Practical No. 1

Objective: Demonstration of a simple RMI application.

Theory:

Steps to Implement RMI: -

1. Create a Remote interface - extending java.rmi Remote interface
2. Implementing Remote interface- by making a server class
3. Write server program - Create the objects of the server class and register them in the rmi registry
4. Write the client program - access the remote server object, through the golbal name in the regsitry
5. Compiling and creating stub and skeleton - by rmic tool
6. Start the rmi registry service
7. Run the server program
8. Run the client program - Running on separate command window
9. javac *.java
10. rmic AddImp
11. start rmiregistry
12. java Server
13. java Client

Source Code:

AddImp.java

```
import
    java.rmi.server.UnicastRemoteObject;
    public class AddImp extends UnicastRemoteObject implements Add
    {
       public AddImp() throws Exception
       {
          super();
       }
       public int add(int x, int y)
       {return (x+y);   }

    }
```

Server.java

```
    Import java.rmi.*;
    public class Server extends AddImp
    {
       public Server() throws Exception
       { }
       public static void main(String args[]) throws Exception
       {
    try
          {
             AddImp ob = new AddImp();
             Naming.bind("A",ob);
             System.out.println("Server started");
          }
          catch(Exception e){
```

```
            System.out.println("Server exception" + e);
            }
        }
    }
```

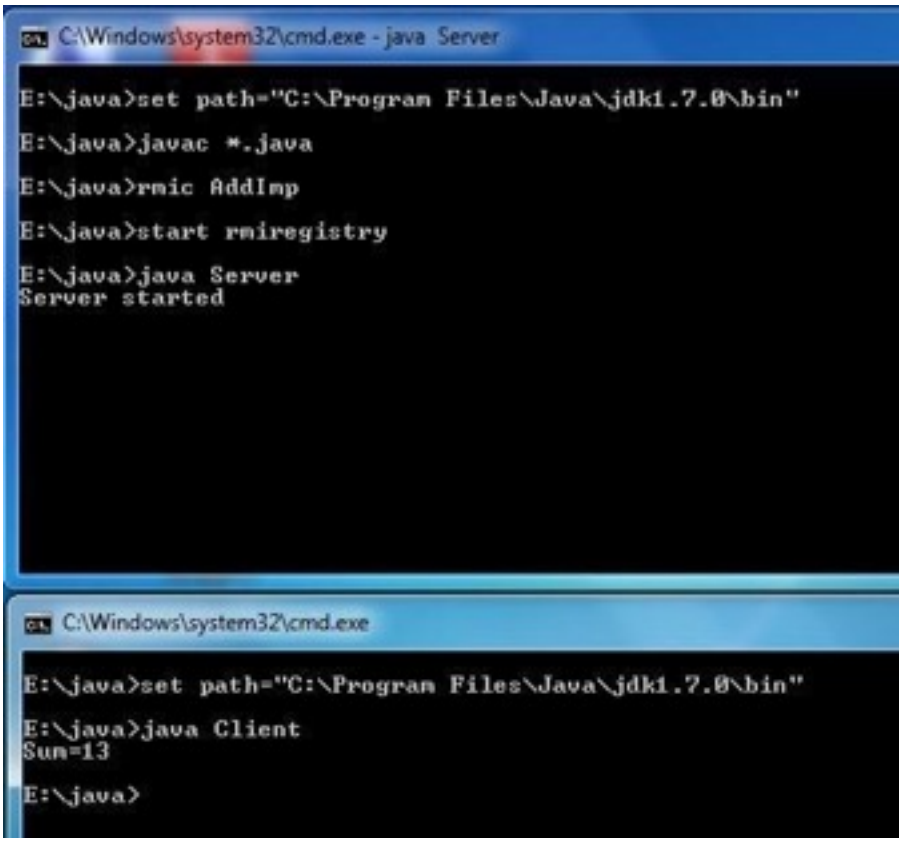
Client.java import

```
java.rmi.*; public
class Client
{
    public Client() { }
    public static void main(String args[]) throws Exception
    {
        Add ob = (Add)Naming.lookup("A");
int c=ob.add(5,8);
System.out.println("Sum="+c);
    }
}
```

Add.java import java.rmi.Remote;

```
import java.rmi.RemoteException;
public interface Add extends Remote


{
    public int add(int x, int y) throws RemoteException; }
```

Output:



Result: The given program has been compiled and executed successfully.

Practical No.2

Objective: Calculator using RMI. **Theory:**

Steps to Implement RMI: -

1. Create a Remote interface - extending java.rmi Remote interface
2. Implementing Remote interface- by making a server class
3. Write server program - Create the objects of the server class and register them in the rmi registry
4. Write the client program - access the remote server object, through the golbal name in the regsitry
5. Compiling and creating stub and skeleton - by rmic tool
6. Start the rmi registry service
7. Run the server program
8. Run the client program - Running on separate command window
9. javac *.java
10. rmic CalcImp
11. start rmiregistry
12. java Server
12. java Client

Source Code:

Calcimp.java import

```
java.rmi.server.UnicastRemoteObject;
public class Calcimp extends UnicastRemoteObject implements Calc
{
   public Calcimp()throws Exception
   {
      super();

   }
   public int calc(int x,int y,char ch)
{
      switch(ch)
      {
         case '+': return(x+y);
case '-': return(x-y);
case 'x': return(x*y);
case '/': return(x/y);
default : return 0;
      }
   }
}
```

Server.java import

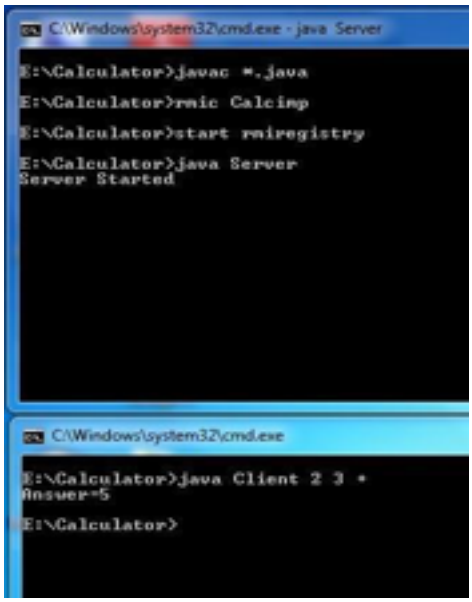
```
java.rmi.*;
public class Server extends Calcimp
{
   public Server() throws Exception
```

```
    {}
            public static void main (String args[]) throws Exception
    {
try
        {
            Calcimp ob = new Calcimp();
            Naming.bind("A",ob);
            System.out.println("Server Started");
                }
        catch(Exception e)
        {
            System.out.println("Server Exception"+e);
        }
    }
}
```

Client.java import

```
java.rmi.*; public
class Client
{
    public Client(){}
    public static void main(String args[]) throws Exception
    {
        int num1,num2;
char ch;
        Calc ob = (Calc)Naming.lookup("A");
        num1 = Integer.parseInt(args[0]);
num2 = Integer.parseInt(args[1]);
ch= args[2].charAt(0);                int c=
ob.calc(num1,num2,ch);
System.out.println("Answer="+c);
    }
}
```

Output:



Result: The given program has been compiled and executed successfully

Practical No. 3

Objective: JDBC connectivity in Java Application using MySQL server at the backend.

Inserting, deleting and updating the records using Statement and Prepared Statement interfaces.

Theory:

Java Database Connectivity (JDBC) is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is Java based data access technology and used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle Corporation. It provides methods to query and update data in a database, and is oriented towards relational databases. A JDBC-to-ODBC bridge enables connections to any ODBC-accessible data source in the Java virtual machine (JVM) host environment.. Java JDBC is a java API to connect and execute query with the database. JDBC API uses jdbc drivers to connect with the database.

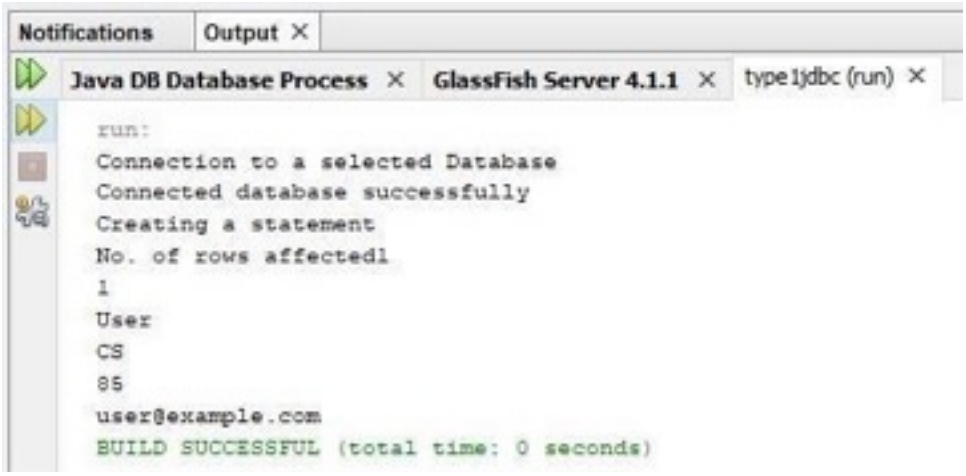
Source Code:

JDBC.java package

```
type1jdbc; import
java.sql.*;


public class Type1jdbc {
        public static void main(String[] args) {
                String JDBC_Driver_1 = "com.mysql.jdbc.Driver";
               String DB_URL = "jdbc:mysql://localhost:3306/test";
                String USER = "root";
                String PASS = "amity";
                Connection conn = null;
         PreparedStatement ps = null;
   try
                                {
try
                                 {
                                        Class.forName(JDBC_Driver_1);
                                }
                        catch(ClassNotFoundException e){System.out.println(e); }
                System.out.println("Connection to a selected Database");
conn = DriverManager.getConnection(DB_URL,USER,PASS);
                        System.out.println("Connected database successfully");
                        System.out.println("Creating a statement");
                       String sql = "insert into test.Studentdata
                       (ID,Name,Branch,Percentage,Email)"+"values(?,?,?,?,?)"
                ;       ps = conn.prepareStatement(sql);                ps.setInt(1,1);
                ps.setString(2,"Shantanu"); ps.setString(3,"CS"); ps.setInt(4,85);
                        ps.setString(5,"shanu@gmail.com");



                        int i = ps.executeUpdate();
                        System.out.println("No. of rows affected"+i); ps =
                       conn.prepareStatement("select * from test.Studentdata"); ResultSet
                       rs = ps.executeQuery(); while(rs.next())
                        {
                            System.out.println(rs.getInt("ID"));
                            System.out.println(rs.getString("Name"));
                            System.out.println(rs.getString("Branch"));
                            System.out.println(rs.getInt("Percentage"));
                            System.out.println(rs.getString("Email"));
```

```
            }
        ps.close();
        conn.close();
            }
                catch(Exception e){System.out.println(e); }
    }
}
```

Output:



Result: The given program has been compiled and executed successfully.

