the virtual machine migration based on the certain condition from one node to the other.
5. Find procedure to install storage controller and interact with it.
6. Find procedure to set up the one node Hadoop cluster.
7. Mount the one node Hadoop cluster using FUSE.
8. Write a program to use the API's of Hadoop to interact with it.
9. Write a word count program to demonstrate the use of Map and Reduce tasks.

Prerequisite of the course

- Core and Advanced Java Programming
- Web Services

Requirements

- Netbeans IDE 8.2
- Cloud Simulator 3.0.3

Instructions for Assessment

- The manual contains the code for all the model exercises that has been prepared and executed on Netbeans IDE 8.0, Cloud Simulator and appropriate supporting softwares and frameworks.
- All the laboratory exercises should be implemented on platform mentioned by the Instructor. The change of platform for implementation is strictly banned.
- All the students are instructed to maintain a hard copy of the manual, separate notebook and file folder for this course.
- When you come to lab, you have to prepare the pseudo code for your exercises on spot and then implement the same on above mentioned platform. The pseudo code must be prepared on the note book and all the observations of your implementation must be recorded in the note book, once you finished your exercises. At the end of the laboratory classes, the note book must be signed by the faculty in charge and assessment will be done.
- When you are coming for the laboratory classes, the completed record sheets for the previous class exercises must be produced for signature of faculty in charge except for first class of the course.
- All the exercises need to be completed in the respective classes itself. The deadline will not be extended at any circumstances. All the assessment will be done in the same class itself.
- You are encouraged to do all the necessary preparation before you come to the lab. Treat every lab class as lab examination.
- Every exercise will be assessed for maximum of 25 marks.

1. DEVELOP A NEW WEB SERVICE FOR CALCULATOR

AIM

To develop a web service for calculator in java using Netbeans IDE

SOFTWARE USED

- Netbeans IDE 8.2
- JDK 8.1
- Glassfish Web Server

PROCEDURE

Step 1: Create a Java Web Project

- Open Netbeans IDE 8.2
- Click on New Project and choose Java Web -> Web Application
- Enter the Project Name: CalculatorWS, using the default settings and then click on “Finish”. Now the Project has been created.

Step 2: Create a Web Service

- Now go to the Project Tree Structure on the left side of the window.
- Right click on the project and select “New” and then choose “Web Service”
- Specify web service name “CalWS” and package name “CalculatorWS”. Click on “Finish”.
- Open CalWS.java file, replace the original hello() function with the following code:

```
@WebMethod(operationName = "add")
public String add(@WebParam(name = "value1") String
value1,@WebParam(name="value2") String value2 ) {
    float value=Float.valueOf(value1)+Float.valueOf(value2);
    return (Float.toString(value));
}
```

Similarly write the code for subtraction, multiplication, division and other calculator operations. Refer the program section for full program.

- Now the web service is created.

Step 3: Deploy and Test Web Service

- Right click on the project and select “Deploy”
- This is to deploy all the web services in this project. If success, you will see:

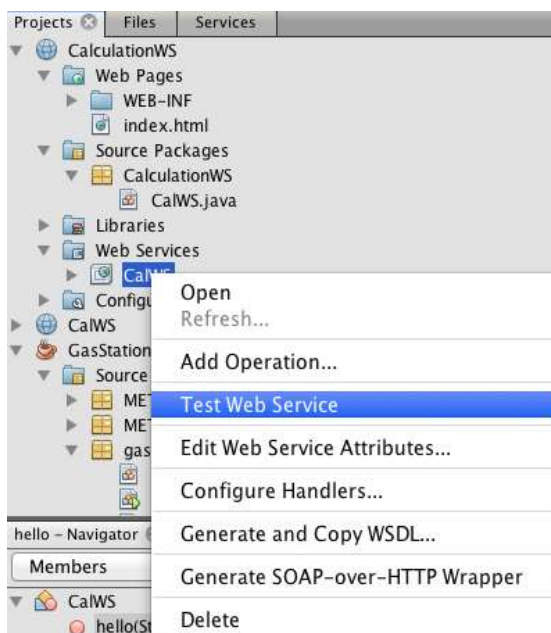
Grid and Cloud Computing Lab-Manual

```
23  @WebMethod(operationName = "Addition")
25  public String Addition(@WebParam(name = "value1") String value1,@WebParam(name = "value2") String value2 ) {
26      float value=Float.valueOf(value1)+Float.valueOf(value2);
27      return (Float.toString(value));
}
```



The screenshot shows the NetBeans IDE with the source code for the Addition web service method. The Output window below shows the build process, including the compilation of the web service and the successful deployment of the web service to the GlassFish Server 4. The build is successful and took 1 second.

- To test the web service, right click on the service and select “Test Web Service”



You will see:



Right Click on the project and select “Clean and Build”, a war file will be automatically generated under “dist” sub-directory.

PROGRAM (Use the below given program for CalWS.java)

package CalculatorWS;

```
import javax.jws.WebService;

import javax.jws.WebMethod;

import javax.jws.WebParam;

/**
 *
 * @author poonkuntran
 */

@WebService(serviceName = "CalWS")

public class CalWS {

    @WebMethod(operationName = "add")

    public String add(@WebParam(name = "value1") String
value1,@WebParam(name="value2") String value2 ) {

        float value=Float.valueOf(value1)+Float.valueOf(value2);

        return (Float.toString(value));

    }

    @WebMethod(operationName= "sub")

    public String sub(@WebParam(name= "value1")String
value1,@WebParam(name="value2")String value2){

        float value=Float.valueOf(value1)-Float.valueOf(value2);

        return (Float.toString(value));

    }

    @WebMethod(operationName= "mul")

    public String mul(@WebParam(name= "value1")String
value1,@WebParam(name="value2")String value2){

        float value=Float.valueOf(value1)*Float.valueOf(value2);

        return (Float.toString(value));

    }

    @WebMethod(operationName= "div")
```

```
public String div(@WebParam(name="value1")String value1,@WebParam(name="value2")String value2){
float value=Float.valueOf(value1)/Float.valueOf(value2);
return (Float.toString(value));
}
```

```
@WebMethod(operationName="mod")
public String mod(@WebParam(name="value1")String value1,@WebParam(name="value2")String value2){
float value=Float.valueOf(value1)%Float.valueOf(value2);
return (Float.toString(value));
}
```

```
@WebMethod(operationName="cube")
public String cube(@WebParam(name="value1")String value1){
float value=Float.valueOf(value1)*Float.valueOf(value1)*Float.valueOf(value1);
return (Float.toString(value));
}
```

```
@WebMethod(operationName="square")
public String square(@WebParam(name="value1")String value1){
float value=Float.valueOf(value1)*Float.valueOf(value1);
return (Float.toString(value));
}
```

```
@WebMethod(operationName="sin")
public String sin(@WebParam(name="value1")String value1){
double value=Math.sin(0);
return (Double.toString(value));
}
```

```
@WebMethod(operationName="cos")
public String cos(@WebParam(name="value1")String value1){
```

Grid and Cloud Computing Lab-Manual

```
double value=Double.valueOf(Math.cos(0));  
return (Double.toString(value));  
}  
  
@WebMethod(operationName= "sqrot")  
public String sqrot(@WebParam(name= "value1")String value1){  
double value=Math.sqrt(4);  
return (Double.toString(value));  
}
```

OUTPUT

```
public abstract java.lang.String calculator.Cal.add(java.lang.String java.lang.String)
```

```
add (  ,  )
```

```
public abstract java.lang.String calculator.Cal.sin(java.lang.String)
```

```
sin (  )
```

```
public abstract java.lang.String calculator.Cal.cos(java.lang.String)
```

```
cos (  )
```

```
public abstract java.lang.String calculator.Cal.sub(java.lang.String java.lang.String)
```

```
sub (  ,  )
```

```
public abstract java.lang.String calculator.Cal.mod(java.lang.String java.lang.String)
```

```
mod (  ,  )
```

```
public abstract java.lang.String calculator.Cal.square(java.lang.String)
```

```
square (  )
```

```
public abstract java.lang.String calculator.Cal.sqrot(java.lang.String)
```

```
sqrot (  )
```

```
public abstract java.lang.String calculator.Cal.cube(java.lang.String)
```

```
cube (  )
```

```
public abstract java.lang.String calculator.Cal.mul(java.lang.String java.lang.String)
```

```
mul (  ,  )
```

Method parameter(s)

Type	Value
java.lang.String	0

Method returned

java.lang.String : "1.0"

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:cos xmlns:ns2="http://calculator/">
      <value1>0</value1>
    </ns2:cos>
  </S:Body>
</S:Envelope>
```

CONCLUSIONS

The web service has been created for calculator operations and it is deployed and tested in Java Web Application using Netbeans IDE.

2. DEVELOP A OGSA COMPLIANT WEBSERVICE

Aim

To develop new open grid service architecture- complaint web service in java using NetBeans IDE 8.2.

Procedure

To solve the following tasks have to be performed:

1. Use the netbeans
2. Make a stud table in the jdbc:derby database
3. Create a Web project
4. Develop a Web Service program for student
5. Create a exam operation in the web service
6. Add enterprise resources for the database
7. Edit source code
8. Build & deploy the project
9. Test the web service **Derby database**

This is a small database software bundled with Netbeans in glassfish server. It is easy to create a database, create a table, make the queries over it. User can connect easily the enterprise program, web service program and web program with derby database software.

Connecting database

- Open the Netbeans software
- Navigate to the services tab
- Open database folder and select the derby database
- Right Click and select start the server as given below in figure 1

Grid and Cloud Computing Lab-Manual

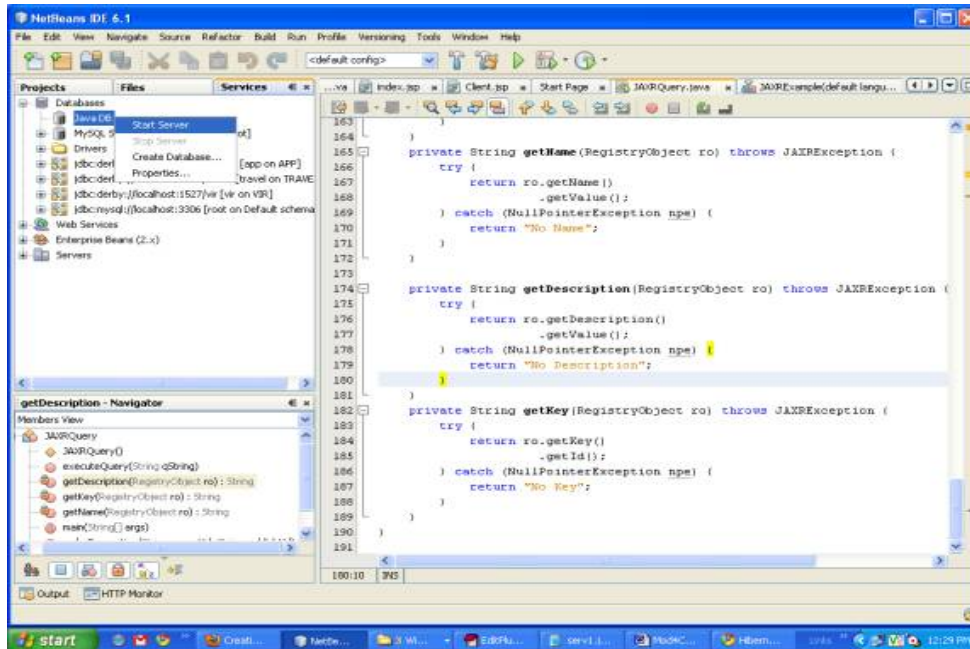


Figure 1

- Select the sample database or you can create one
- Right Click and Select Connect as given below in Figure 2.
- It will connect to the sample database

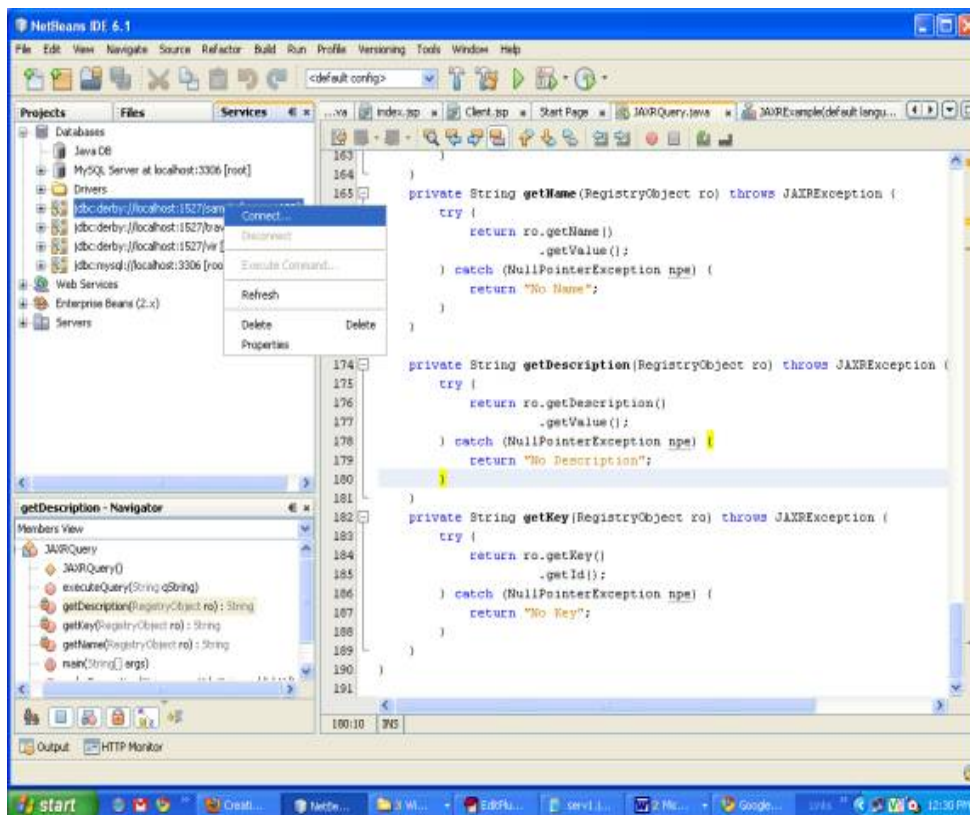


Figure 2

- Right Click and select Create table as given
- It will Open a dialog box as shown below in Figure 3

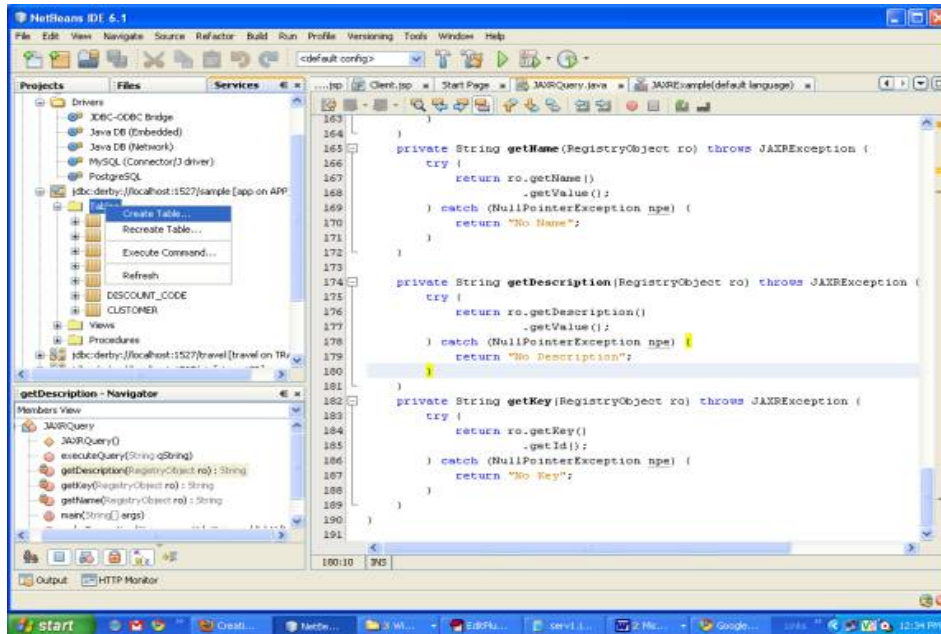


Figure 3

- In this dialog box give the table name as stud
- Now add the columns
- Add the two columns
- Roll with data type numeric and Name with data type char
- Do all steps as shown below in Figure 4.
- It create a stud table a sample database

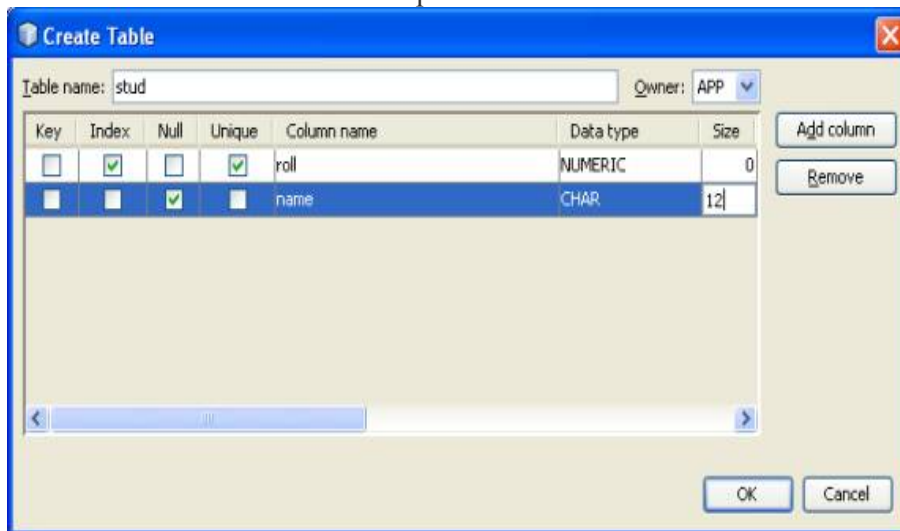


Figure. 4

Creating Web Service project

- Create a new web project
- Give a name as jaax2
- Select the server as glassfish and java EE web 6 version

Grid and Cloud Computing Lab-Manual

- Click on the finish as shown below in Figure 5

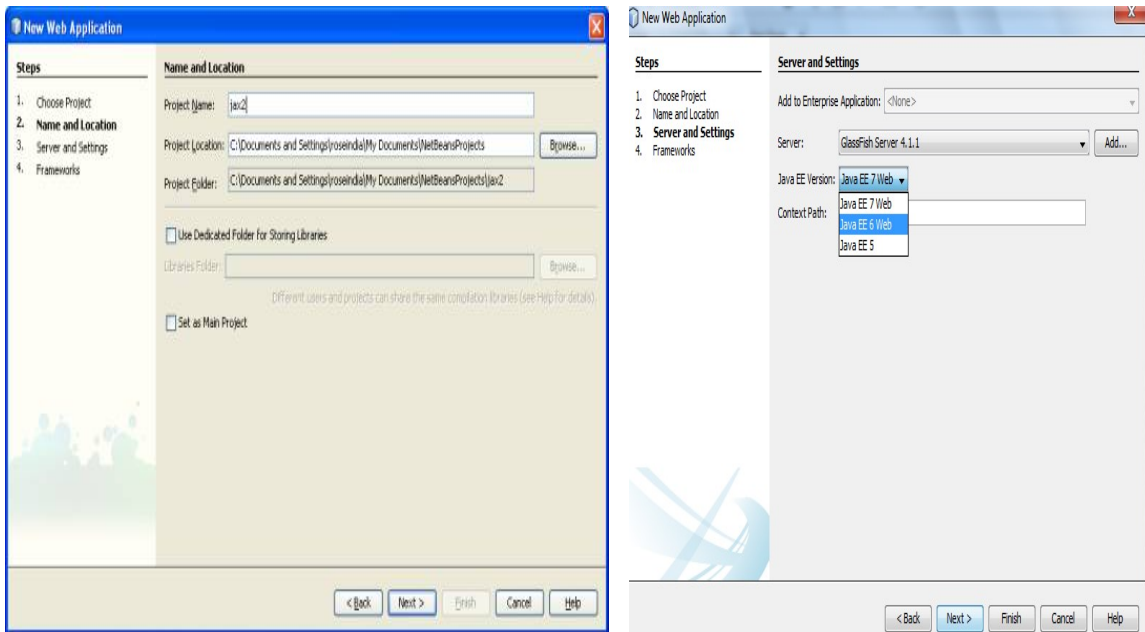


Figure 5

Create a web service

- Right Click on the project
- Select New->Web Service
- Type name as stud
- Type package as pack1 as shown below in Figure 6

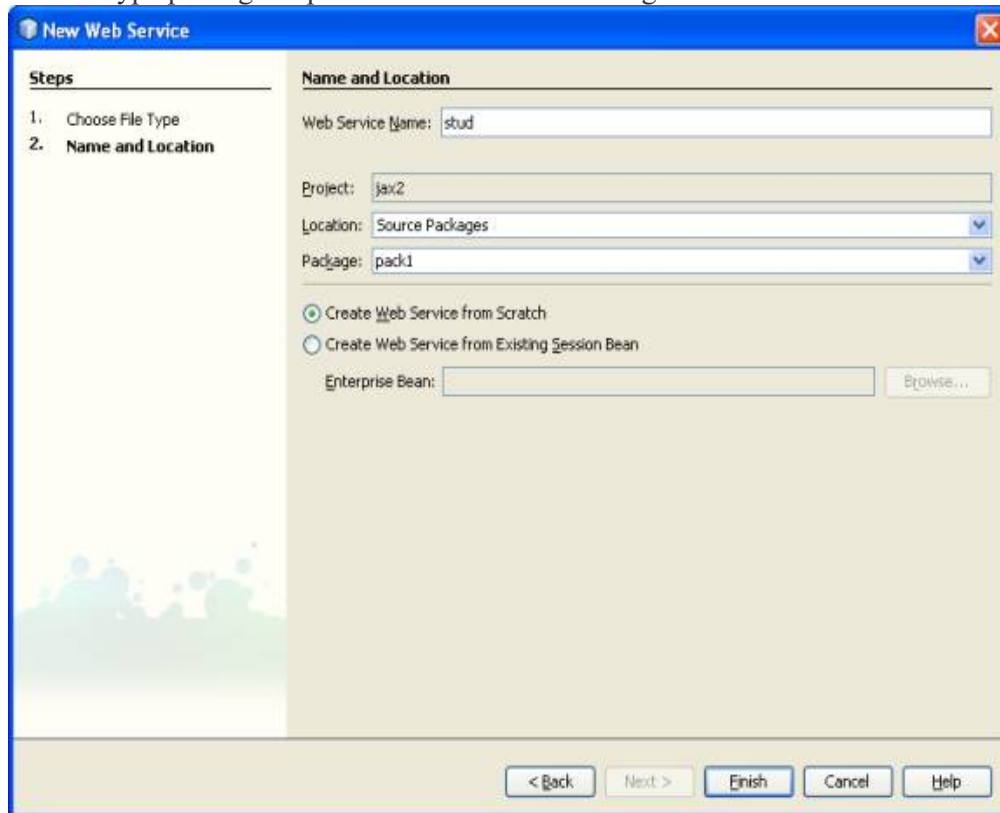


Figure 6

- This will create a stud Web Service class

Adding operation

- In the design view Click on the Add Operation
- It will generate a Add operation Dialog Box as shown below n Figure. 7

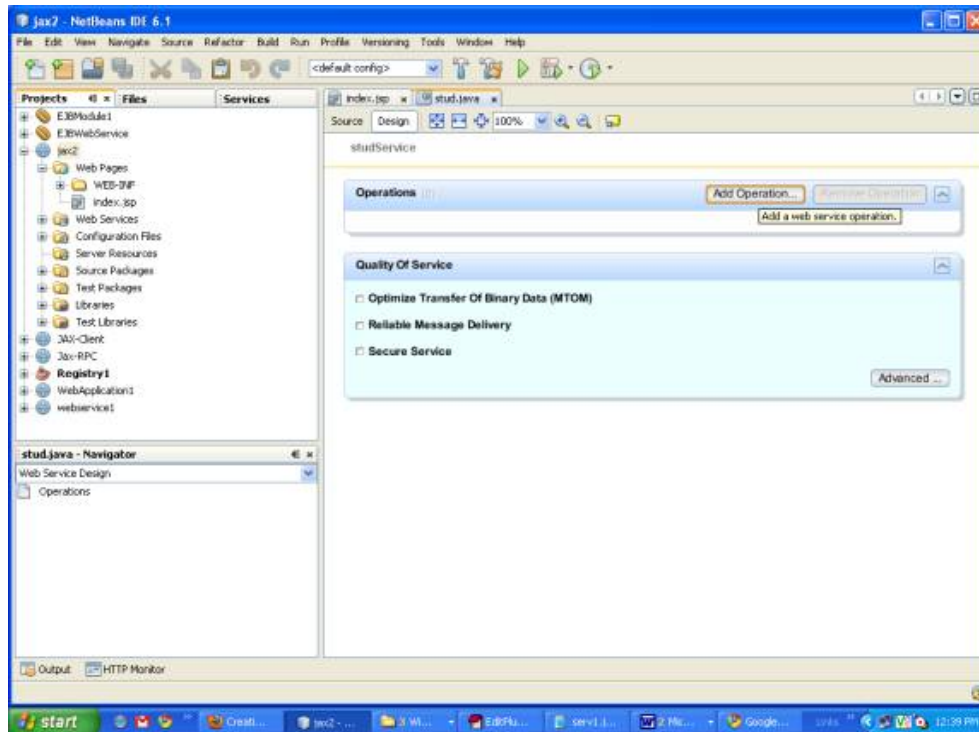


Figure. 7

- In the pop up dialog box give the operation name and parameters
 - Operation name exam
 - Return type String
 - Parameters name roll and name
- Follow the steps according to Fig 8 given below

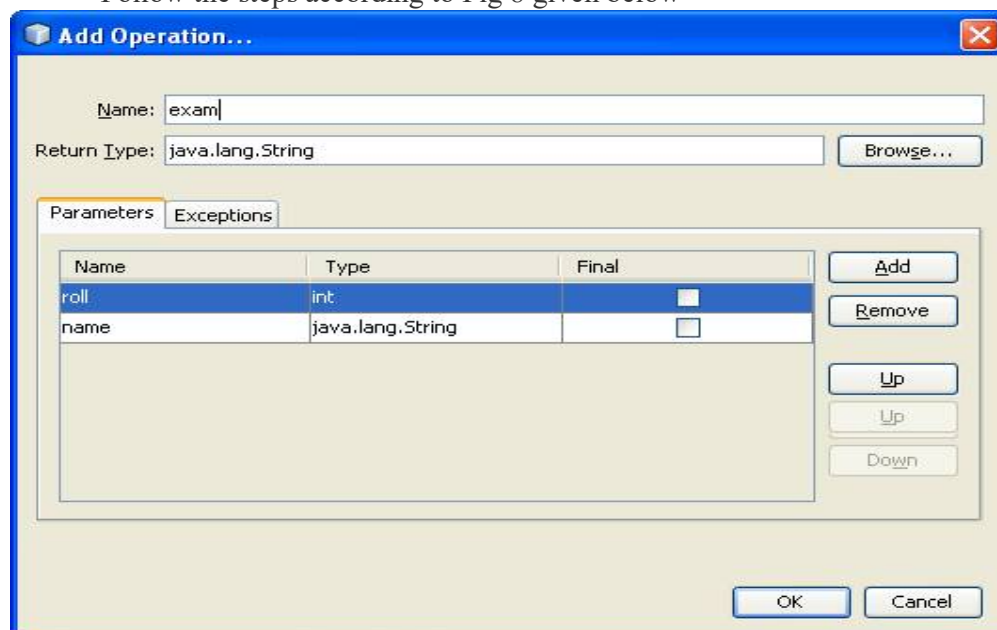


Figure. 8

Grid and Cloud Computing Lab-Manual

Now click source->It creates a full web service class as shown below

```
package pack1;

import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;

