**Đề bài: Xây dựng ứng dụng quản lý danh sách nhân viên**

**Mục tiêu**:  
Xây dựng một ứng dụng web giúp người dùng tìm kiếm danh sách nhân viên dựa trên các tiêu chí khác nhau. Sử dụng **JSP**, **Servlet**, và **JDBC** để hoàn thiện ứng dụng, với cơ sở dữ liệu mẫu được cung cấp.

**Yêu cầu chi tiết**

**1. Chức năng**

* Tìm kiếm nhân viên với các tiêu chí sau:
  + **Name**: Tìm kiếm theo tên nhân viên (có thể sử dụng từ khóa).
  + **Salary**: Tìm kiếm nhân viên có mức lương bằng hoặc cao hơn giá trị nhập vào.
  + **From Hire Date** và **To Hire Date**: Tìm kiếm nhân viên có ngày tuyển dụng nằm trong khoảng thời gian.
  + **Position**: Tìm kiếm theo chức vụ (có thể sử dụng từ khóa).
  + **Department**: Tìm kiếm nhân viên thuộc phòng ban cụ thể (có thể chọn từ danh sách các phòng ban).
* Các trường tìm kiếm có thể để trống. Nếu để trống, kết quả sẽ bỏ qua tiêu chí đó.
* Hiển thị kết quả tìm kiếm dưới dạng bảng bao gồm các cột:
  + Employee ID
  + Name
  + Position
  + Salary
  + Department Name
  + Hire Date
* Nếu không tìm thấy kết quả nào, hiển thị thông báo: "Không tìm thấy nhân viên phù hợp."

**2. Cơ sở dữ liệu**

Cơ sở dữ liệu đã được cung cấp:

* **Bảng departments**:
  + department\_id: Mã phòng ban (Primary Key).
  + department\_name: Tên phòng ban.
  + location: Địa điểm phòng ban.
* **Bảng employees**:
  + employee\_id: Mã nhân viên (Primary Key).
  + name: Tên nhân viên.
  + position: Chức vụ.
  + salary: Lương.
  + department\_id: Mã phòng ban (Foreign Key tham chiếu đến bảng departments).
  + hire\_date: Ngày tuyển dụng.

Dữ liệu trong bảng employees liên kết với bảng departments thông qua department\_id.

**3. Công nghệ sử dụng**

* **Frontend**: JSP để tạo giao diện người dùng.
* **Backend**:
  + **Servlet**: Xử lý logic, nhận yêu cầu từ giao diện và truy xuất dữ liệu từ cơ sở dữ liệu.
  + **JDBC**: Kết nối và thao tác với cơ sở dữ liệu.
* **Cơ sở dữ liệu**: MySQL.

**4. Yêu cầu kỹ thuật**

1. **Giao diện JSP**:
   * Tạo form nhập liệu với các trường:
     + Name (input text).
     + Salary (input number).
     + From Hire Date (input date).
     + To Hire Date (input date).
     + Position (input text).
     + Department (dropdown list hiển thị danh sách các phòng ban từ bảng departments).
   * Nút "Search" để gửi yêu cầu tìm kiếm đến Servlet.
   * Hiển thị kết quả tìm kiếm dưới dạng bảng (table).
2. **Servlet**:
   * Lấy tham số từ request:

String name = request.getParameter("name");

String salary = request.getParameter("salary");

String fromHireDate = request.getParameter("fromHireDate");

String toHireDate = request.getParameter("toHireDate");

String position = request.getParameter("position");

String departmentId = request.getParameter("departmentId");

* + Kiểm tra từng tham số:
    - Nếu tham số trống hoặc null, bỏ qua tiêu chí này trong truy vấn.
  + Kết nối cơ sở dữ liệu bằng JDBC, thực thi câu lệnh SQL và trả kết quả về JSP.

