G.PULLAIAH COLLEGE OF ENGINEERING AND TECHNOLOGY

Accredited by NAAC with 'A' Grade and NBA Accredited (EEE, CSE & ECE)
Approved by AICTE, New Delhi & Permanently Affiliated to JNTUA, Ananthapuramu
(Recognized by UGC under 2(f) &12(B) & ISO 9001:2008 Certified Institution)
Nandikotkur Road, Venkayapalli, Kurnool-518452

LABORATORY RECORD WORK

GRID AND CLOUD COMPUTING LAB (15A05710)

IV YEAR B.TECH I SEMESTER

(Computer Science and Engineering)

NAME OF THE STUDENT	
ROLL NUMBER	
BRANCH & SECTION	


GPCET
Pioneering Innovative Education
G.PULLAIAH COLLEGE OF ENGINEERING AND TECHNOLOGY
Accredited by NAAC with 'A' Grade and NBA Accredited (EEE,CSE & ECE)
Approved by AICTE, New Delhi & Permanently Affiliated to JNTUA, Ananthapuramu
(Recognized by UGC under 2(f) & 12(B) and ISO 9001:2008 Certified Institution)
Nandikotkur Road, Venkayapalli, Kurnool-518452
Department of Computer Science and Engineering
CERTIFICATE
Certified that this is the bonafied record of practical work done by Mr/Ms _____
Roll number _____ of ___ B.Tech ___ semester,
in the _____ during the year _____
Date
Faculty Incharge
Head of the Department
Submitted for the practical examination held on:
Internal Examiner
External Examiner

15A05710 GRID AND CLOUD COMPUTING LABORATORY

Course Objectives:

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

Course Outcomes:

- The student should be able to
- Design and Implement applications on the Cloud.
- Design and implement applications on the Grid.
- Use the grid and cloud tool kits.

GRID COMPUTING PROGRAMS USING GRIDSIM

Installations Steps :-

Requirements:

1. JDK
2. GridSim Toolkit

Steps:

1. Install JDK toolkit
2. Set path for JDK toolkit
   Path=C:/jdk1.8/bin
   Classpath=C:/jdk1.8/jre/lib/rt.jar;
3. Download GridSim 5.2
4. Extract GridSim into one folder.
5. Set path = C:/gridsim/bin;
6. Set Classpath = C:/gridsim/jar/*;
7. Set Classpath = C:/gridsim/examples;
8. Set variable GridSim=C:/gridsim

G.PULLAIAH COLLEGE OF ENGINEERING AND TECHNOLOGY

Accredited by NAAC with 'A' Grade of UGC, Approved by AICTE,New Delhi
Permanently Affiliated to JNTUA, Ananthapuramu
(Recognized by UGC under 2(f) &12(B) & ISO 9001:2008 Certified Institution)
Nandikotkur Road,Kurnool-518452

INDEX

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Mr/MS…….Roll Number……

GRID COMPUTING PROGRAMS USING GRIDSIM

Grid computing programs using Use Globus Toolkit or equivalent

S NO	Date	Title of Experiment	Page No	Marks	Signature of the staff
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 PROGRAMS ON SaaS

S No	Date	Title Of Experiment	Page No	Marks	Signature Of The Staff
1		Create an word document of your class time table and store locally and on cloud with doc and pdf format			
2		Create a spread sheet which contains employee salary information and calculate gross and total salary using formula DA=10%OF BASIC,HRA=30%OF BASIC PF= 10% OF BASIC IF BASIC<=3000 12% OF BASIC IF BASIC>3000 TAX=10% OF BASIC IF BASIC<=1500 11% OF BASIC IF BASIC>1500 AND BASIC<=2500 12% OF BASIC IF BASIC>2500 NET_SALARY=BASIC_SALARY+DA+HRA-PF-TAX			
3		Prepare a ppt on cloud computing-			

		introduction, models, services and architecture			
4		Create your resume in a neat format using Google and Zoho cloud			

PROGRAMS ON PaaS

S NO	Date	Title of Experiment	Page No	Marks	Signature of the staff
1		Write a Google app engine program to generate n even numbers and deploy it to Google cloud			
2		Google app engine program multiply two matrices			
3		Google app engine program to validate user; create a database login(username, password)in mysql and deploy to cloud			
4		Write a Google app engine program to display nth largest no from the given list of numbers and deploy it in Google cloud			
5		Google app engine program to validate the user use mysql to store user info and deploy on to cloud			
6		Implement prog1-5 using Microsoft Azure			

GRID COMPUTING PROGRAMS USING GRIDSIM

1. Program to creates one Grid resource with three machines

```
import java.util.Calendar;
import java.util.LinkedList;


import gridsim.*;


class Example1
{
   public static void main(String[] args)
   {
      System.out.println("Starting example of how to create one Grid " +  "resource");


      try
      {
          int num_user = 0;
         Calendar calendar = Calendar.getInstance();
         boolean trace_flag = true;


         String[] exclude_from_file = { "" };
         String[] exclude_from_processing = { "" };


         String report_name = null;
```

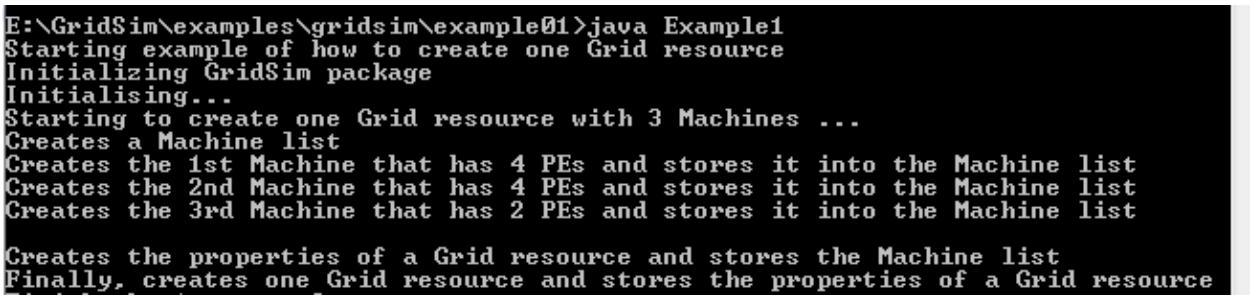
```
        System.out.println("Initializing GridSim package");
        GridSim.init(num_user, calendar, trace_flag, exclude_from_file,
                exclude_from_processing, report_name);

        GridSim.init(num_user, calendar, trace_flag);

        GridResource gridResource = createGridResource();
        System.out.println("Finish the 1st example");

    }
    catch (Exception e)
    {
        e.printStackTrace();
        System.out.println("Unwanted error happens");
    }
}

 private static GridResource createGridResource()
{
    System.out.println("Starting to create one Grid resource with " + "3 Machines ...");

    MachineList mList = new MachineList();
    System.out.println("Creates a Machine list");

    int mipsRating = 377;
```

```
mList.add( new Machine(0, 4, mipsRating));   // First Machine
System.out.println("Creates the 1st Machine that has 4 PEs and " + "stores it into the
                        Machine list");
mList.add( new Machine(1, 4, mipsRating));   // Second Machine
System.out.println("Creates the 2nd Machine that has 4 PEs and " +
      "stores it into the Machine list");
mList.add( new Machine(2, 2, mipsRating));   // Third Machine
System.out.println("Creates the 3rd Machine that has 2 PEs and " +
      "stores it into the Machine list");

String arch = "Sun Ultra";      // system architecture
String os = "Solaris";          // operating system
double time_zone = 9.0;         // time zone this resource located
double cost = 3.0;              // the cost of using this resource

ResourceCharacteristics resConfig = new ResourceCharacteristics(
      arch, os, mList, ResourceCharacteristics.TIME_SHARED,
      time_zone, cost);

System.out.println();
System.out.println("Creates the properties of a Grid resource and " +
      "stores the Machine list");

String name = "Resource_0";         // resource name
double baud_rate = 100.0;           // communication speed
```

```
long seed = 11L*13*17*19*23+1;
double peakLoad = 0.0;       // the resource load during peak hour
double offPeakLoad = 0.0;    // the resource load during off-peak hr
double holidayLoad = 0.0;    // the resource load during holiday

LinkedList<Integer> Weekends = new LinkedList<Integer>();
Weekends.add(new Integer(Calendar.SATURDAY));
Weekends.add(new Integer(Calendar.SUNDAY));

// incorporates holidays. However, no holidays are set in this example
LinkedList<Integer> Holidays = new LinkedList<Integer>();

GridResource gridRes = null;
try
{
   gridRes = new GridResource(name, baud_rate, seed,
      resConfig, peakLoad, offPeakLoad, holidayLoad, Weekends,
      Holidays);
}
catch (Exception e) {
   e.printStackTrace();
}

System.out.println("Finally, creates one Grid resource and stores " +
      "the properties of a Grid resource");
```

```
            return gridRes;
        }
    }
```

Output:



2. Program to create one or more Grid users. A Grid user contains one or more Gridlets.

```
import java.util.*;

import gridsim.*;


class Example2

{

   public static void main(String[] args)

   {

      System.out.println("Starting example of how to create Grid users");

      System.out.println();


      try
```

```
    {
        GridletList list = createGridlet();
        System.out.println("Creating " + list.size() + " Gridlets");
        ResourceUserList userList = createGridUser(list);
        System.out.println("Creating " + userList.size() + " Grid users");
        printGridletList(list);
        System.out.println("Finish the example");
    }
    catch (Exception e)
    {
        e.printStackTrace();
        System.out.println("Unwanted error happens");
    }
}
private static GridletList createGridlet()
{
    GridletList list = new GridletList();
    int id = 0;
    double length = 3500.0;
    long file_size = 300;
    long output_size = 300;
    Gridlet gridlet1 = new Gridlet(id, length, file_size, output_size);
    id++;
    Gridlet gridlet2 = new Gridlet(id, 5000, 500, 500);
    id++;
```

```
    Gridlet gridlet3 = new Gridlet(id, 9000, 900, 900);
    list.add(gridlet1);
    list.add(gridlet2);
    list.add(gridlet3);
    Random random = new Random();
    GridSimStandardPE.setRating(100);
    int count = 5;
    double min_range = 0.10;
    double max_range = 0.50;
    for (int i = 1; i < count+1; i++)
    {
       length = GridSimStandardPE.toMIs(random.nextDouble()*output_size);
       file_size = (long) GridSimRandom.real(100, min_range, max_range,
                         random.nextDouble());
       output_size = (long) GridSimRandom.real(250, min_range, max_range,
                         random.nextDouble());
       Gridlet gridlet = new Gridlet(id + i, length, file_size,
                         output_size);
       list.add(gridlet);
    }

    return list;
}
private static ResourceUserList createGridUser(GridletList list)
{
```

```
    ResourceUserList userList = new ResourceUserList();

    userList.add(0);    // user ID starts from 0
    userList.add(1);
    userList.add(2);

    int userSize = userList.size();
    int gridletSize = list.size();
    int id = 0;
    for (int i = 0; i < gridletSize; i++)
    {
        if (i != 0 && i % userSize == 0)
            id++;

        ( (Gridlet) list.get(i) ).setUserID(id);
    }

    return userList;
}

private static void printGridletList(GridletList list)
{
    int size = list.size();
    Gridlet gridlet;
```

```
        String indent = "    ";
        System.out.println();
        System.out.println("Gridlet ID" + indent + "User ID" + indent +
              "length" + indent + " file size" + indent +
              "output size");

        for (int i = 0; i < size; i++)
        {
           gridlet = (Gridlet) list.get(i);
           System.out.println(indent + gridlet.getGridletID() + indent +
                 indent + indent + gridlet.getUserID() + indent + indent +
                 (int) gridlet.getGridletLength() + indent + indent +
                 (int) gridlet.getGridletFileSize() + indent + indent +
                 (int) gridlet.getGridletOutputSize() );
        }
     }
} // end class
```

3. Program to shows how two GridSim entities interact with each other ; main(ie example3) class creates Gridlets and sends them to the other GridSim entities, i.e. Test class

```
import java.util.*;

import gridsim.*;

class Example3 extends GridSim

{
```

```
private String entityName_;
private GridletList list_;
private GridletList receiveList_;
Example3(String name, double baud_rate, GridletList list) throws Exception
{
   super(name);
   this.list_ = list;
   receiveList_ = new GridletList();
   entityName_ = "Test";
   new Test(entityName_, baud_rate);
}
public void body()
{
   int size = list_.size();
   Gridlet obj, gridlet;
   for (int i = 0; i < size; i++)
   {
      obj = (Gridlet) list_.get(i);
      System.out.println("Inside Example3.body() => Sending Gridlet " +
            obj.getGridletID());
      super.send(entityName_, GridSimTags.SCHEDULE_NOW,
            GridSimTags.GRIDLET_SUBMIT, obj);
      gridlet = super.gridletReceive();
      System.out.println("Inside Example3.body() => Receiving Gridlet "+
            gridlet.getGridletID());
```

```
        receiveList_.add(gridlet);
    }
    super.send(entityName_, GridSimTags.SCHEDULE_NOW,
            GridSimTags.END_OF_SIMULATION);
}
public GridletList getGridletList() {
    return receiveList_;
}
public static void main(String[] args)
{
    System.out.println("Starting Example3");
    System.out.println();

    try
    {
        int num_user = 0;   // number of users need to be created
        Calendar calendar = Calendar.getInstance();
        boolean trace_flag = true;  // mean trace GridSim events
        String[] exclude_from_file = { "" };
        String[] exclude_from_processing = { "" };
        String report_name = null;
        System.out.println("Initializing GridSim package");
        GridSim.init(num_user, calendar, trace_flag, exclude_from_file,
              exclude_from_processing, report_name);
        GridletList list = createGridlet();
```

```
        System.out.println("Creating " + list.size() + " Gridlets");
        Example3 obj = new Example3("Example3", 560.00, list);
        GridSim.startGridSimulation();
        GridletList newList = obj.getGridletList();
        printGridletList(newList);
        System.out.println("Finish Example3");
    }
    catch (Exception e)
    {
        e.printStackTrace();
        System.out.println("Unwanted errors happen");
    }
}
private static GridletList createGridlet()
{
    GridletList list = new GridletList();
    int id = 0;
    double length = 3500.0;
    long file_size = 300;
    long output_size = 300;
    Gridlet gridlet1 = new Gridlet(id, length, file_size, output_size);
    id++;
    Gridlet gridlet2 = new Gridlet(id, 5000, 500, 500);
    id++;
    Gridlet gridlet3 = new Gridlet(id, 9000, 900, 900);
```

```
    list.add(gridlet1);
    list.add(gridlet2);
    list.add(gridlet3);
    long seed = 11L*13*17*19*23+1;
    Random random = new Random(seed);
    GridSimStandardPE.setRating(100);
    int count = 5;
    for (int i = 1; i < count+1; i++)
    {
       length = GridSimStandardPE.toMIs(random.nextDouble()*50);
       file_size = (long) GridSimRandom.real(100, 0.10, 0.40,
                         random.nextDouble());
       output_size = (long) GridSimRandom.real(250, 0.10, 0.50,
                         random.nextDouble());
       Gridlet gridlet = new Gridlet(id + i, length, file_size,
                         output_size);
       list.add(gridlet);
    }

    return list;
}
private static void printGridletList(GridletList list)
{
    int size = list.size();
    Gridlet gridlet;
```

```
        String indent = "    ";
        System.out.println();
        System.out.println("========== OUTPUT ==========");
        System.out.println("Gridlet ID" + indent + "STATUS");

        for (int i = 0; i < size; i++)
        {
           gridlet = (Gridlet) list.get(i);
           System.out.print(indent + gridlet.getGridletID() + indent
                 + indent);

           if (gridlet.getGridletStatus() == Gridlet.SUCCESS)
              System.out.println("SUCCESS");
        }
     }
}
```

4. Program shows how a grid user submits its Gridlets or tasks to one grid resource entity

```
import java.util.*;
import gridsim.*;
class Example6 extends GridSim
{
   private Integer ID_;
   private String name_;
   private GridletList list_;
   private GridletList receiveList_;
   private int totalResource_;

   Example6(String name, double baud_rate, int total_resource)
         throws Exception
   {
      super(name, baud_rate);
      this.name_ = name;
      this.totalResource_ = total_resource;
      this.receiveList_ = new GridletList();
      this.ID_ = new Integer( getEntityId(name) );
      System.out.println("Creating a grid user entity with name = " +
            name + ", and id = " + this.ID_);
      this.list_ = createGridlet( this.ID_.intValue() );
      System.out.println(name + ":Creating "+ this.list_.size() +
            " Gridlets");
   }
   public void body()
```

```
{
   int resourceID[] = new int[this.totalResource_];
   double resourceCost[] = new double[this.totalResource_];
   String resourceName[] = new String[this.totalResource_];

   LinkedList resList;
   ResourceCharacteristics resChar;
   while (true)
   {
      super.gridSimHold(1.0);    // hold by 1 second
      resList = super.getGridResourceList();
      if (resList.size() == this.totalResource_)
         break;
      else
      {
         System.out.println(this.name_ +
               ":Waiting to get list of resources ...");
      }
   }
   int i = 0;
   for (i = 0; i < this.totalResource_; i++)
   {
      resourceID[i] = ( (Integer)resList.get(i) ).intValue();
      super.send(resourceID[i], GridSimTags.SCHEDULE_NOW,
            GridSimTags.RESOURCE_CHARACTERISTICS, this.ID_);
```

```
    resChar = (ResourceCharacteristics) super.receiveEventObject();
    resourceName[i] = resChar.getResourceName();
    resourceCost[i] = resChar.getCostPerSec();
    System.out.println(this.name_ + ":Received ResourceCharacteristics from " +
        resourceName[i] + ", with id = " + resourceID[i]);
    super.recordStatistics("\"Received ResourceCharacteristics " +
        "from " + resourceName[i] + "\"", "");
}
Gridlet gridlet;
String info;
int id = 0;
for (i = 0; i < this.list_.size(); i++)
{
    gridlet = (Gridlet) this.list_.get(i);
    info = "Gridlet_" + gridlet.getGridletID();
    id = GridSimRandom.intSample(this.totalResource_);
    System.out.println(this.name_ + ":Sending " + info + " to " +
        resourceName[id] + " with id = " + resourceID[id]);
    super.gridletSubmit(gridlet, resourceID[id]);
    super.recordStatistics("\"Submit " + info + " to " +
        resourceName[id] + "\"", "");
    gridlet = super.gridletReceive();
    System.out.println(this.name_ + ":Receiving Gridlet " +
        gridlet.getGridletID() );
    super.recordStatistics("\"Received " + info +  " from " +
```

```
                resourceName[id] + "\"", gridlet.getProcessingCost());
        this.receiveList_.add(gridlet);
    }
    super.shutdownGridStatisticsEntity();
    super.shutdownUserEntity();
    super.terminateIOEntities();
    System.out.println(this.name_ + ":%%%% Exiting body()");
}
public GridletList getGridletList() {
    return this.receiveList_;
}
private GridletList createGridlet(int userID)
{
    GridletList list = new GridletList();
    int id = 0;
    double length = 3500.0;
    long file_size = 300;
    long output_size = 300;
    Gridlet gridlet1 = new Gridlet(id, length, file_size, output_size);
    id++;
    Gridlet gridlet2 = new Gridlet(id, 5000, 500, 500);
    id++;
    Gridlet gridlet3 = new Gridlet(id, 9000, 900, 900);

    // setting the owner of these Gridlets
```

```
gridlet1.setUserID(userID);
gridlet2.setUserID(userID);
gridlet3.setUserID(userID);

// Store the Gridlets into a list
list.add(gridlet1);
list.add(gridlet2);
list.add(gridlet3);
GridSimStandardPE.setRating(100);

// creates 5 Gridlets
int max = 5;
int count = GridSimRandom.intSample(max);
for (int i = 1; i < count+1; i++)
{
   length = GridSimStandardPE.toMIs(GridSimRandom.doubleSample()*50);
   file_size = (long) GridSimRandom.real(100, 0.10, 0.40,
                                                GridSimRandom.doubleSample());
   output_size = (long) GridSimRandom.real(250, 0.10, 0.50,
               GridSimRandom.doubleSample());
   Gridlet gridlet = new Gridlet(id + i, length, file_size,
                     output_size);
   gridlet.setUserID(userID);
   list.add(gridlet);
}
```

```
    return list;
}
public static void main(String[] args)
{
    System.out.println("Starting Example6");

    try
    {
        int num_user = 3;   // number of grid users
        Calendar calendar = Calendar.getInstance();
        boolean trace_flag = false;  // mean don't trace GridSim events
        String[] exclude_from_file = { "" };
        String[] exclude_from_processing = { "" };
        String report_name = null;
        GridSim.init(num_user, calendar, trace_flag, exclude_from_file,
              exclude_from_processing, report_name);
        GridResource resource0 = createGridResource("Resource_0");
        GridResource resource1 = createGridResource("Resource_1");
        GridResource resource2 = createGridResource("Resource_2");
        int total_resource = 3;
        Example6 user0 = new Example6("User_0", 560.00, total_resource);
        Example6 user1 = new Example6("User_1", 250.00, total_resource);
        Example6 user2 = new Example6("User_2", 150.00, total_resource);
        GridSim.startGridSimulation();
```

```
        GridletList newList = null;
        newList = user0.getGridletList();
        printGridletList(newList, "User_0");

        newList = user1.getGridletList();
        printGridletList(newList, "User_1");

        newList = user2.getGridletList();
        printGridletList(newList, "User_2");

        System.out.println("Finish Example6");
    }
    catch (Exception e)
    {
        e.printStackTrace();
        System.out.println("Unwanted errors happen");
    }
}
private static GridResource createGridResource(String name)
{
    MachineList mList = new MachineList();
    int mipsRating = 377;
    mList.add( new Machine(0, 4, mipsRating));   // First Machine
    mList.add( new Machine(1, 4, mipsRating));   // Second Machine
    mList.add( new Machine(2, 2, mipsRating));   // Third Machine
```

```
String arch = "Sun Ultra";      // system architecture
String os = "Solaris";          // operating system
double time_zone = 9.0;         // time zone this resource located
double cost = 3.0;              // the cost of using this resource

ResourceCharacteristics resConfig = new ResourceCharacteristics(
        arch, os, mList, ResourceCharacteristics.TIME_SHARED,
        time_zone, cost);

// 5. Finally, we need to create a GridResource object.
double baud_rate = 100.0;           // communication speed
long seed = 11L*13*17*19*23+1;
double peakLoad = 0.0;       // the resource load during peak hour
double offPeakLoad = 0.0;    // the resource load during off-peak hr
double holidayLoad = 0.0;    // the resource load during holiday
LinkedList Weekends = new LinkedList();
Weekends.add(new Integer(Calendar.SATURDAY));
Weekends.add(new Integer(Calendar.SUNDAY));
LinkedList Holidays = new LinkedList();
GridResource gridRes = null;
try {
   gridRes = new GridResource(name, baud_rate, seed,
        resConfig, peakLoad, offPeakLoad, holidayLoad, Weekends,
        Holidays);
}
```

```
    catch (Exception e) {
        e.printStackTrace();
    }

    System.out.println("Creates one Grid resource with name = " + name);
    return gridRes;
}
private static void printGridletList(GridletList list, String name)
{
    int size = list.size();
    Gridlet gridlet;

    String indent = "    ";
    System.out.println();
    System.out.println("========== OUTPUT for " + name + " ==========");
    System.out.println("Gridlet ID" + indent + "STATUS" + indent +
          "Resource ID" + indent + "Cost");

    for (int i = 0; i < size; i++)
    {
        gridlet = (Gridlet) list.get(i);
        System.out.print(indent + gridlet.getGridletID() + indent
              + indent);

        if (gridlet.getGridletStatus() == Gridlet.SUCCESS)
```

```
            System.out.print("SUCCESS");


          System.out.println( indent + indent + gridlet.getResourceID() +
                indent + indent + gridlet.getProcessingCost() );
       }
    }

} // end class
```