@WebService()
public class stud {

    @WebMethod(operationName = "exam")
    public String exam(@WebParam(name = "roll")
    int roll, @WebParam(name = "name")
    String name) {
        //TODO write your implementation code here:
        return null;
    }
}
```

Adding Database capabilities

- Database and table created above is used in the web service
- Right Click in the code section of web service and click Insert Code(fig 9)->list display as in figure 9.1 and click use database in Figure 9.1

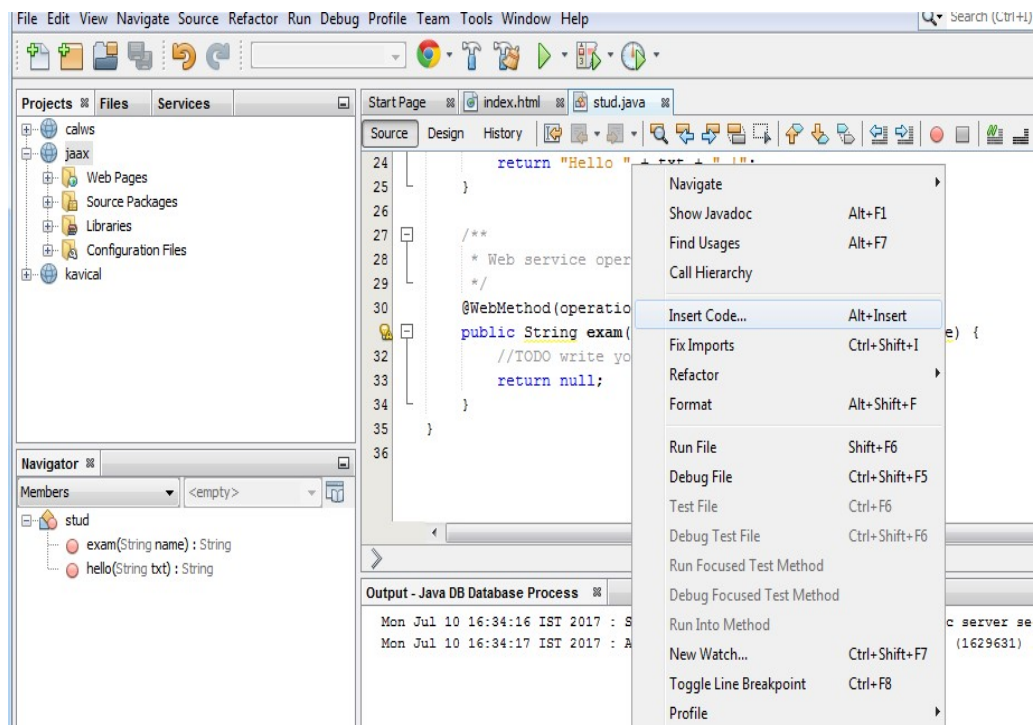


Figure 9

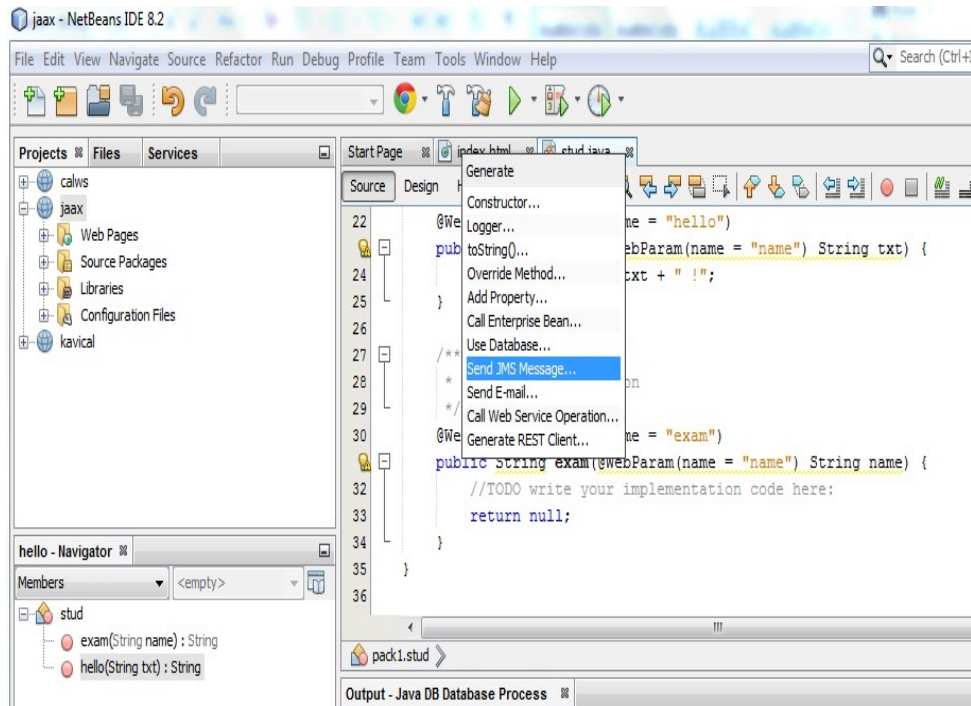


Figure 9.1

- Fig 9.2 displayed and then click Add
- Add the data source reference
- Type Reference name data1
- Select jdbc/sample Server Data Source as shown below in Fig 10.

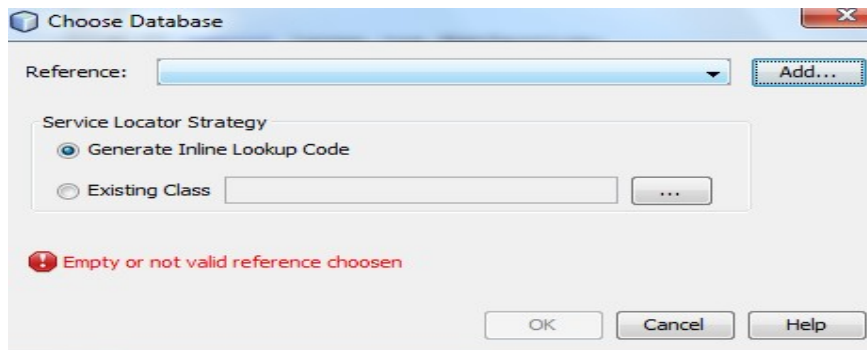


Figure 9.2

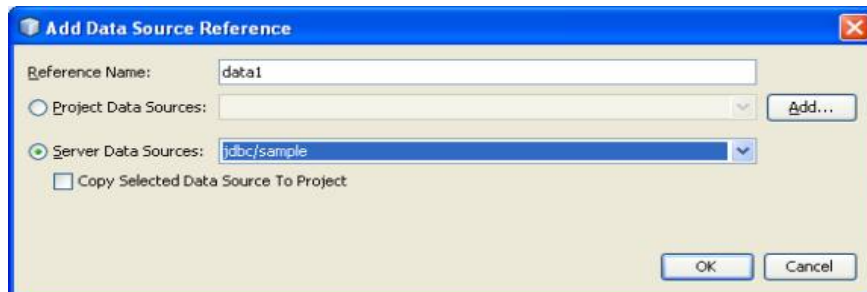


Figure. 10

- Click on Ok as shown below in Fig. 11.

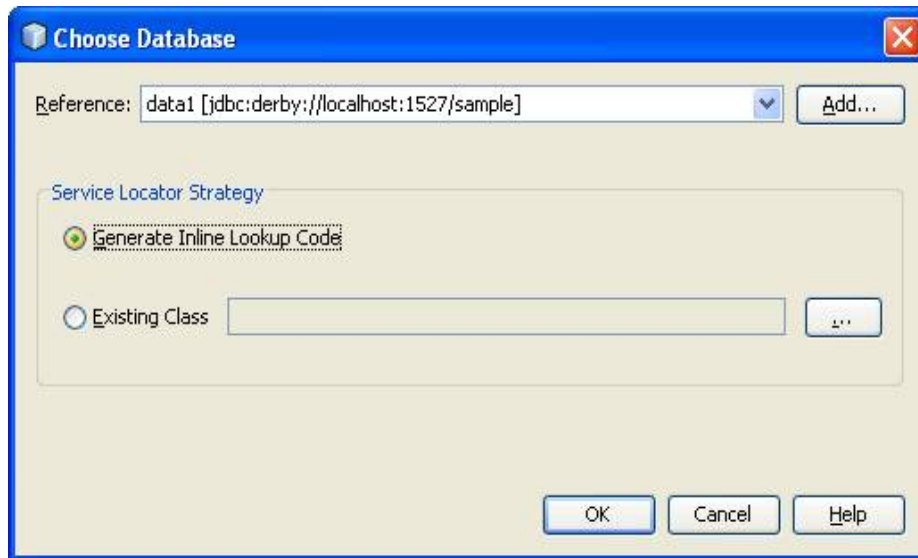


Figure. 11

- This creates Data Source reference variable data1 at the top of method
- Now make changes in the code for database connection as shown in code package pack1;

```
import java.sql.Connection;           --→ Add this statement
import java.sql.PreparedStatement;    --→ Add this statement
```

```
import javax.annotation.Resource;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
import javax.sql.DataSource;
```

```
@WebService()
public class stud {
    @Resource(name = "data1")
    private DataSource data1;
```

```
    @WebMethod(operationName = "exam")
    public String exam(@WebParam(name = "roll")
        int roll, @WebParam(name = "name")String name) {
        String status="record not inserted";
        try {
            Connection con=data1.getConnection();
            PreparedStatement ps=con.prepareStatement("INSERT INTO stud VALUES(?,?)");
            ps.setInt(1,roll);
            ps.setString(2,name);
            int i=ps.executeUpdate();
            if(i!=0) {
                status="record inserted";
            }
        }
        catch(Exception e){
            System.out.println("error in strong data"+e);
        }
    }
}
```

```
return status;  
}  
}
```

Running the project

- Build the above created project
- Deploy the project on the server as shown below in Figure. 12
- This deploys the project on the server
- We can now run our Web Service

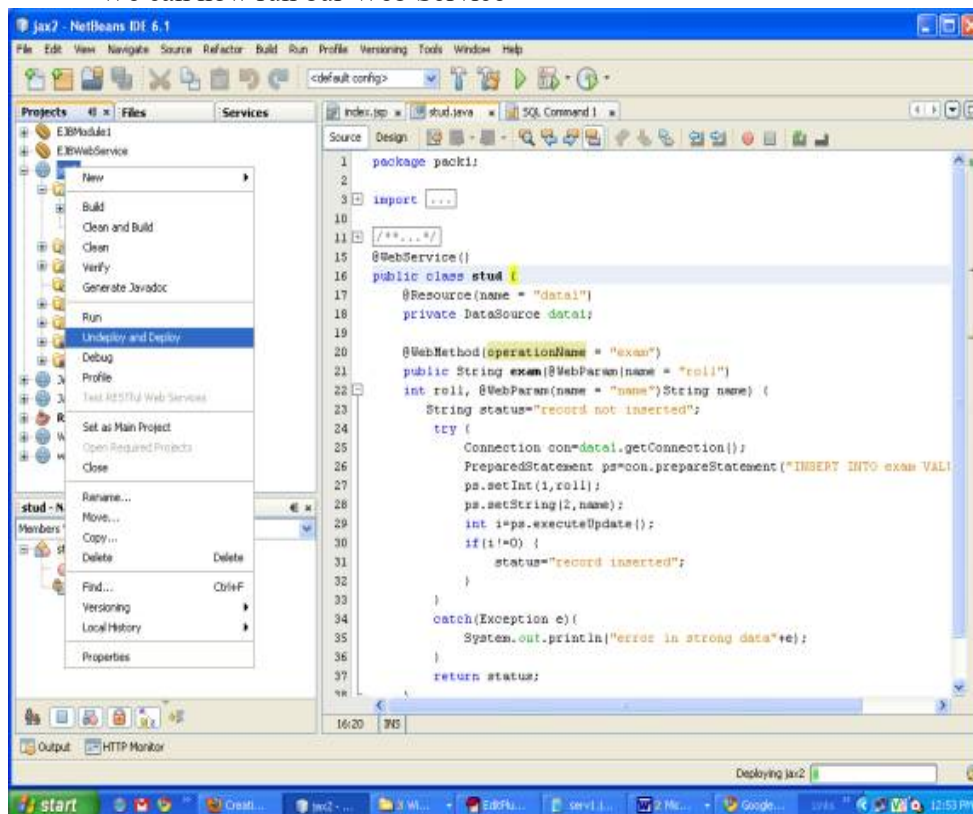


Figure. 12

- Right Click on Web Service stud
- Select Test Web Service as shown below in Figure. 13

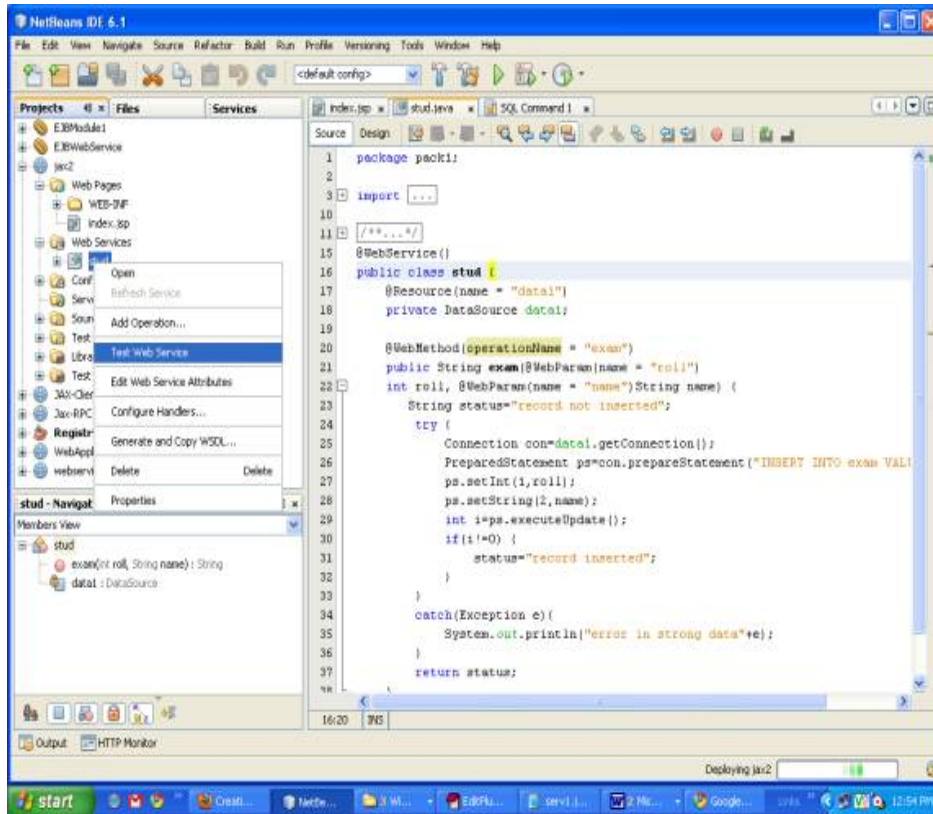


Figure 13

- It open Web Service in the browser
- Give the 1 and jack in text boxes
- Text boxes are actually arguments of web method
- Click on exam button as shown below in Fig. 14



Figure.

14

- Exam Buttons are actually method name of the Web Service
- This will result the value with SOAP request and SOAP response, As shown below in Figure 15.
- It will insert the values in the Derby Database table

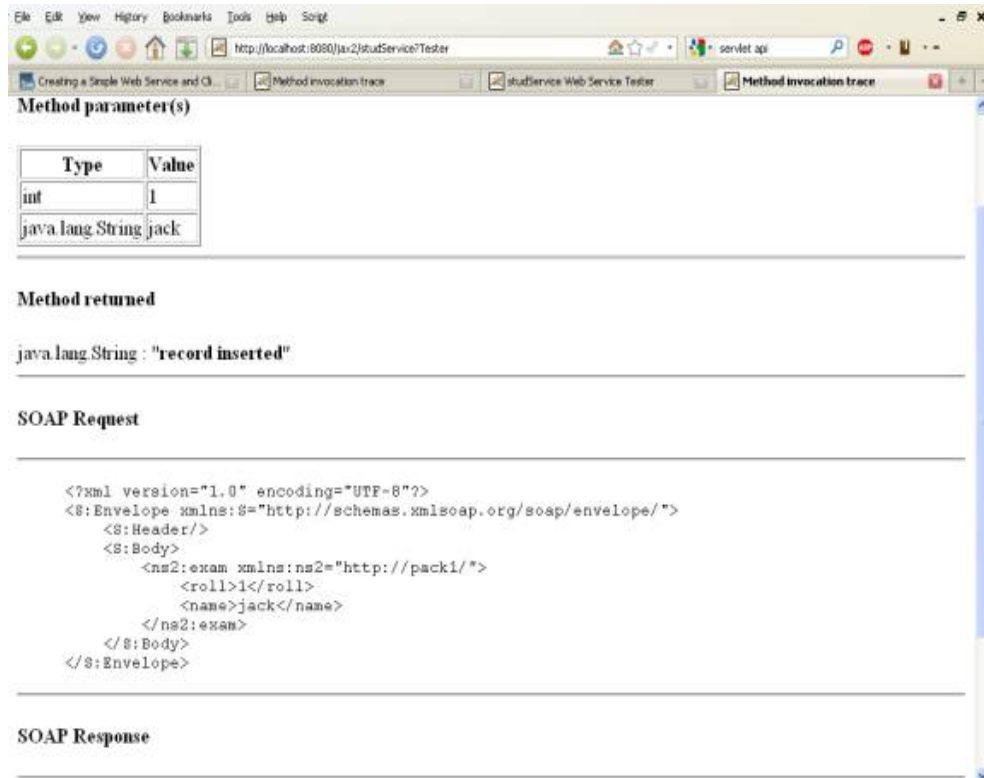


Figure 15

- Check the inserted values in the table
- Right Click on the stud table in derby database
- Select View Data as shown below in Figure 16.
- It shows the inserted data as shown below in Figure 17.

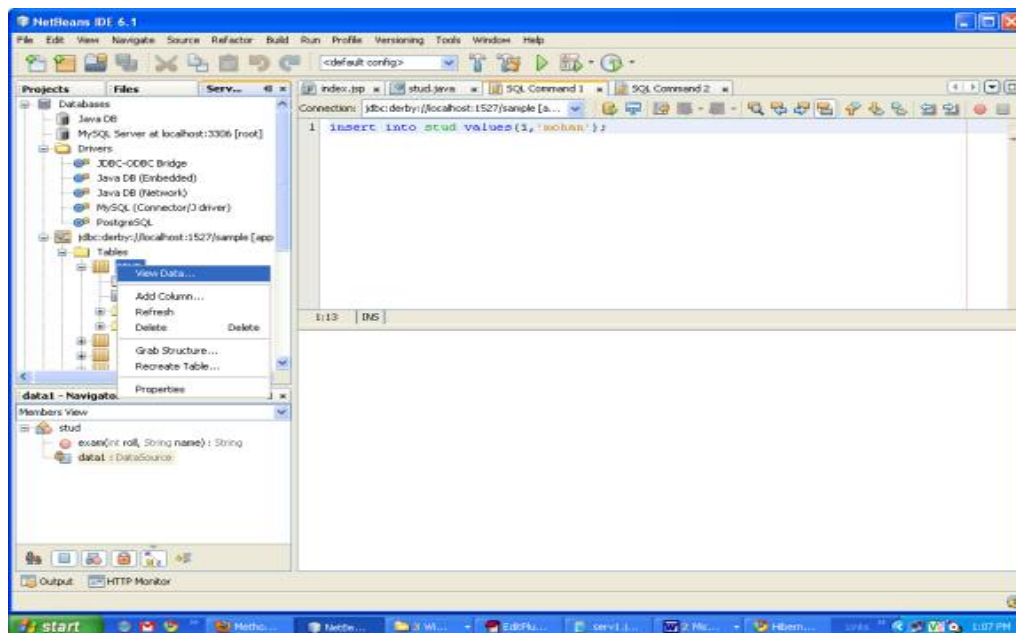


Figure 16

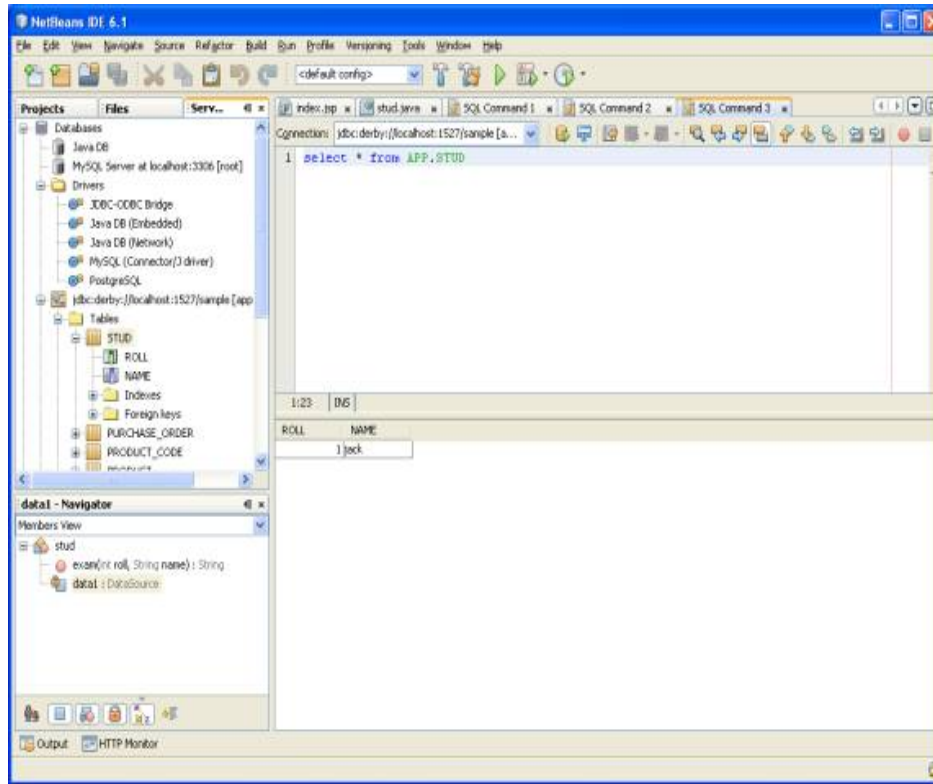


Figure 17

Client Web Service

//create a new web application

- Take a new Web Application Project
- Type name jax2Client
- Click on Next as shown below in Figure. 18

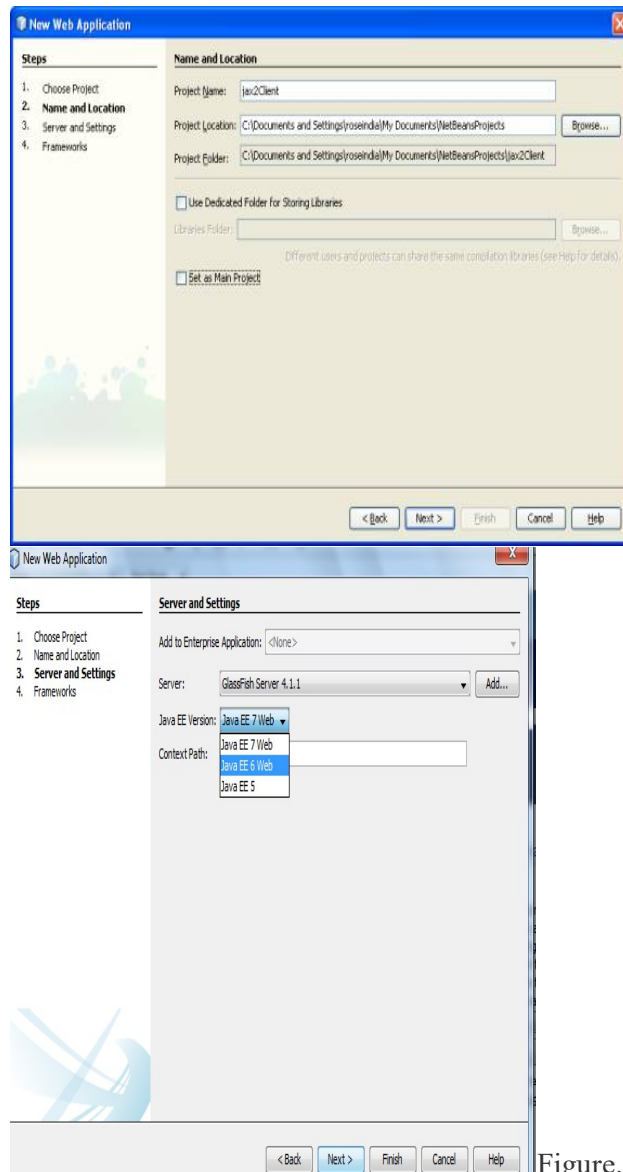


Figure. 18

- Right Click on the project jax2Client.
- Select the New->other->in categories ->click->Web services-> in file type-> click **Web Service Client**-> click next-> as in fig 19
- It creates a dialog box for WSDL and Client Location

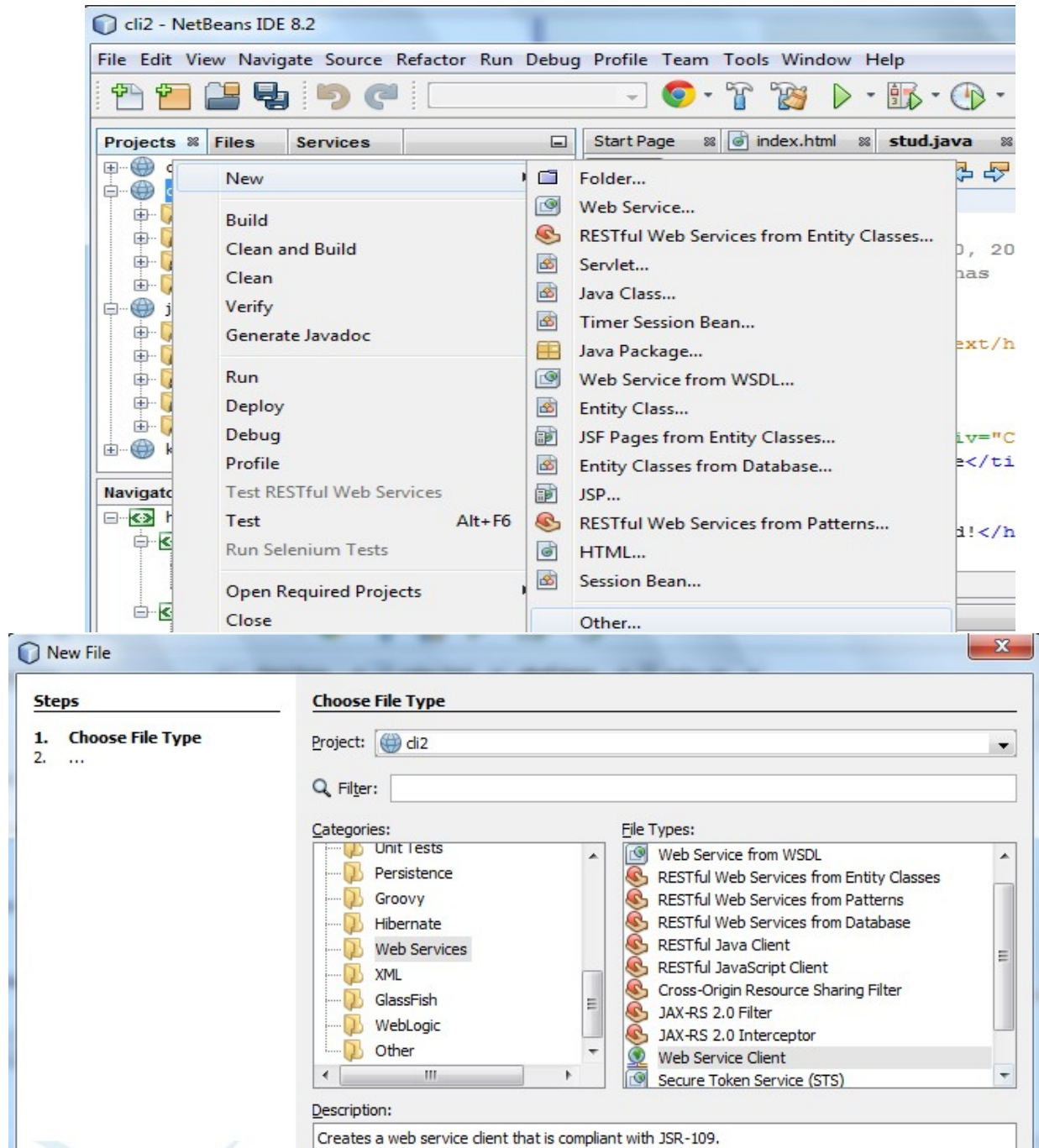


Figure 19

- Now select either the project or give the WSDL URL
 - Click on Next as shown below in Figure. 20
 - Click on the Finish Button
- If you select wsdl URL mean

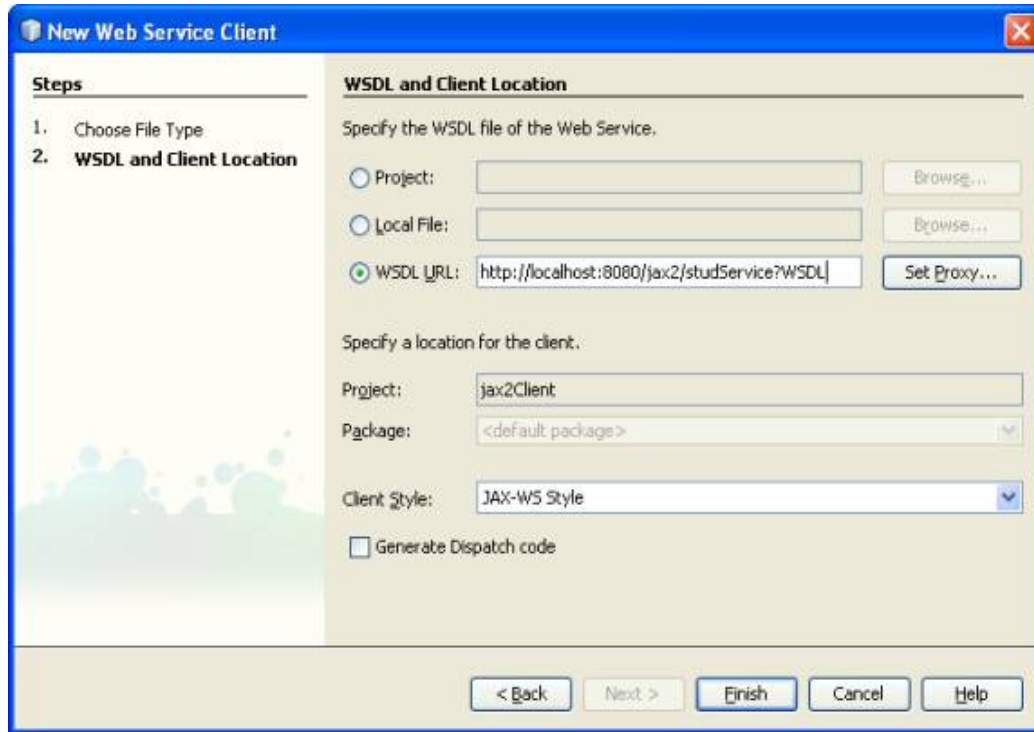
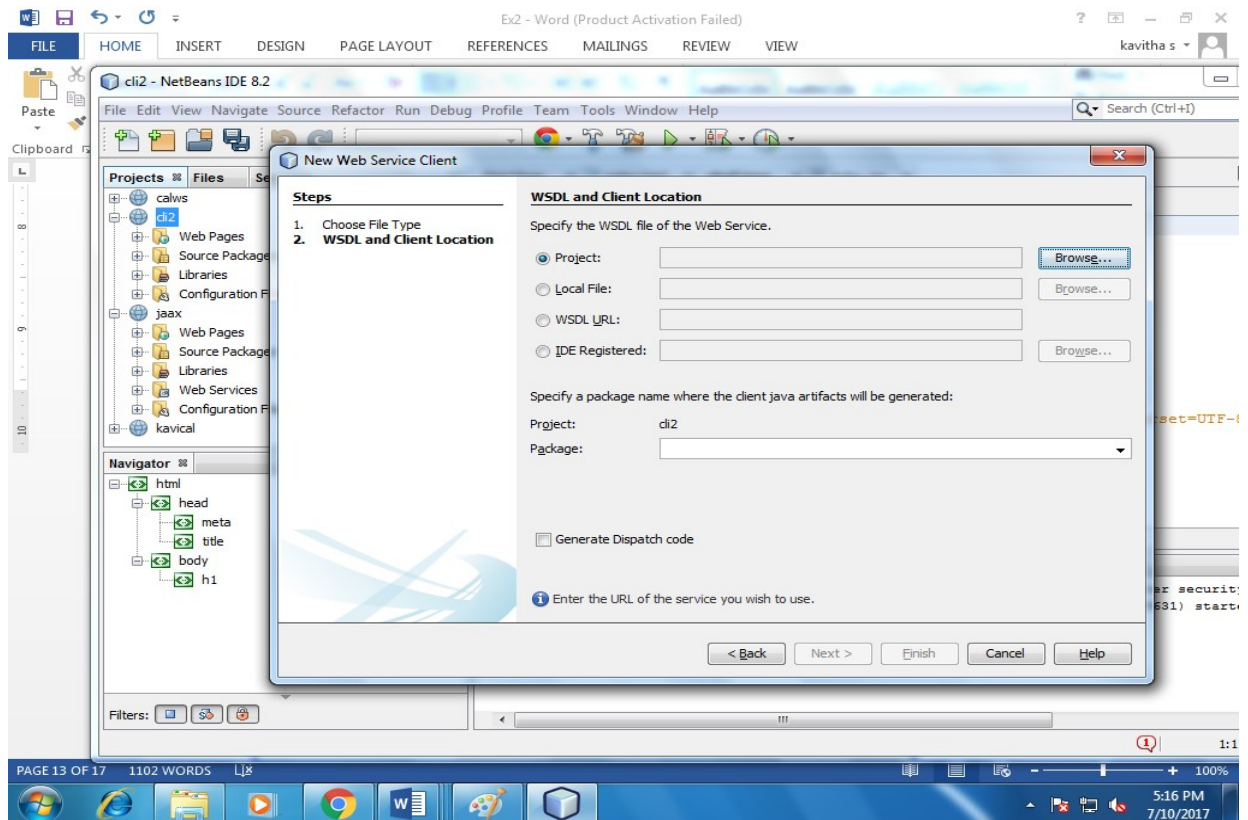
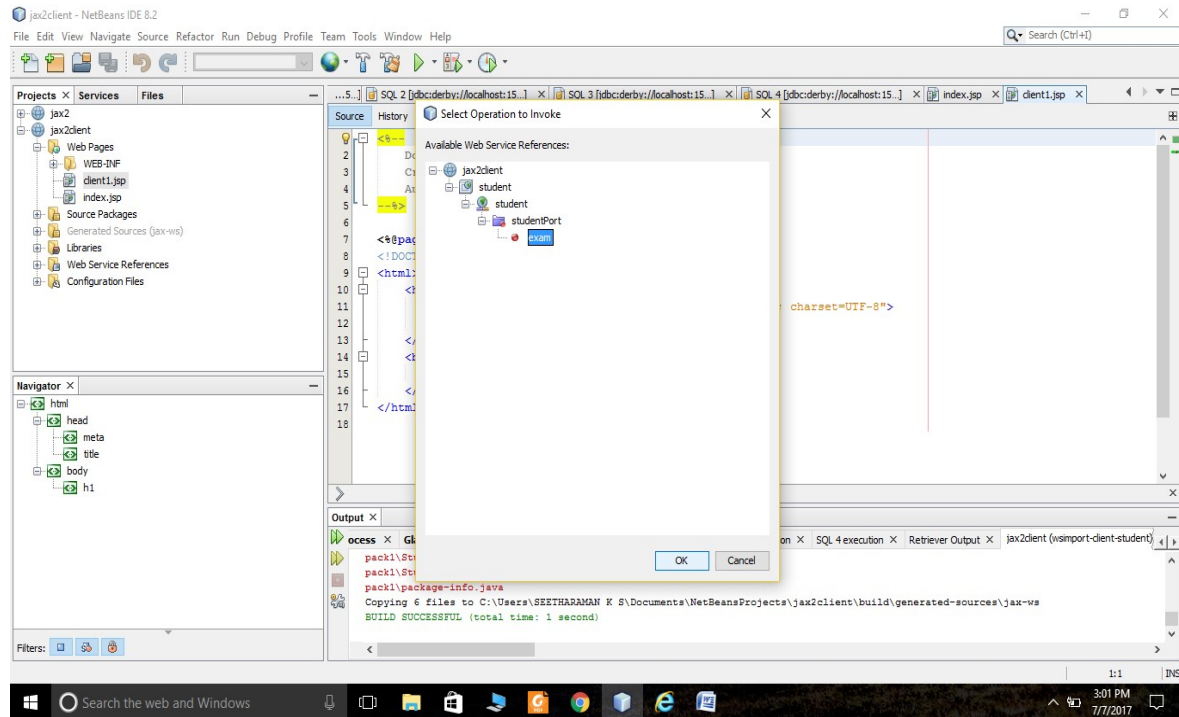


Figure. 20

Or if you select project tab and browse as in fig 20.1



Grid and Cloud Computing Lab-Manual



Client.jsp

- Now make a Client.jsp file
- Right Click on the Project jax2Client
- Select NewJsp file
- Give the name as Client1.jsp
- Right Click in the code of Client1.jsp
- Select Web Service Client Resources as shown below in Figure. 21

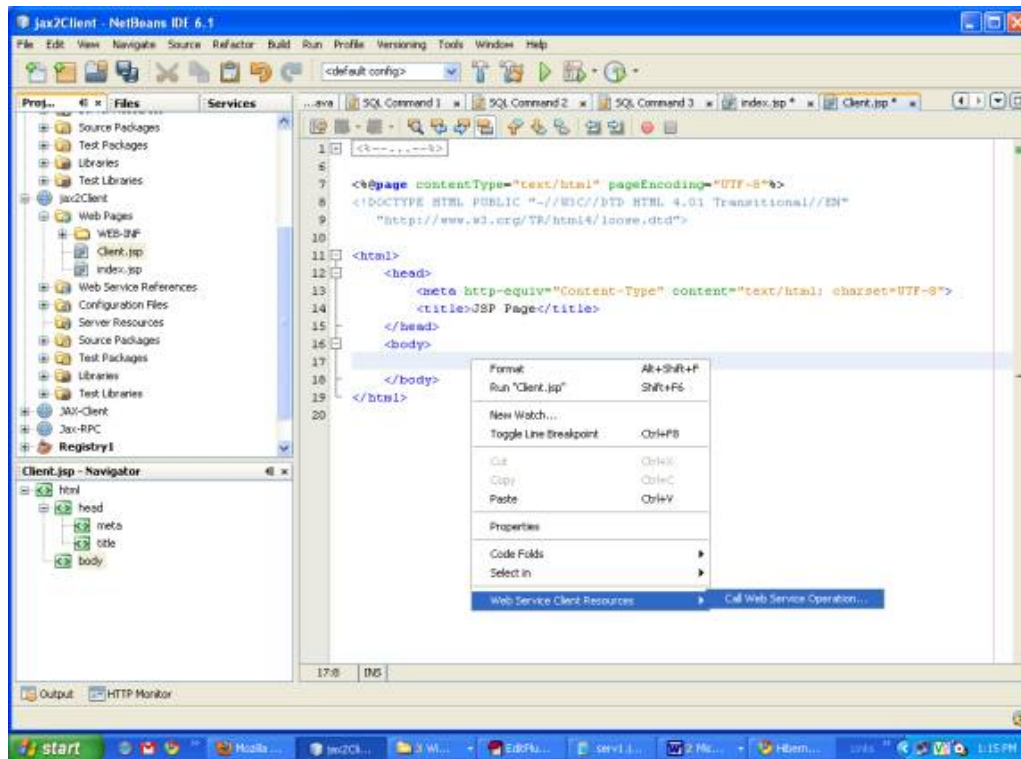


Figure. 21

- Select the operation in the Client project jax2Client\studService\studPort\exam
- As shown below in Figure. 22

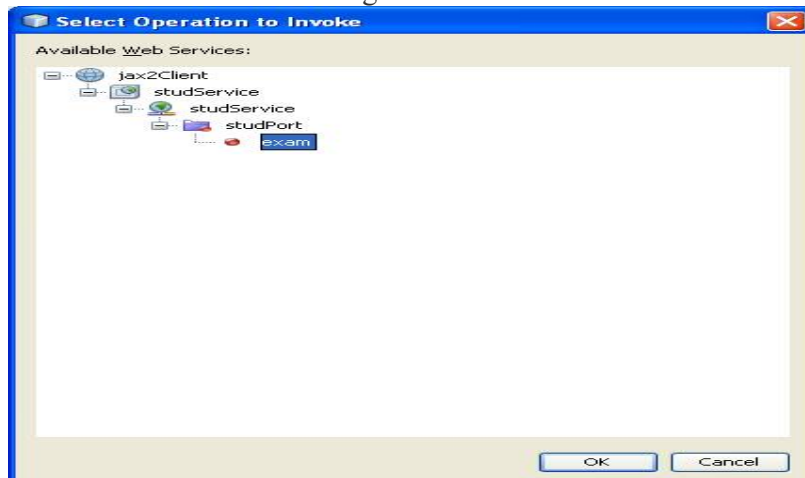


Figure. 22

- The above steps generates code in Client.jsp
- It gives the stud Web Service object, Stud port and operation name code
- In two arguments name and roll initialize the value.

As shown below in Figure. 23.

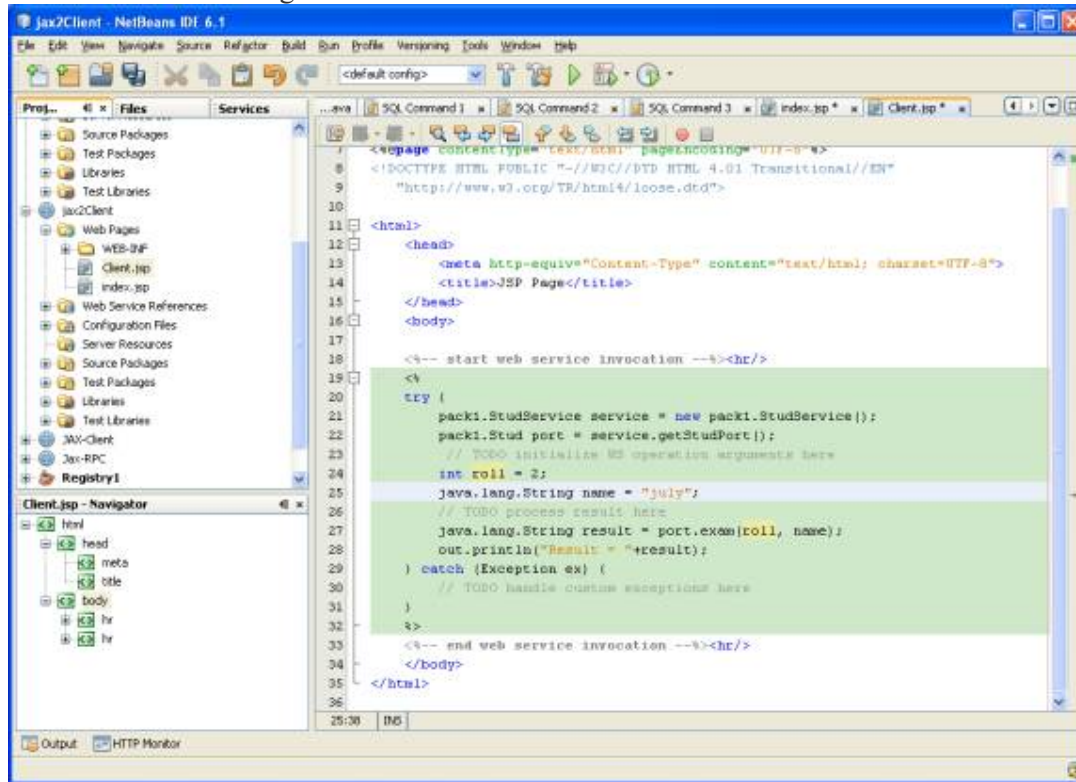


Figure. 23

Running The Client file

- Deploy the jax2Client project
- Right Click in Client.jsp and select Run Client.jsp

As shown below in Figure. 24.

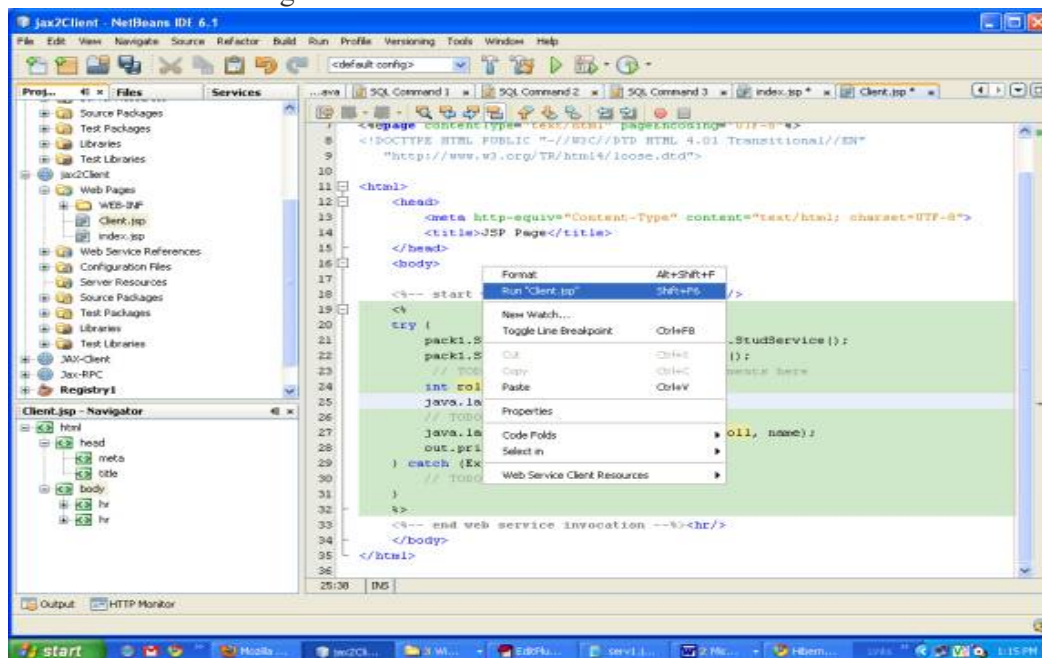


Figure. 24

Grid and Cloud Computing Lab-Manual

- It runs the file in the Internet Browser
- It gives the status message record inserted
- In case of failure it will display record not inserted

As shown below in Figure. 25

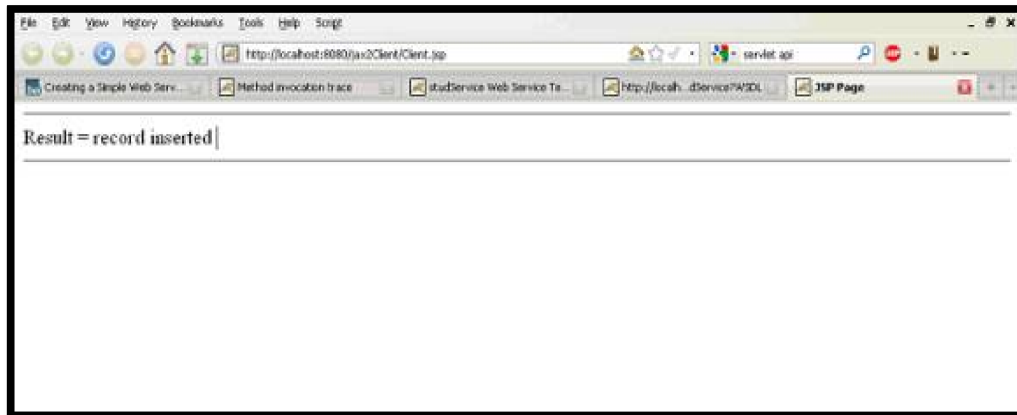


Figure. 25

- On running the Client.jsp it inserts the value into the table
- To view the table data Right Click on the stud table in sample database in derby
- It fetches the records and display it in table as shown below in Figure 26

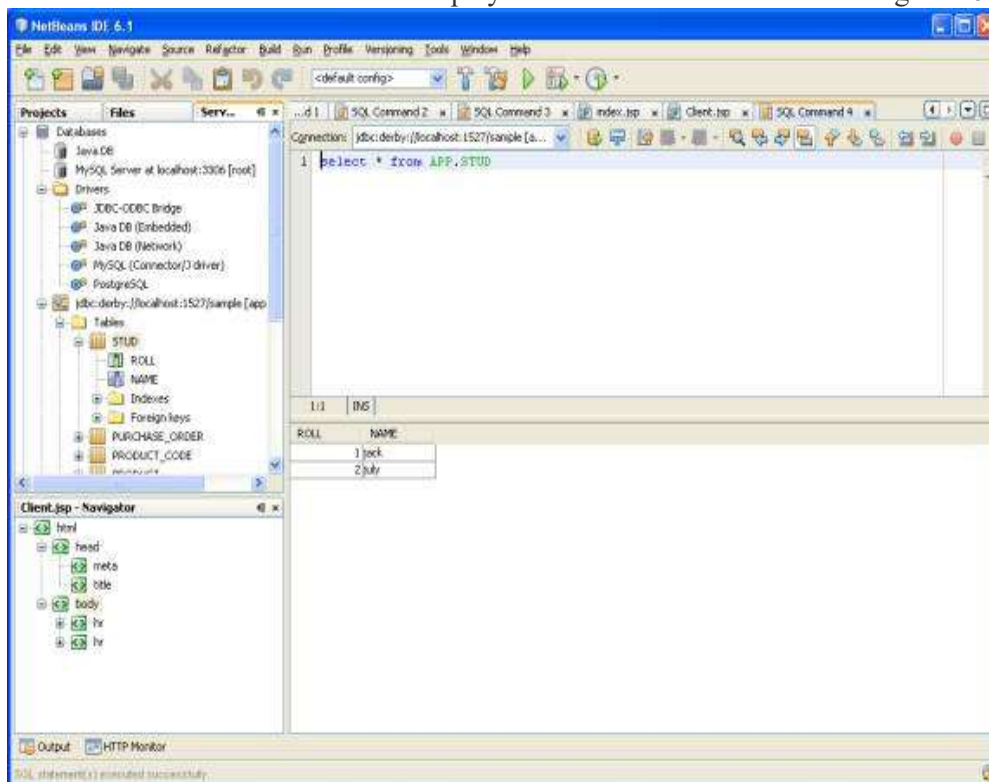


Figure. 26

PROGRAM

Stud.java

