

**BÀI TẬP MÔN LẬP TRÌNH WWW JAVA**  
**(ADVANCED WEB PROGRAMMING WITH JAVA)**  
**HỆ: ĐẠI HỌC**  
**(DÀNH CHO CHUYÊN NGÀNH: SE)**

## **BÀI TẬP TUẦN 01-02 MÔN LẬP TRÌNH WWW JAVA..... 3**

<i>Chương 1: Application Model - Web Application Architecture .....</i>	<i>3</i>
<i>Chương 2: Java Servlets.....</i>	<i>3</i>
<i>Bài 1. Phân tích yêu cầu của Bài tập lớn .....</i>	<i>4</i>
<i>Bài 2. Layout Bài tập lớn - Bài tập cá nhân .....</i>	<i>6</i>
<i>Bài 3. Cài đặt 7.0 Server .....</i>	<i>7</i>
<i>Bài 4. Java Servlet - Thao tác với doGet(), doPost().....</i>	<i>10</i>
<i>Bài 5. Java Servlet - Filter.....</i>	<i>11</i>
<i>Bài 7. Java Servlet - Upload hình, lưu CSDL.....</i>	<i>16</i>
<i>Bài 8: Java Servlet - JavaMail API .....</i>	<i>17</i>

## **BÀI TẬP TUẦN 03-04 MÔN LẬP TRÌNH WWW JAVA..... 22**

<i>Chương 3: Jakarta Server Pages - JSPs.....</i>	<i>22</i>
<i>Bài 1. JSPs - Thao tác với Form.....</i>	<i>25</i>
<i>Bài 2. JSPs - Model View Controller .....</i>	<i>28</i>
<i>Bài 3. JSPs – Thao tác với session.....</i>	<i>31</i>
<i>Bài 4. JSPs – Thao tác với session.....</i>	<i>41</i>
<i>Bài 5: Bài tập tổng hợp .....</i>	<i>42</i>
<i>Bài 6. JSPs - Bài tập tổng hợp 2.....</i>	<i>47</i>

# BÀI TẬP TUẦN 01-02 MÔN LẬP TRÌNH WWW JAVA

## Chương 1: Application Model - Web Application Architecture

## Chương 2: Java Servlets

*Mục tiêu:*

- *(Review) Trình bày được mô hình ứng dụng Web các khái niệm liên quan.*
- *(Review) Thực hiện được layout của trang Web dùng HTML/CSS và thực hiện kiểm tra dữ liệu nhập phía Client dùng JavaScript.*
- *Hiểu được cấu trúc HTTP Request, HTTP Response.*
- *Phân tích yêu cầu theo đề tài bài tập lớn. Thực hiện các mô hình UML với yêu cầu tối thiểu.*
- *Cài đặt và cấu hình được Project với Tomcat 11 hoặc Glassfish 7.0*
- *Hiểu được một cấu trúc ứng dụng Web Back-End với Java Servlets.*
- *Thực hiện các bài tập FormData, Session Tracking, Send Mail, Upload Files với Java Servlets.*

*Yêu cầu:*

- *Tất cả các bài tập lưu trong thư mục: **T:\MSSV\_HoTen\_Tuan01***
- *Tạo Project **MSSV\_HoTen\_Tuan01** trong thư mục vừa tạo trong IDE IntelliJ Jakarta EE 11 (Jakarta Platform Enterprise Edition 11). Mỗi bài tập có thể lưu trong từng package riêng biệt.*
- *Cuối mỗi buổi thực hành, SV phải nén (.rar hoặc .zip) thư mục làm bài và nộp lại bài tập đã thực hiện trong buổi đó.*

## Bài 1. Phân tích yêu cầu của Bài tập lớn

Yêu cầu của các đề tài 01 - 50, chức năng tối thiểu:

- Website bao gồm 3 loại người dùng tương tác: người dùng không có tài khoản (guest), người dùng có tài khoản (customer), người quản trị hệ thống (admin).
- Người dùng không có tài khoản (guest) có các chức năng:
  - Xem danh sách sản phẩm (thiết bị máy tính, mỹ phẩm, quần áo ... tùy theo đề tài, danh sách này lấy từ CSDL)
  - Xem chi tiết của từng sản phẩm từ danh sách sản phẩm.
  - Chọn mua từng sản phẩm (có thể chọn mua từ trang Web danh sách sản phẩm hay từ trang Web chi tiết của từng sản phẩm), sản phẩm sau khi chọn mua sẽ được đưa vào trong giỏ hàng.
    - Xem giỏ hàng (danh sách sản phẩm đã chọn mua, thông tin này lưu trong biến Session, không cần cập nhật CSDL).
- Khi xem giỏ hàng, có thể chỉnh sửa số lượng của từng sản phẩm trong giỏ hàng (nếu chỉnh sửa số lượng là 0 ☐ bỏ sản phẩm đó ra khỏi giỏ hàng)
- Có thể đăng ký tài khoản của website với các thông tin cần thiết (email không trùng với tài khoản khác), sau khi đăng ký thành công với thông tin hợp lệ, lưu trữ CSDL + gửi email + thông báo về tài khoản.
- Người dùng có tài khoản (customer) có thể thực hiện các chức năng của Người dùng không có tài khoản (guest), ngoài ra người dùng có tài khoản (customer) còn có thể:
  - Xử lý thanh toán (chức năng này thực hiện khi giỏ hàng đã có sản phẩm và người dùng đăng nhập thành công vào hệ thống): cập nhật thông tin vào CSDL + gửi email + thông báo đăng ký đặt hàng thành công với các thông tin kèm theo. Sau khi xử lý thành công, Session được xóa về null.
- Người quản trị hệ thống (admin) có thể thực hiện được chức năng như một người dùng có tài khoản (customer). Ngoài ra, chức năng khác dành cho người quản trị hệ thống (admin) - Phần Back-End:
  - Tìm kiếm thông tin về sản phẩm/loại sản phẩm, tài khoản người dùng, các đơn đặt sản phẩm.
- Quản lý thông tin sản phẩm/loại sản phẩm:
  - Xem danh sách sản phẩm/loại sản phẩm.
  - Xem chi tiết từng sản phẩm/loại sản phẩm.
  - Xóa sản phẩm/loại sản phẩm trong trường hợp sản phẩm chưa có trong đơn hàng nào hoặc loại sản phẩm chưa có sản phẩm nào.
  - Thêm mới, cập nhật thông tin sản phẩm/loại sản phẩm.
- Quản lý thông tin tài khoản người dùng:
  - Xem danh sách các tài khoản người dùng đã đăng ký.
  - Xem chi tiết từng tài khoản người dùng, không xem được password của người dùng.
  - Xóa tài khoản người dùng nếu người dùng chưa thực hiện đặt hàng online lần nào.
  - Cập nhật thông tin tài khoản người dùng.
- Quản lý thông tin đơn hàng trực tuyến:
  - Xem danh sách các đơn hàng (sắp xếp theo ngày mua)
  - Xem chi tiết đơn hàng.

- Cập nhật số lượng của mặt hàng trong đơn hàng trực tuyến ○ Lưu ý cho các chức năng quản lý thông tin:
- Ràng buộc khi xóa dữ liệu
- Trường hợp thêm hay cập nhật dữ liệu có thể kiểm tra phía Client bằng JavaScript/jQuery hoặc kiểm tra bằng Model phía Server, không dùng Functions/Check constraints/Stored Procedures trong hệ quản trị CSDL.

Yêu cầu của các đề tài 51 □ 54, chức năng tối thiểu:

- Website bao gồm 3 loại người dùng tương tác: người dùng không có tài khoản (guest), người dùng có tài khoản (customer), người quản trị hệ thống (admin).
- Người dùng không có tài khoản (guest) có các chức năng:
  - Xem danh sách các báo giấy/tour/phòng khách sạn/dịch vụ bảo hiểm (tùy theo đề tài, danh sách này lấy từ CSDL) ○ Xem chi tiết của từng báo giấy/tour/phòng khách sạn/dịch vụ bảo hiểm từ danh sách.
  - Chọn từng tờ báo giấy/tour/phòng khách sạn/dịch vụ bảo hiểm (có thể chọn từ trang Web danh sách hay từ trang Web chi tiết), các tờ báo giấy/tour/phòng khách sạn/dịch vụ bảo hiểm sản phẩm sau khi chọn đặt sẽ được đưa vào trong giỏ hàng.
  - Xem giỏ hàng (danh sách đã chọn, thông tin này lưu trong biến Session, không cần cập nhật CSDL). ○ Khi xem giỏ hàng, có thể chỉnh sửa số lượng của từng thành phần trong giỏ hàng (nếu chỉnh sửa số lượng là 0 □ bỏ sản phẩm đó ra khỏi giỏ hàng)
  - Có thể đăng ký tài khoản của website với các thông tin cần thiết (email không trùng với tài khoản khác), sau khi đăng ký thành công với thông tin hợp lệ, lưu trữ CSDL + *gửi email* + thông báo về tài khoản.
- Người dùng có tài khoản (customer) có thể thực hiện các chức năng của Người dùng không có tài khoản (guest), ngoài ra người dùng có tài khoản (customer) còn có thể:
  - Xử lý thanh toán (chức năng này thực hiện khi giỏ hàng đã có thông tin và người dùng đăng nhập thành công vào hệ thống): cập nhật thông tin vào CSDL + *gửi email* + thông báo đăng ký đặt báo giấy/tour/phòng khách sạn/dịch vụ bảo hiểm thành công với các thông tin kèm theo. Sau khi xử lý thành công, Session được xóa về null.
- Người quản trị hệ thống (admin) có thể thực hiện được chức năng như một người dùng có tài khoản (customer). Ngoài ra, chức năng khác dành cho người quản trị hệ thống (admin) - Phần Back-End:
  - Tìm kiếm thông tin về báo giấy/tour/phòng khách sạn/dịch vụ bảo hiểm và các loại dịch vụ, tài khoản người dùng, các đơn đặt sản phẩm.
  - Quản lý thông tin báo giấy/tour/phòng khách sạn/dịch vụ bảo hiểm/loại:
    - Xem danh sách báo giấy/tour/phòng khách sạn/dịch vụ bảo hiểm/loại.
    - Xem chi tiết từng báo giấy/tour/phòng khách sạn/dịch vụ bảo hiểm/loại.
    - Xóa báo giấy/tour/phòng khách sạn/dịch vụ bảo hiểm/loại trong trường hợp thông tin cần xóa có trong đơn hàng nào hoặc loại sản phẩm chưa có sản phẩm nào.
    - Thêm mới, cập nhật thông tin sản phẩm/loại sản phẩm.
  - Quản lý thông tin tài khoản người dùng:

- Xem danh sách các tài khoản người dùng đã đăng ký.
- Xem chi tiết từng tài khoản người dùng, không xem được password của người dùng.
- Xóa tài khoản người dùng nếu người dùng chưa thực hiện đặt hàng online lần nào. □  
Cập nhật thông tin tài khoản người dùng.
- Quản lý thông tin đơn hàng trực tuyến:
  - Xem danh sách các đơn hàng (sắp xếp theo ngày mua) □ Xem chi tiết đơn hàng.
  - Cập nhật số lượng của mặt hàng trong đơn hàng trực tuyến o Lưu ý cho các chức năng quản lý thông tin:
  - Ràng buộc khi xóa dữ liệu
  - Trường hợp thêm hay cập nhật dữ liệu có thể kiểm tra phía Client bằng
  - JavaScript/jQuery hoặc kiểm tra bằng Model phía Server, không dùng Functions/Check constraints/Stored Procedures trong hệ quản trị CSDL.

## Bài 2. Layout Bài tập lớn - Bài tập cá nhân

- Layout trang web dùng HTML/CSS dùng chung cho Project bài tập lớn (dùng cho nhóm)
- Yêu cầu cho layout bài tập cá nhân hàng tuần:

<header>		
<menu>		
<nav>	Nội dung	<section>
<footer>		

Hoặc tương tự:



## Bài 3. Cài đặt 7.0 Server

Cài đặt GlassFish và thiết lập JDK 21 trong IDE IntelliJ Ultimate  
I. Tải và cài đặt GlassFish



Tải GlassFish 7 (hỗ trợ Jakarta EE 11)

- Truy cập: <https://projects.eclipse.org/projects/ee4j.glassfish/downloads>
- Tải bản ZIP hoặc TAR.GZ phù hợp hệ điều hành
- Giải nén vào thư mục ví dụ: C:\glassfish7 hoặc /opt/glassfish7

