**CS6512 INTERNET PROGRAMMING LABORATORY L T P C 0 0 3 2**

**OBJECTIVES:**

**The student should be made to:**

 Be familiar with Web page design using HTML/XML and style sheets

 Be exposed to creation of user interfaces using Java frames and applets.

 Learn to create dynamic web pages using server side scripting.

 Learn to write Client Server applications.

 Be familiar with the frameworks JSP Strut, Hibernate, Spring

 Be exposed to creating applications with AJAX

**LIST OF EXPERIMNENTS:**

**IMPLEMENT THE FOLLOWING:**

**WEBPAGE CONCEPTS**

a) Create a web page with the following using HTML

a. To embed a map in a web page

b. To fix the hot spots in that map

c. Show all the related information when the hot spots are clicked.

b) Create a web page with the following.

a. Cascading style sheets.

b. Embedded style sheets.

c. Inline style sheets. Use our college information for the web pages.

c) Create and save an XML document at the server, which contains 10 users Information. Write a Program, which takes user Id as an input and returns the User details by taking the user information from the XML document.

**SOCKETS & SERVLETS**

a) Write programs in Java using sockets to implement the following:

i. HTTP request

ii. FTP

iii. SMTP

iv. POP3

b) Write a program in Java for creating simple chat application with datagram sockets and datagram packets.

c) Write programs in Java using Servlets:

i. To invoke servlets from HTML forms

ii. To invoke servlets from Applets

d) Write programs in Java to create three-tier applications using servlets for conducting on-line examination for displaying student mark list. Assume that student information is available in a database which has been stored in a database server.

e) Write a program to lock servlet itself to a particular server IP address and port number. It requires an init parameter key that is appropriate for its servlet IP address and port before it unlocks itself and handles a request

f) Session tracking using hidden form fields and Session tracking for a hit count

g) Install TOMCAT web server. Convert the static webpages of programs 1&2 into dynamic web pages using servlets (or JSP) and cookies. Hint: Users information (user id, password, credit card number) would be stored in web.xml. Each user should have a separate Shopping Cart.

**ADVANCE CONCEPTS:**

a) Implement a simple program using following frameworks

a. JSP Struts Framework b. Hibernate c. Spring

b) Explore the following application in AJAX: Searching in real time with live searches, Getting the answer with auto complete, Chatting with friends ,Dragging and dropping with Ajax, Getting instant login feedback, Ajax-enabled popup menus, Modifying Web pages on the fly.

c) Write a web services for finding what people think by asking 500 people’s opinion for any consumer product

d) Write a web services for predicting for any product sales

**TOTAL: 45 PERIODS**

**OUTCOMES:**

**At the end of the course, the student should be able to**

 Design Web pages using HTML/XML and style sheets

 Create user interfaces using Java frames and applets.

 Create dynamic web pages using server side scripting.

 Write Client Server applications.

 Use the frameworks JSP Strut, Hibernate, Spring

 Create applications with AJAX

**REFERENCE:**

spoken-tutorial.org.

**LIST OF EQUIPMENT FOR A BATCH OF 30 STUDENTS SOFTWARE:**

Java, Dream Weaver or Equivalent, MySQL or Equivalent, Apache Server

**HARDWARE:**

Standalone desktops 30 Nos

**WEBPAGE CONCEPTS**

a) Create a web page with the following using HTML

a. To embed a map in a web page

b. To fix the hot spots in that map

c. Show all the related information when the hot spots are clicked.

**Ex No: 1 IMAGE MAPPING**

**Date:**

**Aim:**

To Create a web page with the following using HTML

i) To embed an image map in a web page

ii) To fix the hot spots

iii) Show all the related information when the hot spots are clicked.

**Algorithm:**

1. Create a image in paint and identify few hotspot about cities in india
2. Use map tag and identify the coordinates position in hotspots
3. Create different html pages for each hotspot in main page
4. Embed image into the main page and provide hyper link in it.
5. Implement all element tags, headers, list, table in html pages
6. Use controls to inbuild hyperlink in program
7. Activate frames to view in split frame.
8. Identify all tags including title and other to display it in web page

**Program:**

**main.html**

<html>

<head>

<title>MAP</title>

</head>

<body align=CENTRE>

<map name=India>

<area shape=rect coords=423,481,430,473 href="Chennai.html"/>

<area shape=rect coords=353,239,358,232 href="Delhi.html"/>

<area shape=rect coords=474,326,481,318 href="Calcutta.html"/>

<area shape=rect coords=321,362,326,356 href="Bombay.html"/>

<h3>click city to view details</h3>

<p><img src=india123.bmp usemap=#India border=2 ><br>

<a href=index.html>Index</a>

</map>

</body>

</html>

**Delhi.html**

<html>

<title>city</title>

<body bgcolor=blue text=white>

<h1 align=centre><marquee>DELHI</marquee></h1>

<p align=centre > <b><i><font name=forte size=6>Delhi</font></i>

is the capital of India<br/></br>Many places are there to vist and it is one of the busy cities in India

</b></p>

<h3><font name=verdana>important places:</font></h3>

<table align=centre border=1 bgcolor=red cellspacing=5 cellpadding=5 frame=border>

<tr>

<th valign=top>places</th>

<th valign=top>temp</th>

</tr>

<tr>

<td>red fort</td>

<td>40</td>

</tr>

<tr>

<td>qutub minar</td>

<td>50</td>

</tr>

</table>

<ul>

<li>Rashtrapati Bhavan</li>

<li>India gate</li>

<li>ashok pillar</li>

</ul><h2> <a href="main.html"> <u>back</u></a>

</h2>

</body>

</html>

**Calcutta.html**

<html>

<title>city</title>

<body bgcolor=green text=yellow>

<h1 align=center><marquee>KOLKATA</marquee></h1>

<p><b><i>

<font name=forte size=6>Kolkata </font></i>is the capital of west Bengal</font>

<h3><font name=verdana>important places</font></h3>

<ol>

<li>Kali Temple</li>

<li>Howrah Bridge</li>

<li>Teesta River</li>

<h2><a href="main.html"><u>back</u></a>

</h2>

</body>

</html>

**Bombay.html**

<html>

<title>Bombay</title>

<body bgcolor=red text=blue>

<h1 align=centre><marquee>MUMBAI</marquee></h1>

<p align=centre><b><i>

<font name=forte size=6>Mumbai </font></i>

is the capital of Maharastra<br/><br/>Many places are there to vist and it is one of the busy cities in India

</b></p>

<ol>

<li>Gateway of India</li>

<li>Elephant Caves</li>

<li>Ajanta and Ellora Caves</li>

<h2><a href="main.html"><u>back</u></a>

</h2>

</body>

</html>

**Chennai.html**

<html>

<title>city</title>

<body bgcolor=yellow text=blue>

<h1 align=centre><marquee>CHENNAI</marquee></h1>

<p align=centre<b><i>

<font name=forte size=6>Chennai</font></i>

is the capital of tamil nadu.Many places are there to vist and it is

one of the important cities in India

</b></p>

<h3><font name=verdana>important places:</font></h3>

<table align=centre border=1 bgcolor=red cellspacing=5 cellpadding=5 frmae=border>

<tr>

<th valign=top>places</th>

<th valign=top>temp</th>

</tr>

<tr>

<td>adyar</td>

<td>30</td>

</tr>

<tr>

<td>anna nagar</td>

<td>30</td>

</tr>

</table>

<ul>

<li>marina</li>

<li>mahalipuram</li>

<li>temple</li>

</ul>

<h2><a href="main.html"><u>back</u></a>

</h2>

</body>

</html>

**Index.html**

<html><head>

<frameset rows=20%,\*>

<frame src=title.html>

<frameset cols=20%,\*>

<frame src=menu.html>

<frame src=main.html name=a>

</frameset>

</frameset></head>

</html>

**Title.html**

<html>

<h1>India Map</h1>

</html>

**Menu.html**

<html>

<body>

<a href=Chennai.html target=a>Chennai</a>

<a href=Delhi.html target=a>Delhi</a>

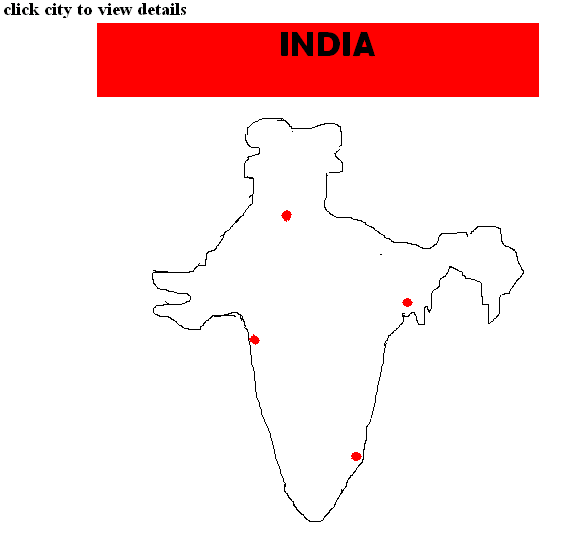
<a href=Calcutta.html target=a>Calcutta</a>

