

Pagination in Servlet and JSP

Contents

- [1 Project Description](#)
- [2 Environment Used](#)
- [3 Setting up development environment](#)
- [4 Program control flow](#)
- [5 What is pagination?](#)
 - [5.1 How to achieve pagination?](#)
- [6 Creating Database and table in MySQL](#)
- [7 Creating Dynamic Web Project](#)
 - [7.1 Download MySQL connector](#)
 - [7.2 Download JSTL JAR File](#)
 - [7.3 Writing Transfer Object class](#)
 - [7.4 Writing Connection Factory class](#)
 - [7.5 Writing DAO class](#)
 - [7.6 Writing Servlet](#)
 - [7.7 Writing web.xml](#)
 - [7.8 Writing displayEmployee.jsp](#)
 - [7.9 Folder Structure](#)
- [8 Output](#)

Project Description

- This example explains how to write an application using Servlet and JSP which uses pagination to display the results.
- This application uses one table Employee and displays employee details to the user.
- We use Eclipse IDE for Java EE Developers and Apache Tomcat to run the Servlet and JSP.
- As a best practice, we use **Singleton** (for making database connection), **Data Access Object (DAO)**, **Transfer Object (TO)** and **Model View Controller (MVC)** patterns.

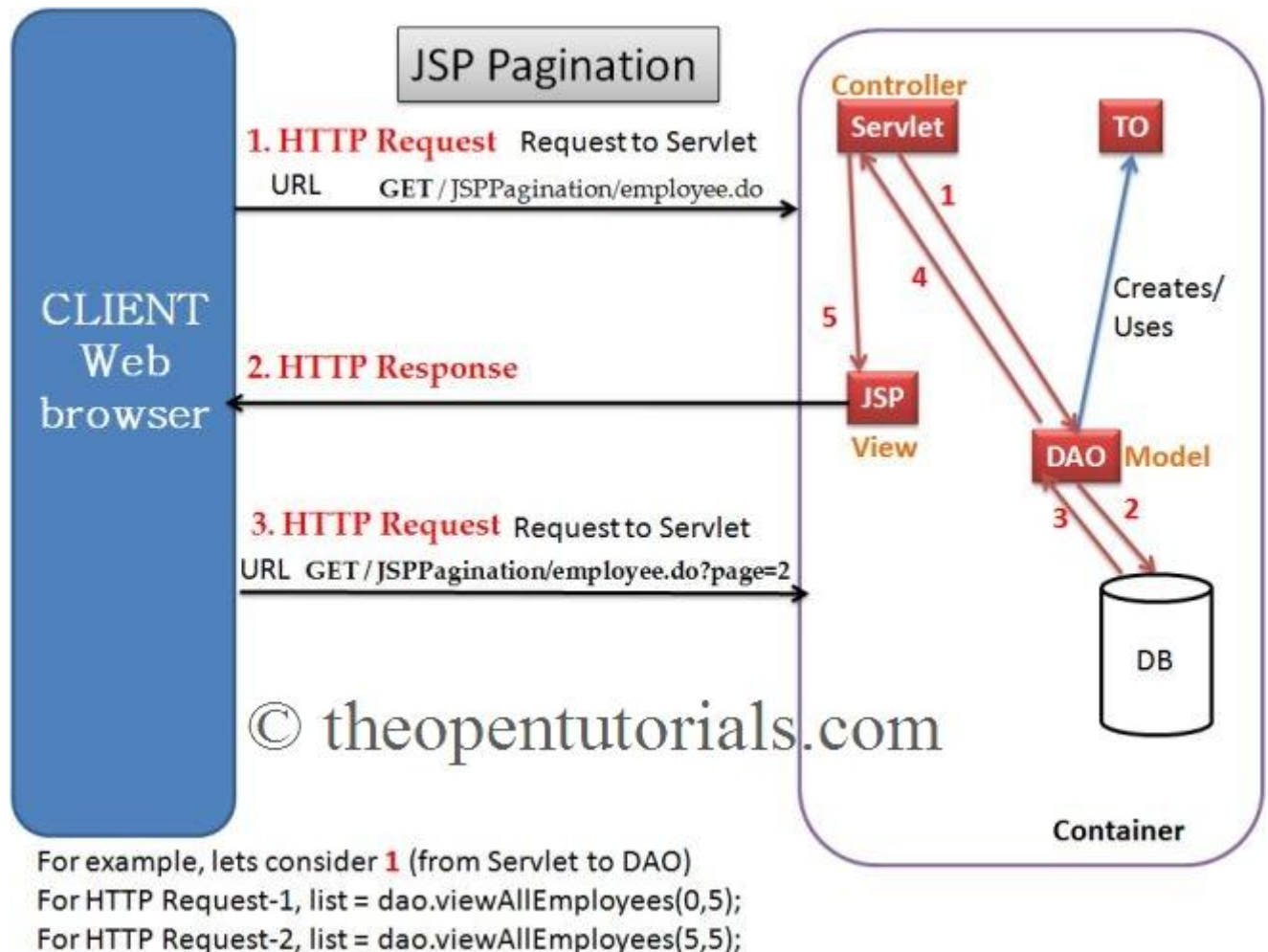
Environment Used

- JDK 6 (Java SE 6)
- Eclipse Indigo IDE for Java EE Developers (3.7.1)
- Apache Tomcat 6.x
- MySQL 5.5
- MySQL Connector/J 5.1 JAR file
- JSTL JAR file (jstl-1.2.jar)
- Java EE 5 API (Servlet 2.5, JSP 2.1, JSTL 1.2, Expression Language (EL))
- Java Database Connectivity (JDBC) API
- [Optional] For monitoring and analyzing HTTP headers between the browser and web servers, you can use one of these add-ons of Firefox
 - Live HTTP Headers
 - HttpFox

Setting up development environment

