



CHARUSAT

CSPIT

Department of Information Technology



Laboratory Manual

Subject:	Advanced Java Programming
Code:	IT 371
Offered to:	3 rd Year Students
Semester:	5 th
Total Credit:	4
Total Lectures:	2 per week
Lab Hours:	4 hrs. per week
Prepared By:	Pinal Shah
Subject Teachers:	Pinal Shah
Lab Teachers:	Jalpesh Vasa, Hemant Yadav



CHARUSAT

CSPIT

Department of Information Technology



File Format for Students

The students will write the today's experiment in the Observation book as per the following format:

- a) Name of the experiment/Aim
- b) Algorithm
- c) Source Program
- d) Test Data
 - a. Valid data sets
 - b. Limiting value sets
 - c. Invalid data sets
- e) Questions and Answers
- f) Errors observed (if any) during compilation/execution
- g) Signature of the Faculty

3.1 Using MySQL with NetBeans IDE

This chapter demonstrates how to set up a connection to a MySQL database from the NetBeans IDE. Once connected, you can begin working with MySQL in the IDE's Database Explorer by creating new databases and tables, populating tables with data, and running SQL queries on database structures and content. This chapter is designed for beginners with a basic understanding of database management, who want to apply their knowledge to working with MySQL in NetBeans IDE.

MySQL is a popular Open Source relational database management system (RDBMS) commonly used in web applications due to its speed, flexibility and reliability. MySQL employs SQL, or Structured Query Language, for accessing and processing data contained in databases.

We have developed all programs using following software and resources.

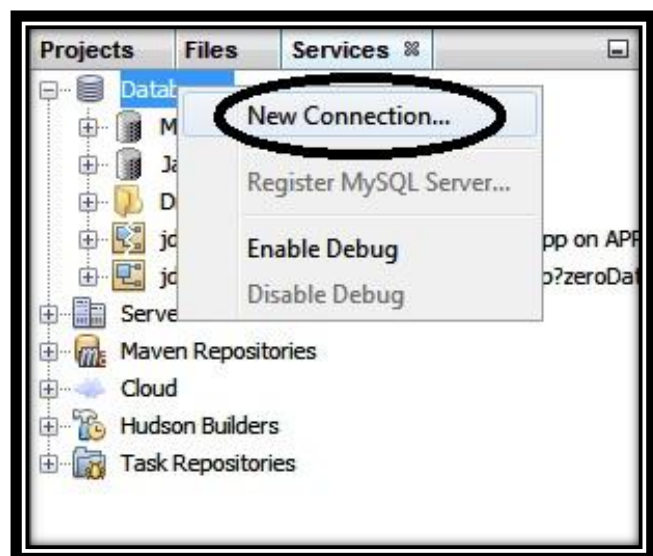
Software or Resource	Version Used
NetBeans IDE	8.0
Java Development Kit (JDK)	Version 7 or 8
MySQL database server	version 5.x

3.1.1 Configuring MySQL server with NetBeans

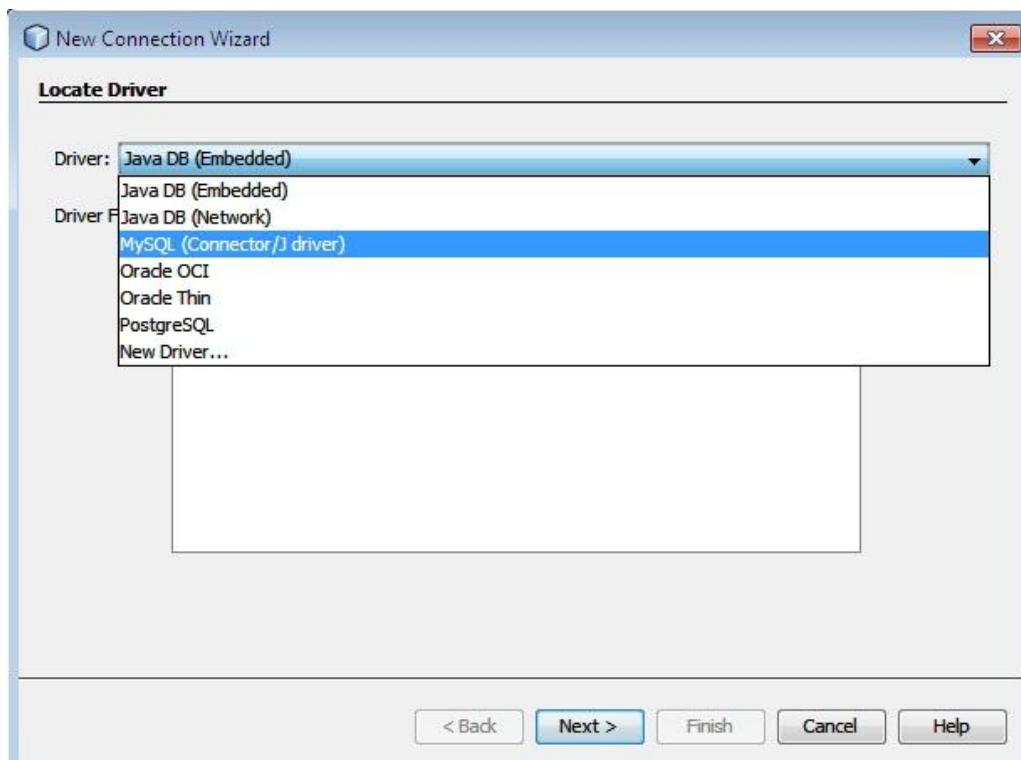
In NetBeans you can see three tabs in the project window. Project, Files and Services.

Go into the services tab as shown in the below figure (left hand side) where you can find Database, Server etc. link.

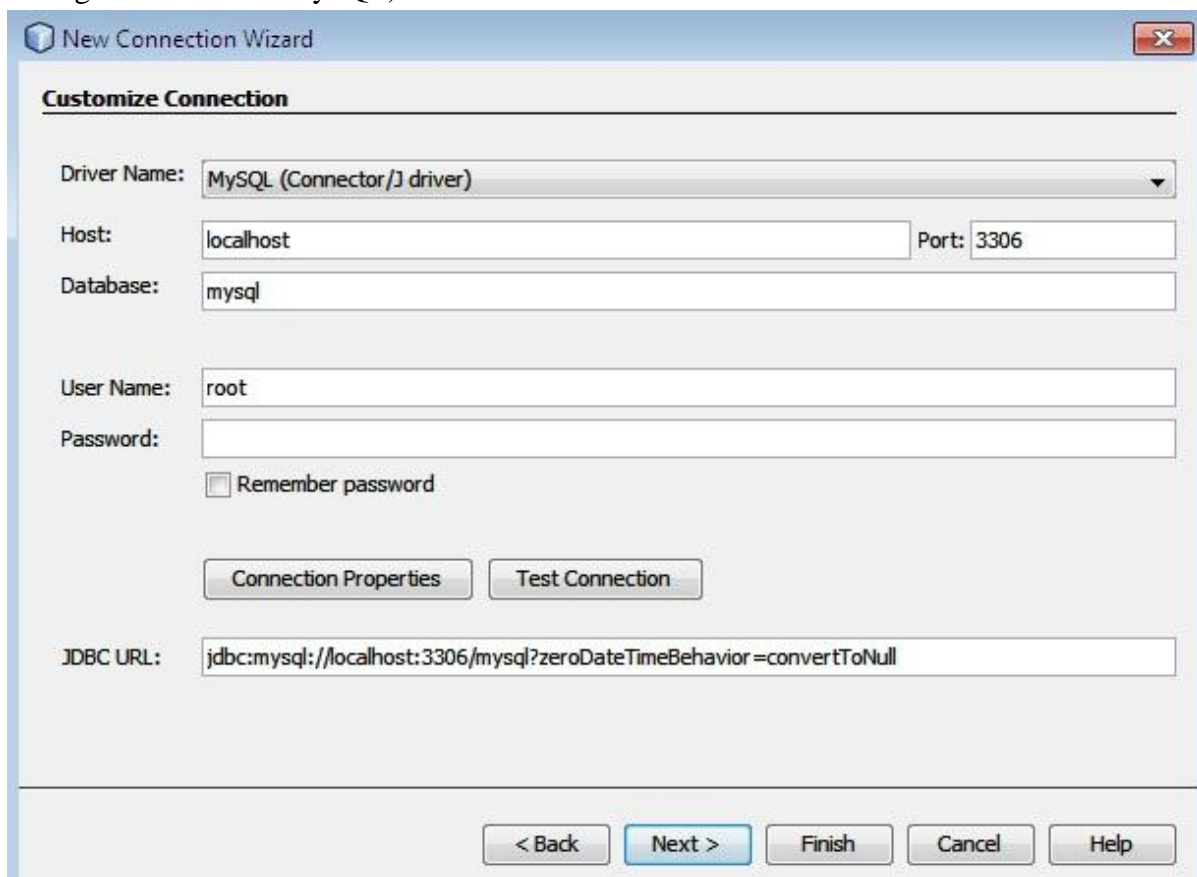
Right click on database link and select New Connection as shown in right side figure.



Out of all available driver select MySQL (Connector/J Driver)



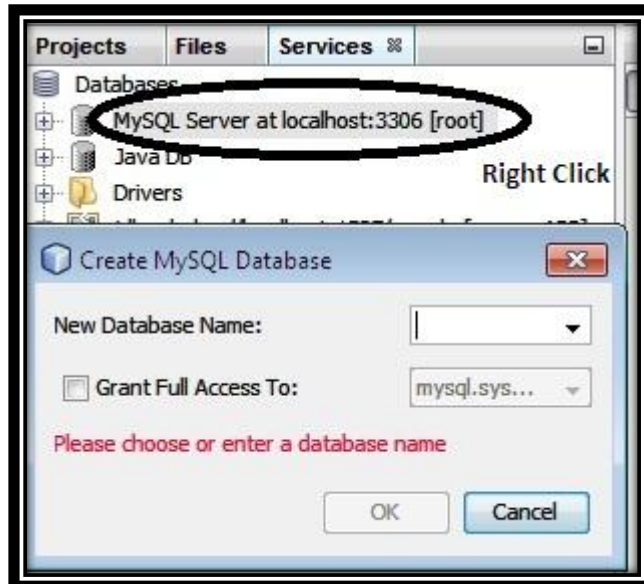
It will open below window which will ask for the password. (Password which you have used during installation of MySQL)



Click on “Test Connection” and finish. Done!!!

3.2 Database Setup in NetBeans

After successful configuration of MySQL in NetBeans now we can create the database and tables.



Database Name

ebookshop

Table Name

books

Column Name

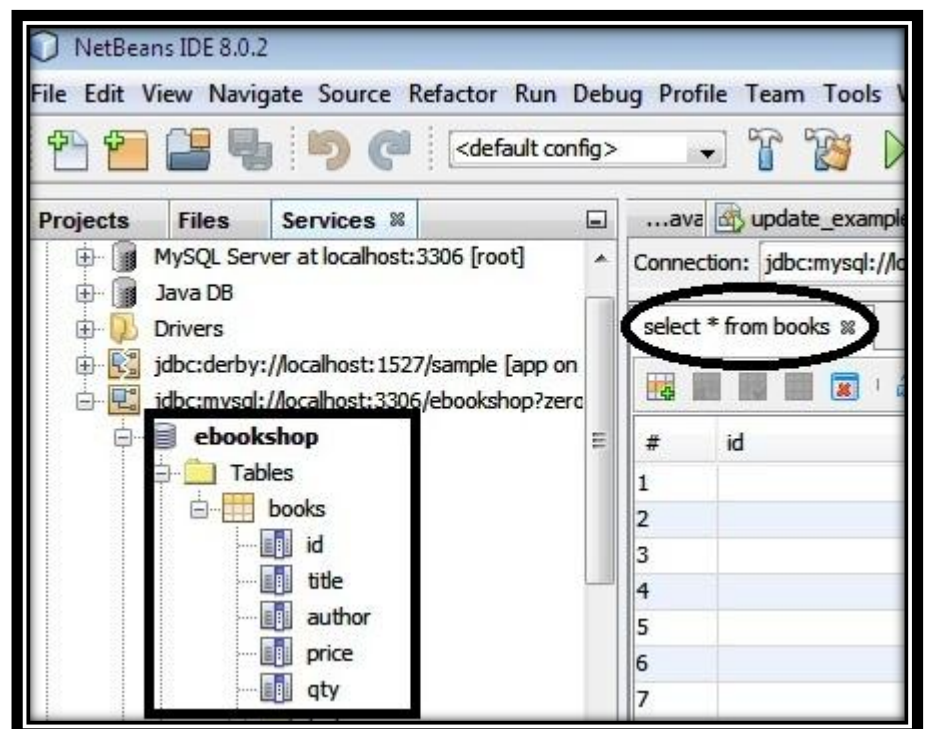
id(INT)

title (VARCHAR),

author (VARCHAR),

price (FLOAT),

qty (INT)



3.3 JDBC Programming By Example

In this section we have followed the above created database and tables.