### II. Cấu hình GlassFish trong IntelliJ IDEA Ultimate

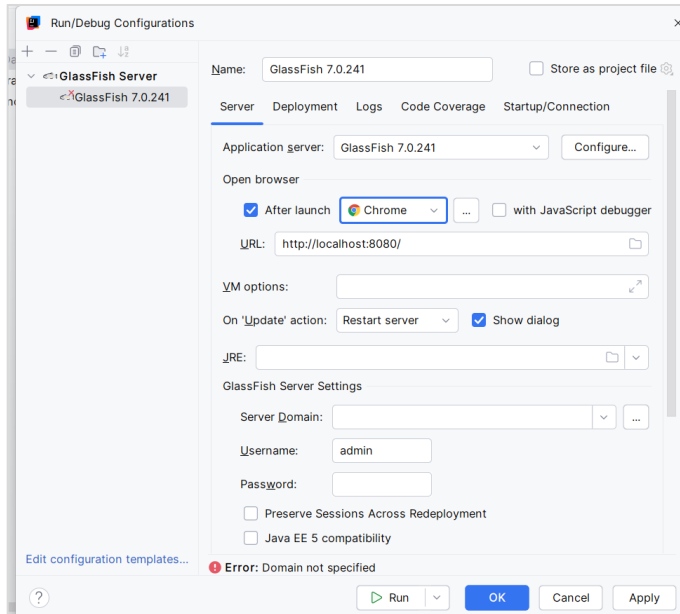
#### 1. Cấu hình Application Server

- Vào menu: File → Settings (hoặc Ctrl + Alt + S)
- Chọn: Build, Execution, Deployment → Application Servers
- Click +, chọn GlassFish Server
- Chỉ đường dẫn tới thư mục glassfish7

IntelliJ sẽ tự nhận thư mục glassfish chứa bin, lib, v.v...

#### 2. Tạo cấu hình chạy (Run Configuration)

- Vào menu: Run → Edit Configurations
- Click +, chọn GlassFish Server → Local
- Chọn cấu hình (Application Server: GlassFish bạn đã thêm ở trên)
- Ở tab Deployment, Click + → chọn Artifact hoặc WAR exploded



3.
  - Tạo project kiểu Java Enterprise
  - Chọn Jakarta EE 11
  - Module: chọn Web Application, bật các framework như JPA, Servlet, CDI nếu cần

### III. Chạy ứng dụng

1. Click Run hoặc Debug
2. IntelliJ sẽ khởi động GlassFish, triển khai ứng dụng của bạn
3. Truy cập ở <http://localhost:8080/tên-ứng-dụng>

### Cách add Tomcat 11 vào IntelliJ Ultimate



1. Chuẩn bị
  - Đảm bảo Tomcat 11 đã cài và chạy được (Java 17+).
  - Ghi nhớ đường dẫn thư mục cài Tomcat, ví dụ:
2. Mở cấu hình Application Server trong IntelliJ
  1. Mở IntelliJ → Vào menu File → Settings (hoặc Ctrl+Alt+S).
  2. Chọn mục Build, Execution, Deployment → Application Servers.
  3. Bấm dấu + → chọn Tomcat Server → Local.

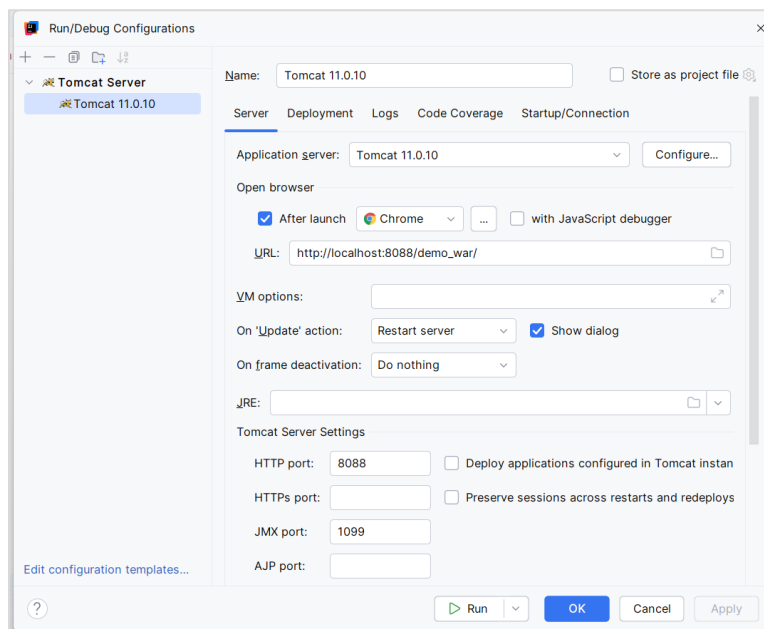


### 3. Chỉ định đường dẫn Tomcat

- Ở mục Tomcat Home → chọn thư mục cài Tomcat 11.
- IntelliJ sẽ tự nhận version, ví dụ Apache Tomcat 11.0.0.
- Ấn OK để lưu.