Practical No. 4

Objective: A simple servlet Application. Taking a “username” and “password” from client in textboxes and displaying it back as response on clicking of “Submit” button.

Theory:

A Java servlet is a Java program that extends the capabilities of a server. Although servlets can respond to any types of requests, they most commonly implement applications hosted on Web servers. Such Web servlets are the Java counterpart to other dynamic Web content technologies such as PHP and ASP.NET.

Source Code:

Web-app:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/
web-app_3_1.xsd">
    <servlet>
        <servlet-name>Intro</servlet-name>
        <servlet-class>P1.Intro</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>Intro</servlet-name>
        <url-pattern>/Intro</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
</web-app>
```

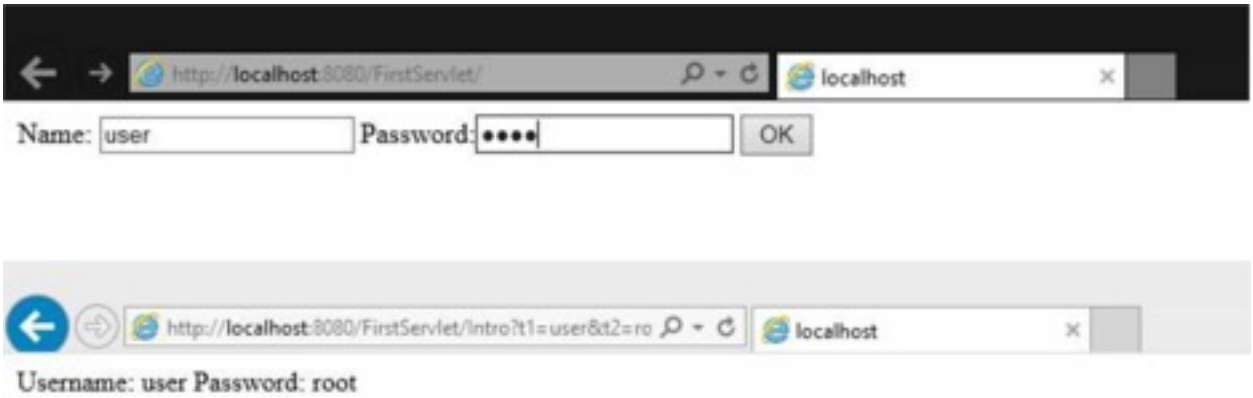
Index.html:

```
 <html>
    <body>
        <form action="NewServlet" method="GET">
        Name: <input type="text" name="t1" value="" />
        Password:<input type="password" name="t2" value="" />
        <input type="submit" value="OK" name="S" />
        </form>
       </body>
</html>
```

Intro.java:

```
package P1;

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 Intro extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    }
  @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
           String name=request.getParameter("t1");
         String password=request.getParameter("t2");
         PrintWriter out = response.getWriter();
         out.write("Username:"+name);
         out.write("\n");
         out.write("password"+password);
        processRequest(request, response);
    }
@Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
 @Override
    public String getServletInfo() {
        return "Short description";
    }
}
```

Output:



Result: The given program has been compiled and executed successfully.

Practical No. 5

Objective: Making a Login Application using Servlet and verify the credentials using a table in the MySQL database.

Theory: A Java servlet is a Java program that extends the capabilities of a server. Although servlets can respond to any types of requests, they most commonly implement applications hosted on Web servers. Such Web servlets are the Java counterpart to other dynamic Web content technologies such as PHP and ASP.NET.

Source Code:

index.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>TODO supply a title</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <form action="Loginvalidate" method="POST">
            UserName:<input type="text" name="Name" value="" /><br>
            Password:<input type="Password" name="Password" value="" /><br>
            <input type="submit" value="OK" />
        </form>
    </body>
</html>
```

Loginvalidate.java

```
package Login; import
java.io.IOException; import
java.io.PrintWriter; import
java.sql.Connection; import
java.sql.DriverManager; import
java.sql.PreparedStatement; import
java.sql.ResultSet; import
java.sql.SQLException; import
java.sql.Statement; import
javax.servlet.RequestDispatcher; 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 Loginvalidate extends HttpServlet {
    protected void processRequest(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException { response.setContentType("text/
html;charset=UTF-8"); try (PrintWriter out =
response.getWriter()) { out.println("<!DOCTYPE html>");
out.println("<html>"); out.println("<head>");
```

```
out.println("<title>Servlet Loginvalidate</title>");
out.println("</head>");                    out.println("<body>");
        out.println("<h1>Servlet Loginvalidate at " + request.getContextPath() + "</h1>");
out.println("</body>");
out.println("</html>");
    }
  }
  @Override
  protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException { processRequest(request, response);
  }
  @Override
  protected void doPost(HttpServletRequest request,
HttpServletResponse response)                    throws ServletException, IOException {
    String str1 = request.getParameter("Name");
    String str2 = request.getParameter("Password");
    String JDBC_DRIVER_1 = "com.mysql.jdbc.Driver"; String
  DB_URL = "jdbc:mysql://localhost:3306/user";
  String USER = "root";
  String PASS = "sana";
Connection conn = null;


  try{
  Class.forName(JDBC_DRIVER_1);              conn =
DriverManager.getConnection(DB_URL, USER, PASS);
    }
    catch(ClassNotFoundException | SQLException e)
    {System.out.println(e);}
PreparedStatement ps = null;
  try{
  System.out.println("Connecting to a selected Database");
  System.out.println("Connected database successfully...");
  System.out.println("Creating a statement...");
  String sql = ("select * from LOGIN where USERNAME=? AND PASS =?");
ps = conn.prepareStatement(sql);                    ps.setString(1, str1);          ps.setString(2, str2);
  ResultSet rs = ps.executeQuery();
if(rs.next()==true){
    HttpSession sess = request.getSession(); sess.setAttribute("User",
    str1);          response.sendRedirect("welcome.jsp");

  }
else{
  PrintWriter pw=response.getWriter();
pw.println("<h2> Invalid User </h2>");
      //response.sendRedirect("index.html");
```

```
            RequestDispatcher rd = request.getRequestDispatcher("index.html");
rd.include(request, response);
    }
        /*if(str1.equals("Admin") && str2.equals("Admin"))
        {
          response.sendRedirect("welcome.html");
        }
else
        {
          PrintWriter pw=response.getWriter();
pw.println("<h2> Invalid User </h2>");
          //response.sendRedirect("index.html");
          RequestDispatcher rd = request.getRequestDispatcher("index.html");
rd.include(request, response);
        }*/
ps.close();
conn.close();

    } catch (SQLException e){
        System.out.println(e);
    }
    }

                                public  String
                                        return



    @Override

getServletInfo() {
"Short description";
    }

}
```

welcome.jsp

```
<%--
    Document      : welcome
    Created on : Jan 31, 2018, 8:48:20 AM
    Author       : DEll
--%>


<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"> <title>JSP
        Page</title>
```

```
    </head>
    <body>
        <form action="logout">
        WELCOME...



        <%= (String)session.getAttribute("User") %> <br/>
        <input type="submit" value="Logout" />
        </form>
    </body>
</html>
```

UserLogin.java package Login; 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 UserLogin extends HttpServlet {

    protected void
HttpServletResponse response)        processRequest(HttpServletRequest                request,
                                                throws

ServletException, IOException { }

    @Override
    protected      void      doGet(HttpServletRequest            request,
HttpServletResponse response)                throws ServletException, IOException {
        String str1 = request.getParameter("Name");


        String str2 = request.getParameter("Email");
PrintWriter pw = response.getWriter();
pw.println("<h1> Welcome </h1>");                pw.println("<h2>
Username:</h2>"+ str1);              pw.println("<h2> Email
:</h2>"+str2);
    }
    @Override
    protected      void      doPost(HttpServletRequest            request,
HttpServletResponse response)                throws ServletException, IOException {

PrintWriter pw = response.getWriter();
pw.println("<h1> Welcome </h1>");                pw.println("<h2>
```

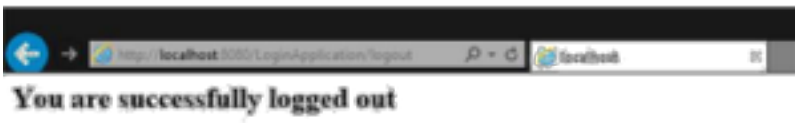
```
Username:</h2>"+ str1);                pw.println("<h2> Email
:</h2>"+str2);
        //processRequest(request, response);
    }

    @Override
getServletInfo() {        public    String
                                    return

"Short description";
    }
}
```

Logout.java package

```
Login;

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
logout extends HttpServlet {
    protected void processRequest(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException { response.setContentType("text/
html;charset=UTF-8"); try (PrintWriter out =
response.getWriter()) { out.println("<!DOCTYPE html>");
out.println("<html>"); out.println("<head>");
            out.println("<title>Servlet logout</title>");
out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet logout at " + request.getContextPath() + "</h1>");
            out.println("</body>");
out.println("</html>");
        }
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
HttpSession sess = request.getSession();
sess.removeAttribute("User"); sess.invalidate(); PrintWriter
        pw=response.getWriter();
            pw.println("<h2> You are successfully logged out </h2><br/>");
pw.println("<h2> Click Here to <a href=index.html>Login</a> </h2>");
    }
    @Override
```

```
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
processRequest(request, response);
    }
    @Override
    public String getServletInfo() {
return "Short description";
    }
}
```

Output:



Result: The given program has been compiled and executed successfully.

Practical No. 6

Objective: Displaying the request headers on client side.

Theory:

Request header is used to pass additional information about the request or itself to the server. Request header can be used by client to pass useful information. getHeaderNames() and getHeader() methods of javax.servlet.http.HttpServletRequestinterface can be used to get the header information.

Source Code:

```
import java.io.IOException; import
java.io.PrintWriter; import
javax.servlet.ServletException; import
javax.servlet.annotation.WebServlet; import
javax.servlet.http.HttpServlet; import
javax.servlet.http.HttpServletRequest; import
javax.servlet.http.HttpServletResponse;
import java.util.*;


@WebServlet(urlPatterns = {"/NewServlet"}) public class
NewServlet extends HttpServlet {
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException
{ response.setContentType("text/html;charset=UTF-8"); try
(PrintWriter out = response.getWriter()) { out.println("<!
DOCTYPE html>");
out.println("<html>");                    out.println("<head>");
            out.println("<title>Servlet NewServlet</title>");
            out.println("</head>");
out.println("<body>");
            out.println("<h1>Servlet NewServlet at " + request.getContextPath() +
"</h1>");              out.println("</body>");                    out.println("</html>");
        }
    }
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException { response.setContentType("text/html"); PrintWriter
        out=response.getWriter();