If you are new to developing Servlet with Tomcat and Eclipse, you can read this page before proceeding with this example.

Program control flow



What is pagination?

- Fetching millions of records from database consumes almost all CPU power and memory of machine.
- Hence we break millions of records into small chunks showing limited number of records (say 20 or 30) per page. The best example of this is Google search pagination which allows user to navigate to next page by page number and explore limited records per pages.

How to achieve pagination?

Pagination logic can be achieved in many ways, some are

Method 1: Greedy approach

- Fetch all records at once and display it to the user after caching the results. This is known as greedy approach.
- This can be achieved by writing a DAO which returns a List<Object>. Whenever the user needs result, the sub list can be retrieved from the cached list instead of querying the database to fetch the next set of results when the user clicks “Next” link.
- Drawback of this approach is that since the data is being cached it becomes stale. If your application changes the data in the result set you may have to consider the accuracy of your results when you choose this solution.

Method 2: Non-greedy approach

- Get range of records each time a user wants to see by limiting the number of rows in the ResultSet.
- What happens if you have more than millions of records? User may have to wait for a long time to get the results. Here, we limit the result set to fetch only number of records the user wants to see.

We use second method to demonstrate pagination.

Creating Database and table in MySQL

This example uses one table Employee and the description of the table is shown below.
‘Employee’ table