```
package student;

import java.sql.Connection;
import java.sql.PreparedStatement;
import javax.annotation.Resource;
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.sql.DataSource;

@WebService(serviceName = "stud")
public class stud {

    @Resource(name = "data1")
    private DataSource data1;

    @WebMethod(operationName = "exam")
    public String exam(@WebParam(name = "roll") int roll, @WebParam(name = "name")
String name) {

        String status="record not inserted";

        try {

            Connection con=data1.getConnection();

            PreparedStatement ps=con.prepareStatement("INSERT INTO stud VALUES(?,?)");

            ps.setInt(1,roll);

            ps.setString(2,name);

            int i=ps.executeUpdate();

            if(i!=0) {

                status="record inserted";

            }

        }

    }

}
```

```
}  
catch(Exception e){  
    System.out.println("error in strong data"+e);  
}  
return status;  
}  
}
```

Client.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>  
<html>  
    <head>  
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
        <title>JSP Page</title>  
    </head> <body> <hr/>  
    <%  
        try {  
            student.Stud_Service service = new student.Stud_Service();  
            student.Stud port = service.getStudPort();  
            int roll = 2;  
            java.lang.String name = "Niraj";  
            java.lang.String result = port.exam(roll, name);  
            out.println("Result = "+result);  
        } catch (Exception ex) {  
        }  
    %>  
    <hr/> </body>  
</html>
```

OUTPUT

stud 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 java.lang.String student.Stud.exam(int,java.lang.String)

exam	(<input type="text"/> , <input type="text"/>)
------	---

exam Method invocation

Method parameter(s)

Type	Value
int	1
java.lang.String	laksh

Method returned

java.lang.String : "record inserted"

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:exam xmlns:ns2="http://student/">
      <roll>1</roll>
      <name>laksh</name>
    </ns2:exam>
  </S:Body>
</S:Envelope>
```

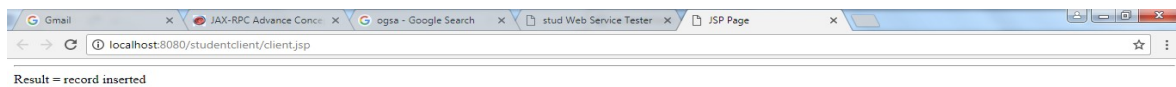
SOAP Response

Grid and Cloud Computing Lab-Manual

```
<?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:examResponse xmlns:ns2="http://student/">
      <return>record inserted</return>
    </ns2:examResponse>
  </S:Body>
</S:Envelope>
```

#	ROLL	NAME
1		1 laksh

#	ROLL	NAME
1		1 laksh
2		2 Niraj



CONCLUSIONS

Thus, a new open grid service architecture- complaint web service in java using NetBeans IDE 8.2 was built and deployed successfully.

3. Develop a Grid Service Using Apache

AIM

To develop a grid service using Apache axis in net beans 8.2

PROCEDURE

Search axis2 download in google

Download both binary distribution and war distribution

Unzip both the downloads

In the binary distribution go to lib folder

Copy all the jar files alone

Then go to the following location(This is as per my pc, check for your pc)

C:\Users\14cse35\AppData\Roaming\NetBeans\8.2\modules\ext\axis2

Delete all the files present there.

Paste the jar files that you copied from lib folder

Go to Tools -> Libraries and select Axis-1.3 (version may differ for you)

Under library class path select all and click remove

Now click Add JAR/Folder

Browse to the location

C:\Users\14cse35\AppData\Roaming\NetBeans\8.2\modules\ext\axis2 (location may differ)

Select all the files and click Add JAR/Folder

Also check for library name version and update it and click OK.

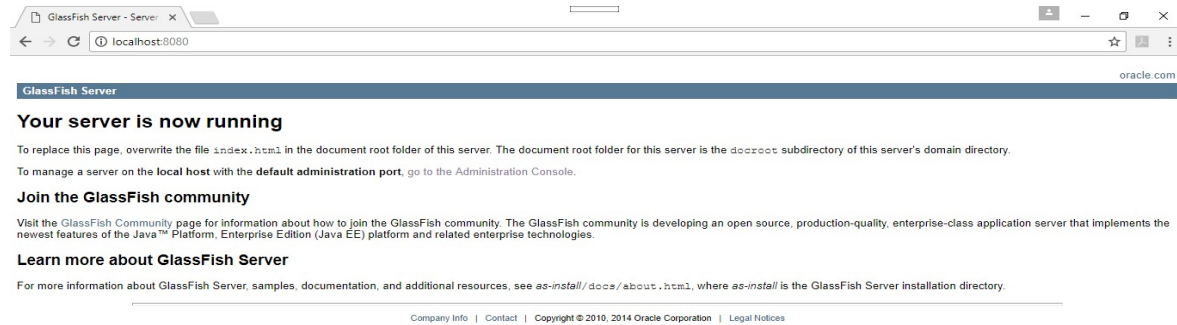
Start the glass fish server in net beans IDE from services -- > server

Go to browser and type

Localhost:8080

If the following comes you are on the right path

Grid and Cloud Computing Lab-Manual



Go to browser and type localhost:4848

Click Applications from the left pane

Click Deploy and then select the file axis2.war from the war distribution

After the file gets loaded click OK

A file axis2 with hyperlink will be there. Click that

Again go to Applications in the left pane and click Launch which is across Axis2 service

A browser page opens click the first link

Click Services and then click Version

It will take you to a wsdl page.

Edit the url as follows

...../axis2/services/Version/getVersion?

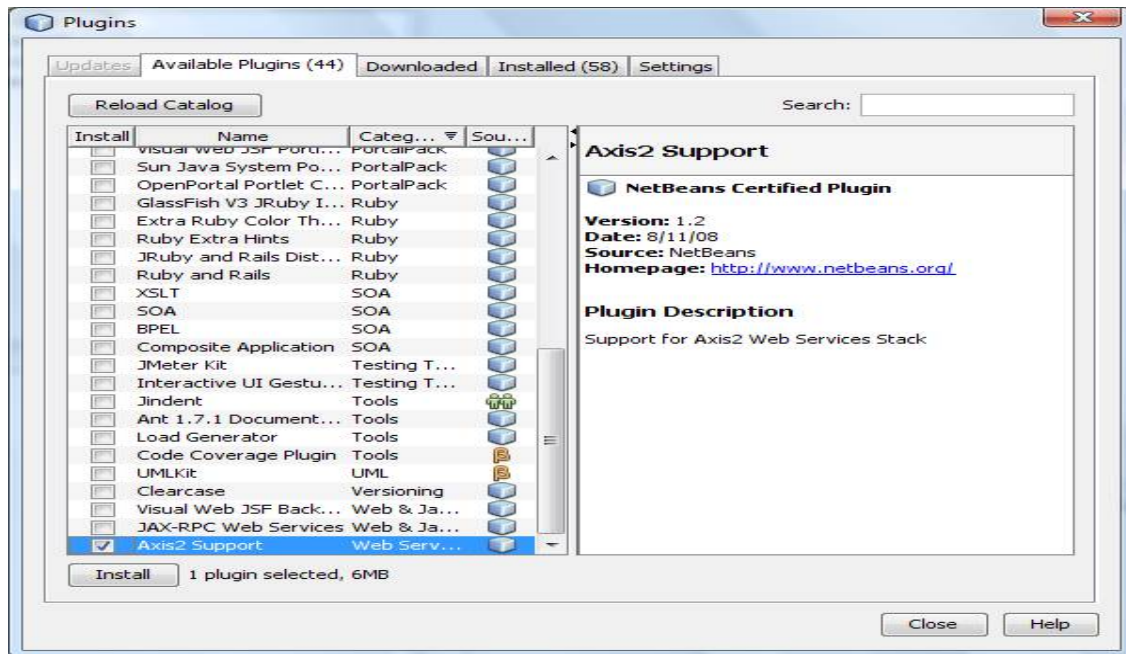
Hit Enter, you will get the version number of deployed service

Installing the Axis2 Support Plugin

- In the IDE, go to the Plugin Manager, under the Tools menu, and check whether Axis2 Support Plugin is installed, under the Installed tab. If it is installed, check whether an update is available in the Updates tab. If the Axis2 support is not installed, install it from -> Go to Tools -> Plugin -> Settings tab and then click Add.

Now give the below url

<http://deadlock.netbeans.org/hudson/job/nbms-and-javadoc/lastStableBuild/artifact/nbuild/nbms/updates.xml.gz>



Setting Up Axis2 Options for GlassFish

- Go to this url and click the first link

<http://www.apache.org/dyn/closer.lua/axis/axis2/java/core/1.7.5/axis2-1.7.5-war.zip>

To set up Axis2 options for Glassfish:

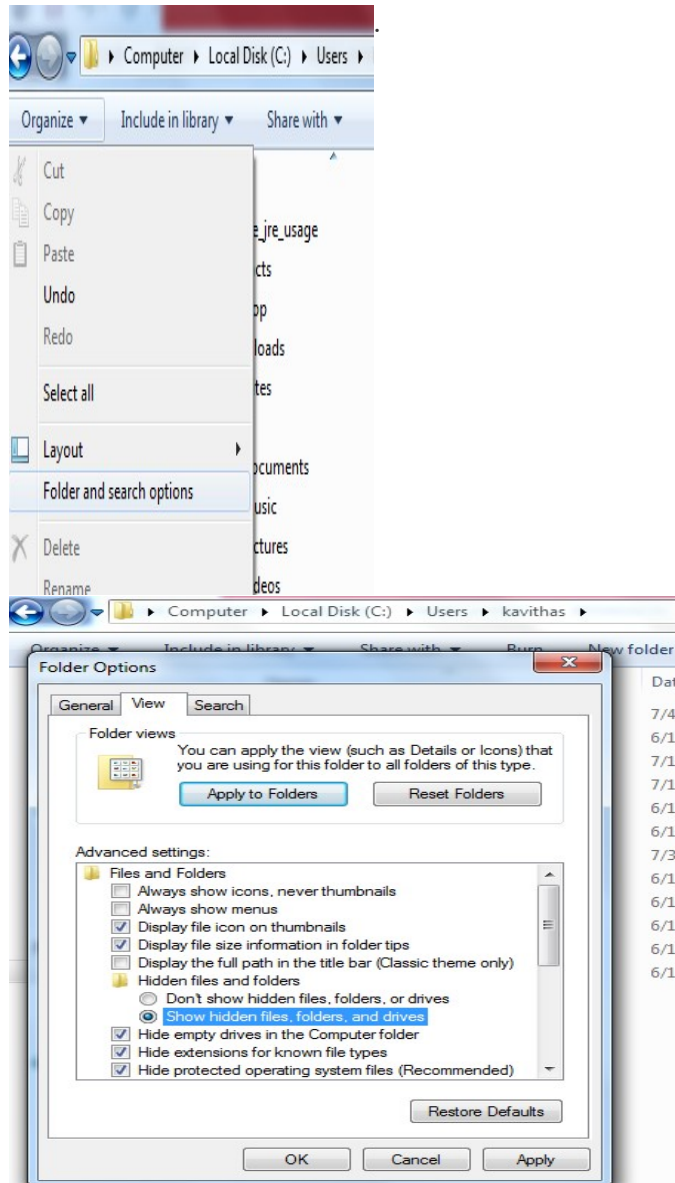
Unpack the downloaded archive file

containing axis2.war to GLASSFISH_HOME/domains/DOMAIN_NAME/autodeploy. To find GLASSFISH_HOME and the name of your domain, start the IDE and open the Services tab. Expand the Servers node. Right-click the GlassFish node and select Properties from the context menu. The Domains folder location and the name of the domain are visible in common tab.

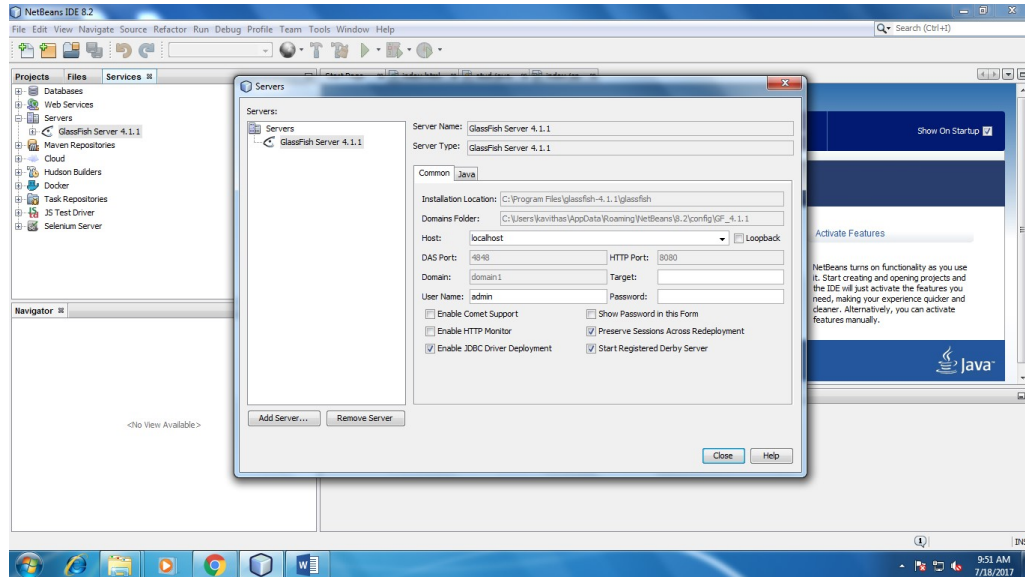
C:\Users\14cse35\AppData\Roaming\NetBeans\8.2\config\GF_4.1.1\domain1\autodeploy

(AppData → is a hidden folder → to open that → after clicking your particular user name → click organize tab → Folders and search option → click view tab → select show hidden files folders → click ok → now AppData folder Visible)

Now click AppData → Roaming → Netbeans → 8.2 → config → GF-4.1.1 → domain1 → Autodeploy → paste the axis.war file in the autodeploy folder

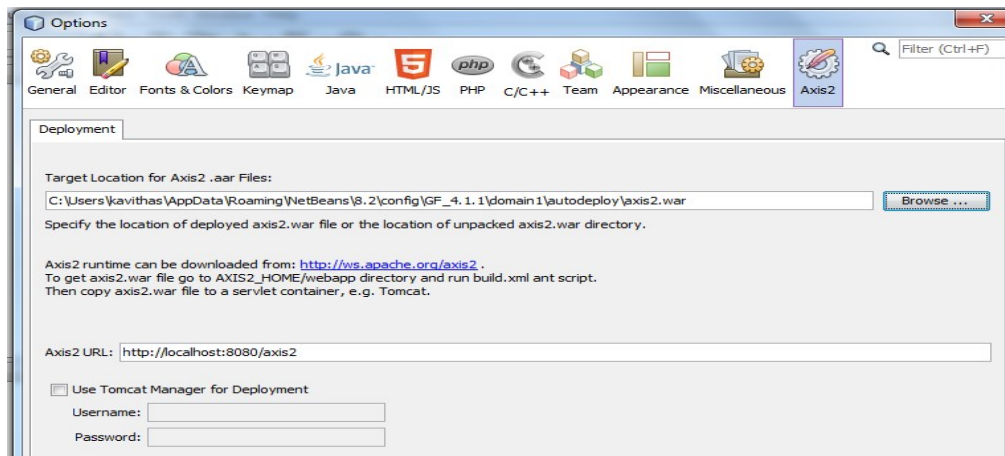


Grid and Cloud Computing Lab-Manual



Start the IDE. From the top menu bar, choose Tools -> Options. The Options dialog opens.

Click the Axis2 icon. The Axis2 deployment options page opens. -> browse link (C:\Users\14cse35\AppData\Roaming\NetBeans\8.2\config\GF_4.1.1\domain1\autodeploy\axis2.war)-> click ok

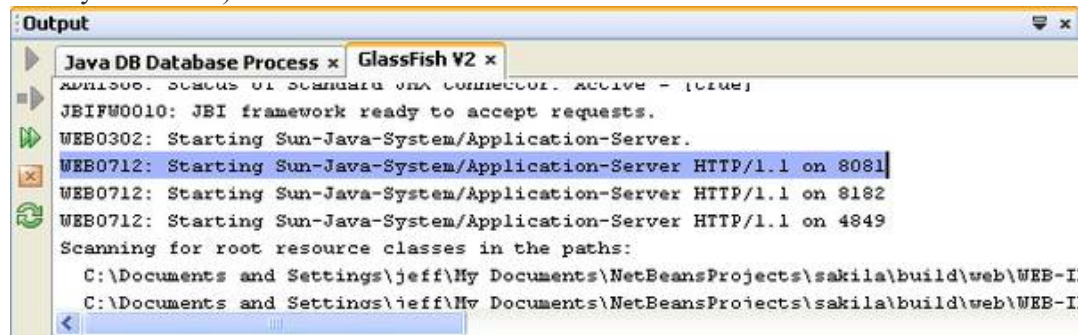


Set the target location for Axis2 AAR files to the axis2.war file you unpacked into the GlassFish autodeploy directory.

By placing axis2.war into autodeploy, you enable GlassFish to automatically redeploy axis2.war every time you alter the file. On GlassFish, however, you cannot redeploy the WAR file while the server is running.

5. Make sure the Axis2 URL field contains the correct port number for your GlassFish server. To check the port number, start GlassFish (from the Services tab or from Tools

-> Servers) and see what 80xx port HTTP 1.1 uses. The default port number is 8080. In the following image, the correct port number is 8081 (because another server already uses 8080).



