CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY

DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY AND RESEARCH

DEPARTMENT OF COMPUTER ENGINEERING

Subject Name: Advanced Java Programming **Semester**: 5th Sem

Subject Code: CE376

Academic year: 2020-21

Practical List

Sr. No	Practical Aim	Page No.
1	Create following table using mysql and perform following task. Database: ebookshop Table: books +-------+----------+----------+---------+-------+ | id | title | author | price | qty | | (INT) | (VARCHAR(50)) | (VARCHAR(50)) | (FLOAT) | (INT) | +-------+----------+----------+---------+-------+ | 1001 | Java for dummies | Tan Ah Teck | 11.11 | 11 | | 1002 | More Java for dummies | Tan Ah Teck | 22.22 | 22 | | 1003 | More Java for more dummies | Mohammad Ali | 33.33 | 33 | | 1004 | A Cup of Java | Kumar | 44.44 | 44 | | 1005 | A Teaspoon of Java | Kevin Jones | 55.55 | 55 | +-------+----------+----------+---------+-------+	3
	a. Fetch and display records from a table using field index	
	b. Fetch and display records from a table using Result set metadata.	
	c. Display database properties using Database metadata	
	d. Using prepared statement perform insert, update and delete operations.	
	e. Perform insert, update and delete using callable statement.	
	f. Perform commit and set auto commit.	

PRACTICAL-1

AIM: [Create following table using mysql and perform following task.]

Database: ebookshop
Table: books

id (INT)	title (VARCHAR(50))	author (VARCHAR(50))	price (FLOAT)	qty (INT)
1001	Java for dummies	Tan Ah Teck	11.11	11
1002	More Java for dummies	Tan Ah Teck	22.22	22
1003	More Java for more dummies	Mohammad Ali	33.33	33
1004	A Cup of Java	Kumar	44.44	44
1005	A Teaspoon of Java	Kevin Jones	55.55	55

a. Fetch and display records from a table using field index

b. Fetch and display records from a table using Result set metadata.

c. Display database properties using Database metadata

d. Using prepared statement perform insert, update and delete operations.

e. Perform insert, update and delete using callable statement.

f. Perform commit and set auto commit.

g. Display Scrollable Record Set

PROGRAM:

```
import java.sql.*;
import java.util.*;

public class first {
    public static void main(String[] args) throws Exception {
        int menu = 0;
        Scanner sr = new Scanner(System.in);

        try{
            Class.forName("com.mysql.jdbc.Driver");
            //MAKING CONNECTION TO DB
```

```
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/ebookshop",
"root", "");


        do{
            System.out.println("\nWHAT DO YOU WANT TO PERFORM?");
            System.out.println("1. Display Records");
            System.out.println("2. ResultSet Metadat");
            System.out.println("3. Database Metadata");
            System.out.println("4. Insert Record");
            System.out.println("5. Update Records");
            System.out.println("6. Delete Records");
            System.out.println("7. Callable Insert");
            System.out.println("8. Callable Update");
            System.out.println("9. Callable Delete");
            System.out.println("10. Commit And Auto-Commit");
            System.out.println("11. Scrollabe Record Set");
            System.out.println("0. Exit");
            menu = sr.nextInt();
            switch(menu){
                case 1:
                    //DISPLAYING RECORDS
                    Statement stmt = con.createStatement();
                    ResultSet rs = stmt.executeQuery("select * from books");
                    System.out.println("\nmsg: Displaying Table:");
                    while(rs.next()){
                        System.out.println(rs.getInt(1) + " | " + rs.getString(2) + " | " + rs.getString(3) + " |
" + rs.getFloat(4) + " | " + rs.getInt(5));
                    }
                break;
```

```
            case 2:
                Statement stmtm = con.createStatement();
                  //Retrieving the data
                  ResultSet rsm = stmtm.executeQuery("select * from books");
                  ResultSetMetaData rsMetaData = rsm.getMetaData();
                  //Number of columns
                  System.out.println("Number of columns: "+rsMetaData.getColumnCount());
                  //Column name
                  System.out.println("1st Column Name: "+rsMetaData.getColumnName(1));
                  //Name of Table
                  System.out.println("Table Name: "+rsMetaData.getTableName(1));
                  //Columns of Table
                  System.out.println("Total columns: "+rsMetaData.getColumnCount());
                  //Type of 1st column
                  System.out.println("1st Column Type: "+rsMetaData.getColumnTypeName(1));
                  break;

            case 3:
                //DATABASE METADATA
                DatabaseMetaData databaseMetaData = con.getMetaData();
                //Print TABLE_TYPE "TABLE"
                ResultSet resultSet = databaseMetaData.getTables(null, null, null, new
String[]{"TABLE"});
                System.out.println("\nPrinting TABLE_NAME:");
                while(resultSet.next()){
                    System.out.println(resultSet.getString("TABLE_NAME"));
                }
                System.out.println("\nDatabase Info: ");
```

```
                System.out.println("Driver Name: "+databaseMetaData.getDriverName());
                System.out.println("Driver Version: "+databaseMetaData.getDriverVersion());
                System.out.println("UserName: "+databaseMetaData.getUserName());
                System.out.println("Database Product Name:
"+databaseMetaData.getDatabaseProductName());
                System.out.println("Database Product Version:
"+databaseMetaData.getDatabaseProductVersion());


                ResultSet columns = databaseMetaData.getColumns(null,null, "books", null);
                System.out.println("\nPrinting COLUMN_INFO:");
                while(columns.next())
                {
                    String columnName = columns.getString("COLUMN_NAME");
                    String datatype = columns.getString("DATA_TYPE");
                    String columnsize = columns.getString("COLUMN_SIZE");
                    String decimaldigits = columns.getString("DECIMAL_DIGITS");
                    String isNullable = columns.getString("IS_NULLABLE");
                    String is_autoIncrment = columns.getString("IS_AUTOINCREMENT");
                    //Printing results
                    System.out.println("Column Name:" +columnName + "--- Datatype:" + datatype +
"--- Column Size" + columnsize + "--- Decimal Digits:" + decimaldigits + "--- isNullable:" +
isNullable + "--- Is autoIncrment:" + is_autoIncrment);
                }
            break;


            case 4:
                //INSERTING RECORDS


                int id, qty;
                float price;
```

```
            String title, author;

            System.out.println("\nInput Data for New Record:");
            System.out.println("id (int)");
            id = sr.nextInt();
            sr.nextLine();
            System.out.println("title (varchar)");
            title = sr.nextLine();
            System.out.println("author (varchar)");
            author = sr.nextLine();
            System.out.println("price (float)");
            price = sr.nextFloat();
            System.out.println("qty (int)");
            qty = sr.nextInt();

            PreparedStatement pstmt = con.prepareStatement("insert into books
values(?,?,?,?,?)");
            pstmt.setInt(1, id);
            pstmt.setString(2, title);
            pstmt.setString(3, author);
            pstmt.setFloat(4, price);
            pstmt.setInt(5, qty);

            int i = pstmt.executeUpdate();
            System.out.println("\nmsg: " + i + " records inserted\n");
        break;

        case 5:
            //UPDATING RECORDS
```