        String title="HTTP Header Request Example";
        String docType="<! doctype html public\"-//w3c//dtd html 4.0"+"transitional // en\">\n";
        out.println(docType +"<html>\n"+"<head><title>"+title+"</title></head >\n"+"<body
bgcolor=\" #f0f0f0 \">\n"+"<h1 align=\"center \">"+title +
                "</h1> \n"+"<table width=\"100% \"border=\"1\"align=\"center\">\n"
        +"<tr bgcolor =\"#949494\"> \n"+"<th>HeaderName</th><th>
HeaderValues</th>\n"+"</tr>\n");
        Enumeration<String> headerNames=request.getHeaderNames();
while(headerNames.hasMoreElements())
        {
            String headerName = (String)headerNames.nextElement();
out.print("<tr><td>"+headerName+"</td>\n");                    String
headValue = request.getHeader(headerName);
```

```
            out.println("<td>"+headValue+"</td></tr>\n");
                }
                out.println("</table>\n</body></html>");
                processRequest(request, response);
            }
            @Override
            protected void doPost(HttpServletRequest request, HttpServletResponse
        response) throws ServletException, IOException
        { doGet(request,response);
                //processRequest(request, response);
            }
            @Override
            public String getServletInfo() {
        return "Short description";
            }
        }
```

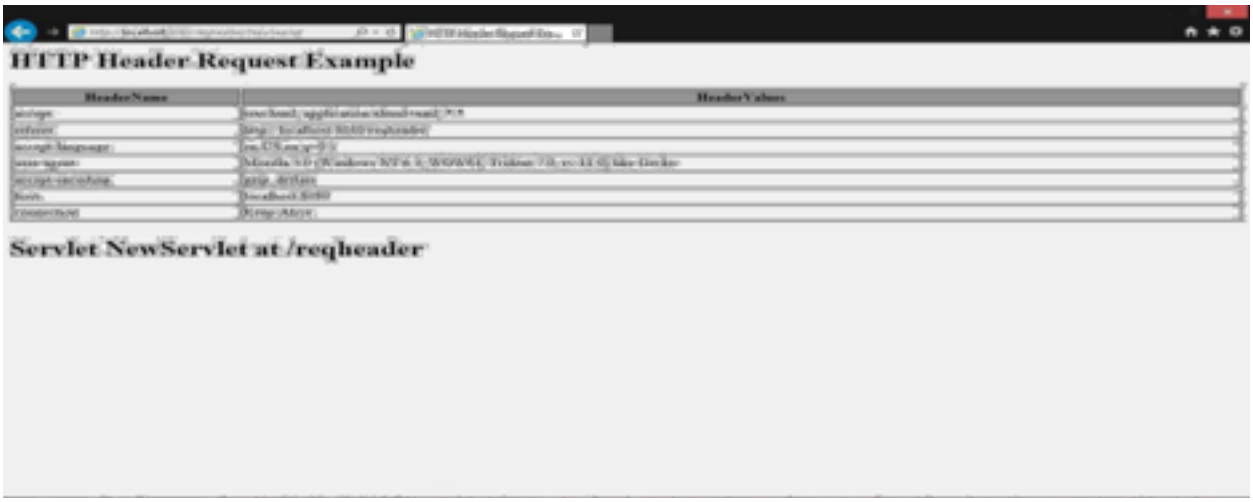
Web-app:

```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-
app_3_1.xsd"
         version="3.1">
   <display-name>Servlet-Form GET </display-name>
   <servlet>
      <servlet-name>readAllHeaders</servlet-name>
      <servlet-class>NewServlet</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>readAllHeaders</servlet-name>
      <url-pattern>/readAllHeaders</url-pattern>
   </servlet-mapping>
</web-app>
```

Index.html:

```
<html>
   <head>
      <title>TODO supply a title</title>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
   </head>
   <body>
      <form action="NewServlet">
         <input type="submit" value="OK" />
      </form>
   </body>
</html>
```

Output:



Result: The given program has been compiled and executed successfully.

Practical No. 7

Objective: Demonstrating data sharing using servlet context.

Theory:

One of the major purposes of Web applications is to keep data separate. Each Web application maintains its own table of sessions and its own servlet context. Each Web application also uses its own class loader; this behavior eliminates problems with name conflicts but means that static methods and fields can't be used to share data among applications. However, it *is* still possible to share data with cookies or by using ServletContext objects that are associated with specific URLs.

Source Code:

SourceServlet.java package

```
scope;


import java.io.IOException; import
java.io.PrintWriter; import
javax.servlet.ServletContext; import
javax.servlet.ServletException; import
javax.servlet.http.HttpServlet; import
javax.servlet.http.HttpServletRequest; import
javax.servlet.http.HttpServletResponse;

public class SourceServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
PrintWriter out = response.getWriter();                    ServletContext




sc = getServletContext();                sc.setAttribute("Name","
Kiran");          out.println("Welcome to First Servlet");
        out.println("<a href = TragetServlet> Get Name </a>");
processRequest(request, response);
    }
}
```

TargetServlet.java package

```
scope;


import java.io.IOException; import
java.io.PrintWriter; import
javax.servlet.ServletContext; import
javax.servlet.ServletException; import
javax.servlet.http.HttpServlet; import
javax.servlet.http.HttpServletRequest; import
javax.servlet.http.HttpServletResponse;
```

```
public class TragetServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
PrintWriter out = response.getWriter();
        ServletContext sc = getServletContext();
String str = sc.getAttribute("Name").toString();
out.println("Welcome"+str);
processRequest(request, response);
    }
}
```

Output:

Result: The given program has been compiled and executed successfully.

Practical No. 8

Objective: Demonstrating send redirect and request dispatcher for sharing data and control across servlets.

Theory:

SendRedirect() will search the content between the servers. It is slow because it has to intimate the browser by sending the URL of the content. Then browser will create a new request for the content within the same server or in another one.

RquestDispatcher is for searching the content within the server. Its the server side process and it is faster compare to the SendRedirect() method. But the thing is that it will not intimate the browser in which server it is searching the required date or content, neither it will not ask the browser to change the URL in URL tab. So it causes little inconvenience to the user.

Source Code:

index.html

```
<!DOCTYPE html>
<html>
    <body>
        <form action="ReqDispatch" >
            UserName: <input type="text" name="t1" value="" /><br>
           Password: <input type="password" name="t2" value="" />
            <input type="submit" value="Submit" />
        </form>
    </body>
</html>
```

Request Dispatcher (ReqDispatch.java):

```
package ReqDispatcher;

import java.io.IOException; import
java.io.PrintWriter; import
javax.servlet.RequestDispatcher;
import
javax.servlet.ServletException;
import
javax.servlet.http.HttpServlet;
import
javax.servlet.http.HttpServletRequ
est; import
javax.servlet.http.HttpServletResp
onse; import
javax.servlet.http.HttpSession;


public class ReqDispatch extends HttpServlet {
    protected void doGet(HttpServletRequest request,HttpServletResponse
response)throws ServletException, IOException
    {
```

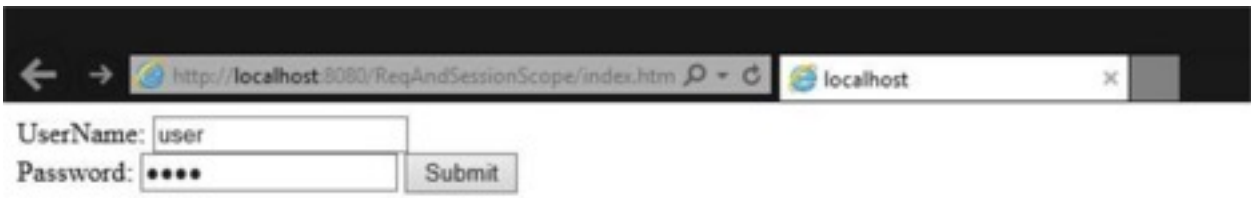
```
        System.out.println("First Servelet");
        PrintWriter pw=response.getWriter();
String str = request.getParameter("t2");
if(str.equals("Sana"))
    {
    RequestDispatcher rd = request.getRequestDispatcher("secondservlet"); rd.forward(request, response); // send the response of second servlet directly to the client

}
 else
    {
        pw.println("<h3>Invalid User</h3>");
        RequestDispatcher rd = request.getRequestDispatcher("index.html"); rd.include(request, response); //include method carries the response of index.html includes it to the response of this page and send it to the client
    }
        //HttpSession session=request.getSession();
        //String str = request.getParameter("t2");
        //session.setAttribute("Pass", str);
       //response.sendRedirect("/secondservlet");//the first servlet redirects the client to the second servlet for service.
    }
}
```

secondservlet.java package

```
ReqDispatcher;

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 secondservlet extends HttpServlet {



public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException
{
  PrintWriter out=response.getWriter();
  String str = request.getParameter("t2");
  //out.write(str);
    //HttpSession session = request.getSession();
//String str = session.getAttribute("Name").toString();
out.write("Welcome"+str);
```

}
}

Output:



Send Redirect (SendRed.java):

```
package SenRedirect;
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
SendRed extends HttpServlet {
   protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
        throws ServletException, IOException {
response.setContentType("text/html;charset=UTF-8");          try (PrintWriter out =
response.getWriter()) {          out.println("<!DOCTYPE html>");
out.println("<html>");          out.println("<head>");
        out.println("<title>Servlet SendRed</title>");
out.println("</head>");          out.println("<body>");
        out.println("<h1>Servlet SendRed at " + request.getContextPath() + "</h1>");
        out.println("</body>");
out.println("</html>");
      }
   }
   @Override
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
HttpSession session = request.getSession();
```

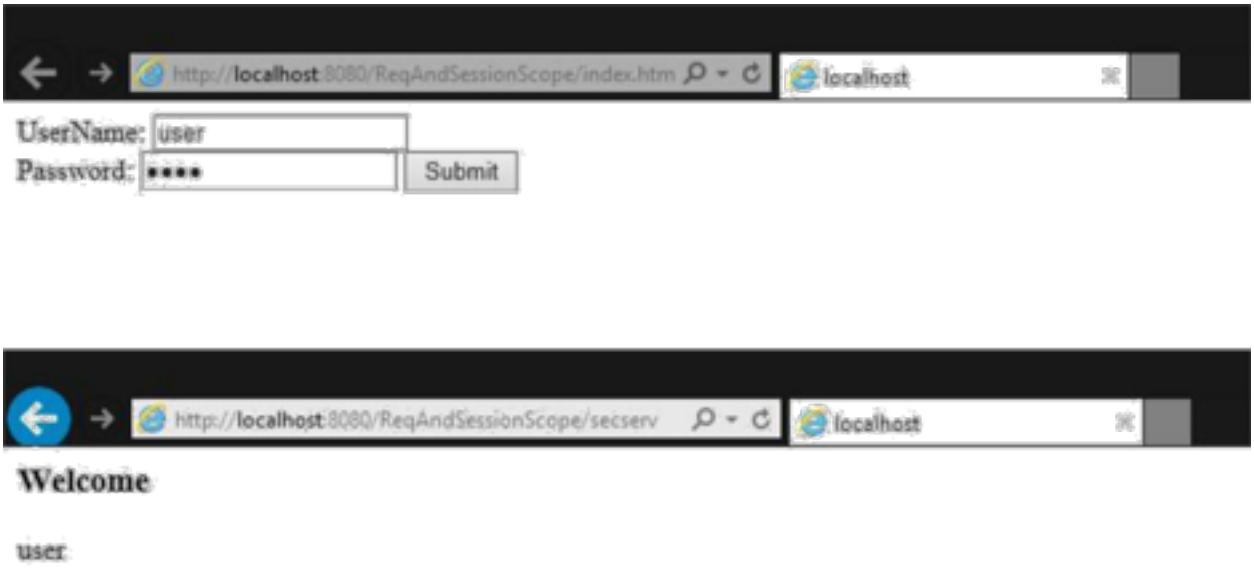
```
            String str2 = request.getParameter("t2");
        String str1 = request.getParameter("t1");

    session.setAttribute("Pass", str2);
        session.setAttribute("User", str1);


            response.sendRedirect("secserv");
        processRequest(request, response);
            }
            @Override
            protected void doPost(HttpServletRequest request, HttpServletResponse response)
                    throws ServletException, IOException {
        processRequest(request, response);
            }
            @Override
            public String getServletInfo() {
        return "Short description";
            }
        }
```

secserv.java package