<a href=Bombay.html target=a>Bombay</a>

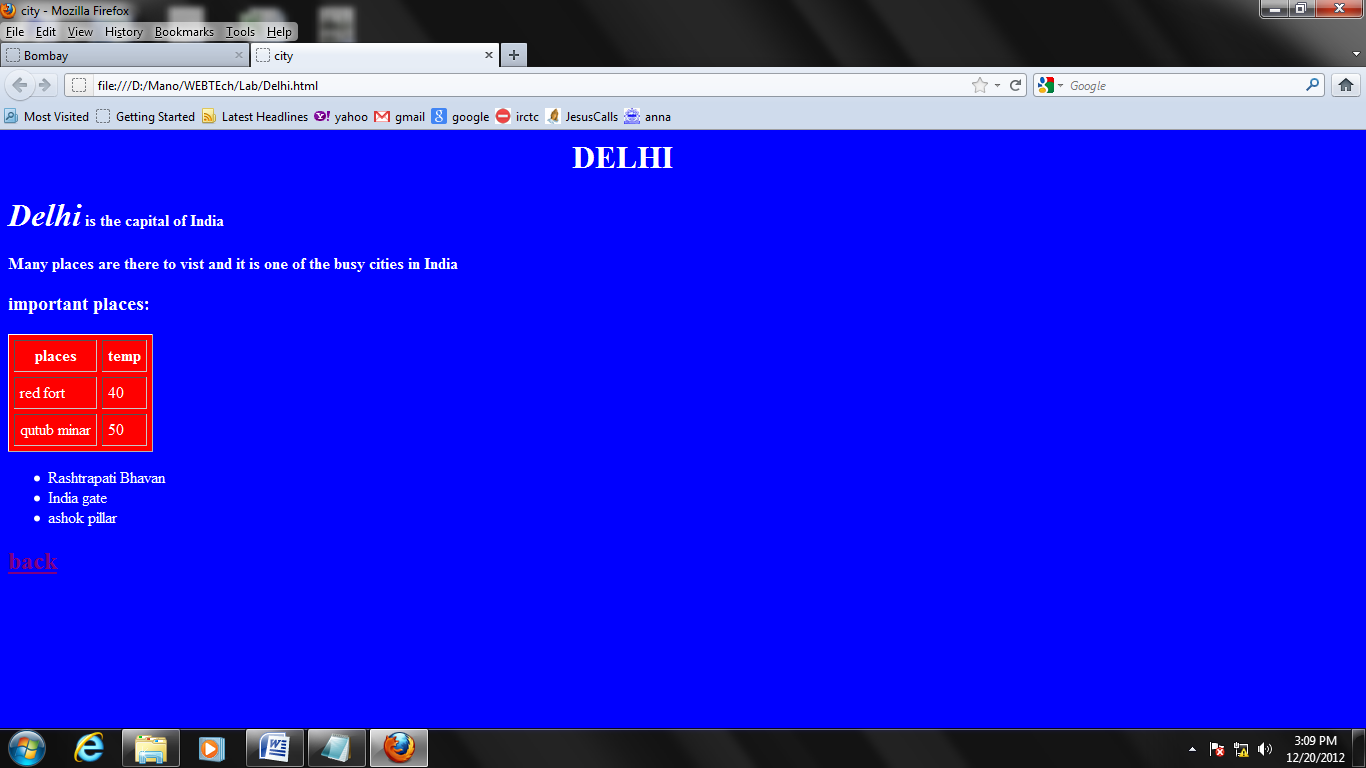
</body>

</html>

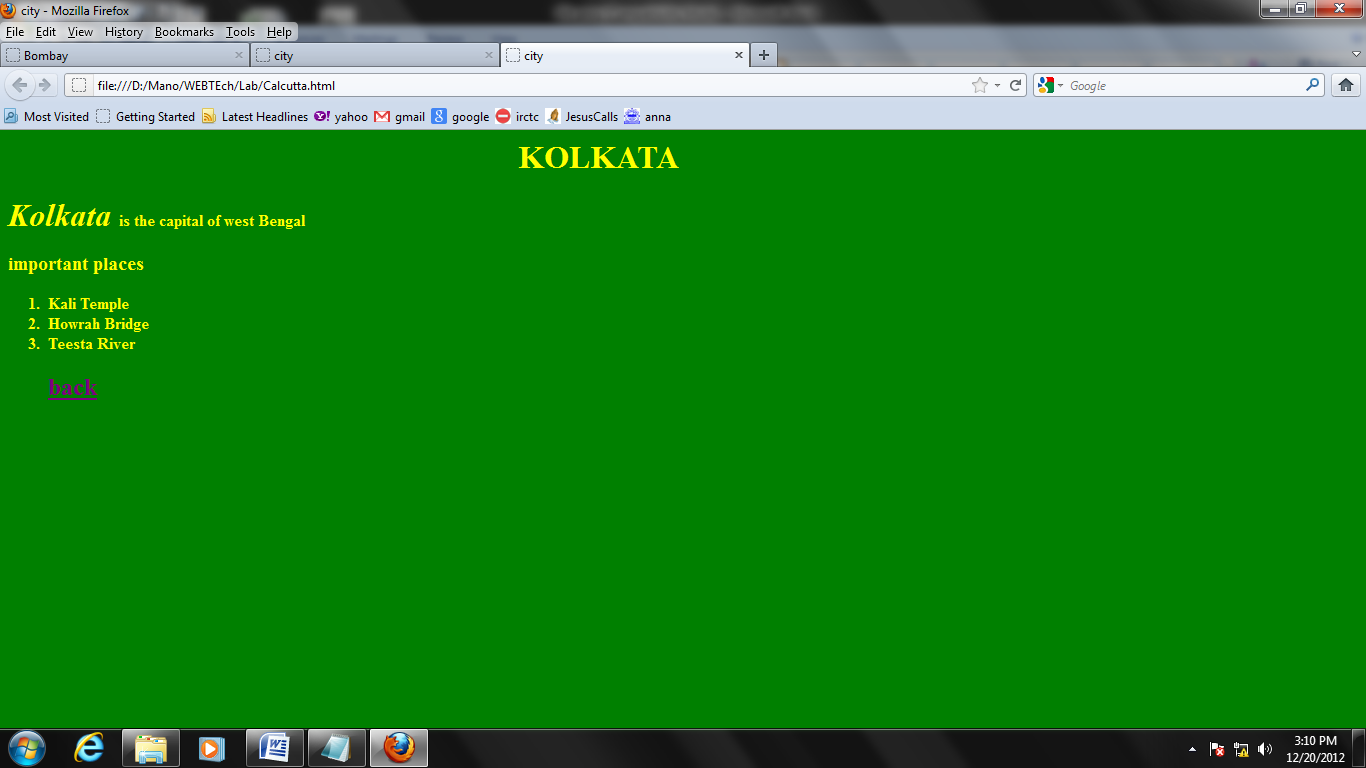
Main.html



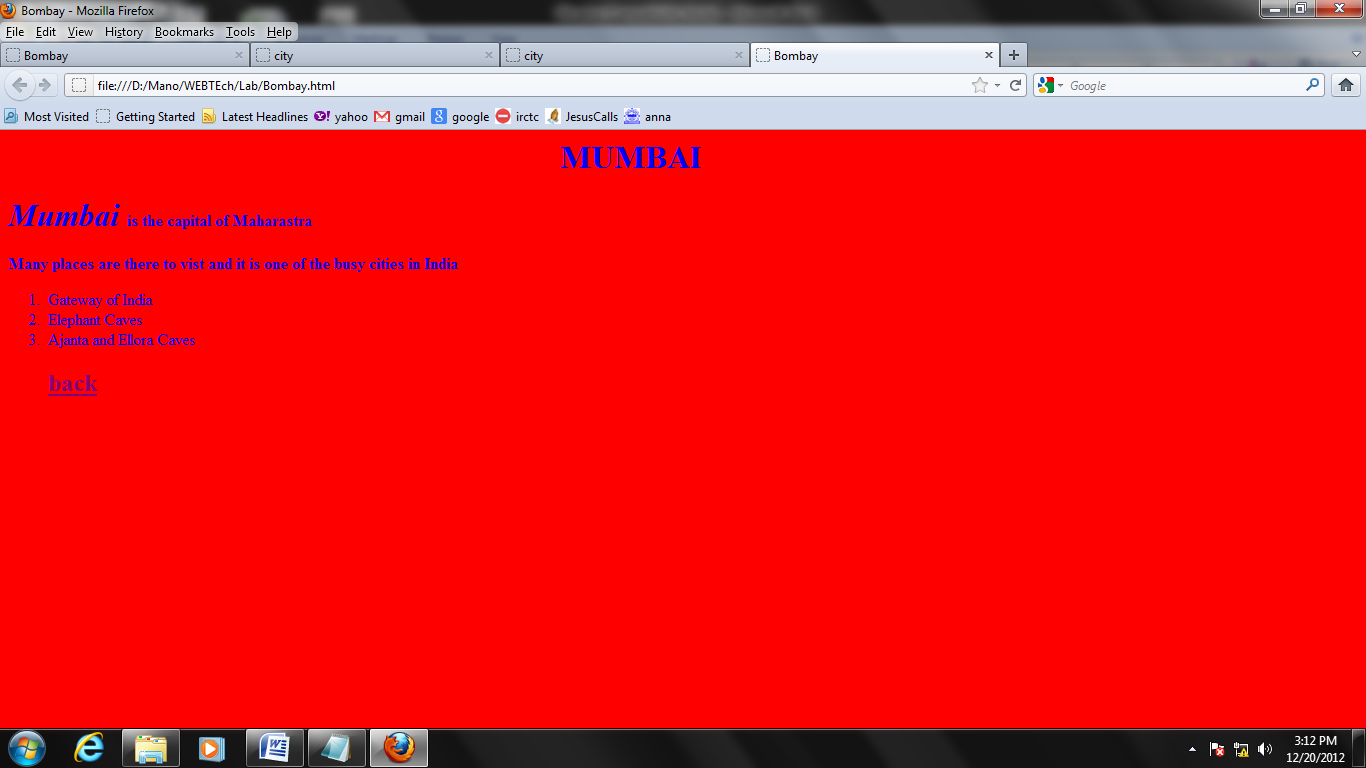
DELHI.HTML



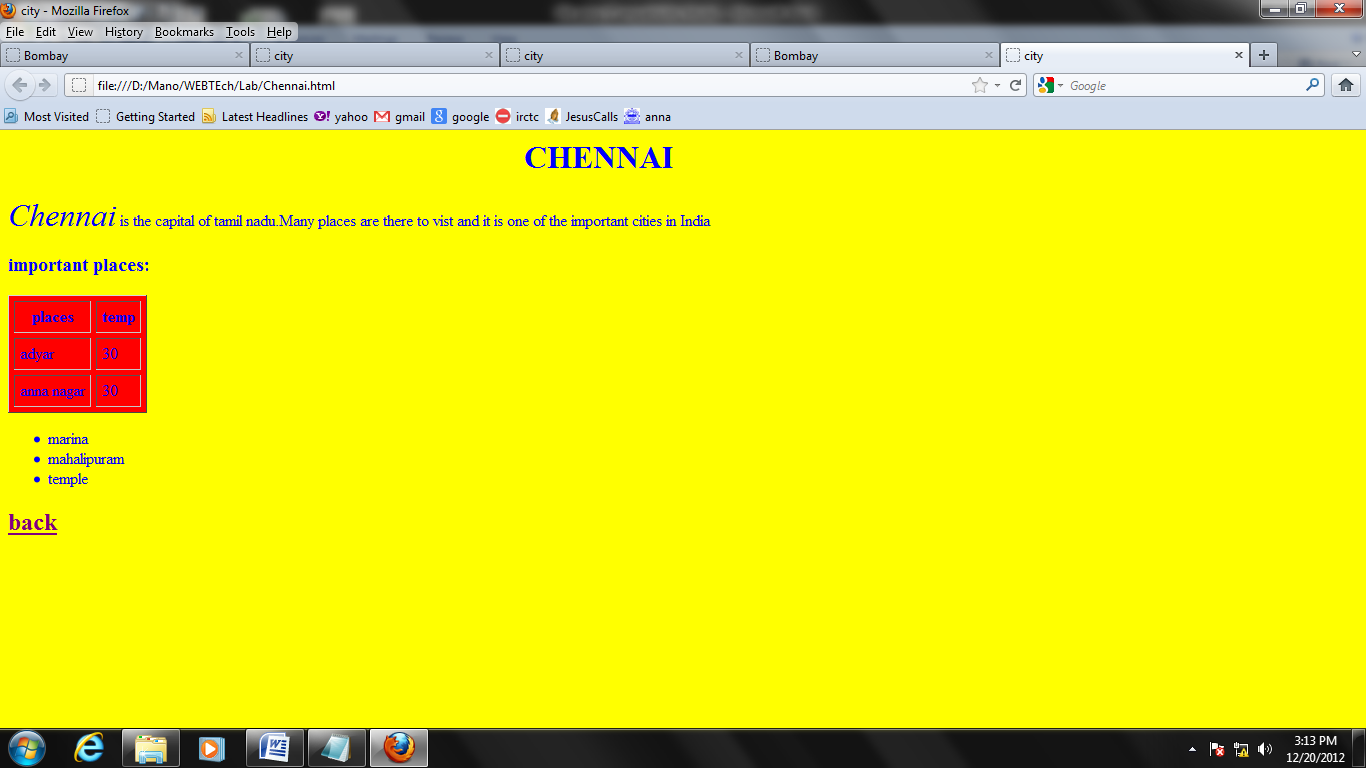
CALCUTTA.HTML



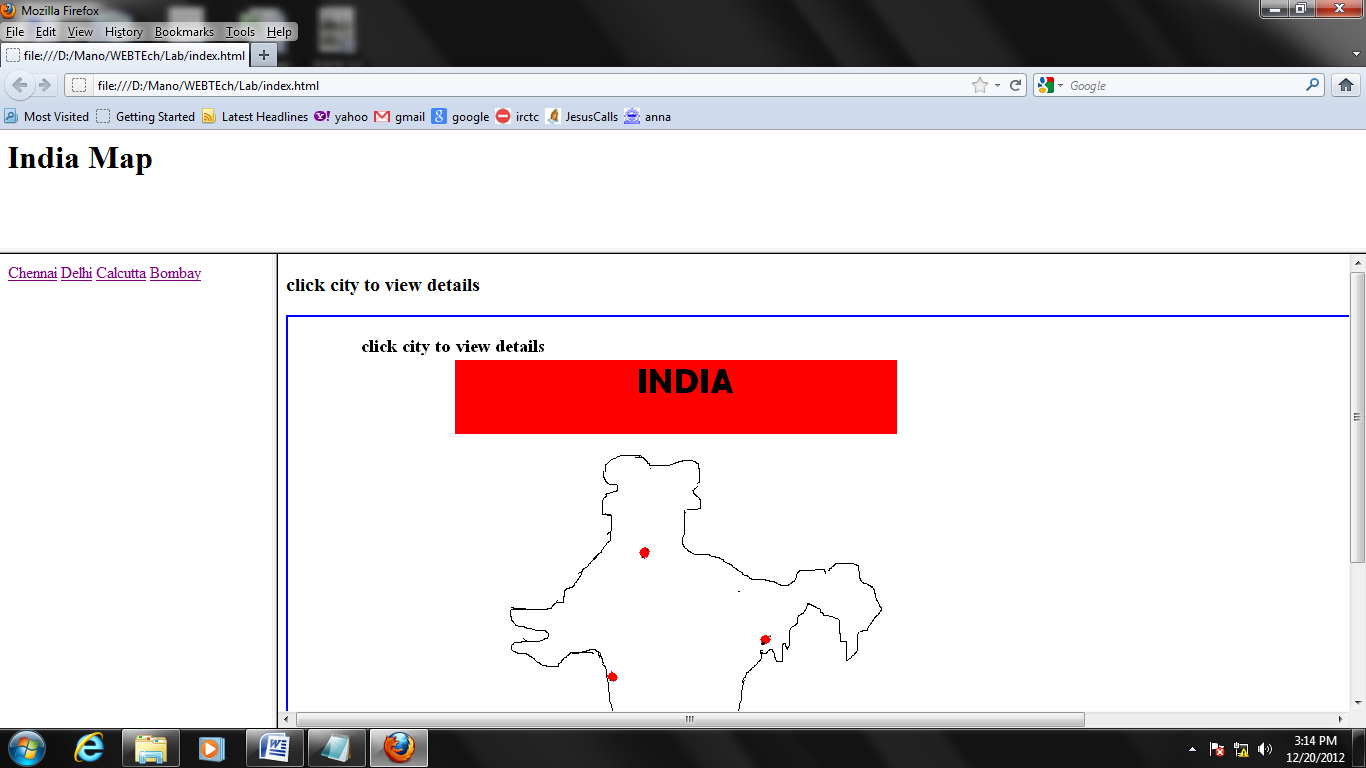
BOMBAY.HTML:



CHENNAI.HTML



INDEX.HTML



**RESULT:**

Thus the program for creating web page and identifying hot spot in an embedded image has done and executed successfully.

b) Create a web page with the following.

a. Cascading style sheets.

b. Embedded style sheets.

c. Inline style sheets. Use our college information for the web pages.

**Ex No:2 CASCADING STYLE SHEETS**

**Date:**

**Aim :**

To Create a web page with all types of Cascading style sheets.

**Algorithm:**

1. Create a web page using html file
2. Identify the category of style sheet either internal, external or inline style sheet
3. In embed, style tag should be done in head section in CSS
4. In inline, style tag should be identified in body section
5. In link, style tag should be given in a separate css file and it can be referred as link with its specification @import
6. Create a web page about college and see its different style differences
7. Execute the web page.

**Program:**

**Embed\_cascade.html or internal:**

<html>

<head>

<title>Welcome</title>

<style>

p{color:green;margin-left:110px;margin-right:110px;text-indent:.7cm;text-align:justify;font-size:19; left:20px}

</style>

</head>

<body bgcolor="white" text="red">

<h1> <center> Welcome</center></h1>

<p>Loyola Institute of Technology &Science is an Engineering college located in Nagercoil. There are many varied courses taught here.<br/>The campus is located about 3 km from the city and has its own bus transport system.

</p>

<h2>Thank You...</h2>

</body>

</html>

**inline\_cascade.html:**

<html>

<head>

<title>IN-LINE STYLE SHEET</title>

</head>

<body style="font-family:sherif; color:magenta; font-style:italic; font-weight:bold; font-size:30; text-align:center; background-color:green;" >

<h1>Loyola </h1>

<h2>Welcome....</h2>

<br>

<br>

</body>

</html>

**link\_cascade.html or external:**

<html>

<head>

<title>Link Cascading</title>

<style type="text/css"> @import URL(style.css);</style>

(OR)

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<body class="style">

<h2><center> welcome</center></h2>

<p class="style1"> Loyola is one of the finest engineering colleges in Nagercoil. </p>

<h3> It was in Thovalai, Kanyakumari District</h3>

</body>

</html>

**style.css:**

h2{font-family:sherif;color:white;font-style:italic;font-weight:bold;font-size:50;}