5. Program to show how a grid user submits its Gridlets or task to many grid resource entities

```
import java.util.*;
import gridsim.*;
public class Test
{
   private static final int MIN = 1;  // min number of test cases
   private static final int MAX = 8;  // max number of test cases

   /**
    * Usage in Unix / Linux:
    *    javac -classpath $GRIDSIM/gridsim.jar:. Test.java
    *    java Test [policy: space | time] [test case number: 1 - 8]
    *
    * For example: java Test space 7 --> running Space-Shared for test case #7
    *           java Test time  3 --> running Time-Shared for test case #3
    *
    * The operation of these Test Cases offer are:
    * Test Case 1: Submit Gridlets - then wait until all Finish to collect
    * Test Case 2: Submit Gridlets - Cancel some of them - Finish
    * Test Case 3: Submit Gridlets - Pause some of them - Cancel - Finish
    * Test Case 4: Submit Gridlets - Pause - Resume - Cancel - Finish
    * Test Case 5: Submit Gridlets - Move some of them - Finish
    * Test Case 6: Submit Gridlets - Pause - Move - Finish
    * Test Case 7: Submit Gridlets - Pause - Resume - Move - Finish
    * Test Case 8: Submit Gridlets - Pause - Resume - Move - Cancel - Finish
```

```
     *
     * NOTE:
     * - Test Case 1 is the simplest and Test Case 8 is the most complicated.
     *
     * - These Test Cases are quite flexible, meaning, you can adjust how big
     *   these experiments are by increasing/decreasing totalUser, totalPE, etc
     *   from main() only. You don't need to modify
     *   any of the Test Case classes.
     *
     * - Be careful when setting the numbers too high (above 200)
     *   since you might get Java "Out of Memory" exception.
     *
     * - For an effective experiment for Gridlet or Job migration, you need to
     *   have a large number of GridResource entities, say more than 6.
     */
    public static void main(String[] args)
    {
        System.out.println("Starting Test Cases");
        try
        {
            // Parse the command line args
            int policy = 0;
            if ( args[0].equals("t") || args[0].equals("time") ) {
                policy = ResourceCharacteristics.TIME_SHARED;
            }
            else if ( args[0].equals("s") || args[0].equals("space") ) {
                policy = ResourceCharacteristics.SPACE_SHARED;
            }
            else {
                System.out.println("Error -- Invalid allocation policy....");
                return;
            }

            // determine which test case number to choose
            int testNum = Integer.parseInt(args[1]);
            if (testNum < MIN || testNum > MAX) {
                testNum = MIN;
            }

            ////////////////////////////////////////
            // First step: Initialize the GridSim package. It should be called
```

```
// before creating any entities. We can't run this example without
// initializing GridSim first. We will get run-time exception
// error.
Calendar calendar = Calendar.getInstance();
boolean trace_flag = false;  // true means tracing GridSim events

// list of files or processing names to be excluded from any
// statistical measures
String[] exclude_from_file = { "" };
String[] exclude_from_processing = { "" };

// the name of a report file to be written. We don't want to write
// anything here.
String report_name = null;

// initialize all revelant variables
double baudRate[] = {1000, 5000}; // bandwidth for even, odd
int peRating[] = {10, 50};   // PE Rating for even, odd
double price[] = {3.0, 5.0};   // resource for even, odd
int gridletLength[] = {1000, 2000, 3000, 4000, 5000};

// Initialize the GridSim package
int totalUser = 2;    // total Users for this experiment
GridSim.init(totalUser, calendar, trace_flag, exclude_from_file,
      exclude_from_processing, report_name);

////////////////////////////////////
// Second step: Creates one or more GridResource objects
int totalResource = 3;  // total GridResources for this experiment
int totalMachine = 1;   // total Machines for each GridResource
int totalPE = 3;       // total PEs for each Machine
createResource(totalResource, totalMachine, totalPE, baudRate,
           peRating, price, policy);

////////////////////////////////////
// Third step: Creates grid users
int totalGridlet = 4;     // total Gridlets for each User
createUser(totalUser, totalGridlet, gridletLength, baudRate,
        testNum);

////////////////////////////////////
```

```
            // Fourth step: Starts the simulation
            GridSim.startGridSimulation();
        }
        catch (Exception e)
        {
            System.out.println("Unwanted errors happen");
            System.out.println( e.getMessage() );
            System.out.println("Usage: java Test [time | space] [1-8]");
        }
        System.out.println("================ END OF TEST ====================");
    }

    /**
     * Creates many GridResources
     */
    public static void createResource(int totalRes, int totalMachine,
                        int totalPE, double[] baudRate, int[] peRating,
                        double[] price, int policy)
    {
        double bandwidth = 0;
        double cost = 0.0;

        // a loop that creates one or more GridResources
        for (int i = 0; i < totalRes; i++)
        {
            String name = "GridResource_" + i;
            if (i % 2 == 0)
            {
                bandwidth = baudRate[0];
                cost = price[0];
            }
            else
            {
                bandwidth = baudRate[1];
                cost = price[1];
            }

            // creates a GridResource
            createGridResource(name, totalMachine, totalPE, bandwidth,
                            peRating, policy, cost);
        }
```

```
}

/**
 * Creates many Grid Users
 */
public static void createUser(int totalUser, int totalGridlet,
                int[] glLength, double[] baudRate, int testNum)
{
   try
   {
      double bandwidth = 0;
      double delay = 0.0;

      for (int i = 0; i < totalUser; i++)
      {
         String name = "User_" + i;
         if (i % 2 == 0) {
            bandwidth = baudRate[0];
            delay = 5.0;
         }
         else {
            bandwidth = baudRate[1];
         }

         // creates a Grid user
         createTestCase(name, bandwidth, delay, totalGridlet, glLength,
                  testNum);
      }
   }
   catch (Exception e) {
      // ... ignore
   }
}

/**
 * A selection of different test cases
 */
private static void createTestCase(String name, double bandwidth,
                double delay, int totalGridlet, int[] glLength,
                int testNum) throws Exception
{
```

```
    switch(testNum)
    {
      case 1:
        new TestCase1(name, bandwidth, delay, totalGridlet, glLength);
        break;

      case 2:
        new TestCase2(name, bandwidth, delay, totalGridlet, glLength);
        break;

      case 3:
        new TestCase3(name, bandwidth, delay, totalGridlet, glLength);
        break;

      case 4:
        new TestCase4(name, bandwidth, delay, totalGridlet, glLength);
        break;

      case 5:
        new TestCase5(name, bandwidth, delay, totalGridlet, glLength);
        break;

      case 6:
        new TestCase6(name, bandwidth, delay, totalGridlet, glLength);
        break;

      case 7:
        new TestCase7(name, bandwidth, delay, totalGridlet, glLength);
        break;

      case 8:
        new TestCase8(name, bandwidth, delay, totalGridlet, glLength);
        break;

      default:
        System.out.println("Not a recognized test case.");
        break;
    }
  }

  /**
```

```
 * Creates one Grid resource. A Grid resource contains one or more
 * Machines. Similarly, a Machine contains one or more PEs (Processing
 * Elements or CPUs).
 */
private static void createGridResource(String name, int totalMachine,
                    int totalPE, double bandwidth, int[] peRating,
                    int policy, double cost)
{
   // Here are the steps needed to create a Grid resource:
   // 1. We need to create an object of MachineList to store one or more
   //    Machines
   MachineList mList = new MachineList();

   int rating = 0;
   for (int i = 0; i < totalMachine; i++)
   {
      // even Machines have different PE rating compare to odd ones
      if (i % 2 == 0) {
         rating = peRating[0];
      }
      else {
         rating = peRating[1];
      }

      // 2. Create one Machine with its id, number of PEs and rating
      mList.add( new Machine(i, totalPE, rating) );
   }

   // 3. Create a ResourceCharacteristics object that stores the
   //    properties of a Grid resource: architecture, OS, list of
   //    Machines, allocation policy: time- or space-shared, time zone
   //    and its price (G$/PE time unit).
   String arch = "Sun Ultra";      // system architecture
   String os = "Solaris";          // operating system
   double time_zone = 0.0;         // time zone this resource located

   ResourceCharacteristics resConfig = new ResourceCharacteristics(
         arch, os, mList, policy, time_zone, cost);

   // 4. Finally, we need to create a GridResource object.
   long seed = 11L*13*17*19*23+1;
```

```
        double peakLoad = 0.0;      // the resource load during peak hour
        double offPeakLoad = 0.0;   // the resource load during off-peak hr
        double holidayLoad = 0.0;   // the resource load during holiday

        // incorporates weekends so the grid resource is on 7 days a week
        LinkedList Weekends = new LinkedList();
        Weekends.add(new Integer(Calendar.SATURDAY));
        Weekends.add(new Integer(Calendar.SUNDAY));

        // incorporates holidays. However, no holidays are set in this example
        LinkedList Holidays = new LinkedList();
        try
        {
           GridResource gridRes = new GridResource(name, bandwidth, seed,
              resConfig, peakLoad, offPeakLoad, holidayLoad, Weekends,
              Holidays);
        }
        catch (Exception e)
        {
           System.out.println("Error in creating GridResource.");
           System.out.println( e.getMessage() );
        }

        System.out.println("Creates one Grid resource with name = " + name);
        return;
      }

    } // end class
```

6. Program to show how to create one or more grid users and submits its Gridlets or task to many grid resource entities

```
import java.util.*;

import gridsim.*;




/**

 * This is the example main program that demonstrates how to
```

```
 * submit / cancel / resume / pause / move Gridlets to different GridResources.
 * You can play around with this class by adjusting few parameters in main()
 * such as totalUser, totalGridlet, etc.
 */
public class Test
{
   private static final int MIN = 1;  // min number of test cases
   private static final int MAX = 8;  // max number of test cases

   /**
    * Usage in Unix / Linux:
    *    javac -classpath $GRIDSIM/gridsim.jar:. Test.java
    *    java Test [policy: space | time] [test case number: 1 - 8]
    *
    * For example: java Test space 7 --> running Space-Shared for test case #7
    *           java Test time  3 --> running Time-Shared for test case #3
    *
    * The operation of these Test Cases offer are:
    * Test Case 1: Submit Gridlets - then wait until all Finish to collect
    * Test Case 2: Submit Gridlets - Cancel some of them - Finish
    * Test Case 3: Submit Gridlets - Pause some of them - Cancel - Finish
    * Test Case 4: Submit Gridlets - Pause - Resume - Cancel - Finish
    * Test Case 5: Submit Gridlets - Move some of them - Finish
    * Test Case 6: Submit Gridlets - Pause - Move - Finish
    * Test Case 7: Submit Gridlets - Pause - Resume - Move - Finish
```

```
 * Test Case 8: Submit Gridlets - Pause - Resume - Move - Cancel - Finish
 *
 * NOTE:
 * - Test Case 1 is the simplest and Test Case 8 is the most complicated.
 *
 * - These Test Cases are quite flexible, meaning, you can adjust how big
 *   these experiments are by increasing/decreasing totalUser, totalPE, etc
 *   from main() only. You don't need to modify
 *   any of the Test Case classes.
 *
 * - Be careful when setting the numbers too high (above 200)
 *   since you might get Java "Out of Memory" exception.
 *
 * - For an effective experiment for Gridlet or Job migration, you need to
 *   have a large number of GridResource entities, say more than 6.
 */
public static void main(String[] args)
{
   System.out.println("Starting Test Cases");
   try
   {
      // Parse the command line args
      int policy = 0;
      if ( args[0].equals("t") || args[0].equals("time") ) {
         policy = ResourceCharacteristics.TIME_SHARED;
```

```
}
else if ( args[0].equals("s") || args[0].equals("space") ) {
   policy = ResourceCharacteristics.SPACE_SHARED;
}
else {
   System.out.println("Error -- Invalid allocation policy....");
   return;
}

// determine which test case number to choose
int testNum = Integer.parseInt(args[1]);
if (testNum < MIN || testNum > MAX) {
   testNum = MIN;
}

//////////////////////////////////////
// First step: Initialize the GridSim package. It should be called
// before creating any entities. We can't run this example without
// initializing GridSim first. We will get run-time exception
// error.
Calendar calendar = Calendar.getInstance();
boolean trace_flag = false;  // true means tracing GridSim events

// list of files or processing names to be excluded from any
// statistical measures
```

```
String[] exclude_from_file = { "" };
String[] exclude_from_processing = { "" };

// the name of a report file to be written. We don't want to write
// anything here.
String report_name = null;

// initialize all revelant variables
double baudRate[] = {1000, 5000}; // bandwidth for even, odd
int peRating[] = {10, 50};   // PE Rating for even, odd
double price[] = {3.0, 5.0};   // resource for even, odd
int gridletLength[] = {1000, 2000, 3000, 4000, 5000};

// Initialize the GridSim package
int totalUser = 2;    // total Users for this experiment
GridSim.init(totalUser, calendar, trace_flag, exclude_from_file,
      exclude_from_processing, report_name);

////////////////////////////////////
// Second step: Creates one or more GridResource objects
int totalResource = 3;  // total GridResources for this experiment
int totalMachine = 1;   // total Machines for each GridResource
int totalPE = 3;       // total PEs for each Machine
createResource(totalResource, totalMachine, totalPE, baudRate,
            peRating, price, policy);
```

```
        ////////////////////////////////////
        // Third step: Creates grid users
        int totalGridlet = 4;     // total Gridlets for each User
        createUser(totalUser, totalGridlet, gridletLength, baudRate,
                testNum);

        ////////////////////////////////////
        // Fourth step: Starts the simulation
        GridSim.startGridSimulation();
    }
    catch (Exception e)
    {
        System.out.println("Unwanted errors happen");
        System.out.println( e.getMessage() );
        System.out.println("Usage: java Test [time | space] [1-8]");
    }
    System.out.println("=============== END OF TEST ==================");
}

/**
 * Creates many GridResources
 */
public static void createResource(int totalRes, int totalMachine,
                    int totalPE, double[] baudRate, int[] peRating,
```

```
                    double[] price, int policy)
{
    double bandwidth = 0;
    double cost = 0.0;

    // a loop that creates one or more GridResources
    for (int i = 0; i < totalRes; i++)
    {
        String name = "GridResource_" + i;
        if (i % 2 == 0)
        {
            bandwidth = baudRate[0];
            cost = price[0];
        }
        else
        {
            bandwidth = baudRate[1];
            cost = price[1];
        }

        // creates a GridResource
        createGridResource(name, totalMachine, totalPE, bandwidth,
                peRating, policy, cost);
    }
}
```

```
/**
 * Creates many Grid Users
 */
public static void createUser(int totalUser, int totalGridlet,
                    int[] glLength, double[] baudRate, int testNum)
{
   try
   {
      double bandwidth = 0;
      double delay = 0.0;

      for (int i = 0; i < totalUser; i++)
      {
         String name = "User_" + i;
         if (i % 2 == 0) {
            bandwidth = baudRate[0];
            delay = 5.0;
         }
         else {
            bandwidth = baudRate[1];
         }

         // creates a Grid user
         createTestCase(name, bandwidth, delay, totalGridlet, glLength,
```

```
                    testNum);
        }
    }
    catch (Exception e) {
        // ... ignore
    }
}

/**
 * A selection of different test cases
 */
private static void createTestCase(String name, double bandwidth,
                    double delay, int totalGridlet, int[] glLength,
                    int testNum) throws Exception
{
    switch(testNum)
    {
        case 1:
            new TestCase1(name, bandwidth, delay, totalGridlet, glLength);
            break;

        case 2:
            new TestCase2(name, bandwidth, delay, totalGridlet, glLength);
            break;
```

```
case 3:
   new TestCase3(name, bandwidth, delay, totalGridlet, glLength);
   break;

case 4:
   new TestCase4(name, bandwidth, delay, totalGridlet, glLength);
   break;

case 5:
   new TestCase5(name, bandwidth, delay, totalGridlet, glLength);
   break;

case 6:
   new TestCase6(name, bandwidth, delay, totalGridlet, glLength);
   break;

case 7:
   new TestCase7(name, bandwidth, delay, totalGridlet, glLength);
   break;

case 8:
   new TestCase8(name, bandwidth, delay, totalGridlet, glLength);
   break;

default:
```

```
            System.out.println("Not a recognized test case.");
            break;
        }
    }

    /**
     * Creates one Grid resource. A Grid resource contains one or more
     * Machines. Similarly, a Machine contains one or more PEs (Processing
     * Elements or CPUs).
     */
    private static void createGridResource(String name, int totalMachine,
                    int totalPE, double bandwidth, int[] peRating,
                    int policy, double cost)
    {
        // Here are the steps needed to create a Grid resource:
        // 1. We need to create an object of MachineList to store one or more
        //    Machines
        MachineList mList = new MachineList();

        int rating = 0;
        for (int i = 0; i < totalMachine; i++)
        {
            // even Machines have different PE rating compare to odd ones
            if (i % 2 == 0) {
                rating = peRating[0];
```

```
    }
    else {
        rating = peRating[1];
    }

    // 2. Create one Machine with its id, number of PEs and rating
    mList.add( new Machine(i, totalPE, rating) );
}

// 3. Create a ResourceCharacteristics object that stores the
//    properties of a Grid resource: architecture, OS, list of
//    Machines, allocation policy: time- or space-shared, time zone
//    and its price (G$/PE time unit).
String arch = "Sun Ultra";      // system architecture
String os = "Solaris";          // operating system
double time_zone = 0.0;         // time zone this resource located

ResourceCharacteristics resConfig = new ResourceCharacteristics(
       arch, os, mList, policy, time_zone, cost);

// 4. Finally, we need to create a GridResource object.
long seed = 11L*13*17*19*23+1;
double peakLoad = 0.0;       // the resource load during peak hour
double offPeakLoad = 0.0;    // the resource load during off-peak hr
double holidayLoad = 0.0;    // the resource load during holiday
```

```
        // incorporates weekends so the grid resource is on 7 days a week
        LinkedList Weekends = new LinkedList();
        Weekends.add(new Integer(Calendar.SATURDAY));
        Weekends.add(new Integer(Calendar.SUNDAY));

        // incorporates holidays. However, no holidays are set in this example
        LinkedList Holidays = new LinkedList();
        try
        {
           GridResource gridRes = new GridResource(name, bandwidth, seed,
              resConfig, peakLoad, offPeakLoad, holidayLoad, Weekends,
              Holidays);
        }
        catch (Exception e)
        {
           System.out.println("Error in creating GridResource.");
           System.out.println( e.getMessage() );
        }

        System.out.println("Creates one Grid resource with name = " + name);
        return;
     }

} // end class
```

7. Program to creates one Grid resource with three machines

Grid computing programs using Use Globus Toolkit or equivalent:

GRID COMPUTING PROGRAMS USING USE GLOBUS TOOLKIT OR EQUIVALENT

1. Develop a new Web Service for Calculator.

OBJECTIVE:

To develop a new Web service for Calculator applications.

PROCEDURE:

When you start Globus toolkit container, there will be number of services starts up. The service for this task will be a simple Math service that can perform basic arithmetic for a client.

The Math service will access a resource with two properties:

1. An integer value that can be operated upon by the service

2. A string values that holds string describing the last operation

The service itself will have three remotely accessible operations that operate upon

value:

(a) add, that adds *a* to the resource property *value*.

(b) subtract that subtracts *a* from the resource property *value*.

(c) getValueRP that returns the current value of *value*.

Usually, the best way for any programming task is to begin with an overall description of what you want the code to do, which in this case is the service interface. The service interface describes how what the service provides in terms of names of operations, their arguments and return values. A Java interface for our service is:

public interface Math {

public void add(int a);

public void subtract(int a);

public int getValueRP();

}

It is possible to start with this interface and create the necessary WSDL file using the standard Web service tool called Java2WSDL. However, the WSDL file for GT 4 has to include details of resource properties that are not given explicitly in the interface above. Hence, we will provide the WSDL file.

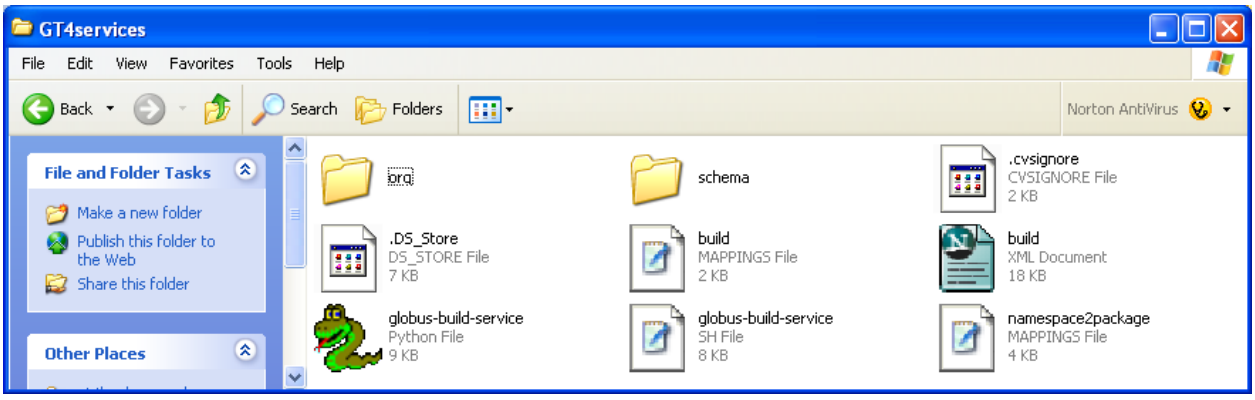
Step 1 Getting the Files

All the required files are provided and comes directly from [1]. The MathService source code files can be found from http://www.gt4book.com

(http://www.gt4book.com/downloads/gt4book-examples.tar.gz)

A Windows zip compressed version can be found at

http://www.cs.uncc.edu/~abw/ITCS4146S07/gt4book-examples.zip. Download and uncompress the file into a directory called **GT4services**. Everything is included (the java source WSDL and deployment files, etc.):



WSDL service interface description file -- ***The WSDL service interface description***

file is provided within the GT4services folder at:

GT4Services\schema\examples\MathService_instance\Math.wsdl

This file, and discussion of its contents, can be found in Appendix A. Later on we will need to modify this file, but first we will use the existing contents that describe the Math service above.

Service code in Java -- For this assignment, both the code for service operations and for the resource properties are put in the same class for convenience. More complex services and resources would be defined in separate classes. The Java code for the service and its resource properties is located within the GT4services folder at:

GT4services\org\globus\examples\services\core\first\impl\MathService.java.

Deployment Descriptor -- The deployment descriptor gives several different important sets of information about the service once it is deployed. It is located within the **GT4services** folder at:

GT4services\org\globus\examples\services\core\first\deploy-server.wsdd.

Step 2 – Building the Math Service

It is now necessary to package all the required files into a GAR (Grid Archive) file. The build tool ant from the Apache Software Foundation is used to achieve this as shown overleaf:

Generating a GAR file with Ant (from http://gdp.globus.org/gt4-

tutorial/multiplehtml/ch03s04.html)

Ant is similar in concept to the Unix make tool but a java tool and XML based.

Build scripts are provided by Globus 4 to use the ant build file. The windows version of the build script for MathService is the Python file called **globus-build-service.py**, which held in the **GT4services** directory. The build script takes one argument, the name of your service that you want to deploy. To keep with the naming convention in [1], this service will be called **first**.

In the *Client Window*, run the build script from the **GT4services** directory with:

globus-build-service.py first

The output should look similar to the following:

Buildfile: build.xml

.

.

.

.

.

BUILD SUCCESSFUL

Total time: 8 seconds

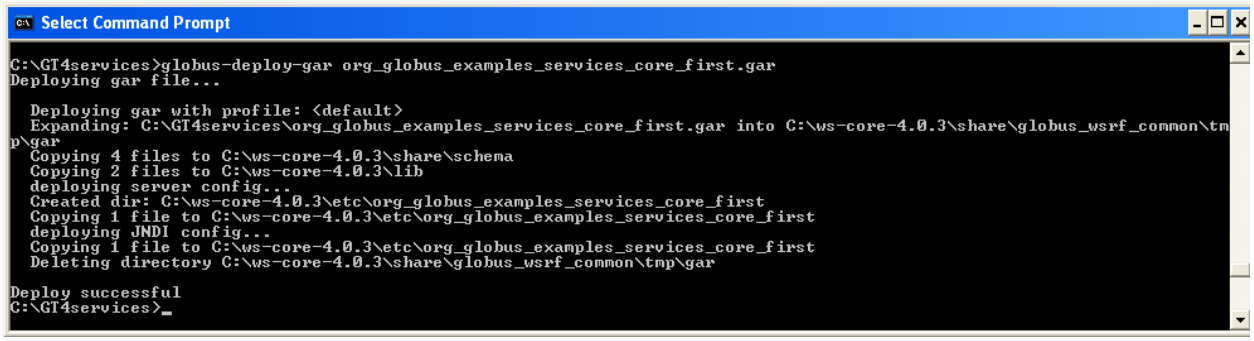
During the build process, a new directory is created in your **GT4Services** directory that is named **build**. All of your stubs and class files that were generated will be in that directory and its subdirectories. More importantly, there is a GAR (Grid Archive) file called **org_globus_examples_services_core_first.gar**. The GAR file is the package that contains every file that is needed to successfully deploy your Math Service into the Globus container. The files contained in the GAR file are the Java class files, WSDL, compiled stubs, and the deployment descriptor.

Step 3 – Deploying the Math Service

If the container is still running in the Container Window, then stop it using Control-C. To deploy the Math Service, you will use a tool provided by the Globus Toolkit called **globus-deploy-gar**. In the *Container Window*, issue the command:

globus-deploy-gar org_globus_examples_services_core_first.gar

Successful output of the command is :



The service has now been deployed.

Check service is deployed by starting container from the *Container Window*:

You should see the service called **MathService**.

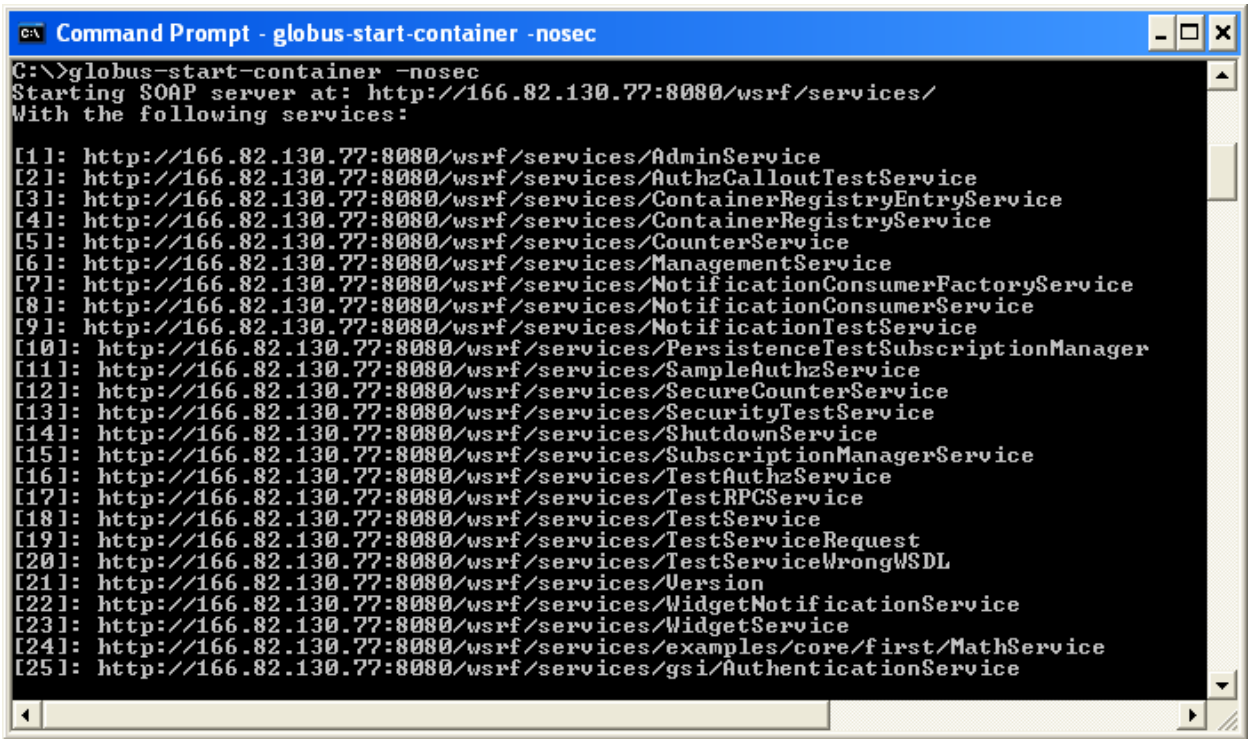
Step 4 – Compiling the Client

A client has already been provided to test the Math Service and is located in the

GT4Services directory at:

GT4Services\org\globus\examples\clients\MathService_instance\Client.java

and contains



You should see the service called **MathService**.