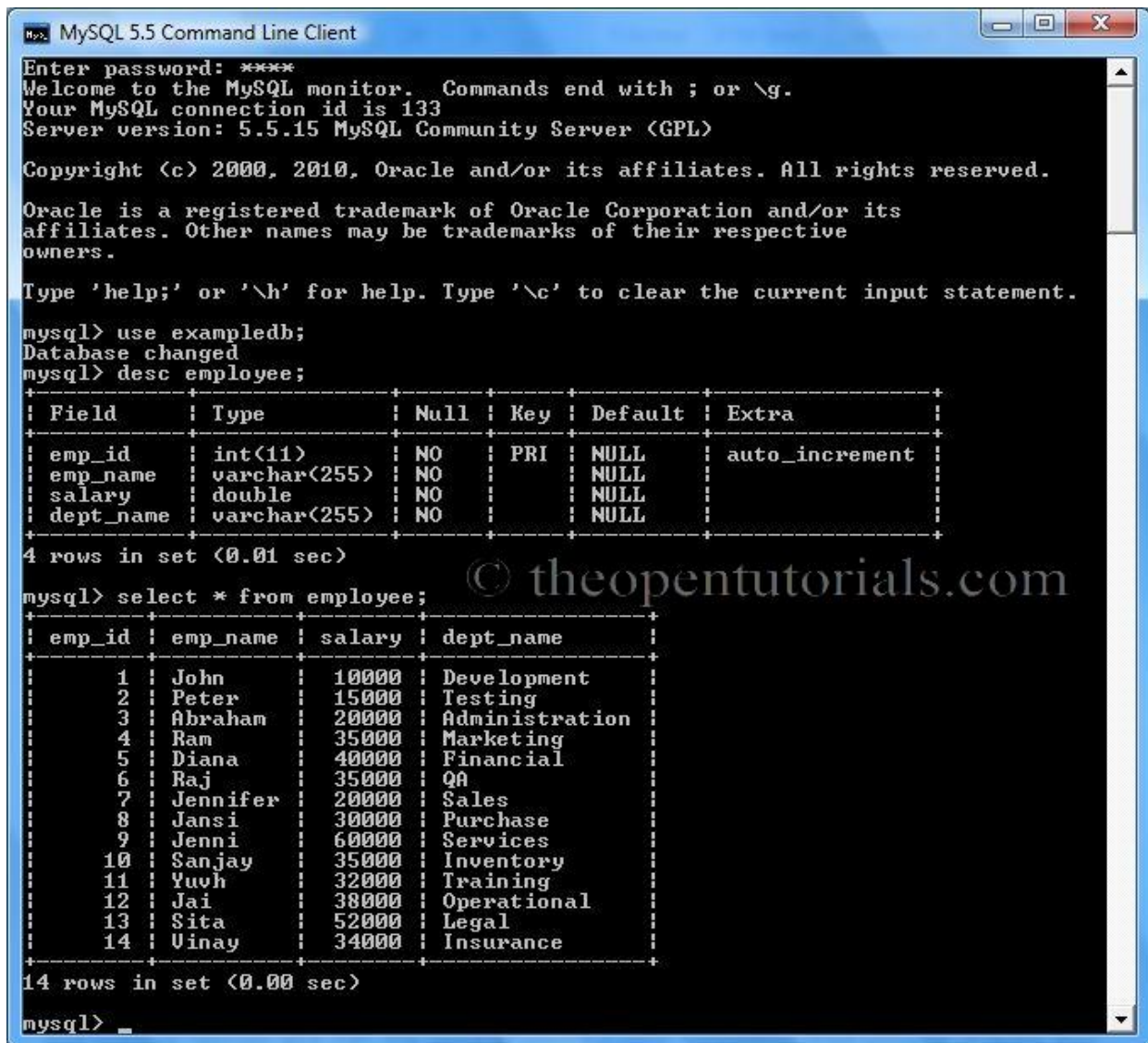
Field	Type	Key	Extra
emp_id	int	Primary Key	auto_increment
emp_name	varchar(255)		
salary	double		
dept_name	varchar(255)		

- Open command prompt (Windows) or Terminal(Linux) and type

```
mysql -u [your-username] -p
```

and press enter and type the password.