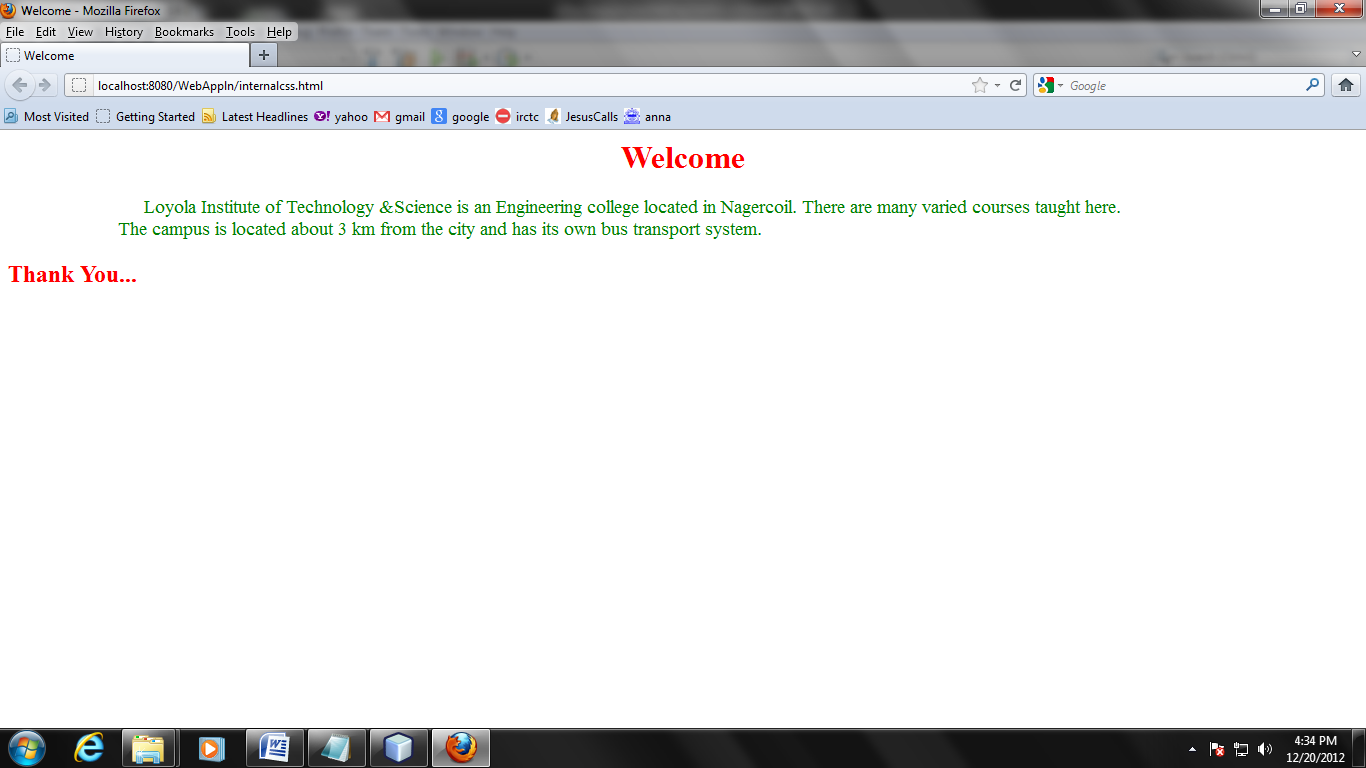
body { background-color:blue; }

p {color:Green;font-size:50;}

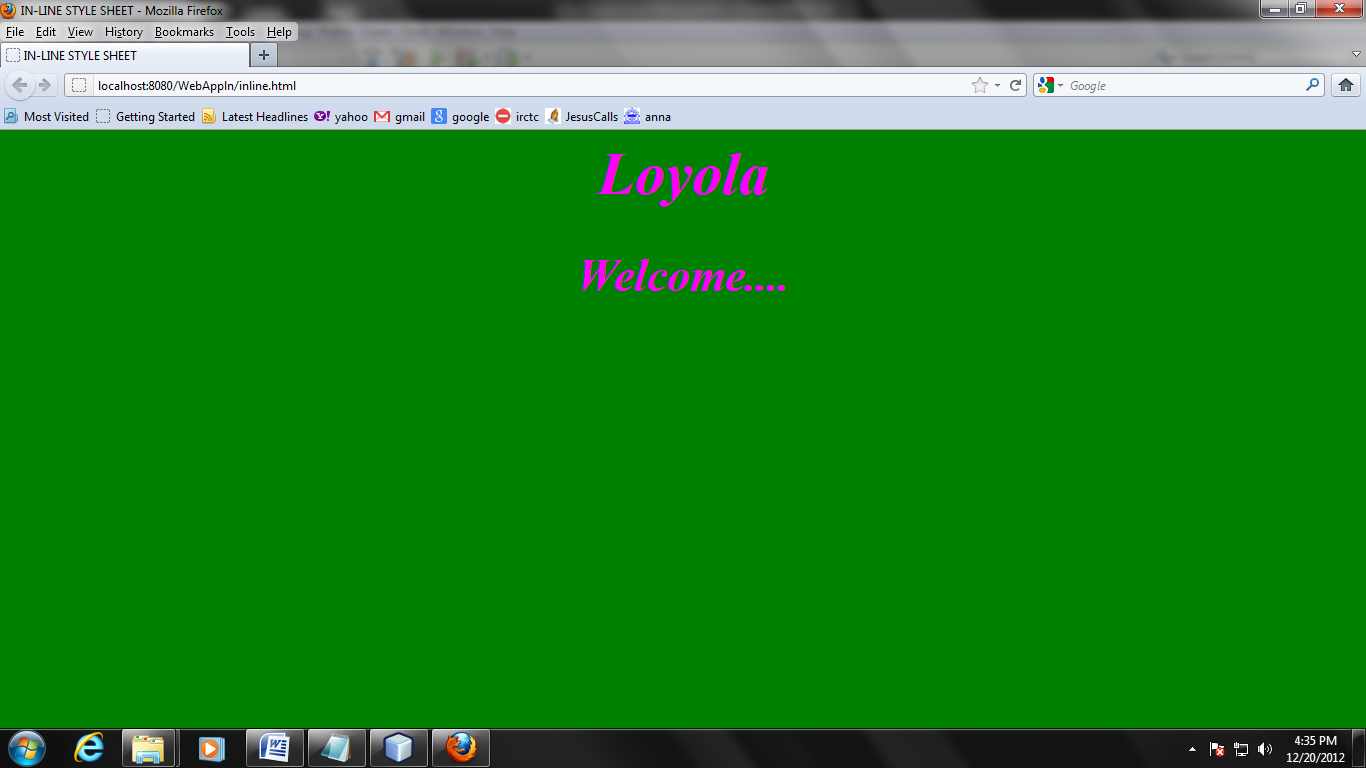
h3{color:red; }

**OUTPUT:**

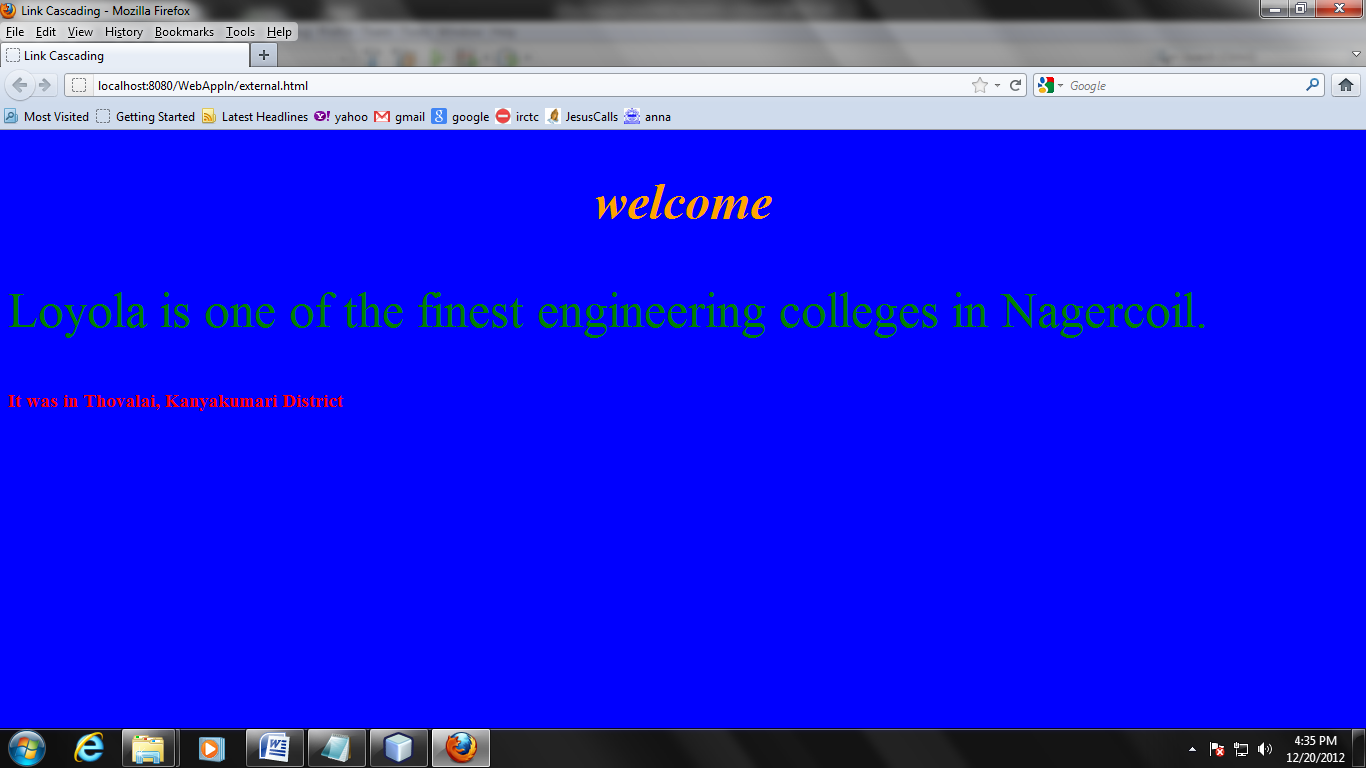
**EMBED\_CASCADE.HTML:**



**INLINE\_CASCADE.HTML:**



**LINK\_CASCADE.HTML:**



**RESULT:**

Thus the program for creating web pages by implementing CSS was done and executed successfully.

c) Create and save an XML document at the server, which contains 10 users Information. Write a Program, which takes user Id as an input and returns the User details by taking the user information from the XML document.

HTMLPgm.html

<html>

<head>

<title>authorization page</title>

<script language="JavaScript" Run At=server>

var xmlDoc=new ActiveXObject("Microsoft.XMLDOM");

xmlDoc.load("XMLpgm.xml");

function search()

{

var id=document.myform.id.value;

var pword=document.myform.pword.value;

var qry=xmlDoc.selectNodes("document/users/user[userid='"+id+"']");

if(qry.length!=0)

{

if(pword!=qry.item(0).childNodes.item(1).text)

//document.body.innerHTML="Password incorrect";

document.write("success");

}

else

document.body.innerHTML="No matching data found. userid incorrect";

}

</script>

</head>

<body>

<form name="myform" method=post action="HTMLPgm.html">

User id:<input type=text name=id><br>

password :<input type=password name=pword><br>

<input type=submit value=Go onClick="search()">

</form>

</body>

</html>

XMLpgm.xml

<?xml version="1.0" ?>

<document>

<users>

<user>

<userid>abc</userid>

<password>abc</password>

<first\_name>raj</first\_name>

<last\_name>kumar</last\_name>

<dob>10/1/1982</dob>

</user>

<user>

<userid>ravi</userid>

<password>ravi</password>

<first\_name>ravi</first\_name>

<last\_name>kumar</last\_name>

<dob>11/11/1982</dob>

</user>

<user>

<userid>sri</userid>

<password>sri</password>

<first\_name>sri</first\_name>

<last\_name>ram</last\_name>

<dob>12/12/1983</dob>

</user>

<user>

<userid>lax</userid>

<password>lax</password>

<first\_name>laxman</first\_name>

<last\_name>kumar</last\_name>

<dob>1/1/1980</dob>

</user>

<user>

<userid>chandu</userid>

<password>chandu</password>

<first\_name>chandana</first\_name>

<last\_name>priya</last\_name>

<dob>9/11/1998</dob>

</user>

<user>

<userid>vysu</userid>

<password>vysu</password>

<first\_name>vyshnavi</first\_name>

<last\_name>matha</last\_name>

<dob>17/6/1982</dob>

</user>

<user>

<userid>prem</userid>

<password>prem</password>

<first\_name>prem</first\_name>

<last\_name>kumar</last\_name>

<dob>11/11/1988</dob>

</user>

<user>

<userid>geeta</userid>

<password>geeta</password>

<first\_name>geeta</first\_name>

<last\_name>anjali</last\_name>

<dob>13/1/1972</dob>

</user>

<user>

<userid>preeti</userid>

<password>preeti</password>

<first\_name>prethi</first\_name>

<last\_name>priya</last\_name>

<dob>7/11/1968</dob>

</user>

<user>

<userid>praveen</userid>

<password>praveen</password>

<first\_name>kumar</first\_name>

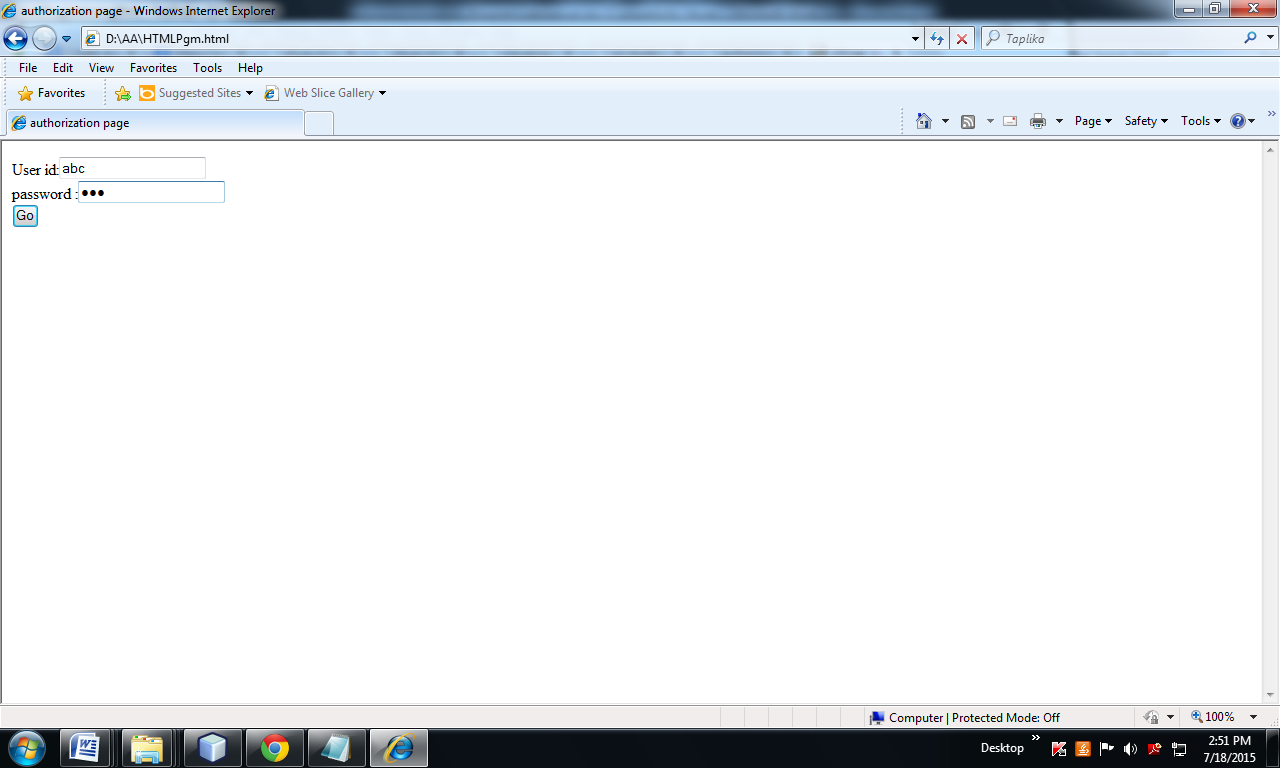
<last\_name>kumar</last\_name>

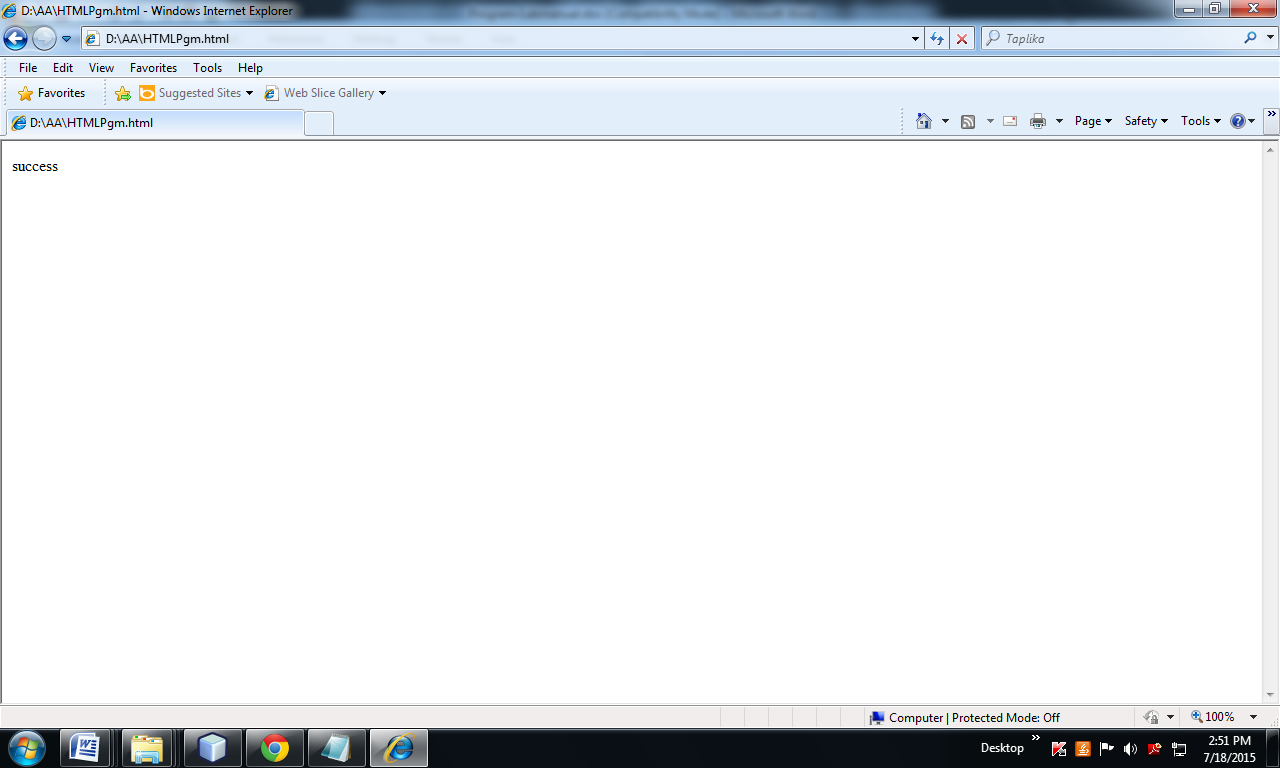
<dob>11/11/1999</dob>

</user>

</users>

</document>





**SOCKETS & SERVLETS**

a) Write programs in Java using sockets to implement the following:

i. HTTP request

ii. FTP

iii. SMTP

iv. POP3

**HTTPServer:**

import java.net.\*;

import java.io.\*;

import java.awt.image.\*;

import javax.imageio.\*;

import javax.swing.\*;

class HTTPServer

{

public static void main(String args[]) throws Exception

{

ServerSocket server=null;