Step 4 – Compiling the Client

A client has already been provided to test the Math Service and is located in the

GT4Services directory at:

GT4Services\org\globus\examples\clients\MathService_instance\Client.java

and contains the following code:

```
package org.globus.examples.clients.MathService_instance;
import org.apache.axis.message.addressing.Address;
import org.apache.axis.message.addressing.EndpointReferenceType;
import org.globus.examples.stubs.MathService_instance.MathPortType;
import org.globus.examples.stubs.MathService_instance.GetValueRP;
import
org.globus.examples.stubs.MathService_instance.service.MathServiceAddressingL
ocator;
public class Client {
public static void main(String[] args) {
MathServiceAddressingLocator locator = new
MathServiceAddressingLocator()
try {
String serviceURI = args[0];
// Create endpoint reference to service
EndpointReferenceType endpoint = new
EndpointReferenceType();
endpoint.setAddress(new Address(serviceURI));
MathPortType math;
// Get PortType
math = locator.getMathPortTypePort(endpoint);
// Perform an addition
math.add(10);
// Perform another addition
math.add(5);
```

```
// Access value
System.out.println("Current value: "
+ math.getValueRP(new GetValueRP()));
// Perform a subtraction
math.subtract(5);
// Access value
System.out.println("Current value: "
+ math.getValueRP(new GetValueRP()));
} catch (Exception e) {
e.printStackTrace();
}
}
}
```

When the client is run from the command line, you pass it one argument. The argument is the URL that specifies where the service resides. The client will create the end point rerference and incorporate this URL as the address. The end point reference is then used with the **getMathPortTypePort** method of a **MathServiceAdressingLocator** object to obtain a reference to the Math interface (portType). Then, we can apply the methods available in the service as though they were local methods Notice that the call to the service (add and subtract method calls) must be in a "**try {} catch(){}**" block because a "RemoteException" may be thrown. The code for the "**MathServiceAddressingLocator**" is created during the build process. (Thus you don't have to write it!)

(a) Settting the Classpath

To compile the new client, you will need the JAR files from the Globus toolkit in your CLASSPATH. Do this by executing the following command in the Client Window:

%GLOBUS_LOCATION%\etc\globus-devel-env.bat

You can verify that this sets your CLASSPATH, by executing the command:

echo %CLASSPATH%

You should see a long list of JAR files.

Running **\gt4\etc\globus-devel-env.bat** only needs to be done *once* for each *Client Window* that you open. It does *not* need to be done each time you compile.

(b) Compiling Client

Once your CLASSPATH has been set, then you can compile the Client code by typing in the following command:

javac -classpath

build\classes\org\globus\examples\services\core\first\impl\:%CLASSPATH%

org\globus\examples\clients\MathService_instance\Client.java

Step 5 – Start the Container for your Service

Restart the Globus container from the *Container Window* with:

globus-start-container -nosec

if the container is not running.

Step 6 – Run the Client

To start the client from your **GT4Services** directory, do the following in the *Client Window*, which passes the GSH of the service as an argument:

java -classpath

build\classes\org\globus\examples\services\core\first\impl\:%CLASSPATH%

org.globus.examples.clients.MathService_instance.Client

http://localhost:8080/wsrf/services/examples/core/first/MathService

which should give the output:

Current value: 15

Current value: 10

Step 7 – Undeploy the Math Service and Kill a Container

Before we can add functionality to the Math Service (Section 5), we must undeploy the service. In the *Container Window*, kill the container with a Control-C. Then to undeploy the service, type in the following command:

globus-undeploy-gar org_globus_examples_services_core_first

which should result with the following output:

Undeploying gar...

Deleting /.

.

.

Undeploy successful

6 Adding Functionality to the Math Service

In this final task, you are asked to modify the Math service and associated files so the srvice supports the multiplication operation. To do this task, you will need to modify:

- Service code (**MathService.java**)
- WSDL file (**Math.wsdl)**

The exact changes that are necessary are not given. You are to work them out yourself. You will need to fully understand the contents of service code and WSDL files and then modify them accordingly. Appendix A gives an explanation of the important parts of these files. Keep all file names the same and simply redeploy the service afterwards. You will also need to add a code to the client code (**Client.java**) to test the modified service to include multiplication.

Result:

2. Develop new OGSA-compliant Web Service

OBJECTIVE:

To develop a new OGSA-compliant web service.

PROCEDURE:

Writing and deploying a WSRF Web Service is easier than you might think. You just have to follow five simple steps

1. Define the service's interface. This is done with *WSDL*

2. Implement the service. This is done with *Java*.

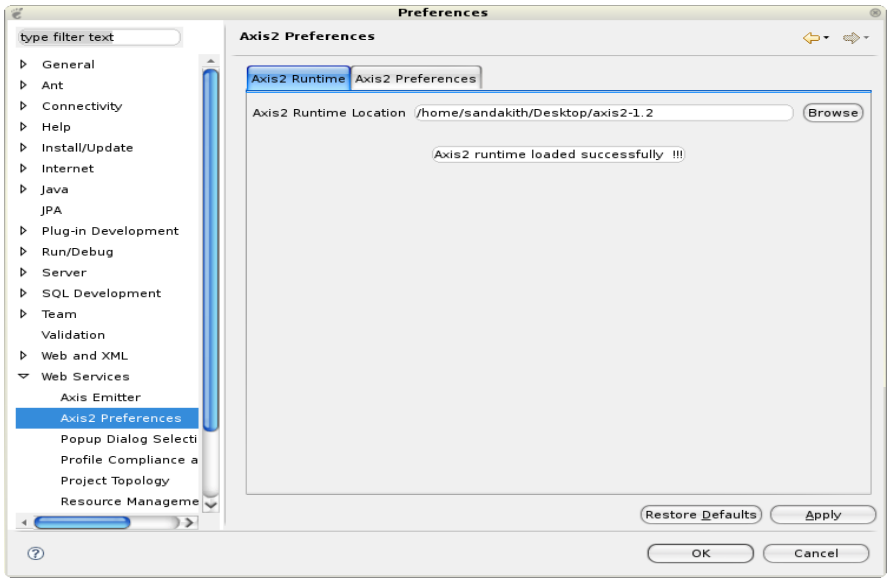
3. Define the deployment parameters. This is done with *WSDD* and *JNDI*

4. Compile everything and generate a GAR file. This is done with *Ant*

5. Deploy service. This is also done with *a GT4 tool*

To run this program, as a minimum you will be required to have installed the following prerequisite software

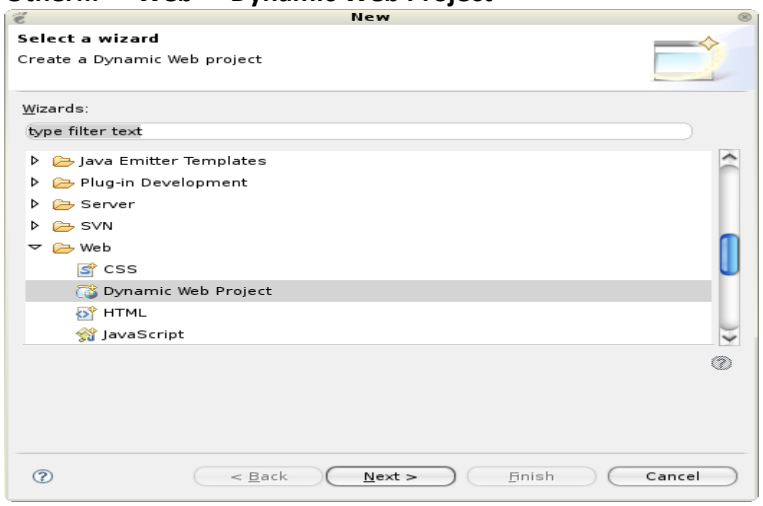
a. Download the latest Axis2 runtime from the above link and extract it. Now we point Eclipse WTP to downloaded Axis2 Runtime. Open **Window -> Preferences -> Web Services -> Axis2 Emitter**



Select the Axis2 Runtime tab and point to the correct Axis2 runtime location. Alternatively at the Axis2 Preference tab, you can set the default setting that will come up on the Web Services Creation wizards. For the moment we will accept the default settings.

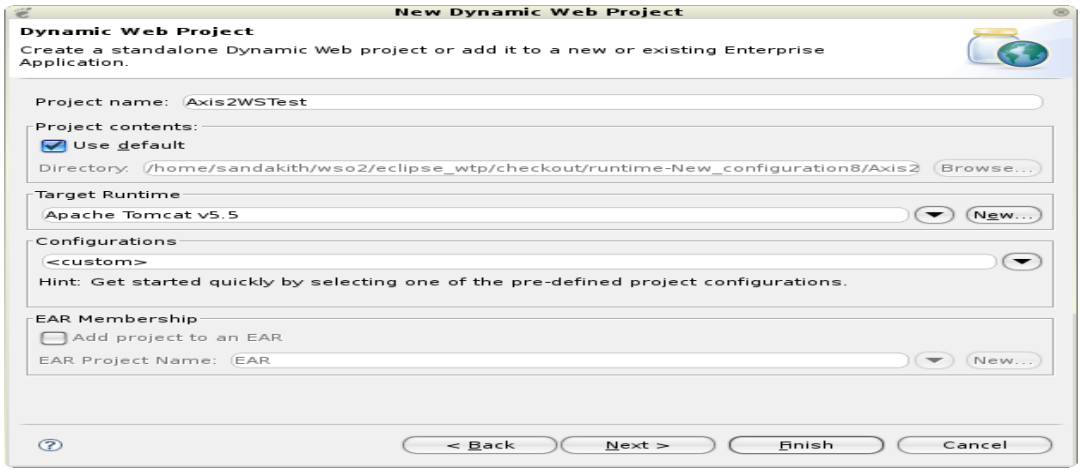
b. Click OK.

c. Next we need to create a project with the support of Axis2 features. Open **File -> New -> Other... -> Web -> Dynamic Web Project**



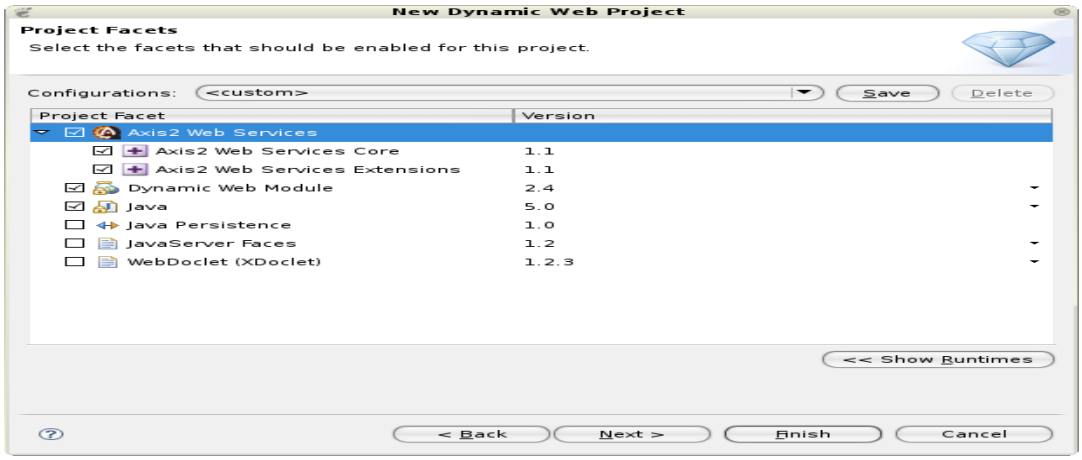
Click next

d. Select the name **Axis2WSTest** as the Dynamic Web project name (you can specify any name you prefer), and select the configured Tomcat runtime as the target runtime.



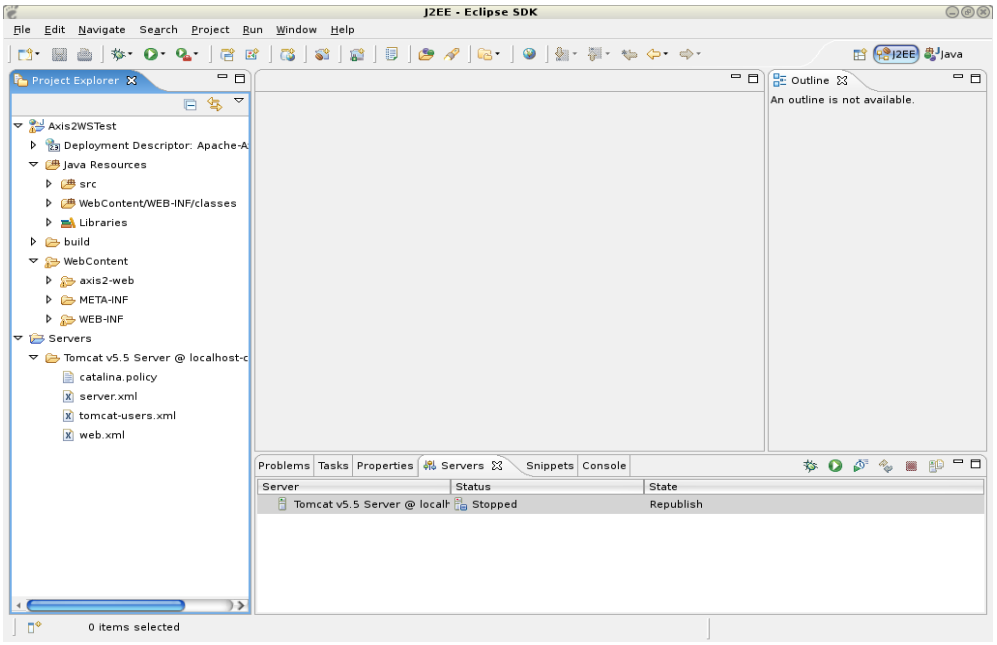
Click next.

e. Select the Axis2 Web service facet

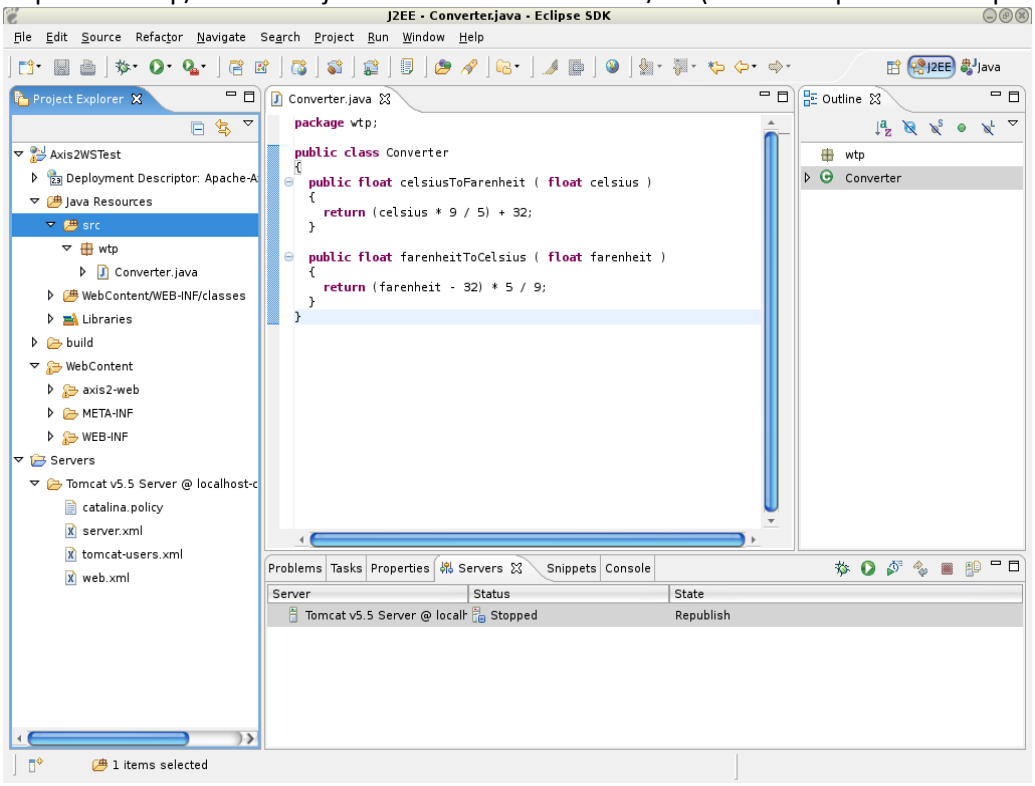


Click Finish.

f. This will create a dynamic Web project in the workbench

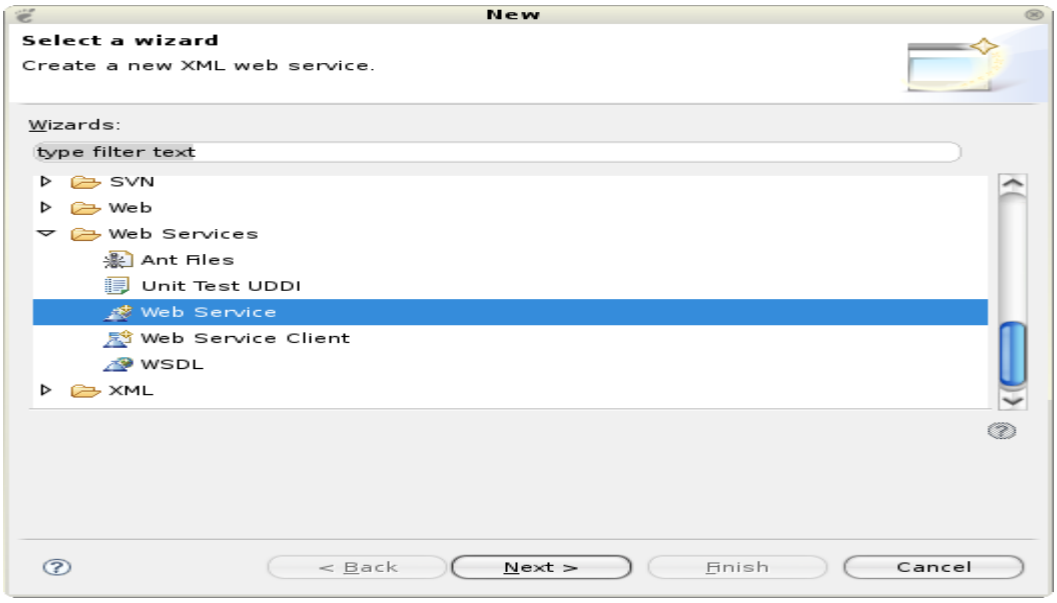


g. Import the wtp/Converter.java class into Axis2WSTest/src (be sure to preserve the package).



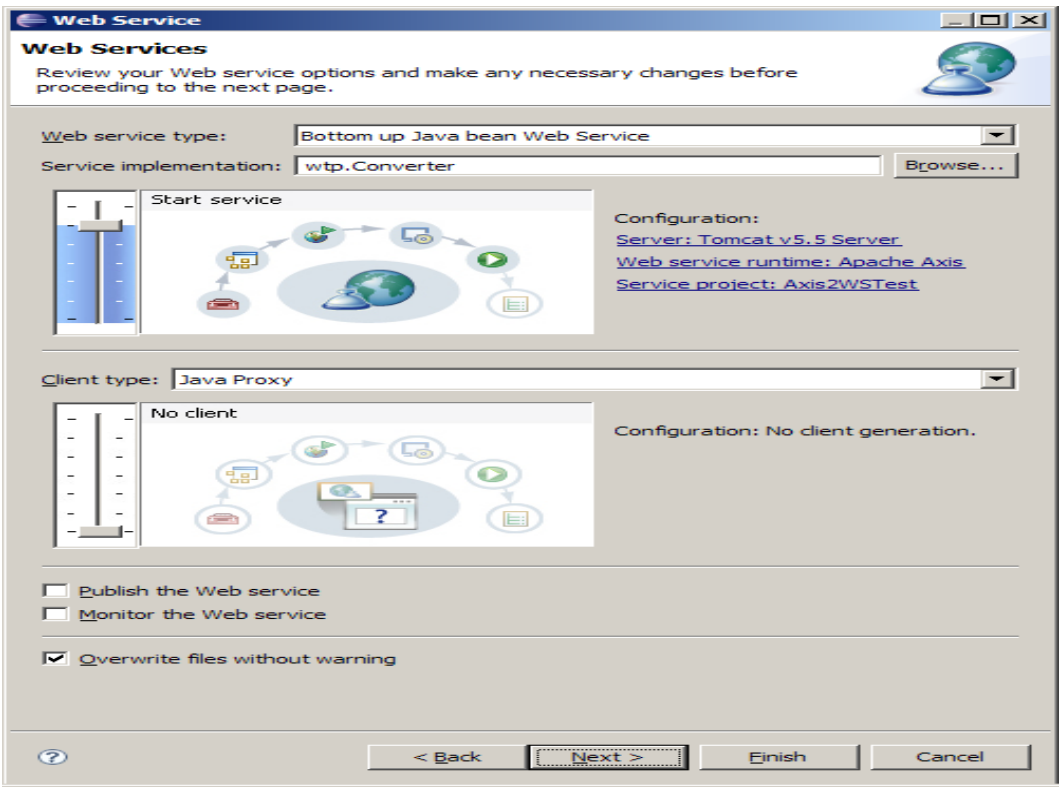
Build the Project, if its not auto build.

h. Select Converter.java, open File -> New -> Other... -> Web Services -> Web Service

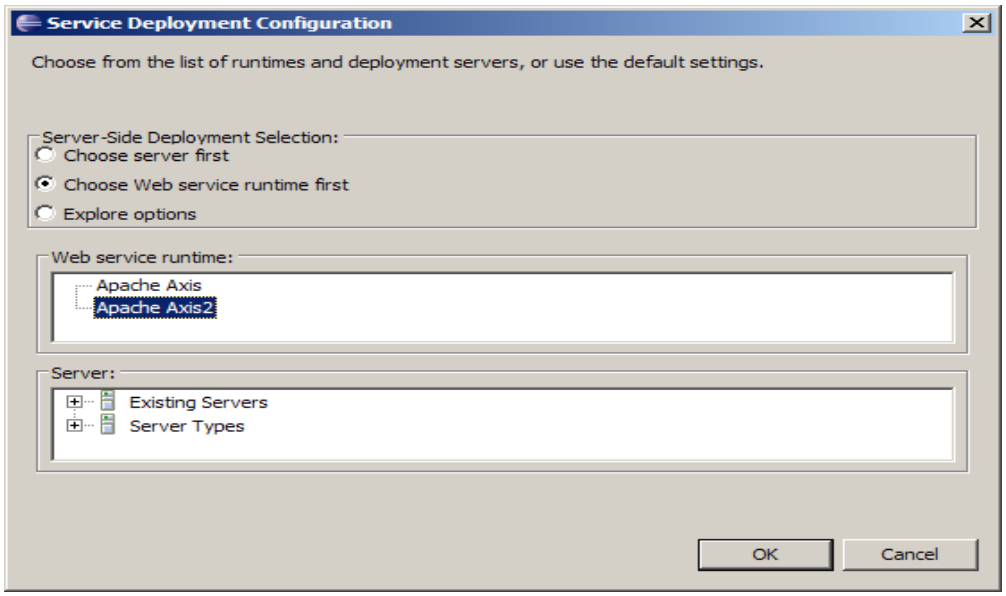


Click next.

i. The Web service wizard would be brought up with Web service type set to **Bottom up Java bean Web Service** with the service implementation automatically filled in. Move the service scale to **Start service**.

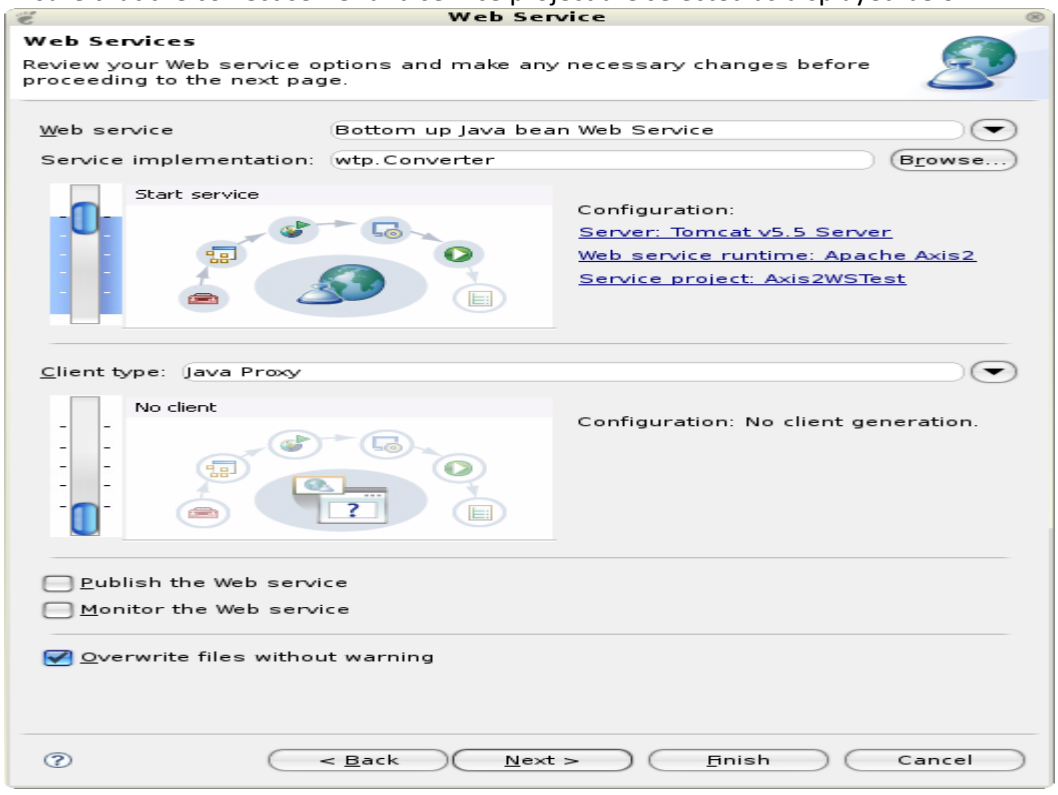


j. Click on the **Web Service runtime** link to select the Axis2 runtime.



