

***Java Technologies for Web Applications***

**Lab Guides**

|  |  |
| --- | --- |
| Document Code | 25e-BM/HR/HDCV/FSOFT |
| Version | 1.1 |
| Effective Date | 20/11/2012 |

**Hanoi, 04/2019**

RECORD OF CHANGES

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Effective Date | Change Description | Reason | Reviewer | Approver |
|  | 25/Jun/2018 | Create a new Lab | Create new | DieuNT1 | VinhNV |
|  | 01/May/2019 | Update Fsoft Template | Update | DieuNT1 | VinhNV |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Contents

[Unit 2 - MVC Model and Session Tracking 4](#_Toc23947315)

[Objectives: 4](#_Toc23947316)

[Descriptions and Guidelines: 4](#_Toc23947317)

[Step1: Create login page the following as: 4](#_Toc23947318)

[Step2: Create Database 6](#_Toc23947319)

[Step3: Create a maven project named “JWEB.M.L201”: 7](#_Toc23947320)

[Step4: Process login 8](#_Toc23947321)

[Step5. Create entity classes 11](#_Toc23947322)

[Step6. Use jQuery/AJAX to send the requests to server 11](#_Toc23947323)

[Step7: Some utility classes 16](#_Toc23947324)

|  |  |
| --- | --- |
|  | **CODE: NWEB.M.L201**  **TYPE: Medium**  **LOC:**  **DURATION: 180 MINUTES** |

# Unit 2 - MVC Model and Session Tracking

Objectives:

* Understand the basic concepts of web development technologies with java (JSP / Servlet)
* Able to write servlets using the Java programming language (Java servlets)
* Create dynamic HTML content with Servlets and JavaServer Pages, using the Expression Language, and the JSP Standard Tag Library (JSTL)
* Create robust web applications using MVC architecture, session management, filters, and database integration (JDBC)
* Make Servlets and JSP work together cleanly
* Create secure web applications using the features of the Java EE web container

Descriptions and Guidelines:

**Link Bootstrap 4:**

<link rel=*"stylesheet"*

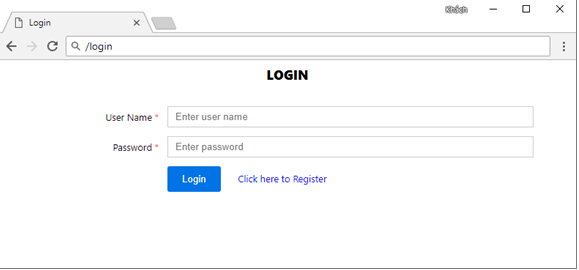
href=*"https://maxcdn.bootstrapcdn.com/bootstrap/4.1.0/css/bootstrap.min.css"*>

**Link Font Awesome:**

<link rel=*"stylesheet"* href=*"https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css"*>

Step1: Create login page the following as:

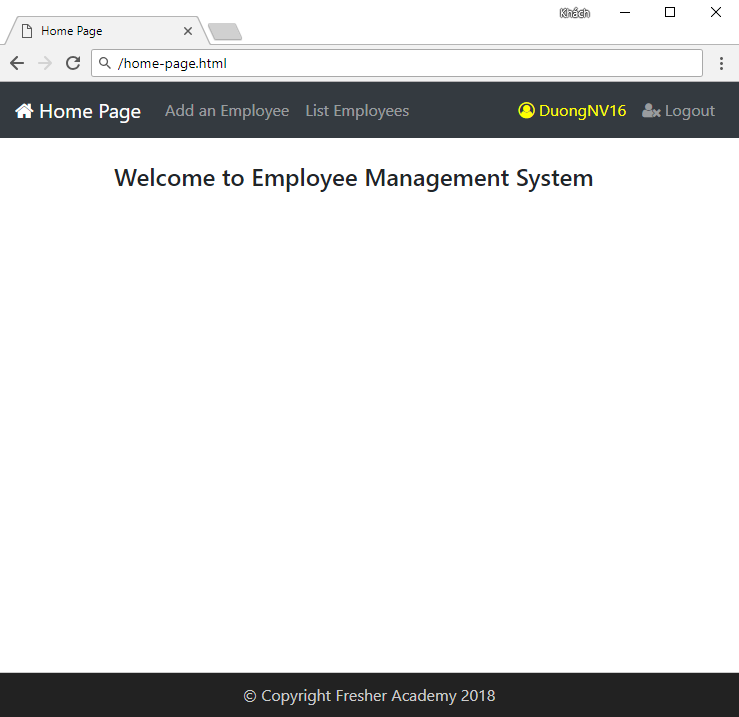
* *login.jsp screen:*



*Screen 01\_Layout 01*

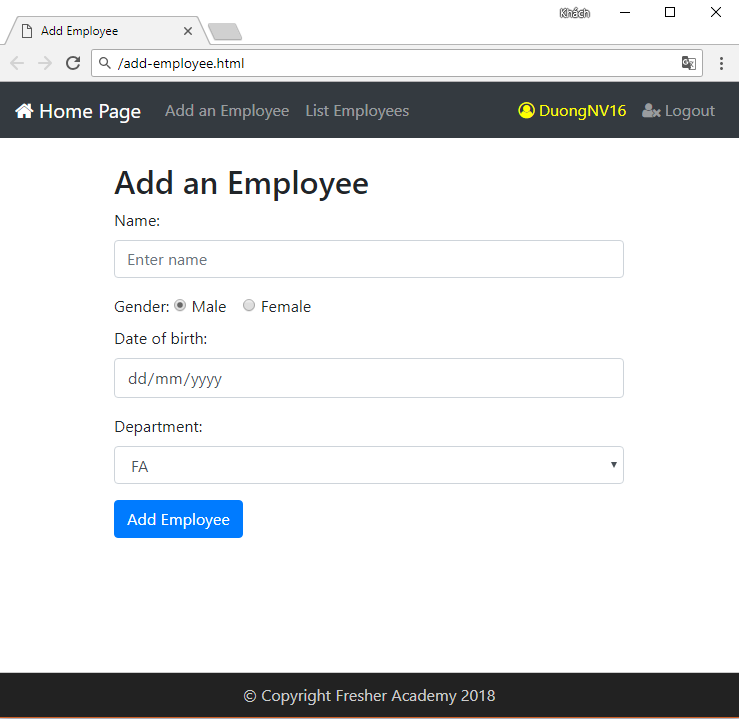
**Source code download here: , , **

* *home-page.jsp screen:*

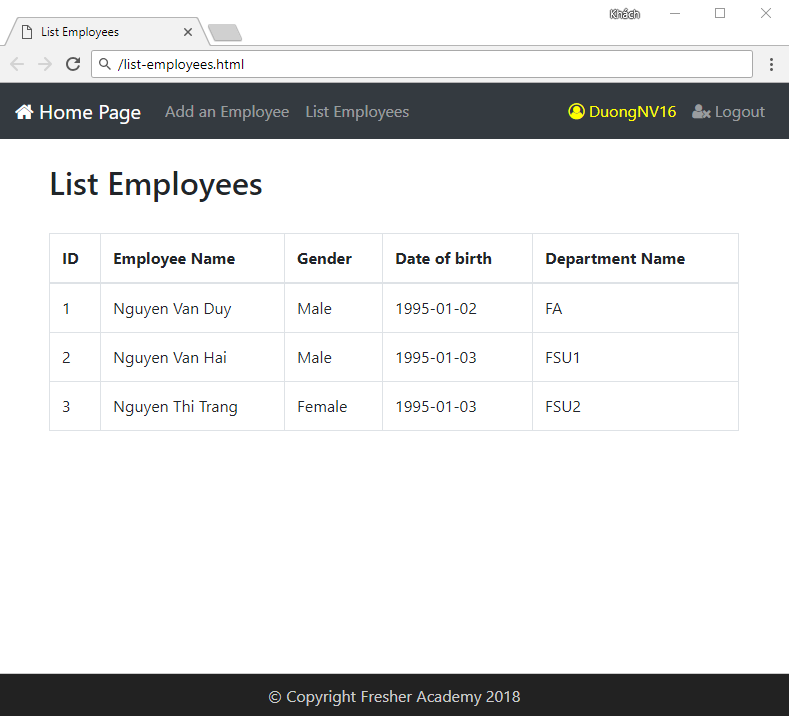


**Source code download here: **

* *add-employee.jsp screen:*

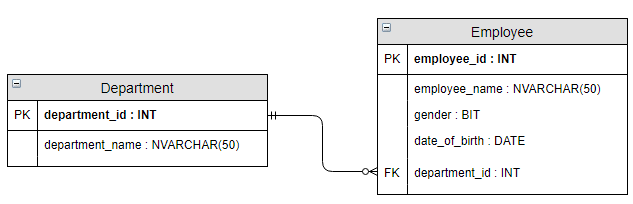


* *list-employees.jsp screen:*



Step2: Create Database

Create a database named “**JNWEBML201\_SMS**” có các bảng và quan hệ như sau:



Tạo stored procedure “**usp\_registerUser**” như sau:

1. CREATE PROC [dbo].[usp\_registerUser]
2. @firstName VARCHAR(50),
3. @lastName VARCHAR(50),
4. @email VARCHAR(100),
5. @userName VARCHAR(50),
6. @password VARCHAR(50)
7. AS
8. BEGIN
9. INSERT INTO Users VALUES (@firstName, @lastName, @email, @userName, @password)
10. END

Step3: Create a maven project named “JWEB.M.L201”:

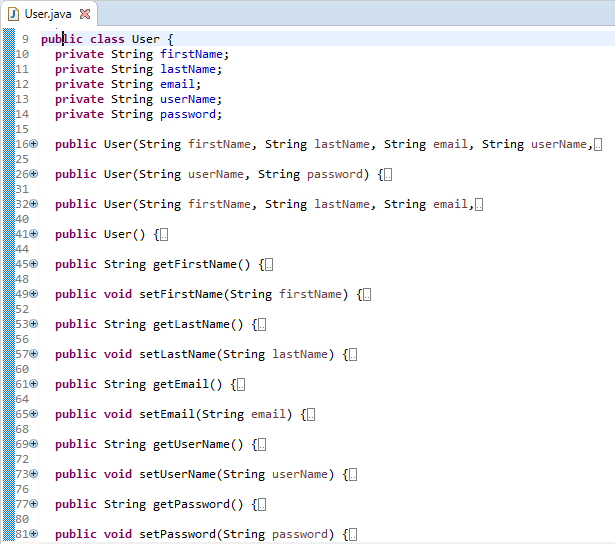
|  |  |
| --- | --- |
|  |  |

File **pom.xml**

1. <dependencies>
2. <dependency>
3. <groupId>javax.servlet</groupId>
4. <artifactId>javax.servlet-api</artifactId>
5. <version>3.1.0</version>
6. </dependency>
7. <dependency>
8. <groupId>com.microsoft.sqlserver</groupId>
9. <artifactId>mssql-jdbc</artifactId>
10. <version>7.0.0.jre8</version>