1. **JDBC**:
   * Chuẩn bị câu truy vấn SQL động:

SELECT e.employee\_id, e.name, e.position, e.salary, d.department\_name, e.hire\_date

FROM employees e

LEFT JOIN departments d ON e.department\_id = d.department\_id

WHERE 1=1

AND (e.name LIKE ? OR ? IS NULL)

AND (e.salary >= ? OR ? IS NULL)

AND (e.hire\_date BETWEEN ? AND ? OR ? IS NULL OR ? IS NULL)

AND (e.position LIKE ? OR ? IS NULL)

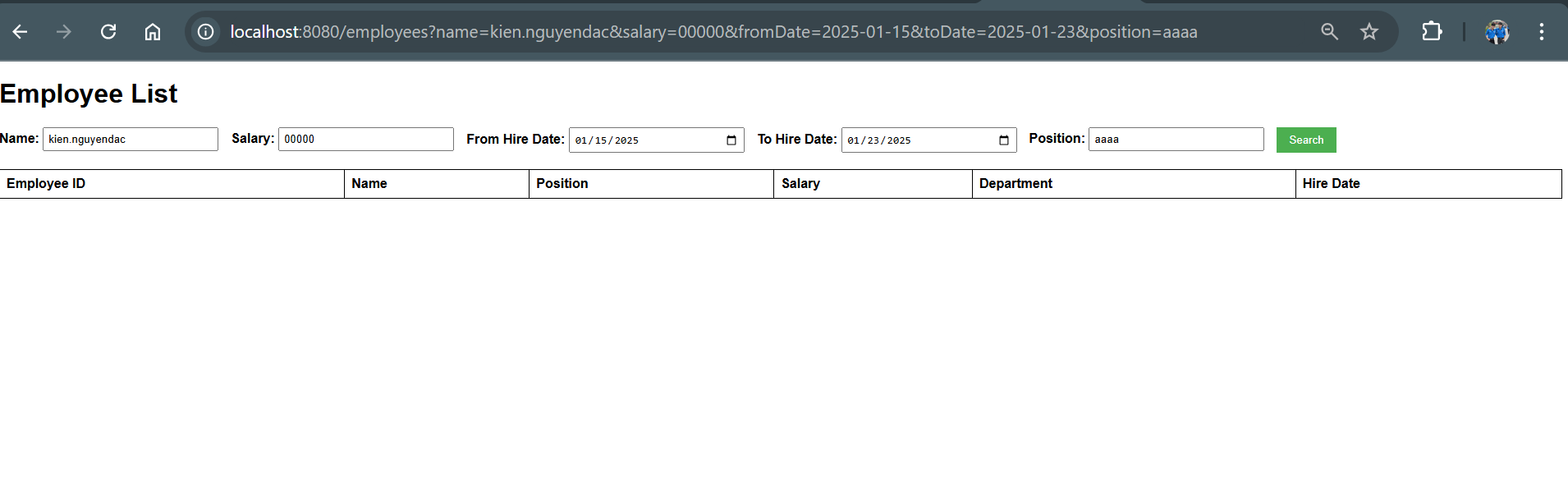
AND (e.department\_id = ? OR ? IS NULL);

* + Sử dụng PreparedStatement để gắn giá trị từ tham số nhằm tránh SQL Injection.

1. **Chạy thử**:
   * Chạy ứng dụng trên Tomcat hoặc một server servlet khác.
   * Kiểm tra các trường hợp:
     + Tìm kiếm với một trường (ví dụ: Name, Salary).
     + Tìm kiếm với nhiều trường (ví dụ: Name và From Hire Date).
     + Không nhập gì (trả về toàn bộ danh sách nhân viên).

**5. Kết quả mong đợi**

* Người dùng có thể tìm kiếm nhân viên bằng cách nhập một hoặc nhiều tiêu chí.
* Kết quả hiển thị chính xác trong bảng hoặc thông báo nếu không tìm thấy nhân viên phù hợp.
* Giao diện đơn giản, dễ sử dụng, và đảm bảo logic hoạt động đúng theo yêu cầu.

Hình ảnh giao diện demo  
  
code jsp tham khảo  


**Tóm tắt đề bài: Xây dựng ứng dụng quản lý danh sách nhân viên**

**Mục tiêu:**

Tạo ứng dụng web quản lý danh sách nhân viên, cho phép người dùng tìm kiếm nhân viên dựa trên các tiêu chí như tên, mức lương, ngày tuyển dụng, chức vụ, và phòng ban.