Socket socket;

server=new ServerSocket(4000);

System.out.println("Server Waiting for image");

socket=server.accept();

System.out.println("Client connected.");

InputStream in = socket.getInputStream();

DataInputStream dis = new DataInputStream(in);

int len = dis.readInt();

System.out.println("Image Size: " + len/1024 + "KB");

byte[] data = new byte[len];

dis.readFully(data);

dis.close();

in.close();

InputStream ian = new ByteArrayInputStream(data);

BufferedImage bImage = ImageIO.read(ian);

JFrame f = new JFrame("Server");

ImageIcon icon = new ImageIcon(bImage);

JLabel l = new JLabel();

l.setIcon(icon);

f.add(l);

f.pack();

f.setVisible(true);

}

}

**HTTPClient:**

import javax.swing.\*;

import java.net.\*;

import java.awt.image.\*;

import javax.imageio.\*;

import java.io.\*;

import java.awt.image.BufferedImage;

import java.io.ByteArrayOutputStream;

import java.io.File;

import java.io.IOException;

import javax.imageio.ImageIO;

public class HTTPClient

{

public static void main(String args[]) throws Exception

{

Socket soc;

BufferedImage img = null;

soc=new Socket("localhost",4000);

System.out.println("Client is running. ");

try

{

System.out.println("Reading image from disk. ");

img = ImageIO.read(new File("D://AA/Desert.jpg"));

ByteArrayOutputStream baos = new ByteArrayOutputStream();

ImageIO.write(img,"jpg",baos);

baos.flush();

byte[] bytes = baos.toByteArray();

baos.close();

System.out.println("Sending image to server. ");

OutputStream out = soc.getOutputStream();

DataOutputStream dos = new DataOutputStream(out);

dos.writeInt(bytes.length);

dos.write(bytes, 0, bytes.length);

System.out.println("Image sent to server. ");

dos.close();

out.close();

}

catch (Exception e)

{

System.out.println("Exception: " + e.getMessage());

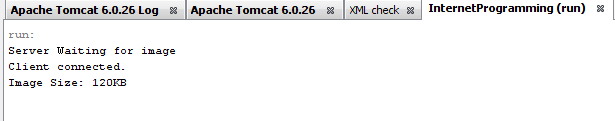
soc.close();

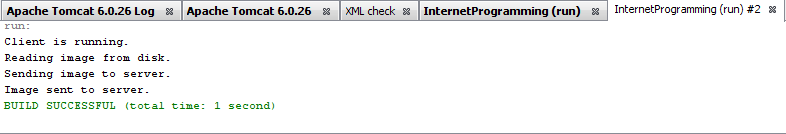
}

soc.close();

}

}







**FTPServer:**

import java.io.\*;

import java.net.\*;

public class FTPServer

{

public static void main (String [] args ) throws IOException

{

try

{

ServerSocket serverSocket = new ServerSocket(4000);

Socket socket = serverSocket.accept();

System.out.println("Accepted connection : " + socket);

File transferFile = new File ("D://AA/send.txt");

byte [] bytearray = new byte [(int)transferFile.length()];

FileInputStream fin = new FileInputStream(transferFile);

BufferedInputStream bin = new BufferedInputStream(fin);

bin.read(bytearray,0,bytearray.length);

OutputStream os = socket.getOutputStream();

System.out.println("Sending Files...");

os.write(bytearray,0,bytearray.length);

os.flush();

socket.close();

System.out.println("File transfer complete");

}

catch(Exception e)

{

}

}

}

**FTPClient:**

import java.io.\*;

import java.net.\*;

public class FTPClient

{

public static void main (String [] args ) throws IOException

{

try

{

int filesize=2022386;

int bytesRead;

int currentTot = 0;

Socket socket = new Socket(InetAddress.getLocalHost(),4000);

byte [] bytearray = new byte [filesize];

InputStream is = socket.getInputStream();

FileOutputStream fos = new FileOutputStream("D://AA/received.txt");

BufferedOutputStream bos = new BufferedOutputStream(fos);

bytesRead = is.read(bytearray,0,bytearray.length);

currentTot = bytesRead;

do

{

bytesRead = is.read(bytearray, currentTot, (bytearray.length-currentTot));

if(bytesRead >= 0)

currentTot += bytesRead;

} while(bytesRead > -1);

fos.write(bytearray, 0 , currentTot);

bos.flush();

bos.close();

socket.close();

}

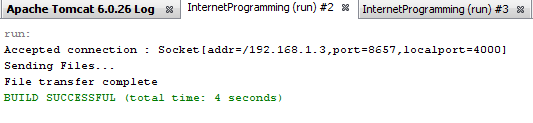
catch(Exception e)

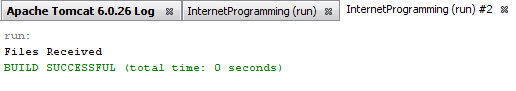
{

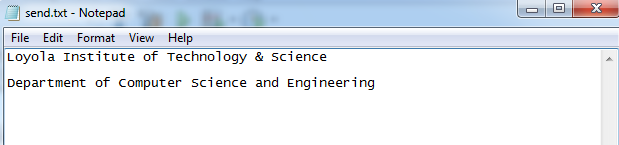
}

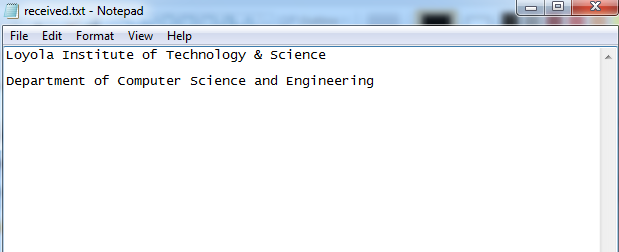
}

}









SMTPServer:

SMTPClient:

POP3Server:

POP3Client:

b) Write a program in Java for creating simple chat application with datagram sockets and datagram packets.

**UDPServer:**

import java.io.\*;

import java.net.\*;

class UDPServer

{

public static void main(String args[])

{

try

{

DatagramSocket Serversock=new DatagramSocket(5678);

BufferedReader in1=new BufferedReader(new InputStreamReader(System.in));

String st1;

while(true)

{

byte[] send\_data1=new byte[1024];

byte[] recv\_data1=new byte[1024];

DatagramPacket recv\_pack1=new DatagramPacket(recv\_data1,recv\_data1.length);

Serversock.receive(recv\_pack1);

String str1=null;

str1=new String(recv\_pack1.getData());

System.out.println("Received from client...");

System.out.println(str1.trim());

System.out.println("Enter the string...");

st1=in1.readLine();

if(st1.equals("quit"))

{

Serversock.close();

}

send\_data1=st1.getBytes();

InetAddress ad1=recv\_pack1.getAddress();

int p1=recv\_pack1.getPort();

DatagramPacket send\_pack1=new DatagramPacket(send\_data1,send\_data1.length,ad1,p1);

Serversock.send(send\_pack1);

}

}

catch(Exception e)

{

System.out.println(e);

}

}

}

**UDPClient:**

import java.io.\*;

import java.net.\*;

class UDPClient

{

public static void main(String args[])

{

try

{

DatagramSocket Clientsock=new DatagramSocket();

BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

String str1;

while(true)

{

byte[] send\_data=new byte[1024];

byte[] recv\_data=new byte[1024];

System.out.println("Enter the string...");

str1=in.readLine();

if(str1.equals("quit"))

{

Clientsock.close();

}

send\_data=str1.getBytes();

InetAddress addr=InetAddress.getLocalHost();

DatagramPacket send\_pack=new DatagramPacket(send\_data,send\_data.length,addr,5678);

Clientsock.send(send\_pack);

DatagramPacket recv\_pack=new DatagramPacket(recv\_data,recv\_data.length);

Clientsock.receive(recv\_pack);

String s1=new String(recv\_pack.getData());

System.out.println("The string received back from server... "+s1.trim());

}

}

catch(Exception e)

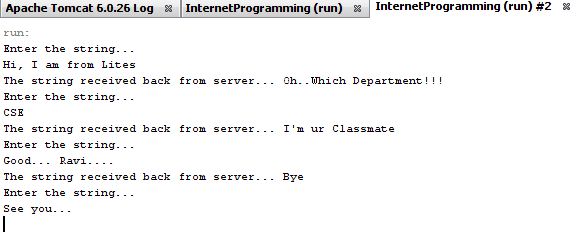
{

System.out.println(e);

}

}

}





c) Write programs in Java using Servlets:

i. To invoke servlets from HTML forms

ii. To invoke servlets from Applets

**Ex No: 4 Servlet Code with HTML**

**Date :**

**Aim:** To execute a servlet java program from HTML Design

**Algorithm:**

1. Create a java program from GenericServlet architecture
2. Use getParameterNames method collect all its method and values
3. Display the values
4. In HTMl program create controls and access it through Servlet file
5. Display the design and get the result

**Program:**

Servlet Code:

import java.io.IOException;

import java.io.PrintWriter;

import java.util.Enumeration;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class ServHtml extends HttpServlet

{

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException

{

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

try {

Enumeration e = request.getParameterNames();

out.print("<html><body><h1>");

out.print("<br/>LITES welcomes you All<br/></h1>");

while(e.hasMoreElements())

{

String pname = (String)e.nextElement();

out.print("<h3>");

out.print(pname + " = ");

String pvalue = request.getParameter(pname);

out.println(pvalue);

out.println("<br/>");

out.print("</html>");

}

} finally

{

out.close();

}

}

}

HTML CODE:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<HTML>

<head>

<TITLE>INVOKING SERVLET FROM HTML</TITLE>

</head>

<BODY>

<CENTER>

<FORM name="PostParam" method="Post" action="ServHtml">

<TABLE>

<tr>

<td><B>Employee </B> </td>

<td><input type="textbox" name="ename" size="25"

value=""></td>

</tr>

<tr>

<td><B>Phone </B> </td>

<td><input type="textbox" name="phoneno" size="25"

value=""></td>

</tr>

</TABLE>

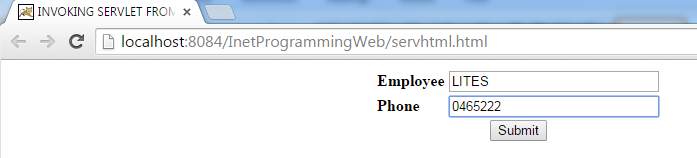
<INPUT type="submit" value="Submit">

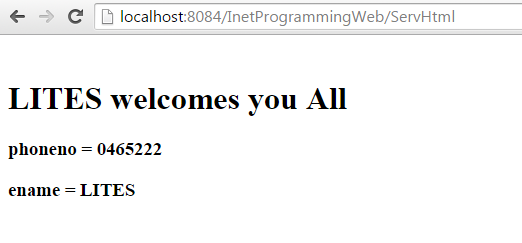
</FORM>

</CENTER>

</body>

</html>

****

****

**Result:**

Thus the program for Accessing servlet through HTML was compiled and executed successfully

d) Write programs in Java to create three-tier applications using servlets for conducting on-line examination for displaying student mark list. Assume that student information is available in a database which has been stored in a database server.

**Ex No: 5 Servlet Code for Online Examination Database**

**Date :**

**Aim:** To execute a servlet java program for Online examination using database concepts.

**Algorithm:**

1. Create a java program from HttpServlet architecture
2. Use getParameter method to collect all its mark values.
3. Display the values
4. Provide connection establishment and create the queries
5. Using statement interfaces display the values and find the total of it.
6. In HTML program create controls and access it through Servlet file
7. Display the design and get the result

**Program:**

**HTML Program:**

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>

<head><title>Database Test</title></head>

<body>

<center>

<h1>Online Examination</h1>

</center>

<form name="frm" action="NewServlet" method="POST">

<div align="left"><br></div>

<b>Seat Number:</b> <input type="text" name="Seat\_no">

<div align="Right">

<b>Name:</b> <input type="text" name="Name" size="50"><br>

</div>

<br><br>

<b>1. Every host implements transport layer.</b><br/>

<input type="radio" name="group1" value="True">True

<input type="radio" name="group1" value="False">False<br>

<b>2. It is a network layer's responsibility to forward packets reliably from source to destination</b><br/>

<input type="radio" name="group2" value="True">True

<input type="radio" name="group2" value="False">False<br>

<b>3. Packet switching is more useful in bursty traffic</b><br/>

<input type="radio" name="group3" value="True">True

<input type="radio" name="group3" value="False">False<br>

<b>4. A phone network uses packet switching</b><br/>

<input type="radio" name="group4" value="True">True

<input type="radio" name="group4" value="False">False<br>

<b>5. HTML is a Protocol for describing web contents</b><br/>

<input type="radio" name="group5" value="True">True

<input type="radio" name="group5" value="False">False<br>

<br><br><br>

<center>

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

</center>

</form>

</body>

</html>

**Servlet Program:**

import java.io.\*;

import java.sql.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class NewServlet extends HttpServlet

{

String message,Seat\_no,Name,ans1,ans2,ans3,ans4,ans5;

int Total=0;

Connection connect;

Statement stmt=null;

ResultSet rs=null;

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException

{

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

try

{

String url="jdbc:odbc:NEO";

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

connect=DriverManager.getConnection(url," "," ");

message="Thank you for participating in online Exam";

}

catch(ClassNotFoundException cnfex)

{

cnfex.printStackTrace();

}

catch(SQLException sqlex)

{

sqlex.printStackTrace();

}

catch(Exception excp)

{

excp.printStackTrace();

}

Seat\_no=request.getParameter("Seat\_no");

Name=request.getParameter("Name");

ans1=request.getParameter("group1");

ans2=request.getParameter("group2");

ans3=request.getParameter("group3");

ans4=request.getParameter("group4");

ans5=request.getParameter("group5");

if(ans1.equals("True"))

Total+=2;

if(ans2.equals("False"))

Total+=2;

if(ans3.equals("True"))

Total+=2;

if(ans4.equals("False"))

Total+=2;

if(ans5.equals("False"))

Total+=2;

try

{

Statement stmt=connect.createStatement();

String query="INSERT INTO student("+"Seat\_no,Name,Total"+") VALUES('"+Seat\_no+"','"+Name+"','"+Total+"')";

int result=stmt.executeUpdate(query);

stmt.close();

}

catch(SQLException ex)

{

ex.printStackTrace();

}

out.println("<html>");

out.println("<head>");

out.println("</head>");

out.println("<body>");

out.println("<center>");

out.println("<h1>"+message+"</h1>\n");

out.println("<h3>Yours results stored in our database</h3>");

out.print("<br><br>");

out.println("<b>"+"Participants and their Marks"+"</b>");

out.println("<table border=5>");

try

{

Statement stmt=connect.createStatement();

String query="SELECT \* FROM student";

rs=stmt.executeQuery(query);

out.println("<th>"+"Seat\_no"+"</th>");

out.println("<th>"+"Name"+"</th>");

out.println("<th>"+"Marks"+"</th>");

while(rs.next())