3.3.1 SQL SELECT

```
import java.sql.*; // Use classes in java.sql package

// JDK 7 and above
public class select_example { // Save as "JdbcSelectTest.java"
    public static void main(String[] args) {
        try (
            // Step 1: Allocate a database "Connection" object
            Connection conn = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/ebookshop", "root", ""); // MySQL

            // Step 2: Allocate a "Statement" object in the Connection
            Statement stmt = conn.createStatement();
        ) {
            // Step 3: Execute a SQL SELECT query, the query result
            // is returned in a "ResultSet" object.
            String strSelect = "select title, price, qty from books";
            System.out.println("The SQL query is: " + strSelect); // Echo For debugging
            System.out.println();

            ResultSet rset = stmt.executeQuery(strSelect);

            // Step 4: Process the ResultSet by scrolling the cursor forward via next().
            // For each row, retrieve the contents of the cells with getXxx(columnName).
            System.out.println("The records selected are:");
            int rowCount = 0;
            while(rset.next()) { // Move the cursor to the next row
                String title = rset.getString("title");
                double price = rset.getDouble("price");
                int qty = rset.getInt("qty");
                System.out.println(title + ", " + price + ", " + qty);
                ++rowCount;
            }
            System.out.println("Total number of records = " + rowCount);

        } catch(SQLException ex) {
            ex.printStackTrace();
        }
        // Step 5: Close the resources - Done automatically by try-with-resources
    }
}
```

3.3.2 SQL UPDATE

```
import java.sql.*; // Use classes in java.sql package

// JDK 7 and above
public class update_example { // Save as "JdbcUpdateTest.java"
    public static void main(String[] args) {
        try (
            // Step 1: Allocate a database "Connection" object
            Connection conn = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/ebookshop", "root", ""); // MySQL

            // Step 2: Allocate a "Statement" object in the Connection
            Statement stmt = conn.createStatement();
        ) {
            // Step 3 & 4: Execute a SQL UPDATE via executeUpdate()
            // which returns an int indicating the number of rows affected.
            // Increase the price by 7% and qty by 1 for id=1001
            String strUpdate = "update books set price = price*0.7, qty = qty+1 where id = 1001";
            System.out.println("The SQL query is: " + strUpdate); // Echo for debugging
            int countUpdated = stmt.executeUpdate(strUpdate);
            System.out.println(countUpdated + " records affected.");

            // Step 3 & 4: Issue a SELECT to check the UPDATE.
            String strSelect = "select * from books where id = 1001";
            System.out.println("The SQL query is: " + strSelect); // Echo for debugging
            ResultSet rset = stmt.executeQuery(strSelect);
            while(rset.next()) { // Move the cursor to the next row
                System.out.println(rset.getInt("id") + ", "
                    + rset.getString("author") + ", "
                    + rset.getString("title") + ", "
                    + rset.getDouble("price") + ", "
                    + rset.getInt("qty"));
            }
        } catch(SQLException ex) {
            ex.printStackTrace();
        }
        // Step 5: Close the resources - Done automatically by try-with-resources
    }
}
```

3.3.3 SQL INSERT and DELETE

```
import java.sql.*; // Use classes in java.sql package

// JDK 7 and above
public class insert_delete_example { // Save as "JdbcUpdateTest.java"
    public static void main(String[] args) {
        try (
            // Step 1: Allocate a database "Connection" object
```

```

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

// Step 2: Allocate a "Statement" object in the Connection
Statement stmt = conn.createStatement();
) {
// Step 3 & 4: Execute a SQL INSERT|DELETE statement via executeUpdate(),
// which returns an int indicating the number of rows affected.

// DELETE records with id>=3000 and id<4000
String sqlDelete = "delete from books where id>=3000 and id<4000";
System.out.println("The SQL query is: " + sqlDelete); // Echo for debugging
int countDeleted = stmt.executeUpdate(sqlDelete);
System.out.println(countDeleted + " records deleted.\n");

// INSERT a record
String sqlInsert = "insert into books " // need a space
    + "values (3001, 'Gone Fishing', 'Kumar', 11.11, 11)";
System.out.println("The SQL query is: " + sqlInsert); // Echo for debugging
int countInserted = stmt.executeUpdate(sqlInsert);
System.out.println(countInserted + " records inserted.\n");

// INSERT multiple records
sqlInsert = "insert into books values "
    + "(3002, 'Gone Fishing 2', 'Kumar', 22.22, 22),"
    + "(3003, 'Gone Fishing 3', 'Kumar', 33.33, 33)";
System.out.println("The SQL query is: " + sqlInsert); // Echo for debugging
countInserted = stmt.executeUpdate(sqlInsert);
System.out.println(countInserted + " records inserted.\n");

// INSERT a partial record
sqlInsert = "insert into books (id, title, author) "
    + "values (3004, 'Fishing 101', 'Kumar')";
System.out.println("The SQL query is: " + sqlInsert); // Echo for debugging
countInserted = stmt.executeUpdate(sqlInsert);
System.out.println(countInserted + " records inserted.\n");

// Issue a SELECT to check the changes
String strSelect = "select * from books";
System.out.println("The SQL query is: " + strSelect); // Echo For debugging
ResultSet rset = stmt.executeQuery(strSelect);
while(rset.next()) { // Move the cursor to the next row
    System.out.println(rset.getInt("id") + ", "
        + rset.getString("author") + ", "
        + rset.getString("title") + ", "
        + rset.getDouble("price") + ", "
        + rset.getInt("qty"));
}
} catch(SQLException ex) {    ex.printStackTrace();    }
// Step 5: Close the resources - Done automatically by try-with-resources

```



```
}  
}
```

3.3.4 PreparedStatement Example

```
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.PreparedStatement;  
  
public class prepare_insert {  
  
    public static void main(String[] args) throws Exception {  
        Connection conn=null;  
        PreparedStatement pstmt=null;  
        try {  
            conn = DriverManager.getConnection(  
                "jdbc:mysql://localhost:3306/ebookshop", "root", ""); // MySQL  
  
            String query = "insert into books(id,title,author,price,qty) values(?, ?, ?, ?, ?)";  
  
            pstmt = conn.prepareStatement(query); // create a statement  
            pstmt.setInt(1, 1); // set input parameter 1  
            pstmt.setString(2, "title"); // set input parameter 2  
            pstmt.setString(3, "author"); // set input parameter 3  
            pstmt.setFloat(4, 12.20f);  
            pstmt.setInt(5,10);  
            pstmt.executeUpdate(); // execute insert statement  
        } catch (Exception e) {  
            e.printStackTrace();  
        } finally {  
            pstmt.close();  
            conn.close();  
        }  
    }  
}
```

3.3.5 Batch Processing Example

```
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.PreparedStatement;  
import java.sql.SQLException;  
  
public class batch_example {  
    public static void main(String[] args) throws SQLException {  
        Connection conn=null;  
        PreparedStatement pstmt=null;  
        try {
```

```

conn = DriverManager.getConnection(
    "jdbc:mysql://localhost:3306/ebookshop", "root", ""); // MySQL

String query = "insert into books(id,title,author,price,qty) values(?, ?, ?, ?, ?)";

pstmt = conn.prepareStatement(query); // create a statement

//insert one row
pstmt.setInt(1, 1); // set input parameter 1
pstmt.setString(2, "title"); // set input parameter 2
pstmt.setString(3, "author"); // set input parameter 3
pstmt.setFloat(4, 12.20f);
pstmt.setInt(5,10);
pstmt.addBatch(); //add row in batch for execution

//insert second row
pstmt.setInt(1, 1); // set input parameter 1
pstmt.setString(2, "title1"); // set input parameter 2
pstmt.setString(3, "author1"); // set input parameter 3
pstmt.setFloat(4, 12.30f);
pstmt.setInt(5,12);
pstmt.addBatch(); //add row in batch for execution

//finally execute the whole batch
pstmt.executeBatch();

} catch (Exception e) {
    e.printStackTrace();
} finally {
    pstmt.close();
    conn.close(); } } }

```

3.3.6 JDBC Transaction Management Example

```

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class transaction_example {
    public static void main(String[] args) throws Exception {
        String url = "jdbc:mysql://localhost/ebookshop";
        String username = "root";
        String password = "";
        Class.forName("com.mysql.jdbc.Driver");
        Connection conn = null;
        try {

```

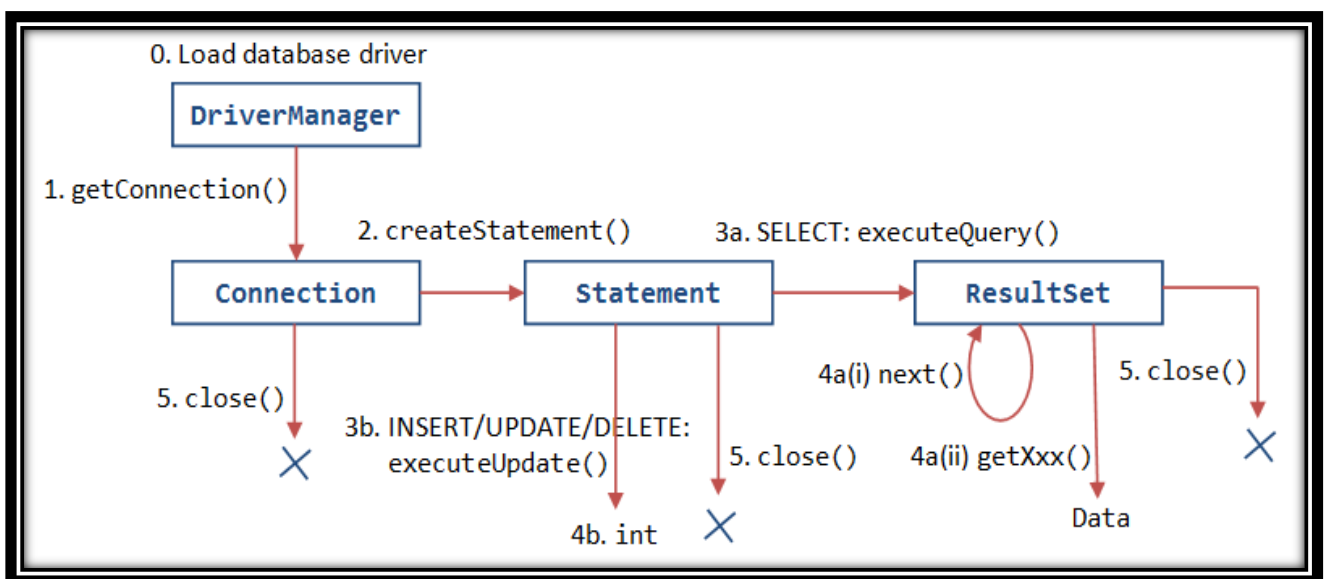
```

conn = DriverManager.getConnection(url, username, password);
conn.setAutoCommit(false);
Statement st = conn.createStatement();
st.execute("INSERT INTO orders (username, order_date) VALUES ('java', '2016-12-13')",
    Statement.RETURN_GENERATED_KEYS);

ResultSet keys = st.getGeneratedKeys();
int id = 1;
while (keys.next()) {
    id = keys.getInt(1);
}
PreparedStatement pst = conn.prepareStatement("INSERT INTO books (id, title, author,
price, qty) VALUES (?, ?, ?, ?, ?)");
pst.setInt(1, id);
pst.setString(2, "1");
pst.setString(3, "2");
pst.setFloat(4, 10.04f);
pst.setInt(5, 100);
pst.execute();
conn.commit();
System.out.println("Transaction commit...");
} catch (SQLException e) {
    if (conn != null) {
        conn.rollback();
        System.out.println("Connection rollback...");
    }
    e.printStackTrace();
} finally {
    if (conn != null && !conn.isClosed()) {
        conn.close();
    } } }

```

3.3.7 Conclude the JDBC in Style!!!



Write a program to display “Hello World” in Servlet.