```
Output
Java DB Database Process x GlassFish V2 x
ADM1300: STATUS OF STANDARD JMX CONNECTOR: Active - (true)
JBIFW0010: JBI framework ready to accept requests.
WEB0302: Starting Sun-Java-System/Application-Server.
WEB0712: Starting Sun-Java-System/Application-Server HTTP/1.1 on 8081
WEB0712: Starting Sun-Java-System/Application-Server HTTP/1.1 on 8182
WEB0712: Starting Sun-Java-System/Application-Server HTTP/1.1 on 4849
Scanning for root resource classes in the paths:
C:\Documents and Settings\jeff\My Documents\NetBeansProjects\sakila\build\web\WEB-I
C:\Documents and Settings\jeff\My Documents\NetBeansProjects\sakila\build\web\WEB-I
```

Developing an Axis2 Web Service

In this section, you use NetBeans IDE to create, deploy, test, and modify an Axis2 web service.

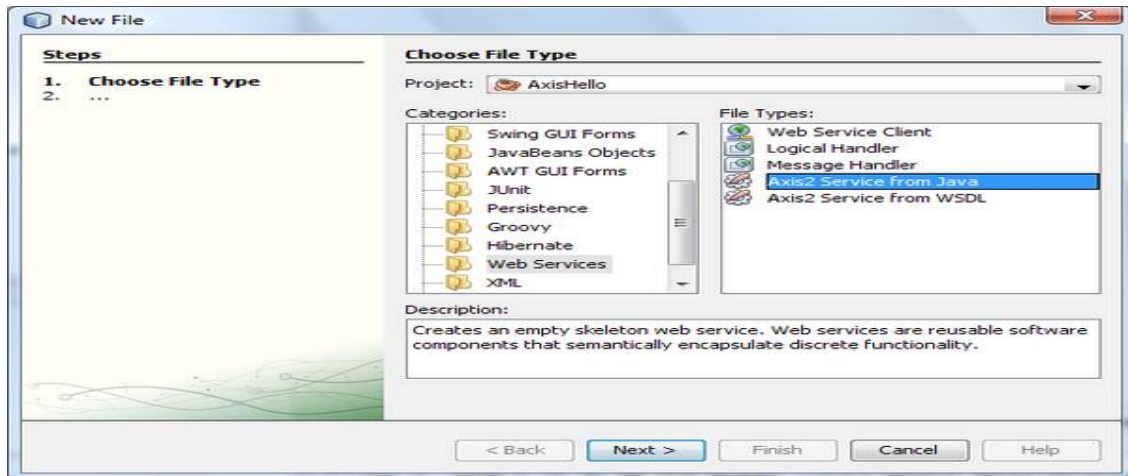
Creating an Axis2 Web Service

With NetBeans IDE, you can create an Axis2 web service from a Java class. You can only do this from a Java application or Java library project. In this tutorial, you create a Java library project (because you do not need a main method), create an Axis2 web service in that project (creating the Java class at the same time) and deploy the Axis2 web service to a server. You can only create an Axis2 web service from a Java or Java Library project. This is because the axis.aar file (the deployable archive into which web services and Axis configuration files are packed) is neither a WAR nor an EAR and cannot be deployed normally as a web (EAR) application.

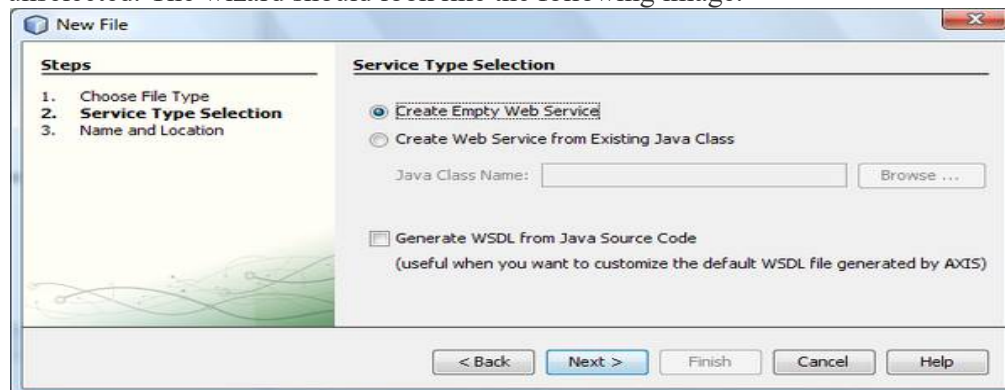
To create an Axis2 web service:

1. Click the New Project icon or File -> New Project. The New Project wizard opens. From the Java category, select a Java class library project. Click Next.
2. Name the project AxisHello. Check that you are using the project folder name and location that you want. It is up to you whether to share the project. Click Finish, and the IDE creates the project.

3. Right-click the project node. The context menu opens. In the context menu, choose New -> Other. The New File wizard opens. From the Web Services category, choose Axis2 Service from Java and click Next.

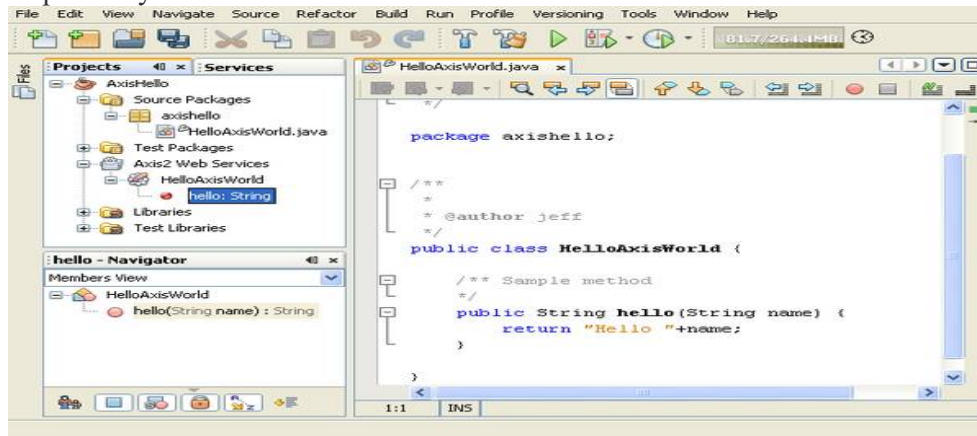


4. The Service Type Selection page of the New File wizard is now open. You do not have any Java classes in the project, so select "Create an Empty Web Service." If you had already coded a Java class, you would have selected Create a Web Service from an Existing Java Class. If you wanted to edit the WSDL of the web service, for example to add or change namespaces, you would select Generate a WSDL from Java Source Code. Editing WSDL is outside the scope of this tutorial, so leave this unselected. The wizard should look like the following image.



5. Click Next. The Name and Location page opens. Name the Java class HelloAxisWorld. Name the package axishello. Leave Generate Sample Method selected. This generates a method in the Java class that returns "Hello, World."
6. Click Finish. The IDE generates a HelloAxisWorld.java class in the axishello source package and a HelloAxisWorld Axis2 web service that mirrors this Java class. You can see that both the Java class and the Axis2 web service have a hello:String

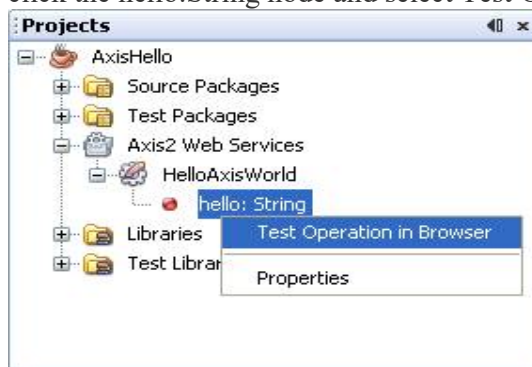
operation, shown in the Navigator tab and as a node of the Axis2 web service, respectively.



Deploying and Testing an Axis2 Web Service

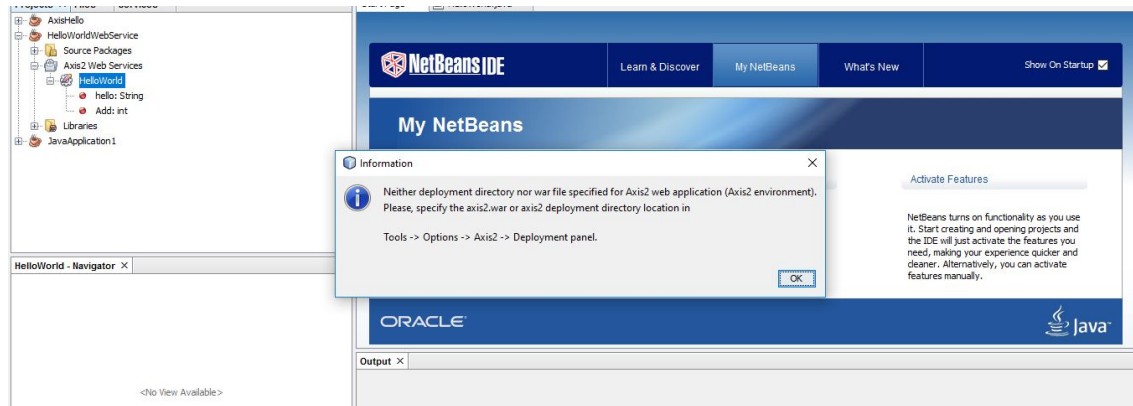
To deploy an Axis2 web service to the server:

1. Right-click the web service's node. The context menu opens. Select Deploy to Server. The IDE compiles an Axis2 AAR file and copies it to the axis2.war file used by the application server.
2. If you have enabled automatic deployment, the web service is deployed to the server. If the server is not running, start it and the web service is automatically deployed.
3. To test the service, expand the web service node to reveal the operations. Right-click the hello:String node and select Test Operation in Browser.



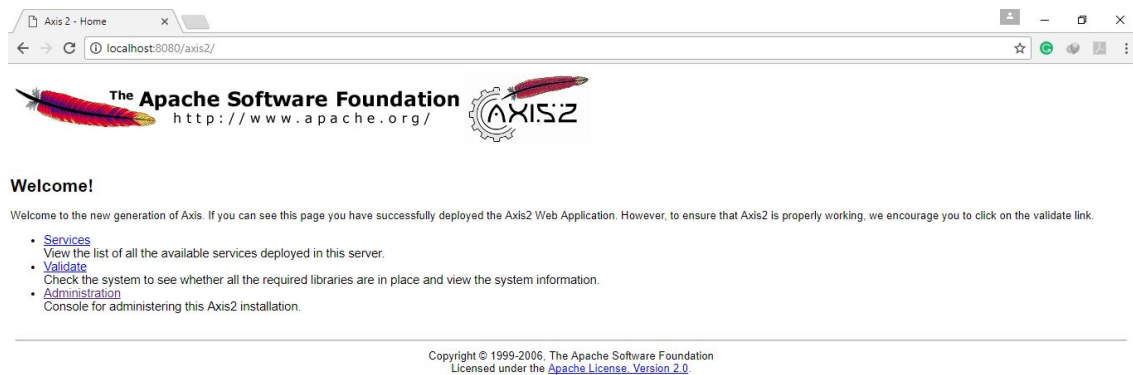
IF THE TEST OPERATION IS NOT DEPLOYED IN SERVER LIKE GIVEN BELOW!!!!

Grid and Cloud Computing Lab-Manual

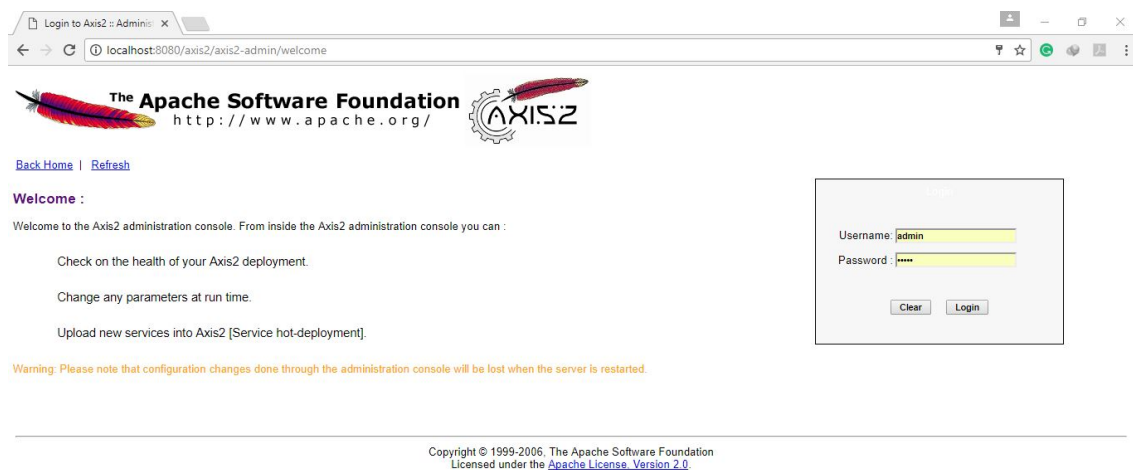


THEN, WE DEPLOY IT MANUALLY,

First Build the java project. then,

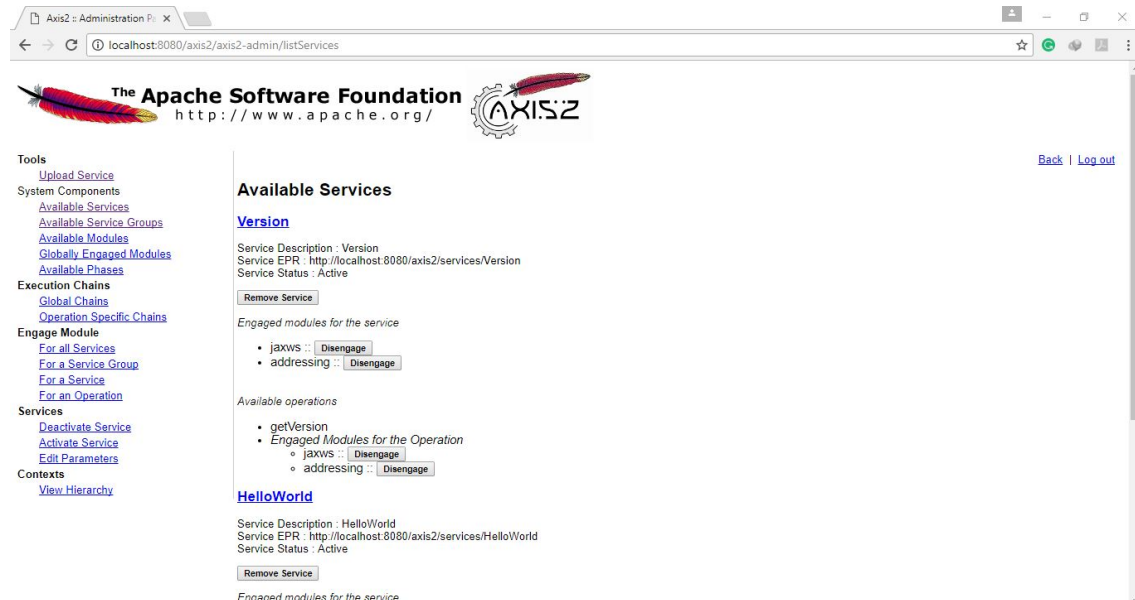


Go to Web-browser->localhost:8080/axis2/



Type user name:admin and Password:axis2

Grid and Cloud Computing Lab-Manual



->Administration->Available Services->it'll show java_class->and give the values as what mentioned in the java directory.

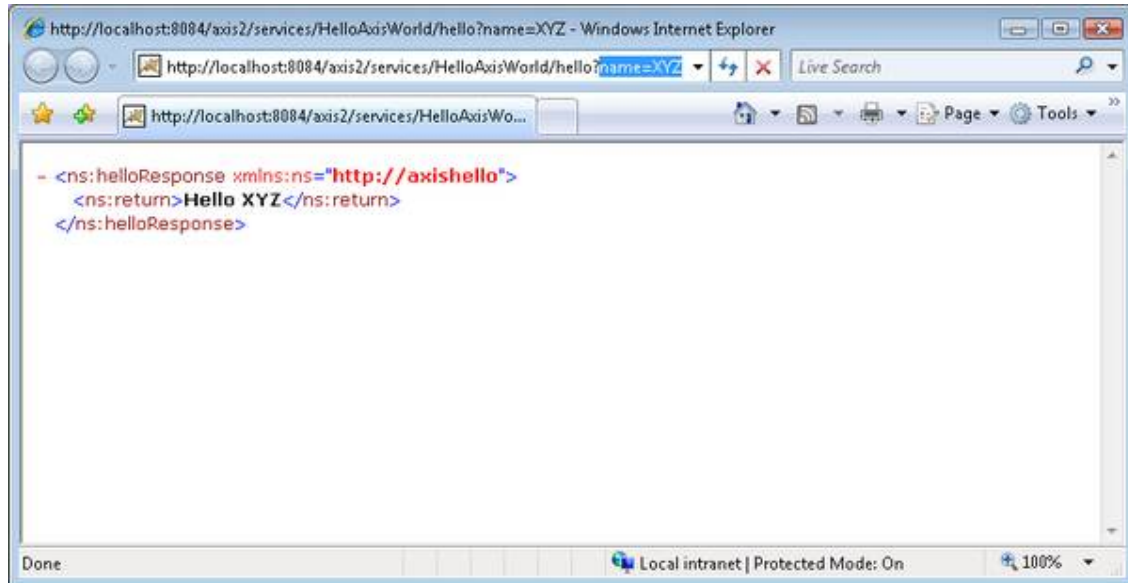
Finally,it run on the browser given above.

*If it not run further go to Netbeans project->extract your java class->In services.xml ->change the input and output messageReceiver namespace like given below.

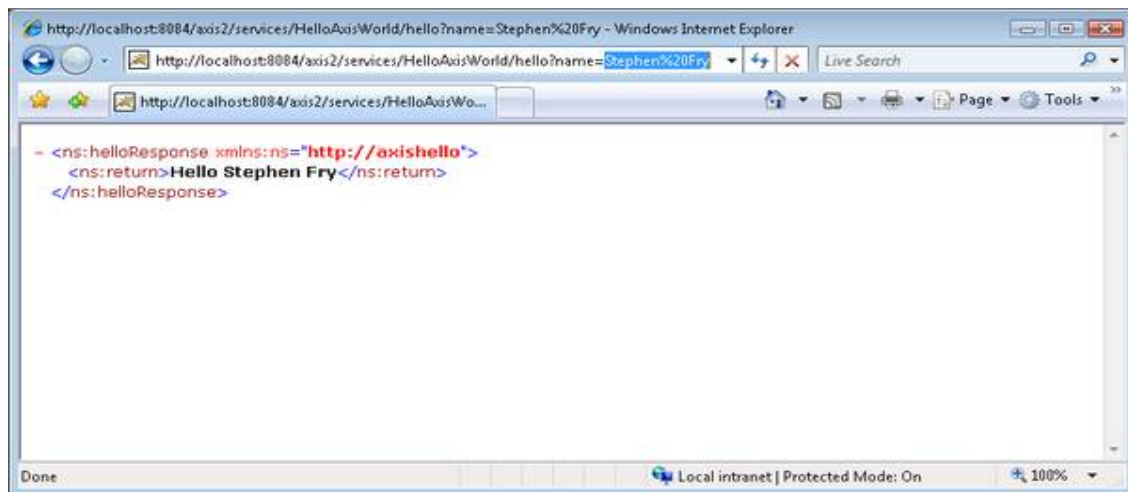
Services.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<serviceGroup><service scope="application"
name="HelloWorld"><description>HelloWorld
service</description><messageReceivers><messageReceiver
class="org.apache.axis2.rpc.receivers.RPCInOnlyMessageReceiver"
mep="http://www.w3.org/ns/wsd/in-only"/><messageReceiver
class="org.apache.axis2.rpc.receivers.RPCMessageReceiver"
mep="http://www.w3.org/ns/wsd/in-out"/></messageReceivers><parameter
name="ServiceClass">services.HelloWorld</parameter></service></serviceGroup>
```

4. Your browser opens with a test value of your variables. The test value is appended to the URL.



5. Change the variable value in the URL and press Enter. The test result changes as well.



Changing the Web Service's Operations

To change the web service operations, edit the Java file in the project. The operations in the web service change simultaneously. Add a simple `add` method to `HelloAxisWorld.java`, as below.

```
public class HelloAxisWorld {
```

```
    /** Sample method
```

```
*/  
public String hello(String name) {  
    return "Hello "+name;  
}  
public int add(int x, int y) {  
    return x+y;  
}  
}
```

Save the Java file, and the operation appears as a subnode of the web service.

Redeploy the web service and test it as described in [Deploying and Testing an Axis2 Web Service](#).

Refer: <http://aragorn.pb.bialystok.pl/~dmalyszko/PaWWW/ps34-axix2.htm>



CONCLUSIONS

Thus, the grid service using Apache axis in net beans 8.2 is developed and the outputs are verified.

4. DEVELOP APPLICATIONS USING JAVA OR C/C++ GRID APIS

AIM

To develop an applications using Java or C/C++ Grid APIs in Net beans 8.2

PROCEDURE

To create RESTful web services, you need a Java Web application project.

To create the project:

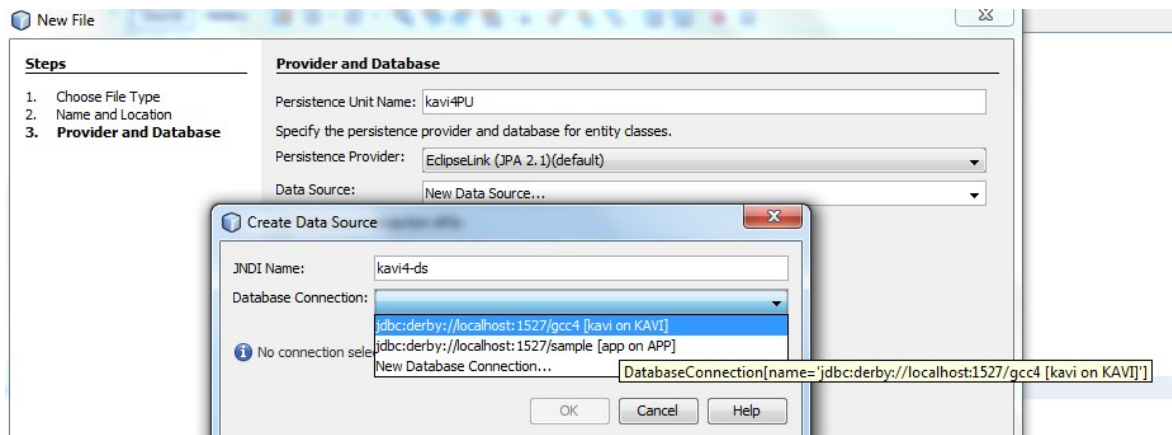
1. Choose File > New Project (Ctrl-Shift-N. Under Categories, select Java Web. Under Projects, select Web Application. Click Next.->Gridlab(project name)->next-> The New Web Application wizard opens.
2. Select either Java EE 6 Web or Java EE 7 Web. Under Server, select the server you want to use, but note that Java EE projects require GlassFishserver . Click through the remaining options and click Finish.

Creating Database

Select services->Database->javaDB->right click->create Database->database name :gridlab->username:gridlab ->password->gridlab ->retype pwd: gridlab->click ok

Select & Right click->jdbc:derby://localhost:1527/ gridlab->connect

Goto project->select your project name(gridlab)->rightclick->new->entity class->class name:seller->next->gotoDatasource->select new datasource->select database connection and click ok->then click finish



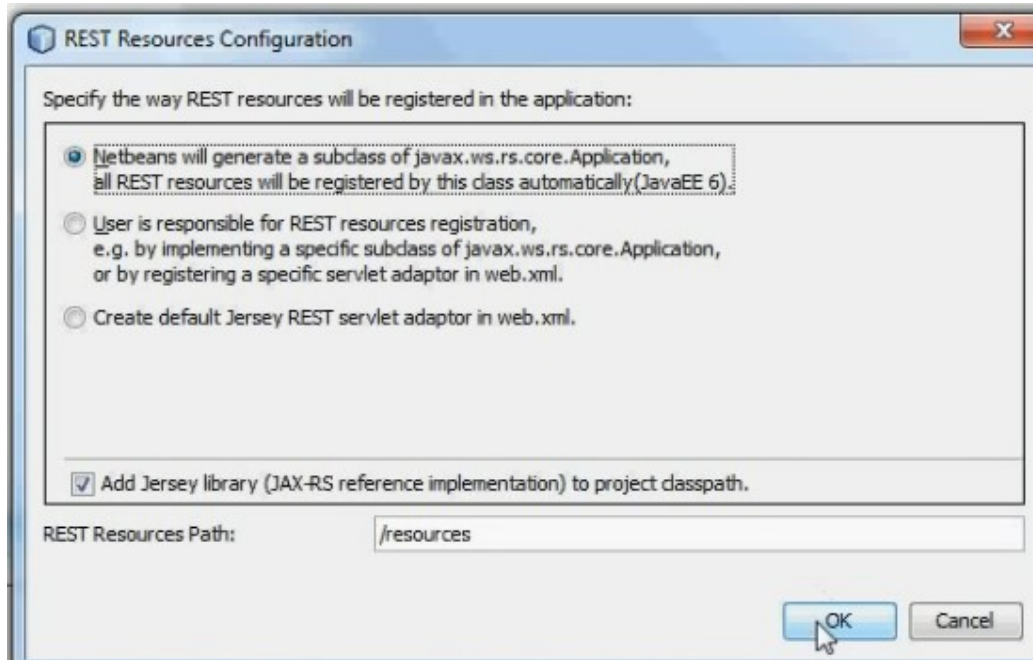
Seller.java code page will open-> right click in the code page->select insert code-> select add property->edit name as->lastname->ok

Again in-> Seller.java code page will open-> right click in the code page->select insert code-> select add property->edit name as->firstname->ok

Again in-> Seller.java code page will open-> right click in the code page->select insert code-> select add property->edit name as->email->ok

Next go to project-> right click->new->select RESTful webservices from Entity classes...

A window open->select from the available entity classes->Seller(com.bonbhel.oracle.kavi4)->click Add->next->edit resource package name->select from dropdown->gridlab and edit as (gridlab.service)->finish->check for option netbeans is selected->click ok



Got your project-> expand gridlab.service->click AbstractFacade.java->code will display

Next click->SellerFacadeREST.java->code will display->in the code section-> edit the following `@path("gridlab.seller")`-> as `@path("/seller")`->

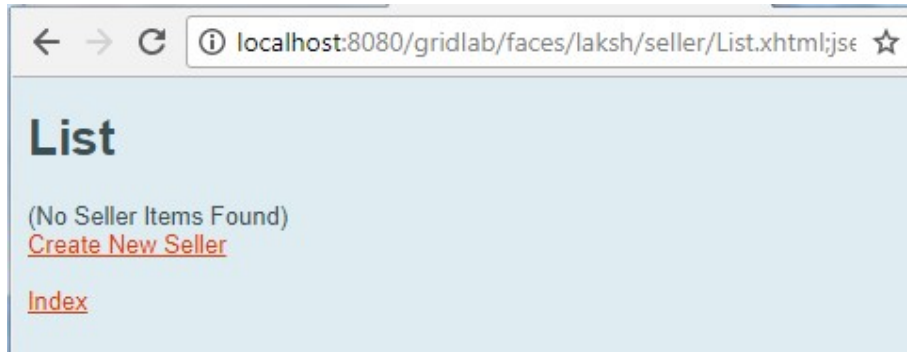
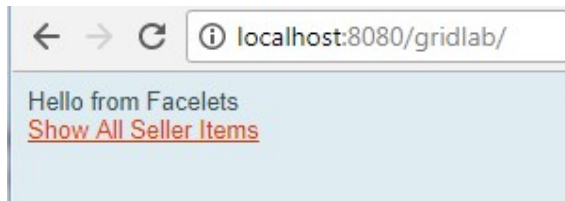
Rightclick project->run->browser page will open and display as Hello world-> in the address bar->edit the address as-><http://localhost:8080/kavi4/resources/seller/> ->reload the page-> output display as `<sellers>`



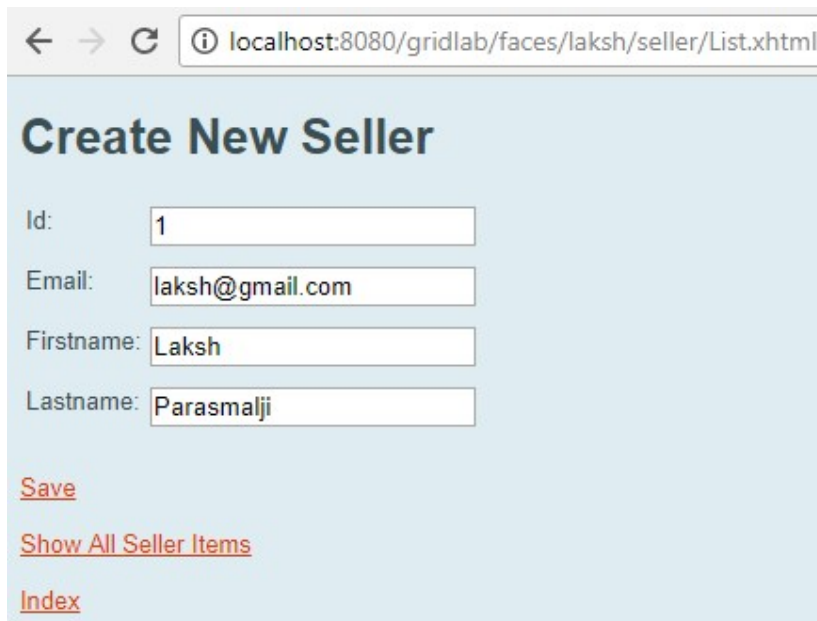
Now go to project->rightclick->new-> JSF pages from entity classes.. -> a window open-> select from available entity classes-> com.bonbhel.oracle.kavi4.seller-> click add-> next-> edit session bean package as-> com.bonbhel.oracle.kavi4.facade-> then edit JSF classes package as-> com.bonbhel.oracle.kavi4.presentation-> then JSF pages Folder as->ui-> next->check for Libraries as->JSF 2.2(version may vary)->finish

OUTPUT

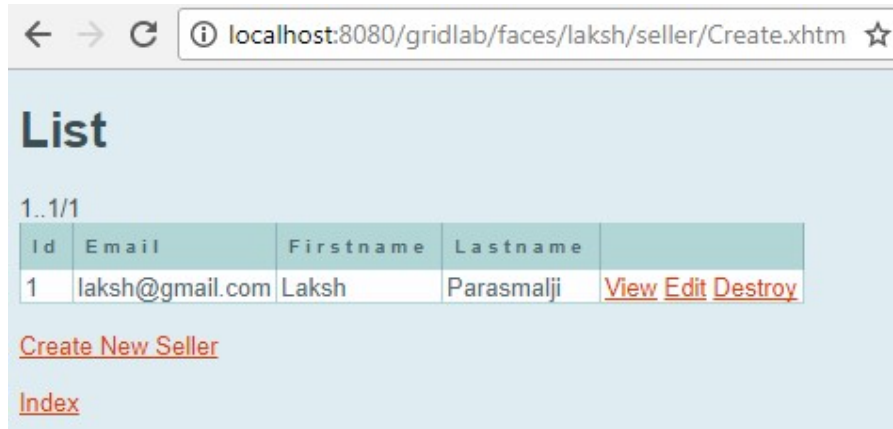
Goto project->right click-> run



Click show all seller items->click create new seller-> type in the fields and click save



-> Then click show All Seller Items



CONCLUSIONS

Thus the implementation of applications using java or c/c++ grid apis has been verified successfully.