{

out.println("<tr>");

out.print("<td>"+rs.getInt(1)+"</td>");

out.print("<td>"+rs.getString(2)+"</td>");

out.print("<td>"+rs.getString(3)+"</td>");

out.println("</tr>");

}

out.println("</table>");

}

catch(SQLException ex)

{

ex.printStackTrace();

}

finally

{

try

{

if(rs!=null)

rs.close();

if(stmt!=null)

stmt.close();

if(connect!=null)

connect.close();

}

catch(SQLException e)

{ e.printStackTrace();

}

}

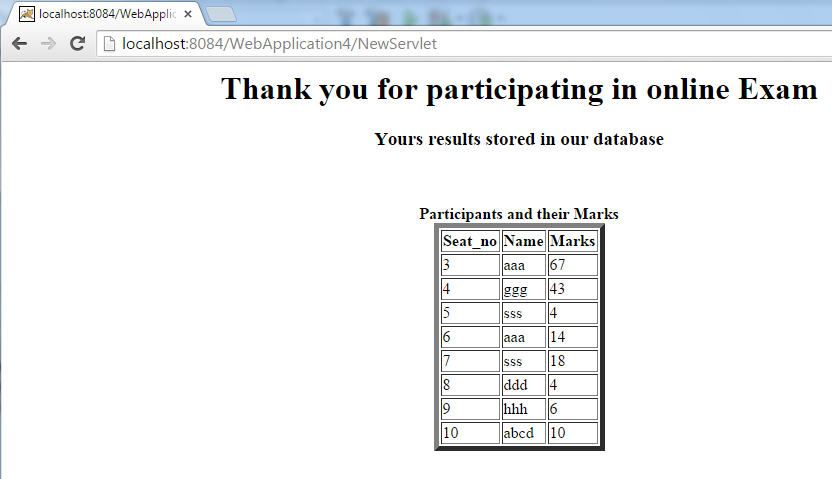
out.println("</center>");

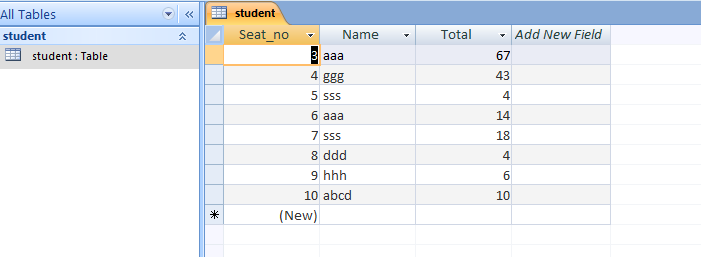
out.println("</body></html>");

Total=0;

}

}





**Result:**

Thus the program for online examination using databases in servlet through HTML was compiled and executed successfully

e) Write a program to lock servlet itself to a particular server IP address and port number. It requires an init parameter key that is appropriate for its servlet IP address and port before it unlocks itself and handles a request

**Servlet locked to a server:**

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.\*;

import java.net.\*;

import java.util.\*;

import javax.servlet.\*;

public class ServKeyLocked extends GenericServlet

{

public void service(ServletRequest req, ServletResponse res)

throws ServletException, IOException

{

res.setContentType("text/plain");

PrintWriter out = res.getWriter();

String key = getInitParameter("key");

String host = req.getServerName();

int port = req.getServerPort();

// Check if the init parameter "key" unlocks this server.

if (! keyFitsServer(key, host, port)) {

out.println("Pirated!");

}

else {

out.println("valid");

}

}

private boolean keyFitsServer(String key, String host, int port) {

if (key == null)

return false;

long numericKey = 0;

try

{

numericKey = Long.parseLong(key);

}

catch (NumberFormatException e)

{

return false;

}

byte hostIP[];

try

{

hostIP = InetAddress.getByName(host).getAddress();

}

catch (UnknownHostException e)

{

return false;

}

// Get the 32-bit IP address

long servercode = 0;

for (int i = 0; i < 4; i++)

{

servercode <<= 8;

servercode |= (hostIP[i] & 255);

}

// Concatentate the 32-bit port number

servercode <<= 32;

servercode |= port;

// Logical not

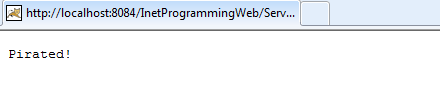
long accesscode = ~numericKey;

// The moment of truth: Does the key match?

return (servercode == accesscode);

}

}



**Snooping the Server:**

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.GenericServlet;

import javax.servlet.ServletException;

import javax.servlet.ServletRequest;

import javax.servlet.ServletResponse;

public class Serv extends GenericServlet

{

public void service(ServletRequest req, ServletResponse res)

throws ServletException, IOException

{

res.setContentType("text/plain");

PrintWriter out = res.getWriter();

out.println("req.getServerName(): " + req.getServerName());

out.println("req.getServerPort(): " + req.getServerPort());

out.println("getServletContext().getServerInfo(): " +

getServletContext().getServerInfo());

out.println("getServerInfo() name: " +

getServerInfoName(getServletContext().getServerInfo()));

out.println("getServerInfo() version: " +

getServerInfoVersion(getServletContext().getServerInfo()));

out.println("getServletContext().getAttribute(\"attribute\"): " +

getServletContext().getAttribute("attribute"));

}

private String getServerInfoName(String serverInfo)

{

int slash = serverInfo.indexOf('/');

if (slash == -1)

return serverInfo;

else

return serverInfo.substring(0, slash);

}

private String getServerInfoVersion(String serverInfo)

{

int slash = serverInfo.indexOf('/');

if (slash == -1)

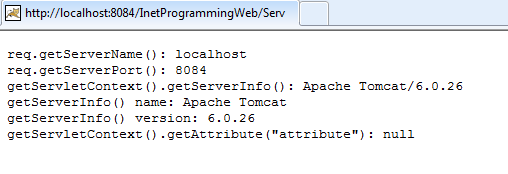
return null;

else

return serverInfo.substring(slash + 1);

}

}



f) Session tracking using hidden form fields and Session tracking for a hit count

**Servlet Program:**

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

public class ServSession extends HttpServlet {

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException

{

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

HttpSession session=request.getSession();

String Heading;

try

{

Integer cnt=(Integer)session.getAttribute("cnt");

out.println("<html>");

out.println("<head>");

out.println("<title>Servlet ServSession</title>");

out.println("</head>");

out.println("<body><br/>");

if(cnt==null)

{

cnt=new Integer(1);

Heading="Welcome you for the first time";

}

else

{

Heading="Welcome Once again";

cnt=new Integer(cnt.intValue()+1);

}

session.setAttribute("cnt", cnt);

out.println("<h1>Welcome to session Program: You are Visiting this site as :"+cnt +

" Visitor </h1>");

out.println("<h1>You are Working Servlet Program at " + request.getContextPath () +" <br/><br/>"+Heading);

out.println("</h1> <h2>Your Hidden Session ID is:"+session.getId()+" :ID</h2>");

out.println("</body>");

out.println("</html>");

} finally

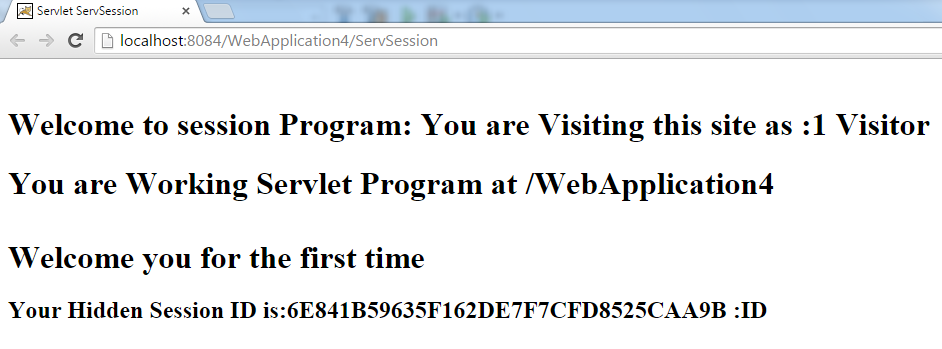
{

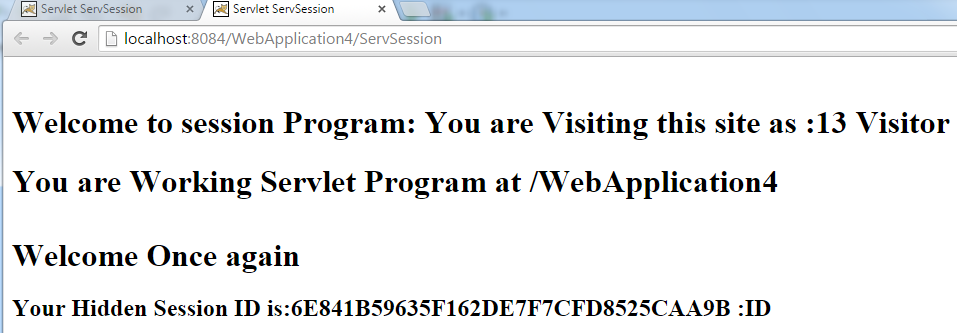
out.close();

}

}

}





g) Install TOMCAT web server. Convert the static web pages of programs 1&2 into dynamic web pages using servlet (or JSP) and cookies. Hint: Users information (user id, password, credit card number) would be stored in web.xml. Each user should have a separate Shopping Cart.

PROCEDURE:

* First install the tomcat into the system.
* Then make a subdirectly(eg., tr) in the \tomcat\webapps.
* Under tr create WEB-INF directory and also place the html files in this tr directory only.
* Next under WEB-INF create two subclasses lib, classes and web.xml
* Next place all the class files under the classes and jar files(servlet-api.jar,classes12.jar etc…) under lib subdirectories.
* After this start tomcat by giving the following command at the install\_dir>tomcat>bin
* Catalina.bat run
* At the I.E(web browser) give the url as http;//localhost:8080//tr/htmlfile or servlet url pattern
* Port no 8080 is assigned for the tomcat.

**Web.xml**

<?xml version="1.0" encoding="UTF-8"?>

<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd">

<init-param>

<param-name>username</param-name>

<param-value>user1</param-value>

</init-param>

<init-param>

<param-name>password</param-name>

<param-value>pass1</param-value>

</init-param>

<init-param>

<param-name>cardid</param-name>

<param-value>c1</param-value>

</init-param>

<servlet>

<servlet-name>LoginServlet</servlet-name>

<servlet-class>LoginServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>LoginServlet</servlet-name>

<url-pattern>/LoginServlet</url-pattern>

</servlet-mapping>

</web-app>

**LoginForm.html**

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>

<head>

<title>Login Form</title>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

</head>

<body>

<form action="LoginServlet" method="post">

Enter Username:<input type="text" name="user"><br/>

Enter Password:<input type="text" name="pass"><br/>

Enter Card ID:<input type="text" value="" name="card"><br/>

<input type="submit" value="Login">

</form>

</body>

</html>

LoginServlet.java

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.Cookie;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class LoginServlet extends HttpServlet

{

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException

{

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

try {

String user=request.getParameter("user");

String pass=request.getParameter("pass");

String card=request.getParameter("card");

boolean flag=true;

String userid[]=getInitParameter("username").split(",");

String password[]=getInitParameter("password").split(",");

String cardid[]=getInitParameter("cardid").split(",");

int i;

for(i=0;i<userid.length;i++)

{

if(userid[i].equals(user) && password[i].equals(pass) && cardid[i].equals(card))

{

flag=false;

Cookie coo=new Cookie("currentUser",user);

out.println("<html><body><h1>");

out.println("Login Page Successfully executed");

//response.sendRedirect("http://localhost:8084/WebApplication2/LoginServletS");

response.addCookie(coo);

out.println("Welcome to "+coo);

}

}

out.println("</h1></body></html>");

} finally {

out.close();

}

}

}

**ADVANCE CONCEPTS:**

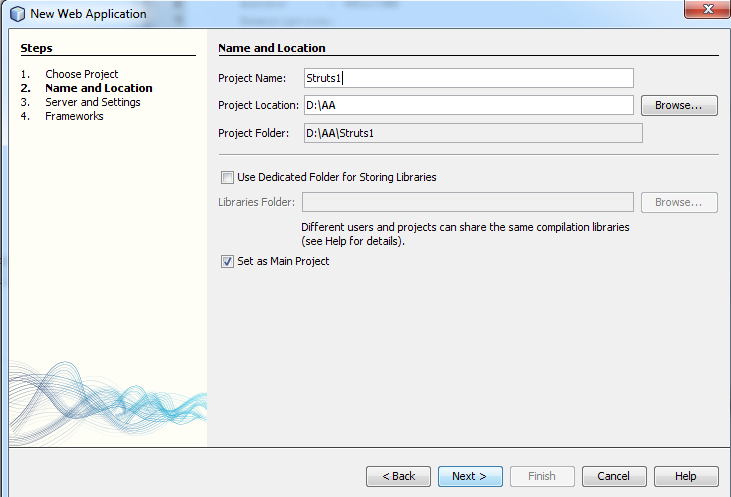
a) Implement a simple program using following frameworks

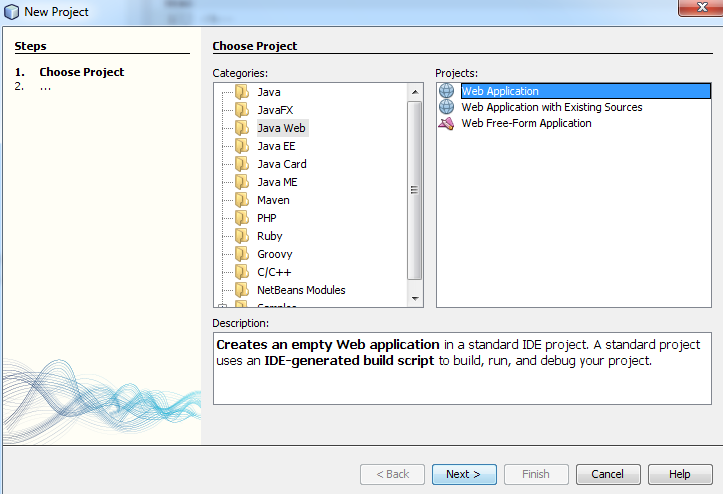
i. JSP Struts Framework

Struts

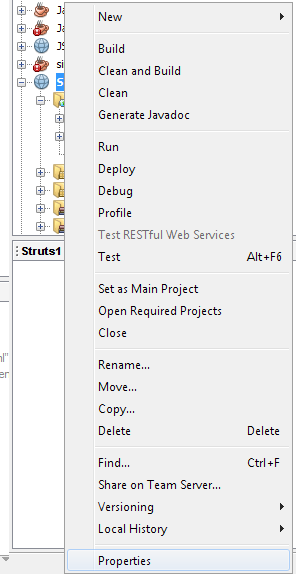
Step:1

New Project – Java Web – Web Application



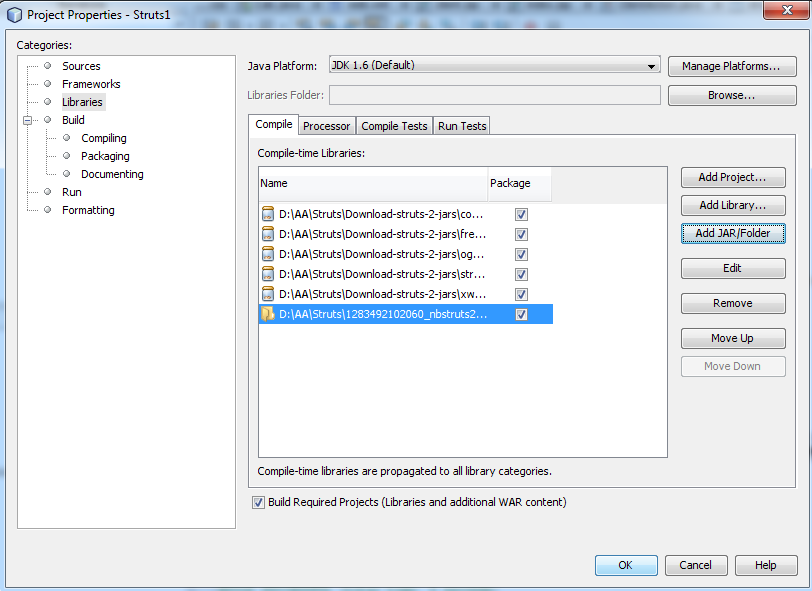


RC - Click Project– Project Properties



Step:2

Add library files for Struts Framework



Step: 3

Edit web.xml file from WEB-INF folder in web pages

Web.xml

<?xml version="1.0" encoding="UTF-8"?>

<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd">

<display-name>Struts</display-name>

<filter>

<filter-name>Struts2</filter-name>

<filter-class>org.apache.struts2.dispatcher.FilterDispatcher</filter-class>

</filter>

<filter-mapping>

<filter-name>Struts2</filter-name>

<url-pattern> /\* </url-pattern>

</filter-mapping>

<welcome-file-list>

<welcome-file>index.jsp</welcome-file>

</welcome-file-list>

</web-app>

Step:4

Edit index.jsp file

Index.jsp

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<%@taglib prefix="s" uri="/struts-tags"%>