We should start understanding the servlets from the beginning. Let's start by making one program which will just print the "Hello World" on the browser. Each time the user visits this page it will display "Hello World" to the user.

As we know that the servlet extends the `HttpServlet` and overrides the `doGet ()` method which it inherits from the `HttpServlet` class. The server invokes `doGet ()` method whenever web server receives the GET request from the servlet. The `doGet ()` method takes two arguments first is `HttpServletRequest` object and the second one is `HttpServletResponse` object and this method throws the `ServletException`.

Program (pr1_hello.java / Servlet file)

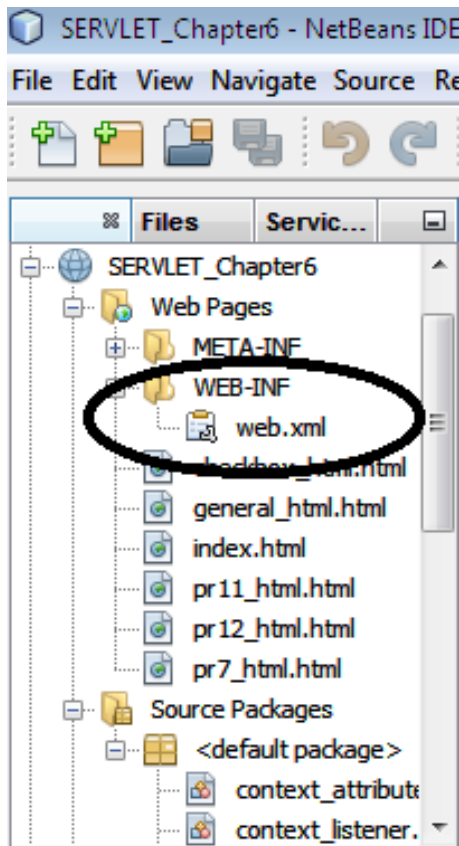
```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class pr1_hello extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        out.println("hello");
    }
}
```

Web.xml (Configuring servlet in web.xml file)

At what location web.xml file is reside in NetBeans? Check the below figure



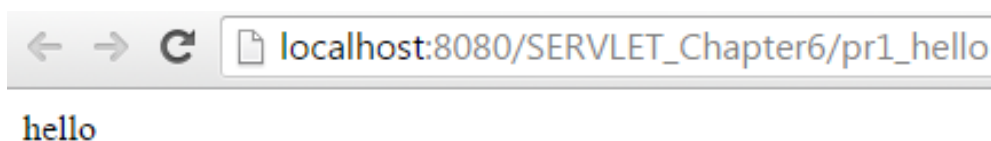
How it looks?

```
<web-app>
<servlet>
<servlet-name>pr1_hello</servlet-name>
<servlet-class>pr1_hello</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>pr1_hello</servlet-name>
<url-pattern>/pr1_hello</url-pattern>
</servlet-mapping>
</web-app>
```

Output

How to execute the servlet program in NetBeans?

After making the java file you need to clean and build the web application in NetBeans. And then deploy the application. After deployment phase now you are ready to run your first servlet program. (Right click on servlet name and select Run option). You can see the following output.



Note

Web.xml file remain same for the whole web application. And you can make number of servlet program within single web application. So there is only one web.xml (configuration file) for the application. In this chapter whatever program we have demonstrate that all the programs are the part of single web application. So in the last we have kept the final web.xml file for all of you.

Write a program to make counter application using servlet which will keep track of user visit.

In this example we are going to know how we can make a program on counter which will keep track how many times the servlet has been accessed.

To make this program firstly we have to make one class. The name of the class should follow the naming convention. Remember to keep the name of the class in such a way that it becomes easy to understand what the program is going to do just by seeing the class name. After making a class define one variable counter which will keep record for how many times the servlet has been accessed. Now use method either doGet () or doPost () to write a logic of the program. Our program logic is simple. We have to just increment the value of the counter by 1. To display the output use the method getWriter () method of the response object which will in turn return the object of the PrintWriter class. Now display the value of the counter.

Program (pr2_counter.java / Servlet file)

```
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 pr2_counter extends HttpServlet {

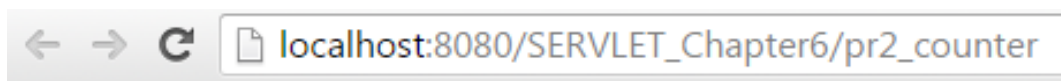
    int count=0;
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        count++;
        out.println("page view is:"+count);

    }

}
```

Output

Press Refresh button five times and you will get the following output.



page view is:5

Write a program to demonstrate the use of Servlet config in detail by example.

In this example we are going to retrieve the init parameter values which we have given in the web.xml file.

Whenever the container makes a servlet it always reads its deployment descriptor file i.e. web.xml. Container creates name/value pairs for the ServletConfig object. Once the parameters are in ServletConfig they will never be read again by the Container.

The main job of the ServletConfig object is to give the init parameters.

To retrieve the init parameters in the program firstly we have made one class. The container calls the servlet's service () method then depending on the type of request, the service method calls either the doGet () or the doPost (). By default it will be doGet () method. Now inside the doGet () method use getWriter () method of the response object which will return an object of the PrintWriter class which helps us to print the content on the browser.

To retrieve all the values of the init parameter use method getInitParameterNames () which will return the Enumeration of the init parameters.

Web.xml file (Deployment Descriptor file)

```
<web-app>
<servlet>
    <servlet-name>pr3_config</servlet-name>
    <servlet-class>pr3_config</servlet-class>
    <init-param>
        <param-name>id</param-name>
        <param-value>pinal</param-value>
    </init-param>
</servlet>
<servlet-mapping>
    <servlet-name>pr3_config</servlet-name>
    <url-pattern>/pr3_config</url-pattern>
```

```
</servlet-mapping></web-app>
```

Program (pr3_config.java / servlet)

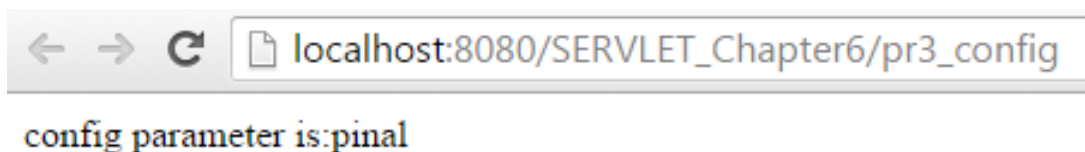
```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class pr3_config extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();

        String str=getServletConfig().getInitParameter("id");
        out.println("config parameter is:"+str);
    }
}
```

Output



Write a program to demonstrate the use of Servlet context in detail by example.

Servlet config object is separate for each servlet available in web application while the context parameter is for web application and not for particular servlet. So you can access context parameter in any servlet of the web application but you cannot access one servlet config parameter in another servlet. If try for this than it will not show any error instead it will display null value in browser.

Web.xml

```
<web-app>

    <context-param>
```



```

    <param-name>college_name</param-name>

    <param-value>charusat</param-value>

</context-param>

<servlet>

    <servlet-name>pr4_context</servlet-name>

    <servlet-class>pr4_context</servlet-class>

</servlet>

<servlet-mapping>

    <servlet-name>pr4_context</servlet-name>

    <url-pattern>/pr4_context</url-pattern>

</servlet-mapping>

</web-app>

```

Program (pr4_context.java / servlet)

```

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 pr4_context extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();

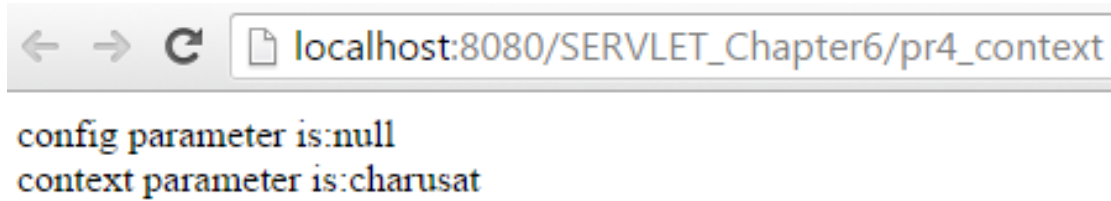
        //not possible to fetch config parameter in this servlet.
        String str=getServletConfig().getInitParameter("id");
        out.println("config parameter is:"+str); //print null

        String str1=getServletConfig().getServletContext().getInitParameter("college_name");
        out.println("context parameter is:"+str1);
    }
}

```

```
}
```

Output



Write a program for passing HTML parameter to servlet file.

This is a very simple example in which we are going to display the name on the browser which we have entered from the Html page.

To get the desired result firstly we have to make one html form which will have only one field in which we will enter the name. And we will also have one submit button, on pressing the submit button the request will go to the server and the result will be displayed to us.

In the servlet which will work as a controller here picks the value from the html page by using the method `getParameter ()`. The output will be displayed to you by the object of the `PrintWriter` class.

HTML File

```
<!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>
    <div>
      <form name="f1" action="pr5_fetch_html_single" method="get">
        username: <input type="text" name="uname"><br>
        password: <input type="password" name="password"><br>
        Gender: <input type="radio" name="gender" value="male">Male
               <input type="radio" name="gender" value="female">Female<br>
        <input type="submit" name="ok" value="submit">
      </form>
    </div>
  </body>
</html>
```

Program (pr5_fetch_html_single.java / servlet)

```
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 pr5_fetch_html_single extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();

        //getParameter() is used to fetch single value from HTML file.
        String username=request.getParameter("uname");
        out.println("username is:"+username);
        out.println("</br>");

        String password=request.getParameter("password");
        out.println("password is:"+password);
        out.println("</br>");

        String gender=request.getParameter("gender");
        out.println("gender is:"+gender);
        out.println("</br>");
    }
}
```

Output



username:

password:

Gender: ☒ Male ☐ Female



username is:advanceJava
password is:123
gender is:male

[Check the URL](#)

Write a program for passing HTML parameter (Checkbox) multiple values to servlet file.

In our program it may be that we may have multiples values for a single parameter like in checkboxes. We are going to make one program over it.

To make such a servlet which we have made one html form from where the values will be passed to the controller. In this program we have used the checkbox which will have the same name but with different values. We have one more button submit, on pressing this button the request will be forwarded.

Now in the servlet that is working like a controller will retrieve the values we have entered in the html form by the method `getParameterValues ()` which returns the array of String. At last to retrieve all the values from the array we need to use for loop. The output will be displayed to you by the `PrintWriter` object.

HTML File

```
<!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>
    <div>
      <form name="f1" action="pr6_fetch_html_multiple" method="get">
        username: <input type="text" name="uname"><br>
        password: <input type="password" name="password"><br>
        Gender: <input type="radio" name="gender" value="male">Male
```

```

        <input type="radio" name="gender" value="female">Female<br>
        Area of Interest: <input type="checkbox" name="area"
value="Java">Java<br>
        <input type="checkbox" name="area" value="Advanced Java">Advanced
Java<br>
        <input type="checkbox" name="area" value="Mobile Application">Mobile
Application<br>
        <input type="checkbox" name="area" value="JSP">JSP<br>

        <input type="submit" name="ok" value="submit">
    </form>
</div>
</body>
</html>

```

Program (pr6_fetch_html_multiple.java / servlet)

```

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 pr6_fetch_html_multiple extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();

        //getParameter() is used to fetch single value from HTML file.
        String username=request.getParameter("uname");
        out.println("username is:"+username);
        out.println("<br>");

        String password=request.getParameter("password");
        out.println("password is:"+password);
        out.println("<br>");

        String gender=request.getParameter("gender");
        out.println("gender is:"+gender);
        out.println("<br>");
    }
}

```

```

//getParameterValues() is used to fetch multiple values for checkbox
String area[]=request.getParameterValues("area");
for (int i=0; i<area.length;i++)
{
    out.println("selected values are:"+area[i]);
}
}
}

```

Output

localhost:8080/SERVLET_Chapter6/checkbox_html.html

username:

password:

Gender: ☒ Male ☐ Female

Area of Interest: ☒ Java

☒ Advanced Java

☐ Mobile Application

☒ JSP

localhost:8080/SERVLET_Chapter6/pr6_fetch_html_multiple?unar

username is:advanceJava
 password is:123
 gender is:male
 selected values are:Java selected values are:Advanced Java selected values are:JSP

Write a program to demonstrate the use of `sendRedirect ()` method of servlet.