5. DEVELOP SECURE WEB SERVICES APPLICATIONS

AIM

Develop a web service program for Square area calculation. Make this web service secured using security mechanism of “Username Authentication with Symmetric Keys”

PROCEDURE

- Adding User to Glassfish in Administration
- Make a web project
- Develop web service
- Secure Web Service Application
- Deploy the project
- Make a Client Web project
- Make a Web Service Client
- Edit properties of Web Service
- Deploy and Run the Client

Adding user to GlassFish

To add users to Glassfish using the Admin Console, follow these steps:

- Start the NetBeans
- Start the Glassfish Server as shown below in Fig.1
- Open the Admin Console in the internet browser with the url <http://localhost:4848>
- Give the User Name admin, password adminadmin

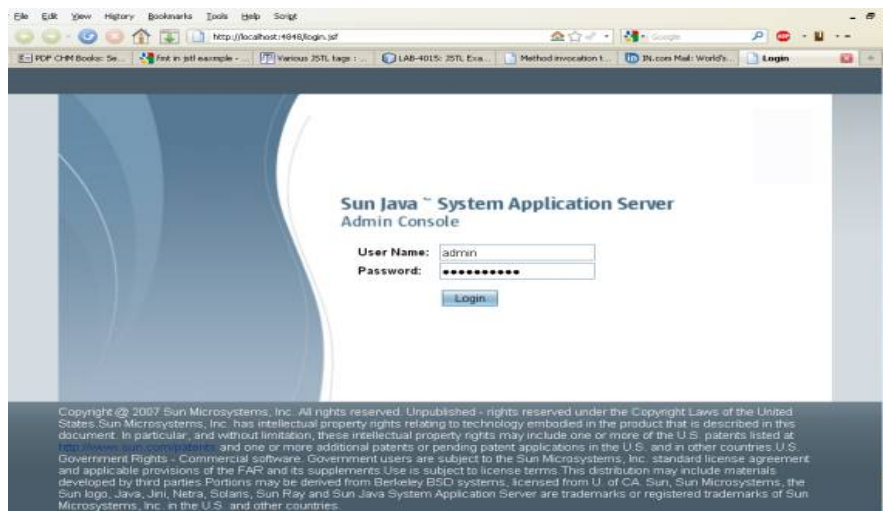


Figure. 1

- After successful login Admin Console gets opened as shown below in Figure. 2

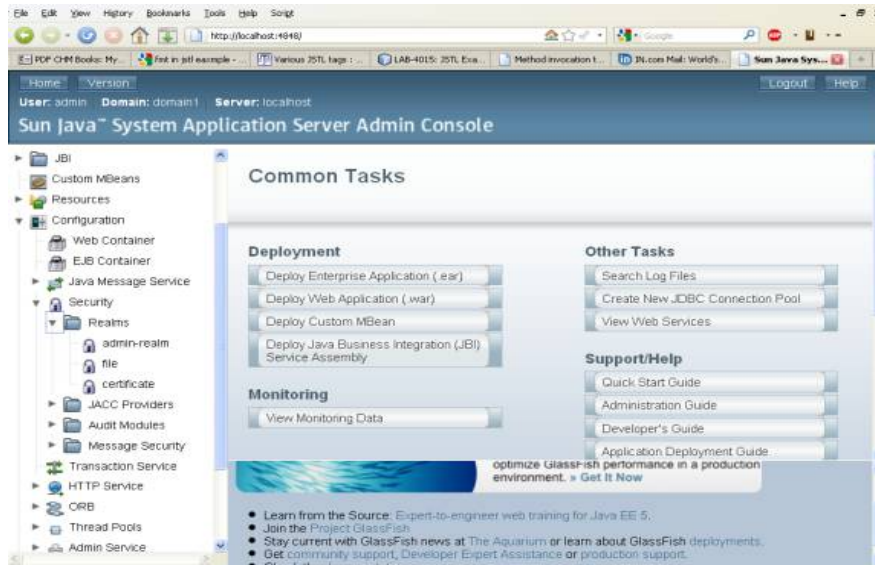


Figure. 2

Expand the Configuration node in the Admin Console tree.

- Expand the Security node in the Admin Console tree.
- Expand the Realms node. Select the file realm as shown below in Fig. 3

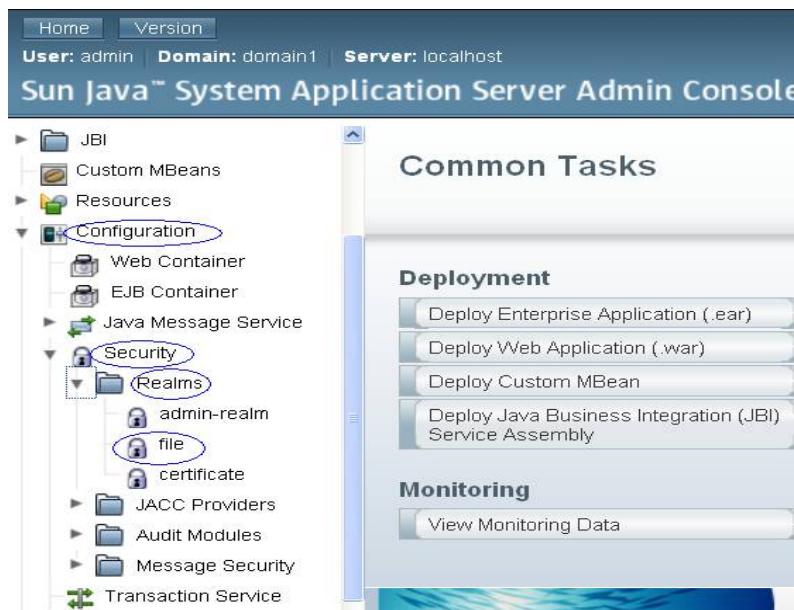


Figure. 3

- Click on New Button for New File Users
- It opens the New File Realm User Form as shown below in Figure 4.

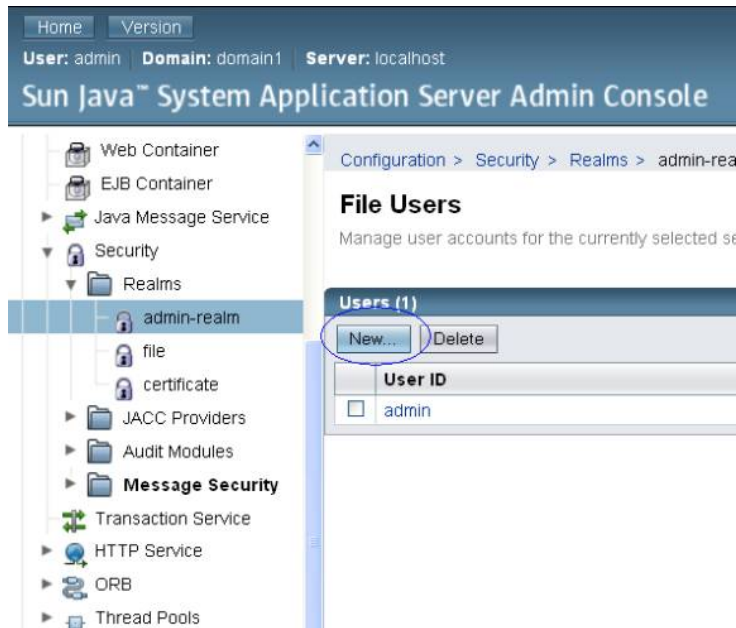


Figure .4

- In the New File Realm User Form give the following values
 - User ID = wsitUser
 - Group List = wsit
 - New Password = changeit
 - Confirm New Password = changeit

as shown below in Figure. 5

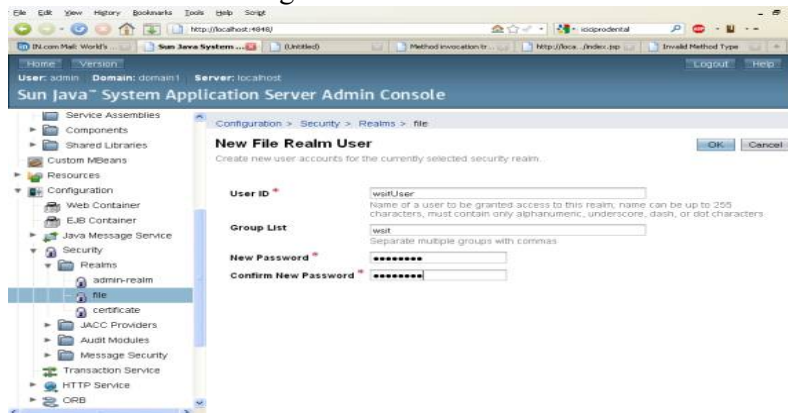


Figure. 5

- Click OK to add this user to the list of users in the realm.
- After user creation, it gets displayed in File Users as shown below in Figure 6.

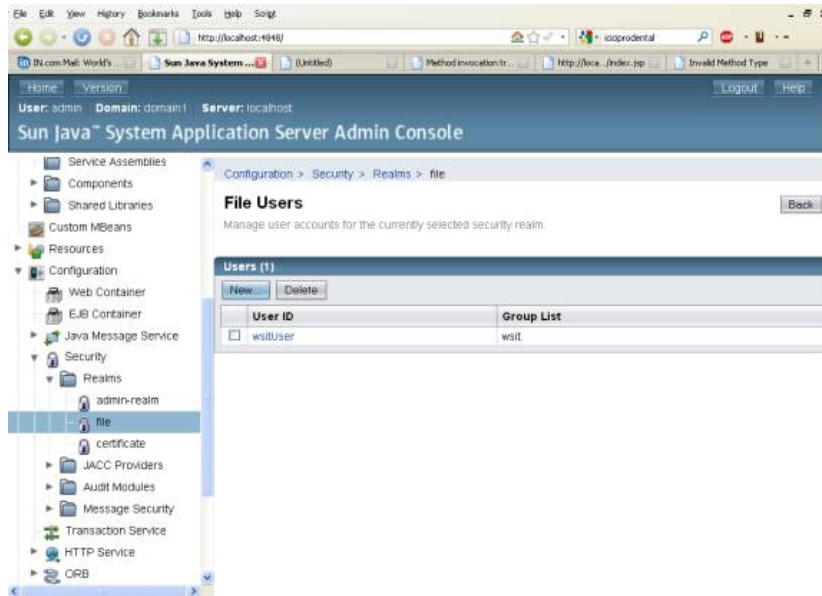


Figure. 6

Web Service Project Creation>

- Create a Web Application Project
- Type the name as WebServiceSecurity.
- Click on Next Button as shown in Figure. 7.

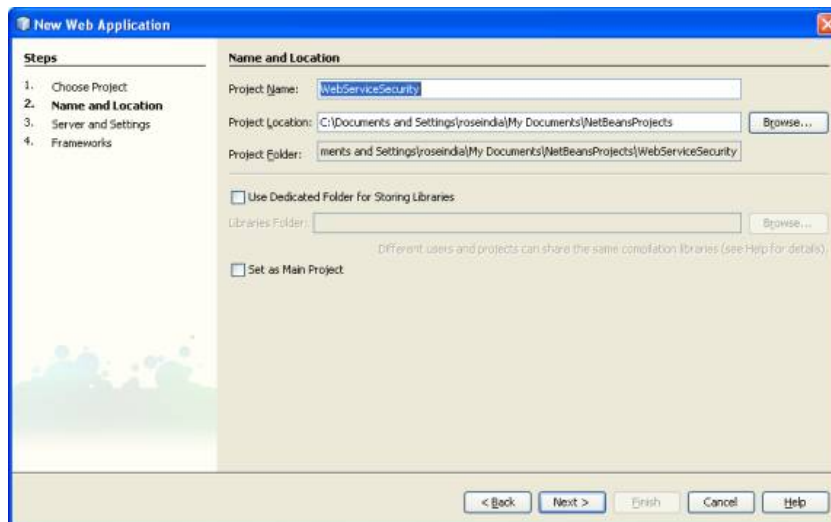


Figure. 7

- Now select the Server
- Select the Glassfish as shown below in Figure 8.
- Click on Finish Button.

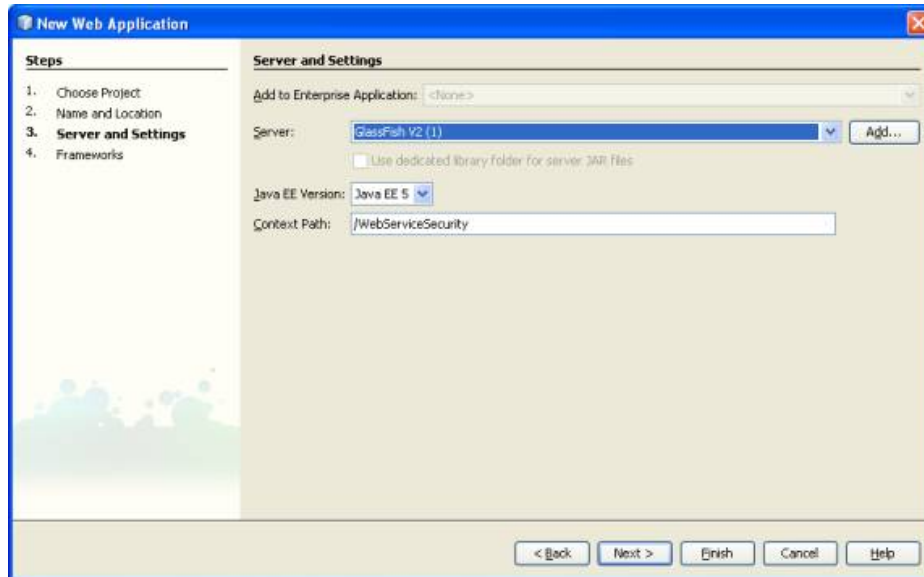


Figure. 8

Web Service File Creation >

- The above steps creates a Web Application Project
- Create a Web Service File
- Right Click on the Web Service Security
- Select New > Web Service as shown below in Figure. 9.

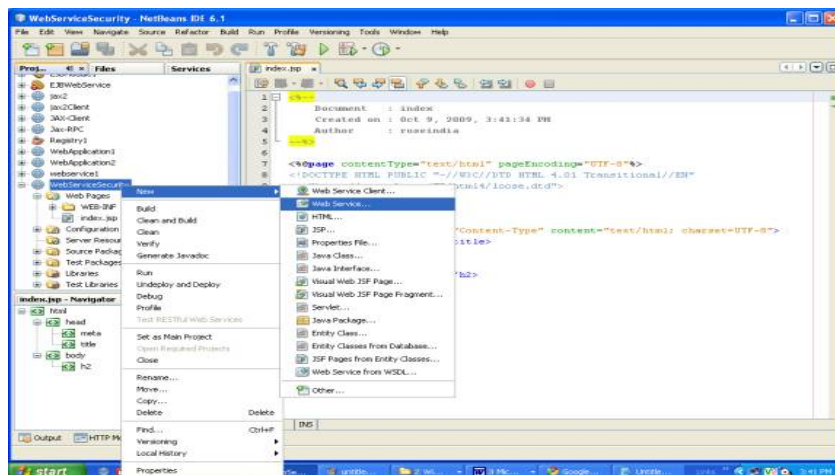


Figure 9

- Type the web service class name as square
- Type the package name as pack1
- Click on Finish as shown below in Figure 10.

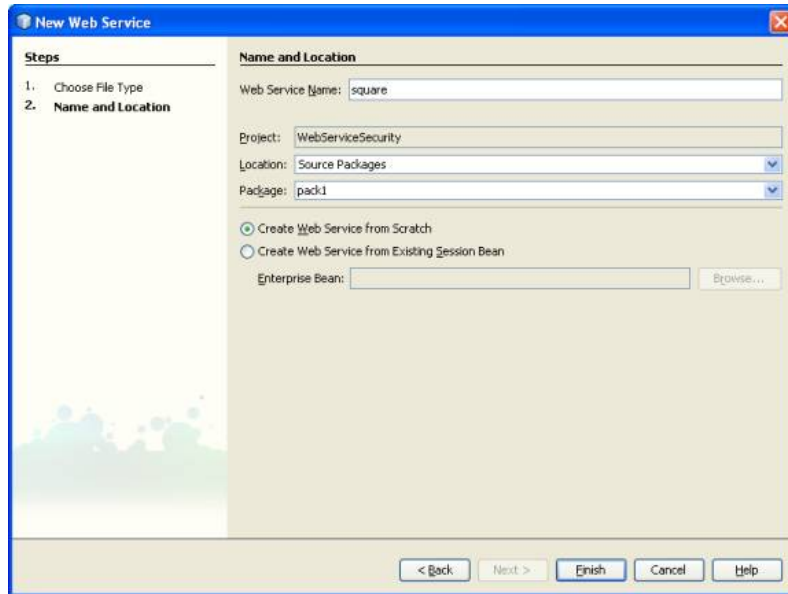


Figure.10

- It creates a Web Service in design view
- Click on Add Operation Button as shown below in Figure 11

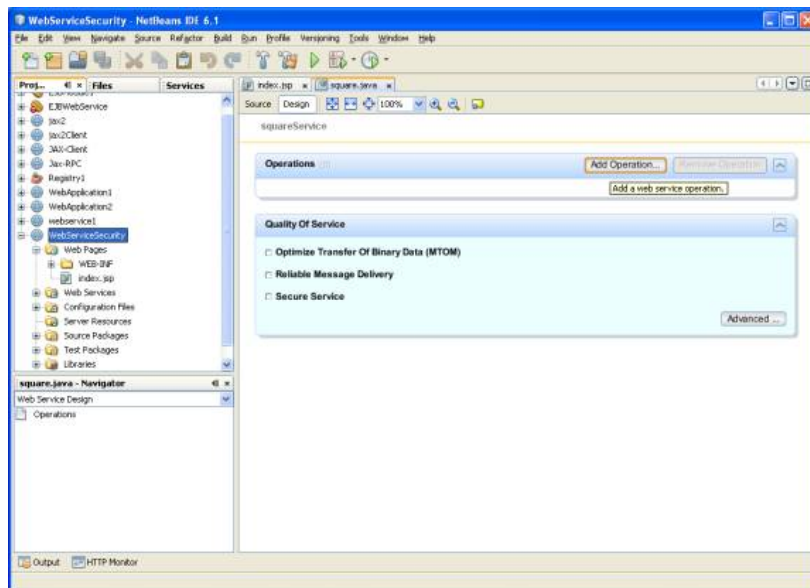


Figure. 11

- Give the operation name as area with return type String
- Click on add to add parameter
- Type the name of parameter as side and type as int.
- Do above steps as shown below in Figure 12.
- It generates the code of operation area.

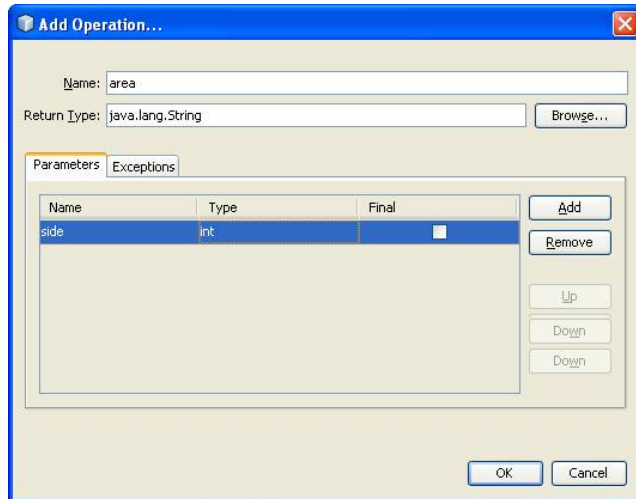


Figure 12

In generated code edit some value as shown below

```
package pack1;
```

```
import javax.jws.WebMethod;  
import javax.jws.WebParam;  
import javax.jws.WebService;
```

```
/**
```

```
*
```

```
* @author roseindia
```

```
*/
```

```
@WebService()
```

```
public class square {
```

```
    @WebMethod(operationName = "area")
```

```
    public String area(@WebParam(name = "side")
```

```
        int side) {
```

```
        return "area of square of side"+side+" is "+(side*side);
```

```
    }
```

```
}
```

Enabling Security in Web Service>

- Open the square.java Web Service program in design view
- Select the Check box Service Secure as shown below in Fig 13.

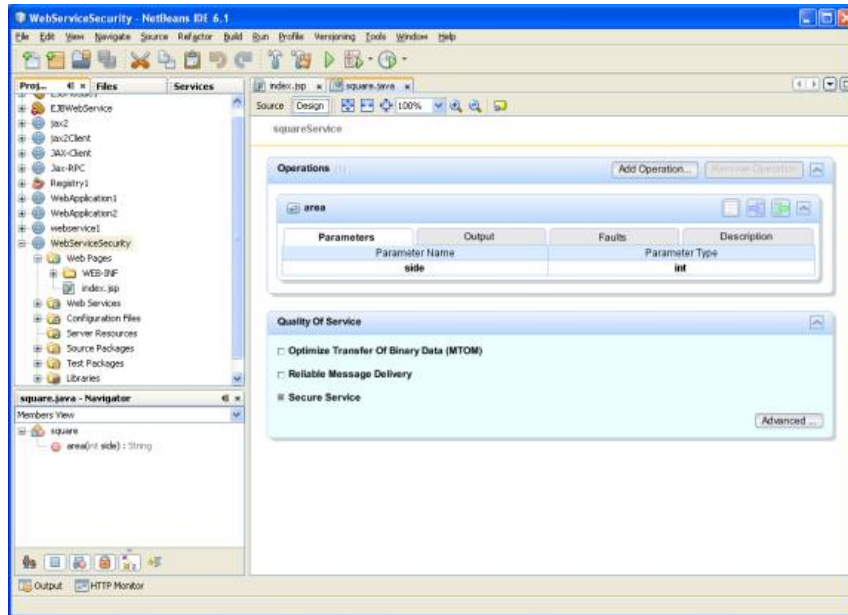


Figure. 13

- Click on Advanced Button
- It opens the dialog box for security
- Select the Secure Service check box as shown below in Figure 14.

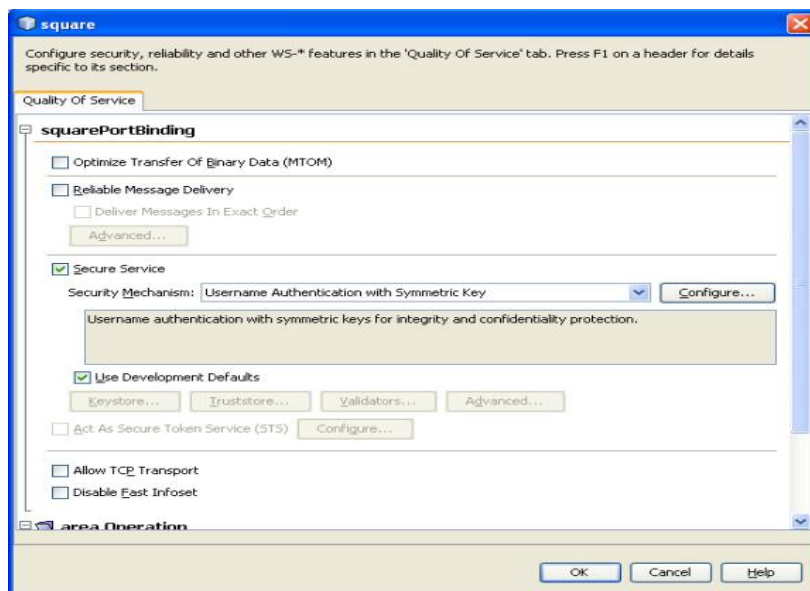


Figure. 14

- After selecting the secure service, it gets applied for Input Message and Output Message parts
- Scroll down, Input Message and Output Message comes as shown below in Figure 15

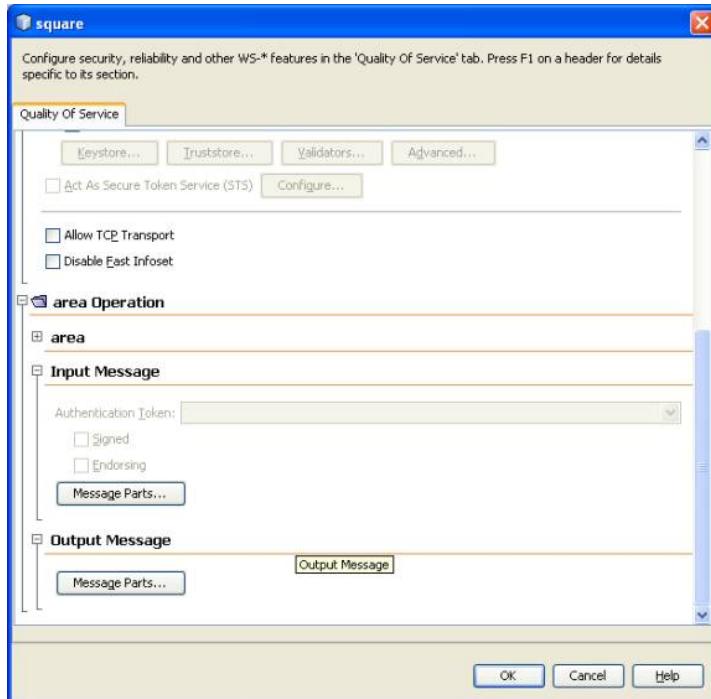


Figure 15

- Click on Message Parts button inside Input Message
- It opens dialog box where all values of Sign is selected
- In Encrypt one value for Body Message part is selected as shown below in Figure 16
- Click on Message Parts button inside Output Message
- It opens dialog box where all values of Sign is selected
- In Encrypt one value for Body Message part is selected as shown below in Figure 17

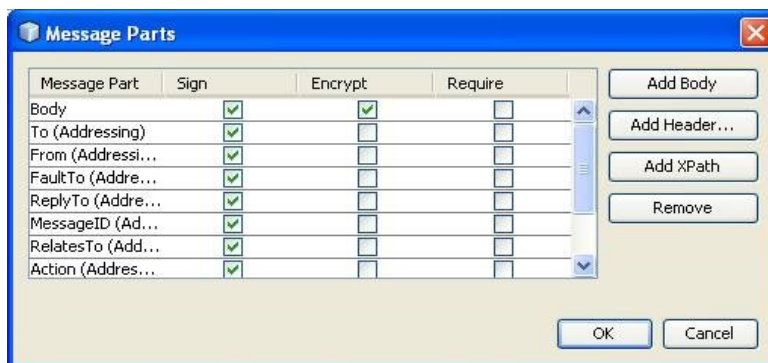


Figure 16

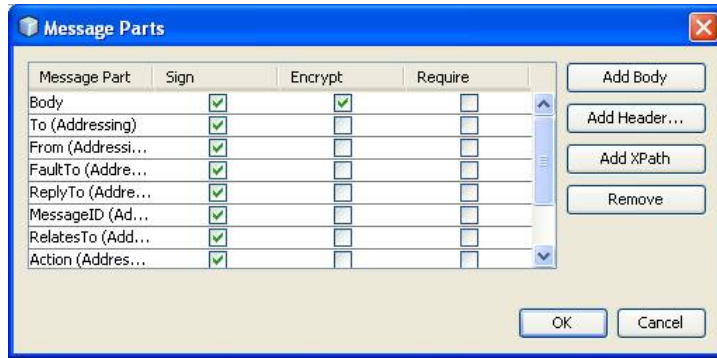


Figure 17

Testing Web Service

- Right Click on the Web Service square
- Select Test Web Service as shown below in Figure 18.

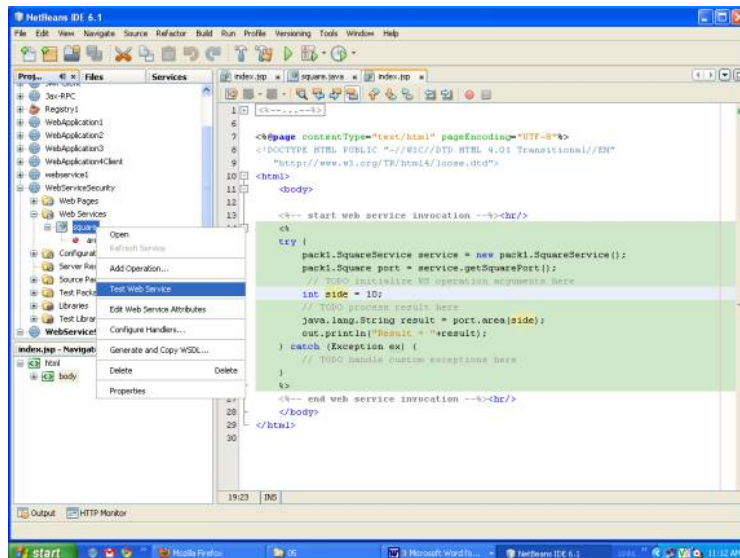


Figure.18

- It gives message that secured Web Service doesn't have support of tester feature as shown below in Figure 19.



Figure. 19

Web Service Client

- For above created Web Service a Client is created

Grid and Cloud Computing Lab-Manual

- Take a new Web Application Project
- Give it a name WebServiceSecurity-Client
- Click on Next button as shown below in Figure 20.

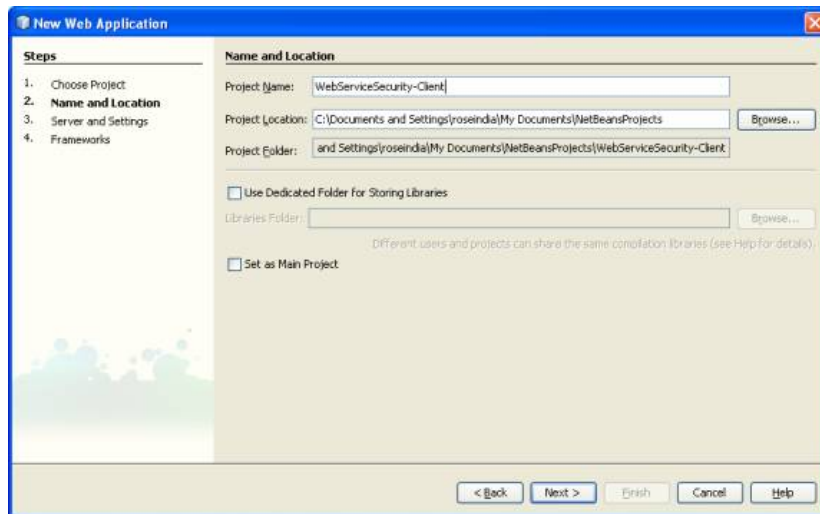


Figure. 20

- Select the Glassfish server
- Click on Next Button as shown below in Figure 21

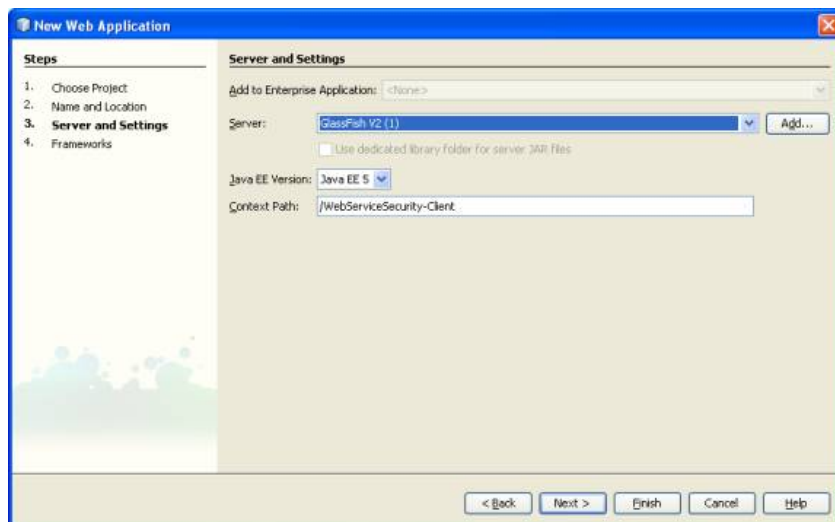


Figure. 21

- A Web Service Client project is created
- Develop a java class for Web Service Client
- Right Click on the WebServiceSecurity-Client select NewàWeb Service Client

As shown below in Figure 22.