### 4. Tạo cấu hình chạy (Run/Debug Configuration)

1. Vào menu Run → Edit Configurations...
2. Nhấn + → chọn Tomcat Server → Local.
3. Ở tab Server:
  - Application server: chọn Tomcat 11 bạn vừa add ở bước 3.
  - HTTP port: mặc định 8080 (bạn có thể đổi nếu port này bị chiếm).
4. Ở tab Deployment:
  - Nhấn + → chọn Artifact hoặc External Source (tùy project của bạn).
  - Nếu là Maven/Gradle → chọn artifact dạng war exploded hoặc war.
5. Ấn OK để lưu cấu hình.



### 5. Chạy thử

- Chọn cấu hình Tomcat vừa tạo ở góc trên phải IntelliJ.
- Bấm Run (Shift+F10).
- Truy cập: <http://localhost:8080> để xem Tomcat chạy.

## Bài 4. Java Servlet - Thao tác với doGet(), doPost()

Bài tập dùng Servlet truyền dữ liệu thông qua Form Data. Tạo form đăng ký thông tin bao gồm các thành phần TextBox, CheckBox, ComboBox, TextArea.

### HTML Form Example with File Upload

Name:

Password:

Gender: ☒ Male ☐ Female

Hobbies: ☐ Reading ☐ Sports ☐ Music

Country:

Birth Date:

Profile Picture:  Không có tệp nào được chọn

Hướng dẫn:

Bước 1: Tạo trang jsp chứa HTML form.

```
<!-- enctype bắt buộc khi upload file -->
<form action="${pageContext.request.contextPath}/processFormUpload"
method="post" enctype="multipart/form-data">
```

Bước 2: Tạo Servlet `@WebServlet("/processFormUpload")` lấy dữ liệu từ form, xuất ra text.

```
@WebServlet("/processFormUpload")
@MultipartConfig(
    fileSizeThreshold = 1024 * 1024, // 1MB
    maxFileSize = 1024 * 1024 * 10, // 10MB
    maxRequestSize = 1024 * 1024 * 15 // 15MB
)
public class FormUploadServlet extends HttpServlet {
```

Lấy data từ thành phần của form:

(Lưu ý các thành phần form text, radio, checkbox, Selection list, ...)

`req.getParameter();` → Trả về String

`req.getParameterValues();` → Trả về mảng String

```

req.setCharacterEncoding("utf-8");
String name = req.getParameter(s: "name");
String password = req.getParameter(s: "password");
String gender = req.getParameter(s: "gender");
String[] hobbies = req.getParameterValues(s: "hobbies");
String country = req.getParameter(s: "country");
String birthDate = req.getParameter(s: "birthDate");

...

```

### *Lấy file và lưu file:*

```

//lấy File
Part filePart = req.getPart(s: "profilePic");
String fileName = filePart.getSubmittedFileName();

// lưu vào thư mục uploads
String uploadPath = System.getProperty("user.home") + File.separator + "uploads";

File uploadDir = new File(uploadPath);
if (!uploadDir.exists()) {
    uploadDir.mkdir();
}

//lưu file vào thư mục
filePart.write(s: uploadPath + File.separator + fileName);

```

### *Xuất dữ liệu:*

```

resp.setContentType("text/html;charset=UTF-8");
resp.getWriter().println("<h2>Form Data Received:</h2>");
resp.getWriter().println("Name: " + name + "<br>");
resp.getWriter().println("Password: " + password + "<br>");
resp.getWriter().println("Gender: " + gender + "<br>");
resp.getWriter().println("Hobbies: " + (hobbies != null ? String.join(delimiter: ", ", hobbies) : "None") + "<br>");
resp.getWriter().println("Country: " + country + "<br>");
resp.getWriter().println("Birth Date: " + birthDate + "<br>");
resp.getWriter().println("Uploaded File: " + (fileName != null ? fileName : "No file") + "<br>");
resp.getWriter().println("Saved to: " + uploadPath + "<br>");

```

## Bài 5. Java Servlet - Filter

Thực hiện Filter qua ứng dụng mô tả như sau: Filter trong Servlet kiểm tra đăng nhập: Nếu đúng thông tin đăng nhập thì thực hiện các chức năng của servlet trong thư mục /secure.