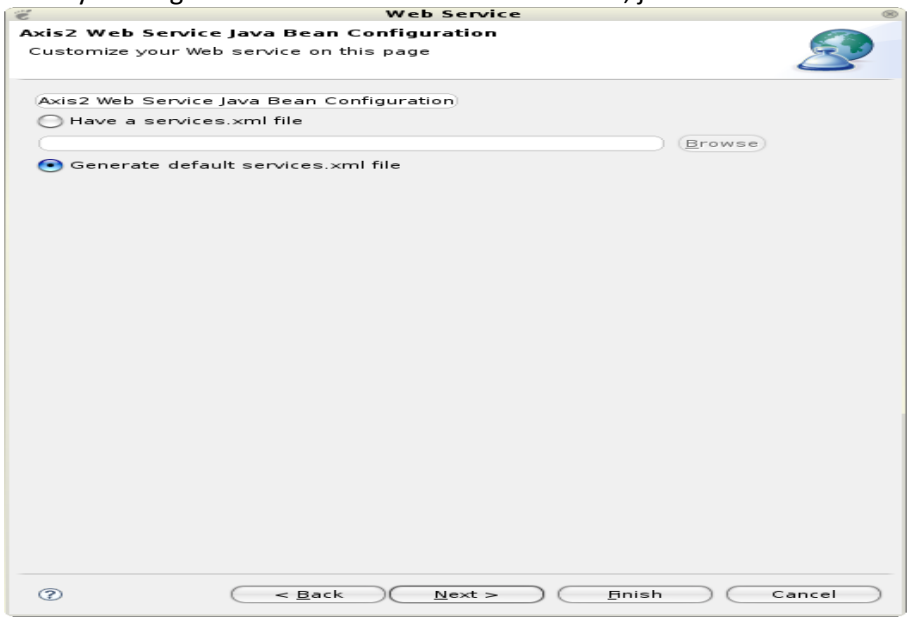
Click OK.

k. Ensure that the correct server and service project are selected as displayed below.



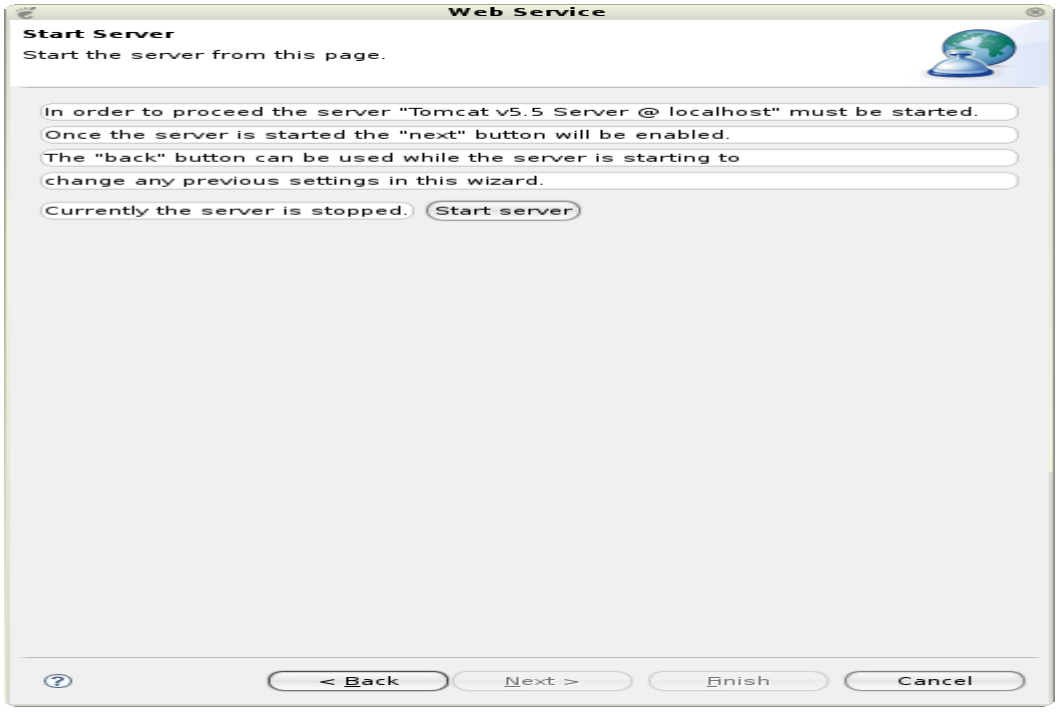
Click next.

l. This page is the service.xml selection page. if you have a custom services.xml, you can include that by clicking the **Browse** button. For the moment, just leave it at the default.



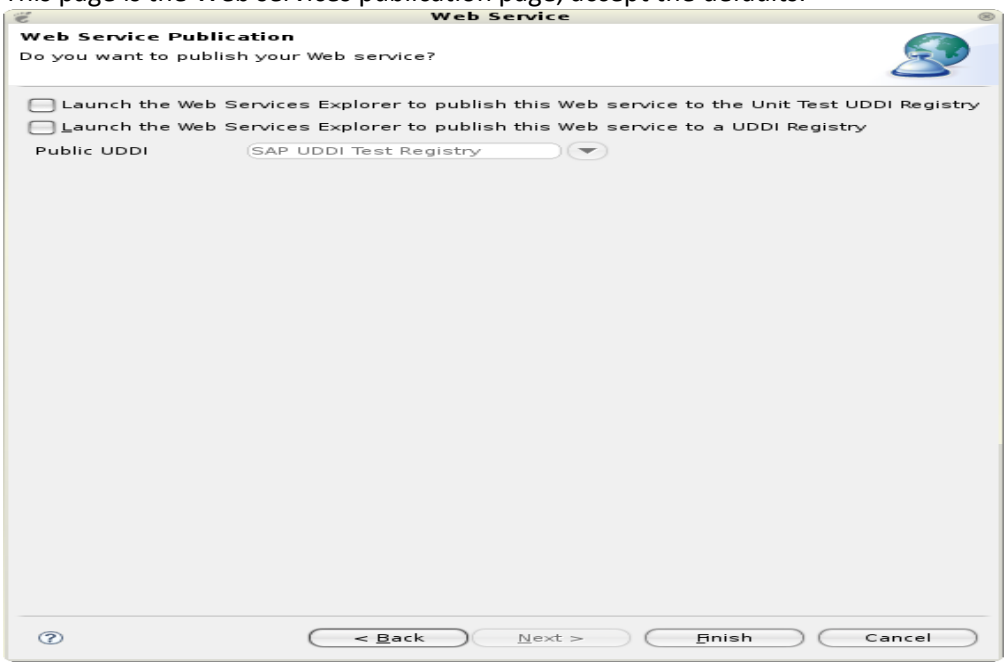
Click next.

m. This page is the Start Server page. It will be displayed if the server has not been started. Click on the **Start Server** button. This will start the server runtime.



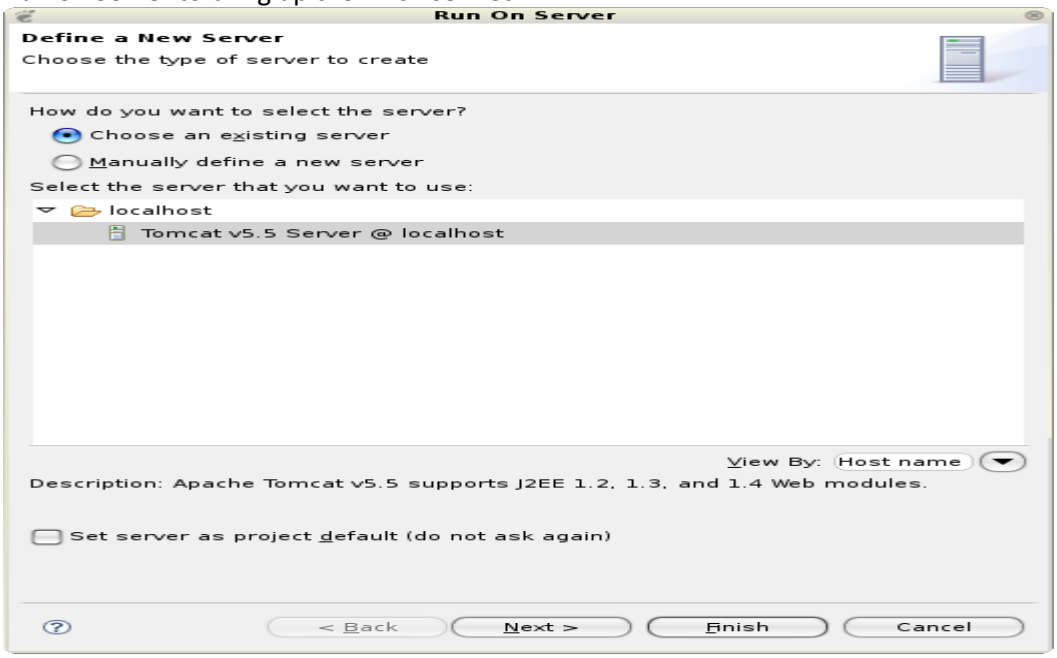
Click next.

n. This page is the Web services publication page, accept the defaults.



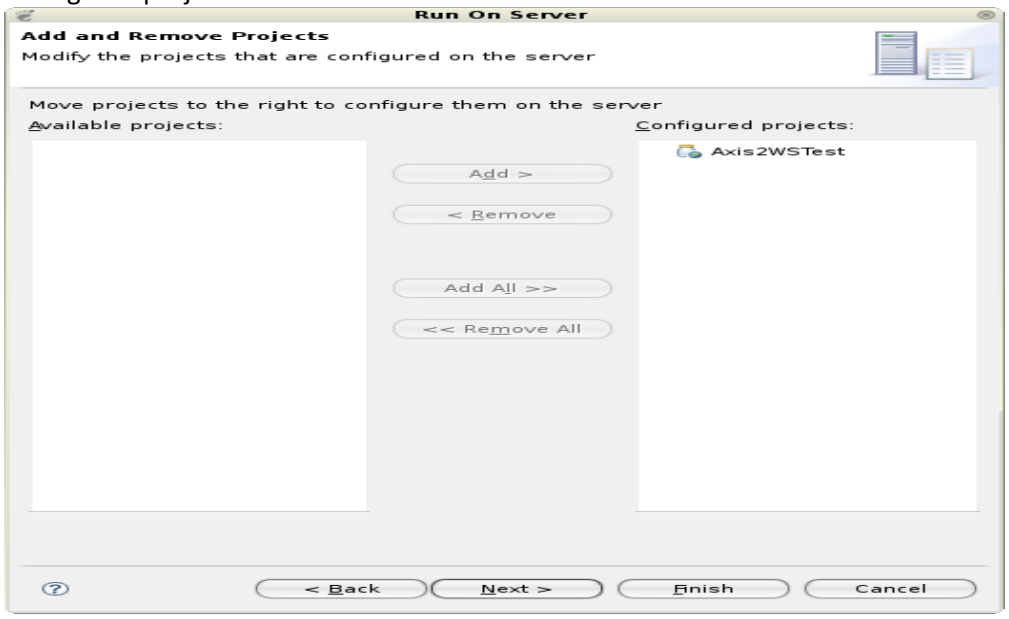
Click Finish.

o. Now, select the **Axis2WSTest** dynamic Web project, right-click and select Run -> Run As -> Run on Server to bring up the Axis2 servlet.



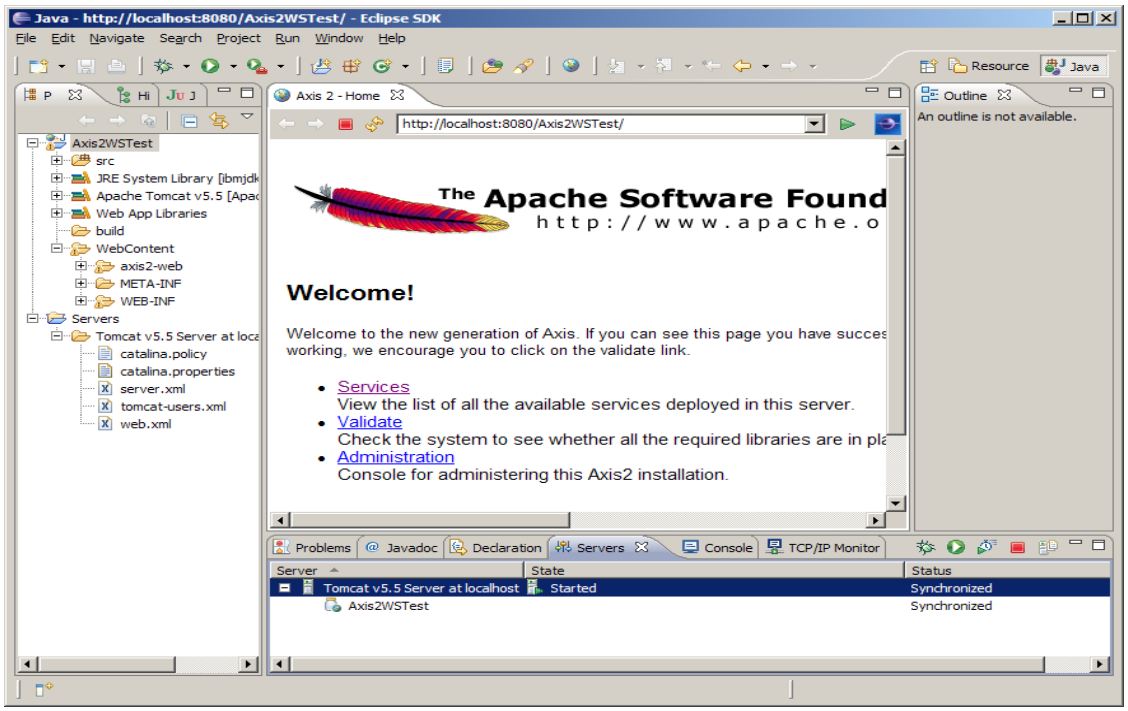
Click Next.

p. Make sure you have the **Axis2WSTest** dynamic Web project on the right-hand side under the Configured project.

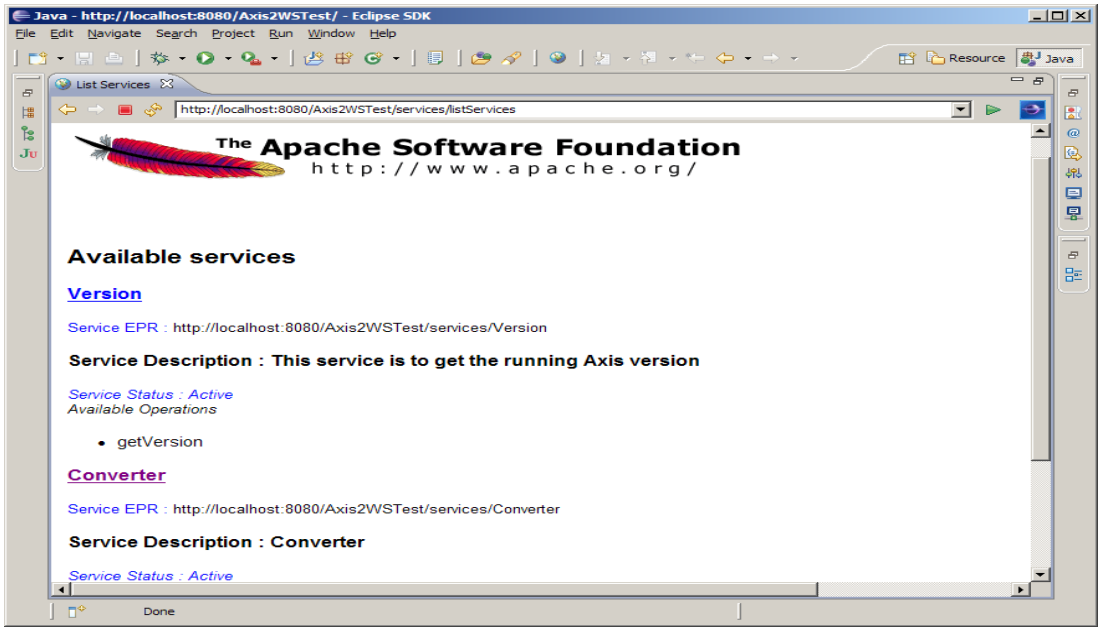


Click Finish.

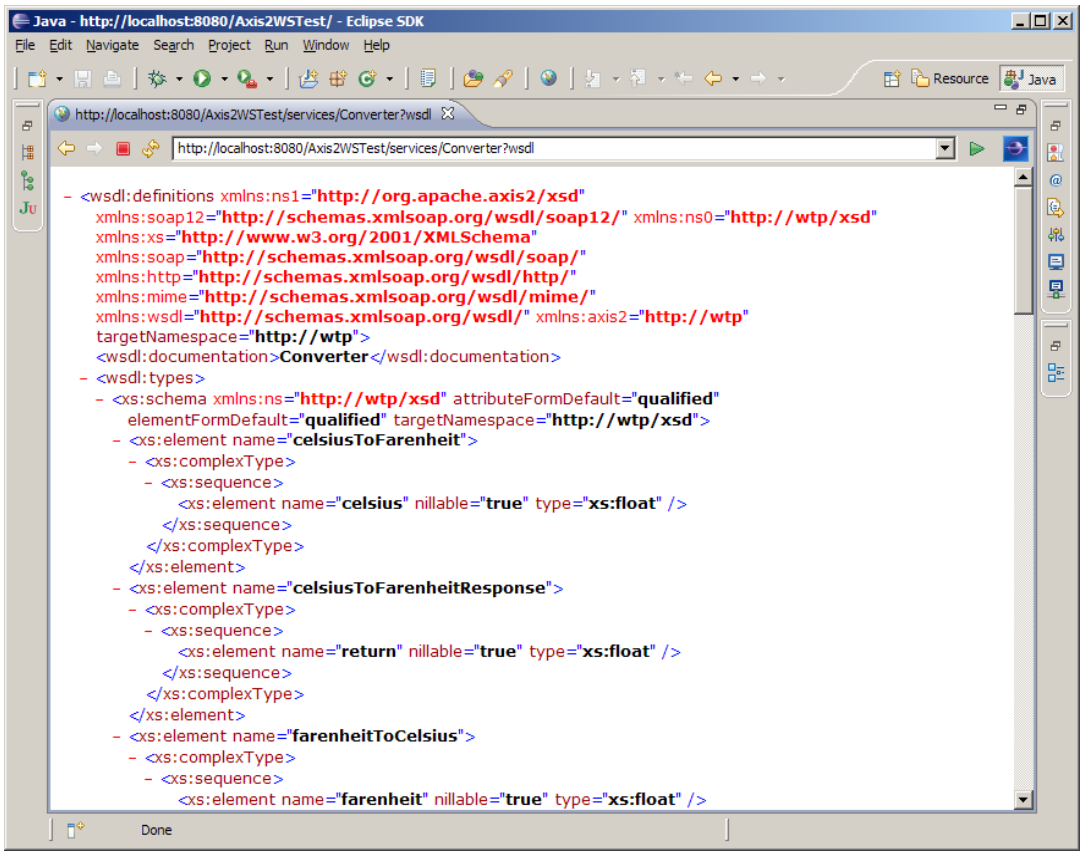
q. This will deploy the Axis2 server webapp on the configured servlet container and will display the Axis2 home page. Note that the servlet container will start up according to the Server configuration files on your workspace.



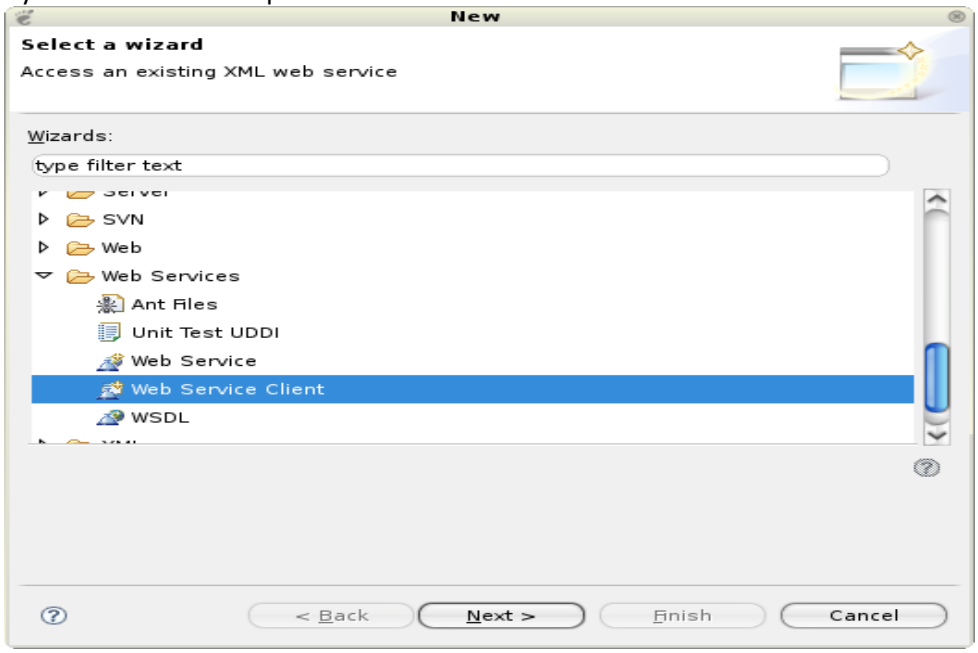
r. Click on the **Services** link to view the available services. The newly created converter Web service will be shown there.



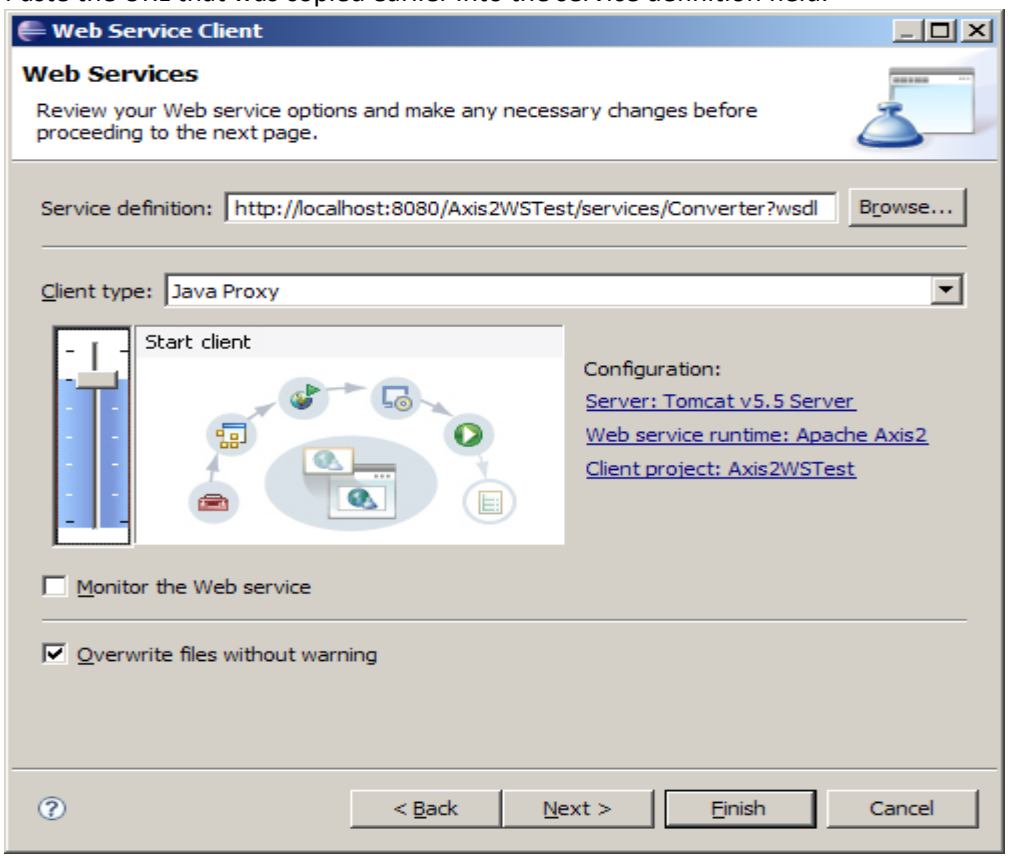
s. Click on the **Converter Service** link to display the wsdl URL of the newly created Web service. Copy the URL.