11. </dependency>
12. <dependency>
13. <groupId>log4j</groupId>
14. <artifactId>log4j</artifactId>
15. <version>1.2.17</version>
16. </dependency>
17. <dependency>
18. <groupId>javax.servlet</groupId>
19. <artifactId>jstl</artifactId>
20. <version>1.2</version>
21. </dependency>
22. </dependencies>
23. <build>
24. <finalName>JavaWeb\_P\_L002</finalName>
25. <plugins>
26. <plugin>
27. <groupId>org.apache.maven.plugins</groupId>
28. <artifactId>maven-compiler-plugin</artifactId>
29. <version>3.7.0</version>
30. <configuration>
31. <source>1.8</source>
32. <target>1.8</target>
33. </configuration>
34. </plugin>
35. <plugin>
36. <groupId>org.apache.maven.plugins</groupId>
37. <artifactId>maven-war-plugin</artifactId>
38. <version>3.2.2</version>
39. <configuration>
40. <warSourceDirectory>src/main/webapp
41. </warSourceDirectory>
42. <failOnMissingWebXml>
43. false
44. </failOnMissingWebXml>
45. </configuration>
46. </plugin>
47. </plugins>
48. </build>

Step4: Process login

* Create **User** class in package **fa.training.entities**:



* Create package **fa.training.servlet** to contains Servlet class.
* Create **LoginServlet** class in package **fa.training.servlet** to handle login the following:

1. @WebServlet(urlPatterns = "/login")
2. **public** **class** LoginServlet **extends** HttpServlet {
3. private static final long *serialVersionUID* = 1L;
4. **private** **static** UserDao *userDao* = **new** UserDao();
5. @Override
6. **protected** **void** doPost(HttpServletRequest request, HttpServletResponse response)
7. **throws** ServletException, IOException {
8. Log4J.*getLogger*().info("Running on doPost method of LoginServlet");
9. String userName = request.getParameter("userName");
10. String password = request.getParameter("password");
11. User user = **new** User(userName, password);
12. **try** {
13. **if** (*userDao*.login(user)) {
14. HttpSession session = request.getSession();
15. // if login successfully, save session user, who have just logined
16. session.setAttribute("userLogin", user);
17. response.sendRedirect(request.getContextPath() + "/home");
18. } **else** {
19. request.setAttribute("userRegister", user);
20. request.setAttribute("loginFail", "User name or password is incorrect");
21. request.getRequestDispatcher("/views/login.jsp").forward(request, response);
22. }
23. } **catch** (ClassNotFoundException e) {
24. Log4J.*getLogger*().
25. error("Class not found exception in method doPost of LoginServlet");
26. } **catch** (SQLException e) {
27. Log4J.*getLogger*().error("SQL exception in method doPost of LoginServlet");
28. }
29. }
30. @Override
31. **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)
32. **throws** ServletException, IOException {
33. Log4J.*getLogger*().info("Running on doGet method of LoginServlet");
34. request.getRequestDispatcher("views/login.jsp").forward(request, response);
35. }
36. }

If the login is successful, it will save the logged-in session user, and redirect to request **/home** to display the home-page.

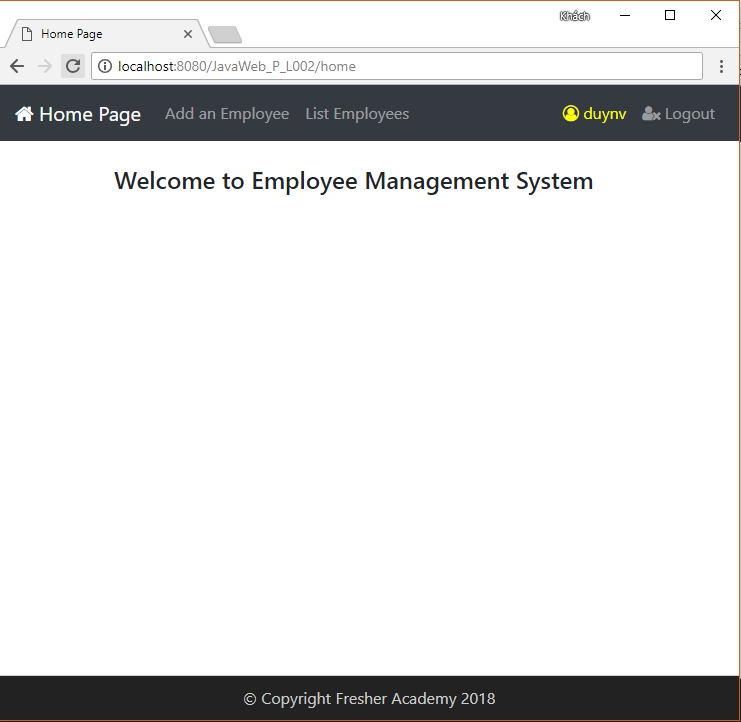
* Create a **HomePageServlet** class to handle this requestion:

1. @WebServlet("/home")
2. **public** **class** HomePageServlet **extends** HttpServlet {
4. private static final long *serialVersionUID* = 1L;
5. @Override
6. **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)
7. **throws** ServletException, IOException {
8. request.getRequestDispatcher("/views/home-page.jsp").forward(request, response);
9. }
10. }

* In **home-page.jsp**, show login **userName** here:

1. <ul class="navbar-nav ml-auto">
2. <li class="nav-item">
3. <a class="nav-link" style="color: yellow" href="#">
4. <i class="fa fa-user-circle-o"></i>
5. ${userLogin.userName}
6. </a>
7. </li>
8. <li class="nav-item">
9. <a class="nav-link" href="<%=request.getContextPath()%>/logout">
10. <i class="fa fa-user-times"></i> Logout
11. </a>
12. </li>
13. </ul>