<html>

<body>

<h1> Struts 2 Application</h1>

Enter Client Details:

<s:form action="ClientAction" method="post">

<s:textfield label="Name" name="name"/>

<s:textfield label="Age" name="age"/>

<s:textfield label="Salary" name="salary"/>

<s:submit/>

</s:form>

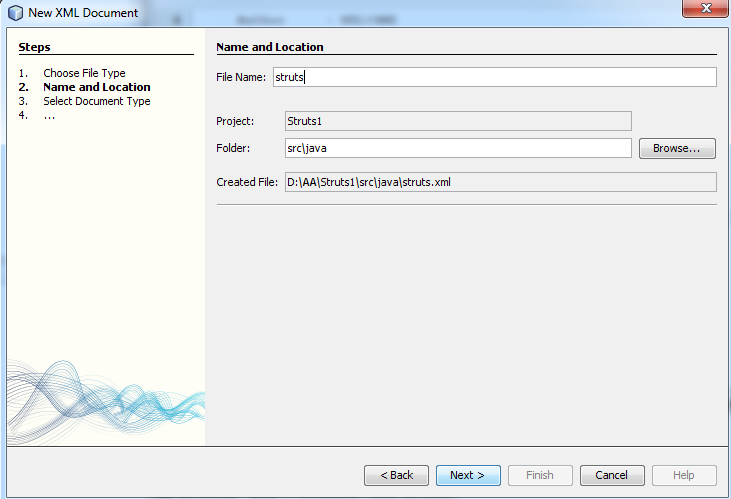
</body>

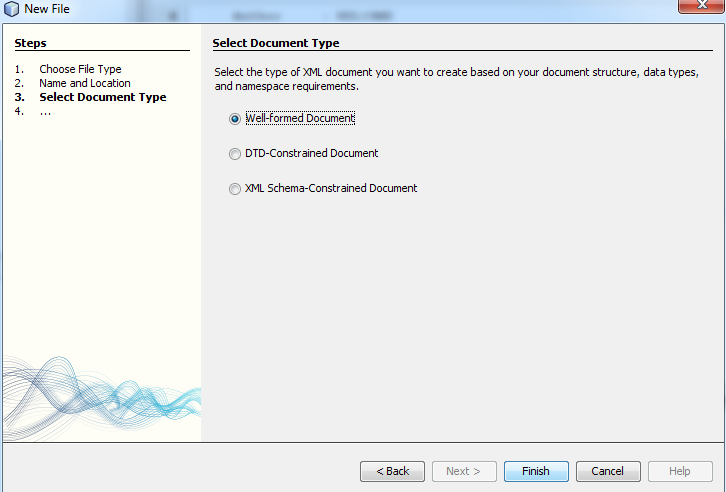
</html>

Step: 5

With the source packages folder create struts.xml file from xml document

Struts.xml





<?xml version="1.0" encoding="UTF-8"?>

<!--

Document : struts.xml

Created on : August 15, 2015, 7:21 PM

Author : WELCOME

Description:

Purpose of the document follows.

-->

<!DOCTYPE struts PUBLIC

"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"

"http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>

<include file="struts-default.xml"/>

<package name="default" extends="struts-default">

<action name="ClientAction" class="ClientAction">

<result name="success">client.jsp</result>

<result name="error">error.jsp</result>

<result name="input">index.jsp</result>

</action>

</package>

</struts>

Step: 6

Create a ClientAction.java file into the source packages

import com.opensymphony.xwork2.ActionSupport;

public class ClientAction extends ActionSupport

{

String name;

int age;

int salary;

public String getName()

{ return name;

}

public void setName(String name)

{ this.name= name;

}

public int getAge()

{ return age;

}

public void setAge(int age)

{ this.age=age;

}

public int getSalary()

{ return salary;

}

public void setSalary(int salary)

{ this.salary=salary;

}

public String execute()throws Exception

{

if((getName().equals("")) ||(getName()==null )||(getAge()==0) || (getSalary()==0))

{ return ERROR; }

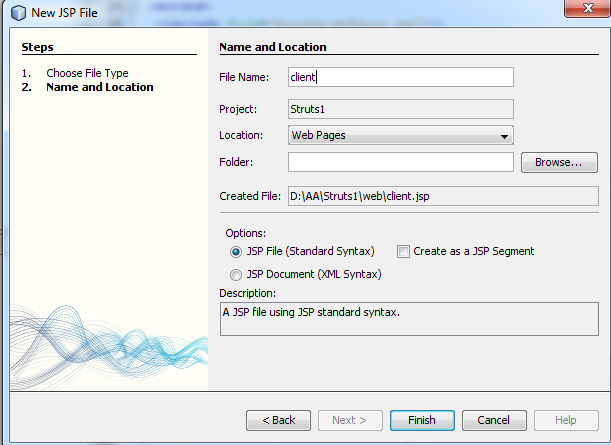
else

{ return SUCCESS; }

}}

Step: 7

Create a client.jsp file in folder and display the values



<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">

<%@taglib prefix="s" uri="/struts-tags"%>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Web java server Page</title>

</head>

<body><h1> The executed Struts Program Output are:</h1>

<h3>Thank you

<s:property value="name"/> <br/>

Your Age: <s:property value="age"/><br/>

Your Salary: <s:property value="salary"/></h3><br/>

</body>

</html>

Step: 8

Create error.jsp file to display error details.

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Error JSP Page</title>

</head>

<body>

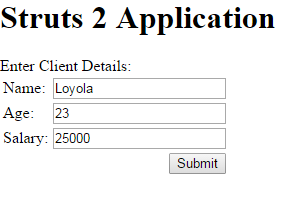
<h4>The error is shown because of blank field, Invalid data entry or null</h4>

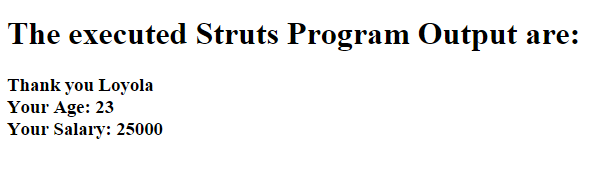
</body>

</html>

Step: 9

Build and run the application





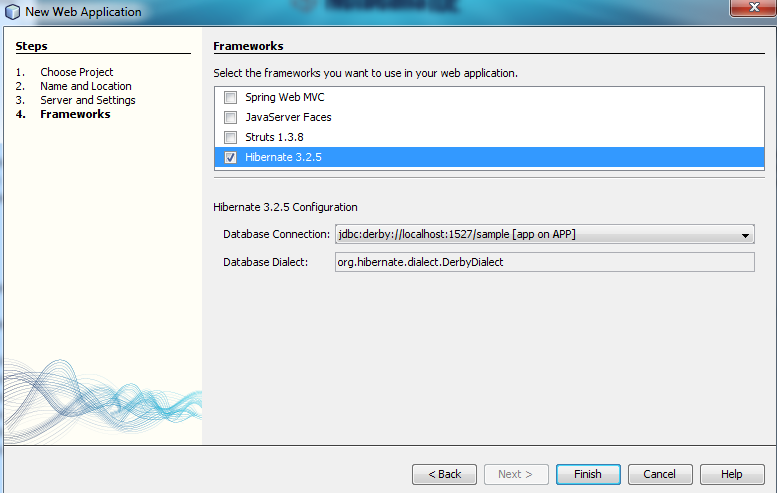
ii. Hibernate

Hibernate Example

Step: 1

Create a new Project from web application

Select the sample database fron the database connection drop down list and click finish

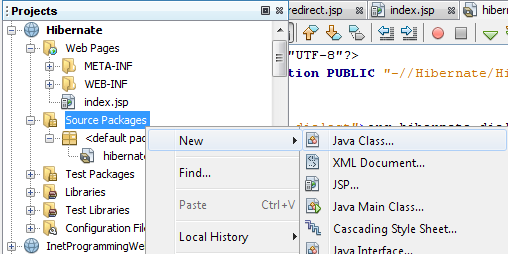


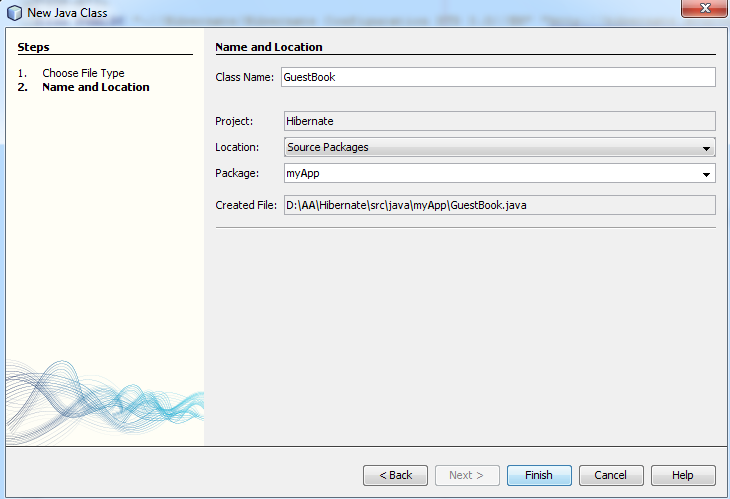
Save the changes to this file

Step:2

RC the source package node in project choose new java class file

Generate getter and setter methods





package myApp;

public class GuestBook implements java.io.Serializable

{

private Integer visitorNo;

private String visitorName;

private String message;

private String messageDate;

public GuestBook(String visitorName, String message, String messageDate)

{

this.visitorName=visitorName;

this.message=message;

this.messageDate=messageDate;

}

public String getMessage() {

return this.message;

}

public void setMessage(String message) {

this.message = message;

}

public String getMessageDate() {

return this.messageDate;

}

public void setMessageDate(String messageDate) {

this.messageDate = messageDate;

}

public String getVisitorName() {

return this.visitorName;

}

public void setVisitorName(String visitorName) {

this.visitorName = visitorName;

}

public Integer getVisitorNo() {

return this.visitorNo;

}

public void setVisitorNo(Integer visitorNo) {

this.visitorNo = visitorNo;

}

}

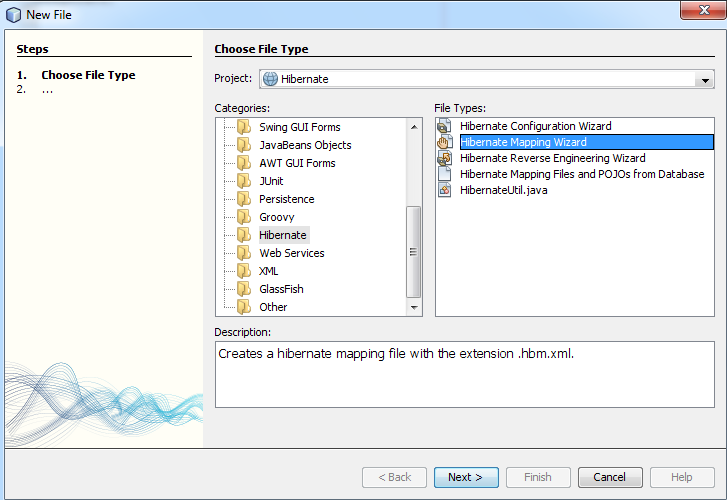
Step:3

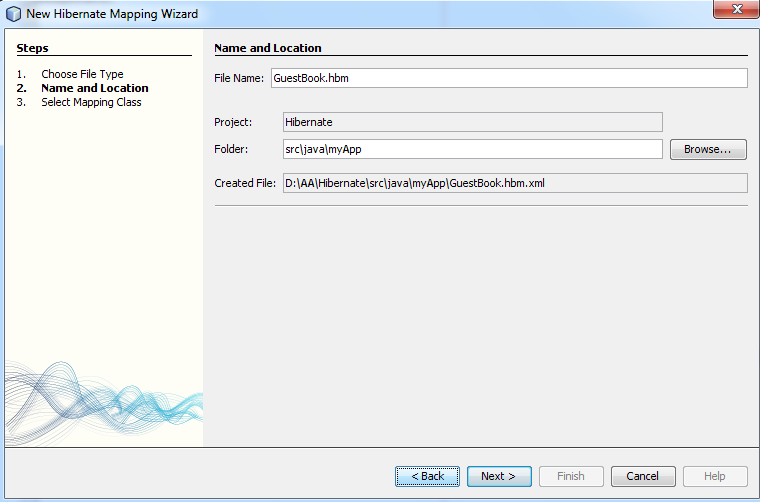
Create a database in service folder (DerbyDatabase)

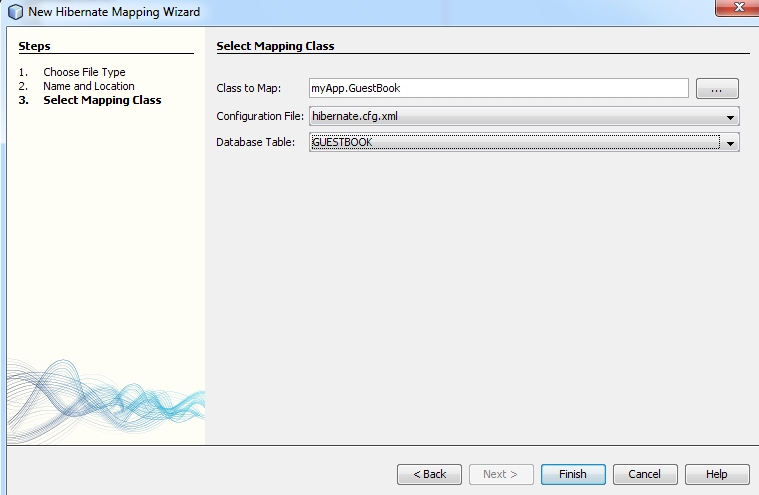
Create table GuestBook(VisitorNo Integer primary key, VisitorName varchar(50), Message varchar(100), MessageDate varchar(40));

Step:4

Go to project – New File – select Hibernate – select Hibernate Mapping wizard







Type GuestBook.hbm for the file and set the folder src/java/myApp and click nexy

Type myApp.GuestBook for the class to map

Select Database

Open the file

When you click finish IDE creates the GuestBook.hbm.xml file in the source packages created.