**Yêu cầu chính:**

1. **Chức năng:**
   * Tìm kiếm nhân viên theo:
     + Tên (có thể sử dụng từ khóa).
     + Lương (>= giá trị nhập).
     + Khoảng thời gian tuyển dụng.
     + Chức vụ (có thể dùng từ khóa).
     + Phòng ban (chọn từ danh sách phòng ban).
   * Kết quả tìm kiếm hiển thị dạng bảng với các thông tin:
     + Mã nhân viên, Tên, Chức vụ, Lương, Phòng ban, Ngày tuyển dụng.
   * Nếu không có kết quả: Hiển thị thông báo "Không tìm thấy nhân viên phù hợp".
2. **Cơ sở dữ liệu:**
   * **Bảng departments:** Chứa thông tin phòng ban.
   * **Bảng employees:** Chứa thông tin nhân viên, liên kết với bảng departments qua department\_id.
3. **Công nghệ:**
   * **Frontend:** Giao diện sử dụng JSP.
   * **Backend:** Servlet xử lý logic và JDBC kết nối cơ sở dữ liệu.
   * **Database:** MySQL.
4. **Yêu cầu kỹ thuật:**
   * **Giao diện JSP:** Form nhập liệu với các trường (Tên, Lương, Ngày tuyển dụng, Chức vụ, Phòng ban) và nút "Search".
   * **Servlet:**
     + Lấy dữ liệu từ form, xử lý logic tìm kiếm (kiểm tra giá trị null hoặc rỗng).
     + Kết nối MySQL qua JDBC, chạy câu truy vấn SQL động.
   * **JDBC:**
     + Sử dụng PreparedStatement để xây dựng truy vấn SQL tránh SQL Injection.
     + Ví dụ câu truy vấn:

sql

SELECT e.employee\_id, e.name, e.position, e.salary, d.department\_name, e.hire\_date

FROM employees e

LEFT JOIN departments d ON e.department\_id = d.department\_id

WHERE 1=1

AND (e.name LIKE ? OR ? IS NULL)

AND (e.salary >= ? OR ? IS NULL)

AND (e.hire\_date BETWEEN ? AND ? OR ? IS NULL OR ? IS NULL)

AND (e.position LIKE ? OR ? IS NULL)

AND (e.department\_id = ? OR ? IS NULL);

1. **Chạy thử:**
   * Kiểm tra các trường hợp tìm kiếm với một, nhiều tiêu chí, hoặc không nhập gì.
2. **Kết quả mong đợi:**
   * Người dùng tìm kiếm được nhân viên theo tiêu chí.
   * Giao diện dễ sử dụng, hiển thị kết quả đúng yêu cầu.

**File JSP:**  
Form nhập liệu với trường tìm kiếm và bảng kết quả.

**File Servlet:**  
Xử lý request, truy xuất dữ liệu và gửi về JSP.

**Kết quả mong muốn:**  
Ứng dụng hoàn chỉnh, chạy trên server Tomcat, hỗ trợ đầy đủ các trường hợp tìm kiếm.

Trong bài toán này, các **logic chính** bao gồm:

**1. Xử lý đầu vào từ người dùng (Form Validation & Handling):**

* Lấy dữ liệu từ form trên giao diện JSP:
  + Các trường nhập liệu: name, salary, fromHireDate, toHireDate, position, departmentId.
  + Dùng request.getParameter() để lấy giá trị từ form.
* Kiểm tra từng trường:
  + Nếu trường nào để trống hoặc null, bỏ qua tiêu chí này trong quá trình truy vấn.
  + Đảm bảo dữ liệu đầu vào đúng định dạng:
    - salary: Phải là số hợp lệ (>= 0).
    - fromHireDate và toHireDate: Phải là ngày hợp lệ, với fromHireDate ≤ toHireDate.

**2. Xây dựng truy vấn SQL động (Dynamic Query Building):**

* Câu truy vấn được xây dựng dựa trên các tiêu chí người dùng nhập.
* Ví dụ:
  + Nếu chỉ nhập tên và lương, SQL sẽ là:

sql

SELECT e.employee\_id, e.name, e.position, e.salary, d.department\_name, e.hire\_date

FROM employees e

LEFT JOIN departments d ON e.department\_id = d.department\_id

WHERE e.name LIKE ? AND e.salary >= ?;

* + Nếu không nhập gì, SQL trả về toàn bộ nhân viên:

sql

SELECT e.employee\_id, e.name, e.position, e.salary, d.department\_name, e.hire\_date

FROM employees e

LEFT JOIN departments d ON e.department\_id = d.department\_id;

* Sử dụng PreparedStatement để truyền tham số vào câu lệnh SQL, đảm bảo an toàn và tránh **SQL Injection**.