Các file cần thực hiện:

- AuthFilter.java

```
@WebFilter("/secure/*")
```

```
public class AuthFilter implements Filter {
```

```
    @Override
```

```
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
```

```
    {  
        HttpServletRequest req = (HttpServletRequest) request;
```

```
        HttpServletResponse res = (HttpServletResponse) response;
```

```
        HttpSession session = req.getSession(create: false);
```

```
        boolean loggedIn = (session != null && session.getAttribute(name: "user") != null);
```

```
        if (loggedIn) {
```

```
            chain.doFilter(request, response); // Cho phép đi tiếp
```

```
        } else {
```

```
            res.sendRedirect(location: req.getContextPath() + "/login.jsp");
```

```
        }
```

```
    }
```

```
}
```

*Filter lọc hết các request, chỉ chấp nhận các request trong thư mục /secure*

- *login.jsp* tương ứng cho phần xử lý backend *loginServlet.java*.

*login.jsp* (SV tự thiết kế)

## Đăng nhập

Tên đăng nhập:

Mật khẩu:

LoginServlet.java

```

@WebServlet("/login")
public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L; no usages
    public LoginServlet() { } no usages

    @Override 1 usage
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
//      super.doPost(req, resp);
    String username = req.getParameter( name: "username");
    String password = req.getParameter( name: "password");
    // username = admin, password=123
    if("admin".equals(username) && "123".equals(password)) {
        HttpSession session = req.getSession();
        session.setAttribute( name: "user", username);
        resp.sendRedirect( location: req.getContextPath() + "/home.jsp");
    }else
    {
        req.setAttribute( name: "error", o: "Sai tài khoản");
        req.getRequestDispatcher( path: "/login.jsp").forward(req, resp);
    }
}
}

```

#### - LogoutServlet.java

```

@WebServlet("/logout")
public class LogoutServlet extends HttpServlet {

    @Override 5 usages
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
//      super.doGet(req, resp);
    HttpSession session = req.getSession( create: false);
    if (session != null) {
        session.invalidate();
    }
    resp.sendRedirect( location: req.getContextPath() + "/login.jsp");
}
}

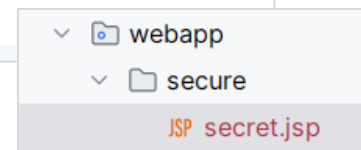
```

#### - home.jsp cho phép đăng nhập bị chặn theo AuthFilter.java

```

<h2>Xin chào, ${sessionScope.user}</h2>
<a href="${pageContext.request.contextPath}/secure/secret.jsp">Trang bảo mật</a><br>
<a href="${pageContext.request.contextPath}/logout">Đăng xuất</a>

```



- /secure/secret.jsp: chỉ cho phép các Servlet trong thư mục /secure thực hiện theo Filter, cấu trúc thư mục /secure trên Project như sau:

secret.jsp

```
<h2>Đây là nội dung bí mật chỉ user đã login mới thấy!</h2>
<a href="${pageContext.request.contextPath}/home.jsp">Về trang chủ</a>
```

## **Bài 6. Java Servlet - Upload files**

Thực hiện upload nhiều file lưu trữ ở Server.

Chức năng **upload nhiều file** trên Tomcat 11 bằng @MultipartConfig và Servlet.

*Hướng dẫn*

*Bước 1. Tạo trang .jsp chứa HTML form. Lưu ý thuộc tính của form enctype="multipart/form-data"*

```
<form action="${pageContext.request.contextPath}/uploadmulti" method="post" enctype="multipart/form-data">
  File #1: <input type="file" name="file"/><br/><br/>
  File #2: <input type="file" name="file"/><br/><br/>
  File #3: <input type="file" name="file"/><br/><br/>
  File #4: <input type="file" name="file"/><br/><br/>
  File #5: <input type="file" name="file"/><br/><br/>
  <input type="submit" value="Upload"/>
  <input type="reset" value="Reset"/>
</form>
```