Modify the xml file

Step:5

Create GuestBook.hbm.xml file

GuestBook.hbm.xml

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name="myApp.GuestBook" table="GUESTBOOK"/>

<id name="visitorNo" type="java.lang.Integer">

<column name="VisitorNo"/>

<generator class="identity"/>

</id>

<property name="visitorName" type="String">

<column name="VisitorName" length="50"/>

</property>

<property name="message" type="String">

<column name="Message" length="100"/>

</property>

<property name="messageDate" type="String">

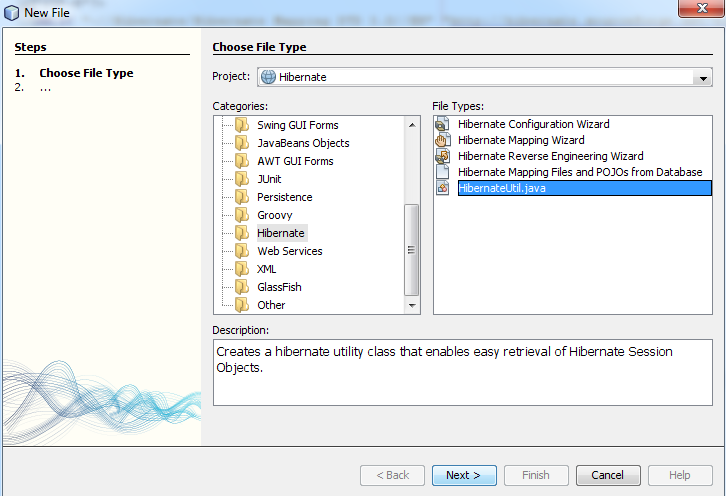
<column name="MessageDate" length="40"/>

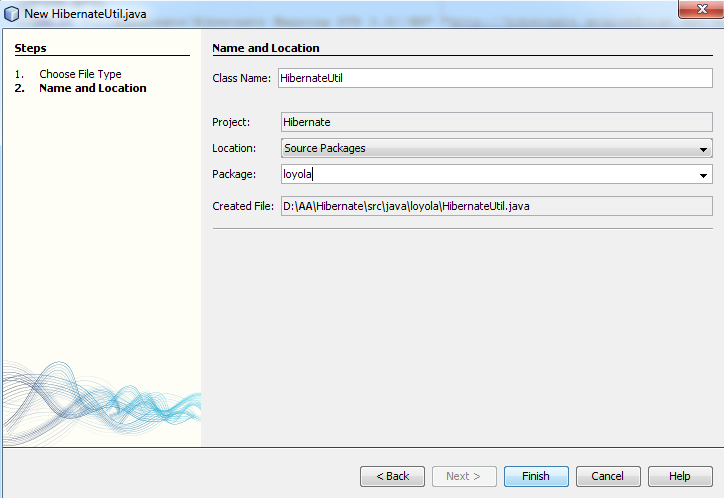
</property>

</hibernate-mapping>

Step:6

RC project folder – select new file – Hibernate – HibernateUtil.Java





When you click finish, the class open in the editor. You can close the file because you do not need to edit the file.

HibernateUtil.java

package loyola;

import org.hibernate.cfg.AnnotationConfiguration;

import org.hibernate.SessionFactory;

public class HibernateUtil {

private static final SessionFactory sessionFactory;

static {

try {

// Create the SessionFactory from standard (hibernate.cfg.xml)

// config file.

sessionFactory = new AnnotationConfiguration().configure().buildSessionFactory();

} catch (Throwable ex) {

// Log the exception.

System.err.println("Initial SessionFactory creation failed." + ex);

throw new ExceptionInInitializerError(ex);

}

}

public static SessionFactory getSessionFactory() {

return sessionFactory;

}

}

Step: 7

Create GuestBookEntry.jsp file

GuestBookEntry.jsp

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Guest Book Page</title>

</head>

<body>

<table> <tr> <td> <table> <tr> <td>Sign the Guest Book </td> </tr> </table>

</td>

</tr>

<tr><td><hr/> </td></tr>

<tr><td>

<form action="<%=request.getContextPath()%>/GuestBookView.jsp" method="post">

<table>

<tr><td> Visitor Name: </td>

<td> <input name="guest" maxlength="25" size="50"/> </td> </tr>

<tr><td>Message: </td>

<td> <textarea rows="50" cols="50" name="message">

</textarea></td></tr>

<tr><td colspan="2"> <input type="submit" name="btnSubmit" value="Submit"/></td>

</tr>

</table>

</form>

</td></tr>

</table>

</body>

</html>

Step: 8

Create GuestBookView.jsp file

GuestBookView.jsp

<%@page contentType="text/html" pageEncoding="UTF-8" import = "org.hibernate.SessionFactory, org.hibernate.cfg.Configuration,

org.hibernate.Session, org.hibernate.Transaction, java.util.List, java.util.Iterator, myapp.GuestBook" %>

<%!

SessionFactory sessionFactory;

org.hibernate.Session hibSession;

List<GuestBook> guestbook;

%>

<% sessionFactory = new Configuration().configure().buildSessionFactory();

hibSession=sessionFactory.openSession();

Transaction tx=null;

String submit=request.getParameter("btnSubmit");

if(submit != null &&("Submit").equals(submit))

{

GuestBook gb=new GuestBook();

try

{

tx=hibSession.beginTransaction();

String guest=request.getParameter("guest");

String message=request.getParameter("message");

String messageDate=new java.util.Date().toString();

gb.setVisitorName(guest);

gb.setMessage(message);

gb.setMessageDate(messageDate);

hibSession.save(gb);

tx.commit();

}

catch(RuntimeException e)

{

if(tx!=null)

tx.rollback();

throw e;

}

response.sendRedirect("GuestBookView.jsp");

}

try{

hibSession.beginTransaction();

guestbook=hibSession.createQuery("from GuestBook").list();

}

catch(RuntimeException e)

{

throw e;

}

hibSession.close();

%>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Guest Book Page</title>

</head>

<body>

<table>

<tr>

<td>

<table>

<tr>

<td>

View the GuestBook

</td>

<td>

Click <a href="<%=request.getContextPath()%>/GuestBookEntry.jsp">Here</a> to sign the guest book

</td>

</tr>

</table>

</td>

</tr>

<tr>

<td>

<table>

<%

Iterator iterator=guestbook.iterator();

while(iterator.hasNext())

{

GuestBook objGb=(GuestBook)iterator.next();

%>

<tr>

<td>

On<%=objGb.getMessageDate()%>,<br/>

<%=objGb.getVisitorName()%>

<%=objGb.getMessgae()%>

<br/><br/>

</td>

</tr>

<%}%>

</table>

</td>

</tr>

</table>

</body>

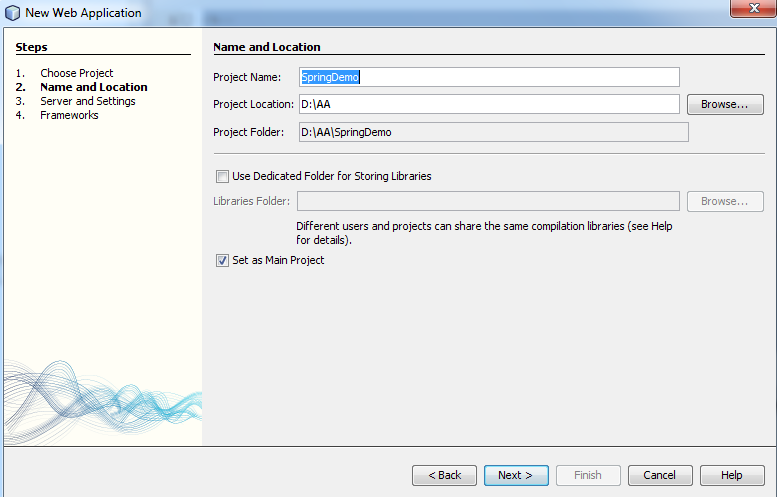
</html>

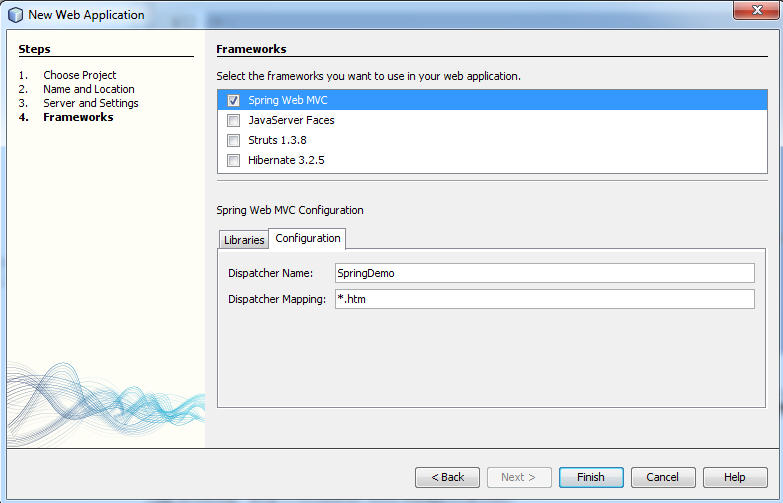
Step: 9

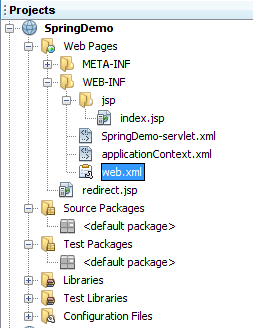
Build the project and execute the JSP file

iii. Spring

Spring Demo







Step:2

Web.xml

<?xml version="1.0" encoding="UTF-8"?>

<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd">

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/applicationContext.xml</param-value>

</context-param>

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<servlet>

<servlet-name>SpringDemo</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<load-on-startup>2</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>SpringDemo</servlet-name>

<url-pattern>\*.htm</url-pattern>

</servlet-mapping>

<session-config>

<session-timeout>

30

</session-timeout>

</session-config>

<welcome-file-list>

<welcome-file>index.jsp</welcome-file>

</welcome-file-list></web-app>

Step : 3

Index.jsp file

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Index Java server Page</title>

</head>

<body>

<h1>Welcome to Loyola</h1>

</body>

</html>

Step : 4

Build and deploy the spring package

b) Explore the following application in AJAX: Searching in real time with live searches, Getting the answer with auto complete, Chatting with friends ,Dragging and dropping with Ajax, Getting instant login feedback, Ajax-enabled popup menus, Modifying Web pages on the fly.

AJAX

Program:

Ajax.html

<html>

<head>

<script type="text/javascript">

function loadXMLDoc(url)

{

var xmlhttp;

var txt,x,xx,i;

if (window.XMLHttpRequest)

{ // code for IE7+, Firefox, Chrome, Opera, Safari

xmlhttp=new XMLHttpRequest();

}

else

{ // code for IE6, IE5

xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");

}

xmlhttp.onreadystatechange=function()

{

if (xmlhttp.readyState==4 && xmlhttp.status==200)

{ txt="<table border='1'><tr><th>Title</th><th>Artist</th></tr>";

x=xmlhttp.responseXML.documentElement.getElementsByTagName("CD");

for (i=0;i<x.length;i++)

{

txt=txt + "<tr>";

xx=x[i].getElementsByTagName("TITLE");

{

try

{ txt=txt + "<td>" + xx[0].firstChild.nodeValue + "</td>";

}

catch (er)

{ txt=txt + "<td> </td>";

}

}

xx=x[i].getElementsByTagName("ARTIST");

{ try

{ txt=txt + "<td>" + xx[0].firstChild.nodeValue + "</td>";

}

catch (er)

{ txt=txt + "<td> </td>";

}

}

txt=txt + "</tr>";

}

txt=txt + "</table>";

document.getElementById('txtCDInfo').innerHTML=txt;

}

}

xmlhttp.open("GET",url,true);

xmlhttp.send();

}

</script>

</head>

<body>

<div id="txtCDInfo">

<button onclick="loadXMLDoc('CDElement.xml')">Get CD info</button>

</div>

</body>

</html>

**CDElement.xml**

<?xml version="1.0" encoding="ISO-8859-1"?>

<!-- Edited by XMLSpy® -->

<CATALOG>

<CD>

<TITLE>Empire Burlesque</TITLE>

<ARTIST>Bob Dylan</ARTIST>

<COUNTRY>USA</COUNTRY>

<COMPANY>Columbia</COMPANY>

<PRICE>10.90</PRICE>

<YEAR>1985</YEAR>

</CD>

<CD>

<TITLE>Still got the blues</TITLE>

<ARTIST>Gary Moore</ARTIST>

<COUNTRY>UK</COUNTRY>

<COMPANY>Virgin records</COMPANY>

<PRICE>10.20</PRICE>

<YEAR>1990</YEAR>

</CD>

<CD>

<TITLE>Eros</TITLE>

<ARTIST>Eros Ramazzotti</ARTIST>

<COUNTRY>EU</COUNTRY>

<COMPANY>BMG</COMPANY>

<PRICE>9.90</PRICE>

<YEAR>1997</YEAR>

</CD>

c) Write a web services for finding what people think by asking 500 people’s opinion for any consumer product

d) Write a web services for predicting for any product sales

Aim:

To create a web service and access the client through Servlet applications.

Description:

JAX-WS Stands for Java API for XML Web services

JAX-WS is a technology for building webservices and clients that communicates using XML.

JAX-WS allows developers to write message oriented as well as RPC oriented Webservices.

In JAX-WS a web service operation invocation is represented by an XML based protocol such as SOAP.

Soap specification defines the envelope structure, encoding rules for representing web service invocation and responses.

CLIENT

JAX-WS runtime

SERVICE

JAX-WS runtime

SOAP message

(Communication between Web service and Client)

Web Service in Java J2EE

Developing a Web service:

Net Beans 6.9.1 IDE takes care of all the implementation details for you, so you can concentrate on coding the business logic of your web service.

Create the project – choose file – New Project – select Java web – Web application Click next

Name the Project with MarkService – Click next – select Server – Next – Finish.

Creating Web Service:

Right Click – Project Node – Choose New – Web service (Name, Package Name – Finish)

In Web Service folder - select - web service application - Right click – Web service name – add operation – enter values – ok.

In program java change the statement - return aa+bb+cc;

Deploying and Testing of web service

Right Click – project node, choose properties, click run.

Context path=/markservices

Tick the Display browser as run.

Relative URL : /MarkWS? Tester – ok.

Consuming the Web Service

Click Project folder and RC Run.