- If you are using Windows, you can also use MySQL command line client which will be available in All programs menu.
- For creating a new database, refer this page. In this example, the database name is ‘**exampledb**’.
- After creating the database type the command “use <database_name>;”
- For creating a new table, refer this page. In this example, the table name is ‘**employee**’
- Insert some records in the table. Refer this page for inserting rows using MySQL.



```
MySQL 5.5 Command Line Client
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 133
Server version: 5.5.15 MySQL Community Server (GPL)

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use exampledb;
Database changed
mysql> desc employee;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| emp_id | int(11) | NO | PRI | NULL | auto_increment |
| emp_name | varchar(255) | NO | | NULL | |
| salary | double | NO | | NULL | |
| dept_name | varchar(255) | NO | | NULL | |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.01 sec)

mysql> select * from employee;
+-----+-----+-----+-----+
| emp_id | emp_name | salary | dept_name |
+-----+-----+-----+-----+
| 1 | John | 10000 | Development |
| 2 | Peter | 15000 | Testing |
| 3 | Abraham | 20000 | Administration |
| 4 | Ram | 35000 | Marketing |
| 5 | Diana | 40000 | Financial |
| 6 | Raj | 35000 | QA |
| 7 | Jennifer | 20000 | Sales |
| 8 | Jansi | 30000 | Purchase |
| 9 | Jenni | 60000 | Services |
| 10 | Sanjay | 35000 | Inventory |
| 11 | Yuvi | 32000 | Training |
| 12 | Jai | 38000 | Operational |
| 13 | Sita | 52000 | Legal |
| 14 | Vinay | 34000 | Insurance |
+-----+-----+-----+-----+
14 rows in set (0.00 sec)

mysql> _
```

Creating Dynamic Web Project

- Open Eclipse IDE
- For writing Servlet and JSP, we need to create a new Dynamic Web project. Create this project and name it as “**JSPPagination**”.

Download MySQL connector

- The connector can be downloaded from:
<http://dev.mysql.com/downloads/connector/j/>.
- This tutorial uses 5.1 version. Unzip the connector to a safe location on your computer which contains MySQL Connector J JAR.
- Copy the MySQL Connector J JAR and paste it inside project’s ‘lib’ folder.

Download JSTL JAR File

- JSTL JAR file can be downloaded from: <http://download.java.net/maven/1/jstl/jars/>
- This tutorial uses jstl-1.2.jar file.

- **Copy the JAR file and paste it inside project's 'lib' folder.**

Writing Transfer Object class

- In src folder, create a **new package** and name it as '**com.theopentutorials.to**'. Create new class in this package as shown below.
- Transfer Object encapsulates business data. To implement this pattern, we write a class with properties defining the table attributes.

Employee.java class

```
01
02
03 package com.theopentutorials.to;
04
05 public class Employee {
06     private int employeeId;
07     private String employeeName;
08     private double salary;
09     private String deptName;
10
11     public int getEmployeeId() {
12         return employeeId;
13     }
14     public void setEmployeeId(int employeeId) {
15         this.employeeId = employeeId;
16     }
17     public String getEmployeeName() {
18         return employeeName;
19     }
20     public void setEmployeeName(String employeeName)
21     {
22         this.employeeName = employeeName;
23     }
24     public double getSalary() {
25         return salary;
26     }
27     public void setSalary(double salary) {
28         this.salary = salary;
29     }
30     public String getDeptName() {
31         return deptName;
32     }
33     public void setDeptName(String deptName) {
34         this.deptName = deptName;
35     }
36 }
```

Writing Connection Factory class

Before writing DAO class, let's write a ConnectionFactory class which has database connection configuration statements and methods to make connection to the database. This class uses singleton pattern.

Create a **new package in src folder** and name it as **com.theopentutorials.db** and copy the following code.