```
        SenRedirect;


        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
        secserv extends HttpServlet {
            protected void processRequest(HttpServletRequest request,
        HttpServletResponse response) throws ServletException,
        IOException { response.setContentType("text/
        html;charset=UTF-8"); try (PrintWriter out =
        response.getWriter()) { out.println("<!DOCTYPE html>");
        out.println("<html>"); out.println("<head>");
                    out.println("<title>Servlet secserv</title>");
        out.println("</head>");                    out.println("<body>");
                    out.println("<h1>Servlet secserv at " + request.getContextPath() + "</h1>");
                    out.println("</body>");
        out.println("</html>");
                }
            }
            @Override
            protected void doGet(HttpServletRequest request, HttpServletResponse response)
```

```
            throws ServletException, IOException {
PrintWriter pw=response.getWriter();
        HttpSession session = request.getSession();
        String str1 = session.getAttribute("User").toString(); String
str2 = session.getAttribute("Pass").toString(); if(str2.equals("Sana"))
    {
    pw.println("<h3> Welcome </h3>" + str1);

    }
 else
    {
        pw.println("<h3>Invalid User</h3>");
response.sendRedirect("index.html");
    }
        //processRequest(request, response);
    }
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
processRequest(request, response);
    }
    @Override
    public String getServletInfo() {
return "Short description";
    }
}
```

Output:



Result: The given program has been compiled and executed successfully.

Practical No. 9

Objective: Session management using:

- HttpSession object
- URL rewriting
- Cookies
- Hidden Form fields

Theory:

Session is a conversional state between client and server and it can consists of multiple request and response between client and server. Since HTTP and Web Server both are stateless, the only way to maintain a session is when some unique information about the session (session id) is passed between server and client in every request and response. Session Management in Java Servlet Web Applications is a very interesting topic. Session in Java Servlet are managed through different ways, such as Cookies, HttpSession API, URL rewriting etc.

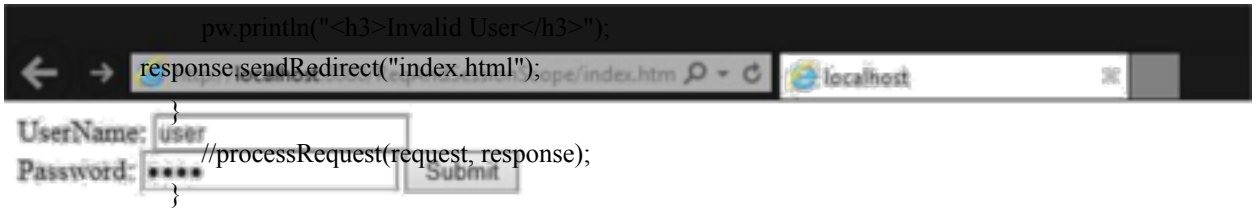
Source Code:

HttpSession Object:

SendRed.java package

```
SenRedirect;

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 SendRed extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        HttpSession session = request.getSession();
        String str2 = request.getParameter("t2");
String str1 = request.getParameter("t1");
session.setAttribute("Pass", str2);
session.setAttribute("User", str1);
        response.sendRedirect("secserv");
processRequest(request, response);
    }
} secserv.java
package SenRedirect;

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 secserv extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
PrintWriter pw=response.getWriter();
        HttpSession session = request.getSession();
        String str1 = session.getAttribute("User").toString(); String
str2 = session.getAttribute("Pass").toString(); if(str2.equals("Sana"))
    {
    pw.println("<h3> Welcome </h3>" + str1);
    }
Else
    {
        pw.println("<h3>Invalid User</h3>");
response.sendRedirect("index.html");
    }
        //processRequest(request, response);
    }
}
```



Output:

URL Rewriting

FirstServlet.java import

```
java.io.*; import
javax.servlet.*; import
javax.servlet.http.*;

public class FirstServlet extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse response){ try{
                response.setContentType("text/html");
                PrintWriter out = response.getWriter();
                String n=request.getParameter("userName");
                //out.print("Welcome "+n);
                out.print("<a href='servlet2?uname="+n+"'>visit</a>");
                out.close();
            }catch(Exception e){System.out.println(e);}
        }
}
```

SecondSevlet.java import

```
java.io.*; import
javax.servlet.*; import
javax.servlet.http.*;

public class SecondServlet extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse response){ try{
                response.setContentType("text/html");
                PrintWriter out = response.getWriter();
        String n=request.getParameter("uname");
out.print("Hello "+n); out.close(); }catch(Exception e)
            {System.out.println(e);}
        }
}
```

Output:

Cookies:

FirstServlet.java import

```
java.io.*; import
javax.servlet.*; import
javax.servlet.http.*;

public class FirstServlet extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse response){ try{

        response.setContentType("text/html"); PrintWriter
out = response.getWriter(); String
n=request.getParameter("userName"); out.print("Welcome
"+n);
Cookie ck=new Cookie("uname",n);//creating cookie object
response.addCookie(ck);//adding cookie in the response
out.print("<form action='servlet2' method='post'>");
out.print("<input type='submit' value='go'>");          out.print("</form>");

        out.close();
```

```
        }catch(Exception e){System.out.println(e);}
    }
}
```

SecondServlet.java import

```
java.io.*; import
javax.servlet.*; import
javax.servlet.http.*;

public class SecondServlet extends HttpServlet {
public void doPost(HttpServletRequest request, HttpServletResponse response){ try{
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
    String str = null;
        Cookie ck[]=request.getCookies();
for(Cookie c : ck)
    {
      if( c.getName().equals("uname"))
          {
            str = c.getValue();
          }
    }
    out.println("Welcome" + str);
    }catch(Exception e){System.out.println(e);}
        }
}
```

Output:

Hidden Form Fields:

FirstServlet1.java import

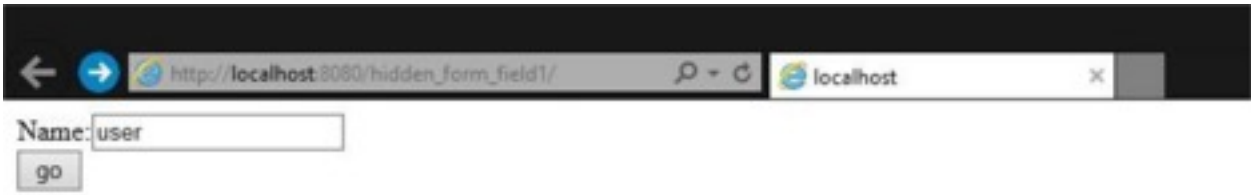
```
java.io.*; import
javax.servlet.*; import
javax.servlet.http.*;

public class FirstServlet1 extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse response){ try{
                response.setContentType("text/html");
                PrintWriter out = response.getWriter();
```

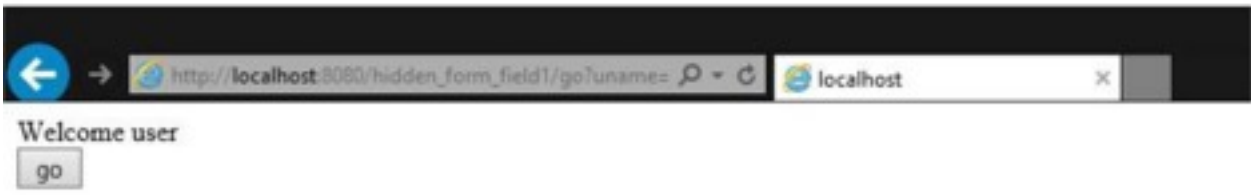
```
String n=request.getParameter("uname");
out.print("Welcome "+n);
                    out.print("<form action='welcome'>");
                    out.print("<input type='hidden' name='uname'
value='"+n+"'>");                    out.print("<input type='submit' value='go'>");
                                        out.print("</form>");  out.close();
                    }catch(Exception e){System.out.println(e);}
            }
}
```

SecondServlet.java import

```
java.io.*; import
javax.servlet.*; import
javax.servlet.http.*;

public class SecondServlet extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse response){ try{
                    response.setContentType("text/html");
                    PrintWriter out = response.getWriter();
                    String n=request.getParameter("uname");
                    out.print("Hello "+n);
out.close();
                }catch(Exception e){System.out.println(e);}
            }
}
```

Output:





Result: The given program has been compiled and executed successfully.

Practical No. 10

Objective: Making a login application in JSP and verify the credentials on **input.jsp** using a table in the MySQL database. On successful login the user must be directed to **welcome.jsp**. On **welcome.jsp**, he is shown a welcome message and a **log out button**. On clicking on logout button he must be redirected to **loout.jsp** where he must be shown Logout message and his session should end. Also he should be **given with a link to be redirected to the input.jsp page.**

Theory:

Java Server Pages(JSP) is a server-side programming technology that enables the creation of dynamic, platform-independent method for building Web-based applications. *JSP* have access to the entire family of Java APIs, including the JDBC API to access enterprise databases.

Source Code:

index.html

```
<html>
    <head>
        <title>TODO supply a title</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
    <body><form action="input.jsp">
        UserName:<input type="text" name="User" value="" /><br>
        Password:<input type="password" name="Pass" value="" /><br>
        <input type="submit" value="LOGIN" />
        </form>
    </body>
</html>
```

input.jsp

```
<%--
    Document      : newjsp
    Created on : Feb 12, 2018, 2:52:57 PM
    Author       : DEll
--%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"> <title>JSP Page</title>
    </head>
    <body>
        <%@page import="java.sql.*" %>
        <%
            Connection con=null;
            PreparedStatement ps =null;
            ResultSet rs = null;
```