**3. Xử lý dữ liệu truy vấn từ cơ sở dữ liệu:**

* Kết nối cơ sở dữ liệu bằng JDBC:
  + Mở kết nối (Connection).
  + Chuẩn bị và thực thi câu lệnh truy vấn (PreparedStatement và ResultSet).
  + Đóng kết nối sau khi hoàn tất.
* Lấy dữ liệu từ ResultSet và lưu vào cấu trúc dữ liệu Danh Sách (List) để hiển thị trên giao diện JSP.

**4. Hiển thị kết quả tìm kiếm:**

* **Trường hợp có kết quả:**
  + Hiển thị danh sách nhân viên dưới dạng bảng:
    - Các cột: Mã nhân viên, Tên, Chức vụ, Lương, Tên phòng ban, Ngày tuyển dụng.
* **Trường hợp không có kết quả:**
  + Hiển thị thông báo: **"Không tìm thấy nhân viên phù hợp."**

**5. Logic xử lý lỗi:**

* **Lỗi kết nối cơ sở dữ liệu:**
  + Hiển thị thông báo lỗi nếu không kết nối được.
* **Lỗi dữ liệu đầu vào không hợp lệ:**
  + Trả về thông báo lỗi trên giao diện nếu:
    - Lương nhập không phải số.
    - Ngày nhập không hợp lệ.
* **Lỗi truy vấn SQL:**
  + Ghi log lỗi và hiển thị thông báo "Đã xảy ra lỗi, vui lòng thử lại sau."

Dưới đây là một ví dụ hoàn chỉnh về cách xây dựng ứng dụng quản lý danh sách nhân viên bằng JSP, Servlet và JDBC theo đúng yêu cầu trong đề bài. Bạn có thể tùy chỉnh lại cho phù hợp với cấu trúc project, tên class, hoặc môi trường của mình.

**1. Cấu trúc thư mục tham khảo**

EmployeeManagement/

┣─ src/

│ ┣─ model/

│ │ ┣─ Department.java

│ │ ┗─ Employee.java

│ ┣─ util/

│ │ ┗─ DBConnectionUtil.java

│ ┗─ controller/

│ ┗─ EmployeeSearchServlet.java

┣─ WebContent/ (hoặc webapp/)

│ ┣─ META-INF/

│ ┣─ WEB-INF/

│ │ ┣─ web.xml

│ ┗─ EmployeeSearch.jsp

┗─ pom.xml (nếu dùng Maven)

Ý tưởng:

* **model/**: Chứa các lớp Java biểu diễn bảng employees, departments.
* **util/**: Chứa các lớp tiện ích, ví dụ lớp DBConnectionUtil dùng để kết nối MySQL.
* **controller/**: Chứa Servlet xử lý logic tìm kiếm.
* **EmployeeSearch.jsp**: Giao diện chính, hiển thị form tìm kiếm và bảng kết quả.

**2. Thiết kế CSDL (tham khảo)**

**Bảng departments:**

CREATE TABLE departments (

department\_id INT PRIMARY KEY AUTO\_INCREMENT,

department\_name VARCHAR(100),

location VARCHAR(100)

);

**Bảng employees:**

CREATE TABLE employees (

employee\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100),

position VARCHAR(100),

salary DECIMAL(10,2),

department\_id INT,

hire\_date DATE,

FOREIGN KEY (department\_id) REFERENCES departments(department\_id)

);

**3. Lớp Model**

**3.1. Department.java**

package model;

public class Department {

private int departmentId;

private String departmentName;

private String location;

public Department() {}

public Department(int departmentId, String departmentName, String location) {

this.departmentId = departmentId;

this.departmentName = departmentName;

this.location = location;

}

public int getDepartmentId() {

return departmentId;

}

public void setDepartmentId(int departmentId) {

this.departmentId = departmentId;

}

public String getDepartmentName() {

return departmentName;

}

public void setDepartmentName(String departmentName) {

this.departmentName = departmentName;

}

public String getLocation() {

return location;

}

public void setLocation(String location) {

this.location = location;

}

}

**3.2. Employee.java**

package model;

import java.sql.Date;

public class Employee {

private int employeeId;

private String name;

private String position;

private double salary;

private int departmentId;

private Date hireDate;

private String departmentName; // Để hiển thị tên phòng ban thay vì chỉ ID

public Employee() {}

public Employee(int employeeId, String name, String position, double salary,

int departmentId, Date hireDate, String departmentName) {

this.employeeId = employeeId;

this.name = name;

this.position = position;

this.salary = salary;

this.departmentId = departmentId;

this.hireDate = hireDate;

this.departmentName = departmentName;