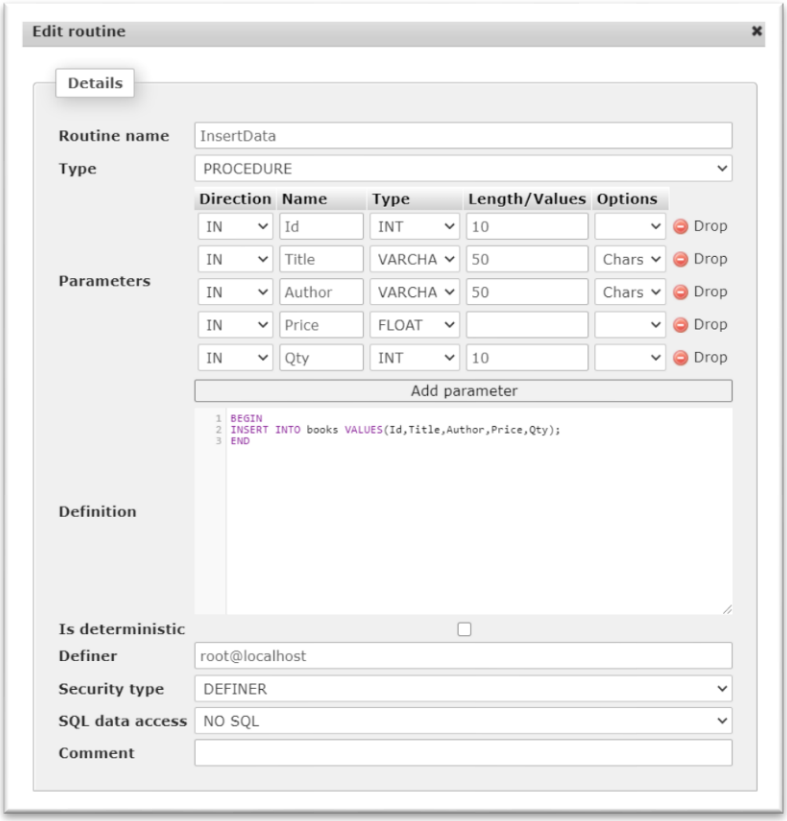
```
            System.out.println("\nInput Data to Update Records:");
            System.out.println("ID of record you want to Update");
            int id2 = sr.nextInt();
            sr.nextLine();
            System.out.println("Update Book title to ");
            String title2 = sr.nextLine();
            PreparedStatement ustmt = con.prepareStatement("UPDATE books SET title = ?
WHERE id = ?");

            ustmt.setString(1, title2);
            ustmt.setInt(2, id2);
            int rowAffected = ustmt.executeUpdate();

            System.out.println("\nmsg: "+rowAffected + " records updated.\n");
        break;

        case 6:
            //DELETING RECORDS
            System.out.println("\nInput Data to Delete Records:");
            System.out.println("ID of record you want to Delete");
            int id3 = sr.nextInt();
            PreparedStatement dstmt = con.prepareStatement("delete from books where id=?");

            dstmt.setInt(1, id3);
            int rowDeleted = dstmt.executeUpdate();

            System.out.println("\nmsg: "+rowDeleted + " records deleted.\n");
        break;
```

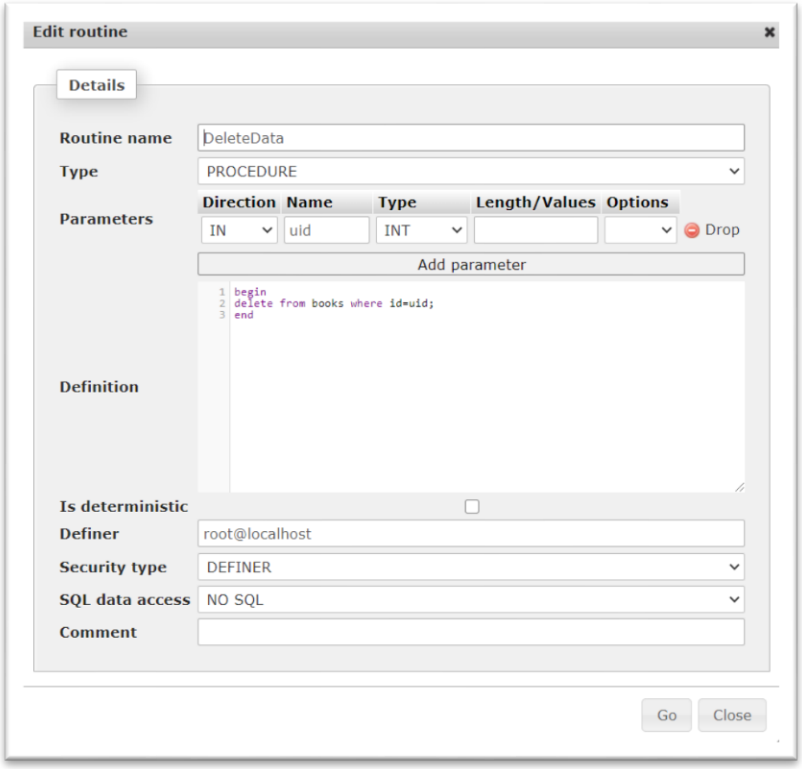
```
case 7:
    //CALLABLE INSERTING RECORDS
    CallableStatement stmti= con.prepareCall("{call InsertData(?,?,?,?,?)}");
    stmti.setInt(1, 1006);
    stmti.setString(2, "Advance Java");
    stmti.setString(3, "Shreyas Shah");
    stmti.setFloat(4, 77);
    stmti.setInt(5, 77);
    stmti.execute();
    System.out.println("Successeful Inserted");
   break;

case 8:
    //CALLABLE UPDATING RECORDS
    CallableStatement stmtu= con.prepareCall("{call UpdateData(?,?)}");
    stmtu.setInt(1, 1007);
    stmtu.setString(2, "Advance Java Programming");
    stmtu.execute();
    System.out.println("Successfully Updated");
    break;

case 9:
    //CALLABLE DELETING RECORDS
    CallableStatement stmtd= con.prepareCall("{call DeleteData(?)}");
    stmtd.setInt(1, 1006);
    stmtd.execute();
    System.out.println("Successfully Deleted");
    break;
```