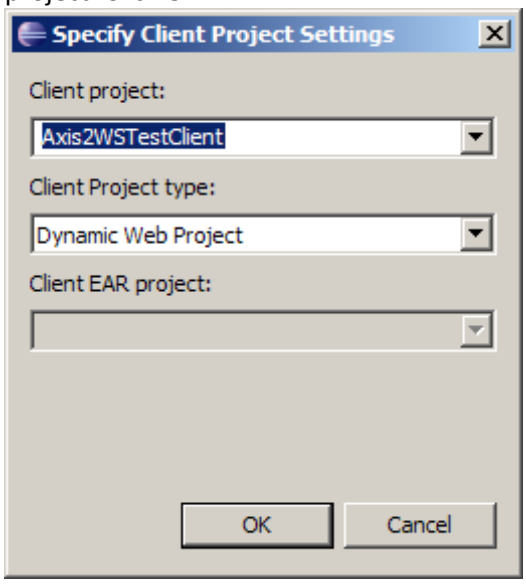
t. Now we'll generate the client for the newly created service by referring the ?wsdl generated by the Axis2 Server. Open File -> New -> Other... -> Web Services -> Web ServiceClient



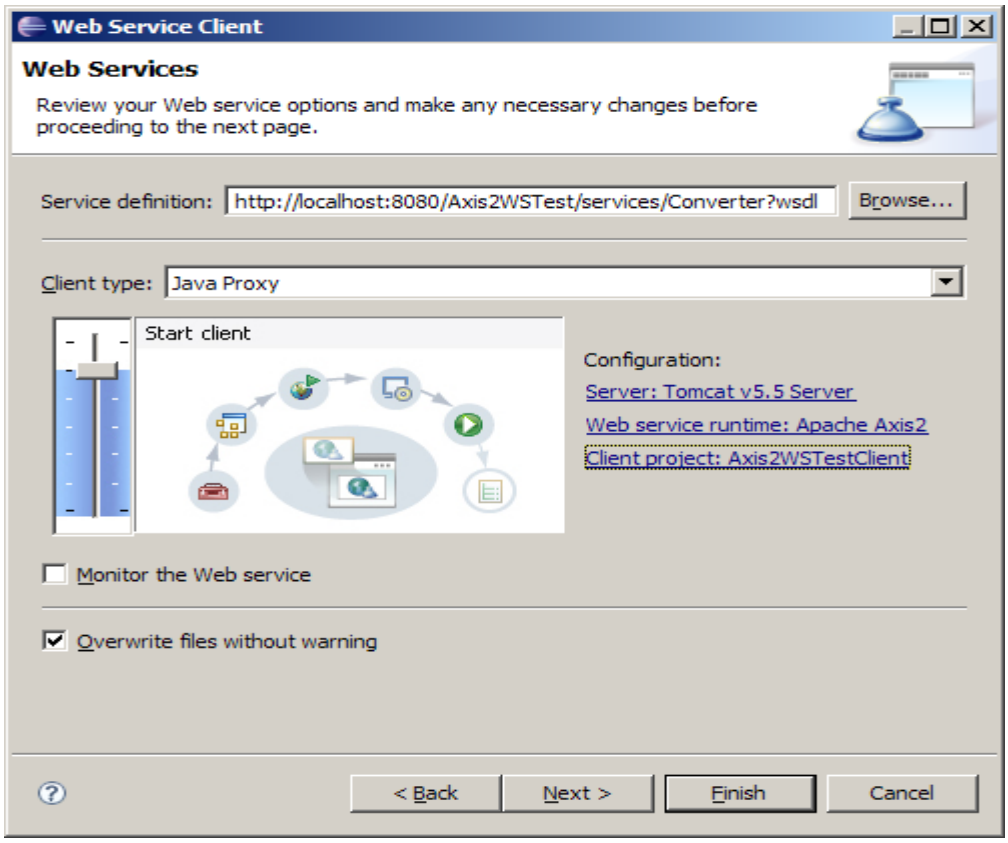
u. Paste the URL that was copied earlier into the service definition field.



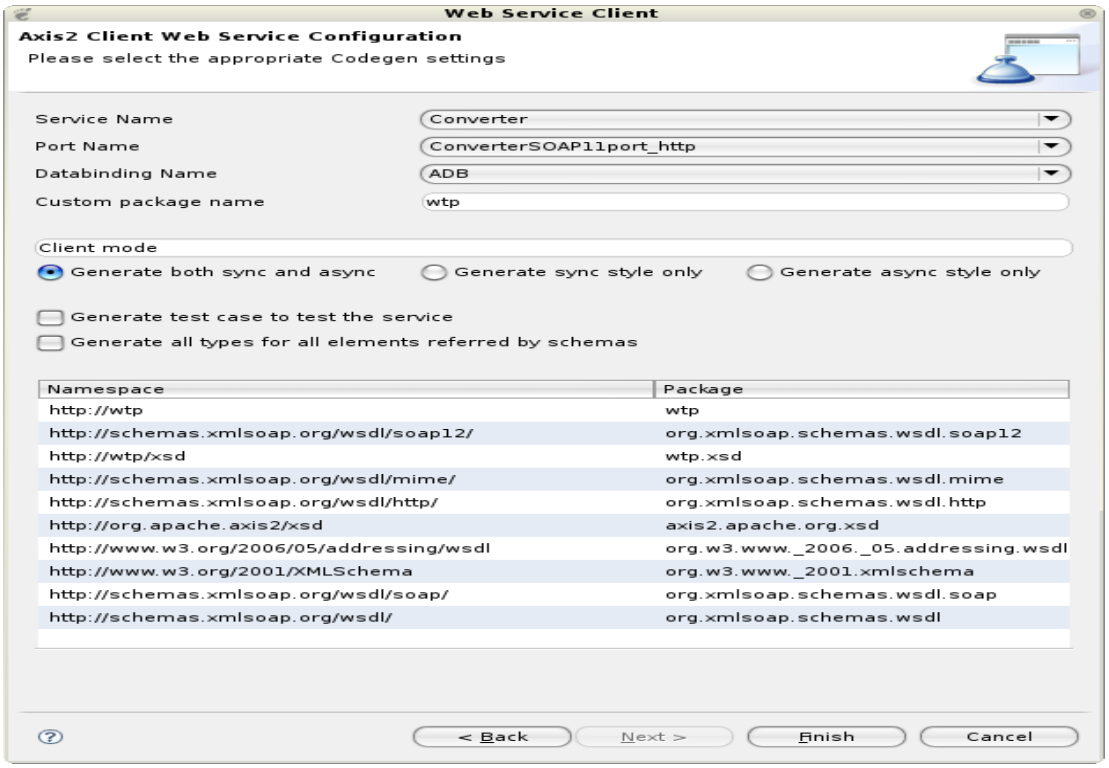
v. Click on the **Client project** hyperlink and enter **Axis2WSTestClient** as the name of the client project. Click OK.



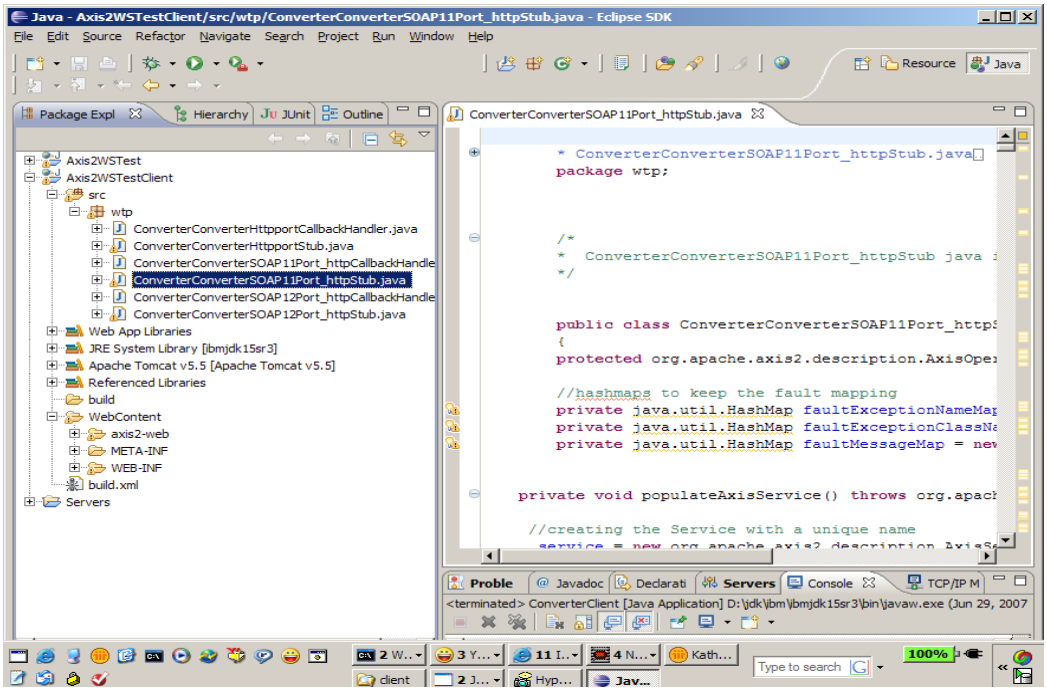
Back on the Web Services Client wizard, make sure the Web service runtime is set to Axis2 and the server is set correctly. Click Next.



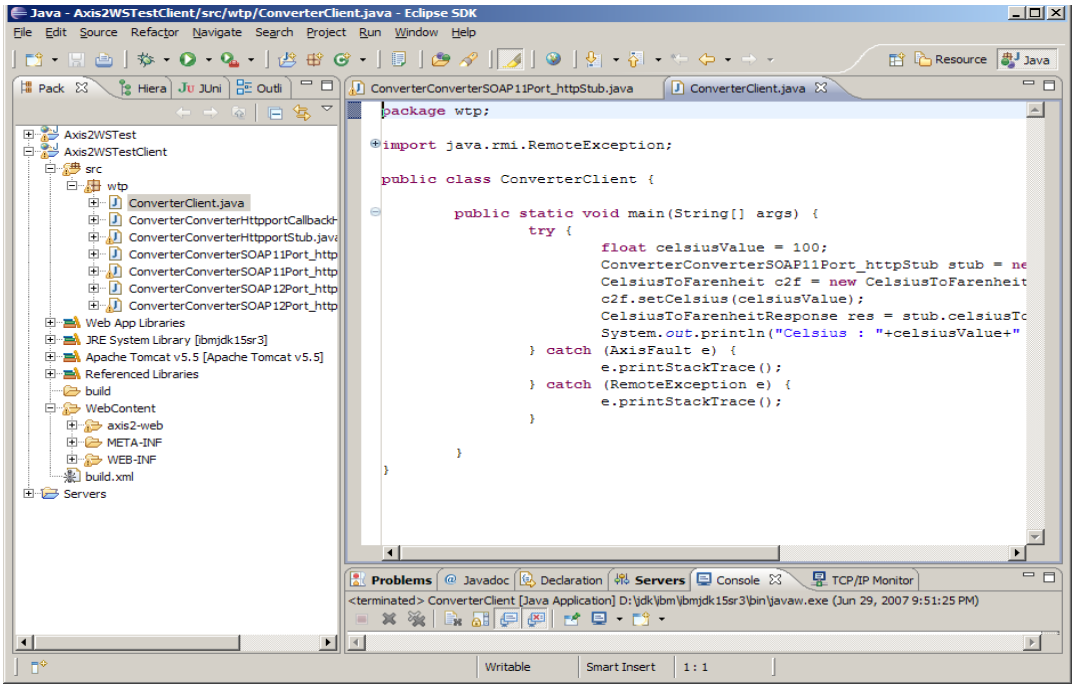
Next page is the Client Configuration Page. Accept the defaults and click Finish.



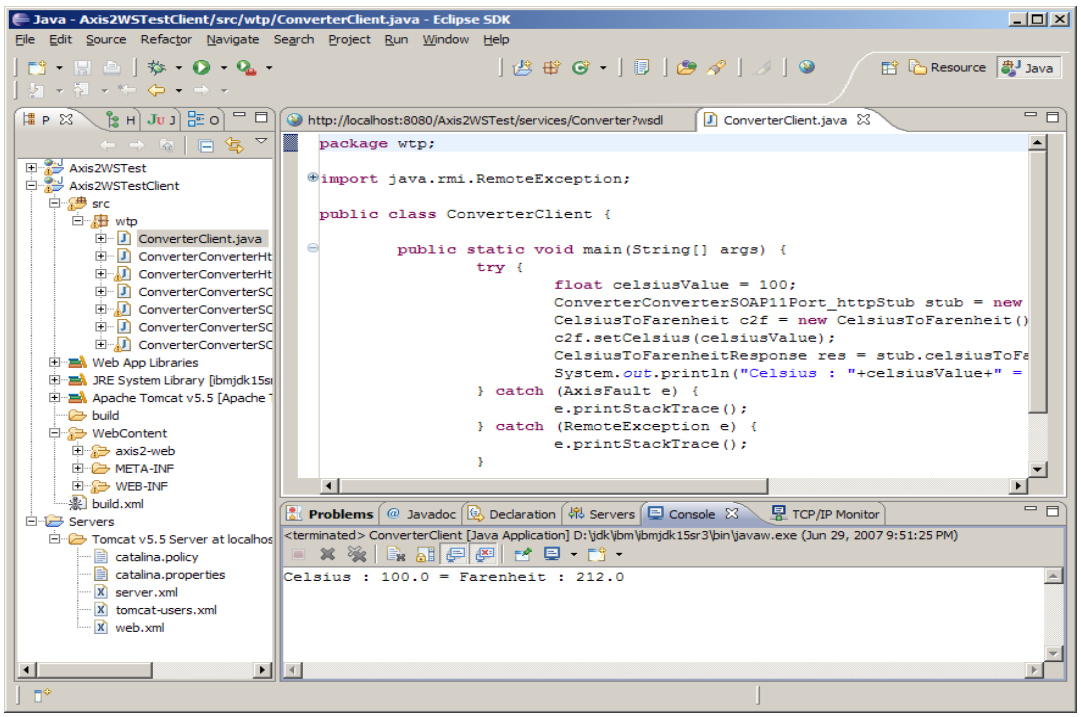
The Clients stubs will be generated to your Dynamic Web project **Axis2WSTestClient**.



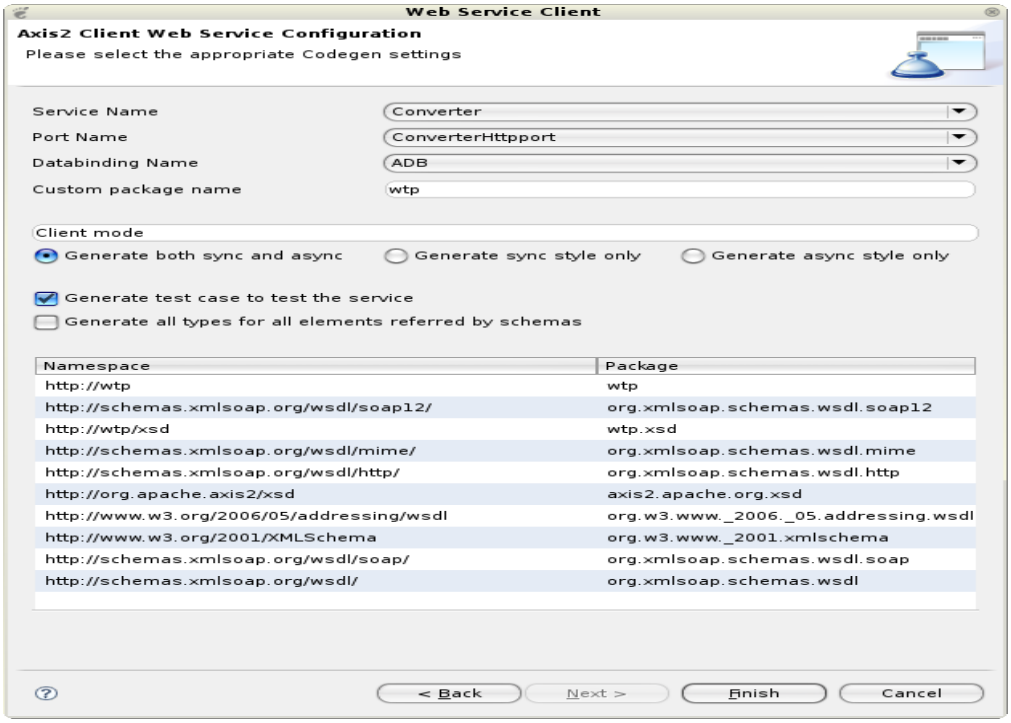
Now we are going to write Java main program to invoke the client stub. Import the ConverterClient.jav file to the workspace into the wtp package in the src folder of **Axis2WSTestClient**.



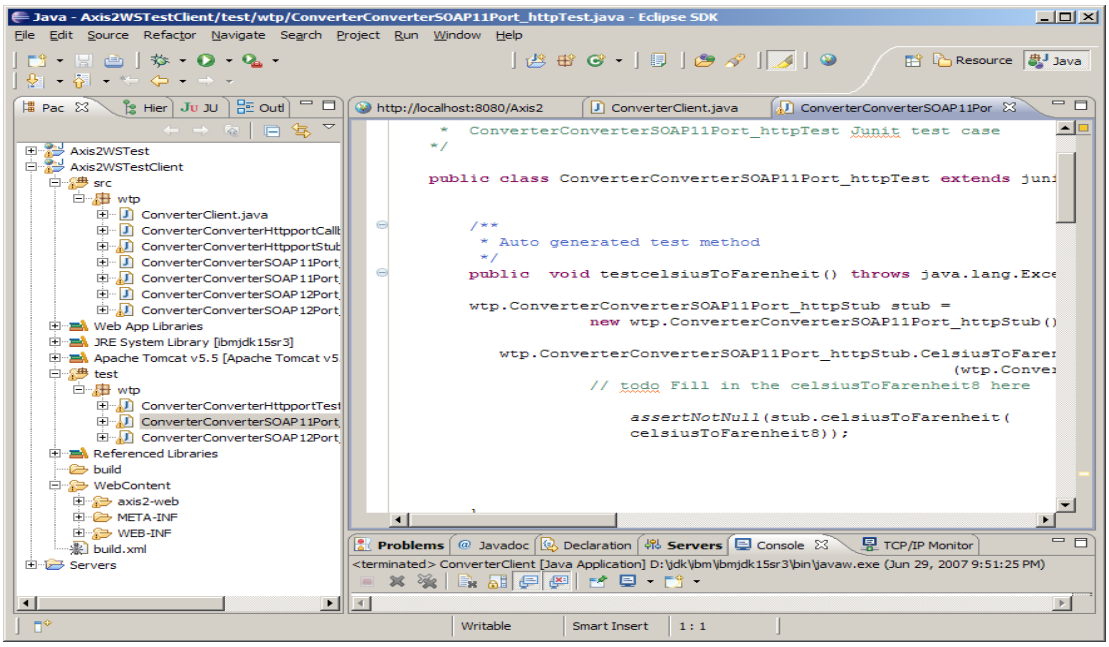
Then select the ConverterClient file, right-click and select Run As -> Java Application. Here's what you get on the server console:



Another way to test and invoke the service is to select **Generate test case to test the service** check box on the Axis2 Client Web Service Configuration Page when going through the Web Service Client wizard.



If that option is selected, the Axis2 emitter will generate JUnit testcases matching the WSDL we provide to the client. These JUnit testcases will be generated to a newly added source directory to the **Axis2WSTestClient** project called **test**.

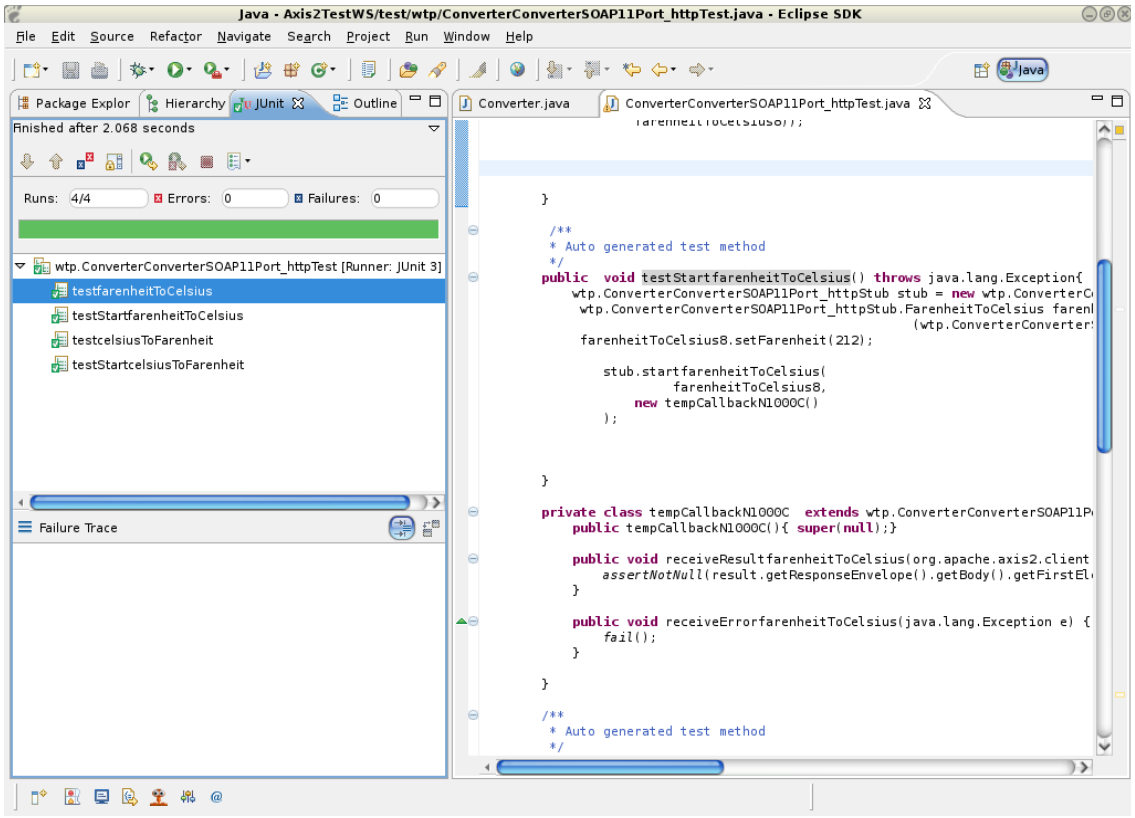


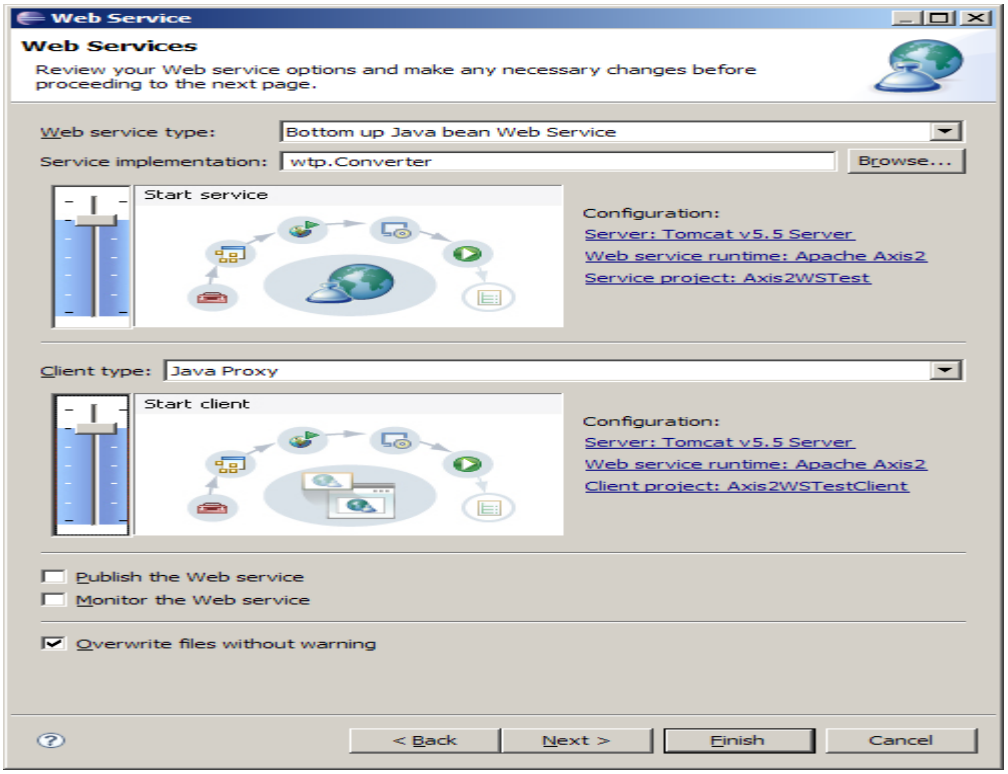
Next thing we need to do is to insert the test case with the valid inputs as the Web service method arguments. In this case, let's test the ConverterConverterSOAP11Port_httpTest.java by provide values for Celsius and Farenheit for the temperature conversion. As an example, replace the generated TODO statement in each test method to fill in the data with values as:

```
testfarenheitToCelsius() -> farenheitToCelsius8.setFarenheit(212);

testStartfarenheitToCelsius() ->farenheitToCelsius8.setFarenheit(212);

testcelsiusToFarenheit() -> celsiusToFarenheit10.setCelsius(100);

testStartcelsiusToFarenheit() -> celsiusToFarenheit10.setCelsius(100);
```

Here the testcases were generated to test both the synchronous and asynchronous clients.

w. After that, select the testcase, right-click, select Run As -> JUnit Test. You will be able to run the unit test successfully invoking the Web service.


Java - Axis2TestWS/test/wtp/ConverterConverterSOAP11Port_httpTest.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Package Explor
Hierarchy
JUnit
Outline
Finished after 2.068 seconds
Runs: 4/4
Errors: 0
Failures: 0
wtp.ConverterConverterSOAP11Port_httpTest [Runner: JUnit 3]
testfarenheitToCelsius
testStartfarenheitToCelsius
testcelsiusToFarenheit
testStartcelsiusToFarenheit
Failure Trace
Converter.java
ConverterConverterSOAP11Port_httpTest.java
/**
* Auto generated test method
*/
public void testStartfarenheitToCelsius() throws java.lang.Exception{
wtp.ConverterConverterSOAP11Port_httpStub stub = new wtp.ConverterC
wtp.ConverterConverterSOAP11Port_httpStub.FarenheitToCelsius farenh
(wtp.ConverterConverter
farenheitToCelsius8.setFarenheit(212);
stub.startfarenheitToCelsius(
farenheitToCelsius8,
new tempCallbackN1000C()
);
}
private class tempCallbackN1000C extends wtp.ConverterConverterSOAP11P
public tempCallbackN1000C(){ super(null);}
public void receiveResultfarenheitToCelsius(org.apache.axis2.client
assertNotNull(result.getResponseEnvelope().getBody().getFirstEl
}
public void receiveErrorfarenheitToCelsius(java.lang.Exception e) {
fail();
}
}
/**
* Auto generated test method
*/



The Web Service wizard orchestrates the end-to-end generation, assembly, deployment, installation and execution of the Web service and Web service client. Now that your Web service is running, there are a few interesting things you can do with this WSDL file. Examples:

- You can choose Web Services -> Test with Web Services Explorer to test the service.
- You can choose Web Services -> Publish WSDL file to publish the service to a public UDDI registry.

RESULT:

Thus the development of a new OGSA-compliant web service was executed successfully.

3.Using Apache Axis develop a Grid Service

OBJECTIVE:

To develop a Grid Service using Apache Axis.

PROCEDURE:

You will need to download and install the following software:

1. Java 2 SDK v1.4.1, http://java.sun.com/j2se/1.4.1/download.html
2. Apache Tomcat v4.124

http://jakarta.apache.org/builds/jakarta-tomcat-4.0/release/v4.1.24/bin/jakarta tomcat4.1.24.exe.