ConnectionFactory.java

```
01
02
03 package com.theopentutorials.db;
04
05 import java.sql.Connection;
06 import java.sql.DriverManager;
07 import java.sql.SQLException;
08
09 public class ConnectionFactory {
10     //static reference to itself
11     private static ConnectionFactory instance =
12         new ConnectionFactory();
13     String url = "jdbc:mysql://localhost/EXAMPLEDB";
14     String user = "<YOUR_DATABASE_USERNAME>";
15     String password = "<YOUR_DATABASE_PASSWORD>";
16     String driverClass = "com.mysql.jdbc.Driver";
17
18     //private constructor
19     private ConnectionFactory() {
20         try {
21             Class.forName(driverClass);
22         } catch (ClassNotFoundException e) {
23             e.printStackTrace();
24         }
25     }
26
27     public static ConnectionFactory getInstance() {
28         return instance;
29     }
30
31     public Connection getConnection() throws SQLException,
32     ClassNotFoundException {
33         Connection connection =
34             DriverManager.getConnection(url, user,
35             password);
36         return connection;
37     }
38 }
```

Fill the username and password for your database and enter your database name in the url string.

Writing DAO class

This class uses Data Access Object (DAO) pattern which encapsulates access to the data source.

EmployeeDAO.java

```

01 package com.theopentutorials.dao;
02
03 import java.sql.Connection;
04 import java.sql.ResultSet;
05 import java.sql.SQLException;
06 import java.sql.Statement;
07 import java.util.ArrayList;
08 import java.util.List;
09 import com.theopentutorials.db.ConnectionFactory;
10 import com.theopentutorials.to.Employee;
11
12 public class EmployeeDAO {
13     Connection connection;
14     Statement stmt;
15     private int noOfRecords;
16
17     public EmployeeDAO() { }
18
19     private static Connection getConnection()
20         throws SQLException,
21         ClassNotFoundException
22     {
23         Connection con = ConnectionFactory.
24             getInstance().getConnection();
25         return con;
26     }
27
28     public List<Employee> viewAllEmployees(
29         int offset,
30         int noOfRecords)
31     {
32         String query = "select SQL_CALC_FOUND_ROWS * from employee limit
33 "
34         + offset + ", " + noOfRecords;
35         List<Employee> list = new ArrayList<Employee>();
36         Employee employee = null;
37         try {
38             connection = getConnection();
39             stmt = connection.createStatement();
40             ResultSet rs = stmt.executeQuery(query);
41             while (rs.next()) {
42                 employee = new Employee();
43                 employee.setEmployeeId(rs.getInt("emp_id"));
44                 employee.setEmployeeName(rs.getString("emp_name"));
45                 employee.setSalary(rs.getDouble("salary"));
46                 employee.setDeptName(rs.getString("dept_name"));
47                 list.add(employee);
48             }
49             rs.close();
50
51             rs = stmt.executeQuery("SELECT FOUND_ROWS()");
52             if(rs.next())
53                 this.noOfRecords = rs.getInt(1);
54         } catch (SQLException e) {
55             e.printStackTrace();
56         } catch (ClassNotFoundException e) {
57             e.printStackTrace();
58         } finally
59         {
60             try {

```

```

51         if(stmt != null)
52             stmt.close();
53         if(connection != null)
54             connection.close();
55     } catch (SQLException e) {
56         e.printStackTrace();
57     }
58     return list;
59 }
60 public int getNoOfRecords() {
61     return noOfRecords;
62 }
63 }

```

A **SELECT** statement may include a **LIMIT clause** to restrict the number of rows the server returns to the client. This **LIMIT** clause takes two arguments; The first argument specifies the offset of the first row to return, and the second specifies the maximum number of rows to return.

In some cases, it is desirable to know how many rows the statement would have returned without the **LIMIT**, but without running the statement again. To obtain this row count, include a **SQL_CALC_FOUND_ROWS** option in the **SELECT** statement, and then invoke **FOUND_ROWS()** afterward:

Writing Servlet

```

01 package com.theopentutorials.servlets;
02
03 import java.io.IOException;
04 import java.util.List;
05 import javax.servlet.RequestDispatcher;
06 import javax.servlet.ServletException;
07 import javax.servlet.http.HttpServlet;
08 import javax.servlet.http.HttpServletRequest;
09 import javax.servlet.http.HttpServletResponse;
10 import com.theopentutorials.dao.EmployeeDAO;
11 import com.theopentutorials.to.Employee;
12
13 /**
14  * Servlet implementation class EmployeeServlet
15  */
16 public class EmployeeServlet extends HttpServlet {
17     private static final long serialVersionUID = 1L;
18
19     public EmployeeServlet() {
20         super();
21     }
22
23     public void doGet(HttpServletRequest request,
24         HttpServletResponse response)
25         throws ServletException, IOException {
26         int page = 1;
27         int recordsPerPage = 5;
28         if(request.getParameter("page") != null)
29             page = Integer.parseInt(request.getParameter("page"));

```



```

25     EmployeeDAO dao = new EmployeeDAO();
26     List<Employee> list = dao.viewAllEmployees((page-
27 1)*recordsPerPage,
                                recordsPerPage);
28     int noOfRecords = dao.getNoOfRecords();
29     int noOfPages = (int) Math.ceil(noOfRecords * 1.0 /
30 recordsPerPage);
31     request.setAttribute("employeeList", list);
32     request.setAttribute("noOfPages", noOfPages);
33     request.setAttribute("currentPage", page);
34     RequestDispatcher view =
35     request.getRequestDispatcher("displayEmployee.jsp");
36     view.forward(request, response);
37 }

```

- In Servlet, we are first getting the value of 'page' parameter and storing it in 'page' variable. We wanted to display five (5) records per page which we are passing as an argument to viewAllEmployees(offset, 5). We are storing three attributes in the request scope and forwarding the request to a JSP page (displayEmployee.jsp);
 - Employee list available in 'list' variable
 - Total number of pages available in 'noOfPages' variable
 - Current page available in 'page' variable

Writing web.xml

```

01 <?xml version="1.0" encoding="UTF-8"?>
02 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
03     xmlns="http://java.sun.com/xml/ns/javaee"
04     xmlns:web="http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"
05     xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
06 http://java.sun.com/xml/ns/javaee
07 /web-app_2_5.xsd" id="WebApp_ID" version="2.5">
08     <display-name>JSPPagination</display-name>
09     <servlet>
10         <servlet-name>EmployeeServlet</servlet-name>
11         <servlet-
12 class>com.theopentutorials.servlets.EmployeeServlet</servlet-class>
13         </servlet>
14         <servlet-mapping>
15             <servlet-name>EmployeeServlet</servlet-name>
16             <url-pattern>/employee.do</url-pattern>
17         </servlet-mapping>
18     </web-app>

```

Writing displayEmployee.jsp

This JSP page uses JSP Standard Tag Library (JSTL) along with Expression Language (EL). It retrieves the attributes from request scope and displays the result. To use JSTL libraries, you must include a <taglib> directive in your JSP page.

```

01 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
02     pageEncoding="ISO-8859-1"%>
03 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
04 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