```
String DriverName = "com.mysql.jdbc.Driver";
String url="jdbc:mysql://localhost:3306/user";
String user="root";
String pass="sana";
String str1 = request.getParameter("User");
String str2 = request.getParameter("Pass");
System.out.println(str1);
System.out.println(str2);              try{


    Class.forName(DriverName);
  con = DriverManager.getConnection(url, user, pass);
  String sql = ("select * from login where USERNAME=? and PASS=?"); ps =
  con.prepareStatement(sql);
ps.setString(1,str1);
ps.setString(2,str2);
ps.executeQuery();
if(rs.next()==true)
  {
  session.setAttribute("User", str1);
response.sendRedirect("welcome.jsp");
  } else{ %>
  <h1>Invalid User</h1> <jsp:include
  page="index.html"/>
    <% }
ps.close();
con.close();
  }
  catch (SQLException e)
      { System.out.println(e);
  }
%>
  </body>
</html>
```

welcome.jsp

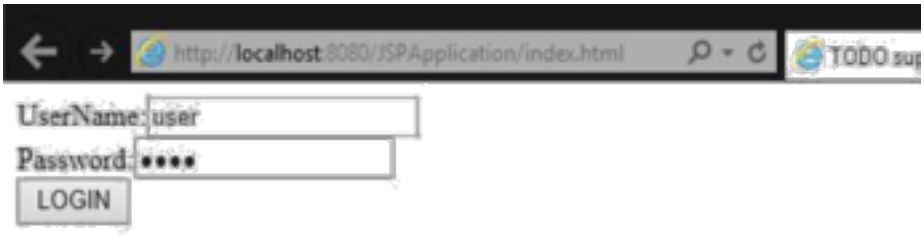
```
<%--
  Document      : welcome
  Created on : Jan 31, 2018, 8:48:20 AM
  Author       : DEll
--%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"> <title>JSP
    Page</title>
  </head>
```

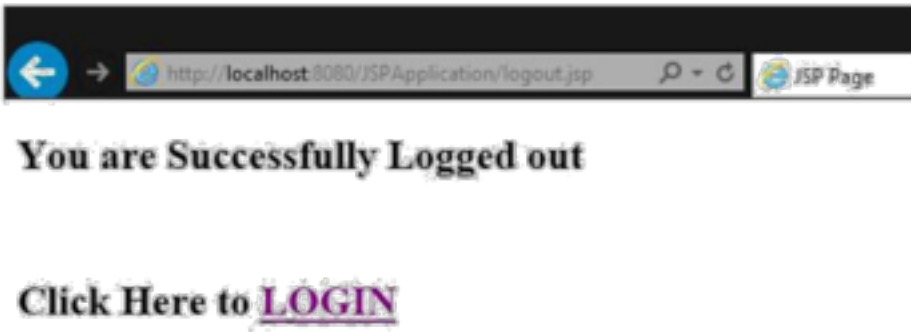
```
    <body>
        <form action="logout.jsp">
 WELCOME...
        <%= (String)session.getAttribute("User") %> <br/> <input
        type="submit" value="Logout" /> </form>
    </body>
</html>
```

logout.jsp

```
<%--
    Document      : logout
    Created on : Feb 11, 2018, 10:47:37 PM
    Author        : DEll
--%>


<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"> <title>JSP
        Page</title>
    </head>
    <body>
<%
          session.invalidate();
        %>
        <h2> You are Successfully Logged out</h2><br>
        <h2> Click Here to <a href ="index.html">LOGIN</a></h2> </body>
</html>
```

Output:





Result: The given program has been compiled and executed successfully.

Practical No. 11

Objective: Make a JSP application which should connect to the table in the MySQL database and it should insert, update and delete data in this table.

Thoery:

MySQL Enterprise Edition includes the most comprehensive set of advanced features, management tools and technical support to achieve the highest levels of MySQL scalability, security, reliability, and uptime. It reduces the risk, cost, and complexity in developing, deploying, and managing business-critical MySQL applications.

Source Code:

index.html

```
<html>
   <body>
      <form method="post" action="insert.jsp">
         Name:<br>
         <input type="text" name="name">
         <br>
         Branch:<br>
         <input type="text" name="branch">
         <br>
         Percentage:<br>
         <input type="text" name="percentage">


         <br>
         Email Id:<br>
         <input type="email" name="email">
         <br><br>
         <input type="submit" value="insert">
      </form>


      <form method="post" action="update.jsp">
         Name:<br>
         <input type="text" name="name">
         <br>
         Branch:<br>
         <input type="text" name="branch">
         <br>
         Percentage:<br>
         <input type="text" name="percentage">
         <br>
         Email Id:<br>
         <input type="email" name="email">
         <br><br>
```

```
                    <input type="submit" value="update">        </form>


            <form method="post" action="delete.jsp">
                Name:<br>
                <input type="text" name="name">
                <br><br>
                <input type="submit" value="delete">
            </form>
        </body>
    </html>
```

insert.jsp

```
    <%@page import="java.sql.SQLException"%>
    <%@page import="java.sql.ResultSet"%>
    <%@page import="java.sql.DriverManager"%>
    <%@page import="java.sql.PreparedStatement"%>
    <%@page import="java.sql.Connection"%>
    <%!String name;%>
    <%!String branch;%>
    <%!String percentage;%>



    <%!String email;%>
    <%!String JDBC_Driver_1 = "com.mysql.jdbc.Driver";%> <%!String
DB_URL = "jdbc:mysql://localhost/test";%> <%!String USER =
"root";%> <%!String PASS = "srijan";%>
    <%!Connection conn;%>
    <%!PreparedStatement ps;%>
    <%
        name=request.getParameter("name");
    branch=request.getParameter("branch");
    percentage=request.getParameter("percentage");
    email=request.getParameter("email");


        String JDBC_Driver_1 = "com.mysql.jdbc.Driver";
        String DB_URL = "jdbc:mysql://localhost/test";
        String USER = "root";
        String PASS = "srijan";
        Connection conn;
        PreparedStatement ps;
```

```
    try
{
try {

            Class.forName(JDBC_Driver_1);
        }
        catch(ClassNotFoundException e) {
            System.out.println(e);

        }
        System.out.println("Connection to a selected Database");
conn = DriverManager.getConnection(DB_URL,USER,PASS);
        System.out.println("Connected database successfully");
        System.out.println("Creating a statement");
        String sql = "insert
test.student(Name,Branch,Percentage,Email)"+"values(?,?,?,?)";
ps = conn.prepareStatement(sql); ps.setString(1,name);
ps.setString(2,branch); ps.setString(3,percentage);
ps.setString(4,email); int i = ps.executeUpdate();
        System.out.println("No. of rows affected: "+i);
ps.close(); conn.close();


    }
    catch(SQLException e) {
        System.out.println(e);
    }
%>
Inserted<br>
<button onclick="goBack()">Go Back</button>
<script> function
goBack() {
window.history.back();
}
</script>
```

into

update.jsp

```
<%@page import="java.sql.SQLException"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.DriverManager"%>
            <%@page import="java.sql.PreparedStatement"%>
<%@page import="java.sql.Connection"%>
<%!String name;%>
```

```
<%!String branch;%>
<%!String percentage;%>
<%!String email;%>
<%!String JDBC_Driver_1 = "com.mysql.jdbc.Driver";%> <%!String
DB_URL = "jdbc:mysql://localhost/test";%> <%!String USER =
"root";%> <%!String PASS = "srijan";%>
<%!Connection conn;%>
<%!PreparedStatement ps;%>
<%
    name=request.getParameter("name");
branch=request.getParameter("branch");
percentage=request.getParameter("percentage");
email=request.getParameter("email");
    String JDBC_Driver_1 = "com.mysql.jdbc.Driver";
    String DB_URL = "jdbc:mysql://localhost/test";
    String USER = "root";
    String PASS = "srijan";
    Connection conn;
    PreparedStatement ps;

    try {
try {
            Class.forName(JDBC_Driver_1);
        }
        catch(ClassNotFoundException e) {
            System.out.println(e);
        }
                System.out.println("Connection to a selected Database");
conn = DriverManager.getConnection(DB_URL,USER,PASS);
        System.out.println("Connected database successfully");
        System.out.println("Creating a statement");
        String sql = "UPDATE test.student SET branch=?, percentage=?,email=? WHERE
name=?;";
        ps = conn.prepareStatement(sql);


ps.setString(4,name);
ps.setString(1,branch);
ps.setString(2,percentage);
ps.setString(3,email);
ps.executeUpdate();




                                        int i =
        System.out.println("No. of rows affected: "+i);
ps.close();
        conn.close();
    }
    catch(SQLException e) {
        System.out.println(e);
```

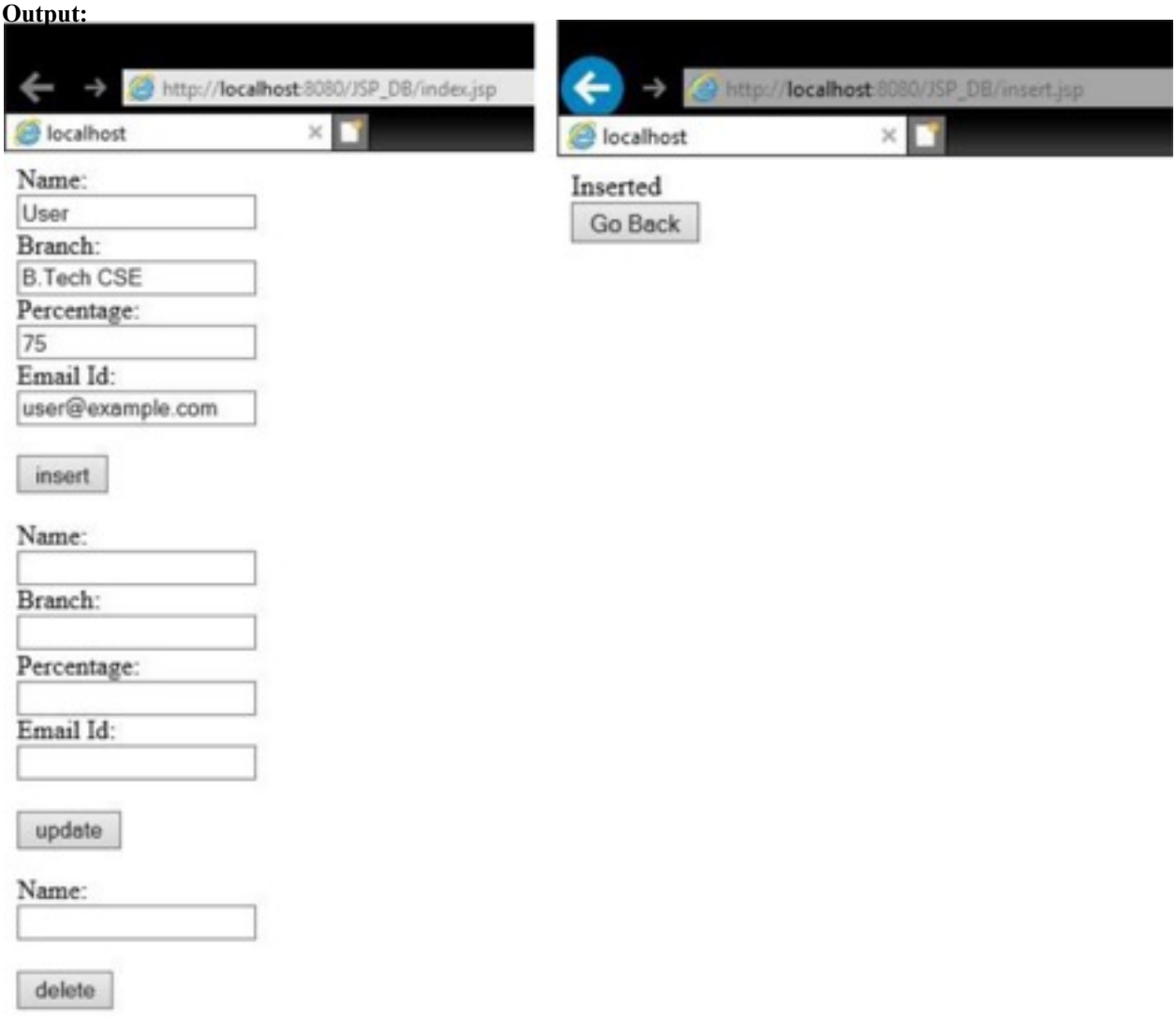
```
    }
%>
Updated<br>
<button onclick="goBack()">Go Back</button>
<script> function
goBack() {
window.history.back();
}
</script>
```

delete.jsp