In this program first we will fetch HTML username, password values and inside the servlet we are going to check that values with some fix value. If anything goes wrong than redirect the response of first servlet to another servlet to display the message like “U r not authorized user”.

HTML File

username:

password:

Gender: ☐ Male ☐ Female

Program (pr7_redirect1.java / servlet)

```
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 pr7_redirect1 extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        out.println("hello");

        String username=request.getParameter("uname");
        out.println("username is:"+username);
        out.println("<br>");

        String password=request.getParameter("password");
        out.println("password is:"+password);
        out.println("<br>");

        String gender=request.getParameter("gender");
        out.println("gender is:"+gender);
        out.println("<br>");

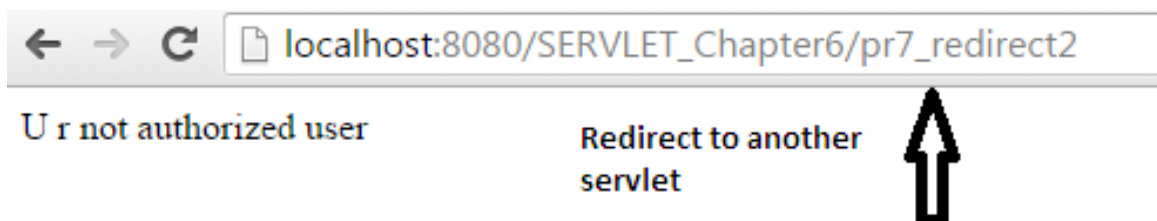
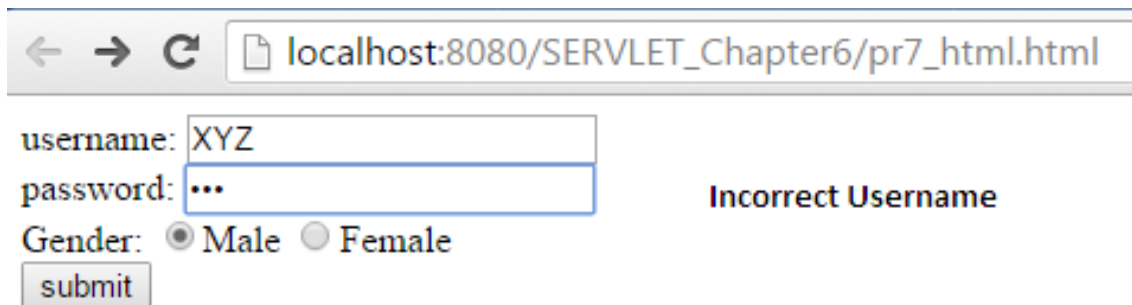
        if(username.equals("abc")&&password.equals("123"))
        {
            out.println("u r welcome");
        }
        else
        {
            response.sendRedirect("/SERVLET_Chapter6/pr7_redirect2");
        }
    }
}
```

pr7_redirect2.java (Second Servlet)

```
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 pr7_redirect2 extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        out.println("U r not authorized user");
    }
}
```

Output



Write a program using servlet to check whether any session is existed or not. If not it will create for me.

As we know that the Http is a stateless protocol, means that it can't persist the information. It always treats each request as a new request. In Http client makes a connection to the server, sends the request, gets the response, and closes the connection.

In session management client first make a request for any servlet or any page, the container receives the request and generate a unique session ID and gives it back to the client along

with the response. This ID gets stores on the client machine. Thereafter when the client request again sends a request to the server then it also sends the session Id with the request. There the container sees the Id and sends back the request.

In this program we are going to make one servlet on session in which we will check whether the session is new or old.

To make this program firstly we need to make one class. Inside the doGet() method, which takes two objects one of request and second of response. Inside this method call the method getWriter () of the response object. Use getSession () of the request object, which returns the HttpSession object. Now by using the HttpSession we can find out whether the session is new or old.

Program (pr8_session_check.java / Servlet)

```
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 pr8_session_check extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();

        //create the session
        HttpSession session=request.getSession();

        //check whether the session is new or old
        if(session.isNew())
        {
            out.print("session created first time");
        }
        else
        {
            out.print("session has already created");
        }
    }
}
```

```
}}
```

Output



After you press refresh button you can see the following output.



Note: If you close your browser tab and reopen with same servlet program still can see the output like “session has already created” because still your session is alive. Once you close your browser than and then your session has been destroyed. So to create a new and fresh session we need to close our browser

Write a program to check different methods of session available in servlet to handle the session

Sometime while developing web application it is necessary to interact with the different values of the Session object. In this example we will explore the different values of the Session object and then learn how to use it in our programming code.

You will learn how to find all the session related information like:

- `getId`. This method is used to find the identifier of the session which is unique.
- `isNew`. This method is used when find, whether session is newly created or preexisted. If session has never seen by user then this method return "true" but if session is preexisted then it return "false".
- `getCreationTime`. This method is used to find the creation time of session. To use of this method we can find the following details about session i.e. day, month, date, time, GMT (Greenwich Mean Time) and year will be displayed.
- `getLastAccessedTime`. This method is used to find the last accessed time of session. It returns the time, in milliseconds.
- `getMaxInactiveInterval`. This method returns the total time, in seconds, during which session remains active if user does not accesses the session for this maximum time interval. After this time the session will be invalidated automatically. A negative value indicates that the session should never timeout.

Program (pr10_session_all_method.java / servlet)

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

```

import java.io.PrintWriter;
import java.util.Date;
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 pr10_session_all_method extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();

        //create the session
        HttpSession session=request.getSession();

        //check whether the session is new or old
        if(session.isNew())
        {
            out.print("session created first time");
            out.print("<br>");
        }
        else
        {
            out.print("session has already created");
            out.print("<br>");
        }

        out.print("session id is:"+session.getId());
        out.print("<br>");
        out.println("session creation time is:"+new Date(session.getCreationTime()));
        out.print("<br>");
        session.setMaxInactiveInterval(10);
        out.print("session last accessed time is:"+new Date(session.getLastAccessedTime()));

    }
}

```

Output

session has already created
session id is:3bfb1a4b0c0f20564125a6d9b8f5
session creation time is:Wed Aug 10 14:49:17 IST 2016
session last accessed time is:Wed Aug 10 14:51:59 IST 2016

Write a program to demonstrate the use of RequestDispatcher class of servlet for redirecting the response of servlet.

In this program we are using the concept of RequestDispatcher for redirecting the response of one servlet to another servlet. Actually this example also shows the difference between sendRedirect () method and forward () method of RequestDispatcher class. You can refer RequestDispatcher class concept in detail in servlet chapter.

HTML File

```
<!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>
    <div>
      <form name="f1" action="pr11_dispatcher" method="get">
        username: <input type="text" name="uname"><br>

        <input type="submit" name="ok" value="submit">
      </form>
    </div>
  </body>
</html>
```

Program (pr11_dispatcher.java / servlet)

```
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.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class pr11_dispatcher extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        out.println("hello in first servlet");

        String username=request.getParameter("uname");
        out.println("username is:"+username);
        out.println("<br>");

        if(username.equals("abc"))
        {
            out.println("u r welcome");
            RequestDispatcher rd=request.getRequestDispatcher("pr11_dispatcher2");
            rd.include(request, response);
        }
        else
        {
            response.sendRedirect("/SERVLET_Chapter6/pr7_redirect2");
        }
    }
}

```

pr11_dispatcher2.java

```

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 pr11_dispatcher2 extends HttpServlet {

    @Override

```

```

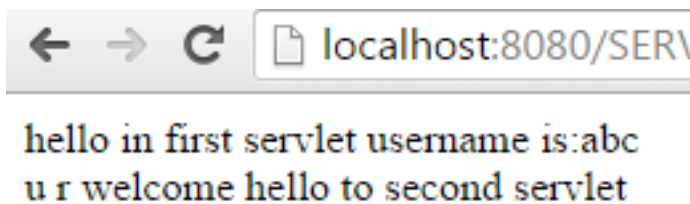
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out=response.getWriter();
    out.println("hello to second servlet");
}
}

```

Output



Now enter the username: abc



Include () output

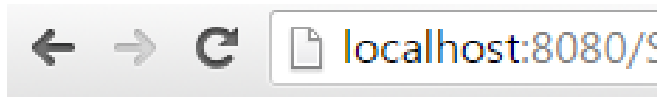
now instead of include() if we are going to use forward () like below

```

if(username.equals("abc"))
{
    out.println("u r welcome");
    RequestDispatcher rd=request.getRequestDispatcher("pr11_dispatcher2");
    rd.forward(request, response);
}

```

Then the output of above program is



hello to second servlet

forward () method

Write a program to pass HTML parameter to servlet and store that parameter in the session.

In this example we are setting the value of session using `setAttribute ()` and after setting the value to fetch that value we need to use `getAttribute ()` method of `HttpSession`.

HTML File

```
<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>
  <div>
    <form name="f1" action="pr12_set_session_value" method="get">
      username: <input type="text" name="uname"><br>

      <input type="submit" name="ok" value="submit">
    </form>
  </div>
</body>
</html>
```

Program (pr12_set_session_value.java / servlet)

```
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 pr12_set_session_value extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
```

```

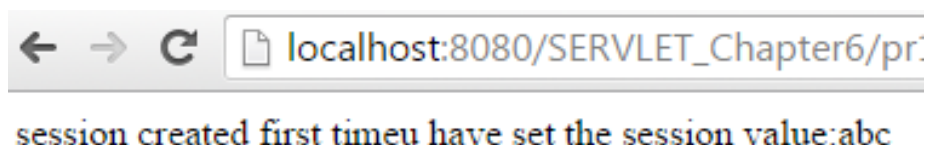
response.setContentType("text/html");
PrintWriter out=response.getWriter();

//create the session
HttpSession session=request.getSession();

//check whether the session is new or old
if(session.isNew())
{
    out.print("session created first time");
}
else
{
    out.print("session has already created");
}
String str1=request.getParameter("uname");
session.setAttribute("username", str1);
out.println("u have set the session value:"+session.getAttribute("username"));
}
}

```

Output

Write a program to demonstrate the use of ServletContextListener in servlet.

Before going into the details of ServletContextListener we should understand what ServletContext is. ServletContext is an interface which helps us to communicate with the servlet container. There is only one ServletContext for the entire web application and the

components of the web application can share it. The information in the ServletContext will be common to all the components. Remember that each servlet will have its own ServletConfig. The ServletContext is created by the container when the web application is deployed and after that only the context is available to each servlet in the web application.

ServletContextListener is an interface which contains two methods:

- public void contextInitialized(ServletContextEvent event)
- public void contextDestroyed(ServletContextEvent event)

When we implement any interface then we have to implement its all methods. This listener will help application to start and shutdown the events.

How ServletContextListener works?

1. ServletContextListener is notified when the context is initialized.

- ServletContextListener gets the context init parameters from the ServletContext.
- It stores the database connection as an attribute, so that the other components in the web application can access it.