```

```

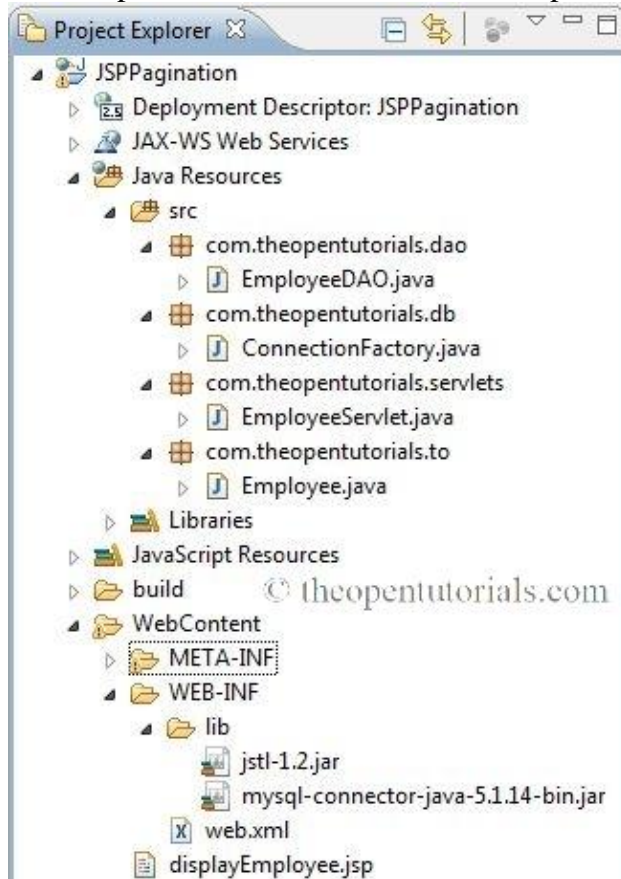
04         "<a href='\"http://www.w3.org/TR/html4/loose.dtd\"'>
05     <html>
06     <head>
07     <meta http-equiv=\"Content-Type\" content=\"text/html; charset=ISO-8859-
08     1\">
09     <title>Employees</title>
10     </head>
11     <body>
12         <table border=\"1\" cellpadding=\"5\" cellspacing=\"5\">
13             <tr>
14                 <th>Emp ID</th>
15                 <th>Emp Name</th>
16                 <th>Salary</th>
17                 <th>Dept Name</th>
18             </tr>
19
20             <c:forEach var=\"employee\" items=\"${employeeList}\">
21                 <tr>
22                     <td>${employee.employeeId}</td>
23                     <td>${employee.employeeName}</td>
24                     <td>${employee.salary}</td>
25                     <td>${employee.deptName}</td>
26                 </tr>
27             </c:forEach>
28         </table>
29
30         <%--For displaying Previous link except for the 1st page --%>
31         <c:if test=\"${currentPage != 1}\">
32             <td><a href=\"employee.do?page=${currentPage -
33             1}\">Previous</a></td>
34         </c:if>
35
36         <%--For displaying Page numbers.
37         The when condition does not display a link for the current page--%>
38         <table border=\"1\" cellpadding=\"5\" cellspacing=\"5\">
39             <tr>
40                 <c:forEach begin=\"1\" end=\"${noOfPages}\" var=\"i\">
41                     <c:choose>
42                         <c:when test=\"${currentPage eq i}\">
43                             <td>${i}</td>
44                         </c:when>
45                         <c:otherwise>
46                             <td><a
47 href=\"employee.do?page=${i}\">${i}</a></td>
48                             </c:otherwise>
49                         </c:choose>
50                     </c:forEach>
51                 </tr>
52             </table>
53
54             <%--For displaying Next link --%>
55             <c:if test=\"${currentPage lt noOfPages}\">
56                 <td><a href=\"employee.do?page=${currentPage +
57                 1}\">Next</a></td>
58             </c:if>
59
60         </body>
61     </html>

```

First 'table' tag displays list of employees with their details. Second 'table' tag displays the page numbers.

Folder Structure

The complete folder structure of this example is shown below.



Output

Use Ctrl + F11 in Eclipse to run the Servlet. The URL for requesting the Servlet is <http://localhost:8080/JSPPagination/employee.do>

Employees

http://localhost:8080/JSPPagination/employee.do

Emp ID	Emp Name	Salary	Dept Name
1	John	10000.0	Development
2	Peter	15000.0	Testing
3	Abraham	20000.0	Administration
4	Ram	35000.0	Marketing
5	Diana	40000.0	Financial

1 2 3

[Next](#)

© theopentutorials.com

Employees

http://localhost:8080/JSPPagination/employee.do?page=2

Emp ID	Emp Name	Salary	Dept Name
6	Raj	35000.0	QA
7	Jennifer	20000.0	Sales
8	Jansi	30000.0	Purchase
9	Jenni	60000.0	Services
10	Sanjay	35000.0	Inventory

[Previous](#)

1 2 3

[Next](#)

© theopentutorials.com