3. XML Security v1.0.4,

http://www.apache.org/dist/xml/security/java-library/xmlsecurity bin1.0.4.zip

4. Axis v1.1, http://ws.apache.org/axis/dist/1_1/axis-1_1.zip

1. Java 2 SDK

- Run the downloaded executable (j2sdk-1_4_1-windows-i586.exe) which will install the
- SDK in C:\j2sdk1.4.1. Set the JAVA_HOME environment variable to point to this directory as follows:
- Click on START->CONTROL PANEL->SYSTEM
- Click on the Advanced tab
- Click on the Environment Variables button
- Click on the New… button in the user variable section and enter the details
- Add the Java binaries to your PATH variable in the same way by setting a user variable called PATH with the value "%PATH%;C:\j2sdk1.4.1\bin"

2. Apache Tomcat

- Run the downloaded executable (jakarta-tomcat-4.1.24.exe), and assume the installation directory is C:\jakarta-tomcat-4.1.24.
- Edit C:\ jakarta-tomcat-4.1.24\conf\tomcat-users.xml and create an “admin” and “manager” role as well as a user with both roles. The contents of the file should be similar to:

```
<?xml version='1.0' encoding='utf8'?>

<tomcat-users>

<role rolename="manager"/>

<role rolename="admin"/>

<user username="myuser" password="mypass" roles="admin,manager"/>

</tomcat-users>
```

- Start Tomcat by running C:\ jakarta-tomcat-4.1.24\bin\startup.bat and test it by browsing http://localhost:8080/
- Stop Tomcat by running C:\ jakarta-tomcat-4.1.24\bin\shutdown.bat.

3. XML Security

- Download and unzip

  http://www.apache.org/dist/xml/security/javalibrary/xmlsecurity-bin 1_0_4.zip
- Copy xml-sec.jar to C:\axis-1_1\lib\
- Set-up your CLASSPATH environment variable to including the following:

C:\axis1_1\lib\xml-sec.jar;

4. Apache Axis

- Unzip the downloaded Axis archive to C: (this will create a directory C:\axis-1_1).
- Extract the file xmlsec.jar from the downloaded security archive to

C:\axis1_1\webapps\axis\WEB-INF\lib.

- Set-up your CLASSPATH environment variable to including the following:
  - The current working directory
  - All the AXIS jar files as found in C:\axis-1_1\lib

    C:\jakarta-tomcat-4.1.24\common\lib\servlet.jar
- Your CLASSPATH should therefore look something like:

  C:\axis-1_1\lib\axis.jar;

  C:\axis 1_1\lib\axis-ant.jar;

  C:\axis-1_1\lib\commons-discovery.jar;

  C:\axis-1_1\lib\commons-logging.jar;

  C:\axis-1_1\lib\jaxrpc.jar;

  C:\axis-1_1\lib\log4j-1.2.8.jar;

  C:\axis-1_1\lib\saaj.jar;

  C:\axis-1_1\lib\wsdl4j.jar;

  C:\axis-1_1\lib\xercesImpl.jar

  C:\axis-1_1\lib\xmlParserAPIs.jar;

  C:\jakarta-tomcat-4.1.24\common\lib\servlet.jar

  C:\axis-1_1\lib\xml-sec.jar;
- Now tell Tomcat about your Axis web application by creating the file

  C:\jakarta- tomcat-4.1.24\webapps\axis.xml with the following content:

  <Context path="/axis" docBase="C:\axis-1_1\webapps\axis" debug="0"

  privileged="true">

  <LoggerclassName="org.apache.catalina.logger.FileLogger"prefix="axis_log."

  suffix=".txt" timestamp="false"/>

5. Deploy a Sample Web service packaged within Axis installations

Deploy one of the sample Web Services to test the system and to create the C:\axis-1_1\webapps\axis\WEB-INF\server-config.wsdd file. From C:\axis-1_1 issue the command (on one line):

java org.apache.axis.client.AdminClient

http://localhost:8080/axis/services/AdminService/samples/stock/deploy.wsdd

This should return the following:

.– Processing file samples/stock/deploy.wsdd

.– <Admin>Done processing</Admin>

RESULT:

Thus the development of a Grid Service using Apache Axis is executed successfully.

4. Develop applications using Java or C/C++ Grid APIs

OBJECTIVE:

To develop an applications using Java or C/C++ Grid APIs.

SAMPLE CODE:

```
import AgentTeamwork.Ateam.*;
import MPJ.*;
public class UserProgAteam extends AteamProg {
private int phase;
public UserProgAteam( Ateam o )
{ }
public UserProgAteam( )
```

```
{ }
// real const
public UserProgAteam( String[] args ) {
phase = 0;
}
// phase recovery
private void userRecovery( ) {
phase = ateam.getSnapshotId( );
}
private void compute( ) {
for ( phase = 0; phase < 10; phase++ ) {
try {
Thread.currentThread( ).sleep( 1000 );
}
catch(InterruptedException e ) {
}
ateam.takeSnapshot( phase );
System.out.println( "UserProgAteam at rank " + MPJ.COMM_WORLD.Rank( ) + " : took a snapshot " + phase );
}
}
public static void main( String[] args ) {
System.out.println( "UserProgAteam: got started" );
MPJ.Init( args, ateam);
UserProgAteam program = null;
// Timer timer = new Timer( );
if ( ateam.isResumed( ) ) {
```

```
program = ( UserProgAteam )
ateam.retrieveLocalVar( "program" );
program.userRecovery( );
}
else
{
program = new UserProgAteam( args );
ateam.registerLocalVar( "program", program );
}
program.compute( );
MPJ.Finalize( );
}
public class UserProgAteam extends AteamProg {
// application body private void compute( ) {
for ( phase = 0; phase < 10; phase++ ) {
try {
Thread.currentThread( ).sleep( 1000 );
}
catch(InterruptedException e ) {
}
ateam.takeSnapshot( phase );
System.out.println ( "UserProgAteam at rank " + MPJ.COMM_WORLD.Rank( ) + " : took a snapshot " + phase );
}}
```

Socket sample code – within some function body

```
import AgentTeamwork.Ateam.GridTcp.*;
private final int port = 2000;
```

```
private GridSocket socket; private GridServerSocket
server; private InputStream input; private OutputStream
output;

for ( int i = start; i < start + trans; i++ ) {

try {

output.write( i % 128 );

} catch ( IOException e ) {

}

System.out.println ( "Sockets with " + myRank + ": " + " output[" + i + "]=" + i % 128 );

}

for ( int i = start; i < start + trans; i++ ) {

try {

System.out.println ( "Sockets with " + myRank + ": " + " input[" + i + "]=" + input.read( ) ); }

catch ( IOException e ) {

}}
```

MPI sample code

```
import AgentTeamwork.Ateam.*;

import MPI.*;

public class UserProgAteam extends AteamProg {

// application body private void compute( ) {

}

public static void main( String[] args ) {

MPJ.Init( args, ateam );

program.compute( ); MPJ.Finalize( );

}

}
```

C/C++ compile.sh – Helloworld.cpp

```
#!/bin/sh

rm -f *.class

javac -classpath MPJ.jar:Ateam.jar:. *.java

# jar cvf GridJNI.jar *.class jar -cvf GridJNI.jar *.class
javah -jni JavaToCpp

g++ -rdynamic JavaToCpp.cpp -o _libJavaToCpp.so_ -shared -ldl g++ -shared -o _libHelloWorld.so_
GridJNI_library.cpp

HelloWorld.cpp
```

C/C++ MPI sample code – Helloworld.cpp

```
#include <iostream.h>

using namespace std;

typedef int MPI_Request, MPI_Status, MPI_Comm;

extern void takeSnapshot(int argc);

extern int MPI_Init(int* argc, char*** argv);

extern void MPI_Finalize();

extern int MPI_Comm_rank(MPI_Comm comm, int *rank);

extern int MPI_Comm_size(MPI_Comm comm, int *size);

int main(int argc, char** argv) {

cerr << "main" << endl;

cerr << "argc = " << argc << endl;

cerr << "argv[0] = " << argv[0] << endl; cerr << "argv[1] = " << argv[1] <<
endl; MPI_Init(&argc, &argv);

cout << "MPI Init Successful!" << endl;

cout << "[HelloWorld.cpp]Calling Rank() and Size()" << endl;
```

```
int rank, size;

MPI_Comm_rank(0,&rank);
MPI_Comm_size(0,&size);

cout << "[HelloWorld.cpp]Rank = " << rank << endl;

cout << "[HelloWorld.cpp]Size = " << size << endl; cerr << "Calling MPI_Finalize()" <<
endl; MPI_Finalize();

cerr << "finished" << endl;

}
```

RESULT:

Thus the development of applications using Java or C/C++ Grid APIs is executed successfully

5.Develop secured applications using basic security mechanisms available in Globus Toolkit

OBJECTIVE:

To develop a secured applications using basic security mechanisms available in Globus.

PROCEDURE:

The Globus Toolkit's Authentication and Authorization components provide the *de facto* standard for the "core" security software in Grid systems and applications. These software development kits (SDKs) provide programming libraries, Java classes, and essential tools for a PKI, certificate-based authentication system with single sign-on and delegation features, in either Web Services or non-Web Services frameworks. ("Delegation" means that once someone accesses a remote system, he can give the remote system permission to use his credentials to access others systems on his behalf.)

WEB SERVICES AUTHENTICATION AND AUTHORIZATION –

A Web services implementation of the Grid Security Infrastructure (GSI), containing the core libraries and tools needed to secure applications using GSI mechanisms. The Grid is a term commonly used to describe a distributed computing infrastructure which will allow "coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations" . The protocols and middleware to enable this Grid infrastructure have been developed by a number of initiatives, most notably the Globus Project .

Web Services are simply applications that interact with each other using Web standards, such as the HTTP transport protocol and the XML family of standards. In particular, Web Services use the SOAP messaging standard for communication between service and requestor. They should be self-describing, self-contained and modular; present a platform and implementation neutral connection layer; and be based on open standards for description, discovery and invocation.

The Grid Security Infrastructure (GSI) is based on the Generic Security Services API (GSS-API) and uses an extension to X509 certificates to provide a mechanism to authenticate subjects and authorise resources. It allows users to benefit from the ease of use of a single sign-on mechanism by using delegated credentials, and time-limited proxy certificates. GSI is used as the security infrastructure for the Globus Toolkit.

Recently, a new proposal for an Open Grid Services Architecture (OGSA) was announced which marries the Grid and Web Services to create a new Grid Services model. One problem, which has not yet been explicitly addressed, is that of security. A possible solution is to use a suitably secure transport binding, e.g. TLS, and extend it to incorporate appropriate support for proxy credentials. It would be useful to test out some of the principles of Grid Services using the currently available frameworks and tools for developing Web Services. Unfortunately, no standards currently exist for implemented proxy credential support to provide authenticated communication between web services. A number of XML/Web Services security standards are currently in development, e.g. XML Digital Signatures, SAML, XKMS, XACML, but the remainder of this document describes an approach proposed by ANL to use GSI over an SSL link.

A generic Job Submission environment, GAP enables researchers and scientists to execute their applications on Grid from a conventional web browser. Both Sequential and Parallel jobs can be submitted to GARUDA Grid through Portal. It provides a web interface for viewing the resources, and for submitting and monitoring jobs.

Pre-requisites for using GAP

Portal users need to set the following in their ~/.bashrc file.

export GLOBUS_LOCATION=/opt/asvija/GLOBUS-4.0.7/

source /opt/asvija/GLOBUS-4.0.7/etc/globus-user-env.sh

export PATH=/usr/local/jdk1.6.0_10/bin:
GW_LOCATION/bin:/opt/garudaresv/bin:/opt/voms_client/bin:$PATH

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/voms_client/lib:

Accessing GAP

Type http://192.168.60.40/GridPortal1.3/ (to access the Portal through GARUDA Network) or http://203.200.36.236/GridPortal1.3 (to access the Portal through Internet) in the address bar of the web browser to invoke the Portal. It is preferable to access the Portal through GARUDA Network, since it is much faster than the Internet.

In order to access the facilities of Grid Portal such as Job Submission, Job Status tracking, Storing(Uploading) of Executables and View Output/Error data, the user has to login into the

Portal using the User's Login Form in the Home page of the Portal.

a) New users are required to click Sign up in the User Login Form, which leads them to home page of Indian Grid Certification Authority (IGCA) (http://ca.garudaindia.in/). Click on Request Certificate and acquire the required user/host certificate(s), details are provided in IGCA section.

b) Registered users are required to provide User Id and Password for logging into the Portal and access various facilities.

Job Management

User can submit their job, monitor the status and view output files using the Job Management interfaces. Types of job submission (Basic and Advanced) and Job information are covered under this section.

Basic Job Submission

This interface can be used to submit sequential as well as parallel jobs. The user should provide the following information:

1. Optional Job Name - User can provide a suitable (alias) name for their job.

2. Type of Job user want to execute,

3. Operating System – Required for their Job,

4. 'Have you reserved the Resources' - An optional parameter contains the Reservation Id's that can be used for job submission instead of choosing the Operating System/Processor parameter.

5. No. of processes required for the job - This parameter is only for the parallel applications that require more than one CPU.

6. Corresponding Executables – uploaded from either local or remote machine,

7. Input file, if required - The executable and the input file can either be uploaded from the local machine or can be selected from the Remote File List, if it is available in the Submit Node

8. STDIN - Required when the user wants to provide any inputs to the application during the runtime.

9. Optional Execution Time - Here the Execution Time is the anticipated job completion time.

10. Any Command Line arguments or Environment Variables, if required.

11. User Specific Output/ Error files - If the application generates output/error files other than standard output/error files and its entries should be separated by comma's or single empty space in case of multiple files.

All those fields marked with * are mandatory fields and should be filled before submitting a job. By clicking on submit button, the portal submits the job to Grid Way Meta Scheduler, which then schedules the job for execution and returns the Job Id. The Job Id has to be noted for future reference to this job. In the event of unsuccessful submission, the corresponding error message is displayed.

All those fields marked with * are mandatory fields and should be filled before submitting a job. By clicking on submit button, the portal submits the job to GridWay Meta Scheduler, which then schedules the job for execution and returns the Job Id. The Job Id has to be noted for future reference to this job.

Advanced Job Submission

This interface is provided for the user to submit their Sequential and Parallel Jobs. The difference from Basic job submission being: it is using GT4 Web Services components for submitting jobs to the Grid instead of Gridway as scheduler.

The user is provided with two modes in this interface:

1. Default mode - Portal creates the XML file for the user.

2. Second mode, recommended for advanced users - The user can provide their-own XML file as the executable, provided the required files are available in the submit node.

The user can view the status of the job submitted through Portal and the output file of the job by specifying the Job Id. The option for downloading the Output/ Error file is also provided, after the job execution. To cancel any of the queued jobs, the user has to select the job and click Cancel Job button, following which the acknowledgment for the job canceled is provided.

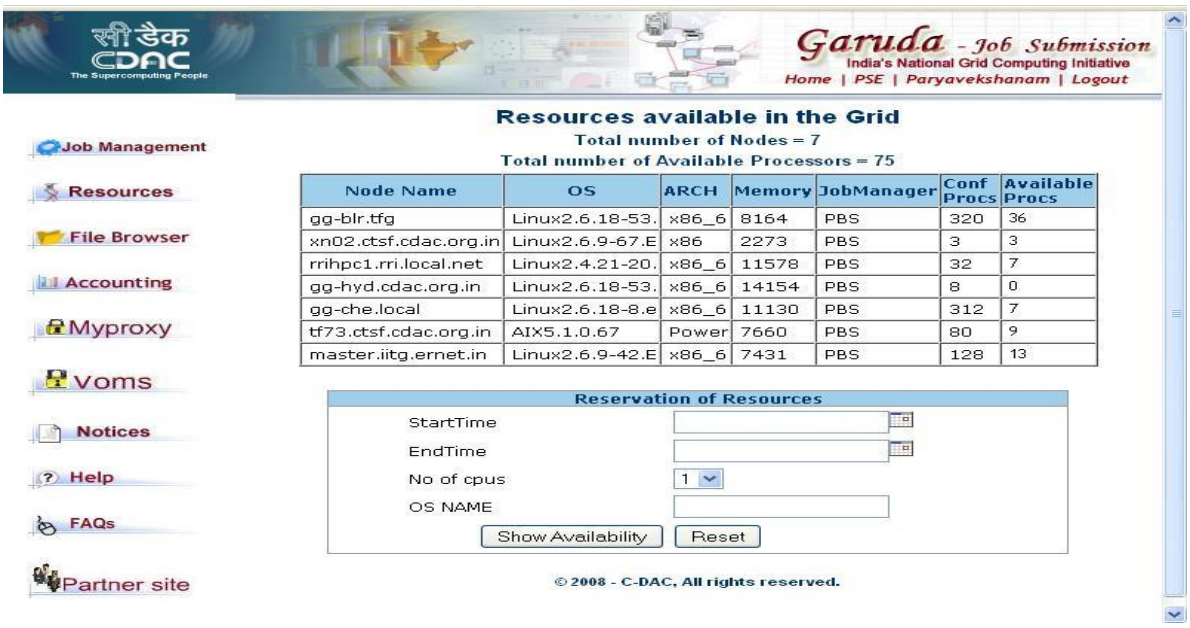
Job Info

The user can view the status of the job submitted through Portal and the output file of the job by specifying the Job Id. The option for downloading the Output/ Error file is also provided, after the job execution. To cancel any of the queued jobs, the user has to select the job and click

Cancel Job button, following which the acknowledgment for the job canceled is provided.

Resources

The GridWay meta-scheduler provides the following information - Node Name, Head Node, OS, ARCH, Load Average, Status, Configured Process and Available Process. This information aids user to select a suitable cluster and reserve them in advance for job submission.



Steps for Reservation of Resources

1. Check the available free resources with valid parameters (Start Time and End Time – duration for which the resource needs to be reserved). The input fields No. of CPUs and OS entries are optional.

Example: starttime= 2009-04-02 17:06:53 endtime=2009-04-02 19:07:10

No. of CPUs=2 OS NAME=Linux

2. Choose the *Available Process* required for the job. Example: Available Procs = 4

3. Select the required resource from the available list of resources.

4. Book the resources for reserving a resource for the requested period of time and process.

5. The reserved resources can be modified/ canceled.

6. Once the reservation process is successfully completed, the Reservation Id is displayed and is made available in the Basic Job Submission page.

File browser

For the logged-in user, the File Browser lists files, such as the uploaded executables and Input/Output/Error files, along with their size and last modified information. It also allows deletion of files.

Accounting

This module provides Accounting information of the jobs that are submitted to GARUDA, such as no. of jobs submitted, and system parameters such as Memory usage, Virtual memory, Wall Time, and CPU time. Last one month data is displayed by default.

MyProxy

MyProxy allows user to upload their Globus Certificates into Myproxy Server and the same can be used for initializing the Grid proxy on the Grid. If the certificate has been already generated for you, but you do not have access to the above- mentioned files, you can download it from GridFS machine (from $HOME/.globus directory) using winscp/scp.

MyProxy Init

By default, the "Myproxy Init" option is enabled for the user. Upload proxy by entering valid inputs - User name, Grid-proxy Passphrase, User certificate file (usercert.pem), User key file (userkey.pem) and Proxy life time (168 hours is the default value).

MyProxyGet

Grid proxy will be initialized on the Grid head node by providing the inputs - User name, Myproxy Passphrase and Life time of the certificate.

VOMS Proxy

The Virtual Organization Management System (VOMS) allows users to belong to Virtual

Organizations (VOs), thereby allowing them to utilize resources earmarked for those VOs.

The user can also request for a new VO by using "Request for VO" link. VOMS proxy initialization with Multiple roles is provided to the user, by selecting more than one entry on the Role combo box.

Steps to be followed to access GSRM from gridfs:

Login to gridfs(192.168.60.40)

Upload your IGCA user certificates

Initialize proxy with grid-proxy-init

Set environmental variables, respectively for whichever client to be used.

Run the SRM commands

GSRM Access points

pvfs2 (172.20.1.81) node should be used to just test all the available SRM client interfaces like StoRM, DPm, BestMan.

gridfs (192.168.60.40) node should, if the user wishes to use GSRM storage for job execution. Users can download/Upload input/output files into GSRM while submitting jobs from gridfs.

Following **Access mechanisms** are available at above mentioned nodes to access GSRM:

1. **gridfs(192.168.60.40)** : gridfs is the Bangalore GARUDA head node. GSRM services can be accessed from here using StoRM command line interface.

If the user wants to use the clientSRM (StoRM Clients) from gridfs machine

Create a valid user proxy using grid-proxy-init

Set the env variable for Globus location path

export GLOBUS_LOCATION= GLOBUS_LOCATION:/usr/local/GARUDA/GLOBUS-4.0.7/

export PATH=$PATH:/opt/gsrm-client/srmv2storm/bin

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/gsrm- client/cgsi_soap/lib

Run the clientSRM command

2. pvfs2 (172.20.1.81): pvfs2 is the GSRM testing node with the following client interfaces installed.

Bestman Java APIs

DPM C APIs

3. GSRM Web Client is accessible from any of the user machines reachable to GSRM server (xn05.ctsf.cdac.org.in), using URL -- **https://xn05.ctsf.cdac.org.in/**

GSRM Client Interfaces

StoRM Command Line Client

1. StoRM command line client format:

clientSRM <requestName> <requestOptions>

2. To get help for clientSRM commands:

clientSRM –h

3. Command to ping to GSRM server:

clientSRM ping -e <GSRM end point>

Bestman Command Line Clients

1. Command to ping to GSRM server

srm-ping –serviceurl httpg://xn05.ctsf.cdac.org.in:8446/dpm/ctsf.cdac.org.in/home/garuda

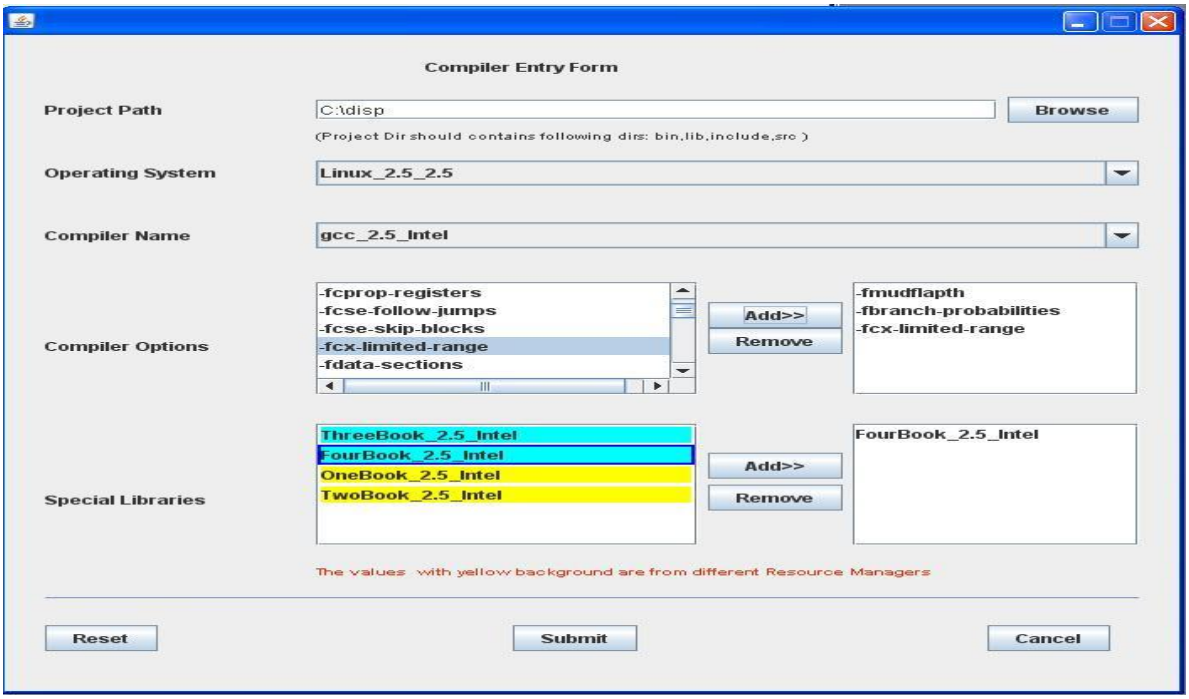
2. Upload file to GSRM server

srm-copy <src url> <target url> <service url>

Pre-requisites for using SOA compiler

1. Java Run Time Environment (JDK1.6+)
2. Web Browser with Java web start support

Compiler GUI



The users are required to adhere to following directory structure. Application Parent Dir-src/,bin/,lib/,include/

1) Login

This method is for logging in to the GARUDA.

Inputs

user name	MyProxy User Name
password	MyProxy Password
life time	Indicates how long is the proxy's life time

Output

Proxy string	Proxy issued by the My proxy server
Login status	Indicates the status of the operation
Last Login Time	Gives when this user was last logged in
Current Login Time	Gives users logging in time

2) uploadProxy

This method uploads a proxy that is generated using other tools, to the MyProxy Server.

Inputs

user name	MyProxy User Name
password	MyProxy Password
proxyBytes	Existing proxy file is given as byte array

Output

uploadStatus	Indicates the status of the operation

3) storeCredential

This method is used for uploading the credentials that is the PKCS12 certificate directly to the MyProxy Server. It will convert the PKCS12 to certificate and stores in server for users to download the proxy until it expires.

Inputs

user name	MyProxy User Name
password	MyProxy Password
p12Bytes	PKCS12 file as byte array

Output

storeStatus	Indicates the status of the operation

Result:

Thus the development of secured applications using basic security mechanisms available in Globus is executed successfully.

6.Develop a Grid portal, where user can submit a job and get the result.

Implement it with and without GRAM concept

OBJECTIVE:

To develop a Grid portal, where user can submit a job and get the result and to implement it with and without GRAM concept.