```
<%@page import="java.sql.SQLException"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.DriverManager"%>


<%@page import="java.sql.PreparedStatement"%> <
%@page import="java.sql.Connection"%> <%!String name;
%>
<%!String JDBC_Driver_1 = "com.mysql.jdbc.Driver";%> <%!String
DB_URL = "jdbc:mysql://localhost/test";%> <%!String USER =
"root";%> <%!String PASS = "srijan";%>
<%!Connection conn;%>
<%!PreparedStatement ps;%>
<%
    name=request.getParameter("name");
    String JDBC_Driver_1 = "com.mysql.jdbc.Driver";
    String DB_URL = "jdbc:mysql://localhost/test";
    String USER = "root";
    String PASS = "srijan";
    Connection conn;
PreparedStatement ps;
    try {
try {
            Class.forName(JDBC_Driver_1);


        }
        catch(ClassNotFoundException e) {
            System.out.println(e);
        }
              System.out.println("Connection to a selected Database");
                conn = DriverManager.getConnection(DB_URL,USER,PASS);
              System.out.println("Connected database successfully");
       System.out.println("Creating a statement"); String sql =
"delete from test.student where Name=?"; ps =
conn.prepareStatement(sql);
ps.setString(1,name); int i = ps.executeUpdate();
System.out.println("No. of rows affected: "+i); ps.close();
conn.close(); }
```

```
    catch(SQLException e) {
        System.out.println(e);
    }
%>
Deleted<br>
<button onclick="goBack()">Go Back</button>

<script>
function goBack() {
    window.history.back();
}</script>
```

Output:



Result: The given program has been compiled and executed successfully.

Practical No. 12

Objective: Demonstrate the use of JSP actions: forward, include, useBean, setproperty and getproperty.

Thoery:

These actions use constructs in XML syntax to control the behavior of the servlet engine. You can dynamically insert a file, reuse JavaBeans components, forward the user to another page, or generate HTML for the Java plugin.

There is only one syntax for the Action element, as it conforms to the XML standard −

<jsp:action_name attribute = "value" />

Source Code:

forward and include:

index.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>TODO supply a title</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <form action="Home" method="post">
            <h2>     Enter your date of birth</h2>
            <h3> Date: <input type="text" name="date" value="" /></h3>
            <input type="submit" value="Submit" />
            <input type="reset" value="Reset" />


        </form>
    </body></html>
```

Home.jsp

```
<%--
    Document      : Home
    Created on : Feb 19, 2018, 8:24:14 PM
    Author        : DEll
--%>


<%@page contentType="text/html" pageEncoding="UTF-8"%> <%@page
import="java.util.Calendar"%> <%@page
import="java.util.GregorianCalendar"%>

<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"> <title>JSP
        Page</title>
    </head>
    <body>
```

```
    <center><h1>Happy Birthday</h1></center> <%
Calendar cal=new GregorianCalendar(); int day =
cal.get(Calendar.DAY_OF_MONTH);

    %>

    <% String date = request.getParameter("date");%> <% int d
    = Integer.parseInt(date);%>

    <% if(day==d){%>

    <jsp:include page="Date.jsp"/>

    <%} else {%>

    <jsp:forward page="Date.jsp"/>

    <%}%>

    </body>

</html>
```

Date.jsp

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

<!DOCTYPE html>

<html>

    <head>

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

    </head>

    <body>

    <center><h1>Welcome to the Home Page</h1></center> </body>

</html>
```

Output:

useBean, setproperty and getproperty:

newhtml.html

```
<html>



    <body>

        <form action="values.jsp">

            Name:<input type="text" name="sname" value="" /><br>
            RollNum:<input type="text" name="rollno" value="" /><br> <input
            type="submit" value="Submit" />

        </form>

    </body>

</html>
```

values.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%> <%@page
import="com.sana.student"%> <!DOCTYPE html>

<html>
```

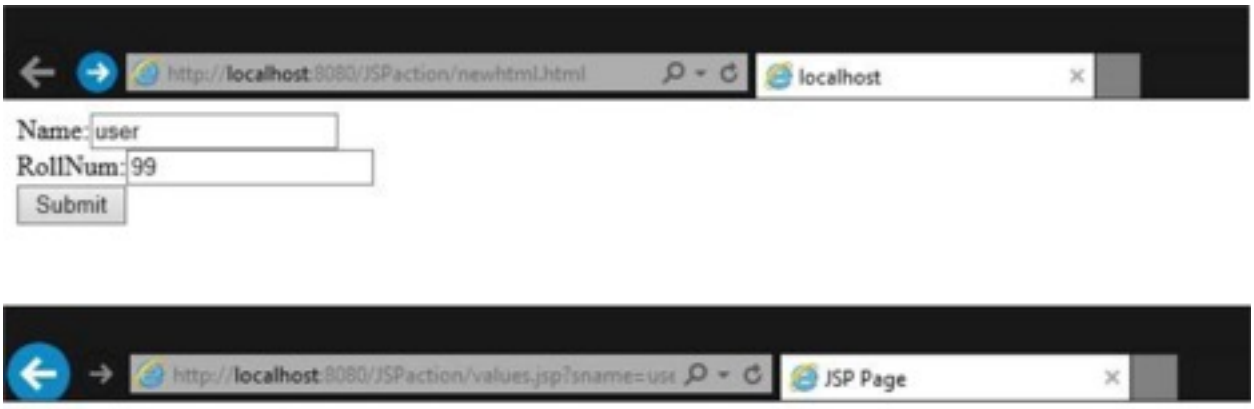
```
    <body>
        <% /*student st=new student();
        st.setRollno(Integer.parseInt(request.getParameter("rollno")));
    st.setName(request.getParameter("sname")); session.setAttribute("stu",st);*/ %>

            <jsp:useBean id="stu1" class="com.sana.student" scope="session"> </
            jsp:useBean>
        <jsp:setProperty name="stu1" property="rollno" value='<
%=Integer.parseInt(request.getParameter("rollno"))%>'/>
        <jsp:setProperty name="stu1" property="name"
    value='<%=request.getParameter("sname")%>'/>
       <%
        RequestDispatcher rd=request.getRequestDispatcher("display.jsp");
    rd.forward(request, response);


        %>
    </body>
</html>
```

display.jsp

```
<%--
    Document      : display
    Created on : Feb 19, 2018, 11:02:10 PM
    Author        : DEll
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%> <%@page
import="com.sana.student"%> <!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"> <title>JSP
        Page</title>
    </head>
    <body>
        <%
       /* student st=(student)session.getAttribute("stu");
    out.println("Name="+st.name); out.println("Rollno="+st.rollno); */

        %>
        <jsp:useBean id="stu1" class="com.sana.student" scope="session"> </
            jsp:useBean>
```

```
            <jsp:getProperty name="stu1" property="rollno"/>
            <jsp:getProperty name="stu1" property="name"/>
        </body>
    </html>
```

Output:



Result: The given program has been compiled and executed successfully.

Practical No. 13

Objective: Count the number of user hits on a JSP page using JSP session and application implicit objects.

Theory:

The interactive time between Web client and Web server on a single connection is known as session. That is, difference of time between connection established time and connection broken time is known as session.

Source Code:

Index.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <%Integer i = (Integer)application.getAttribute("count"); %>
        <% if (i==null || i==0)

    {
                    i=1;          }

else

    {
                    i++;          }




        out.println("You have visited the page:"+ i+"times");
application.setAttribute("count",i);
        %>
    </body>
</html>
```

Output:

1

2

3

Result: The given program has been compiled and executed successfully.

Practical No. 14

Objective: Design an error page in JSP and this page must be called whenever there is some error in the main.jsp. Use exception implicit object to display the details of error occurred.

Theory:

The exception is normally an object that is thrown at runtime. Exception Handling is the process to handle the runtime errors. There may occur exception any time in your web application. So handling exceptions is a safer side for the web developer. In JSP, there are two ways to perform exception handling:

1. By errorPage and isErrorPage attributes of page directive 2. By <error-page> element in web.xml file

Source Code:

index.jsp