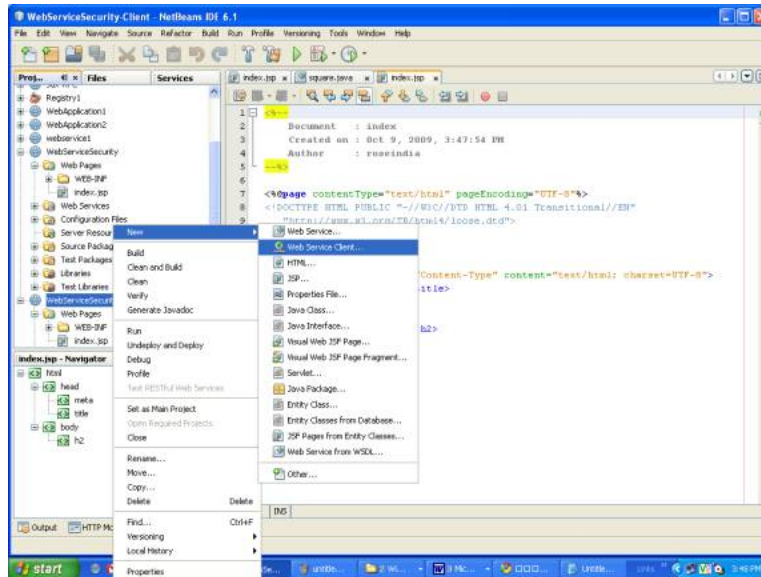


Figure. 22

- It opens a dialog box for WSDL and Client location
- Click on the browse button as shown in Figure 23.

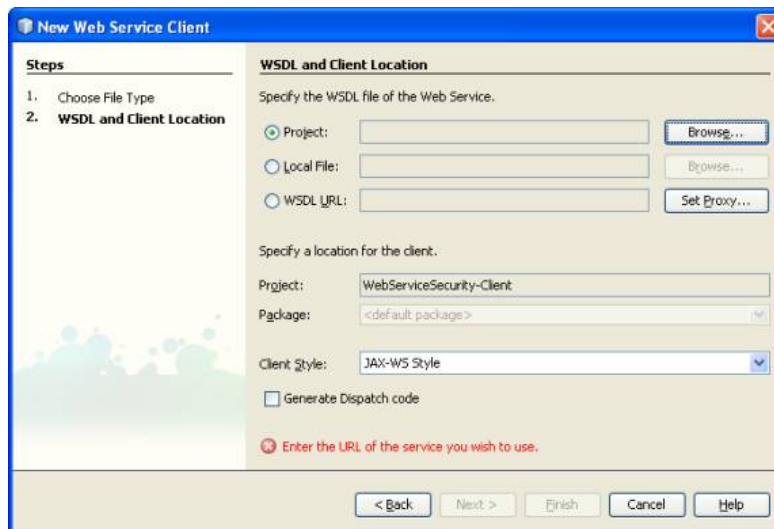


Figure. 23

- In the pop up dialog box select the Web Service
- Select WebServiceSecurity as square
- Click on OK as shown below in Figure 24.

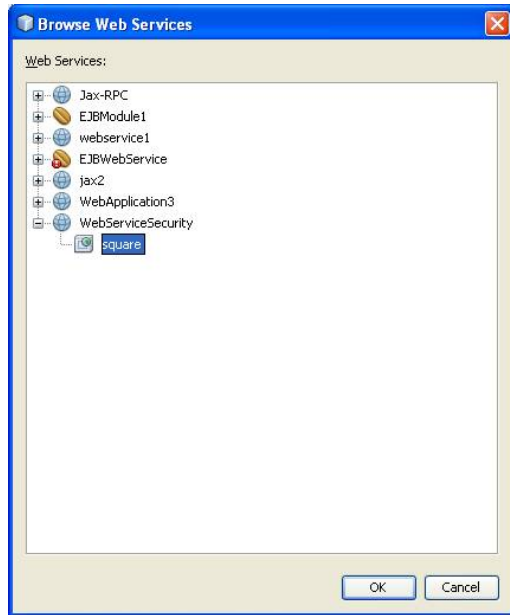


Figure 24

- This creates a Web Service References Directory
- In side that it creates squareService, squarePort and area method

As shown below in Figure 25.

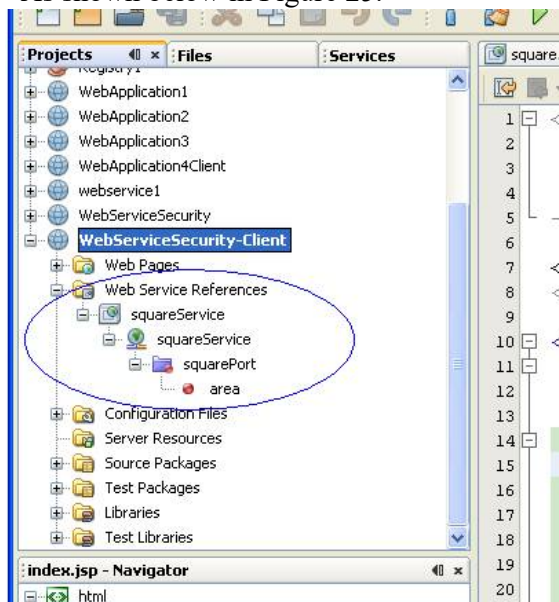


Figure. 25

Calling Web Service Client Resources

- Right Click in default created index.jsp
- Select Web Service Client ResourcesàCall Web Service Operation as shown below in Figure 26.

- It creates the required code in the index.jsp

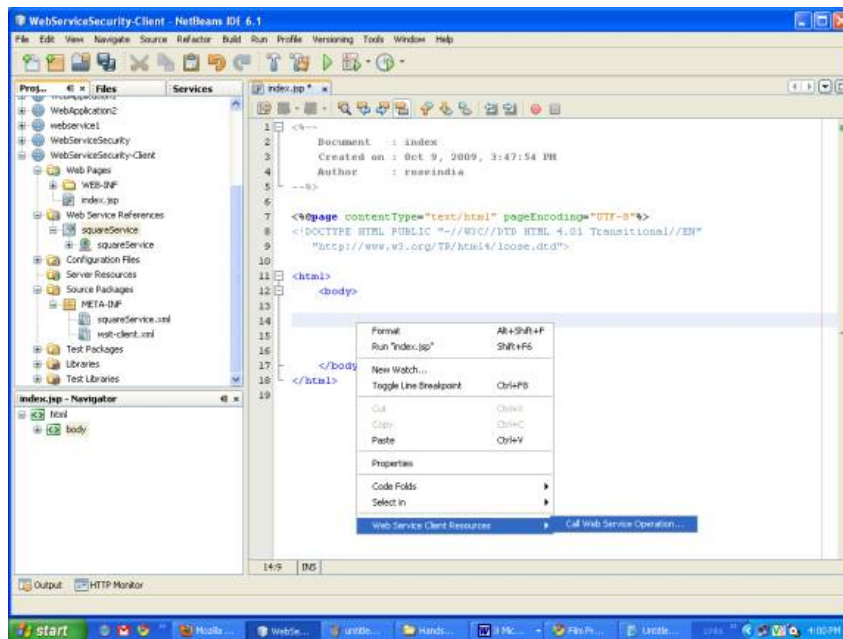


Figure. 26

- Edit the index.jsp and give the value `int side=10;`

```
<html>
<body>

<%-- start web service invocation --%><hr/>
<%
try {
pack1.SquareService service = new pack1.SquareService();
pack1.Square port = service.getSquarePort();
// TODO initialize WS operation arguments here
int side = 10;
// TODO process result here
java.lang.String result = port.area(side);
out.println("Result = "+result);
} catch (Exception ex) {
// TODO handle custom exceptions here
}
%>
<%-- end web service invocation --%><hr/>
</body>
</html>
```

Edit Web Service Attributes

- As the web service is secure its client should be edited
- Right Click on squareService in Web Service References

Grid and Cloud Computing Lab-Manual

- Select Edit Web Service Attributes as shown below in Figure 27.

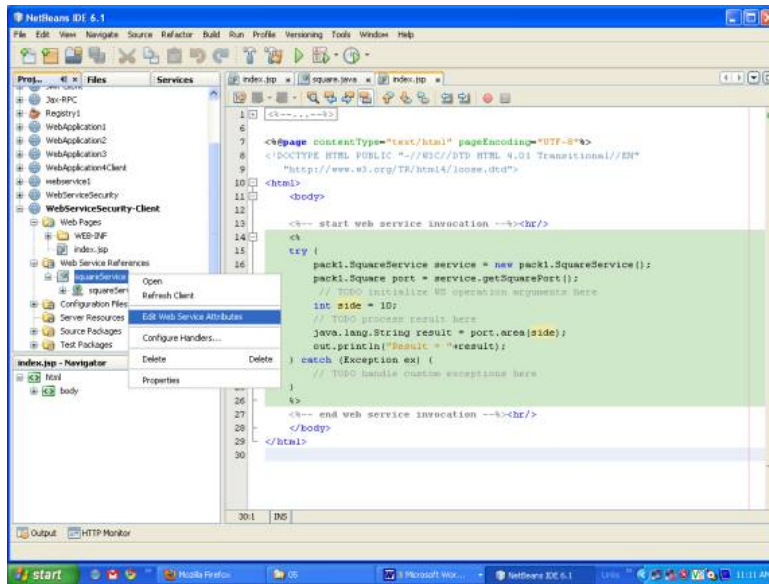


Figure. 27

- In the opened dialog box select the Quality Of Service
- Check the Use development defaults in security
- Click on OK as shown in Figure 28.

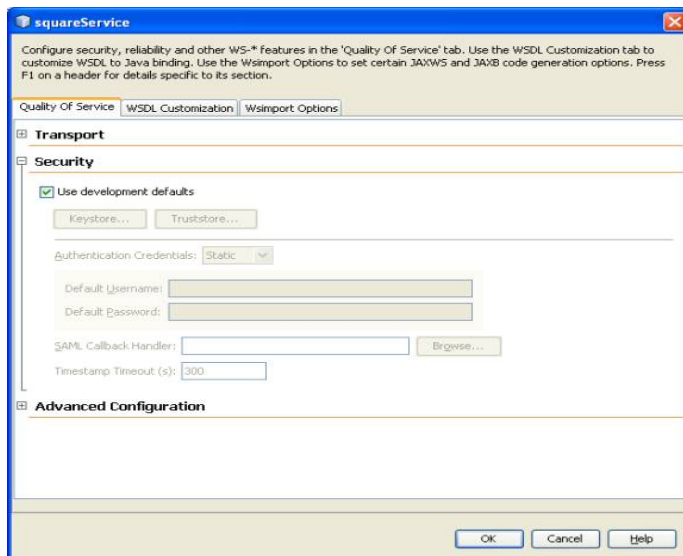


Figure. 28

Running Client Web service >

- Deploy the WebServiceSecurity-Client project
- Right Click in the index.jsp
- Select run index.jsp as shown below in Figure 29.

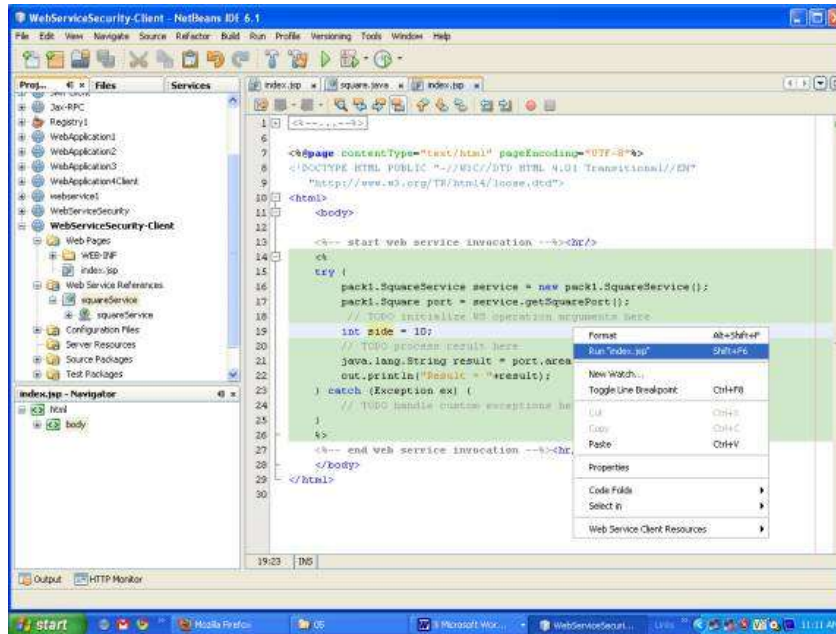
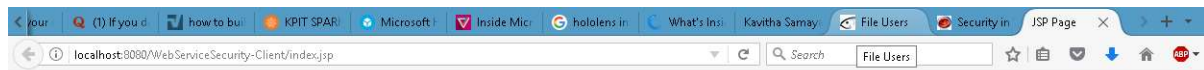


Figure. 29

- It deploys the project on the server and run it in the browser
- It executes and give the output as shown below in Fig 30.

OUTPUT



Hello World!

Result = area of square of side 10 is 100

CONCLUSIONS

Thus a secure web service has been created and verified successfully.

6. Find procedure to run the virtual machine of different configuration. Check how many virtual machines can be utilized at particular time.

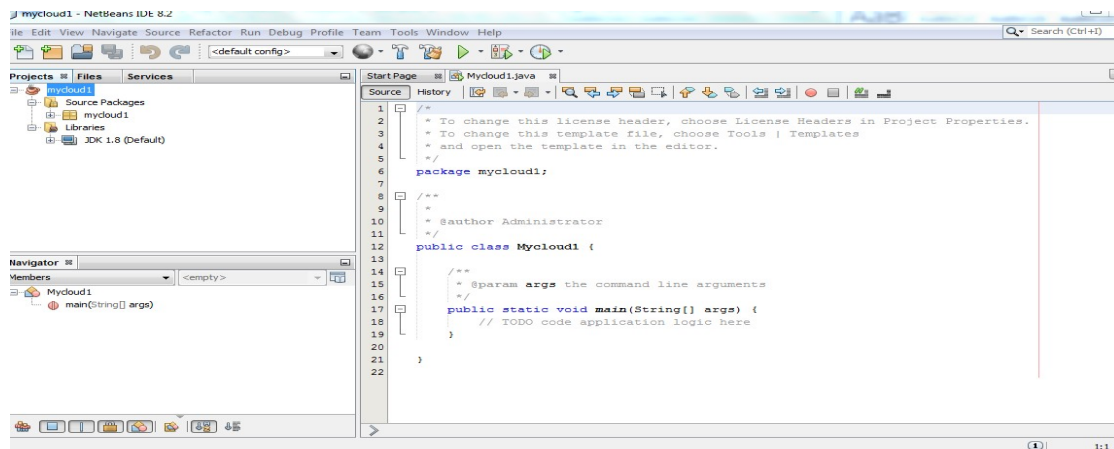
AIM:

TO DEVELOP A VIRTUAL MACHINE USING CLOUD SIM.

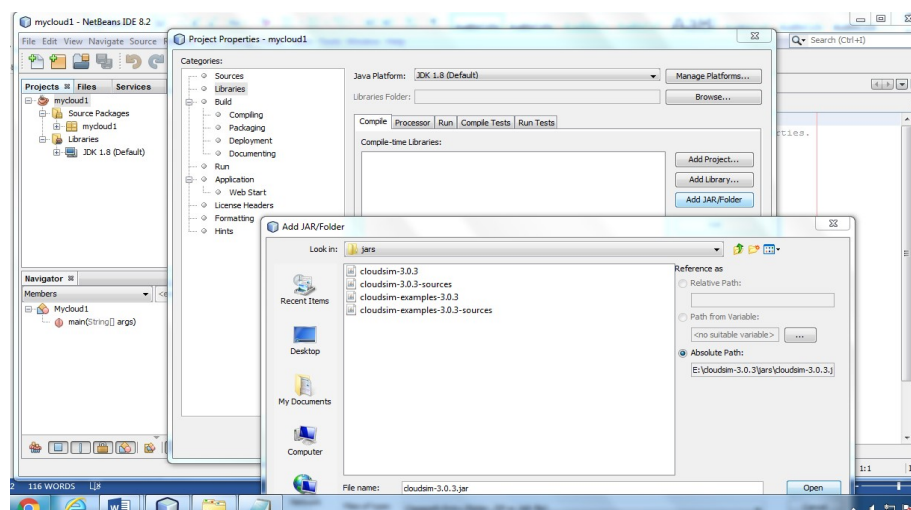
ALGORITHM:

Step1: Copy the cloudsim folder in E:drive

Step2: Open the netbeans, Newproject->Java->JavaApplication->project name->mycloud->finish

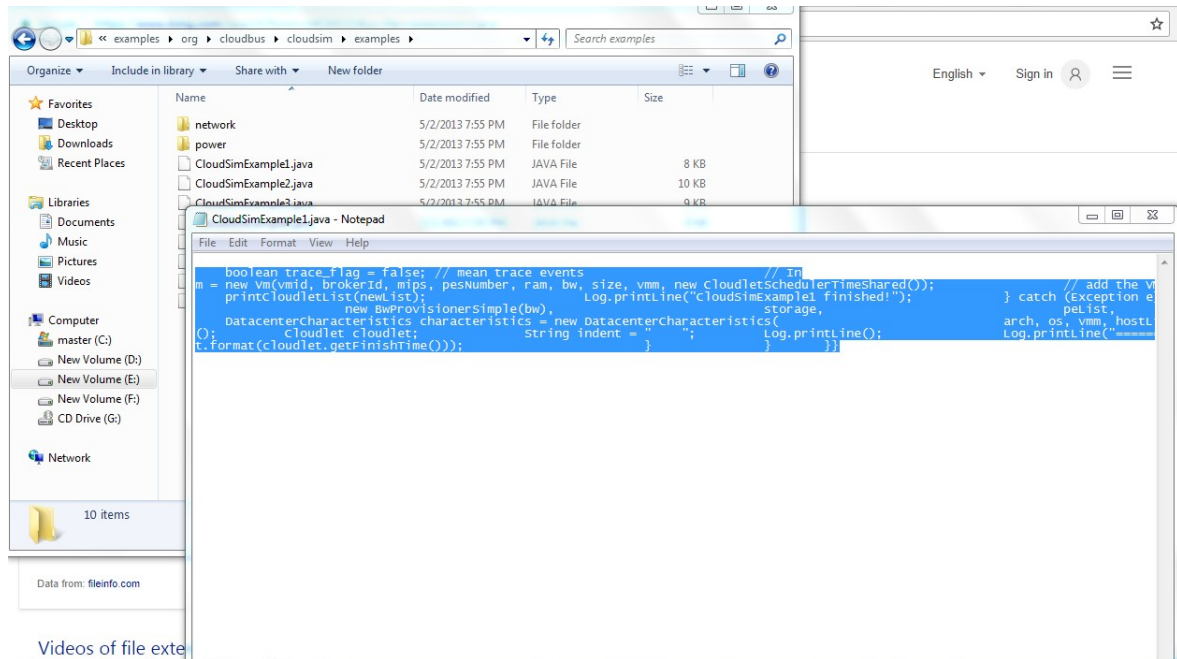


Step 3:right click on mycloud->properties->libraries->add jar folder->browse (mycomputer->E drive-> unzipped cloudsim folder->jar->cloudsim3.0.3.jar file-> click ok)



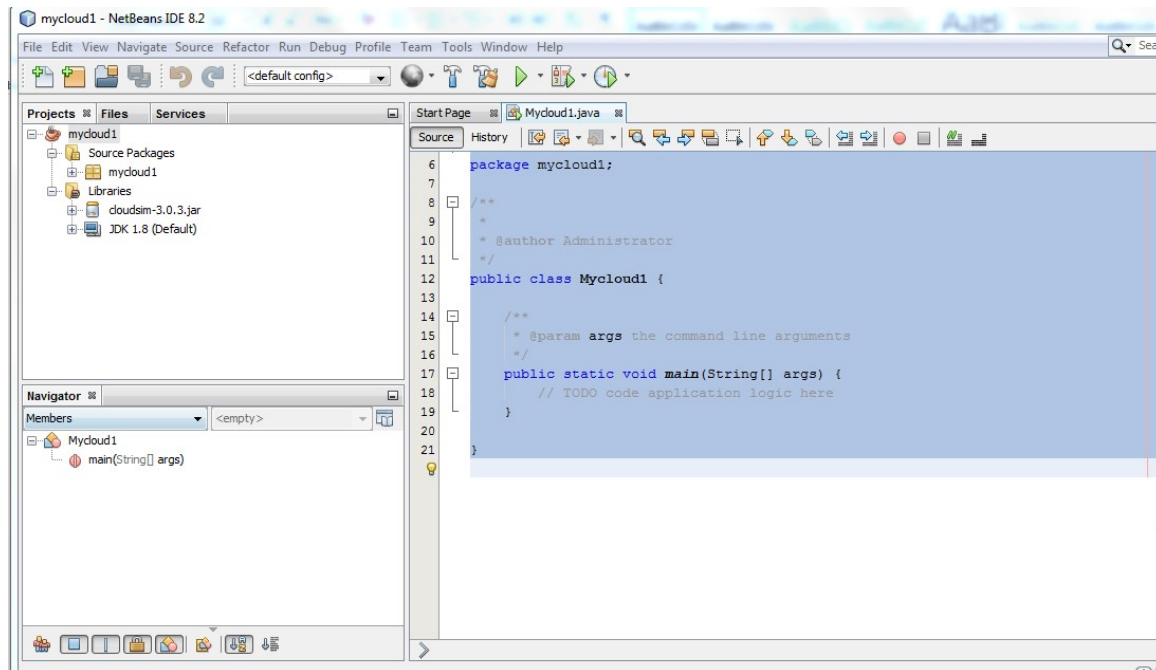
Step 4: Go to drive E:\cloudsim-3.0.3\examples\org\cloudbus\cloudsim\examples\Cloudsimexample1.java

Grid and Cloud Computing Lab-Manual



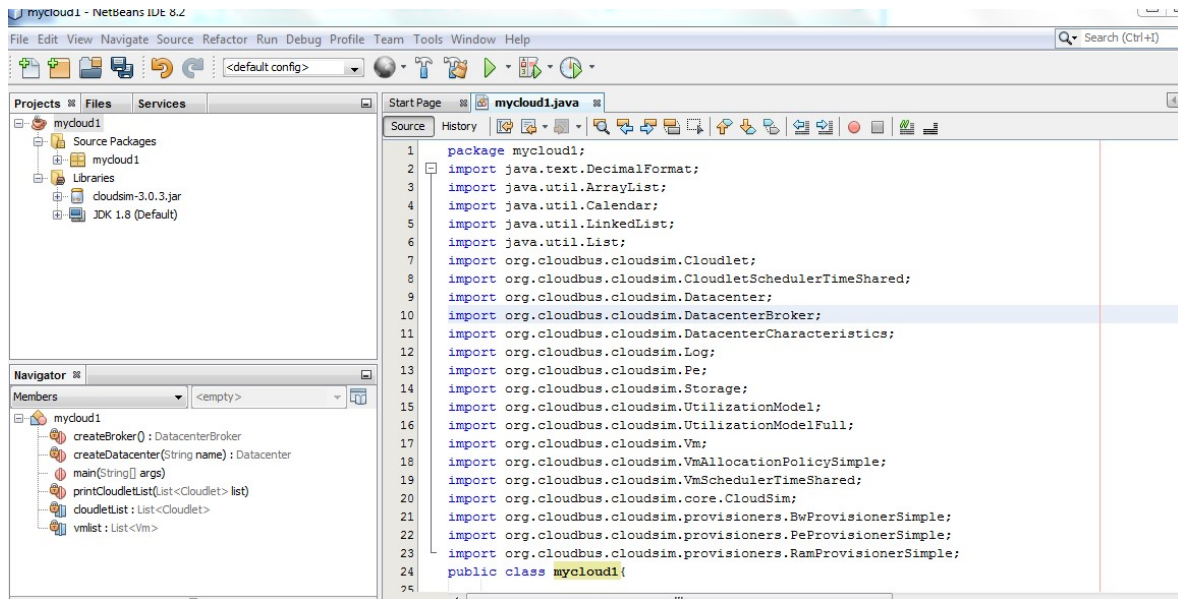
Step5: Right click on Cloudsimexample1.java and copy the code

Step 6: Open Source page in netbeans, delete the code and paste the code (Step5)

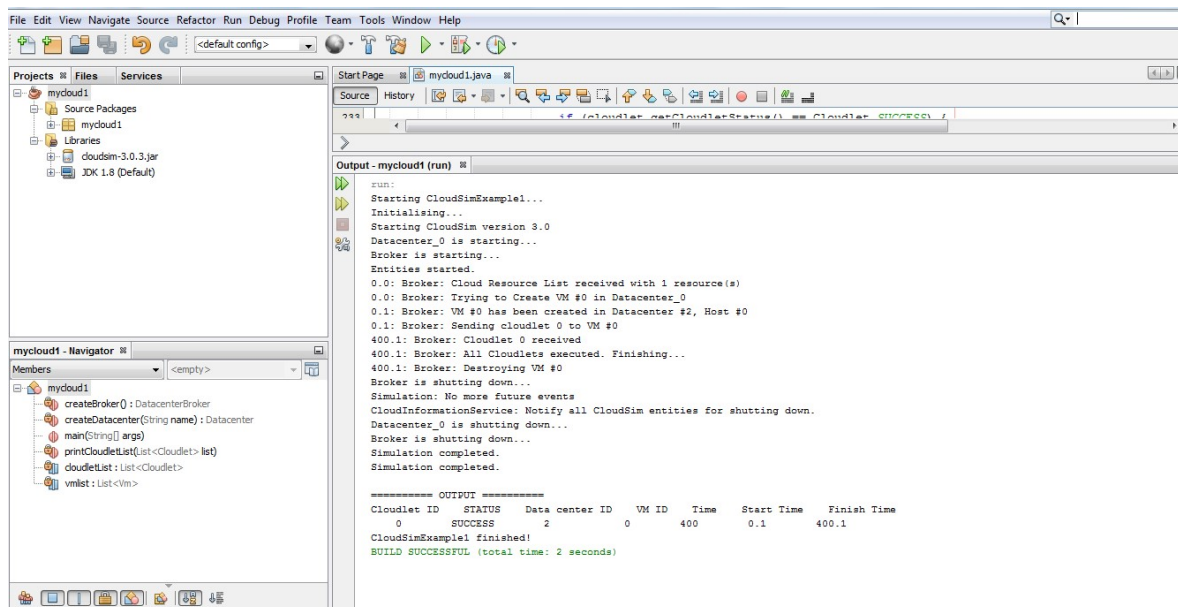


Step 7: change the package name and class name as mycloud.

Grid and Cloud Computing Lab-Manual



Step 8: Run ,Output Vm is created



CONCLUSIONS

Thus the virtual machine has been created successfully.

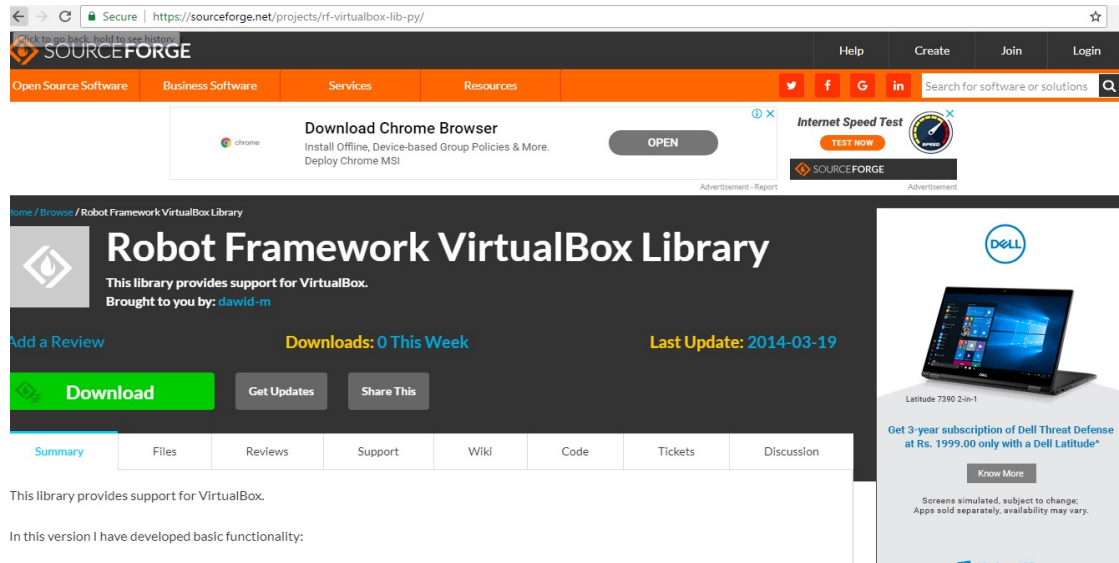
7. Find a procedure to attach Virtual Box to a Virtual Machine

AIM:

To attach a virtual box to a virtual machine.