*Bước 2: Tạo Servlet xử lý :*

- Đọc dữ liệu của form: dữ liệu text + dữ liệu file

```

@WebServlet("/uploadmulti")
@MultipartConfig
    (
        fileSizeThreshold = 1024 * 1024,
        maxFileSize = 1024 * 1024 * 10,
        maxRequestSize = 1024 * 1024 * 50
    )

public class MultiFileUploadServlet extends HttpServlet {
    private static final long serialVersionUID = 1L; no usages
    public MultiFileUploadServlet() {} no usages
    private static final String UPLOAD_DIR = "uploads"; 1 usage

@Override 1 usage
protected void doPost(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {
    super.doPost(req, resp);
    // Thư mục lưu file trên server (trong thư mục webapp)
    String uploadPath = getServletContext().getRealPath(s: "") + UPLOAD_DIR;
    //File.separator +

    File uploadDir = new File(uploadPath);
    if (!uploadDir.exists()) {
        uploadDir.mkdir();
    }

    // Lấy tất cả file từ request
    for (Part part : req.getParts()) {
        String fileName = getFileName(part);
        if (fileName != null && !fileName.isEmpty()) {
            part.write(s: uploadPath + File.separator + fileName);
        }
    }

    resp.setContentType("text/html;charset=UTF-8");
    resp.getWriter().println("<h3>Files uploaded successfully to " + uploadPath + "</h3>");
}

```

```

private String getFileName(Part part) { 1 usage
    String contentDisp = part.getHeader(s: "content-disposition");
    String[] tokens = contentDisp.split(regex: ";");
    for (String token : tokens) {
        if (token.trim().startsWith("filename")) {
            return token.substring(token.indexOf("=") + 2, token.length() - 1)
        }
    }
    return null;
}
}
}

```

## Bài 7. Java Servlet - Upload hình, lưu CSDL

Thực hiện trang Web cho phép upload hình ảnh và lưu dạng file name trong CSDL MariaDB/SQL Server.

Lưu ý: trong file pom.xml add thêm dependencies cho MariaDB/SQL Server

- MariaDB: mariadb-java-client (version 3.5.0)
- SQL Server: mssql-jdbc (chú ý JDK tương ứng)

▪ CSDL: (MariaDB: storedb table users)

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default
1	ID	INT	10	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCRE...
2	FIRSTNAME	VARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
3	LASTNAME	VARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
4	PICFILE	VARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL

Hướng dẫn:

Bước 1: Tạo trang .jsp chứa các thành phần HTML form. Lưu ý thuộc tính của form `enctype="multipart/form-data"`



```
<form action="${pageContext.request.contextPath}/uploaddatabase" method="post" enctype="multipart/form-data">
    First Name: <input type="text" name="firstName"><br><br>
    Last Name: <input type="text" name="lastName"><br><br>
    Portrait Photo: <input type="file" name="photo"><br><br>
    <input type="submit" value="Save">
</form>
```

*Bước 2: Tạo Servlet xử lý:*

- *Đọc dữ liệu của form: dữ liệu text + dữ liệu file*

```
String firstName = req.getParameter( s: "firstName");
String lastName = req.getParameter( s: "lastName");
Part filePart = req.getPart( s: "photofile");
String filename = filePart.getSubmittedFileName();
```

- *Kết nối CSDL : jdbc kết nối CSDL*

```
// Kết nối theo mariadb database storedb
String url = "jdbc:mariadb://localhost:3306/storedb";
String dbUser = "root";
String dbPass = "root";
```

- *Insert dữ liệu vào bảng gồm cả dữ liệu Text và dữ liệu Image*

```
Class.forName( className: "com.mysql.jdbc.Driver");
Connection conn = DriverManager.getConnection(url, dbUser, dbPass);

String sql = "INSERT INTO users (FIRSTNAME, LASTNAME, PICFILE) values (?, ?, ?)";
PreparedStatement statement = conn.prepareStatement(sql);

statement.setString( parameterIndex: 1, firstName);
statement.setString( parameterIndex: 2, lastName);
statement.setString( parameterIndex: 3, filename);
statement.executeUpdate();
```

## Bài 8: Java Servlet - JavaMail API

Gửi mail trong Servlet dùng JavaMail API và Java Activation Framework (JAF). Form gửi mail bao gồm tiêu đề, người nhận, nội dung và file đính kèm.