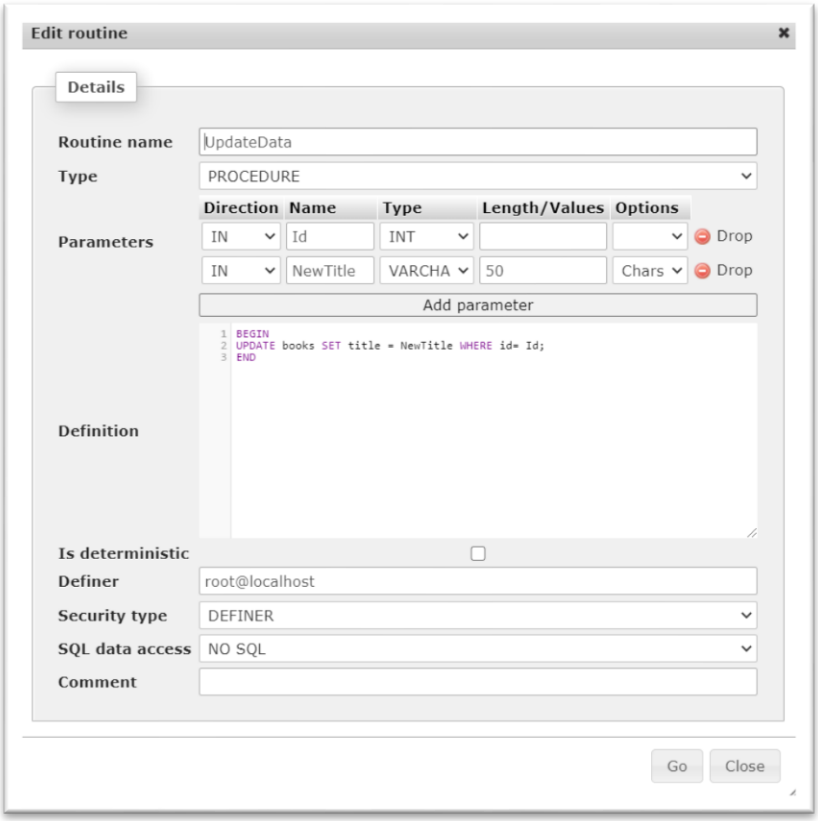
```
            case 10:
//COMMIT AND AUTO-COMMIT
                con.setAutoCommit(false);


                System.out.println("Type 1 to commit the querry");
                short flag=sr.nextShort();
                if (flag==1){
                  PreparedStatement comstnt = con.prepareStatement("insert into books
values(?,?,?,?,?)");
                  comstnt.setInt(1, 1006);
                comstnt.setString(2, "Advance Java");
                comstnt.setString(3, "Shreyas Shah");
                comstnt.setFloat(4, 77);
                comstnt.setInt(5, 77);
                comstnt.executeUpdate();
                System.out.println("Quesry Executed.");
                con.commit();
                System.out.println("Quesry Commited.");
                }
                 else{
                  System.out.println("Quesry Rollbacked.");
                  con.rollback();
                 }
                break;



            case 11:
                //SCROLLABLE RECORD TYPE
```

```
                Statement
str=con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_READ_O
NLY);
                ResultSet rsr = str.executeQuery("select * from books");


                rsr.first();
                System.out.println("First Record...");
                System.out.println(rsr.getInt(1) + "->"+ rsr.getString(2));
                rsr.absolute(3);
                System.out.println("Third Record...");
                System.out.println(rsr.getInt(1) + "->"+ rsr.getString(2));
                rsr.last();
                System.out.println("Last Record...");
                System.out.println(rsr.getInt(1) + "->"+ rsr.getString(2));
                rsr.previous();
                //rsr.relative(-1);
                System.out.println("Last to First Record...");
                System.out.println(rsr.getInt(1) + "->"+ rsr.getString(2));
                break;

            case 0:
                System.out.println("\nExiting...\n");
            break;

            default:
                System.out.println("\nWrong Input!\n");
            break;
        }
    } while(menu != 0);
```

```
        //CLOSING CONNECTION TO DB
        con.close();

    } catch(Exception e){
        System.out.println("\nError: " + e);
    }
  }
}
```




Edit routine
Details
Routine name
DeleteData
Type
PROCEDURE
Parameters
Direction Name Type Length/Values Options
IN uid INT Drop
Add parameter
Definition
begin
delete from books where id=uid;
end
Is deterministic
Definer
root@localhost
Security type
DEFINER
SQL data access
NO SQL
Comment
Go
Close


Edit routine
Details
Routine name
UpdateData
Type
PROCEDURE
Parameters
Direction Name Type Length/Values Options
IN Id INT Drop
IN NewTitle VARCHA 50 Chars Drop
Add parameter
Definition
BEGIN
UPDATE books SET title = NewTitle WHERE id= Id;
END
Is deterministic
Definer
root@localhost
Security type
DEFINER
SQL data access
NO SQL
Comment
Go
Close

OUTPUT:

id ▲ 1	title	author	price	qty
1001	Java for durries	Tan Ah Teck	11.11	11
1002	More java for durries	Tan Ah Teck	22.22	22
1003	More java for rore durries	Moharrad Ali	33.33	33
1004	A cup of java	Kurar	44.44	44
1005	A Teaspoon of java	Kevin Jones	55.55	55

```
WHAT DO YOU WANT TO PERFORM?
1. Display Records
2. ResultSet Metadat
3. Database Metadata
4. Insert Record
5. Update Records
6. Delete Records
7. Callable Insert
8. Callable Update
9. Callable Delete
10. Commit And Auto-Commit
11. Scrollabe Record Set
0. Exit
```

a. Fetch and display records from a table using field index:

```
1

msg: Displaying Table:
id | title | author | price | qty
1001 | Java for durries | Tan Ah Teck | 11.11 | 11
1002 | More java for durries | Tan Ah Teck | 22.22 | 22
1003 | More java for rore durries | Moharrad Ali | 33.33 | 33
1004 | A cup of java | Kurar | 44.44 | 44
1005 | A Teaspoon of java | Kevin Jones | 55.55 | 55
```

b. Fetch and display records from a table using Result set metadata:

```
2
Number of columns: 5
1st Column Name: id
Table Name: books
Total columns: 5
1st Column Type: INT
```

c. Display database properties using Database metadata:

```
s

Printing TABLE_NAME:
books

Database Info:
Driver Name: MySQL Connector Java
Driver Version: mysql-connector-java-5.1.49 ( Revision: ad86f36e100e104cd926c6b81c8cab9565750116 )
UserName: root@localhost
Database Product Name: MySQL
Database Product Version: 5.5.39

Printing COLUMN_INFO:
Column Name:id--- Datatype:4--- Column Size10--- Decimal Digits:0--- isNullable:NO--- Is autoIncrment:NO
Column Name:title--- Datatype:12--- Column Size50--- Decimal Digits:null--- isNullable:NO--- Is autoIncrment:NO
Column Name:author--- Datatype:12--- Column Size50--- Decimal Digits:null--- isNullable:NO--- Is autoIncrment:NO
Column Name:price--- Datatype:7--- Column Size12--- Decimal Digits:null--- isNullable:NO--- Is autoIncrment:NO
Column Name:qty--- Datatype:4--- Column Size10--- Decimal Digits:0--- isNullable:NO--- Is autoIncrment:NO
```

d. Using prepared statement perform insert, update and delete operations:

1)Insert:

```
4

Input Data for New Record:
id (int)
1006
title (varchar)
Advance Java
author (varchar)
John
price (float)
66.66
qty (int)
66

msg: 1 records inserted
```

id ▲ 1	title	author	price	qty
1001	Java for durries	Tan Ah Teck	11.11	11
1002	More java for durries	Tan Ah Teck	22.22	22
1003	More java for rore durries	Moharrad Ali	33.33	33
1004	A cup of java	Kurar	44.44	44
1005	A Teaspoon of java	Kevin Jones	55.55	55
1006	Advance Java	John	66.66	66

2)Update:

```
5

Input Data to Update Records:
ID of record you want to Update
1006
Jpdate Book title to
Advance Java Programming

msg: 1 records updated.
```

id ▲ 1	title	author	price	qty
1001	Java for durries	Tan Ah Teck	11.11	11
1002	More java for durries	Tan Ah Teck	22.22	22
1003	More java for rore durries	Moharrad Ali	33.33	33
1004	A cup of java	Kurar	44.44	44
1005	A Teaspoon of java	Kevin Jones	55.55	55
1006	Advance Java Programming	John	66.66	66

3)Delete:

```
6

Input Data to Delete Records:
ID of record you want to Delete
1006

msg: 1 records deleted.
```

id ▲ 1	title	author	price	qty
1001	Java for durries	Tan Ah Teck	11.11	11
1002	More java for durries	Tan Ah Teck	22.22	22
1003	More java for rore durries	Moharrad Ali	33.33	33
1004	A cup of java	Kurar	44.44	44
1005	A Teaspoon of java	Kevin Jones	55.55	55

e. Perform insert, update and delete using callable statement:

1)Insert:

```
7
Successeful Inserted
```

id ▲ 1	title	author	price	qty
1001	Java for durries	Tan Ah Teck	11.11	11
1002	More java for durries	Tan Ah Teck	22.22	22
1003	More java for rore durries	Moharrad Ali	33.33	33
1004	A cup of java	Kurar	44.44	44
1005	A Teaspoon of java	Kevin Jones	55.55	55
1006	Advance Java	Mark	66	66