2. It will be notified when the context is destroyed. It closes the database connection.

Program (listener class) - context_listener.java

```
import javax.servlet.ServletContext;
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;

public class context_listener implements ServletContextListener {

    @Override
    public void contextInitialized(ServletContextEvent sce) {
        System.out.println("created");
        ServletContext sc=sce.getServletContext();
        String str=sc.getInitParameter("college_name");
        sc.setAttribute("collegename", str);
    }

    @Override
    public void contextDestroyed(ServletContextEvent sce) {
        System.out.println("destroyed");
    }
}
```

Servlet file - pr13_context.java

```
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;
import javax.servlet.http.HttpSession;

public class pr13_context extends HttpServlet {

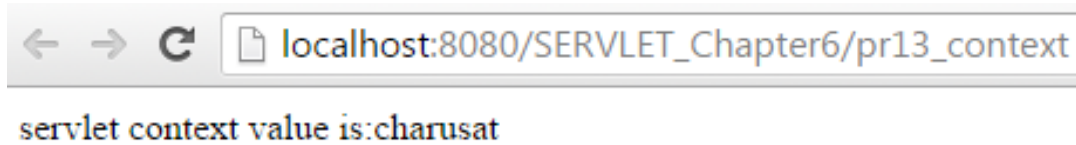
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        ServletContext sc=getServletContext();
        out.print("servlet context value is:"+sc.getAttribute("collegename"));

    }
}
```

web.xml (Deployment Descriptor file)

```
<web-app>
  <context-param>
    <param-name>college_name</param-name>
    <param-value>charusat</param-value>
  </context-param>
  <listener>
    <description>ServletContextListener</description>
    <listener-class>context_listener</listener-class>
  </listener>
  <servlet>
    <servlet-name>pr13_context</servlet-name>
    <servlet-class>pr13_context</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>pr13_context</servlet-name>
    <url-pattern>/pr13_context</url-pattern>
  </servlet-mapping>
  <session-config>
  </web-app>
```

Output



Write a program to demonstrate the use of ServletContextAttributeListener in servlet.

The listener `ServletContextAttributeListener` is an interface and extends the `java.util.EventListener` class. This listener comes into existence when this interface receives notifications of changes to the attribute list on the servlet context of a web application. Remember one thing before gets notified by the container, you should make sure that the implementation class is configured in deployment descriptor for the web application. This listener is used when we want to know when an attribute has been added in a context, when an attribute has been removed and when it is replaced by another attribute. We can also say that this attribute is used when the developer is interested to be notified when the context attribute changes. Now you may be wondering what an attribute is. An attribute is an object set or you can simply say that it is a name/value pair where the name refers to a `String` and a value refers to the `Object`.

`javax.servlet.ServletContextAttributeListener` interface has the following methods:

- `attributeAdded (ServletContextAttributeEvent event)`: It notifies whenever a new attribute is added to the servlet context.
- `attributeRemoved (ServletContextAttributeEvent event)`: It notifies whenever the attribute is removed from the servlet context.
- `attributeReplaced (ServletContextAttributeEvent event)`: It notifies whenever the attribute gets replaced on the servlet context.

In the above methods you can see that we have used `ServletContextAttributeEvent` class as a parameter to the above methods. This class is an event class which is used for notifications when the changes are made to the attributes of `ServletContext` in an application.

The class `ServletContextAttributeEvent` has two methods:

- `getName ()`: This method returns the name of the attribute that has been changed on the `ServletContext`.
- `getValue ()`: This method will return the value of the attribute that has been added, removed or replaced by other attribute.

Program (Listener class only)

```
import com.sun.corba.se.spi.presentation.rmi.StubAdapter;  
import javax.servlet.ServletContext;  
import javax.servlet.ServletContextAttributeEvent;
```

```

import javax.servlet.ServletContextAttributeListener;

public class context_attribute implements ServletContextAttributeListener {

    ServletContext sc;

    @Override
    public void attributeAdded(ServletContextAttributeEvent event) {
        sc=event.getServletContext();
        sc.setAttribute("username", "hello");
        System.out.println("attribute added");
    }

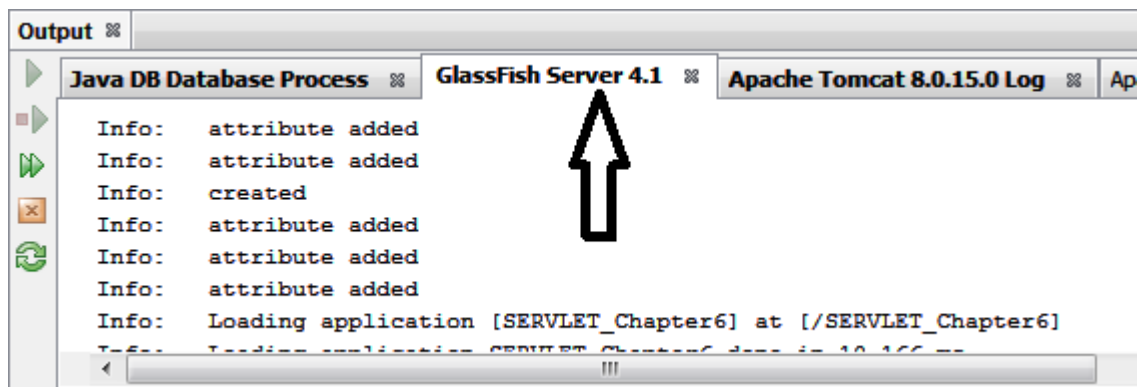
    @Override
    public void attributeRemoved(ServletContextAttributeEvent event) {
        sc=event.getServletContext();
        sc.removeAttribute("username");
        System.out.println("attribute removed");
    }

    @Override
    public void attributeReplaced(ServletContextAttributeEvent event) {
        sc=event.getServletContext();
        sc.setAttribute("username", "hi");
        System.out.println("attribute replaced");
    }
}

```

Output

You can check the server log file in NetBeans for the output of all listener like below



Write a program to demonstrate the use of HttpSessionListener in Servlet.

Listeners listen the events. Whenever any event occurs it is listened by the listener. The listener will be controller by the web servers.

HttpSessionListener is an interface which extends java.util.EventListener class. The main purpose of this listener is to notify whenever there is a change in the list of active sessions in a web application

This interface has two methods:

- sessionCreated(HttpSessionEvent event): It will notify when the session is created.
- sessionDestroyed(HttpSessionEvent event): It will notify when the session gets invalidated.

In the above methods we can see that we have used HttpSessionEvent as the parameter of both the methods. This class has only one method getSession() which returns the current session.

Program (listener class)

```
import javax.servlet.http.HttpSession;
import javax.servlet.http.HttpSessionEvent;
import javax.servlet.http.HttpSessionListener;

public class httpsession implements HttpSessionListener {

    @Override
    public void sessionCreated(HttpSessionEvent se) {
        HttpSession sc=se.getSession();
        System.out.println("session created");
    }

    @Override
    public void sessionDestroyed(HttpSessionEvent se) {
        System.out.println("session destroyed");
    }
}
```

Servlet file (session_list_demo.java)

```
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;
```

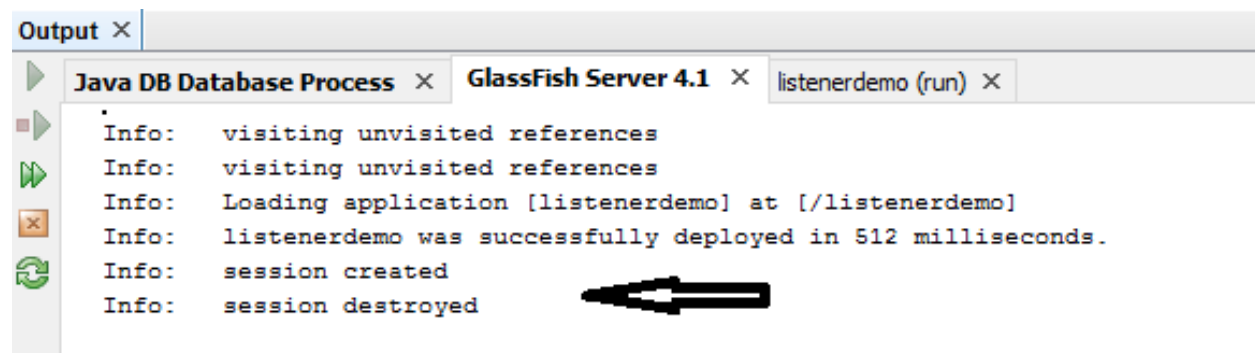
```

public class session_list_demo extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        //at this point sessionCreated() called
        HttpSession sc=request.getSession();
        sc.invalidate(); //sessionDestroyed() called
    }
}

```

Output



Write a program to demonstrate the use of HttpSessionAttributeListener in Servlet.

The listener `HttpSessionAttributeListener` is an interface and extends the `java.util.EventListener` class. This listener will be called by the container whenever there will be change to the attribute list on the servlet session of a web application. This listener is used when we want to know when a attribute has been added in a session, when a attribute has been removed and when it is replaced by another attribute. We can also say that this attribute is used when the developer is interested to be notified when the session attribute changes. Now you may be wondering what an attribute is. An attribute is an object set or you can simply say that it is name/value pair where the name refers to a `String` and a value refers to the `Object`.

`javax.servlet.http.HttpSessionAttributeListener` interface has following methods:

- `attributeAdded (HttpSessionBindingEvent event)`: It notifies whenever a new attribute is added to the servlet session.
- `attributeRemoved (HttpSessionBindingEvent event)`: It notifies whenever the attribute is removed from the servlet session.
- `attributeReplaced (HttpSessionBindingEvent event)`: It notifies whenever the attribute gets replaced on the servlet session.

In the above methods you can see that we have used HttpSessionBindingEvent class as a parameter to the above methods. This class is an event class which is used for notifications when the changes are made to the attributes of in a session.

The class HttpSessionBindingEvent has two methods:

- getName() : This method returns the name of the attribute that has been change in the session.
- getValue(): This method will return the value of the attribute that has been added, removed or replaced by other attribute.
- getSession(): This method will return the session that has been changed.

Listener file

```
import javax.servlet.http.HttpSession;
import javax.servlet.http.HttpSessionAttributeListener;
import javax.servlet.http.HttpSessionBindingEvent;

public class NewServletListener implements HttpSessionAttributeListener {

    HttpSession sc;
    @Override
    public void attributeAdded(HttpSessionBindingEvent event) {
        sc=event.getSession();
        System.out.println("attribute added in session");
    }

    @Override
    public void attributeRemoved(HttpSessionBindingEvent event) {

        System.out.println("attribute removed from session");
    }

    @Override
    public void attributeReplaced(HttpSessionBindingEvent event) {
        sc.setAttribute("username", "java");
        System.out.println("attribute replaced in session");
    }
}
```

Servlet file (session_attribute.java)

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
```

```

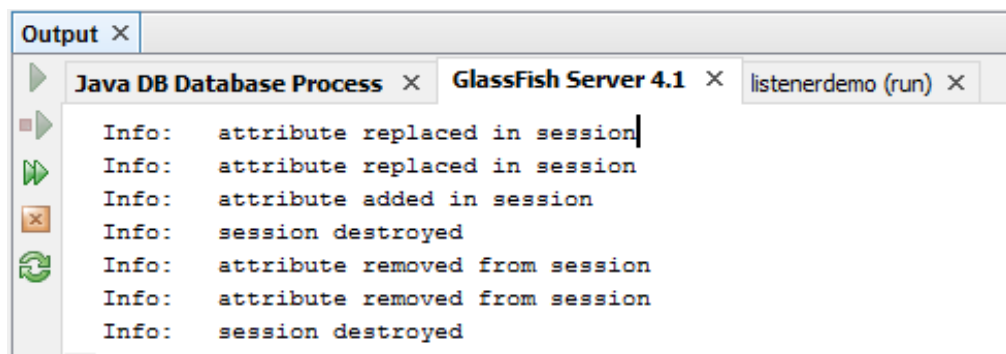
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

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

        response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        HttpSession sc=request.getSession();
        sc.setAttribute("username", "ajava");
        out.println("session value is:"+sc.getAttribute("username"));
        sc.removeAttribute("username");
    }
}

```

Output



Write a program to insert the record in the database table using servlet.

In this program we are going to insert the data in the database from our java program in the table stored in the database. We can insert the data in the database only and only if there is a connectivity between our database and the java program. To establish the connection between our database and the java program we have already seen the basic steps of JDBC in part A. You need to write all JDBC steps within the servlet service method `doGet ()`.

Program (servlet_jdbc_insert.java / servlet file)

```

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.DriverManager;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;

```



```

import javax.servlet.http.HttpServletResponse;
import java.sql.*;