}

public int getEmployeeId() {

return employeeId;

}

public void setEmployeeId(int employeeId) {

this.employeeId = employeeId;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPosition() {

return position;

}

public void setPosition(String position) {

this.position = position;

}

public double getSalary() {

return salary;

}

public void setSalary(double salary) {

this.salary = salary;

}

public int getDepartmentId() {

return departmentId;

}

public void setDepartmentId(int departmentId) {

this.departmentId = departmentId;

}

public Date getHireDate() {

return hireDate;

}

public void setHireDate(Date hireDate) {

this.hireDate = hireDate;

}

public String getDepartmentName() {

return departmentName;

}

public void setDepartmentName(String departmentName) {

this.departmentName = departmentName;

}

}

**4. Lớp tiện ích kết nối CSDL DBConnectionUtil.java**

package util;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class DBConnectionUtil {

private static final String URL = "jdbc:mysql://localhost:3306/employee\_db";

private static final String USER = "root";

private static final String PASSWORD = "123456";

public static Connection getConnection() throws SQLException {

try {

Class.forName("com.mysql.jdbc.Driver");

return DriverManager.getConnection(URL, USER, PASSWORD);

} catch (ClassNotFoundException e) {

e.printStackTrace();

throw new SQLException("Driver not found");

}

}

}

**Lưu ý**:

* Nếu dùng driver mới, có thể là com.mysql.cj.jdbc.Driver.
* Nhớ thêm thư viện MySQL Connector (JAR) vào project hoặc cấu hình Maven.

**5. Servlet xử lý tìm kiếm EmployeeSearchServlet.java**

package controller;

import model.Employee;

import util.DBConnectionUtil;

import javax.servlet.ServletException;

import javax.servlet.http.\*;

import java.io.IOException;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class EmployeeSearchServlet extends HttpServlet {

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

// Lấy các tham số từ JSP

String name = request.getParameter("name");

String salaryStr = request.getParameter("salary");

String fromHireDate = request.getParameter("fromHireDate");

String toHireDate = request.getParameter("toHireDate");

String position = request.getParameter("position");

String departmentId = request.getParameter("departmentId");

// Xử lý tình huống null hoặc rỗng

name = (name == null || name.trim().isEmpty()) ? null : name.trim();

salaryStr = (salaryStr == null || salaryStr.trim().isEmpty()) ? null : salaryStr.trim();

fromHireDate = (fromHireDate == null || fromHireDate.trim().isEmpty()) ? null : fromHireDate.trim();

toHireDate = (toHireDate == null || toHireDate.trim().isEmpty()) ? null : toHireDate.trim();

position = (position == null || position.trim().isEmpty()) ? null : position.trim();

departmentId = (departmentId == null || departmentId.trim().isEmpty()) ? null : departmentId.trim();

List<Employee> employees = new ArrayList<>();

String sql = "SELECT e.employee\_id, e.name, e.position, e.salary, d.department\_name, e.hire\_date "+

"FROM employees e " +

"LEFT JOIN departments d ON e.department\_id = d.department\_id " +

"WHERE 1=1 "; // Bắt đầu WHERE với 1=1 để nối điều kiện dễ hơn

// Tạo danh sách tham số

List<Object> paramList = new ArrayList<>();

// Mỗi tiêu chí, nếu != null, ta thêm câu lệnh AND + placeholder

if (name != null) {

sql += " AND e.name LIKE ? ";

}

if (salaryStr != null) {

sql += " AND e.salary >= ? ";

}

if (fromHireDate != null && toHireDate != null) {

sql += " AND e.hire\_date BETWEEN ? AND ? ";

}

if (position != null) {

sql += " AND e.position LIKE ? ";

}

if (departmentId != null) {

sql += " AND e.department\_id = ? ";

}

try (Connection conn = DBConnectionUtil.getConnection();

PreparedStatement ps = conn.prepareStatement(sql)) {

int idx = 1;

// Gán giá trị cho từng placeholder

if (name != null) {

ps.setString(idx++, "%" + name + "%");

}

if (salaryStr != null) {

double salary = Double.parseDouble(salaryStr);

ps.setDouble(idx++, salary);