2)Update:

```
3
Successfully Updated
```

id ▲ 1	title	author	price	qty
1001	Java for durries	Tan Ah Teck	11.11	11
1002	More Java for durries	Tan Ah Teck	22.22	22
1003	More Java for more durries	Moharrad Ali	33.33	33
1004	A cup of java	Kurar	44.44	44
1005	A Teaspoon of java	Kevin Jones	55.55	55
1006	Advance Java Programming	Mark	66	66

3)Delete:

```
9
Successfully Deleted
```

id ▲ 1	title	author	price	qty
1001	Java for durries	Tan Ah Teck	11.11	11
1002	More java for durries	Tan Ah Teck	22.22	22
1003	More java for rore durries	Moharrad Ali	33.33	33
1004	A cup of java	Kurar	44.44	44
1005	A Teaspoon of java	Kevin Jones	55.55	55

f. Perform commit and set auto commit:

```
10
Type 1 to commit the querry
1
Quesry Executed.
Quesry Commited.
```

id ▲ 1	title	author	price	qty
1001	Java for durries	Tan Ah Teck	11.11	11
1002	More Java for durries	Tan Ah Teck	22.22	22
1003	More java for more durries	Moharrad Ali	33.33	33
1004	A Cup of java	Kurar	44.44	44
1005	A Teaspoon of java	Kevin Jones	55.55	55
1006	Advance Java	Shreyas Shah	77	77

g. Display Scrollable Record Set:

```
11
First Record...
1001->Java for durries
Third Record...
1003->More java for more durries
Last Record...
1006->Advance Java
Last to First Record...
1005->A Teaspoon of java
```

CONCLUSION:

We studied about JDBC and how to connect MySQL with in and perform basic tasks on records of the table.

PRACTICAL-2

AIM: [Write a Servlet which prints a greeting message on Browser]

PROGRAM:

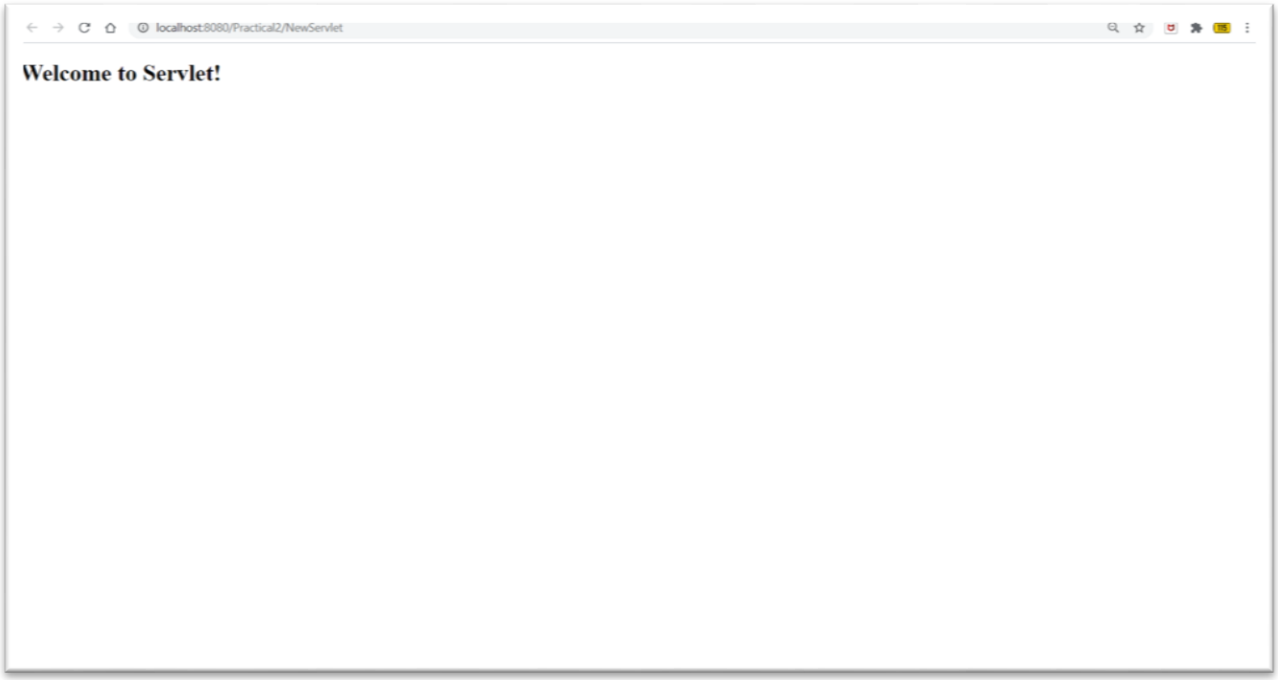
```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

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;

/**
 *
 * @author kashy
 */
public class NewServlet extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
```

```
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
            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>Welcome to Servlet!"+"</h1>");
            out.println("</body>");
            out.println("</html>");
        }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left
to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
```

```
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    processRequest(request, response);
}

/**
 * Handles the HTTP <code>POST</code> method.
 *
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    processRequest(request, response);
}

/**
 * Returns a short description of the servlet.
 *
 * @return a String containing servlet description
 */
@Override
```

```
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>


}
```

OUTPUT:



CONCLUSION:

In this practical we learnt how to develop a simple WebApplication project & write a Servlet which prints a greeting message on Browser.

PRACTICAL-3

AIM: [Write a Servlet which takes two numbers from client from HTML form and display addition of both the numbers]

PROGRAM:

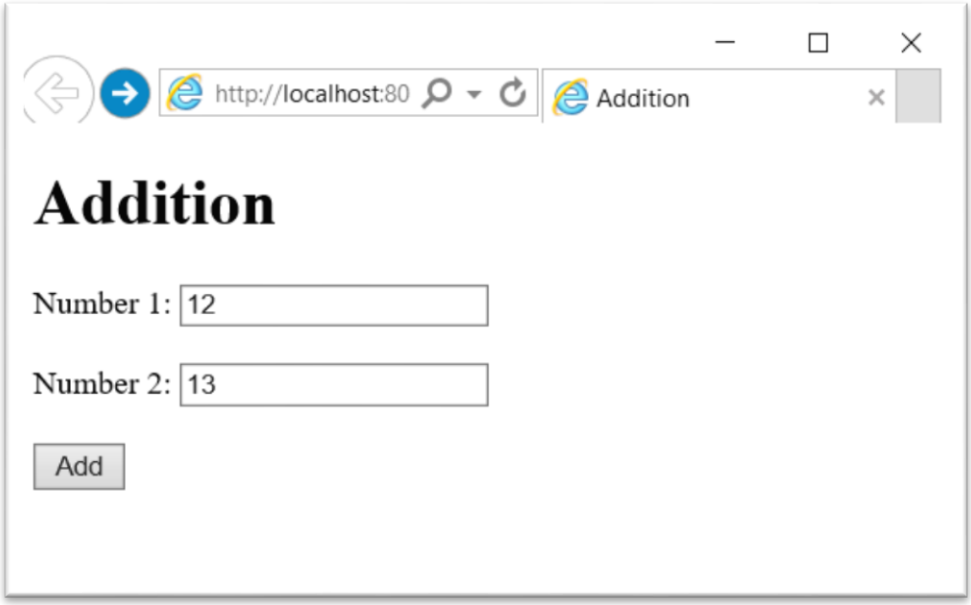
Index.html

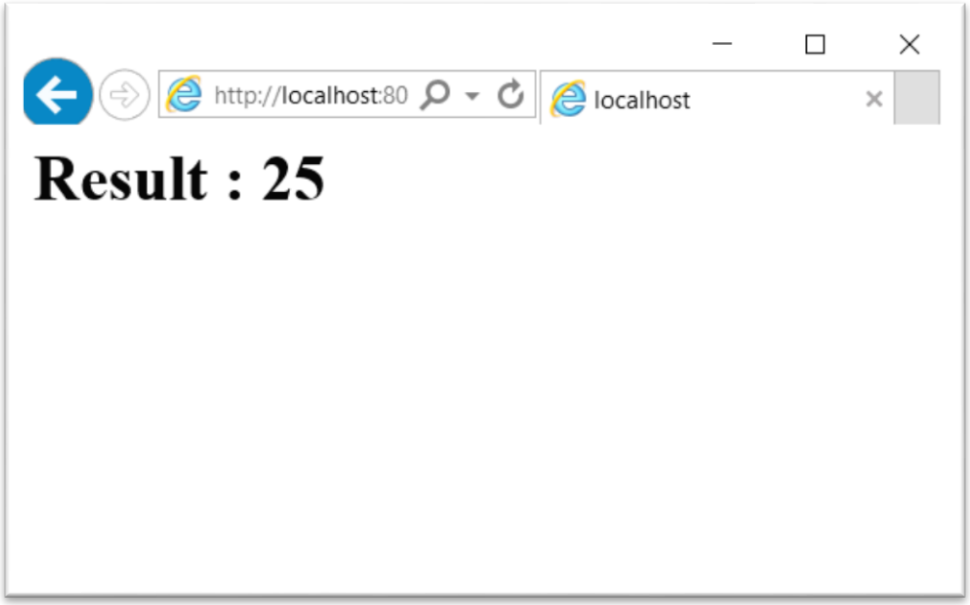
```
<!DOCTYPE html>
<html>
   <head>
      <title>Addition</title>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">

   </head>
   <body>
      <h1>Addition</h1>
      <form action="Addition">
         Number 1: <input type="text" name="no1"> <br><br>
         Number 2: <input type="text" name="no2"> <br><br>
         <input type="submit" value="Add">
      </form>
   </body>
```

Addition.java

```
@Override

   protected void doGet(HttpServletRequest request, HttpServletResponse response)

         throws ServletException, IOException {

      //processRequest(request, response);

      PrintWriter out = response.getWriter();

      int no1 = Integer.parseInt(request.getParameter("no1"));

      int no2 = Integer.parseInt(request.getParameter("no2"));

      out.println("<h1>Result : "+(no1+no2)+"</h1>");

   }
```

OUTPUT:





CONCLUSION:

In this practical we learnt how to develop a html file in servlet & perform addition of two numbers on Browser.

PRACTICAL-4

AIM: [Write a Servlet to demonstrate the difference between Request Dispatcher's forward method and sendRedirect method]

PROGRAM:

1.Servlet

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

@WebServlet("/MyServ")
public class MyServ extends HttpServlet {
        private static final long serialVersionUID = 1L;
        public MyServ() {
      super();
      // TODO Auto-generated constructor stub
    }
        protected void doGet(HttpServletRequest request, HttpServletResponse  response)
throws ServletException, IOException {
                response.setContentType("text/html");
            PrintWriter out = response.getWriter();
            out.print("<h3>Servlet Config object</h3>");
```

```
ServletConfig config=getServletConfig(); String
name=config.getInitParameter("name"); out.print("Name:
"+name);
            String id=config.getInitParameter("id");
            out.print("ID: "+id);

            out.print("<h3>Servlet Config object</h3>");
            ServletContext context=getServletContext();
            String prac=context.getInitParameter("practical");
            out.println("practical aim is="+prac);
        }
 }
```

2.Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" id="WebApp_ID" version="3.1">
  <display-name>Config_Context</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
```

```
  <servlet>
<servlet-name>MyServ</servlet-name>
<servlet-class>MyServ</servlet-class>

<init-param>
<param-name>name</param-name>
<param-value>Servlet</param-value>
</init-param>
<init-param>
<param-name>id</param-name>
<param-value>DEPSTAR-CSE</param-value>
</init-param>

</servlet>
<context-param>
<param-name>practical</param-name>
<param-value>AJP Practical</param-value>
</context-param>
<servlet-mapping>
<servlet-name>MyServ</servlet-name>
<url-pattern>/MyServ</url-pattern>
</servlet-mapping>
</web-app>
```

OUTPUT:

Servlet Config object

ServletDEPSTAR-CSE

Servlet Config object

AJP Practical

CONCLUSION:

In this practical we learnt about ServletConfig and ServletContext objects and how to use them.

PRACTICAL-5

AIM: [Write a Servlet to demonstrate the difference between Request Dispatcher's forward method and sendRedirect method]

PROGRAM:

Index.html

```
 <!DOCTYPE html>
 <html>
    <head>
       <title>Login</title>
       <meta charset="UTF-8">
       <meta name="viewport" content="width=device-width, initial-scale=1.0">

    </head>
    <body>
       <h1>Log In</h1>
       <form action="Login">
          Username: <input type="text" name="user"> <br><br>
          Password: <input type="text" name="pass"> <br><br>
          <input type="submit" value="Login">
       </form>
    </body>
</html>
```

NewServlet.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
import java.io.IOException;
import java.io.PrintWriter;
import static java.lang.System.out;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 *
 * @author smart
 */
public class NewServlet extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
```

```
            /* TODO output your page here. You may use following sample code. */
//          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>");


        }
    }


    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the
left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        //processRequest(request, response);
```

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

        //out.println("hello  "+user);
        if(user.equals("kashyap") && pass.equals("123")){
            //response.sendRedirect("Dashboard.jsp");
            RequestDispatcher rd = request.getRequestDispatcher("Dashboard.jsp");
            rd.forward(request, response);
        }
        else
        {
            out.println("Incorrect Details!");
        }
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
```

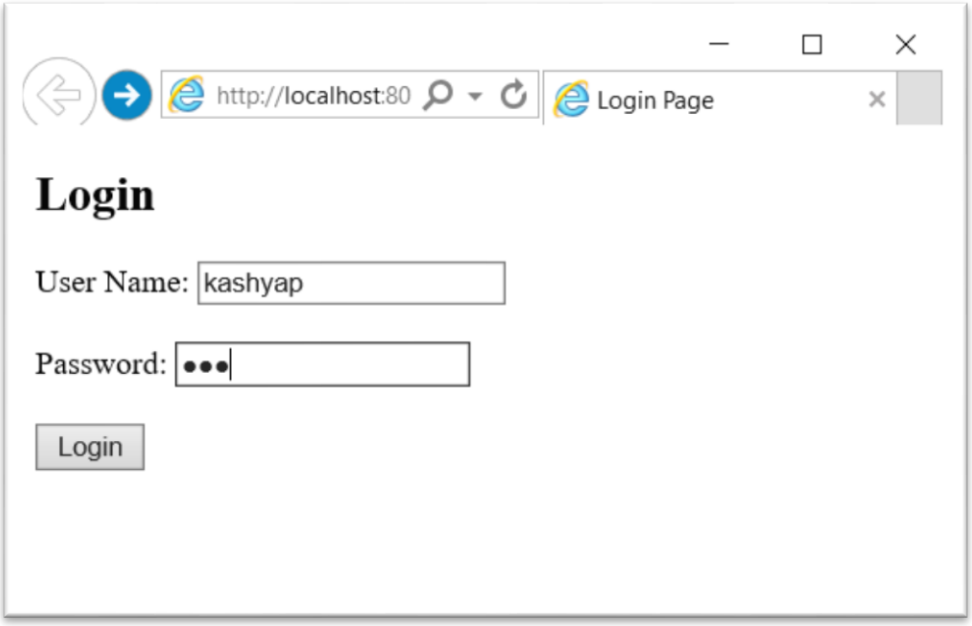
```
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>
}
```