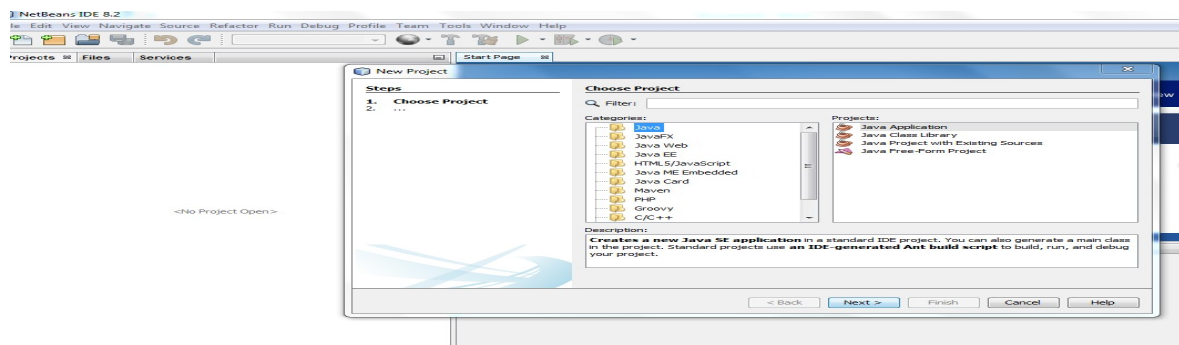
ALGORITHM:

Step1:Goto :<https://sourceforge.net/projects/robot-framework-virtualbox-lib-py/>



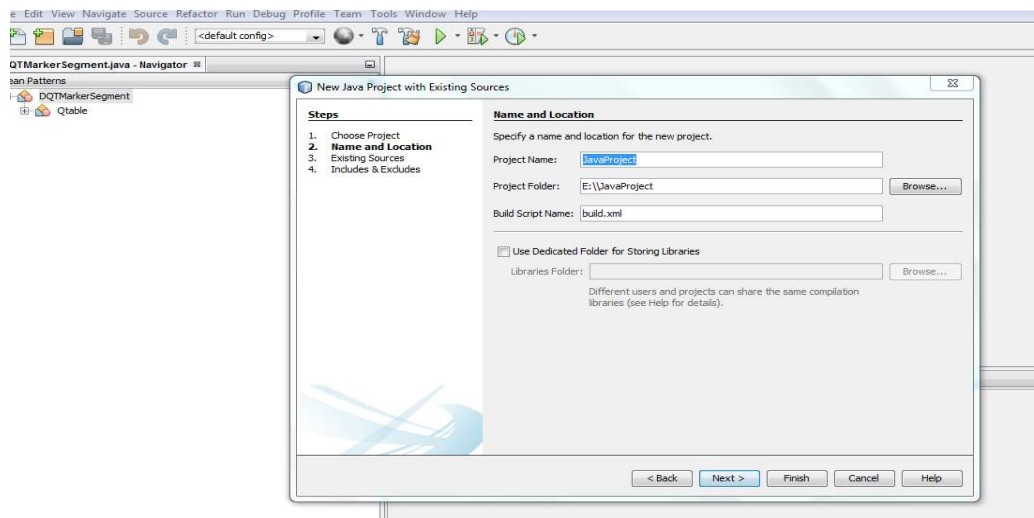
Step2:download and copy the folder in E:drive

Step3:open the IDE→new project→java→ Java project with Existing resources

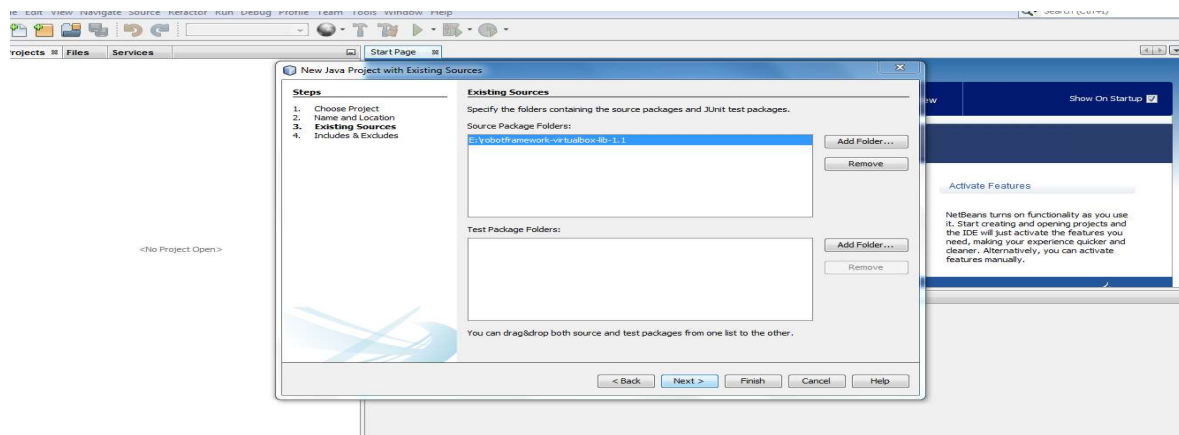


Grid and Cloud Computing Lab-Manual

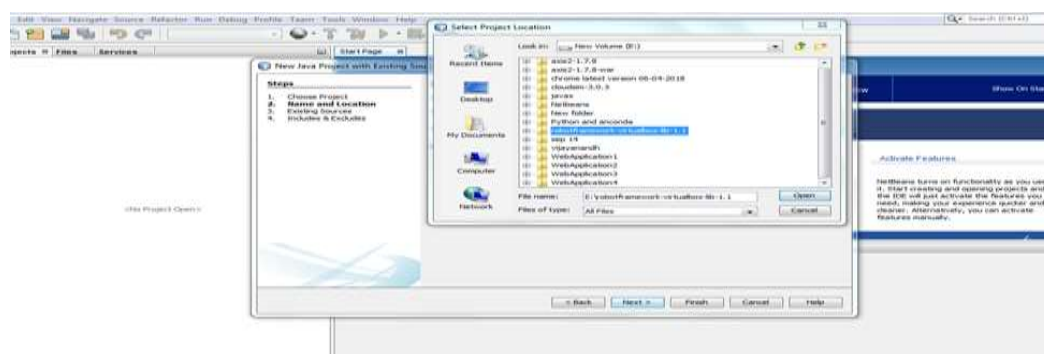
Step4:Goto Name and Location→Project Name→Next



Step5:Existing Sources→Add Folder→E:\ Robot framework virtualbox lib-1.1

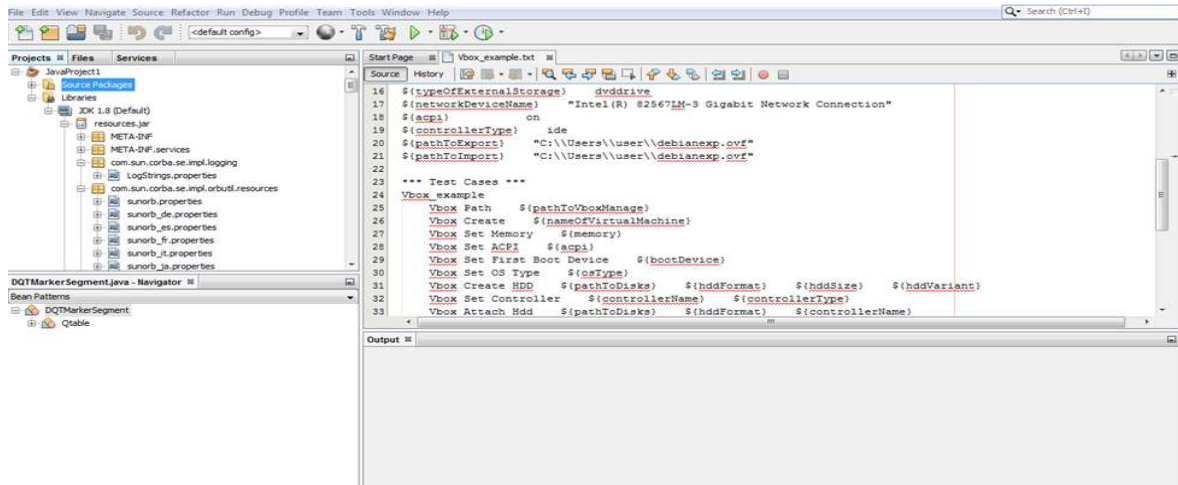


Step6:Click Add Folder→Mycomputer→E drive→Robot framework virtualbox lib-1.1→open

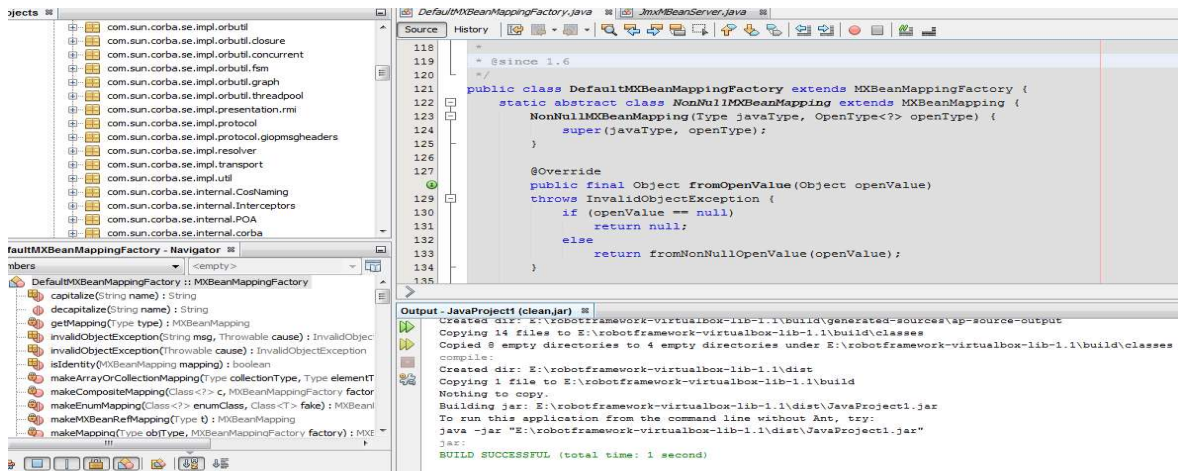


Grid and Cloud Computing Lab-Manual

Step7:Attached the virtual box in the virtual machine



Step8: Attached the virtual block to the virtual machine in the cloud sim



CONCLUSIONS

Thus virtual box has been attached with virtual machine.

8. Install a C compiler in the virtual machine and execute a sample program.

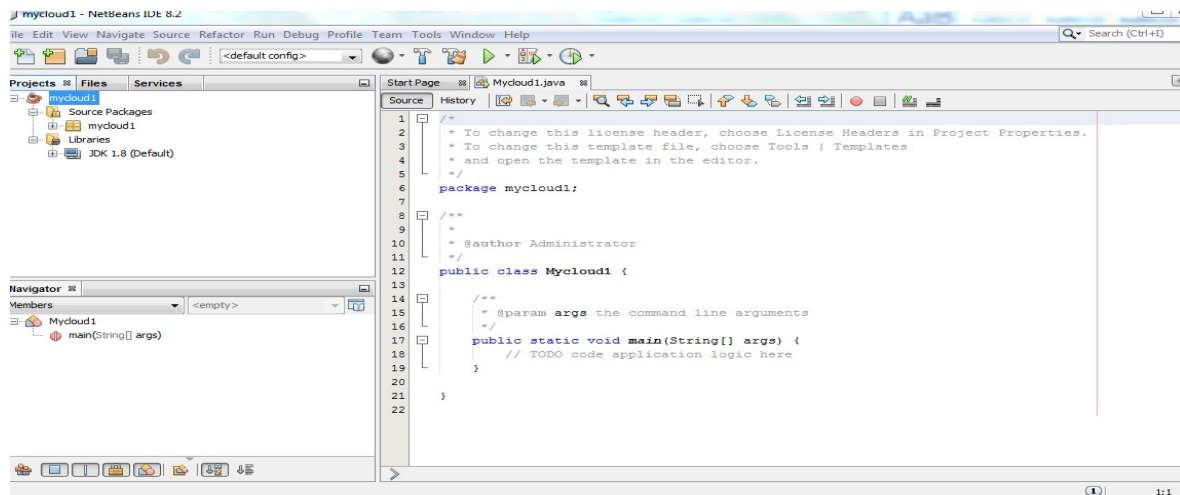
Aim:

To Install a C compiler in the virtual machine and execute a sample program

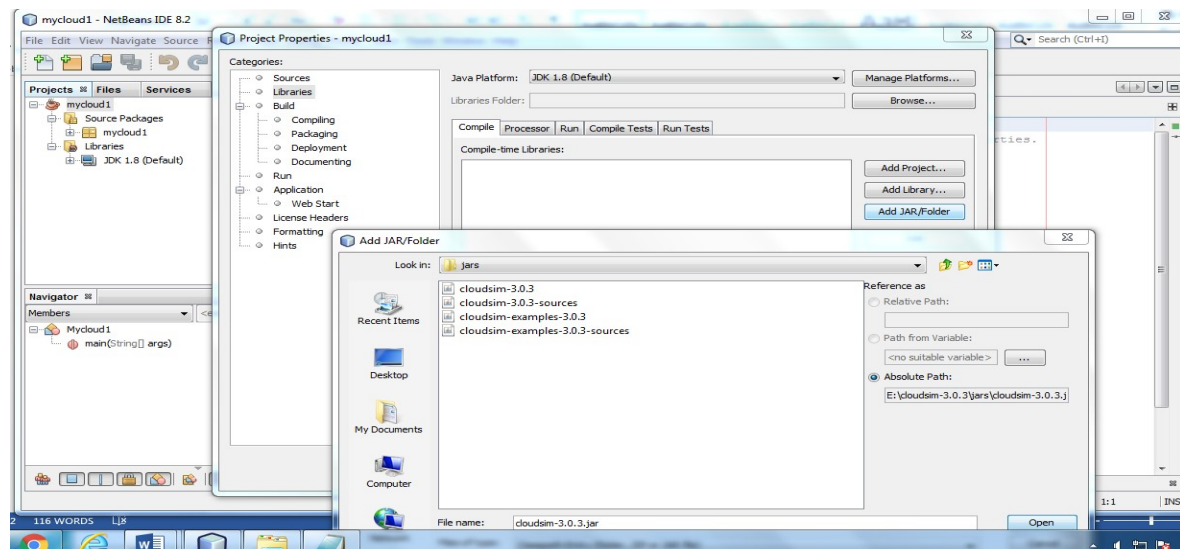
Algorithm:

Step1: Copy the cloudsim folder in E:drive

Step2: Open the netbeans, Newproject->C/C++->CApplication->project name->mycloud->finish



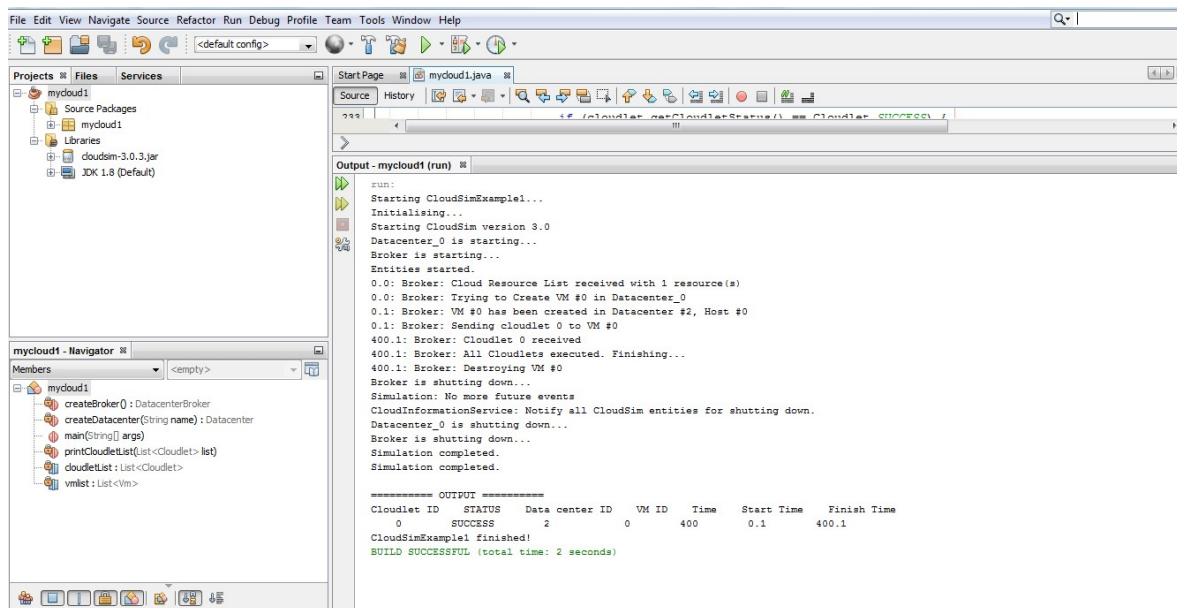
Step 3:right click on mycloud->properties->libraries->add jar folder->browse (mycomputer->E drive-> unzipped cloudsim folder->jar->cloudsim3.0.3.jar file-> click ok)



Grid and Cloud Computing Lab-Manual

Step 7: change the package name and class name as mycloud.

Step 8: Run ,Output Vm is created



```
run:
Starting CloudSimExample1...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: Sending cloudlet 0 to VM #0
400.1: Broker: Cloudlet 0 received
400.1: Broker: All Cloudlets executed. Finishing...
400.1: Broker: Destroying VM #0
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down..
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

===== OUTPUT =====
Cloudlet ID   STATUS   Data center ID   VM ID   Time   Start Time   Finish Time
0            SUCCESS   2                0       400   0.1          400.1
CloudSimExample1 finished!
BUILD SUCCESSFUL (total time: 2 seconds)
```

Conclusions

Thus the c compiler has been implemented successfully.

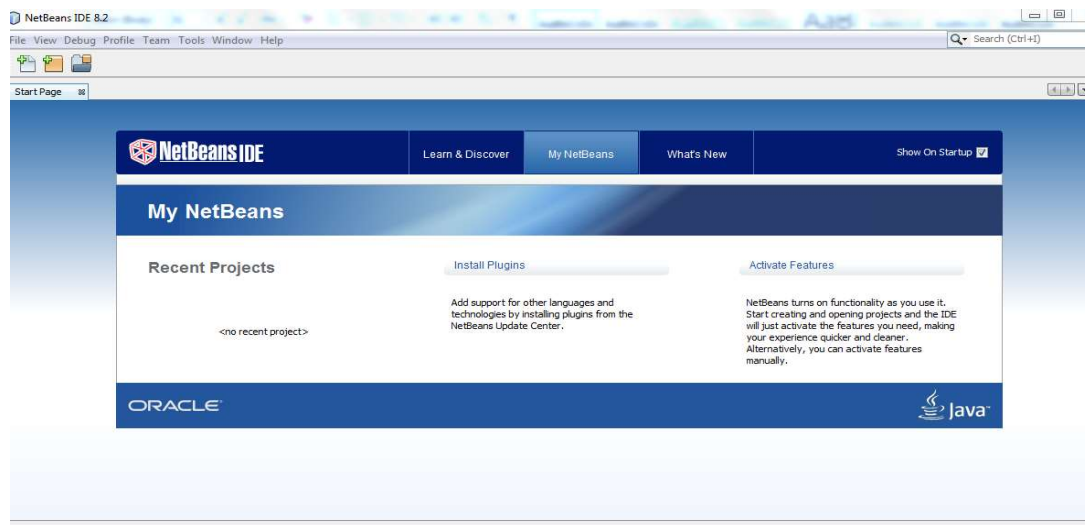
9. Show a Virtual Machine Migration based on the certain condition from one node to another

Aim:

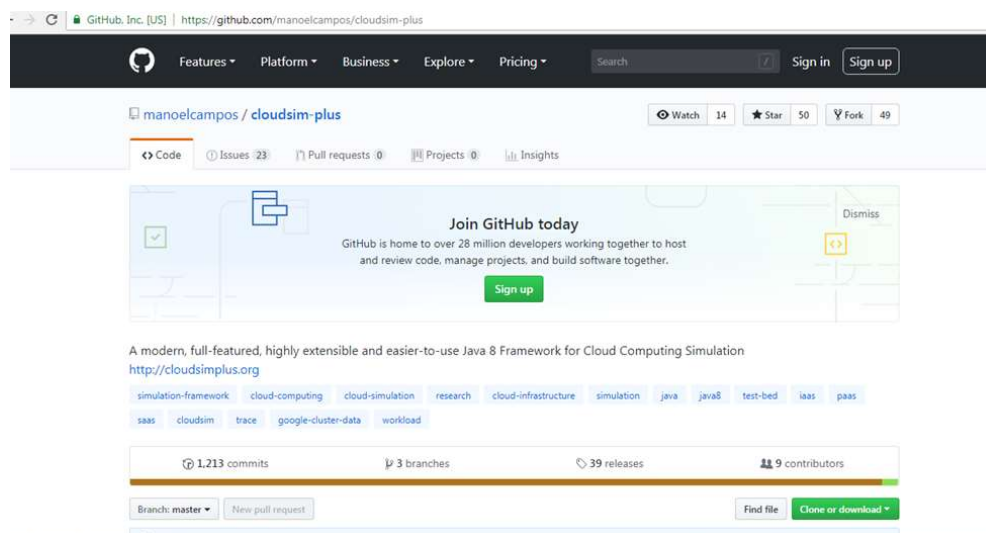
To perform virtual machine migration based on the certain condition from one node to another.

Algorithm:

Step1: Open NetBeans8.2

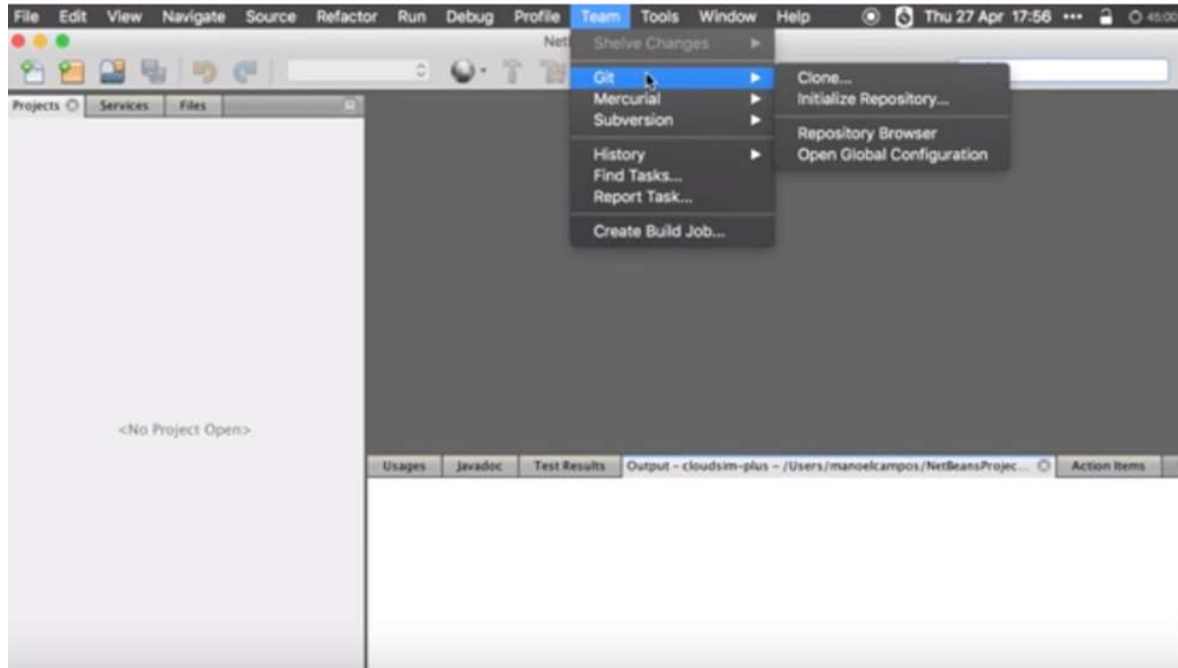


Step2: Open the url: <https://github.com/manoelcampos/cloudsim-plus>

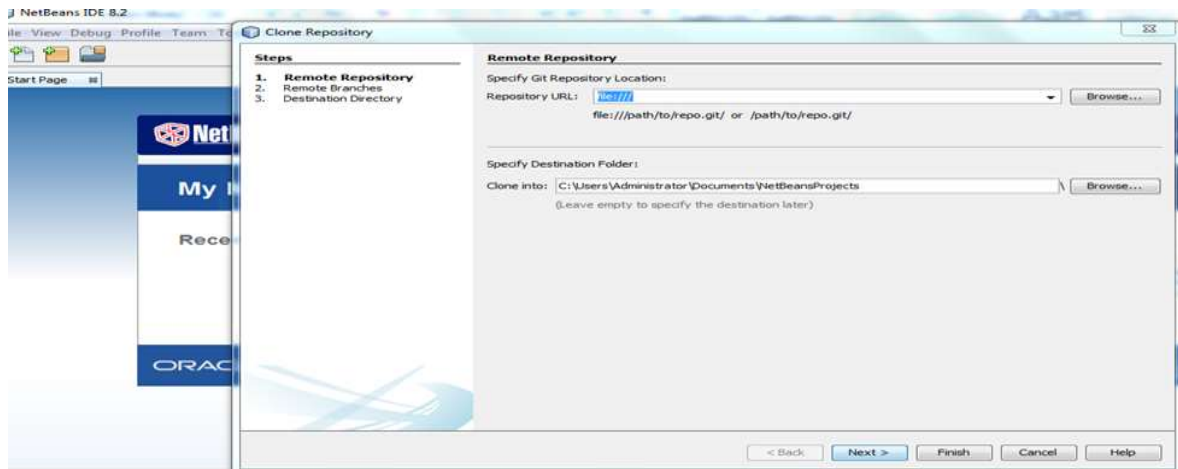


Step3: In IDE →Goto Team-> Git-> Clone

Grid and Cloud Computing Lab-Manual

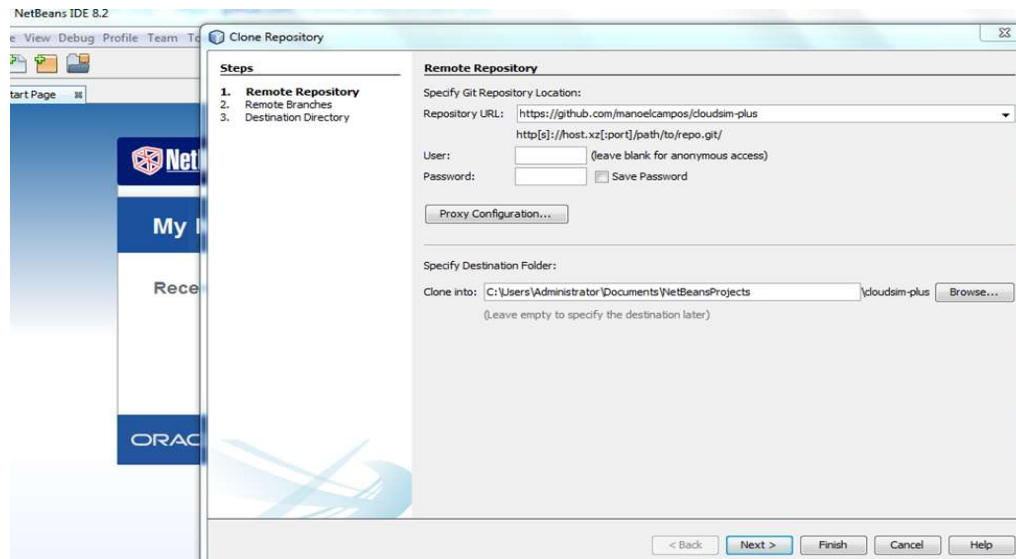


Step4:Remote Repository→Repository URL appears on the screen

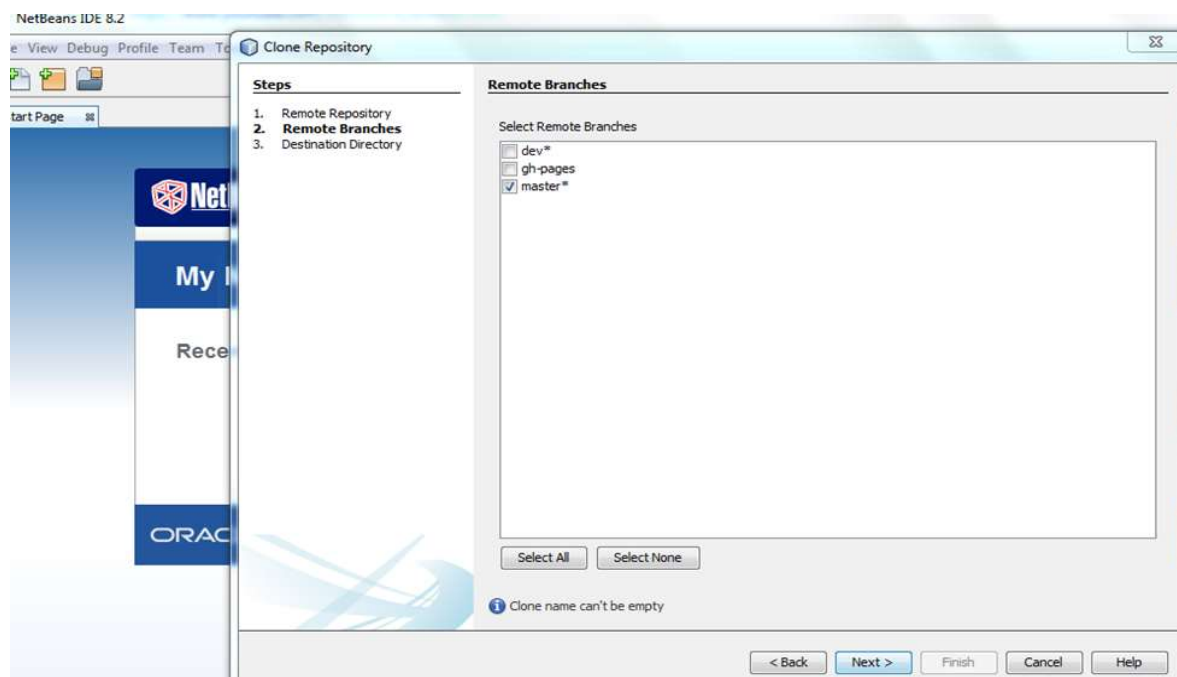


Step5:Type the url in the Repository URL <https://github.com/manoelcampos/cloudsim-plus> and Click---> Next

Grid and Cloud Computing Lab-Manual

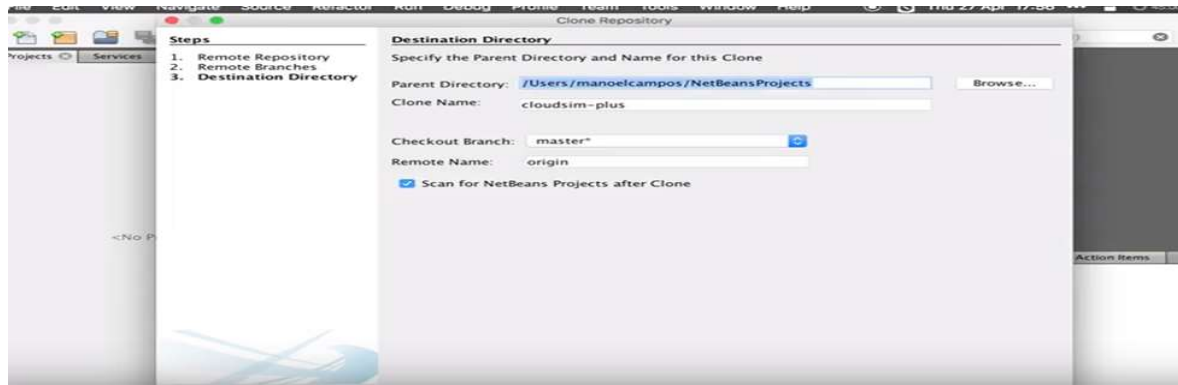


Step6:Now it is connected to Git Repository →Remote Repository opens on the screen ,click-→Next

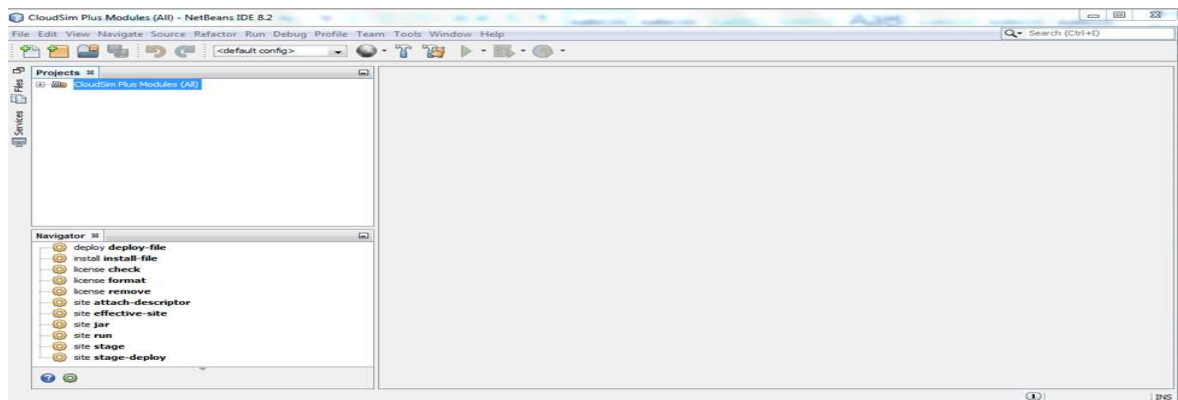
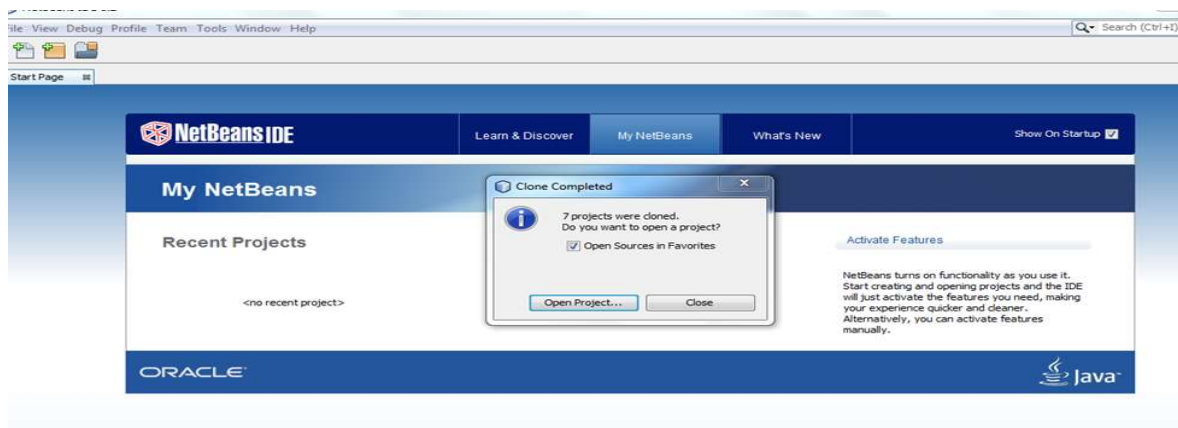