}

if (fromHireDate != null && toHireDate != null) {

ps.setDate(idx++, Date.valueOf(fromHireDate));

ps.setDate(idx++, Date.valueOf(toHireDate));

}

if (position != null) {

ps.setString(idx++, "%" + position + "%");

}

if (departmentId != null) {

ps.setInt(idx++, Integer.parseInt(departmentId));

}

ResultSet rs = ps.executeQuery();

while (rs.next()) {

Employee emp = new Employee();

emp.setEmployeeId(rs.getInt("employee\_id"));

emp.setName(rs.getString("name"));

emp.setPosition(rs.getString("position"));

emp.setSalary(rs.getDouble("salary"));

emp.setDepartmentName(rs.getString("department\_name"));

emp.setHireDate(rs.getDate("hire\_date"));

employees.add(emp);

}

} catch (SQLException e) {

e.printStackTrace();

}

// Gửi danh sách employees tìm được về JSP

request.setAttribute("employeeList", employees);

// Forward về trang JSP để hiển thị

request.getRequestDispatcher("/EmployeeSearch.jsp").forward(request, response);

}

// Nếu dùng POST để submit form, có thể forward về doGet

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

doGet(request, response);

}

}

**Giải thích**:

* Servlet đọc các tham số từ request.getParameter(...).
* Kiểm tra nếu tham số rỗng hoặc null thì bỏ qua tiêu chí trong SQL (WHERE).
* Dùng PreparedStatement để tránh SQL Injection.
* Thêm kết quả vào một List<Employee> rồi đưa sang JSP bằng request.setAttribute(...).
* Cuối cùng, forward (chuyển hướng nội bộ) về EmployeeSearch.jsp để hiển thị.

**6. Giao diện JSP: EmployeeSearch.jsp**

<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<%@ page import="java.util.\*, model.Employee" %>

<!DOCTYPE html>

<html>

<head>

<title>Employee List</title>

<style>

/\* CSS cơ bản cho bảng và form \*/

table {

border-collapse: collapse;

width: 100%;

margin-top: 20px;

}

table, th, td {

border: 1px solid #ccc;

}

th, td {

padding: 8px;

text-align: left;

}

.search-form {

margin-bottom: 10px;

}

.search-form label {

margin-right: 10px;

}

.search-input {

margin-right: 15px;

}

</style>

</head>

<body>

<h1>Employee List</h1>

<!-- Form tìm kiếm -->

<form class="search-form" action="<%= request.getContextPath() %>/EmployeeSearchServlet" method="get">

<label>Name:</label>

<input class="search-input" type="text" name="name" value="<%= request.getParameter("name") == null ? "" : request.getParameter("name") %>" />

<label>Salary (>=):</label>

<input class="search-input" type="number" name="salary" value="<%= request.getParameter("salary") == null ? "" : request.getParameter("salary") %>" />

<label>From Hire Date:</label>

<input class="search-input" type="date" name="fromHireDate" value="<%= request.getParameter("fromHireDate") == null ? "" : request.getParameter("fromHireDate") %>" />

<label>To Hire Date:</label>

<input class="search-input" type="date" name="toHireDate" value="<%= request.getParameter("toHireDate") == null ? "" : request.getParameter("toHireDate") %>" />

<label>Position:</label>

<input class="search-input" type="text" name="position" value="<%= request.getParameter("position") == null ? "" : request.getParameter("position") %>" />

<label>Department:</label>

<select class="search-input" name="departmentId">

<option value="">--All--</option>

<%

// Đoạn này bạn có thể load danh sách phòng ban từ DB

// hoặc bạn đã hard-code sẵn.

// Ví dụ, ta hard-code một số giá trị departmentId:

%>

<option value="1" <%= "1".equals(request.getParameter("departmentId")) ? "selected" : "" %>>Sales</option>

<option value="2" <%= "2".equals(request.getParameter("departmentId")) ? "selected" : "" %>>HR</option>

<option value="3" <%= "3".equals(request.getParameter("departmentId")) ? "selected" : "" %>>IT</option>

<option value="4" <%= "4".equals(request.getParameter("departmentId")) ? "selected" : "" %>>Finance</option>