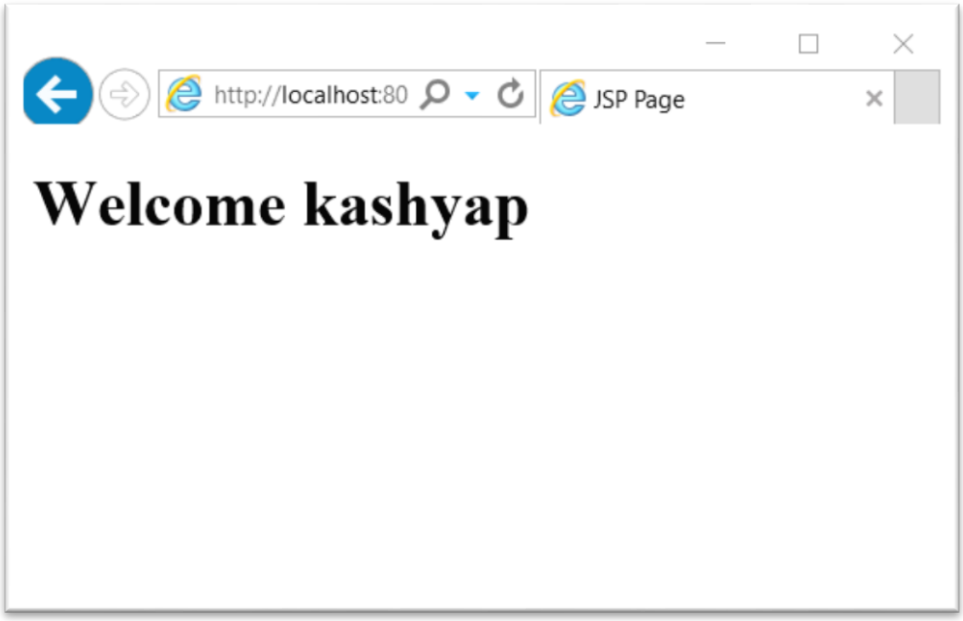
Dashboard.jsp

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Welcome <%=request.getParameter("user")%></h1>
    </body>
</html>
```

OUTPUT:

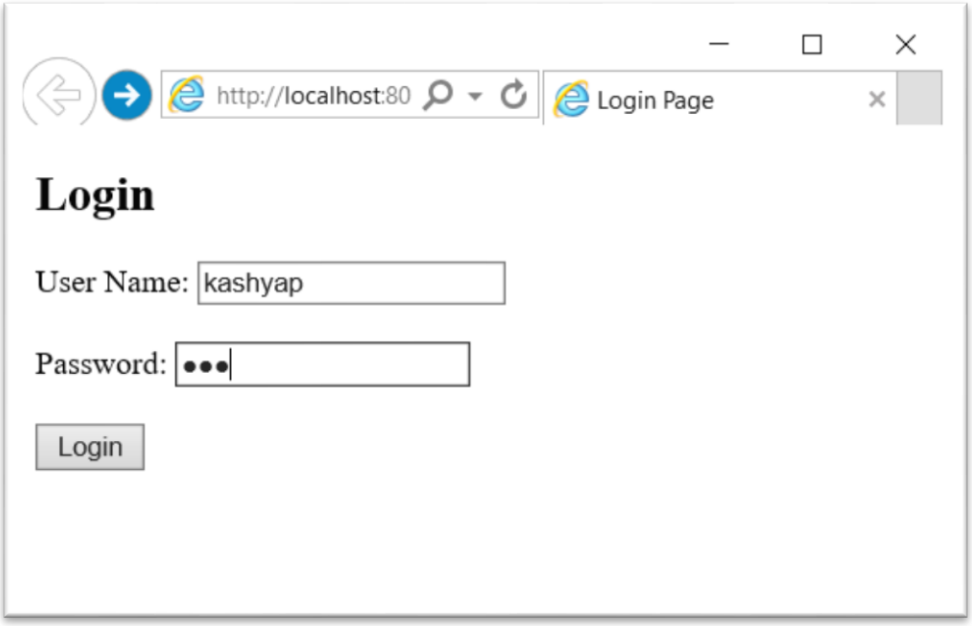
With RequestDispatcher's Forward:

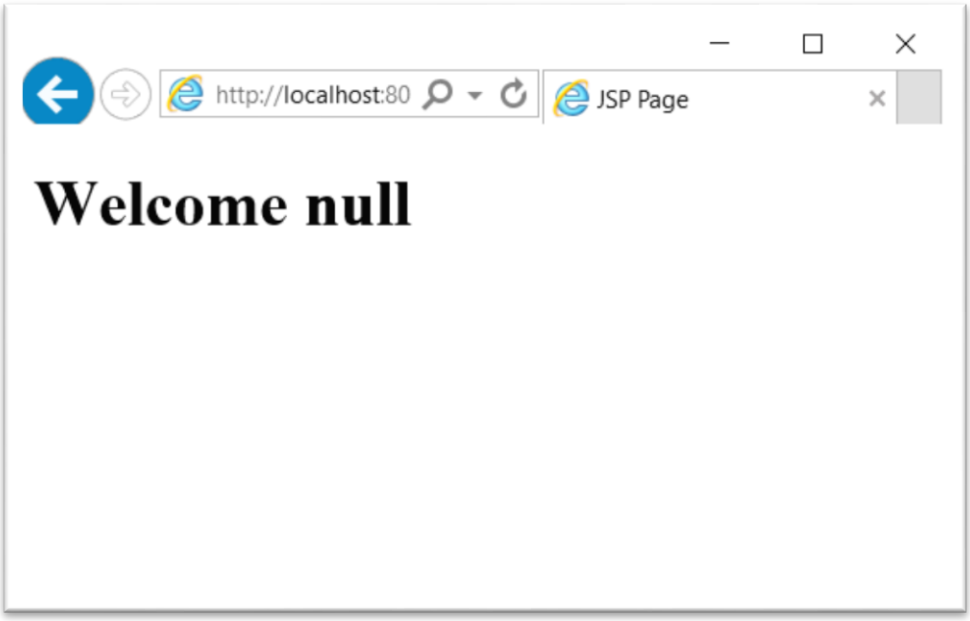




The object isn't lost as it was forwarded further to Dashboard.jsp with RequestDispatcher's Forward method.

With sendRedirect() method:





The object request and response is lost, so the output in null

CONCLUSION:

We studied the difference between RequestDispatcher's forward method and sendRedirect method with a simple example of login.

PRACTICAL 6

AIM: [Create Login and Logout modules using Servlet and HttpSession.]

PROGRAM:

Index.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>Login</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">

    </head>
    <body>
        <h1>Log In</h1>
        <form action="Login">
            Username: <input type="text" name="user"> <br><br>
            Password: <input type="text" name="pass"> <br><br>
            <input type="submit" value="Login">
        </form>
    </body>
</html>
```