```
<form action="process.jsp">
No1:<input type="text" name="n1" /><br/><br/>
No1:<input type="text" name="n2" /><br/><br/>
<input type="submit" value="divide"/>
</form>
```

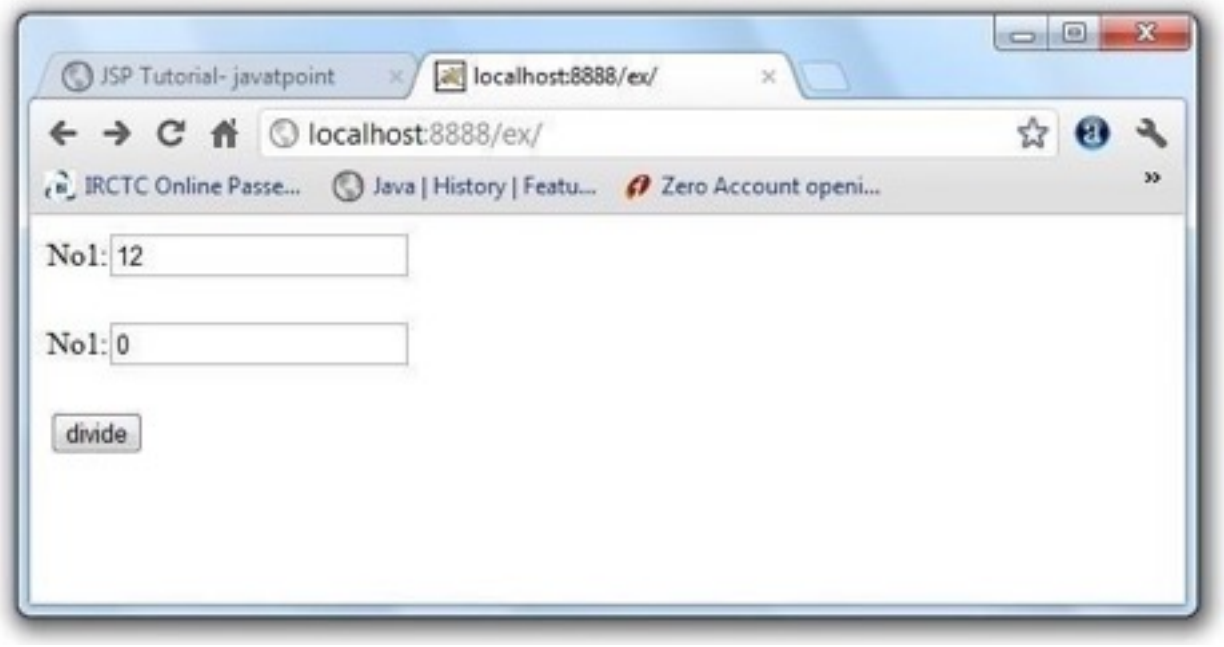
process.jsp

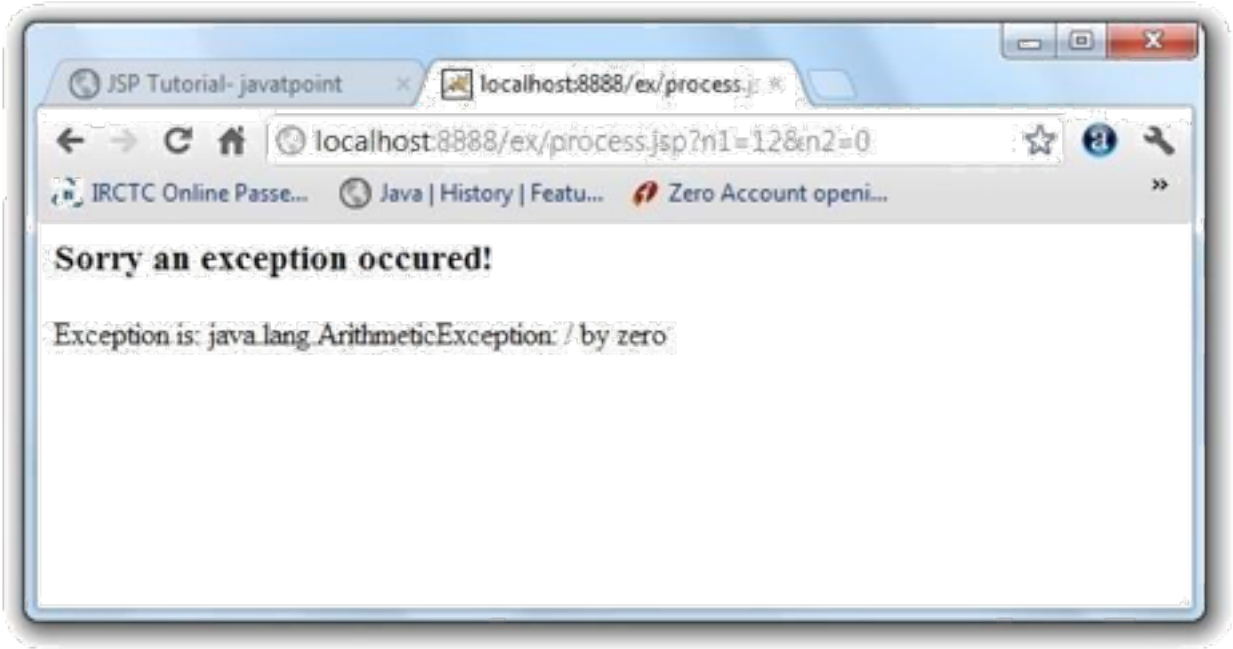
```
<%@ page errorPage="error.jsp" %>
<%
String num1=request.getParameter("n1");
String num2=request.getParameter("n2");
int a=Integer.parseInt(num1);          int
b=Integer.parseInt(num2);          int c=a/b;
out.print("division of numbers is: "+c);
%>
```

error.jsp

```
<%@ page isErrorPage="true" %>
<h3>Sorry an exception occured!</h3> Exception is: <%=
exception %>
```

Output:





Result: The given program has been compiled and executed successfully.

Practical No. 15

Objective: Demonstrate the use of Stateless session beans using RMI.

Theory:

Stateless Session bean *is a business object that represents business logic* only. It doesn't have state (data).In other words, *conversational state* between multiple method calls is not maintained by the container in case of stateless session bean. The stateless bean objects are pooled by the EJB container to service the request on demand. It can be accessed by one client at a time. In case of concurrent access, EJB container routes each request to different instance.

Source Code:

SquareEJB-ejb(SquareSessionBean.java)

```
package squareejbdemo; import
javax.ejb.Stateless;
@Stateless
public class SquareSessionBean implements SquareSessionBeanRemote,
SquareSessionBeanLocal {
    @Override
    public int getSquare(int var) {
return (var*var);
    }
}
```

SquareEJB-war(square.java) import

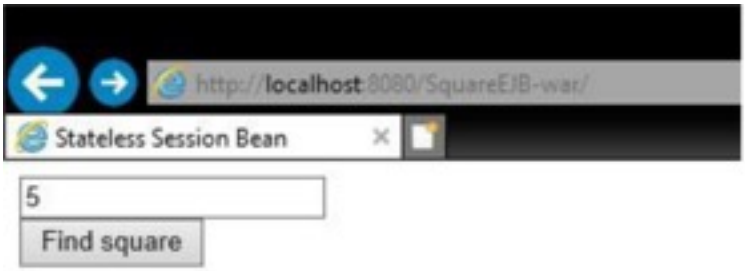
```
java.io.IOException; import
java.io.PrintWriter; import
javax.ejb.EJB;


import javax.servlet.ServletException; import
javax.servlet.http.HttpServlet; import
javax.servlet.http.HttpServletRequest; import
javax.servlet.http.HttpServletResponse;
import
squareejbdemo.SquareSessionBeanRemote;

public class square extends HttpServlet {

    @EJB
    private SquareSessionBeanRemote squareSessionBean;
protected void processRequest(HttpServletRequest request,
HttpServletResponse response)                    throws
ServletException, IOException {
response.setContentType("text/html;charset=UTF-8");
try (PrintWriter out = response.getWriter()) {                    int x =
Integer.parseInt(request.getParameter("t1"));
```

```
out.println("<!DOCTYPE html>");
out.println("<html>");                    out.println("<head>");
        out.println("<title>Servlet square</title>");
out.println("</head>");                   out.println("<body>");
        out.println("<h1>Square of " + x + " is " +
squareSessionBean.getSquare(x) + "</h1>");
out.println("</body>");                   out.println("</html>");
    }
  }
  @Override


  protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
processRequest(request, response);
  }
  @Override
  protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
processRequest(request, response);
  }
  @Override
  public String getServletInfo() {
return "Short description";
  }
}
```

index.html

```
<html>
  <head>
    <title>Stateless Session Bean</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0"> </head>
  <body>
    <form action ="square">
      <input type="text" name="t1">
      <br>


      <input type="submit" value="Find square">
    </form>
  </body></html>
```

Output:



Result: The given program has been compiled and executed successfully.

Practical No. 16

Objective: Demonstrate the use of Stateful session beans using RMI.

Theory:

Steps for creating a Stateful Session Bean using RMI

1. Create a **Java Class Library – "TutorialInterface**"
2. Create a **Java EE →EJB Module –"MyListBean**"
3. In "**MyListBean**" create a **session bean** --- create package **"com.sess.bean"→new sessionbean "TestBean.java"→ type=Stateful→ RemoteInteface→"TutorialInterface".**
4. This will create a remote interface with name "**TestBeanRemote" under the same package "com.sess.bean"** in TutorialInterface.
5. In the **TestBean** add business methods.
6. **Build the class library project "TutorialInterface**".
7. In the EJB Module- "MyListBean" create a **GlassFishDescriptor file**.
8. Inside the GlassFishDescriptor file select **EJB→TestBean→JNDI Name(Tutorial)**
9. **Deploy the EJB Module.**
10. Create a client application. **New project→add Java webapplication. Eg. EJBwebClient.**
11. In the libraryfolder **of EJBwebClient →add Project→TutorialInterface**.
12. Now in the **EJBwebcleint→ create a JSP webclient.jsp** 13. Inside this just write a scriptlet to define the **jspInit( ).**
14. Inside the **jspInit( ) initialize the InitialContext object and call the lookup( )** with this object.
15. Rest in the html part create a html form and inside that add a text field and two buttons, ADD and REM
16. The value entered in the textbox should be added or rem from the list by clicking on ADD or REM button.
17. The affected list must be shown below.
18. **Deploy this JSP(webclient.jsp). Then Run it.**

Source Code:

TutorialInterface (TestBeanRemote.java) package

```
com.sess.bean;

import java.util.List;
import javax.ejb.Remote;

@Remote
public interface TestBeanRemote {
void addElement(int e);          void
removeElement(int e);
   List getElement();
}
```

MyListBean (TestBean.java) package

```
com.sess.bean;
```

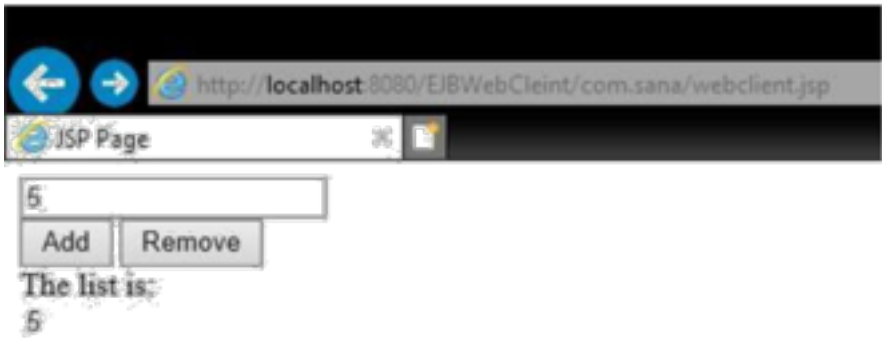
```
import java.util.ArrayList;
import java.util.List; import
javax.ejb.Stateful;

@Stateful
public class TestBean implements TestBeanRemote
    { List<Integer> values=new ArrayList(); @Override
    public void addElement(int e) {
values.add(e);