Step7:Destination Directory→Browse
→C:\Users\Administrator\Documents\NetBeansprojects→click finish

Grid and Cloud Computing Lab-Manual

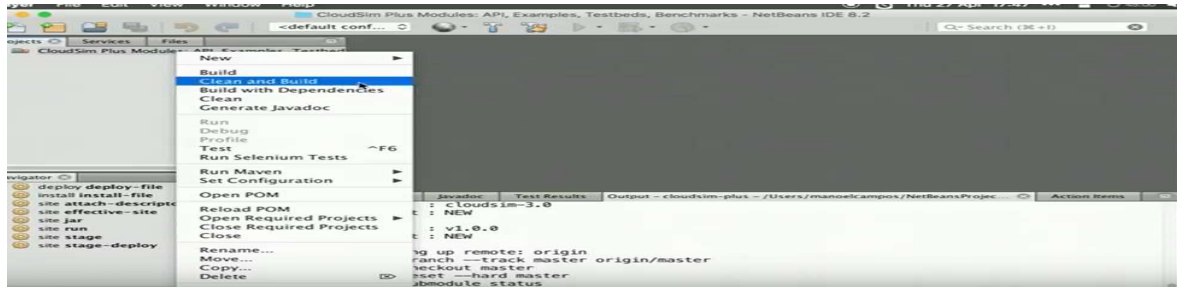


Step8: Open Project→cloudplus module

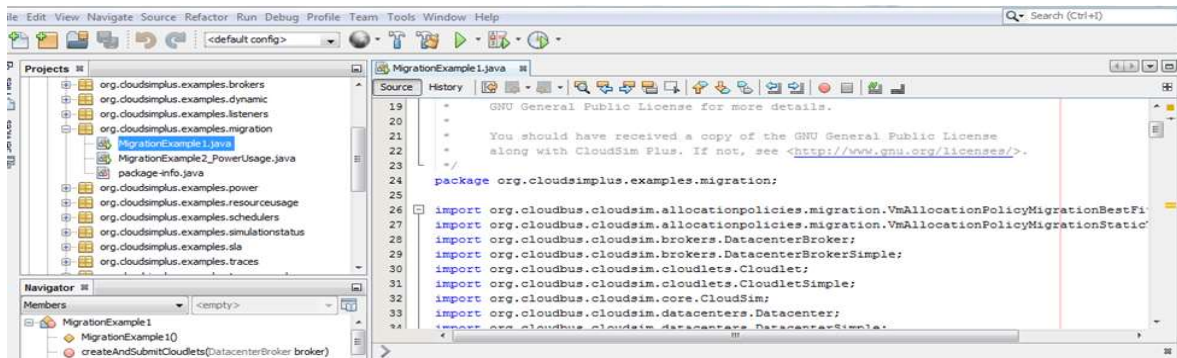


Step:9 Right click cloudsim module→clean and build

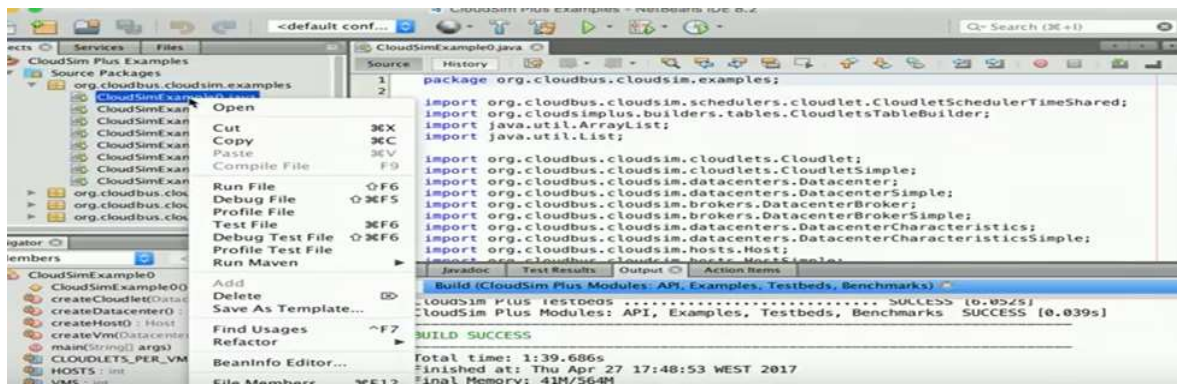
Grid and Cloud Computing Lab-Manual



Step:10 Click cloudsimplus→modules→cloudsimplus examples→source packages→click cloud migration.java



Step:11 Click migration →Run file →SUCESSFULLY MIGRATED



Conclusions

Thus virtual migration has been performed successfully.

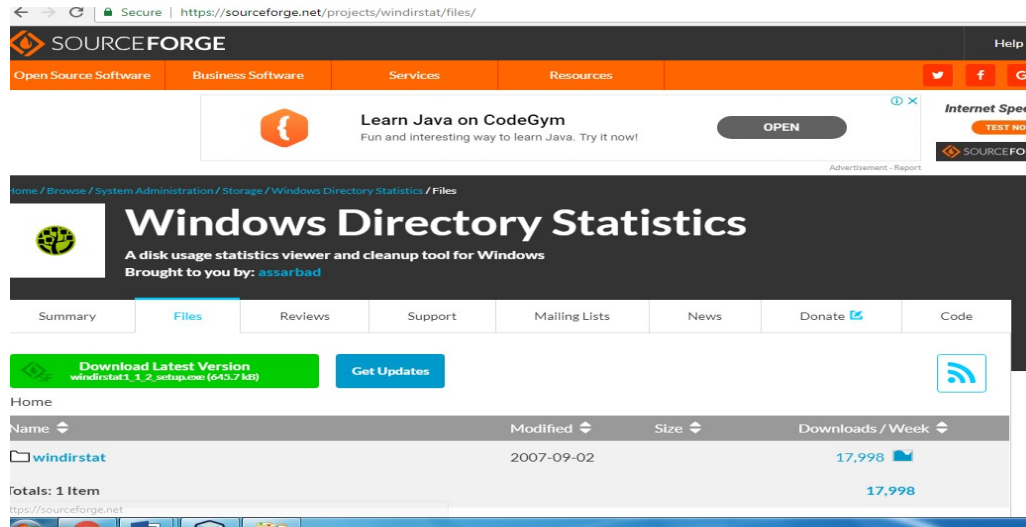
10. Find procedure to install storage controller and interact with it.

Aim:

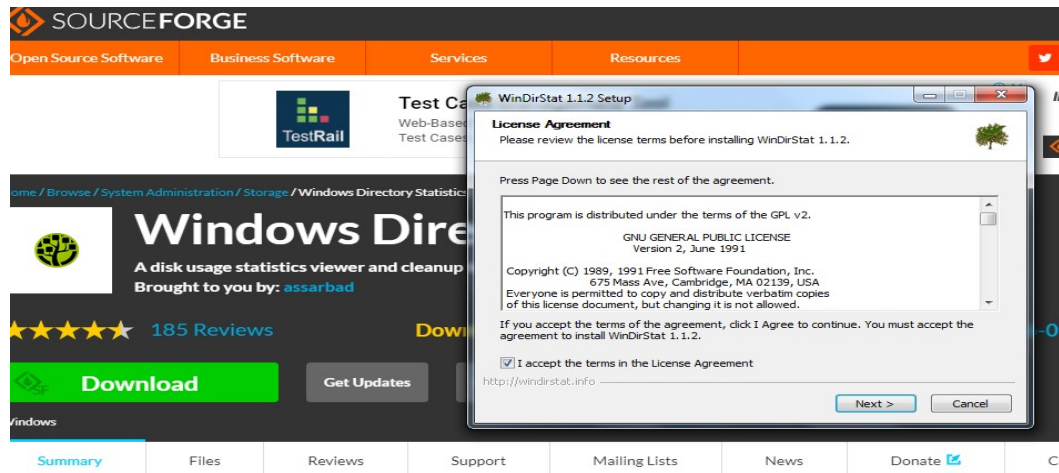
To install storage controller and interact with it.

Algorithm:

Step1:Goto <https://sourceforge.net/projects/windirstat/files/>

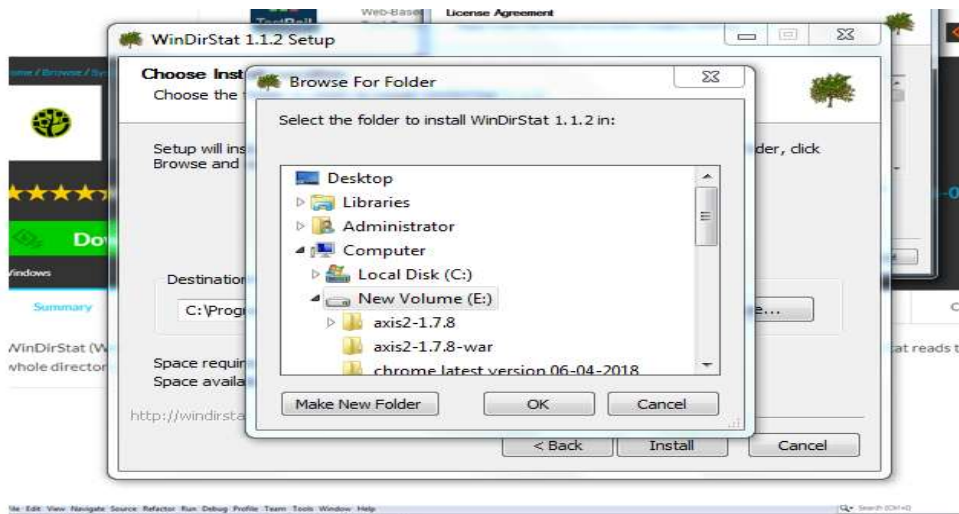


Step2: download and Run the Winstart1_1_2



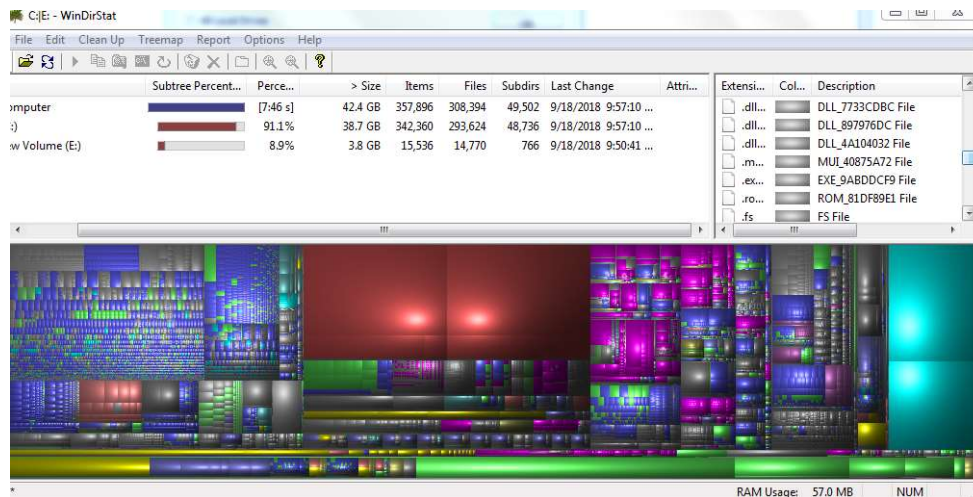
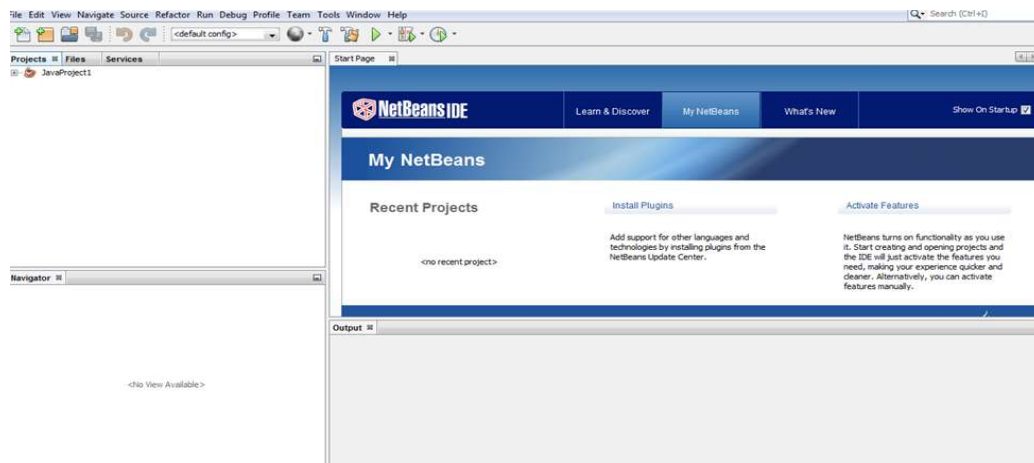
WinDirStat (Windows Directory Statistics) is a disk usage statistics viewer and cleanup tool for Windows. On start up, WinDirStat reads the whole directory tree once and then presents it in three useful views:

Grid and Cloud Computing Lab-Manual

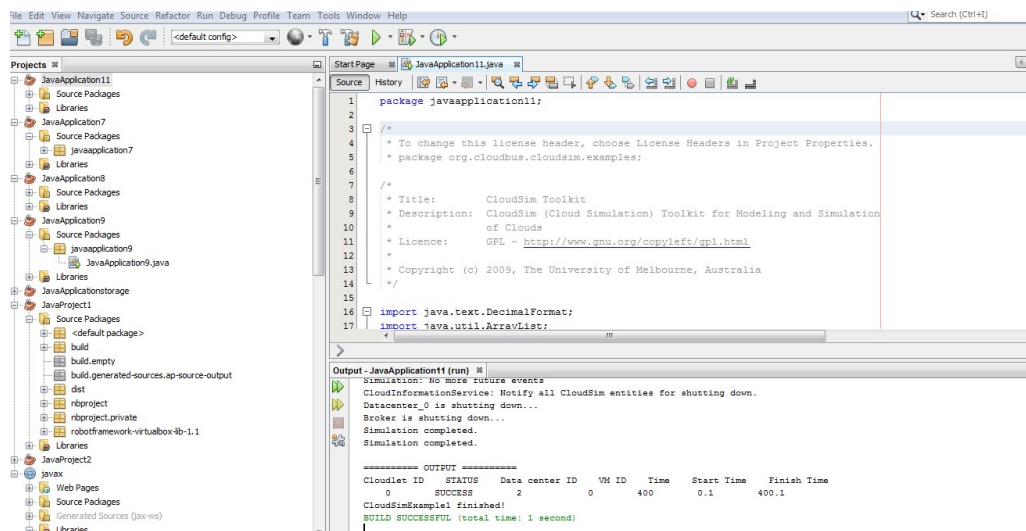
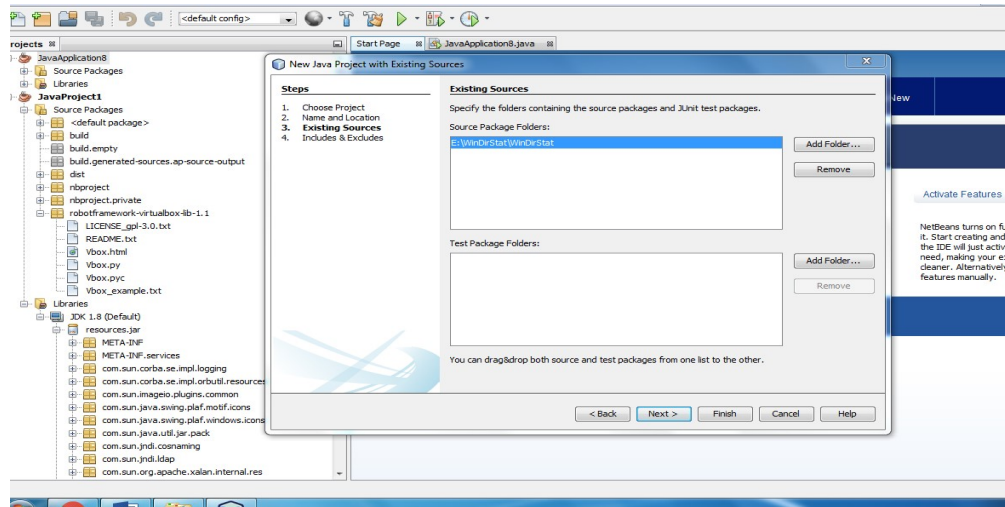


Step4: choose the E:drive and Do the Installation

Step5:Open IDE-→SELECT New Project



Grid and Cloud Computing Lab-Manual



Conclusions

Thus the storage controller has been implemented successfully.

11. Write a word count program to demonstrate the use of Map and Reduce tasks

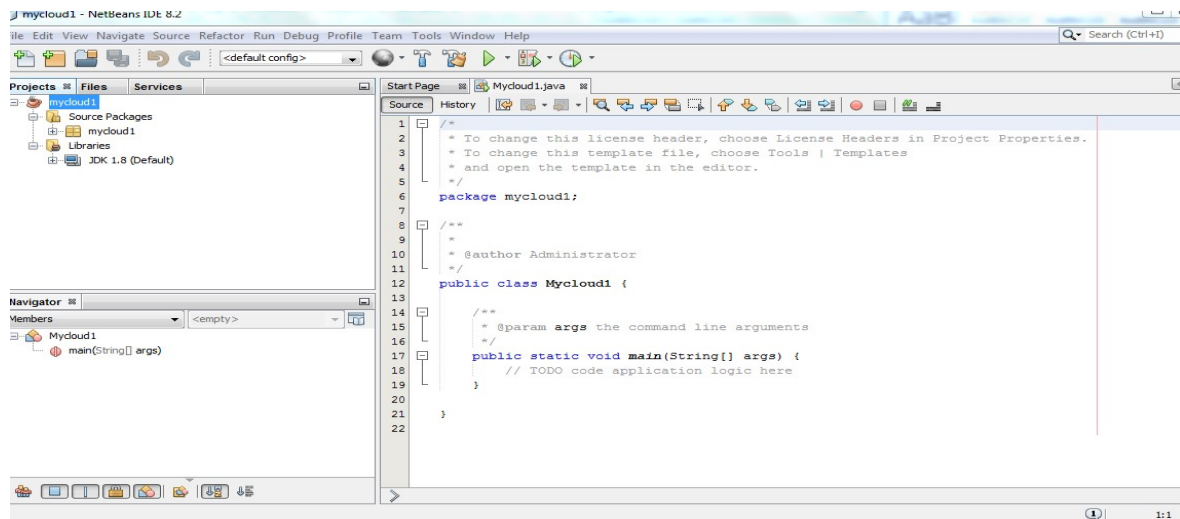
AIM:

To write a wordcount program to demonstrate the use of map and reduce using hadoop framework.

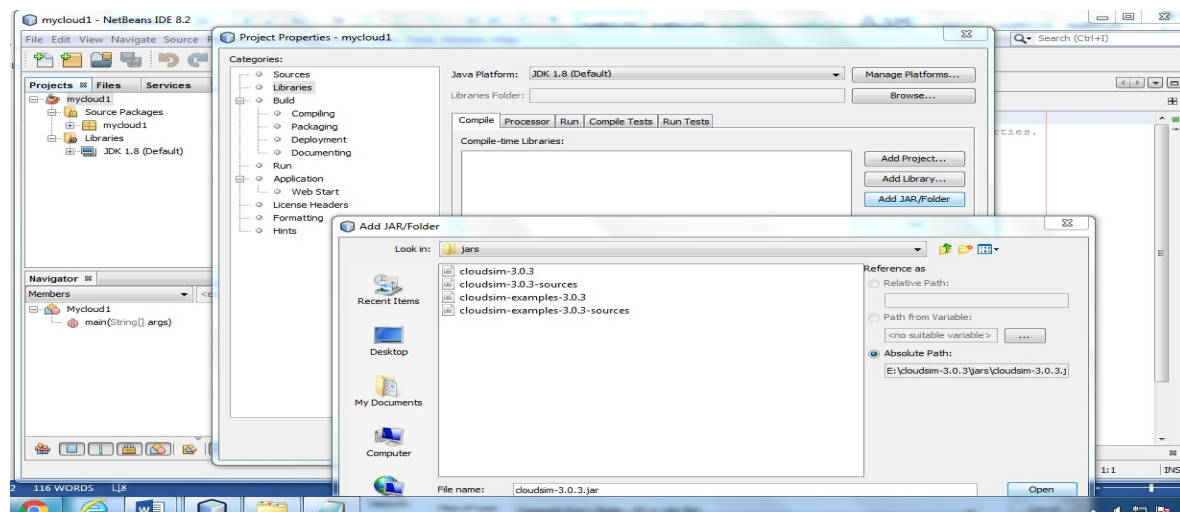
ALGORITHM:

Step1: copy the cloudsim folder in E:drive

Step2: Open the netbeans, Newproject->Java->JavaApplication->project name->hadoop->finish



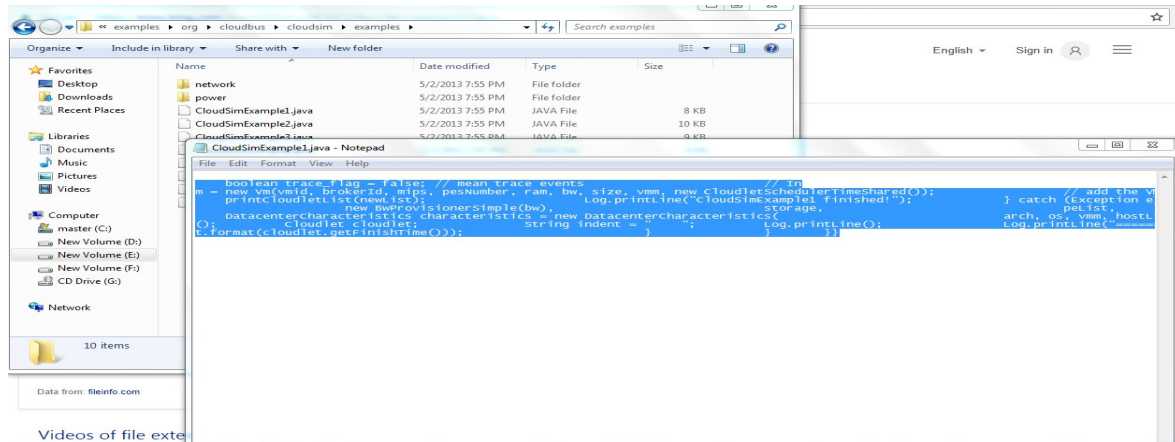
Step 3:right click on mycloud->properties->libraries->add jar folder->browse (mycomputer->E drive-> unzipped cloudsim folder->jar->cloudsim3.0.3.jar file-> click ok)



Step 4: Go to drive E:\cloudsim-3.0.3\examples\org\cloudbus\cloudsim\examples\mapper.java

Step 5: Go to drive E:\cloudsim-3.0.3\examples\org\cloudbus\cloudsim\examples\reducer.java

Step 6: Go to drive E:\cloudsim-3.0.3\examples\org\cloudbus\cloudsim\examples\hadoop.java



Step 7: run the program

Step 8: get the result

PROGRAM:

MapClass.java

```
import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class MapClass extends Mapper<LongWritable, Text, Text, IntWritable>{
```

```
private final static IntWritable one = new IntWritable(1);  
private Text word = new Text();
```

```
@Override
```

```
protected void map(LongWritable key, Text value,  
    Context context)
```

```
throws IOException, InterruptedException {
```

```
String line = value.toString();
```

```
StringTokenizer st = new StringTokenizer(line, " ");
```

```
while(st.hasMoreTokens()){
```

```
    word.set(st.nextToken());
```

```
    context.write(word,one);
```

```
}
```

```
}
```

```
}
```

ReduceClass.java

```
import java.io.IOException;
```

```
import java.util.Iterator;
```

```
import org.apache.hadoop.io.IntWritable;
```

```
import org.apache.hadoop.io.Text;
```

```
import org.apache.hadoop.mapreduce.Reducer;
```

```
public class ReduceClass extends Reducer{
```

```
    @Override
```

```
    protected void reduce(Text key, Iterable values, Context context)
```

```
        throws IOException, InterruptedException {  
    int sum = 0;  
  
    Iterator valuesIt = values.iterator();  
  
    while(valuesIt.hasNext()){  
        sum = sum + valuesIt.next().get();  
    }  
  
    context.write(key, new IntWritable(sum));  
}  
}
```

hadoop.java

```
import org.apache.hadoop.conf.Configured;  
  
import org.apache.hadoop.fs.Path;  
  
import org.apache.hadoop.io.IntWritable;  
  
import org.apache.hadoop.io.Text;  
  
import org.apache.hadoop.mapreduce.Job;  
  
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  
  
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  
  
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;  
  
import org.apache.hadoop.util.Tool;  
  
import org.apache.hadoop.util.ToolRunner;  
  
public class WordCount extends Configured implements Tool{  
    public static void main(String[] args) throws Exception{  
        int exitCode = ToolRunner.run(new WordCount(), args);  
        System.exit(exitCode);  
    }  
  
    public int run(String[] args) throws Exception {  
        if (args.length != 2) {
```



```
        System.err.printf("Usage: %s needs two arguments, input and output
files\n", getClass().getSimpleName());
        return -1;
    }
    Job job = new Job();
    job.setJarByClass(WordCount.class);
    job.setJobName("WordCounter");
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    job.setOutputFormatClass(TextOutputFormat.class);
    job.setMapperClass(MapClass.class);
    job.setReducerClass(ReduceClass.class);
    int returnValue = job.waitForCompletion(true) ? 0:1;
    if(job.isSuccessful()) {
        System.out.println("Job was successful");
    } else if(!job.isSuccessful()) {
        System.out.println("Job was not successful");
    }
    return returnValue;
}
}
```

Output:

```
Hadoop 1
The 2
This 2
above 1
all 1
alphabets. 1
also 1
article 1
as 1
brown 1
code 1
contains 1
count 1
dog. 1
ecosystem. 1
english 1
example 4
examples 1
famous 1
file 1
for 2
fox 1
geek 1
hello 1
is 3
java 1
jumps 1
knows 1
```

Conclusions:

Thus the map and reduce has been implemented successfully.

12. MOUNT THE ONE NODE HADOOP CLUSTER USING FUSE

Aim:

To write a program to use the API's of Hadoop to interact with it

Procedure:

Interfaces

Following are the important interfaces:

- Client<-->ResourceManager

By using YarnClient objects.

- ApplicationMaster<-->ResourceManager

By using AMRMClientAsync objects, handling events asynchronously by

AMRMClientAsync.CallbackHandler

- ApplicationMaster<-->NodeManager

Launch containers. Communicate with NodeManagers by using NMClientAsync objects, handling container

events by NMClientAsync.CallbackHandler

Writing a Simple Yarn Application

Writing a simple Client

- The first step that a client needs to do is to initialize and start a YarnClient.
- `YarnClient yarnClient = YarnClient.createYarnClient();`
- `yarnClient.init(conf);`
- `yarnClient.start();`
- Once a client is set up, the client needs to create an application, and get its application id.
- `YarnClientApplication app = yarnClient.createApplication();`
- `GetNewApplicationResponse appResponse = app.getNewApplicationResponse();`
- The response from the YarnClientApplication for a new application also contains information about the cluster such as the minimum/maximum resource capabilities of the cluster. This is required so that to ensure that you can correctly set the specifications of the container in which the ApplicationMaster would be launched.
- The main crux of a client is to setup the ApplicationSubmissionContext which defines all the information
- needed by the RM to launch the AM. A client needs to set the following into the context:
- Application info: id, name

- Queue, priority info: Queue to which the application will be submitted, the priority to be assigned for the application.
 - User: The user submitting the application
 - ContainerLaunchContext: The information defining the container in which the AM will be launched and run.
 - The ContainerLaunchContext, as mentioned previously, defines all the required information needed to run
 - the application such as the local *Resources (binaries, jars, files etc.), Environment settings (CLASSPATH
 - etc.), the Command to be executed and security T*okens (RECT).
 - The ApplicationReport received from the RM consists of the following:
 - General application information: Application id, queue to which the application was submitted, user who
 - submitted the application and the start time for the application.
 - ApplicationMaster details: the host on which the AM is running, the rpc port (if any) on which it is
 - listening for requests from clients and a token that the client needs to communicate with the AM.
 - Application tracking information: If the application supports some form of progress tracking, it can set a
 - tracking url which is available via ApplicationReport's getTrackingUrl() method that a client can look at to
 - monitor progress.
 - Application status: The state of the application as seen by the ResourceManager is available via Application Report#getYarnApplicationState. If the YarnApplicationState is set to FINISHED, the client should refer to ApplicationReport#getFinalApplicationStatus to check for the actual success/failure of the application task itself. In case of failures, ApplicationReport#getDiagnostics may be useful to shed some more light on the the failure.
 - If the ApplicationMaster supports it, a client can directly query the AM itself for progress updates via the host:rpcport information obtained from the application report. It can also use the tracking url obtained From the report if available.
 - In certain situations, if the application is taking too long or due to other factors, the client may wish to kill the application. YarnClient supports the killApplication call that allows a client to send a kill signal to the AM via the ResourceManager. An ApplicationMaster if so designed may also support an abort call via its rpc layer that a client may be able to leverage.
- yarnClient.killApplication(appId);

Writing an ApplicationMaster (AM)

- The AM is the actual owner of the job. It will be launched by the RM and via the client will be provided all the necessary information and resources about the job that it has been tasked with to oversee and complete.
- As the AM is launched within a container that may (likely will) be sharing a physical host with other containers, given the multi-tenancy nature, amongst other issues, it cannot make any assumptions of things like pre-configured ports that it can listen on.
- When the AM starts up, several parameters are made available to it via the environment. These include the ContainerId for the AM container, the application submission time and details about the NM (NodeManager) host running the ApplicationMaster. Ref ApplicationConstants for parameter names.
- All interactions with the RM require an ApplicationAttemptId (there can be multiple attempts per application in case of failures). The ApplicationAttemptId can be obtained from the AM's container id.
- There are helper APIs to convert the value obtained from the environment into objects.
- In setupContainerAskForRM(), the follow two things need some set up:
- Resource capability: Currently, YARN supports memory based resource requirements so the request should define how much memory is needed. The value is defined in MB and has to less than the max capability of the cluster and an exact multiple of the min capability. Memory resources correspond to physical memory limits imposed on the task containers. It will also support computation based resource (vCore), as shown in the code.
- Priority: When asking for sets of containers, an AM may define different priorities to each set. For example, the Map-Reduce AM may assign a higher priority to containers needed for the Map tasks and a lower priority for the Reduce tasks' containers. After container allocation requests have been sent by the application manager, containers will be launched asynchronously, by the event handler of the AMRMClientAsync client. The handler should implement AMRMClientAsync.CallbackHandler interface.
- When there are containers allocated, the handler sets up a thread that runs the code to launch
- containers. Here we use the name LaunchContainerRunnable to demonstrate. We will talk about the LaunchContainerRunnable class in the following part of this article.
- heNMClientAsync object, together with its event handler, handles container events. Including
- container start, stop, status update, and occurs an error.
- After the ApplicationMaster determines the work is done, it needs to unregister itself through the AMRMclient, and then stops the client.

Conclusions

Thus the procedure to mount the one node Hadoop cluster using FUSE was executed successfully