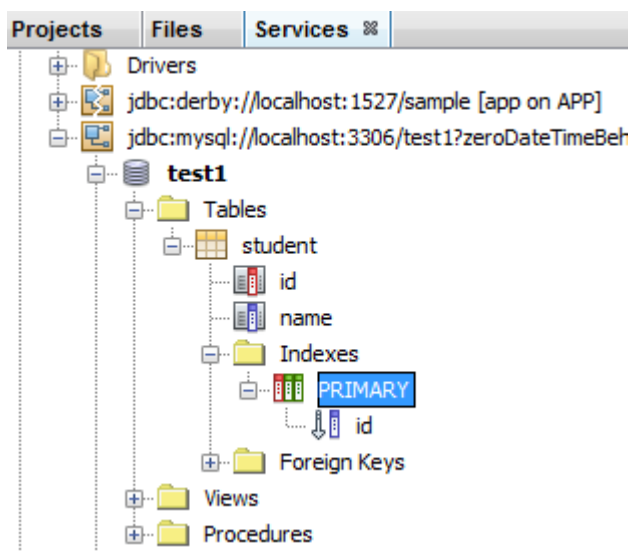
public class servlet_jdbc_insert extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/test1",
"root", "");
            Statement st=con.createStatement();
            String str="INSERT INTO student(id,name)values('1','pinal')";
            st.executeUpdate(str);
            out.print("data inserted in the table");
        } catch (ClassNotFoundException | SQLException ex) {
            Logger.getLogger(servlet_jdbc_insert.class.getName()).log(Level.SEVERE, null, ex);
        }

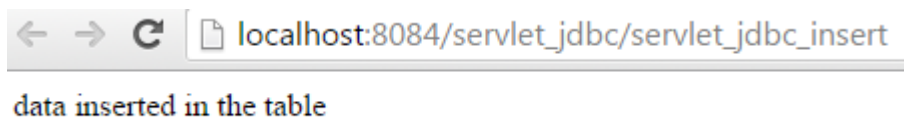
    }
}

```

Database Structure (Database Name: test1, Table name: student, columns: id, name)



Output



Do not forget to add library (MySQL JDBC) in your project.

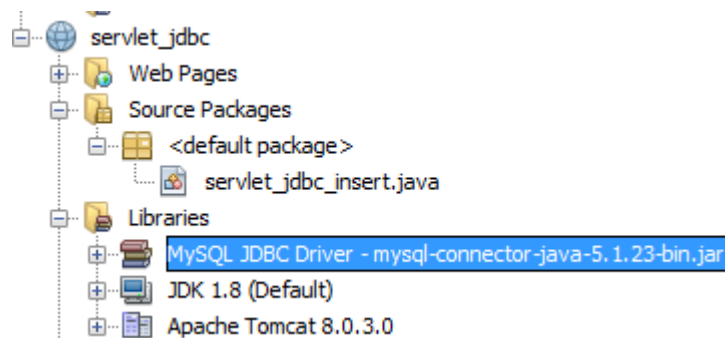


Table Structure after inserting a record in the database table student

select * from student		
Page Size: 20 Total Rows: 1 Page: 1 of 1		
#	id	name
1	1	pinal

Write a program to retrieve the record from the database table using servlet.

In this program we are following the same database structure that we have follow in above program. After inserting some random records now we are trying to retrieve those records from the table using SELECT query of SQL.

Program (servlet_jdbc_select.java / servlet file)

```
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.DriverManager;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.sql.*;
```

```

public class servlet_jdbc_select extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/test1",
"root", "");
            Statement st=con.createStatement();
            String str="SELECT * from student";
            ResultSet rs=st.executeQuery(str);


            while(rs.next())
            {
                String s1=rs.getString("id");
                String s2=rs.getString("name");
                out.println("id is:"+s1+" "+"name is:"+s2);
                out.print("<br>");
            }

        } catch (ClassNotFoundException | SQLException ex) {
            Logger.getLogger(servlet_jdbc_insert.class.getName()).log(Level.SEVERE, null, ex);
        }

    }}

```

Output



id is:1 name is:pinal
 id is:2 name is:jalpesh
 id is:3 name is:sandip

Write a program to insert the HTML data into database through servlet using preparedStatement.

HTML file

```

<!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 name="f1" method="get" action="html_servlet_insert">
    id: <input type="text" name="id">
    <br>
    name: <input type="text" name="name"></br>
    <input type="submit" name="submit" value="ok">

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

Servlet (html_servlet_insert.java)

```
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.DriverManager;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.sql.*;

public class html_servlet_insert extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        String ids=request.getParameter("id");
        String names=request.getParameter("name");
        try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/test1",
"root", "");
            String str="INSERT INTO student(id,name)values(?,?)";
            PreparedStatement pst=con.prepareStatement(str);
```

```

        pst.setString(1, ids);
        pst.setString(2, names);
        pst.executeUpdate();
        out.print("data inserted in the table");
    } catch (ClassNotFoundException | SQLException ex) {
        Logger.getLogger(servlet_jdbc_insert.class.getName()).log(Level.SEVERE, null, ex);
    }
}
}}

```

Output

The screenshot shows a web browser at `localhost:8084/servlet_jdbc/servlet_insert.html`. It contains a form with two input fields: "id:" with the value "4_html" and "name:" with the value "new_html". Below the fields is an "ok" button. Below the form, the browser address bar shows `localhost:8084/servlet_jdbc/html_servlet_insert?id=4_html&name=new_html&submit=ok`. The page content displays "data inserted in the table". Below this, there is a table viewer showing the results of a query "select * from student". The table has columns "#", "id", and "name". It contains four rows of data.

#	id	name
1	1	pinal
2	2	jalpesh
3	3	sandip
4	4_html	new_html

Write a program in servlet to fetch total number of columns count and column name of the table.

Consider a situation where there is a need to know about the name of the columns without touching our database. As we are the programmers so why we need to worry about the database. We want to do the manipulation by sitting on our computer through our program without going into the database.

After the establishment of the connection with the database pass a query for retrieving all the records from the database and this will return the PreparedStatement object. To get the column names from the database we firstly need a reference of ResultSetMetaData object and we will get it only when if we have the ResultSet object. To get the object of the ResultSet we will call the method `executeQuery ()` of the PreparedStatement interface. Now we have the object of the ResultSet. By the help of the ResultSet we can get the object of

ResultSetMetaData. We will get it by calling the method getMetaData () of the ResultSet interface. The names of the columns will be retrieved by the method getColumnNames () of the ResultSetMetaData interface. The output will be displayed to you by the PrintWriter object.

Program (column_name.java / servlet file)

```
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.DriverManager;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.sql.*;

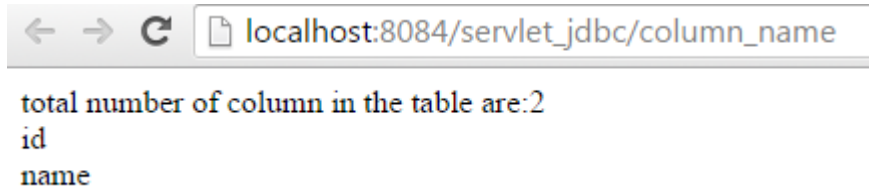
public class column_name extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/test1",
"root", "");
            Statement st=con.createStatement();
            String str="SELECT * from student";
            ResultSet rs=st.executeQuery(str);
            ResultSetMetaData rsmd=rs.getMetaData();
            int column_count=rsmd.getColumnCount();
            out.println("total number of column in the table are:"+column_count);
            out.println("<br>");
            for(int i=1;i<=column_count;i++)
            {
                String names=rsmd.getColumnName(i);
                out.println(names);
                out.println("<br>");
            }

        } catch (ClassNotFoundException | SQLException ex) {
```

```
        Logger.getLogger(servlet_jdbc_insert.class.getName()).log(Level.SEVERE, null, ex);
    }
}}
```

Output



localhost:8084/servlet_jdbc/column_name

total number of column in the table are:2
id
name

Similarly you can perform many SQL operations in JDBC and servlet like

- Counting total number of rows
- Add column dynamically in the table
- Delete all rows from the table
- Perform natural join operation
- Perform left join operation
- Altering a table
- Changing column name dynamically

10.1 Write a program to print “Hello World!” message in 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>
    <h1>Hello World!</h1>
  </body>
</html>
```

Output



Hello World!

10.2 WAP to demonstrate the use of declarative element in JSP

[illegible]

Output



total value for the count is: 1 2 3 4 5 6

10.3 WAP to demonstrate the use of Scriptlet element in 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>
    <%
      int count=0;
      for(int i=0;i<10;i++)
      {
        count ++;
        out.print("count is:"+count);
        out.print("<br>");
      }
    %>
  </body>
</html>
```

Output



count is:1
count is:2
count is:3
count is:4
count is:5
count is:6
count is:7
count is:8
count is:9
count is:10

10.4 WAP to demonstrate the use of all elements in a single page of JSP.

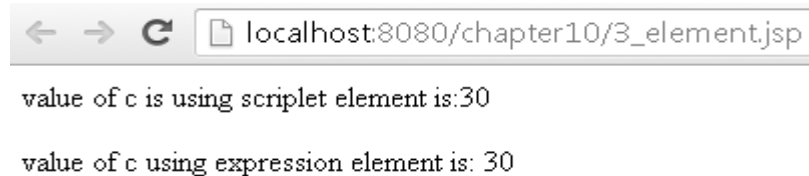
```

<% @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>
    <%!
      int a=10;
      int b=20;
      %>
    <%
      int c=a+b;
      out.println("value of c is using scriptlet element is:"+c);
      %>
    <br>
    <p> value of c using expression element is:
      <%= (a+b)%>

  </body>
</html>

```

Output



value of c is using scriptlet element is:30

value of c using expression element is: 30

10.5 Use of Request (implicit) object

```

<% @page contentType="text/html" pageEncoding="UTF-8" import="java.util.*"%>
<!DOCTYPE html>
<html>

<body>

<p>Request Information:</p>

<table border="0" cellpadding="0" cellspacing="0" width="70%" bgcolor="#EEFFCA">

<tr>

<td width="33%"><b><font color="#800000">Request Method:</font></b></td>

<td width="67%"><font color="#FF0000"><%=request.getMethod()%></font></td>

</tr>

```

```
<tr>

<td width="33%"><b><font color="#800000">Request URI:</font></b></td>

<td width="67%"><font color="#FF0000"><%=request.getRequestURI()%></font></td>

</tr>

<tr>

<td width="33%"><b><font color="#800000">Request Protocol:</font></b></td>

<td width="67%"><font color="#FF0000"><%=request.getProtocol()%></font></td>

</tr>

<tr>

<td width="33%"><b><font color="#800000">Path Info:</font></b></td>

<td width="67%"><font color="#FF0000"><%=request.getPathInfo()%></font></td>

</tr>

<tr>

<td width="33%"><b><font color="#800000">Path translated:</font></b></td>

<td width="67%"><font color="#FF0000"><%=request.getPathTranslated()%></font></td>

</tr>

<tr>

<td width="33%"><b><font color="#800000">Query String:</font></b></td>

<td width="67%"><font color="#FF0000"><%=request.getQueryString()%></font></td>

</tr>

<tr>

<td width="33%"><b><font color="#800000">Content length:</font></b></td>

<td width="67%"><font color="#FF0000"><%=request.getContentLength()%></font></td>

</tr>

<tr>
```

<td width="33%">Content type:</td>	
<td width="67%"><%=request.getContentType()%></td>	
</tr>	
<tr>	
<td width="33%">Server name:</td>	
<td width="67%"><%=request.getServerName()%></td>	
</tr>	
<tr>	
<td width="33%">Server port:</td>	
<td width="67%"><%=request.getServerPort()%></td>	
</tr>	
<tr>	
<td width="33%">Remote user:</td>	
<td width="67%"><%=request.getRemoteUser()%></td>	
</tr>	
<tr>	
<td width="33%">Remote address:</td>	
<td width="67%"><%=request.getRemoteAddr()%></td>	
</tr>	
<tr>	
<td width="33%">Remote host:</td>	
<td width="67%"><%=request.getRemoteHost()%></td>	
</tr>	
<tr>	
<td width="33%">Authorization scheme:</td>	

```

<td width="67%"><font color="#FF0000"><%=request.getAuthType()%></font></td>

</tr>

</table>

</div>

</body>

</html>