**Screen result:**



* Continue, create a **LogoutServlet** class the following as:

1. @WebServlet("/logout")
2. **public** **class** **LogoutServlet** **extends** HttpServlet {
3. **private** **static** **final** **long** ***serialVersionUID*** = 1L;
4. @Override
5. **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)
6. **throws** ServletException, IOException {
7. Log4J.*getLogger*().info("Logging out");
8. // remove session userLogin
9. request.getSession().removeAttribute("userLogin");
10. // redirect to /login
11. response.sendRedirect(request.getContextPath() + "/login");
12. }
13. }

Step5. Create entity classes

* Create package fa**.training.entities** and the entity classes the following as:

Department.java class:

1. **public** **class** Department {
2. **private** **int** departmentId;
3. **private** String departmentName;
5. // Constructors and getters/setters
6. }

**Employee.java** class:

1. **public** **class** Employee {
2. **private** **int** employeeId;
3. **private** String employeeName;
4. **private** **byte** gender;
5. **private** Date dateOfBirth;
6. **private** **int** departmentId;
7. // Constructors and getters/setters
8. }

Step6. Use jQuery/AJAX to send the requests to server

* Create **home-page.js** the following as:

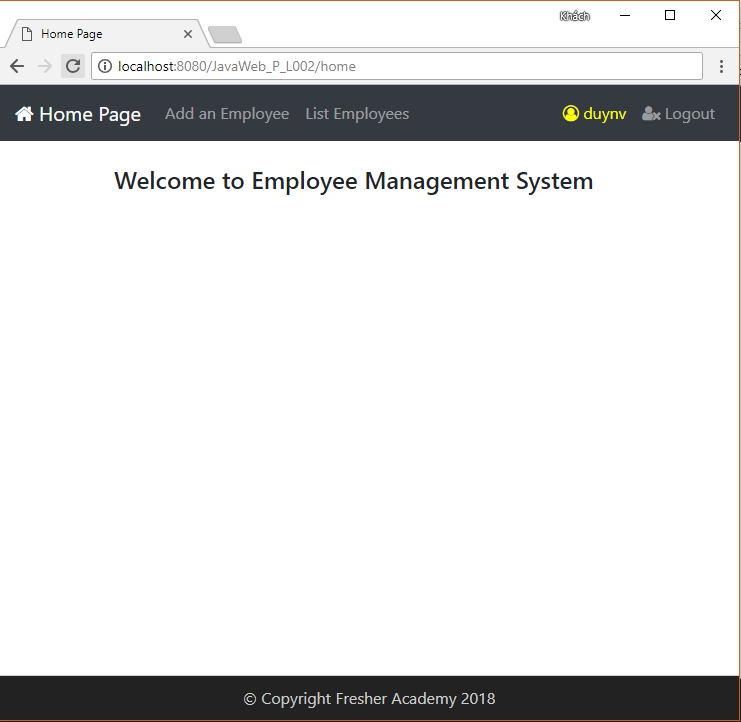
1. $(document).ready(**function**() {
2. $("#addEmpLink").click(**function**() {
3. $.get({
4. url : "/JavaWeb\_P\_L002/add-emp",
5. success : **function**(response) {
6. $(".container").html(response);
7. }
8. });
9. });
10. $("#listEmpsLink").click(**function**() {
11. $.get({
12. url : "/JavaWeb\_P\_L002/list-employees",
13. success : **function**(response) {
14. $(".container").html(response);
15. }
16. });
17. });
18. });

* Link to **home-page.js** in **home-page.jsp**:

1. <script type="text/javascript"
2. src="<%=request.getContextPath()%>/resources/js/home-page.js"></script>

* Create **AddEmployeeServlet** class in **fa.training.servlet** the following as:

When the user selects “**Add an Employee**” link, the request will be processed by ***doGet***() method in **AddEmployeeServlet** class:



1. @WebServlet("/add-emp")
2. **public** **class** AddEmployeeServlet **extends** HttpServlet {
3. **private** **static** **final** **long** ***serialVersionUID*** = 1L;
4. **private** DepartmentDao departmentDao = **new** DepartmentDao();
5. **private** EmployeeDao employeeDao = **new** EmployeeDao();
6. @Override
7. **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)
8. **throws** ServletException, IOException {
9. **try** {
10. // Get all of departments from DB and display on selected-box in
11. // add-employee.jsp page
12. List<Department> listOfDepartment = departmentDao.findAllDepartment();
13. request.setAttribute("listOfDepartment", listOfDepartment);
14. // This method doGet(): trả về response là trang add-employee.jsp cho ajax để
15. // hiển thị trên trang home-page
16. request.getRequestDispatcher("/views/add-employee.jsp").
17. forward(request, response);
18. } **catch** (ClassNotFoundException | SQLException e) {
19. Log4J.*getLogger*().error(e.getMessage());
20. }
21. }
22. // Method doPost(): xử lý khi click button “Add Employee”
23. @Override
24. **protected** **void** doPost(HttpServletRequest request, HttpServletResponse response)
25. **throws** ServletException, IOException {
27. **int** deptId = Integer.*parseInt*(request.getParameter("deptId"));
28. String employeeName = request.getParameter("employeeName");
29. **byte** gender = Byte.*parseByte*(request.getParameter("gender"));
30. Date dateOfBirth = **null**;
32. **try** {
33. dateOfBirth = DateUtils.*convertStringToDate*(request.
34. getParameter("dateOfBirth"));
35. } **catch** (ParseException e) {
36. Log4J.*getLogger*().error("Parse Exception when convert string to date");
37. }

40. Employee employee = **new** Employee(employeeName, gender, dateOfBirth, deptId);
42. **try** {
43. employeeDao.addEmployee(employee);
44. List<Department> listOfDepartment = departmentDao.findAllDepartment();
45. request.setAttribute("listOfDepartment", listOfDepartment);
46. request.setAttribute("employee", employee);
47. request.setAttribute("message", "Add new employee successfully");
48. request.getRequestDispatcher("/views/add-employee.jsp").
49. forward(request, response);
50. } **catch** (ClassNotFoundException | SQLException e) {
51. Log4J.*getLogger*().error("An exception occurs");
52. }
53. }
54. }

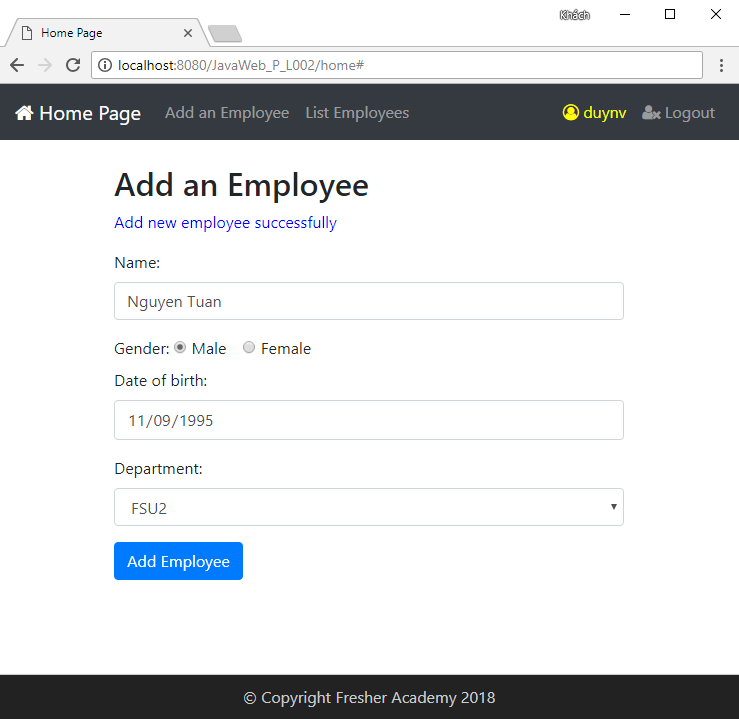
* **add-employee.jsp** page:

1. <%@ page language="java" contentType="text/html; charset=UTF-8"
2. pageEncoding="UTF-8"%>
3. <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
4. <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
5. <body>
6. <form action="#" method="post" name="frm-addEmp">
7. <div class="row">
8. <div class="col-md-6 offset-md-3">
9. <h2>Add a Employee</h2>
10. <p style="color: blue">${message}</p>
11. <div class="form-group">
12. <label for="employeeName">Name:</label>
13. <input type="text" class="form-control" id="employeeName"
14. placeholder="Enter name" name="employeeName"
15. value="${employee.employeeName}">
16. </div>
17. <label for="gender">Gender:</label>
18. <div class="form-check-inline">
19. <label class="form-check-label">
20. <input type="radio" class="form-check-input" name="gender"
21. value="1" checked>Male
22. </label>
23. </div>
25. <div class="form-check-inline">
26. <label class="form-check-label">
27. <input type="radio" class="form-check-input" name="gender"
28. value="0" ${employee.gender==0 ? 'checked' : '' }>Female
29. </label>
30. </div>
31. <div class="form-group">
32. <label for="dateOfBirth">Date of birth:</label>
33. <input type="date" class="form-control" id="dateOfBirth"
34. placeholder="Enter date of birth" name="dateOfBirth"
35. value="<fmt:formatDate
36. value='${employee.dateOfBirth}' pattern='yyyy-MM-dd' />">
37. </div>
39. <div class="form-group">
40. <label for="dept">Department:</label>
41. <select class="form-control" id="dept">
42. <c:forEach items="${listOfDepartment}" var="department">
43. <option value="${department.departmentId}"
44. ${department.departmentId==employee.departmentId
45. ? 'selected' : '' }>
46. ${department.departmentName}
47. </option>
48. </c:forEach>
49. </select>
50. </div>
51. </div>
52. </div>
53. <div class="row">
54. <div class="col-md-6 offset-md-3">
55. <button type="button" id="btn-addEmp" class="btn btn-primary">
56. Add Employee
57. </button>
58. </div>
59. </div>
60. </form>
61. <script type="text/javascript" src="<%=request.getContextPath()%>/resource">
62. </script>
63. </body>

* **add-employee.js** file:

1. $(document).ready(**function**() {
2. $("#btn-addEmp").click(**function**() {
3. **var** employeeName = $("#employeeName").val();
4. **var** gender = $("input[name=gender]:checked").val();
5. **var** dateOfBirth = $("#dateOfBirth").val();
6. **var** deptId = $("#dept").val();
7. $.post({
8. url : "/JavaWeb\_P\_L002/add-emp",
9. data : {
10. employeeName : employeeName,
11. gender : gender,
12. dateOfBirth : dateOfBirth,
13. deptId : deptId
14. },
15. success : **function**(response) {
16. $(".container").html(response);
17. }
18. });
19. });
20. });

**Result page:**



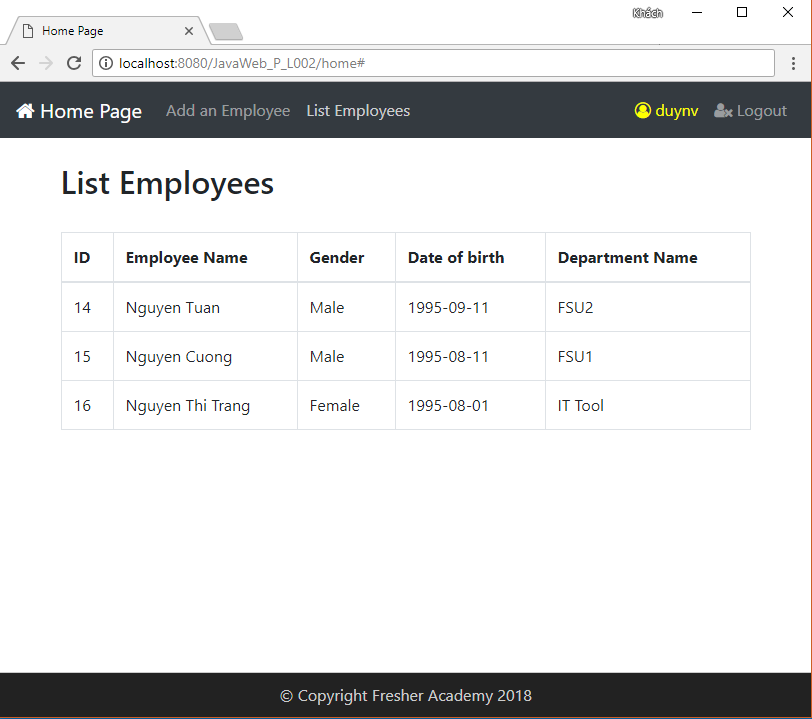
* Create **ListEmployeeServlet** class in fa.training.servlet the following as:

1. @WebServlet("/list-employees")
2. **public** **class** ListEmployeesServlet **extends** HttpServlet {
3. private static final long *serialVersionUID* = 1L;
4. **private** EmployeeDao employeeDao = **new** EmployeeDao();
5. **private** DepartmentDao departmentDao = **new** DepartmentDao();
6. @Override
7. **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)
8. **throws** ServletException, IOException {
9. **try** {
10. List<Employee> listOfEmployee = employeeDao.findAllEmployee();
11. List<Department> listOfDepartment = departmentDao.findAllDepartment();
12. request.setAttribute("listOfEmployee", listOfEmployee);
13. request.setAttribute("listOfDepartment", listOfDepartment);
14. request.getRequestDispatcher("/views/list-employees.jsp").
15. forward(request, response);
16. } **catch** (ClassNotFoundException | SQLException e) {
17. Log4J.getLogger().error(e.getMessage());
18. }
19. }
20. }

* **list-employee.jsp** page:

1. <%@ page language="java" contentType="text/html; charset=UTF-8"
2. pageEncoding="UTF-8"%>
3. <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
4. <body>
5. <br />
6. <h2>List of Employees</h2>
7. <br />
8. <table class="table table-bordered">
9. <thead>
10. <tr>
11. <th>ID</th>
12. <th>Employee Name</th>
13. <th>Gender</th>
14. <th>Date of birth</th>
15. <th>Department Name</th>
16. </tr>
17. </thead>
18. <tbody>
19. <c:forEach items="${listOfEmployee}" var="employee">
20. <tr>
21. <td>${employee.employeeId}</td>
22. <td>${employee.employeeName}</td>
23. <td>${employee.gender == 1 ? 'Male' : 'Female'}</td>
24. <td>${employee.dateOfBirth}</td>
25. <c:forEach items="${listOfDepartment}" var="department">
26. <c:if test="${employee.departmentId ==
27. department.departmentId}">
28. <td>${department.departmentName}</td>
29. </c:if>
30. </c:forEach>
31. </tr>
32. </c:forEach>
33. </tbody>
34. </table>
35. </body>

**Result page:**



Step7: Some utility classes

Constants class

1. **package** fa.training.utils;
2. **public** **class** Constants {
3. **public** **static** **final** String ***REGISTER\_SUCCESSFULLY\_MESSAGE*** = "Register user successfully";
4. **public** **static** **final** String ***REGISTER\_FAIL\_MESSAGE*** = "Register user fail";
5. }

DateUtils class

1. package fa.training.utils;
2. import java.text.ParseException;
3. import java.text.SimpleDateFormat;
4. import java.util.Date;
5. public class DateUtils {
6. public static Date convertStringToDate(String dateString)
7. throws ParseException {
8. SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
9. Date date = formatter.parse(dateString);
10. return date;
11. }
12. public static java.sql.Date convertJavaDateToSqlDate(Date javaDate) {
13. java.sql.Date sqlDate = new java.sql.Date(javaDate.getTime());
14. return sqlDate;
15. }
16. }

**Log4J class**

1. package fa.training.utils;
2. import org.apache.log4j.Logger;
3. import org.apache.log4j.PropertyConfigurator;
4. /\*\*
5. \* Class Log4J utility
6. \* @author FA
7. \*
8. \*/
9. public class Log4J {
10. private static final Logger logger = Logger.getLogger(Log4J.class);
11. /\*\*
12. \* method configure Log4J.
13. \*
14. \* @return Logger logger
15. \*/
16. public static Logger getLogger() {
17. PropertyConfigurator.configure(
18. Log4J.class.getResourceAsStream("/properties/log4jFILE.properties"));
19. return logger;
20. }
21. }

**-- THE END --**