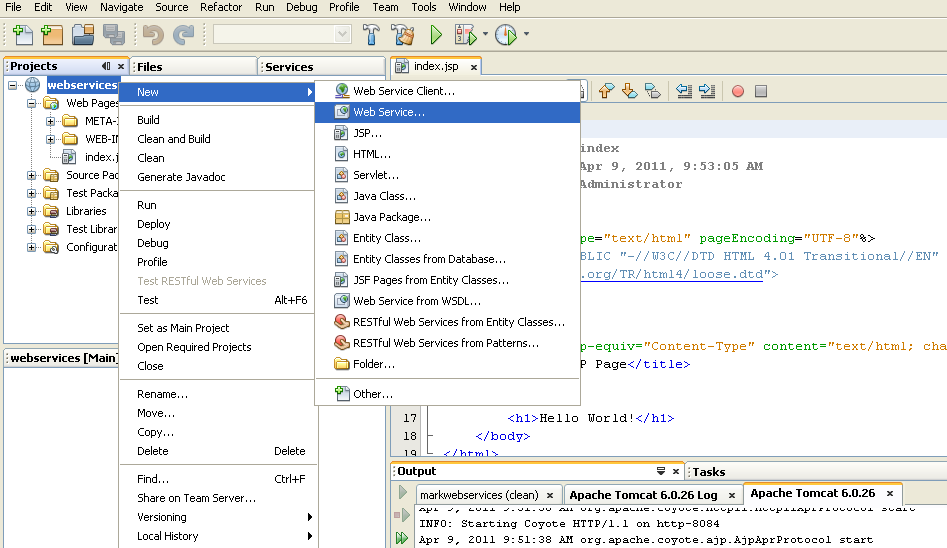
**Java Servlet client in Web Application**

File – New Project – Java Web – Web Application – next

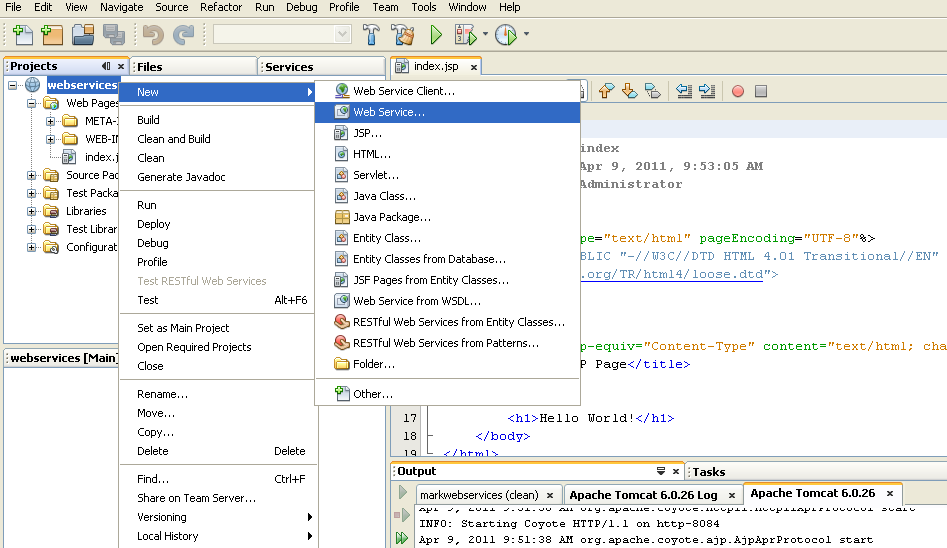
Project Name – MarkServletClient

Finish

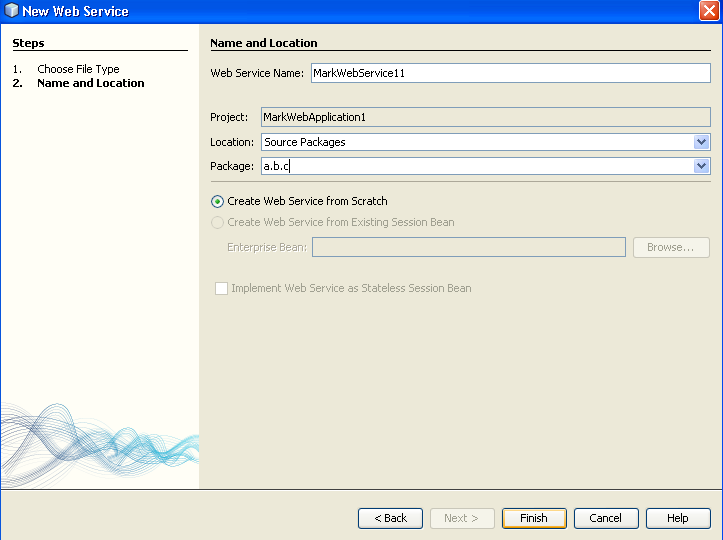
MarkServletClient – RightClick – new – WebServiceClient – click

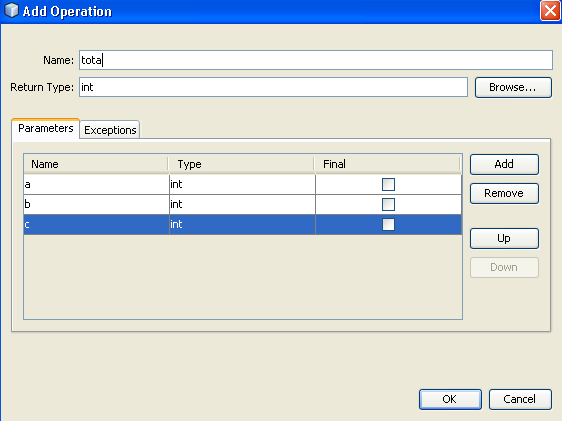


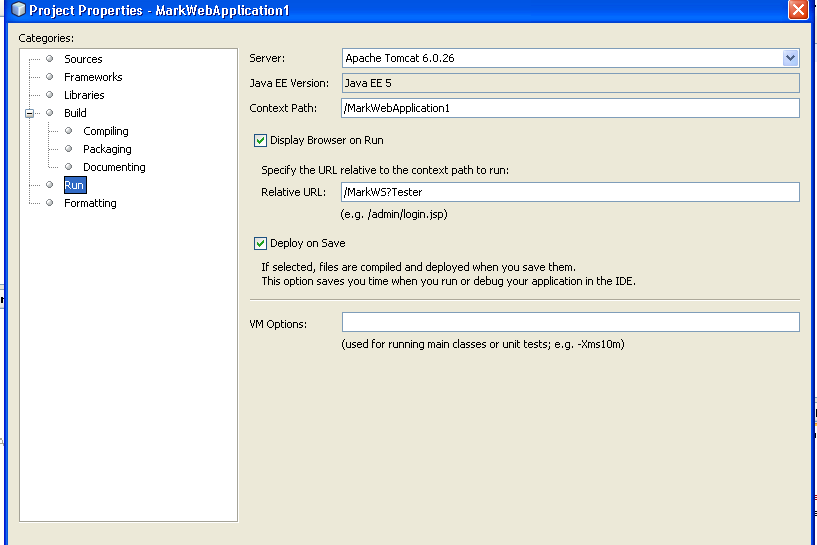
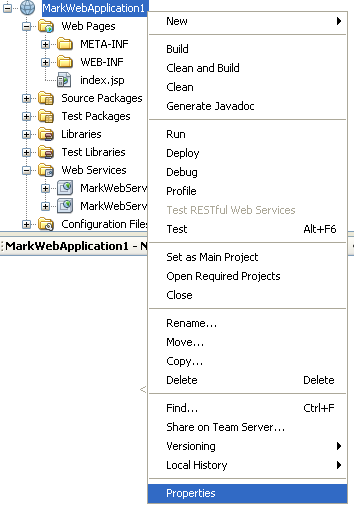
Project – Browse – select Webservice – click – oK

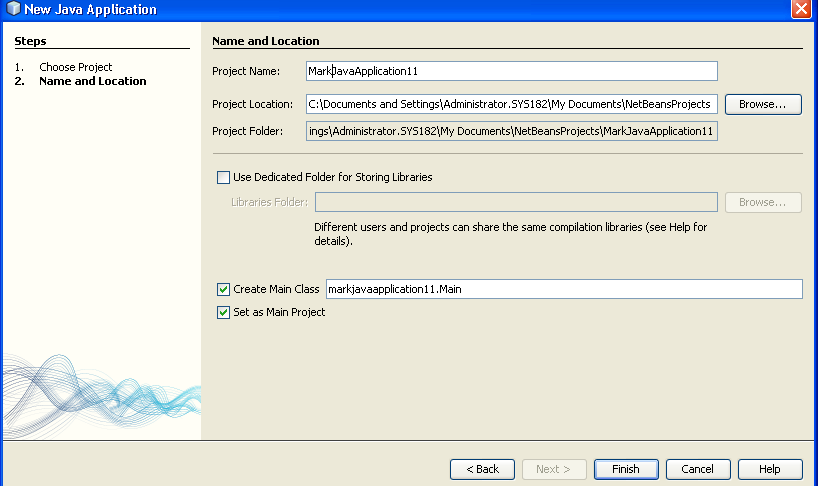


Package – finish

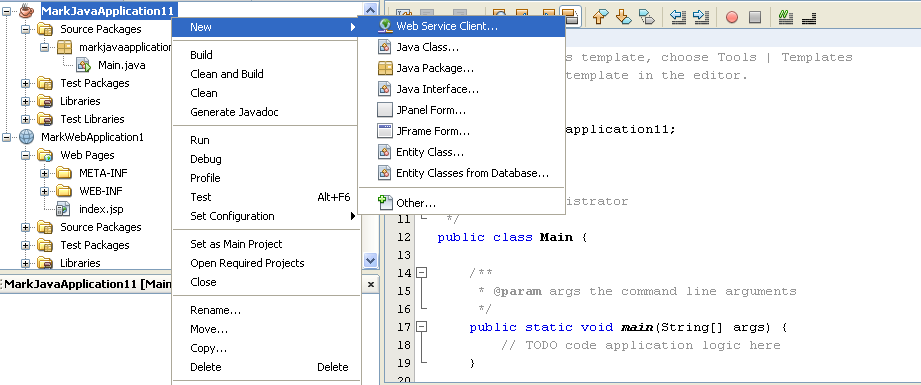






File – new – Servlet - project – Package – Give its name

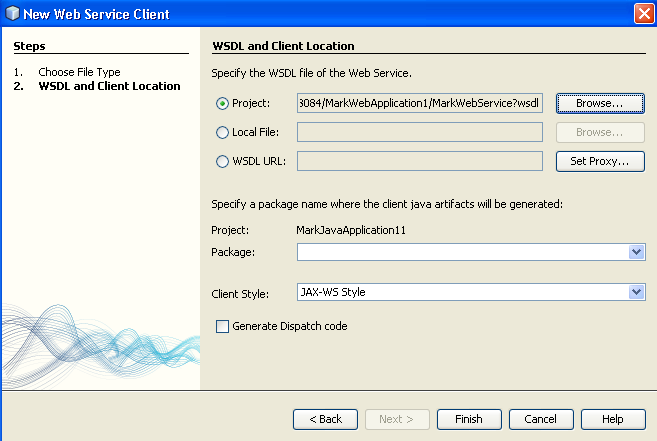
Select webserviceClient



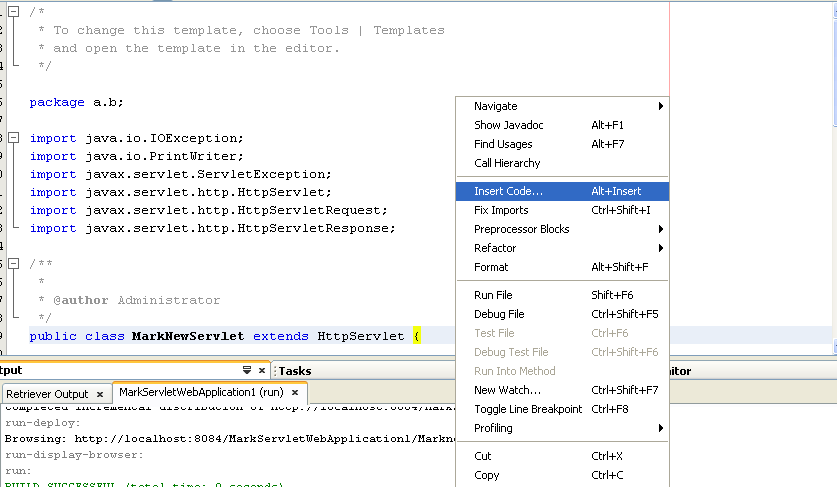
Right Click – properties – run

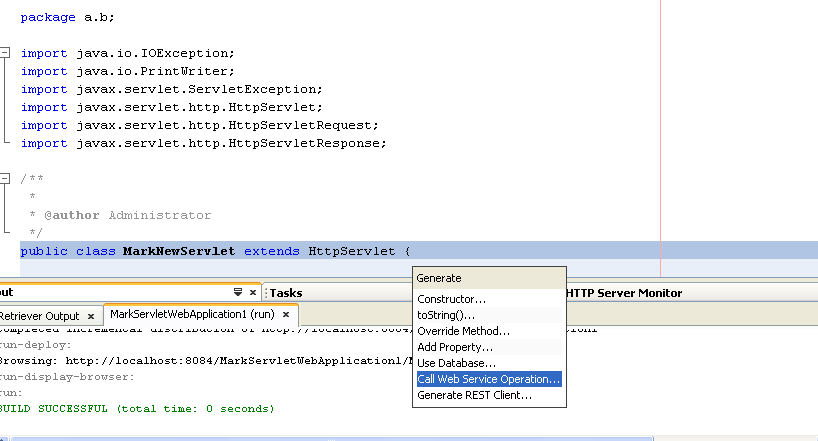
Context Path: /MarkServletClient

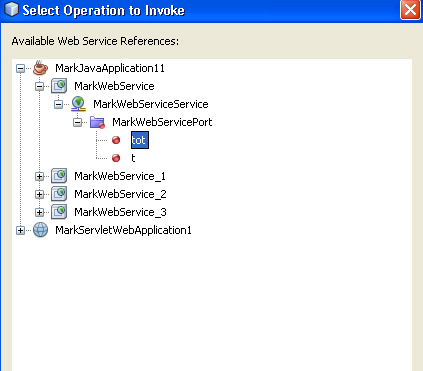
URL: /ClientServlet ok



Program – Right click – insert code – Call Web Service Operation





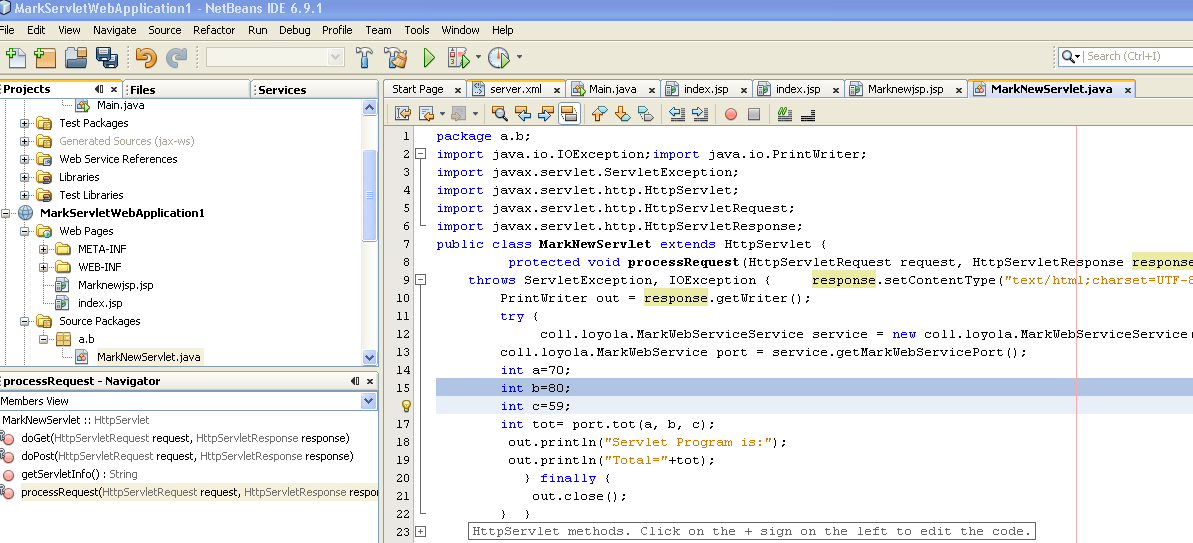


Change the code by including

Int a=50; Int b=60; Int c=70;

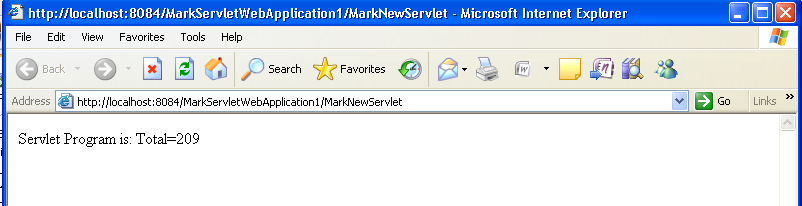
Int total = port.total(a,b,c);

System. out. println (total);



Right click project – Run

See the output in the browser window.



Conclusion: Thus the program for Web Service using Java client, Servlet client and JSP client is created and accessed successfully.