</select>

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

</form>

<!-- Hiển thị kết quả tìm kiếm -->

<%

List<Employee> employeeList = (List<Employee>) request.getAttribute("employeeList");

if (employeeList != null) {

if (employeeList.isEmpty()) {

%>

<h3>Không tìm thấy nhân viên phù hợp.</h3>

<%

} else {

%>

<table>

<tr>

<th>Employee ID</th>

<th>Name</th>

<th>Position</th>

<th>Salary</th>

<th>Department</th>

<th>Hire Date</th>

</tr>

<%

for (Employee emp : employeeList) {

%>

<tr>

<td><%= emp.getEmployeeId() %></td>

<td><%= emp.getName() %></td>

<td><%= emp.getPosition() %></td>

<td><%= emp.getSalary() %></td>

<td><%= emp.getDepartmentName() %></td>

<td><%= emp.getHireDate() %></td>

</tr>

<%

}

%>

</table>

<%

}

}

%>

</body>

</html>

**Giải thích**:

* Phần form nhập liệu có các trường: name, salary, fromHireDate, toHireDate, position, và departmentId (chọn từ dropdown).
* Khi người dùng bấm **Search**, form sẽ gửi GET request đến EmployeeSearchServlet.
* Servlet xử lý, truy vấn DB, trả lại List<Employee> cho JSP.
* JSP duyệt qua danh sách nhân viên và hiển thị. Nếu danh sách rỗng, in thông báo **"Không tìm thấy nhân viên phù hợp."**.

**7. Cấu hình web.xml (nếu không dùng @WebServlet)**

Nếu không dùng annotation @WebServlet("/EmployeeSearchServlet"), bạn cần cấu hình trong web.xml:

xml

<servlet>

<servlet-name>employeeSearch</servlet-name>

<servlet-class>controller.EmployeeSearchServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>employeeSearch</servlet-name>

<url-pattern>/EmployeeSearchServlet</url-pattern>

</servlet-mapping>

Nếu bạn dùng @WebServlet("/EmployeeSearchServlet") ở đầu class EmployeeSearchServlet, thì không cần khai báo trong web.xml.

**8. Chạy thử**

1. **Triển khai** (deploy) project lên Tomcat (hoặc server Servlet khác).
2. Mở trình duyệt và truy cập đường dẫn:

http://localhost:8080/EmployeeManagement/EmployeeSearch.jsp

1. Gõ thông tin tìm kiếm và bấm **Search**.
2. Kiểm tra các trường hợp:
   * Tìm kiếm theo tên (ví dụ: “kien”).
   * Tìm kiếm theo lương (salary >= 5000).
   * Tìm kiếm theo khoảng ngày tuyển dụng (From Hire Date, To Hire Date).
   * Tìm kiếm theo position.
   * Tìm kiếm theo department.
   * Không nhập gì => trả về toàn bộ danh sách.
   * Nếu không có kết quả, in ra “Không tìm thấy nhân viên phù hợp.”

**9. Các lưu ý và mở rộng**

* Khi so sánh ngày trong SQL, chú ý **định dạng**. Ở ví dụ trên, Servlet dùng Date.valueOf(...) (Java LocalDate dạng YYYY-MM-DD). Bạn cần chắc chắn rằng chuỗi date đến từ <input type="date"> cũng là yyyy-MM-dd.
* Nếu muốn hiển thị danh sách phòng ban động (thay vì hard-code), bạn có thể tạo một hàm riêng trong Servlet (hoặc DAO) để truy vấn bảng departments và đẩy danh sách Department sang JSP. Sau đó, JSP sẽ duyệt in <option> cho từng phòng ban.
* Nếu dùng Maven, nhớ cấu hình dependency MySQL trong pom.xml:

xml

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.32</version>

</dependency>

* Chú ý loại driver: com.mysql.cj.jdbc.Driver nếu bạn dùng Connector mới.

**Tóm lại**

Với cấu trúc và code ví dụ trên, bạn đã có một ứng dụng web JSP/Servlet/JDBC thỏa mãn các yêu cầu:

1. Tìm kiếm theo **Name**, **Salary >=**, **Hire Date** trong khoảng, **Position**, **Department**.
2. Sử dụng **PreparedStatement** để tránh SQL Injection.
3. Kết quả hiển thị dạng bảng, hoặc in thông báo **"Không tìm thấy nhân viên phù hợp."** khi không có dữ liệu.

Bạn chỉ cần chỉnh sửa lại cho đúng thông tin kết nối DB, tên bảng, và field nếu có sự khác biệt, sau đó triển khai lên Tomcat để kiểm tra. Chúc bạn thành công!