```

Output

localhost:8080/chapter10/4_request.jsp

Request Information:

Request Method:	GET
Request URI:	/chapter10/4_request.jsp
Request Protocol:	HTTP/1.1
Path Info:	null
Path translated:	null
Query String:	null
Content length:	-1
Content type:	null
Server name:	localhost
Server port:	8080
Remote user:	null
Remote address:	0:0:0:0:0:0:1
Remote host:	0:0:0:0:0:0:1
Authorization scheme:	null

10.6 WAP to fetch HTML parameter into JSP page.

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 name="f1" method="get" action="5_html.jsp">
      username:<input type="text" name="uname" value="20" size="20" />
      <input type="submit" value="submit" name="submit" />
    </form>
  </body>
</html>

```

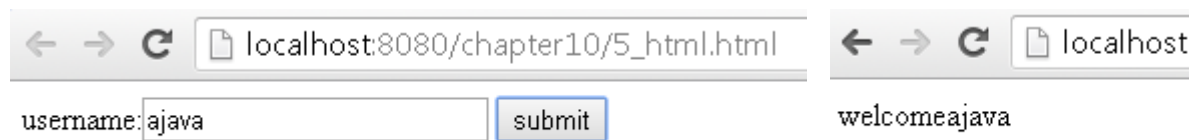
```
</form>
</body>
</html>
```

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>
    <%
      String str=request.getParameter("uname");
      out.println("welcome"+str);
    %>

  </body>
</html>
```

Output



10.7 Store and Retrieve the value of session in JSP using Session Handling Mechanism.

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 name="f1" method="get" action="6_session_1.jsp">
      username:<input type="text" name="uname" value="20" size="20" />
      <input type="submit" value="submit" name="submit" />
    </form>
  </body>
</html>
```

JSP 1

```
<% @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>
    <%
      String username=request.getParameter("uname");
      if(username==null) username="";
      session.setAttribute("username",username);
    %>

    <p><a href="6_session_2.jsp">Next Page to view the session value</a><p>
  </body>
</html>

```

JSP 2

```

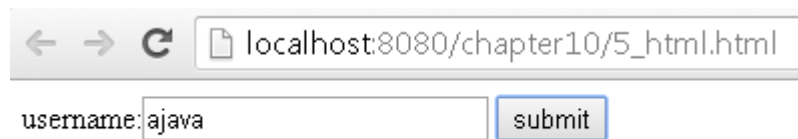
<% @page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <%
      String username=(String) session.getAttribute("username");
      if(username==null) username="";
    %>

    <p>Welcome: <%=username%><p>

  </body>
</html>

```

Output




[Next Page to view the session value](#)

Welcome: ajava

10.8 Managing Cookie in JSP

HTML

```
<html>
```

```

<head>
<title>Cookie Input Form</title>
</head>
<body>
<form method="post" action="7_cookie_1.jsp">
<p><b>Enter Your Name: </b><input type="text" name="username"><br>
<input type="submit" value="Submit">

</form>

</body>
</html>

```

JSP 1

```

<% @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>
    <%
      String username=request.getParameter("username");
      if(username==null) username="";
      Cookie cookie = new Cookie ("username",username);
      cookie.setMaxAge(365 * 24 * 60 * 60);
      response.addCookie(cookie);

      %>
    <p><a href="7_cookie_2.jsp">Next Page to view the cookie value</a><p>

  </body>
</html>

```

JSP 2

```

<% @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>
    <%
      String cookieName = "username";
      Cookie cookies [] = request.getCookies ();
      Cookie myCookie = null;
      if (cookies != null)
      {

```



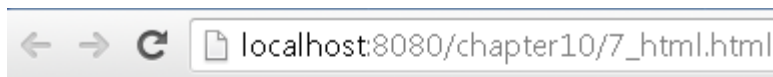
```

        for (int i = 0; i < cookies.length; i++)
        {
            if (cookies [i].getName().equals (cookieName))
            {
                myCookie = cookies[i];
                break;
            }
        }
    }
    %>

    <%
if (myCookie == null) {
%>
No Cookie found with the name <%=cookieName%>
<%
} else {
%>
<p>Welcome: <%=myCookie.getValue()%>.
<%
}
%>
    </body>
</html>

```

Output



Enter Your Name:



[Next Page to view the cookie value](#)

Welcome: ajava.

10.9 Combination of JSP and JDBC for inserting a record in the database table.

```

<% @page contentType="text/html" pageEncoding="UTF-8" import="java.sql.*"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <%
            Class.forName("com.mysql.jdbc.Driver");

```

```

        Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/test1","root","admin");
Statement st=con.createStatement();
String str="INSERT INTO student(id,name)values(1,'pinal)";
st.executeUpdate(str);
out.print("data inserted successfully");

    %>
</body>
</html>

```

10.10 Demonstrate the Java Bean Component in JSP

Bean Class (Details.java)

```

package com.me;

public class Details {
    public Details() {
    }
    private String username;
    private int age;
    private String password;
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
}

```

JSP 1

```

<% @page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head><title>useBean, getProperty and setProperty example</title></head>
<form action="9_bean_jsp_2.jsp" method="post">

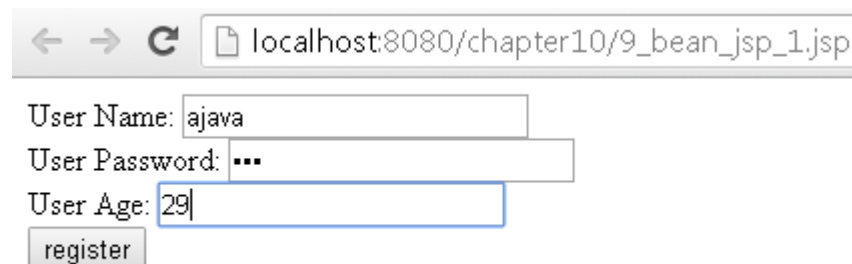
```

```
User Name: <input type="text" name="username"><br>
User Password: <input type="password" name="password"><br>
User Age: <input type="text" name="age"><br>
<input type="submit" value="register">
</form>
</html>
```

JSP 2

```
<%@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>
    <jsp:useBean id="userinfo" class="com.me.Details"></jsp:useBean>
    <jsp:setProperty property="*" name="userinfo"/>
    You have entered below details:<br>
    <jsp:getProperty property="username" name="userinfo"/><br>
    <jsp:getProperty property="password" name="userinfo"/><br>
    <jsp:getProperty property="age" name="userinfo" /><br>
  </body>
</html>
```

Output

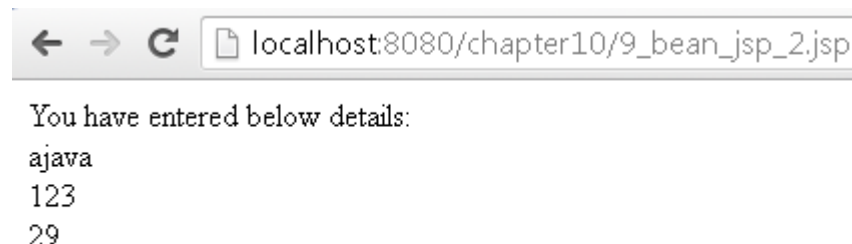


← → ↻ 📄 localhost:8080/chapter10/9_bean_jsp_1.jsp

User Name:

User Password:

User Age:



← → ↻ 📄 localhost:8080/chapter10/9_bean_jsp_2.jsp

You have entered below details:

ajava

123

29

10.11 WAP to insert record in the database table using JSTL SQL Tag Library.

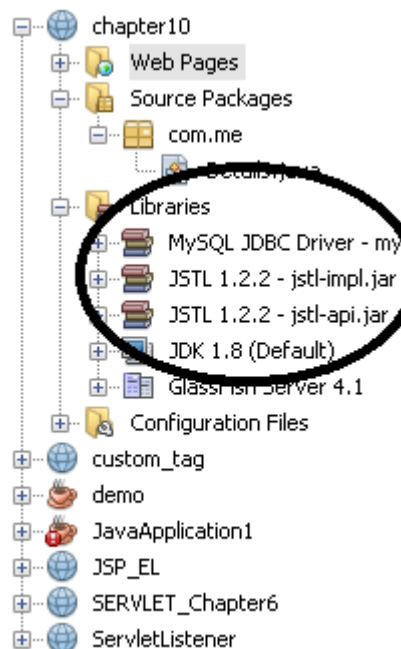
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>
```

```

<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <sql:setDataSource driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost:3306/test1" user="root" password="admin" var="con" />
    <sql:update dataSource="${con}" var="test">
      INSERT INTO student(id,name)VALUES(3,'jalpesh');
    </sql:update>
  </body>
</html>

```

Do not forget to add JSTL library in your web application execution path like below.



10.12 Fetch the record from the database table using JSTL SQL Tag library.

```

<% @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>
    <sql:setDataSource driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost:3306/test1" user="root" password="admin" var="con" />
    <sql:query dataSource="${con}" var="result">
      SELECT * from student;
    </sql:query>
  </body>
</html>

```

```

<table border="1" width="100%">
  <tr>
    <th>id</th>
    <th>Name</th>
  </tr>

  <c:forEach var="row" items="${result.rows}">

    <tr>
      <td><c:out value="${row.id}"/></td>
      <td><c:out value="${row.name}"/></td>
    </tr>

  </c:forEach>
</table>
</body>
</html>

```

10.13 Login Application using Servlet, JSP and JDBC

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>LOGIN Page</title>
  </head>
  <body>
    <form action="LoginServlet" method="post">
      <fieldset style="width: 300px">
        <legend> Login to App </legend>
        <table>
          <tr>
            <td>User ID</td>
            <td><input type="text" name="username" required="required" /></td>
          </tr>
          <tr>
            <td>Password</td>
            <td><input type="password" name="userpass" required="required" /></td>
          </tr>
          <tr>
            <td><input type="submit" value="Login" /></td>
          </tr>
        </table>
      </fieldset>
    </form>
  </body>
</html>

```

LoginServlet.java (servlet file)

```
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.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class LoginServlet extends HttpServlet{

    private static final long serialVersionUID = 1L;

    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        String n=request.getParameter("username");
        String p=request.getParameter("userpass");

        HttpSession session = request.getSession(false);
        if(session!=null)
            session.setAttribute("name", n);

        if(LoginDao.validate(n, p)){
            RequestDispatcher rd=request.getRequestDispatcher("welcome.jsp");
            rd.forward(request,response);
        }
        else{
            out.print("<p style=\"color:red\">Sorry username or password error</p>");
            RequestDispatcher rd=request.getRequestDispatcher("index.jsp");
            rd.include(request,response);
        }

        out.close();
    }
}
```

LoginDao.java (Validation Java File)

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
```

```

public class LoginDao {
    public static boolean validate(String name, String pass) {
        boolean status = false;
        Connection conn = null;
        PreparedStatement pst = null;
        ResultSet rs = null;

        String url = "jdbc:mysql://localhost:3306/";
        String dbName = "form";
        String driver = "com.mysql.jdbc.Driver";
        String userName = "root";
        String password = "";
        try {
            Class.forName(driver).newInstance();
            conn = DriverManager
                .getConnection(url + dbName, userName, password);

            pst = conn
                .prepareStatement("select * from login where user=? and password=?");
            pst.setString(1, name);
            pst.setString(2, pass);

            rs = pst.executeQuery();
            status = rs.next();

        } catch (ClassNotFoundException | IllegalAccessException | InstantiationException |
SQLException e) {
            System.out.println(e);
        } finally {
            if (conn != null) {
                try {
                    conn.close();
                } catch (SQLException e) {
                }
            }
            if (pst != null) {
                try {
                    pst.close();
                } catch (SQLException e) {
                }
            }
            if (rs != null) {
                try {
                    rs.close();
                } catch (SQLException e) {
                }
            }
        }
        return status;
    }
}

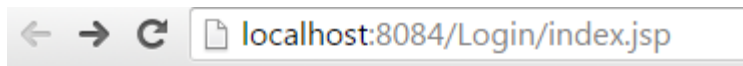
```

```
}
```

Welcome.jsp

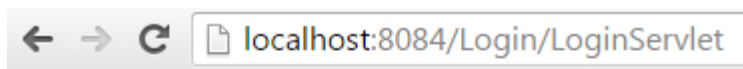
```
<% @page contentType="text/html" pageEncoding="UTF-8"% >
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <h3>Login successful!!!</h3>
    <h4>
      Hello,
      <%=session.getAttribute("name")%></h4>
    </body>
  </html>
```

Output



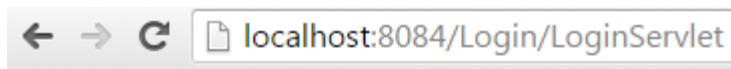
Login to App

User ID	<input type="text" value="admin"/>
Password	<input type="password" value="....."/>
<input type="button" value="Login"/>	



Login successful!!!