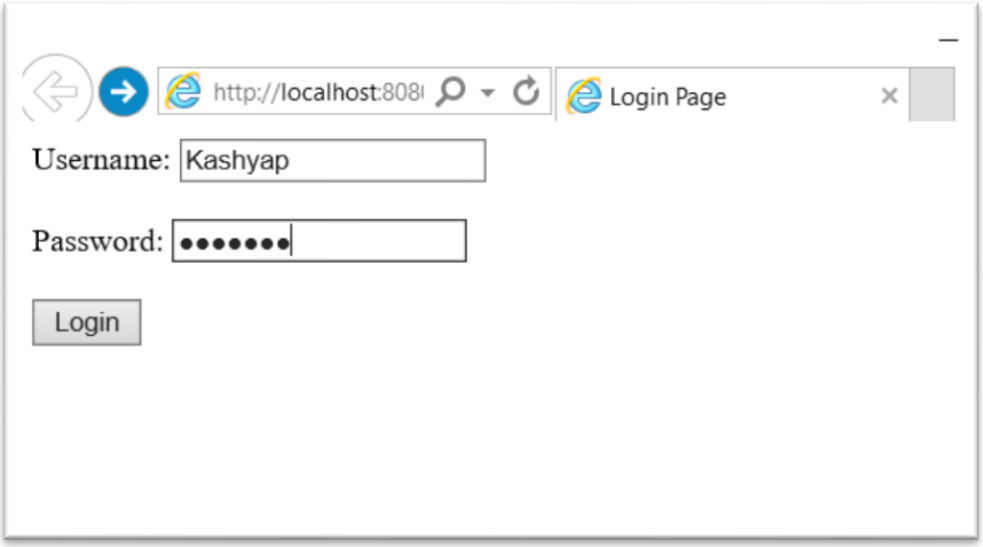
Login.java (Servlet)

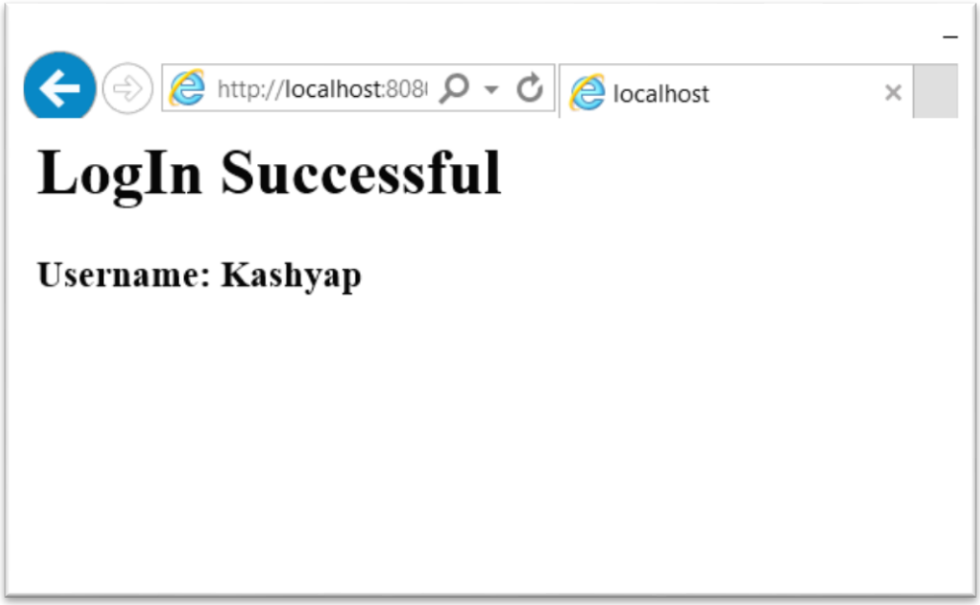
```
package com.demo.controller;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class Login extends HttpServlet {
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
```

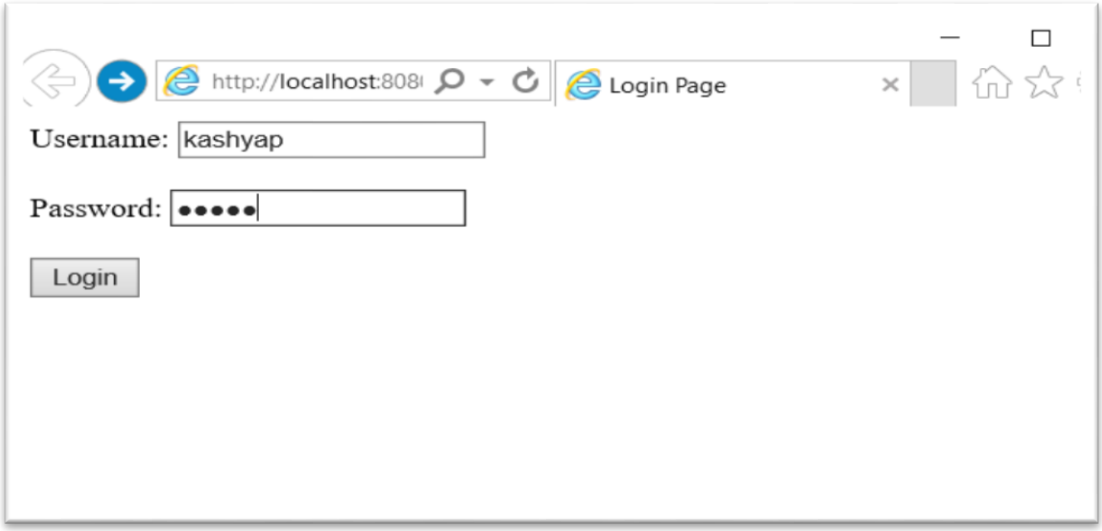
```
        throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        /* TODO output your page here. You may use following sample code. */
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet Login</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("</body>");
        out.println("</html>");
    }
}

@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    PrintWriter out = response.getWriter();
    String user = request.getParameter("user");
    String pass = request.getParameter("pass");
    if("Mark".equals(user)){
        if("mark".equals(pass)){
            out.println("<h1>LogIn Successful</h1>");
            out.println("<h3>Username: "+user+"</h3>");
        }
        else{
            out.println("<h1>LogIn unsuccessful</h1>");
            out.println("<h3>Message: Wrong Password.</h3>");
        }
    }
    else{
        out.println("<h1>LogIn unsuccessful</h1>");
        out.println("<h3>Message: User not found.</h3>");
    }
```

```
    }
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>
}
```

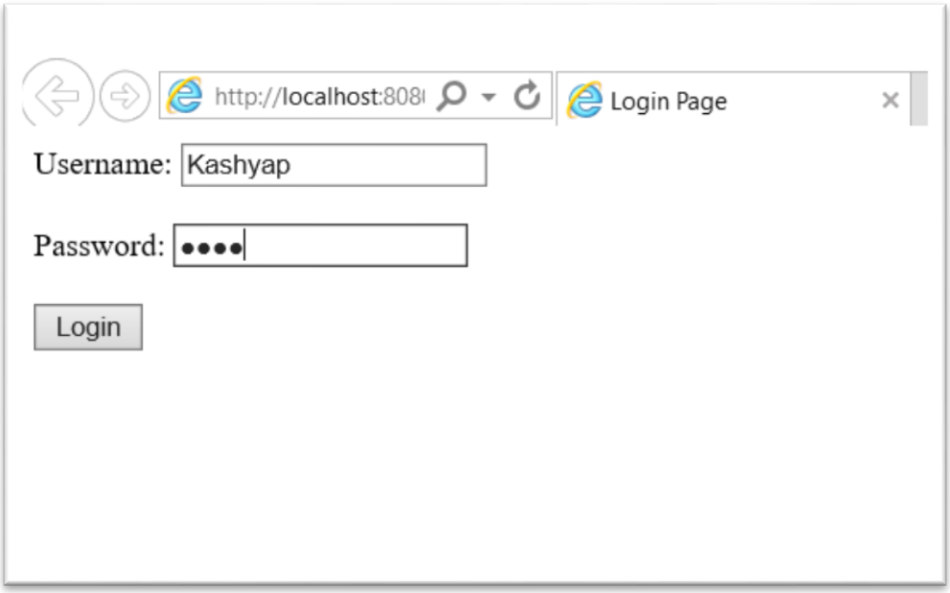
OUTPUT:

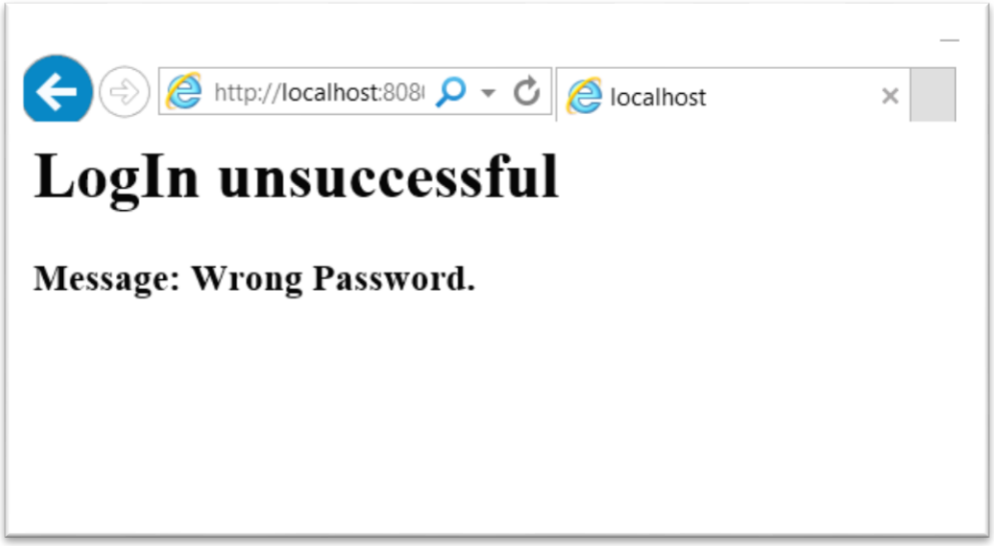





http://localhost:808
Login Page
Username: kashyap
Password:
Login


http://localhost:808
local
LogIn unsuccessful
Message: User not found.


http://localhost:808
Login Page
Username: Kashyap
Password:
Login



CONCLUSION:
We created basic Servlet project for Login process and displaying messages accordingly.

PRACTICAL 7

AIM: [Create a filter which allows only specific set of IP addresses to access application. Allowed IP addresses to be configured in context parameter.]

PROGRAM:

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebFilter; import javax.servlet.http.HttpServletRequest; import
javax.servlet.http.HttpServletResponse;

@WebFilter("/Servlet")
public class IPFilter implements Filter {

        private static String allowIP = "0:0:0:0:0:0:0:1";

 public IPFilter() {

        // TODO Auto-generated constructor stub

 }

public void destroy() {
// TODO Auto-generated method stub

}
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
throws IOException, ServletException {
        HttpServletRequest req = (HttpServletRequest) request;
        HttpServletResponse res = (HttpServletResponse) response;
```

```
        PrintWriter out = res.getWriter();
        String ipAddr = req.getRemoteAddr();
         System.out.println("IP-Addr: " + ipAddr);
if (ipAddr.equals(allowIP)) {
chain.doFilter(request, response);
} else {
        out.println("Request Denied!");
 }
        chain.doFilter(request, response);
}
/**
* @see Filter#init(FilterConfig)
*/
public void init(FilterConfig fConfig) throws ServletException { /
        / TODO Auto-generated method stub
        }
}
```

CONCLUSION:

In this practical, we learnt how Filters can be used in Servlet programming.

PRACTICAL 8

AIM: [Create a JSP page that display number of hits to the page.]

PROGRAM:

```
<%@ page import = "java.io.*,java.util.*" %>
<html>
        <head>
                <title>Application object in JSP</title>
         </head>
<body>
 <%
        Integer hitsCount = (Integer)application.getAttribute("hitCounter"); if( hitsCount ==null
        || hitsCount == 0 ) {
                /* First visit */
                out.println("Hit Counter Practical!");
                hitsCount = 1;
        } else {

                /* return visit */
                out.println("Welcome back to my website!");
                hitsCount += 1;
        }
        application.setAttribute("hitCounter", hitsCount);
         %>
        <center>

                <p>Total number of visits: <%= hitsCount%></p>
        </center>
</body>
</html>
```

CONCLUSION:

In this practical, we learnt about application implicit object in JSP.

PRACTICAL 9

AIM: [Create a JSP page to demonstrate various JSP directives and actions.]

PROGRAM:

include.html

```
!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1"> <title>Insert title here</title>
</head>
<body>
<h3>Content using include action</h3> </body>
</html>
```

page1.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %> <!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1"> <title>Insert title here</title> </head>
<body>
<h3>JSP include action tag</h3>
<jsp:include page="include.html"></jsp:include> <form action="redirect.jsp">
<input type=submit value="Click me to redirect!"> </form>
</body>
</html>
```

redirect.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ page import="java.util.Date"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %> <!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1"> <title>Insert title here</title> </head>
<body>
<h3>Date: </h3>
<p>Date class imported using import attribute of page directive and core tags are imported using
taglib directive to display date value</p>
<c:set var="Date" value="<%=new java.util.Date()%>" />
<c:out value="${Date}"></c:out>
</body>
</html>
```

OUTPUT:

Date:

Date class imported using import attribute of page directive and core tags are imported using taglib directive to display date value

Wed Oct 21 16:38:07 IST 2020

CONCLUSION:

In this practical, we learnt about various JSP directives and actions and used a few in this practical.

PRACTICAL 10

AIM: [Create a JSP page which accepts a number from user and display the table of that nuber using core tag library.]

PROGRAM:

index.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1"> <title>Insert title here</title> </head>
<body>
<form action="table.jsp">
Enter a number
<input type="text" name="num"> </form>
</body>
</html>
```

table.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %> <!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1"> <title>Insert title here</title> </head>
<body>
<c:set var="val" scope="session" value="${param.num}"/> <c:forEach var="i" begin="1"
end="10">
<c:out value="${i*val}"/><p> </c:forEach>
</body>
 </html>
```

OUTPUT:

Enter a number 10 ×

10

20

30

40

50

60

70

80

90

100

CONCLUSION:

In this practical, we learnt about core tags of JSTL.

PRACTICAL 11

AIM: [Develop a JSP page that display Student information from database using SQL Tag Library.]

PROGRAM:

```
<%@ page language="java" contentType="text/html; pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %> <%@ taglib
uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1"> <title>Dsiplay using JSP</title> </head>

<body>

<sql:setDataSource var="db" driver="com.mysql.jdbc.Driver" url="jdbc:mysql://localhost/db_jsp"
user="root" password=""/>

<sql:query dataSource="${db}" var="rs"> SELECT * from student;
</sql:query>

<table border="2" width="100%"> <tr>
<th>ID</th>
<th>Name</th>

</tr>

<c:forEach var="t" items="${rs.rows}"> <tr>
<td><c:out value="${t.sid}"/></td>

 <td><c:out value="${t.sname}"/></td>

 <td><c:out value="${t.branch}"/></td>

 <td><c:out value="${t.semester}"/></td>

</tr> </c:forEach>

</table>

</body>

</html>
```

OUTPUT:

1. Data from DB

+ Options

sid	sname	branch	semester
A100	AAABBBCCC	CSE	5
A101	DDDEEEFFF	CE	5
A102	PPPQQQRRR	IT	5

2. Displayed using JSP

ID	Name		
A100	AAABBBCCC	CSE	5
A101	DDDEEEFFF	CE	5
A102	PPPQQQRRR	IT	5

CONCLUSION:

In this practical, we learnt about sql tags of JSTL.