    }
    @Override
    public void removeElement(int e) {
Integer i = new Integer(e);
values.remove(i);

    }
    @Override
                public List
getElement() {
                    return
values;
    }
}
```

EJBWebCleint (webclient.jsp)

```
<%--
    Document      : webclient
    Created on : Feb 21, 2018, 2:19:49 AM
    Author       : DEll
--%>

<%@page import="javax.naming.Context"%>

<%@page import="java.util.List"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%> <%@page
import="javax.naming.InitialContext"%> <%@page
import="com.sess.bean.TestBeanRemote"%> <%@page
import="com.sess.bean.TestBean"%> <!DOCTYPE html> <%!




    private static TestBeanRemote r;

    public void jspInit(){
try{
   InitialContext ic=new InitialContext();
    r =(TestBeanRemote) ic.lookup("Tut");
```

```
}
catch(Exception e){
    System.out.println("Uninitialised object");
}
}
%>
<%
    if(request.getParameter("ADD")!=null) { int e =
Integer.parseInt(request.getParameter("t1")); out.println(e);
        r.addElement(e);
    }
     if(request.getParameter("REM")!=null)
     {
         int e = Integer.parseInt(request.getParameter("t1"));
r.removeElement(e);
     }
%>
<html>
    <head>
                            <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">



        <title>JSP Page</title>
    </head>
    <body>
        <form method="POST" action="webclient.jsp"> <input
            type="text" name="t1" value="" /><br> <input type="submit"
            value="Add" name="ADD"/>

            <input type="submit" value="Remove" name="REM"/><br>
        </form>
        The list is:
        <% if(r!=null){
            List<Integer> num=r.getElement();
for(int i : num){
out.println("<br>"+i);
            }
        }%>
    </body>
</html>
```

Output:



Result: The given program has been compiled and executed successfully.

Practical No. 17

Objective: Demonstrate the use of entity beans.

Theory:

An "Entity Bean" is a type of Enterprise JavaBean, a server-side Java EE component, that represents persistent data maintained in a database. An entity bean can manage its own persistence (Bean managed persistence) or can delegate this function to its EJB Container (Container managed persistence). An entity bean is identified by a primary key. If the container in which an entity bean is hosted crashes, the entity bean, its primary key, and any remote references survive the crash.

Source Code:

Index.jsp

```
<%@ page language="java" import="java.util.*" pageEncoding="ISO-88591"%>
<%String path = request.getContextPath();

String basePath =

request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()

+path+"/";

%>

<html><head>

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link rel="stylesheet"

href="http://code.jquery.com/mobile/1.3.2/jquery.mobile-1.3.2.min.css">

    <script src="http://code.jquery.com/jquery-1.9.1.min.js"></script>

    <script src="http://code.jquery.com/mobile/1.3.2/
jquery.mobile-1.3.2.min.js"></script>

    <base href="<%=basePath%>">

    <title>My JSP 'index.jsp' starting page</title> <meta http-
equiv="pragma" content="no-cache"> <meta http-equiv="cache-
control" content="no-cache"> <meta http-equiv="expires"
content="0">

<meta http-equiv="keywords" content="keyword1,keyword2,keyword3"> <meta http-
equiv="description" content="This is my page </head><body>

    <form action="way1.jsp">

 <table border="1">

 <tr>

 <td>Name</td>

  <td><input type="text"name="tname"></td</tr> <tr>
<td>Phone</td>
 <td><input type="text"name="telephone"></td>

 </tr><tr>

 <td>City</td>
 <td><input type="text" name="tcity" ></td></tr> <tr>

  <td colspan="2" align="center">

   <div data-role="fieldcontain">

   </div>

   <input type="submit" value="submit"></tr>

  </table></                </body></
  form>                        html>
```

Employee.java:

```
packag
e          pack;
```

```
public class Employee {                 private
String  name,city,phone;                 public
String getName() {          return name;}
public void setName(String name) {
this.name = name}
public String getCity() {
return city;}
public void setCity(String city) {
this.city = city;} public String
getPhone() { return phone;}
public void setPhone(String phone) {
this.phone = phone;}}  Way1.jsp:
<%@ page language="java" import="java.util.*" pageEncoding="ISO-88591"%> <%
String path = request.getContextPath();
String basePath =
request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()
+path+"/";
%>
<html><head>
    <base href="<%=basePath%>">
    <title>My JSP 'way1.jsp' starting page</title> <meta http-
   equiv="pragma" content="no-cache">
 <meta http-equiv="cache-control" content="no-cache"> <meta
 http-equiv="expires" content="0">
 <meta http-equiv="keywords" content="keyword1,keyword2,keyword3"> <meta http-
 equiv="description" content="This is my page">
 <!--<link rel="stylesheet" type="text/css" href="styles.css">--> </
head><body>
    <jsp:useBean id="emp" scope="page" class="pack.Employee"></jsp:useBean>
    <jsp:setProperty property="name" name="emp"value="<
%=request.getParameter("tname") %>"/>
    <jsp:setProperty property="phone" name="emp"value="<
%=request.getParameter("telephone") %>"/>
    <jsp:setProperty property="city" name="emp"value="<
%=request.getParameter("tcity") %>"/>
<hr> Name:<jsp:getProperty property="name" name="emp"/><br> phone:
<jsp:getProperty property="phone" name="emp"/><br> city: <jsp:getProperty
property="city" name="emp"/><br>
  </body>
</html>
```

Output:


Name
Phone
City
submit


192.168.1.6:8000/lb/
Name
ujwal
Phone
8005000099
City
ik
submit

Practical No. 18

Objective: Demonstrate a user login application using Struts.

Theory:

Apache Struts 2 is an elegant, extensible framework for creating enterprise-ready Java web applications. This framework is designed to streamline the full development cycle from building, to deploying and maintaining applications over time. Apache Struts 2 was originally known as Web Work 2.

Source Code:

Login.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>

        <form name="login" action="login.do">

            <input type="text" name="username" value=""/> <input
            type="text" name="email" value=""/>
            <input type="submit" name="submit" value="submit"/>

        </form>
    </body>
</html>
```

Success.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"> <title>JSP
        Page</title>
    </head>
    <body>
        Login Successful..!!
        Your Username is
        <bean:write name="LoginActionFormBean" property="username"/> <br> Your Email
        ID is
        <bean:write name="LoginActionFormBean" property="email"/> <br> </body>
</html>
```

Failure.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"> <title>JSP
        Page</title>
    </head>
    <body>
        Login Failed..!!
        Your Username is
        <bean:write name="LoginActionFormBean" property="username"/> <br> Your Email
        ID is
        <bean:write name="LoginActionFormBean" property="email"/> <br>
    </body>
</html>
```

LoginActionFormBean.java

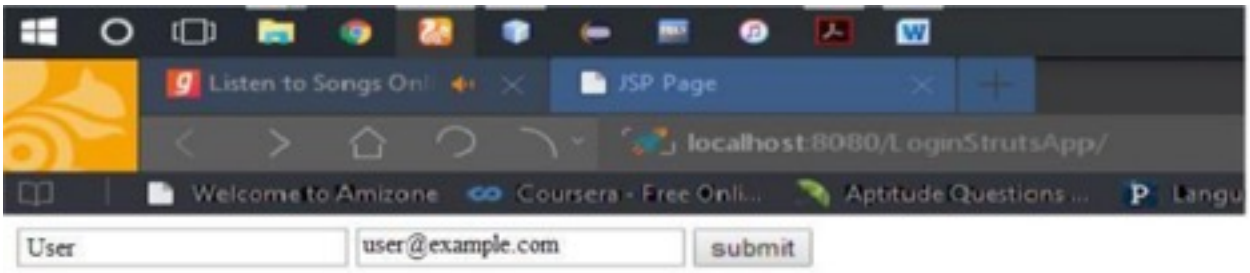
```
package com.myapp.struts;
import javax.servlet.http.HttpServletRequest; import
org.apache.struts.action.ActionErrors; import
org.apache.struts.action.ActionMapping; import
org.apache.struts.action.ActionMessage;
public class LoginActionFormBean extends org.apache.struts.action.ActionForm {


    private String name;
private int number;
                                private
String username;          private
String email;         public String
getUsername() {             return
username;
    }

    public void setUsername(String username) {
this.username = username;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
this.email = email;
    }
    public String getName() {
return name;
    }
```

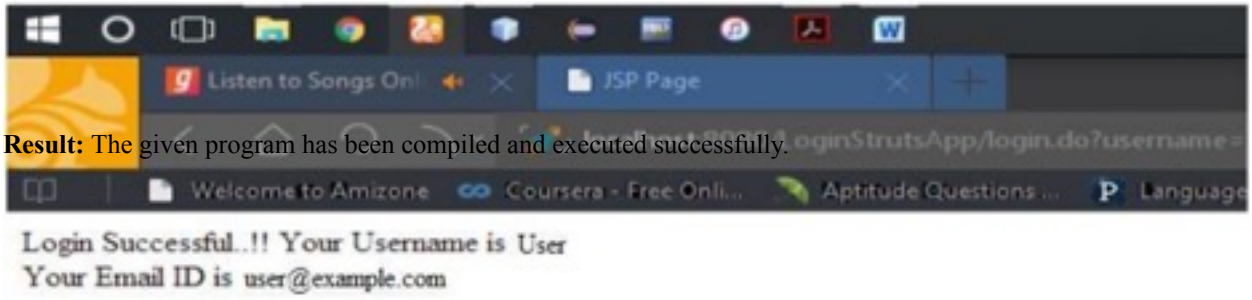
```
    public void setName(String string) {
name = string;
    }
    public int getNumber() {
return number;
    }
    public void setNumber(int i) {
number = i;
    }


    public LoginActionFormBean() {
        super(); }
     public ActionErrors validate(ActionMapping mapping, HttpServletRequest request) {
          ActionErrors errors = new ActionErrors(); if
(getName() == null || getName().length() < 1) {
              errors.add("name", new ActionMessage("error.name.required"));
          }
          return errors; }}
```

LoginAction.java package

```
com.myapp.struts;


import javax.servlet.http.HttpServletRequest; import
javax.servlet.http.HttpServletResponse; import
org.apache.struts.action.ActionForm; import
org.apache.struts.action.ActionForward; import
org.apache.struts.action.ActionMapping;


public class LoginAction extends org.apache.struts.action.Action { private static
final String SUCCESS = "success"; private static final String FAILURE =
"failure"; @Override
public ActionForward execute(ActionMapping mapping, ActionForm form,
HttpServletRequest request, HttpServletResponse response) throws Exception {


                LoginActionFormBean lfb=(LoginActionFormBean)form;

if(lfb.getUsername().equals("yash")&&lfb.getEmail().equals("yash.ojha@gmail.c om"))
                {
                        return mapping.findForward(SUCCESS);
                }
else
        {
            return mapping.findForward(FAILURE);
        }}}
```

Output:





Result: The given program has been compiled and executed successfully.