Hello, admin



Sorry username or password error

Login to App	
User ID	<input type="text"/>
Password	<input type="password"/>
<input type="button" value="Login"/>	

10.14 WAP to upload an image into database table.

Uploadimage.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>
    <h1>Upload any image to mysql Database</h1>
    <form method="post" action="UploadServlet"
      enctype="multipart/form-data">
      <table>
        <tr>
          <td>First Name:</td>
          <td><input type="text" name="firstName" size="10"
            required="required" /></td>
        </tr>
        <tr>
          <td>Last Name:</td>
          <td><input type="text" name="lastName" size="10"
            required="required" /></td>
        </tr>
        <tr>
          <td>Choose Image:</td>
          <td><input type="file" name="photo" size="10"
            required="required" /></td>
        </tr>
        <tr>
          <td><input type="submit" value="Submit"></td>
          <td><input type="reset" value="Clear" /></td>
        </tr>
      </table>
    </form>
  </body>
</html>
```

UploadServlet.java

```
import java.io.IOException;
import java.io.InputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import javax.servlet.ServletException;
import javax.servlet.annotation.MultipartConfig;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.Part;

@WebServlet("/fileUpload")
@MultipartConfig(maxFileSize = 16177215) // upload file up to 16MB
public class UploadServlet extends HttpServlet {

    private static final long serialVersionUID = -1623656324694499109L;
    private Connection conn;

    public UploadServlet() {
        String url = "jdbc:mysql://localhost:3306/";
        String dbName = "appdb";
        String driver = "com.mysql.jdbc.Driver";
        String userName = "root";
        String password = "";
        try {
            Class.forName(driver).newInstance();
            conn = DriverManager
                .getConnection(url + dbName, userName, password);
        }
        catch (ClassNotFoundException | IllegalAccessException | InstantiationException |
SQLException e) {
            System.out.println(e);
        }
    }

    protected void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {

        // gets values of text fields
        String firstName = request.getParameter("firstName");
        String lastName = request.getParameter("lastName");

        InputStream inputStream = null;
```

```

// obtains the upload file part in this multipart request
Part filePart = request.getPart("photo");
if (filePart != null) {
    // debug messages
    System.out.println(filePart.getName());
    System.out.println(filePart.getSize());
    System.out.println(filePart.getContentType());

    // obtains input stream of the upload file
    inputStream = filePart.getInputStream();
}

String message = null; // message will be sent back to client

try {
    // constructs SQL statement
    String sql = "INSERT INTO contacts (first_name, last_name, photo) values (?, ?, ?)";
    PreparedStatement statement = conn.prepareStatement(sql);
    statement.setString(1, firstName);
    statement.setString(2, lastName);

    if (inputStream != null) {
        // fetches input stream of the upload file for the blob column
        statement.setBlob(3, inputStream);
    }

    // sends the statement to the database server
    int row = statement.executeUpdate();
    if (row > 0) {
        message = "Image is uploaded successfully into the Database";
    }
} catch (SQLException ex) {
    message = "ERROR: " + ex.getMessage();
    ex.printStackTrace();
}
// sets the message in request scope
request.setAttribute("Message", message);

// forwards to the message page
getServletContext().getRequestDispatcher("/submit.jsp").forward(
    request, response);
}
}

```

Submit.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>
  <h3><%=request.getAttribute("Message")%></h3>
</body>
</html>

```

Output

localhost:8084/upload_image/uploadImage.jsp

Upload any image to mysql Database

First Name:

Last Name:

Choose Image: login1.png

localhost:8084/upload_image/UploadServlet

Image is uploaded successfully into the Database

select * from contacts ☒				
Page Size: 20 Total Rows: 2 Page: 1 of 1 Matching Rows				
#	contact_id	first_name	last_name	photo
1		1 pinal	shah	<BLOB 10 kB>
2		2 pinal	shah	<BLOB 10 kB>

select * from contacts ☒				
Page Size: 20 Total Rows: 2 Page: 1 of 1 Matching Rows				
#	contact_id	first_name	last_name	photo
1		1 pinal	shah	<BLOB 10 kB>
2		2 pinal	shah	<BLOB 10 kB>

10.15 WAP to send an email using SMTP in JSP.

email.jsp

A form for sending an e-mail message would contain the following fields at least:

- Recipient address: text field.
- Subject: text field.
- Content: text area.

```
<%@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="EmailSendingServlet" method="post">
      <table border="0" width="35%" align="center">
        <caption><h2>Send New E-mail</h2></caption>
        <tr>
          <td width="50%">Recipient address </td>
          <td><input type="text" name="recipient" size="50"/></td>
        </tr>
        <tr>
          <td>Subject </td>
          <td><input type="text" name="subject" size="50"/></td>
        </tr>
        <tr>
          <td>Content </td>
          <td><textarea rows="10" cols="39" name="content"></textarea> </td>
        </tr>
        <tr>
          <td colspan="2" align="center"><input type="submit" value="Send"/></td>
        </tr>
      </table>
    </form>
  </body>
</html>
```

EmailSendingServlet.java (Servlet file)

Now we implement a servlet that does the following tasks:

- Read SMTP server settings from web.xml file.
- Take input from EmailForm.jsp page.
- Invoke the EmailUtility class to send an e-mail message.
- Return a response to the user.

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

```

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;

/**
 * A servlet that takes message details from user and send it as a new e-mail
 * through an SMTP server.
 *
 * @author www.codejava.net
 */
@WebServlet("/EmailSendingServlet")
public class EmailSendingServlet extends HttpServlet {
    private String host;
    private String port;
    private String user;
    private String pass;

    public void init() {
        // reads SMTP server setting from web.xml file
        ServletContext context = getServletContext();
        host = context.getInitParameter("host");
        port = context.getInitParameter("port");
        user = context.getInitParameter("user");
        pass = context.getInitParameter("pass");
    }

    protected void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
        // reads form fields
        String recipient = request.getParameter("recipient");
        String subject = request.getParameter("subject");
        String content = request.getParameter("content");

        String resultMessage = "";

        try {
            EmailUtility.sendEmail(host, port, user, pass, recipient, subject,
                content);
            resultMessage = "The e-mail was sent successfully";
        } catch (Exception ex) {
            ex.printStackTrace();
            resultMessage = "There were an error: " + ex.getMessage();
        } finally {
            request.setAttribute("Message", resultMessage);
            getServletContext().getRequestDispatcher("/Result.jsp").forward(
                request, response);
        }
    }
}

```

```
}  
}  
}
```

As we can see, the servlet, EmailSendingServlet - loads configuration for SMTP server upon initialization in its init() method, and handles user's requests in the doPost() method. It reads message details (recipient, subject and content) from EmailForm.jsp page and tries to sending the e-mail by invoking the sendEmail() method of the EmailUtility class. The users will finally receives a Result.jsp page which tells them whether the e-mail was sent or not.

Emailutility.java (Configure SMTP Server)

The class has one static method, sendEmail() – which takes SMTP server settings and message details as its arguments. We will put SMTP server settings in web.xml file of the application.

```
import java.util.Date;  
import java.util.Properties;  
  
import javax.mail.Authenticator;  
import javax.mail.Message;  
import javax.mail.MessagingException;  
import javax.mail.PasswordAuthentication;  
import javax.mail.Session;  
import javax.mail.Transport;  
import javax.mail.internet.AddressException;  
import javax.mail.internet.InternetAddress;  
import javax.mail.internet.MimeMessage;  
  
/**  
 * A utility class for sending e-mail messages  
 * @author www.codejava.net  
 */  
public class EmailUtility {  
    public static void sendEmail(String host, String port,  
        final String userName, final String password, String toAddress,  
        String subject, String message) throws AddressException,  
        MessagingException {  
  
        // sets SMTP server properties  
        Properties properties = new Properties();  
        properties.put("mail.smtp.host", host);  
        properties.put("mail.smtp.port", port);  
        properties.put("mail.smtp.auth", "true");  
        properties.put("mail.smtp.starttls.enable", "true");  
  
        // creates a new session with an authenticator  
        Authenticator auth = new Authenticator() {  
            public PasswordAuthentication getPasswordAuthentication() {  
                return new PasswordAuthentication(userName, password);  
            }  
        }  
    }  
}
```

```

    }
};

Session session = Session.getInstance(properties, auth);

// creates a new e-mail message
Message msg = new MimeMessage(session);

msg.setFrom(new InternetAddress(userName));
InternetAddress[] toAddresses = { new InternetAddress(toAddress) };
msg.setRecipients(Message.RecipientType.TO, toAddresses);
msg.setSubject(subject);
msg.setSentDate(new Date());
msg.setText(message);

// sends the e-mail
Transport.send(msg);

}
}

```

Result.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>
      <h3><%=request.getAttribute("Message")%></h3>
    </center>
  </body>
</html>

```

Output

Send New E-mail

Recipient address	<input type="text" value="pinalcp604@gmail.com"/>
Subject	<input type="text" value="Say Hello"/>
Content	<div><div>hi...how r U?</div></div>
<input type="button" value="Send"/>	

10.16 Demonstrate the use of JSTL function library.

function.jsp

```
<% @page contentType="text/html" pageEncoding="UTF-8"%>
<% @ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<% @ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
<body>
<h1>substring function</h1>
<c:set var="string1" value="This is first String."/>
<c:set var="string2" value="{fn:substring(string1, 5, 15)}" />
<p>Final sub string : ${string2}</p>

<h1>indexOf function</h1>
<p>Index (1) : ${fn:indexOf(string1, "first")}</p>
<p>Index (2) : ${fn:indexOf(string2, "second")}</p>

<h1>replace function</h1>
<c:set var="string1" value="This is first String."/>
<c:set var="string2" value="{fn:replace(string1,
'first', 'second')}" />
<p>Final String : ${string2}</p>

<h1>join function</h1>
```

```
<c:set var="string1" value="This is first String."/>
<c:set var="string2" value="{fn:split(string1, ' ')}" />
<c:set var="string3" value="{fn:join(string2, '-')}" />
<p>Final String : {string3}</p>
```

split function</h1>

```
<c:set var="string1" value="This is first String."/>
<c:set var="string2" value="{fn:split(string1, ' ')}" />
<c:set var="string3" value="{fn:join(string2, '-')}" />
<p>String (3) : {string3}</p>
<c:set var="string4" value="{fn:split(string3, '-')}" />
<c:set var="string5" value="{fn:join(string4, ' ')}" />
<p>String (5) : {string5}</p>
```

lowercase/uppercase function</h1>

```
<c:set var="string1" value="This is first String."/>
<c:set var="string2" value="{fn:toLowerCase(string1)}" />
<p>Final string : {string2}</p>
```

trim function</h1>

```
<c:set var="string1" value="This is first String" />
<p>String (1) Length : {fn:length(string1)}</p>
```

```
<c:set var="string2" value="{fn:trim(string1)}" />
<p>String (2) Length : {fn:length(string2)}</p>
<p>Final string : {string2}</p>
```

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

substring function

Final sub string : is first S

indexOf function

Index (1) : 8

Index (2) : -1

replace function

Final String : This is second String.

join function

Final String : This-is-first-String.

split function

String (3) : This-is-first-String.

String (5) : This is first String.

lowercase/uppercase function

Final string : this is first string.

trim function

String (1) Length : 29

String (2) Length : 20

Final string : This is first String