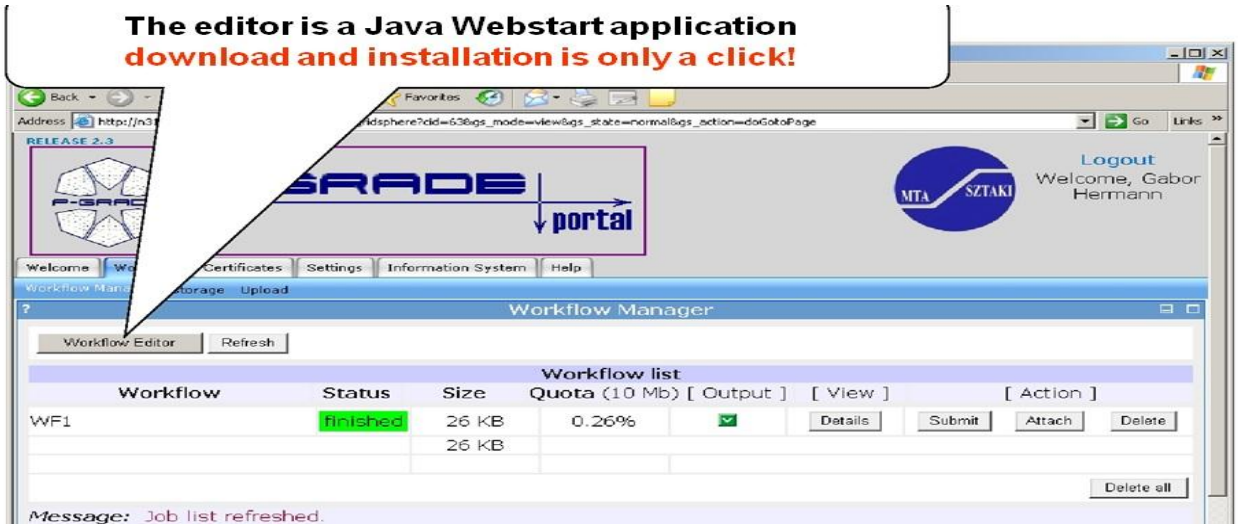
PROCEDURE:

1. Opening the workflow editor

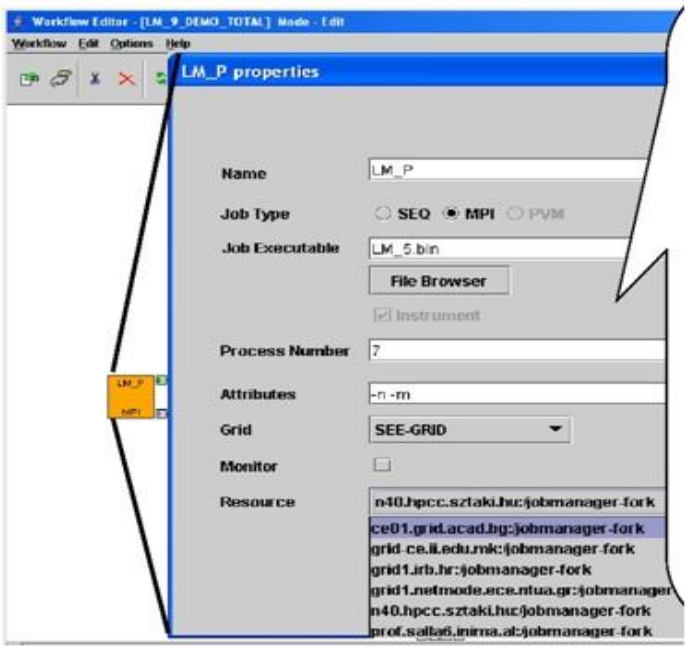
The editor is a Java Webstart application download and installation is only a click.

2. Java Webstart application

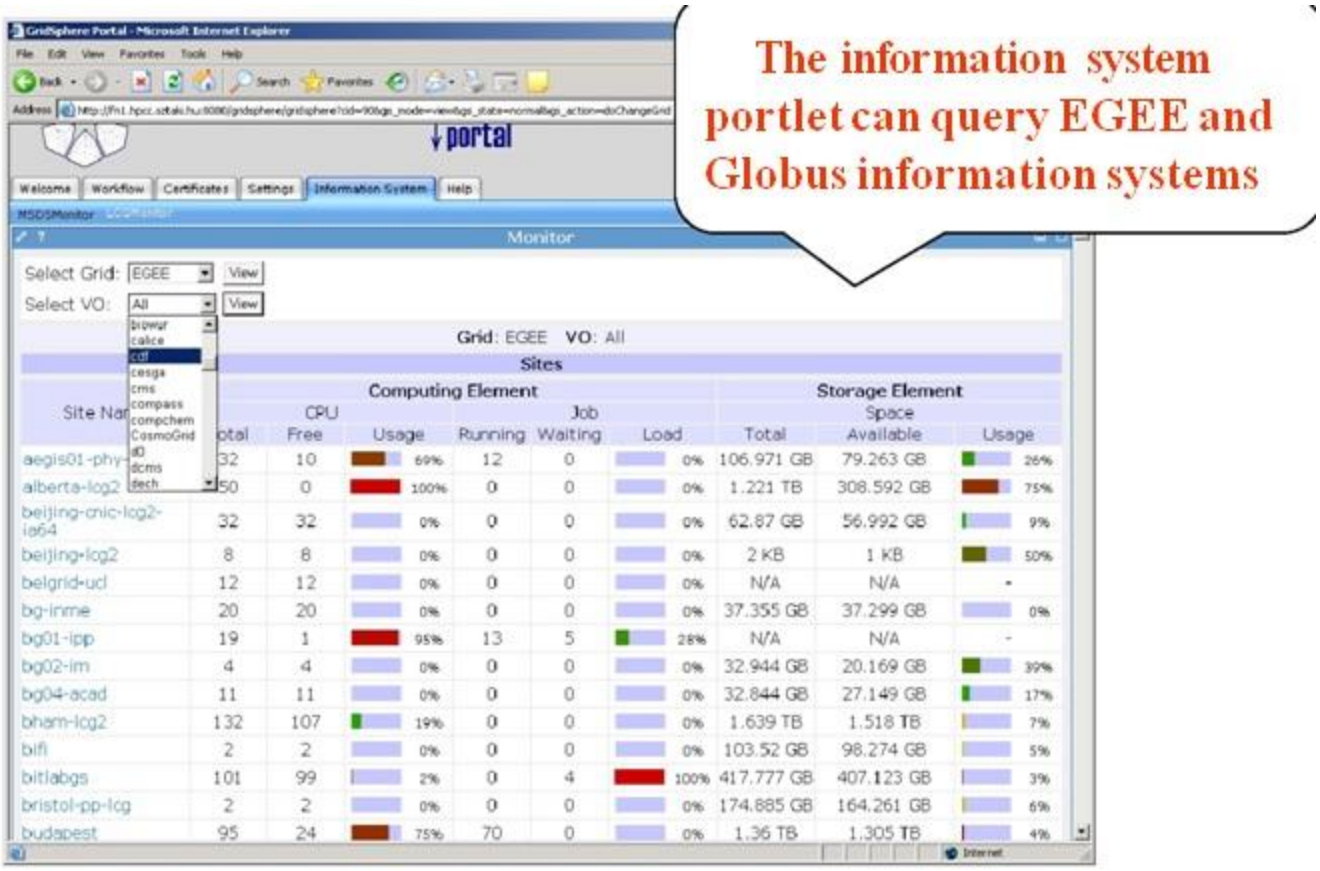
Download and install



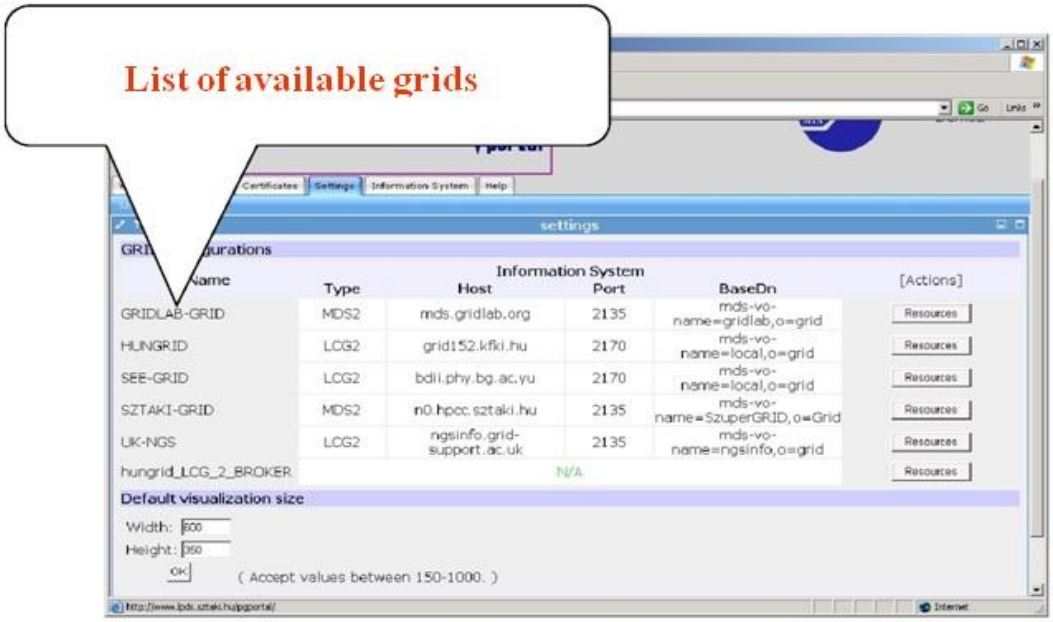
3. Job property window:



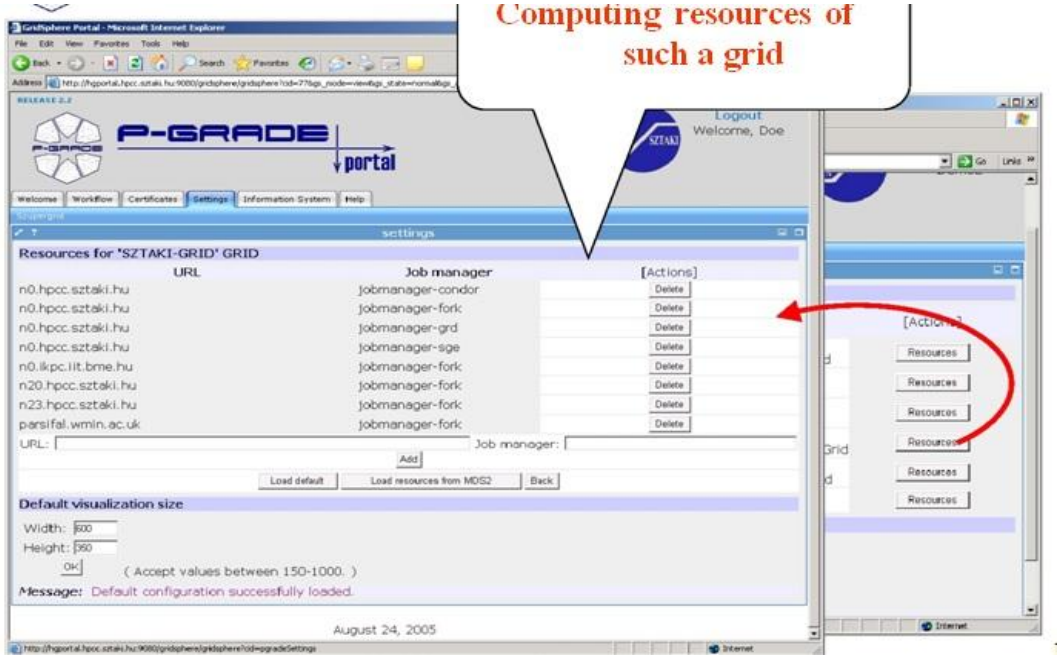
4. The information system can query EGEE and Globus information systems



5. List of available grids



6. Computing resources of such a grid

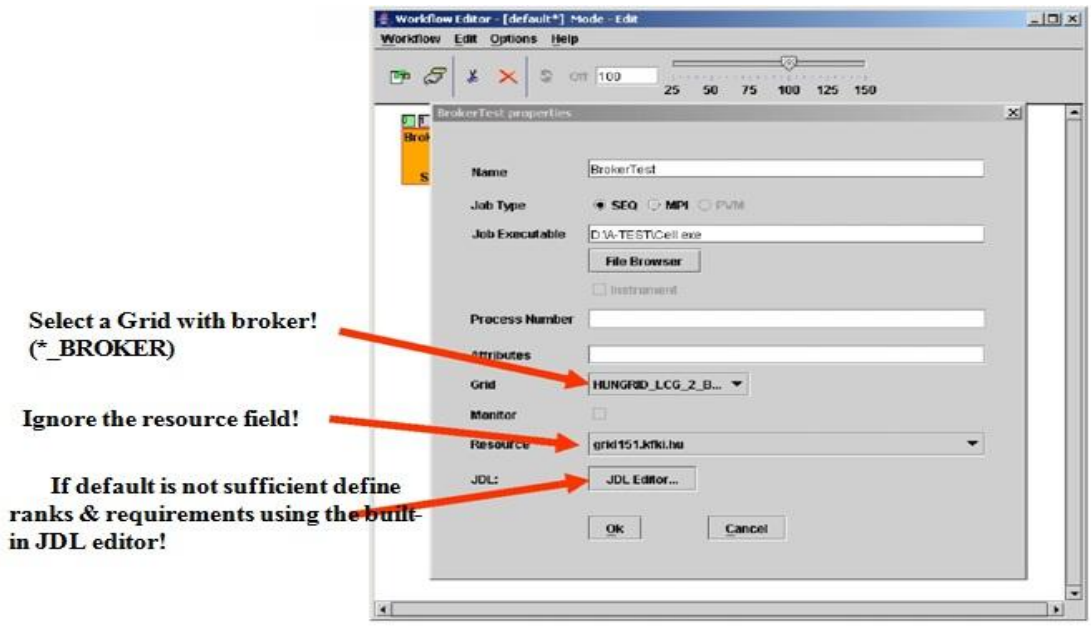


7. Broker resource selection

-> Select a Broker Grid for the job

-> Specify extra ranks and requirements for the job in Job description language.

-> The broker will find the best resource for your job.



8. Defining input/output data for jobs

File type

Input: required by the job

Output: produced by the job

File location:

local: my desktop

remote: grid storage resource

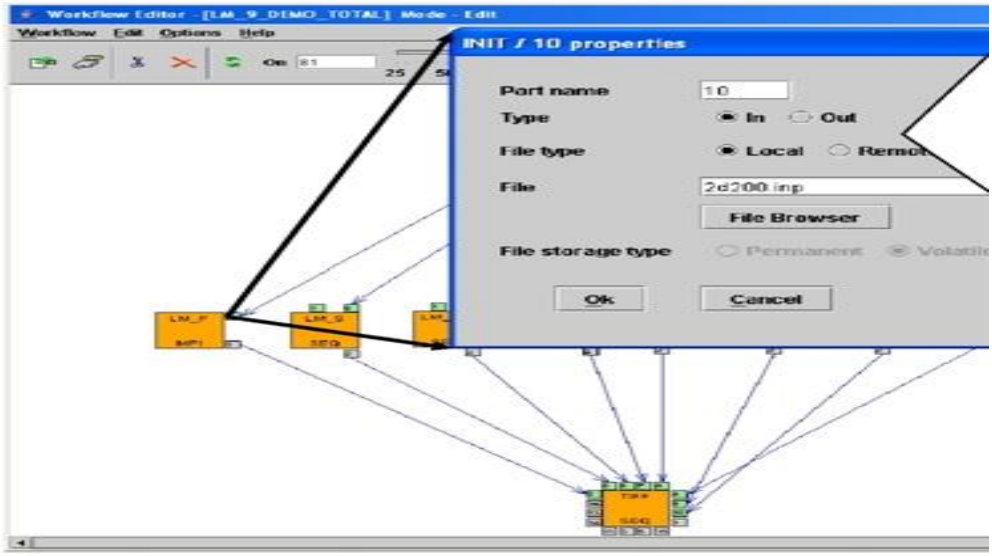
File name:

Unique name of the file

File storage type:

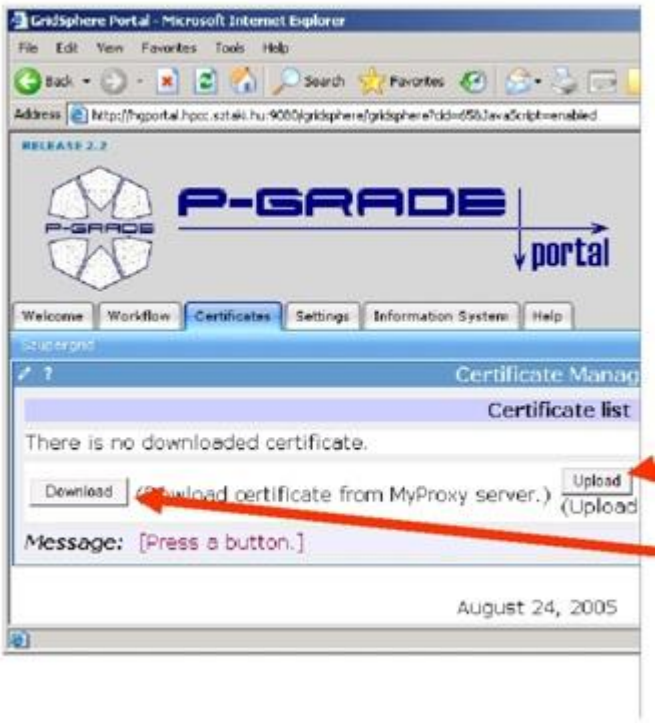
Permanent: final result of WF

Volatile: only used for inter-job data transfer

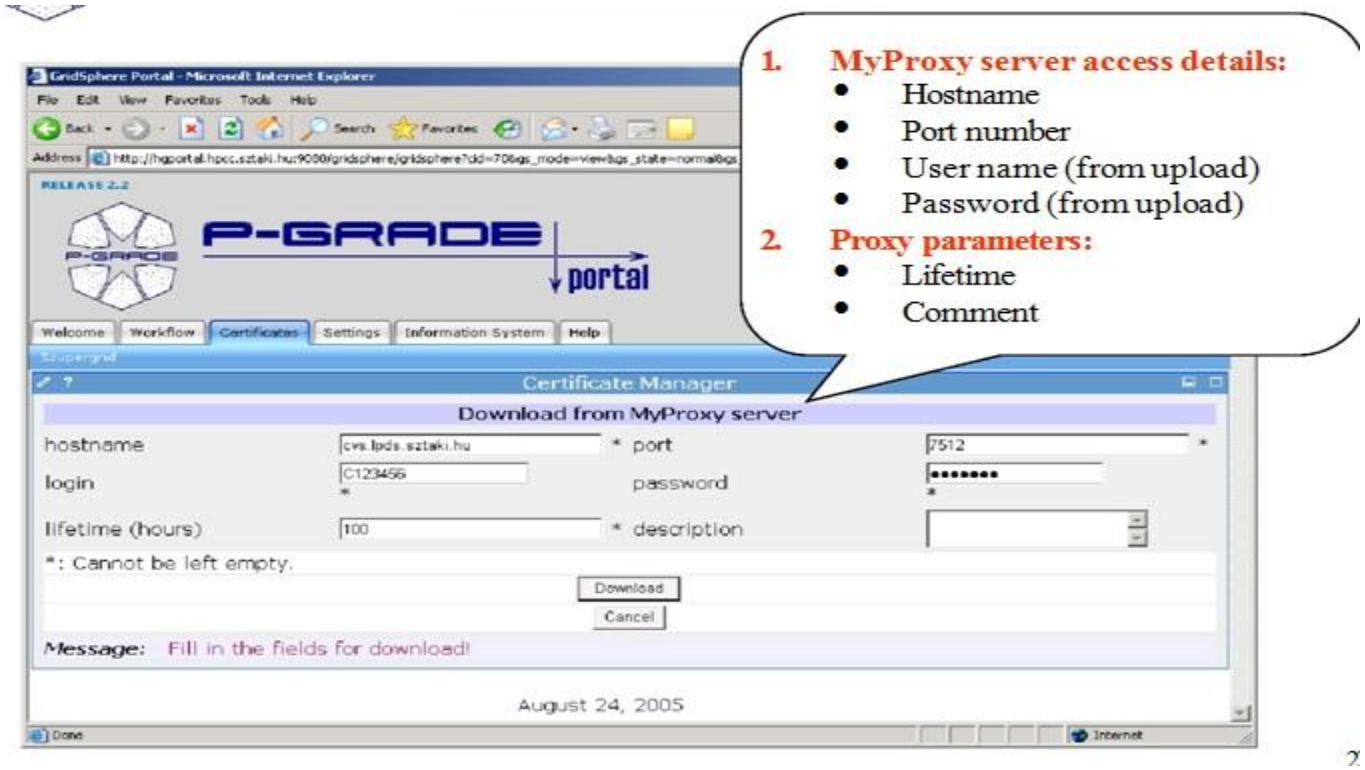


9. Executing workflows with the P-Grade portal

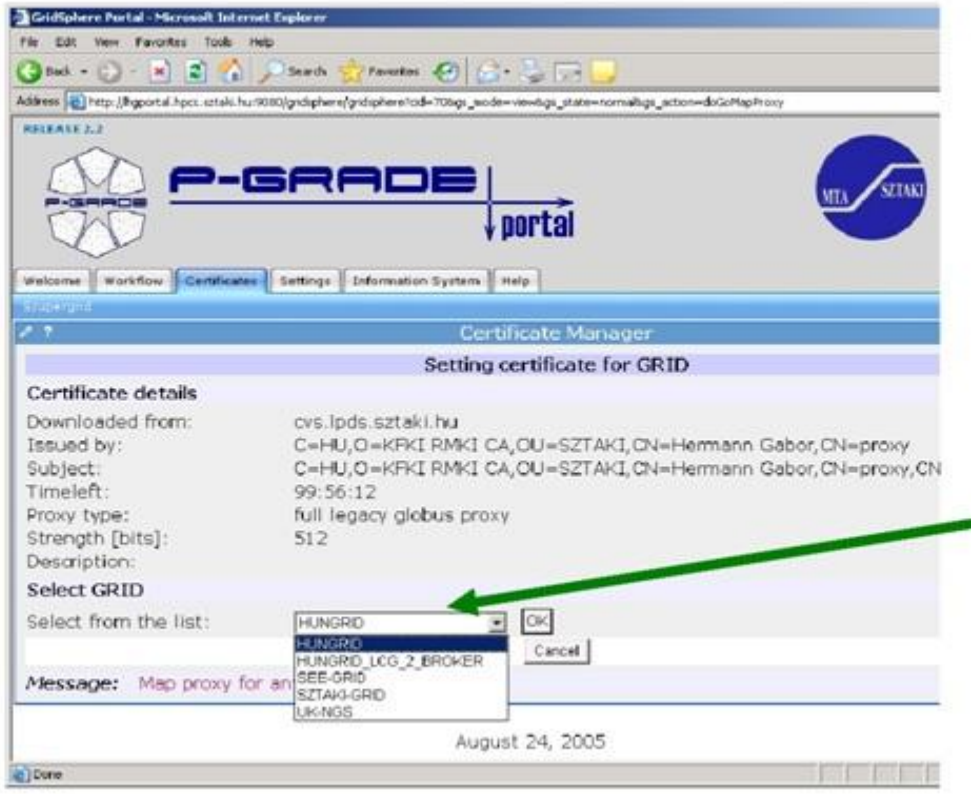
Download proxies



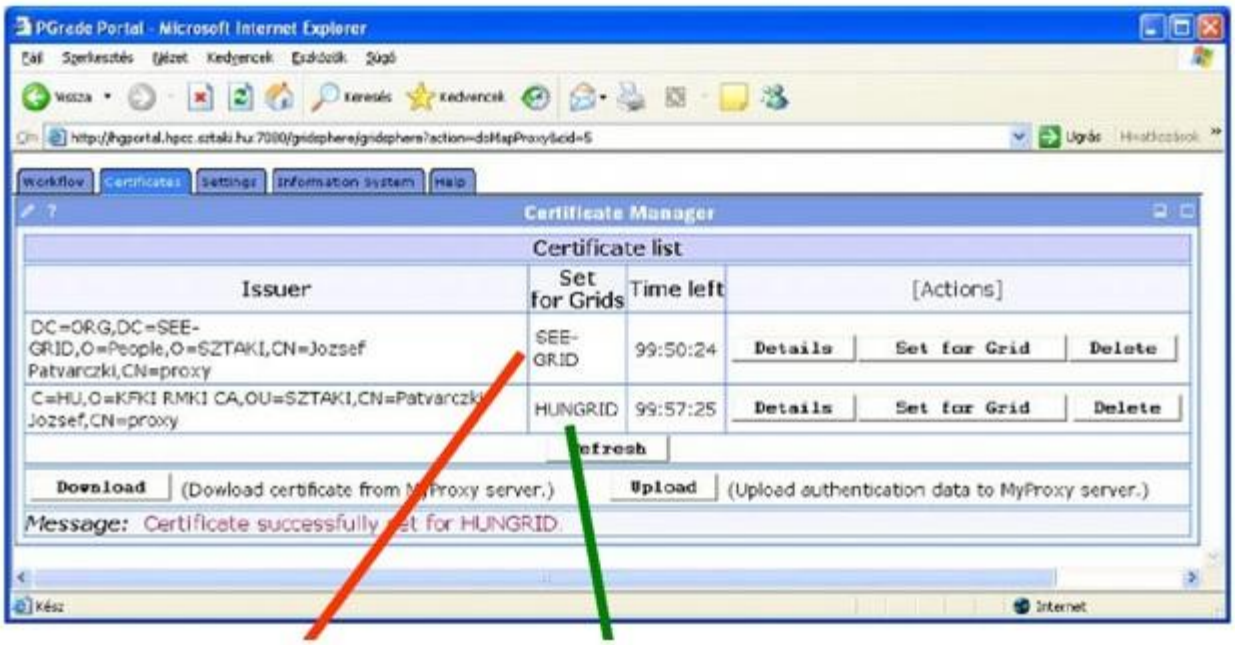
10. Downloading a proxy



11. Associating the proxy with a grid

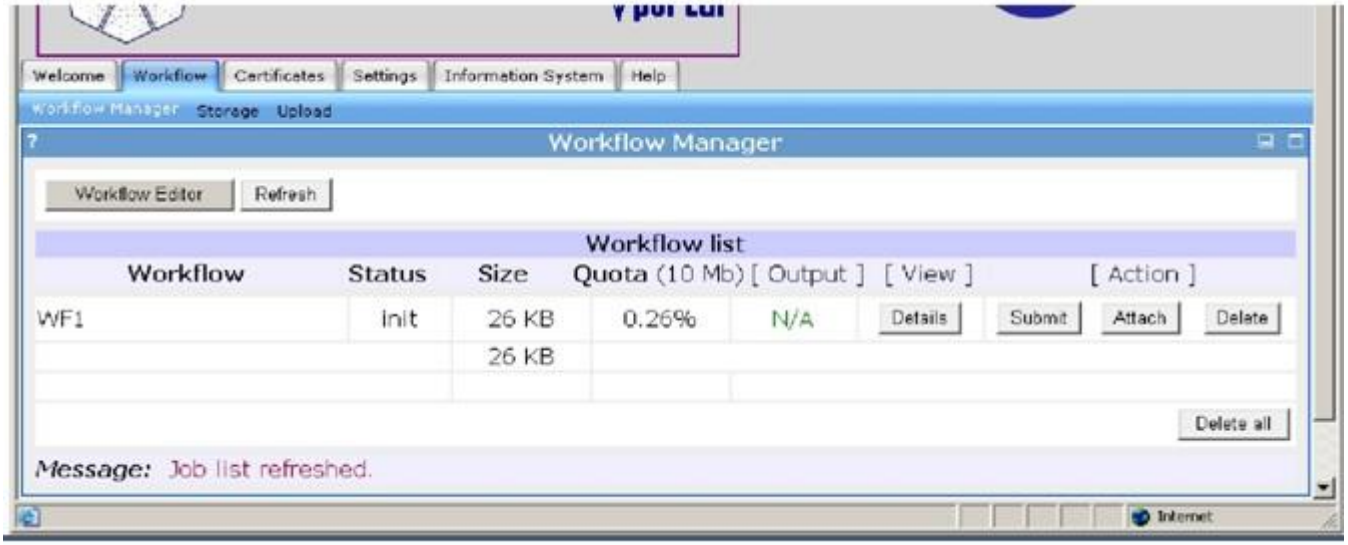


12. Browsing Proxies

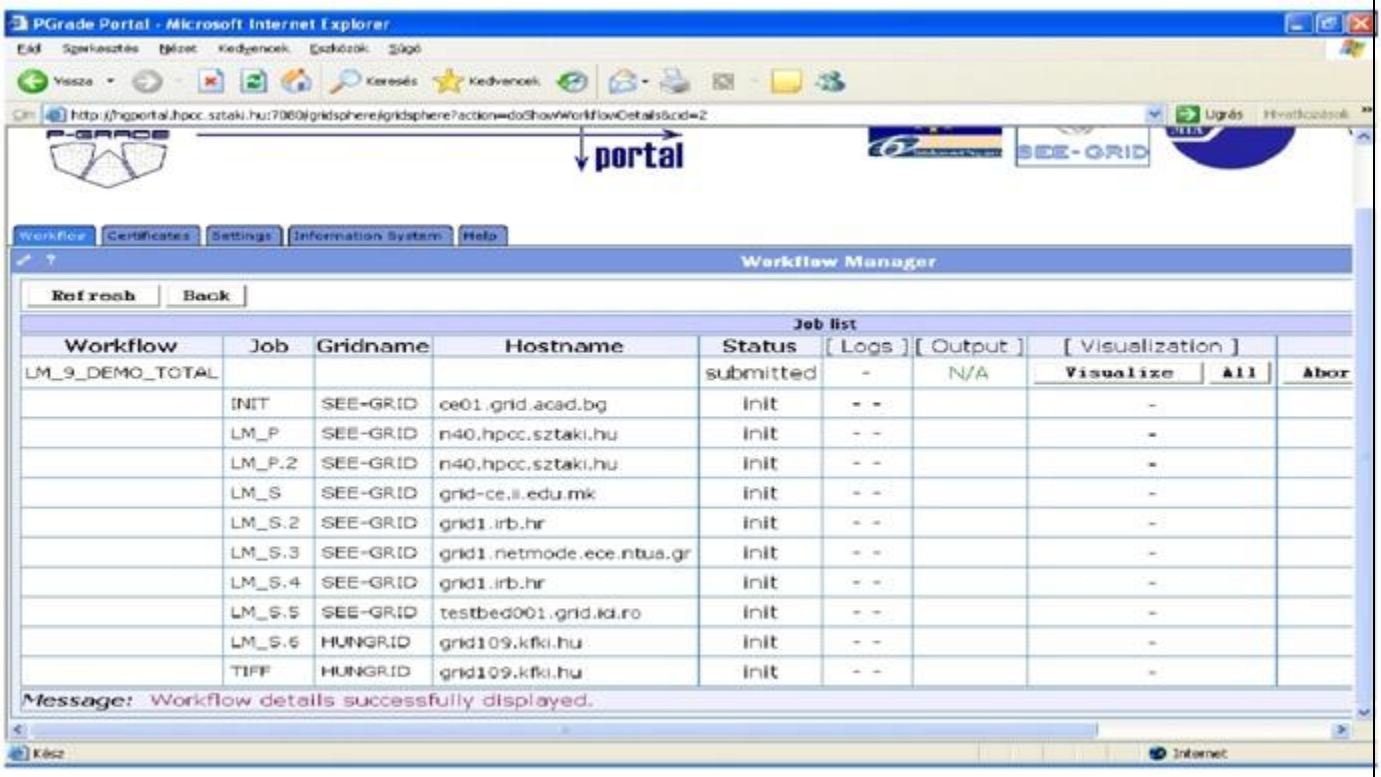


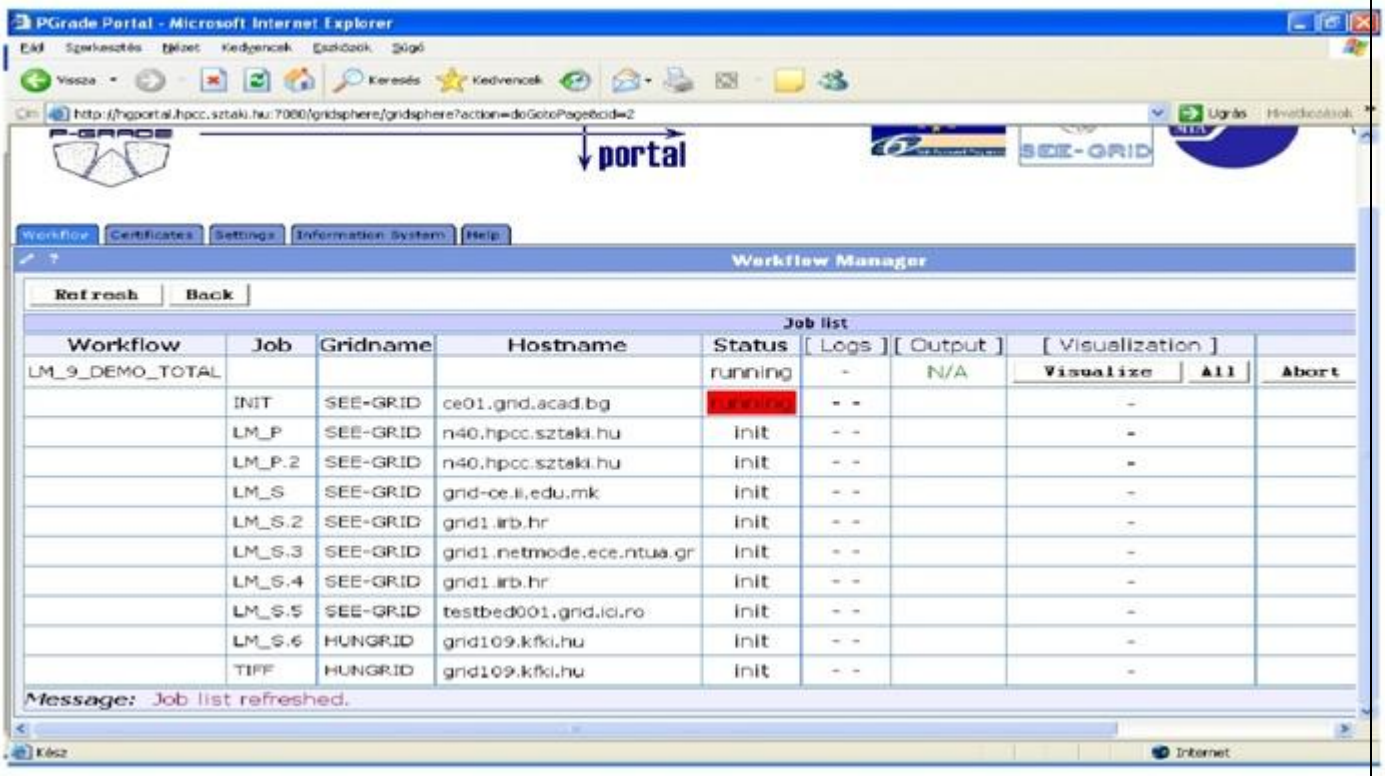
13. Workflow execution

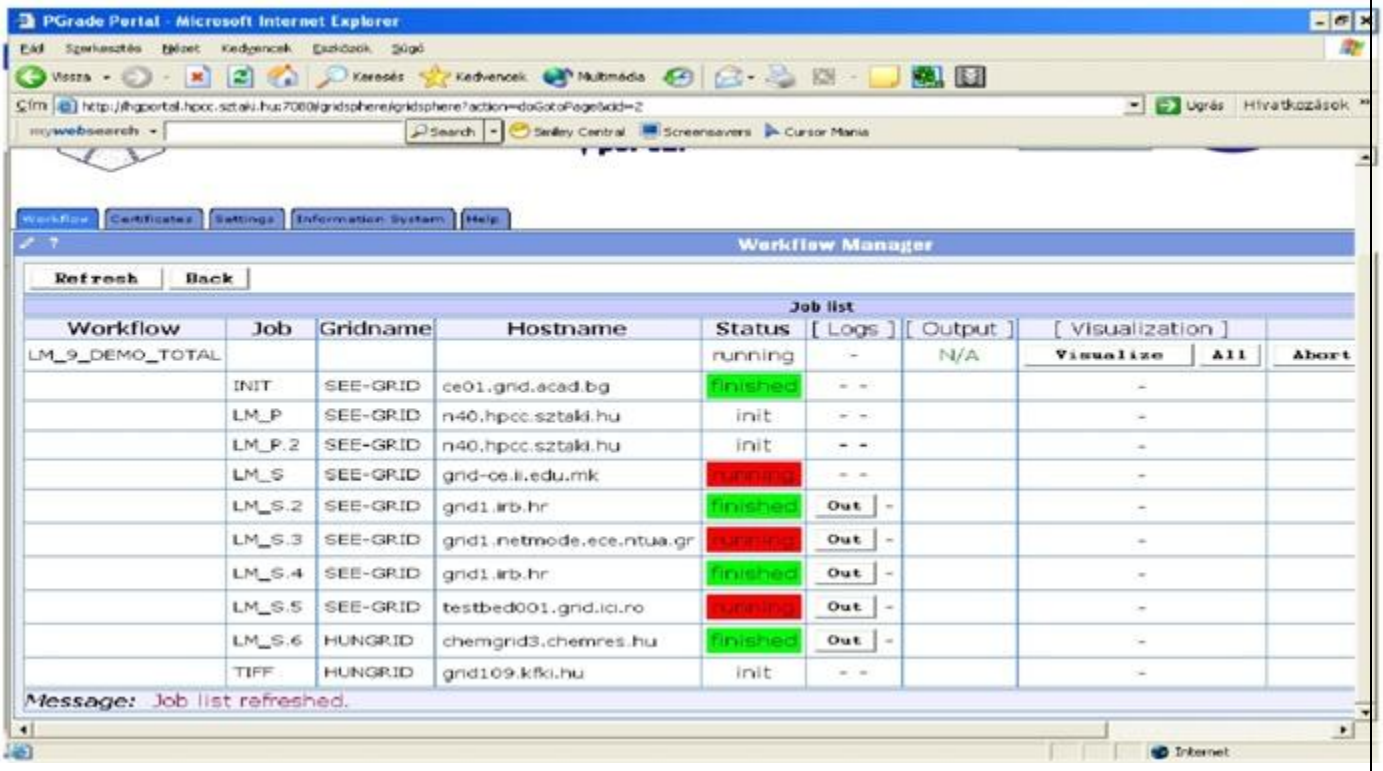
Workflow portlet

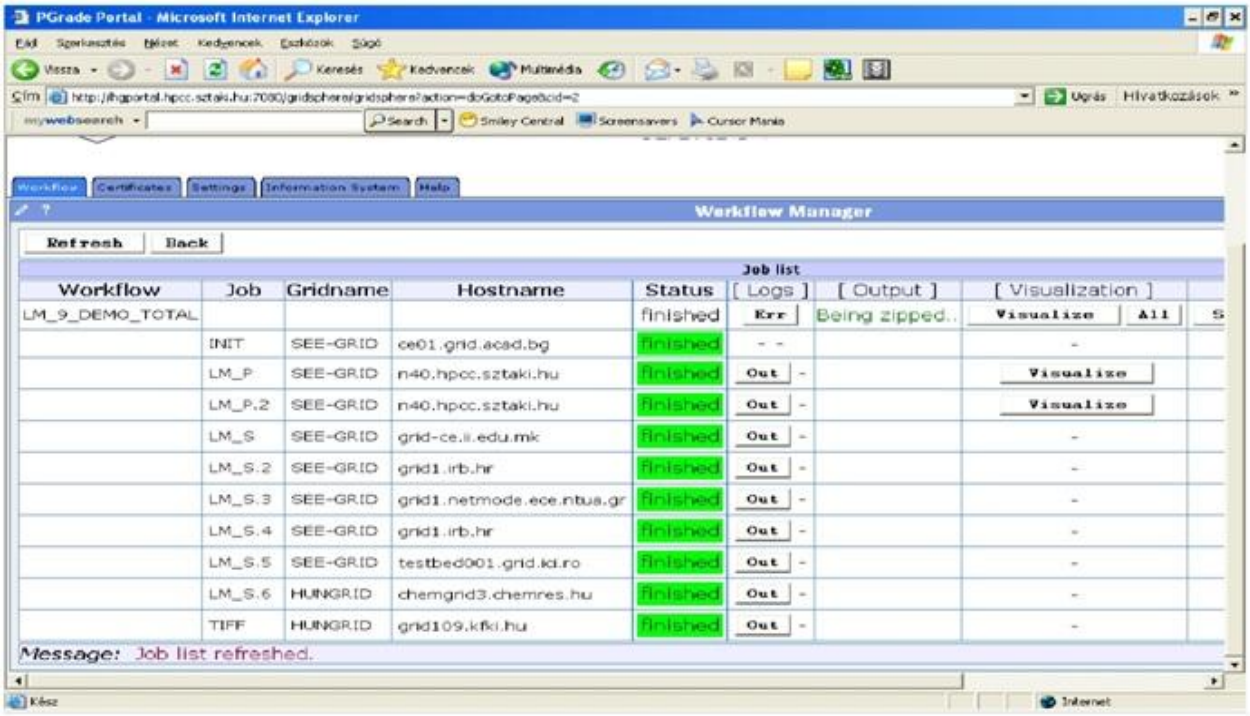


14. Observation by the workflow portlet




PGrade Portal - Microsoft Internet Explorer
portal
SEE-GRID
Workflow
Certificates
Settings
Information System
Help
Workflow Manager
Refresh
Back
Job list
Workflow | Job | Gridname | Hostname | Status | [Logs] | [Output] | [Visualization] |
LM_9_DEMO_TOTAL | | | | running | - | N/A | Visualize | All | Abort
INIT | SEE-GRID | ce01.grid.acad.bg | running | - -
LM_P | SEE-GRID | n40.hpcc.sztaki.hu | init | - -
LM_P.2 | SEE-GRID | n40.hpcc.sztaki.hu | init | - -
LM_S | SEE-GRID | grid-ce.ii.edu.mk | init | - -
LM_S.2 | SEE-GRID | grid1.irb.hr | init | - -
LM_S.3 | SEE-GRID | grid1.netmode.ece.ntua.gr | init | - -
LM_S.4 | SEE-GRID | grid1.irb.hr | init | - -
LM_S.5 | SEE-GRID | testbed001.grid.ici.ro | init | - -
LM_S.6 | HUNGRID | grid109.kfki.hu | init | - -
TIFF | HUNGRID | grid109.kfki.hu | init | - -
Message: Job list refreshed.


PGrade Portal - Microsoft Internet Explorer
Workflow
Certificates
Settings
Information System
Help
Workflow Manager
Refresh
Back
Job list
Workflow | Job | Gridname | Hostname | Status | [Logs] | [Output] | [Visualization] |
LM_9_DEMO_TOTAL | | | | running | - | N/A | Visualize | All | Abort
INIT | SEE-GRID | ce01.grid.acad.bg | finished | - -
LM_P | SEE-GRID | n40.hpcc.sztaki.hu | init | - -
LM_P.2 | SEE-GRID | n40.hpcc.sztaki.hu | init | - -
LM_S | SEE-GRID | grid-ce.ii.edu.mk | running | - -
LM_S.2 | SEE-GRID | grid1.irb.hr | finished | Out | -
LM_S.3 | SEE-GRID | grid1.netmode.ece.ntua.gr | running | Out | -
LM_S.4 | SEE-GRID | grid1.irb.hr | finished | Out | -
LM_S.5 | SEE-GRID | testbed001.grid.ici.ro | running | Out | -
LM_S.6 | HUNGRID | chemgrid3.chemres.hu | finished | Out | -
TIFF | HUNGRID | grid109.kfki.hu | init | - -
Message: Job list refreshed.



15. Downloading the results

RESULT

Thus the development of a Grid portal, where user can submit a job and get the result and to implement it with and without GRAM is executed successfully

CLOUD COMPUTING

Programs on SaaS

1 Create an word document of your class time table and store locally and on the cloud with doc,and pdf format . (use www.zoho.com and docs.google.com)

Steps:

With Google Docs, you can create and edit text documents right in your web browser—no special software is required. Even better, multiple people can work at the same time, you can see people's changes as they make them, and every change is saved automatically.

To start, you need a document to work with.

In this section, you learn how to:

- Create a new document
- Import and convert old documents to Docs

Create a new document

You can create a new document right in Docs or in Google Drive.

In Docs, click Create new document.

In Drive, click New > Google Docs > Blank document or From a template.

Import and convert old documents to Docs

If you have existing text documents, such as Microsoft® Word® or Adobe®PDF files, you can import and convert them to Docs.

- Go to Drive.
- Click New > File Upload and choose a text document from your computer. Supported files include .doc, .docx, .dot, .html, plain text (.txt), .odt, and .rtf.
- Right-click the file you want to convert and select Open with > Google Docs.

Converting your document from another program creates a copy of your original file in Docs format. You can then edit it in your browser like any other document.

Create Class timetable

Share documents

1. Open the file you want to share.
2. Click Share.
3. Enter the email addresses or Google Groups you want to share with.

Note: If you can't add people outside your company, see your G Suite administrator.

4. Choose what kind of access you want to grant people:
   - **Can edit**—Collaborators can add and edit content as well as add comments.
   - **Can comment**—Collaborators can add comments, but not edit content.
   - **Can view**—People can view the file, but not edit or add comments.

   Click Send.

Everyone you shared the document with receives an email with a link to the document.

2 Create a spread sheet which contains employee salary information and calculate gross and total sal using the formula

DA=10% OF BASIC

HRA=30% OF BASIC

PF=10% OF BASIC IF BASIC<=3000

12% OF BASIC IF BASIC>3000

TAX=10% OF BASIC IF BASIC<=1500

=11% OF BASIC IF BASIC>1500 AND BASIC<=2500

=12% OF BASIC IF BASIC>2500

(use www.zoho.com and docs.google.com)

NET_SALARY=BASIC_SALARY+DA+HRA-PF-TAX

If you're accustomed to creating your spreadsheets using an office suite or software like Microsoft Excel, you won't have any issue in creating a Google Spreadsheet. Google Spreadsheet works

the same as Excel, and you can do most of the important spreadsheet tasks with it. You can use Google Spreadsheet directly from your web browser or from its mobile app.

1. Sign into Google Sheets. Visit docs.google.com/spreadsheets and sign in with your Google or Gmail account. Your Gmail account gives you free access to Google Sheets.

2. View your existing sheets. Upon logging in, you will be brought to the main directory. If you already have existing spreadsheets, you can see and access them from here.

3. Create a new spreadsheet. Click the large red circle with a plus sign on the lower right corner. A new window or tab will be opened with the web-based spreadsheet.

4. Name the spreadsheet. "Untitled spreadsheet" appears on the top left corner. This is the current name of the spreadsheet. Click on it, and a small window will appear. Type in the name of the spreadsheet here, and click the "OK" button. You will see the name immediately change.

5. Work on the spreadsheet. You can work on Google Sheets much like how you would work on Microsoft Excel. There's a header menu and a toolbar with functions very similar to those of Microsoft Excel.
    1. calculate gross and total sal using the formula
    2. DA=10% OF BASIC
    3. HRA=30% OF BASIC
    4. PF=10% OF BASIC IF BASIC<=3000
    5. 12% OF BASIC IF BASIC>3000
    6. TAX=10% OF BASIC IF BASIC<=1500
    7. =11% OF BASIC IF BASIC>1500 AND BASIC<=2500
    8. =12% OF BASIC IF BASIC>2500
    9. (use www.zoho.com and docs.google.com)
    10. NET_SALARY=BASIC_SALARY+DA+HRA-PF-TAX

6. There's no need to save with Google Sheets as everything you do is automatically saved at regular intervals.

7. Exit the spreadsheet when you're finished. If you're done with your current document, you can just simply close the window or tab. Everything is saved automatically. You can access your document from Google Sheets or Google Drive.

3. Prepare a ppt on cloud computing –introduction , models, services ,and Architecture ppt should contain explanations, images and at least 20 pages (use www.zoho.com and docs.google.com)

Step 1: Login with your gmail id, at http://docs.google.com/

Step 2: Once you login, you will have a workspace area to work with your documents, spreadsheets and presentations. Just below the logo, you will find the “Create New” button, when you click on that, you will have all the available options. Select Presentation there.

Step 3: Now you have an empty presentation being created in your workspace.The look and feel doesn’t look like a web application at all!

Step 4: You can goto the Format Menu item and change the presentation Theme or Background. There are lots of templates and themes available!

Step 5: As you would do on any desktop office client, you can click on the new slide button, and select the Slide Design.

Step 6: If you need to insert a drawing, Shape or an Image, You can goto the Insert Menu item and select the same. Once you select image, You have options to select the image as a URL image or even a web upload!

Step 7: If you need to insert Tables to your slide deck, you can use the Table Menu item, and select the number of Rows and Colums.

Step 8: Following the above steps prepare presentation on cloud computing.

Step 9: You can now share the presentation through email attachment, give a link to the presentation.. so that one can watch it online in Google Docs and even Embed this presentation to your website too..!

4. Create your resume in a neat format using google and zoho cloud Programs on PaaS

Steps:

With Google Docs, you can create and edit text documents right in your web browser—no special software is required. Even better, multiple people can work at the same time, you can see people's changes as they make them, and every change is saved automatically.

To start, you need a document to work with.

In this section, you learn how to:

- Create a new document
- Import and convert old documents to Docs

Create a new document

You can create a new document right in Docs or in Google Drive.

In Docs, click Create new document.

In Drive, click New > Google Docs > Blank document or From a template.

Import and convert old documents to Docs

If you have existing text documents, such as Microsoft® Word® or Adobe®PDF files, you can import and convert them to Docs.

- Go to Drive.
- Click New > File Upload and choose a text document from your computer. Supported files include .doc, .docx, .dot, .html, plain text (.txt), .odt, and .rtf.
- Right-click the file you want to convert and select Open with > Google Docs.

Converting your document from another program creates a copy of your original file in Docs format. You can then edit it in your browser like any other document.

Create / design a neat resume

Share documents

1. Open the file you want to share.
2. Click Share.
3. Enter the email addresses or Google Groups you want to share with.

Note: If you can't add people outside your company, see your G Suite administrator.

4. Choose what kind of access you want to grant people:
   - **Can edit**—Collaborators can add and edit content as well as add comments.
   - **Can comment**—Collaborators can add comments, but not edit content.
   - **Can view**—People can view the file, but not edit or add comments.

   Click Send.

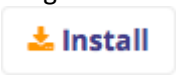
Everyone you shared the document with receives an email with a link to the document.

1 Write a Google app engine program to generate n even numbers and

deploy it to google cloud

Steps:

1. Install Eclipse IDE for Java EE Developers, version 4.6 or later:
2. If you have the Google Plugin for Eclipse installed, complete the migrating from GPE procedures.
3. Installing Cloud Tools for Eclipse
4. To install the plugin:

5. Drag the install button into your running Eclipse workspace:
   

6. Or from inside Eclipse, select Help > Eclipse Marketplace... and search for Google Cloud.
7. Restart Eclipse when prompted.

App engine Program to generate n even numbers using java servlet application

```
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml" lang="en">

 <head>

  <meta http-equiv="content-type" content="application/xhtml+xml; charset=UTF-8" />

  <title>Hello App Engine</title>

 </head>


 <body>

  <h1>Hello App Engine!</h1>

 <form action="/hello" method="get">

Enter 1st Number <input type="text" name="n1"><br>

<input type="Submit" value="Submit">

 </form>




 </body>

</html>
```

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

```
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet(
   name = "HelloAppEngine",
   urlPatterns = {"/hello"}
)
public class HelloAppEngine extends HttpServlet {

  @Override
  public void doGet(HttpServletRequest request, HttpServletResponse response)
      throws IOException {

   response.setContentType("text/plain");
   response.setCharacterEncoding("UTF-8");

   int a1= Integer.parseInt(request.getParameter("n1"));
   for (int i = 0; i < a1; ++i)
   {
          if(i%2==0)
                response.getWriter().print("\t"+i);
```

```
    }


 }
}
```

2 Google app engine program multiply two matrices

[Refer Program 1 Steps to perform matrix multiplication]

3 Google app engine program to validate user ; create a database login(username, password) in mysql and deploy to cloud

[Follow the program 1 steps to deploy fexible user validation sample program from git repository]

4 Write a Google app engine program to display nth largest no from the given list of numbers and deploy it into google cloud

[Refer Program 1 Steps to find nth largest no from the given list of numbers]

5. Google app engine program to validate the user Use mysql to store user info and deploy on the cloud