### JavaMail hỗ trợ nhận gửi các loại mail:

1. SMTP (Simple Mail Transfer Protocol) → gửi email từ ứng dụng của bạn đến máy chủ (SMTP server).

2. POP3 (Post Office Protocol v3) → nhận email theo kiểu tải về hộp thư INBOX; ít tính năng, thường không đồng bộ trạng thái đọc/chưa đọc tốt.
3. IMAP (Internet Message Access Protocol) → nhận email, làm việc trực tiếp trên server (nhiều thư mục, cờ/nhãn, search, xem từng phần, tải từng đính kèm...). \*\*\*

## Lớp trong JavaMail

- Session: Là cấu hình cho một “phiên làm việc gửi/nhận mail”: **host, port, TLS/SSL**, có cần auth không, debug, đại diện cho một lần gửi nhận mail

```
Properties p = new Properties();  
p.put("mail.transport.protocol", "smtp");  
p.put("mail.smtp.host", "smtp.gmail.com");  
p.put("mail.smtp.port", "587");  
p.put("mail.smtp.auth", "true");  
p.put("mail.smtp.starttls.enable", "true");  
Session session = Session.getInstance(p, new Authenticator() {  
    @Override protected PasswordAuthentication getPasswordAuthentication() {  
        return new PasswordAuthentication("user@gmail.com", "app-password");  
    }  
});
```

- Message/ MimeMessage: Các thành phần cho **một email**: người gửi/nhận, tiêu đề (subject), nội dung (body), **multipart**, đính kèm...

```
MimeMessage msg = new MimeMessage(session);  
msg.setFrom(new InternetAddress("user@gmail.com"));  
msg.addRecipient(Message.RecipientType.TO, new  
InternetAddress("friend@example.com"));  
msg.setSubject("Chào bạn", "UTF-8");
```

// Phần thân HTML

```
MimeBodyPart body = new MimeBodyPart();  
body.setContent("<b>Xin chào</b>, đây là mail test.", "text/html; charset=UTF-8");
```

// Đính kèm

```
MimeBodyPart attach = new MimeBodyPart();
```

```
attach.setDataHandler(new DataHandler(new ByteArrayDataSource(fileBytes,
"application/pdf")));
attach.setFileName("tai-lieu.pdf");
```

// Gộp multipart

```
MimeMultipart mp = new MimeMultipart();
mp.addBodyPart(body);
mp.addBodyPart(attach);
msg.setContent(mp);
```

▪ Transport: Thực thể **gửi** Message qua Internet dựa trên cấu hình Session (giao thức SMTP)

```
Transport t = session.getTransport("smtp");
t.connect();
t.sendMessage(msg, msg.getAllRecipients());
t.close();
```

▪ Address: Internet Address là đối tượng thực thi tạo các địa chỉ cho việc gửi nhận email trên Internet.

```
InternetAddress[] to = InternetAddress.parse("a@x.com,b@y.com");
msg.setRecipients(Message.RecipientType.TO, to);
```

▪ Authenticator: xác thực tài khoản

Cách bạn **cấp thông tin đăng nhập** cho Session.

Trả về new PasswordAuthentication(username, password) khi server yêu cầu.

## Luồng gửi mail trong Servlet

<ol style="list-style-type: none"> <li>1. <b>User nhập form JSP</b> → to, subject, content, file.</li> <li>2. <b>Servlet nhận request</b>, tạo Session với SMTP.</li> <li>3. Tạo Message → set From, To, Subject, Body.</li> <li>4. Nếu có file đính kèm → dùng MimeBodyPart + Multipart.</li> <li>5. Gọi Transport.send(message).</li> <li>6. Hiển thị kết quả cho người dùng.</li> </ol>	<ol style="list-style-type: none"> <li>1. Form HTML (multipart/form-data) chứa: To, Subject, Body, và &lt;input type="file" multiple&gt;.</li> <li>2. Servlet gắn @MultipartConfig để đọc Part của tệp.</li> <li>3. Tạo Session SMTP (host/port/TLS + Authenticator).</li> <li>4. Tạo MimeMessage: <ul style="list-style-type: none"> <li>- Một MimeBodyPart cho body (plain text hoặc HTML).</li> <li>- Với mỗi file: tạo MimeBodyPart + DataHandler (JAF) → addBodyPart.</li> </ul> </li> <li>5. Transport.send(msg) để gửi.</li> </ol>
--	---

Add Dependencies:

```
jakarta.activation-api 2.1.3
angus-activation 2.0.2
jakarta.mail-api 2.1.3
angus-mail 2.0.3
```

localhost:8088/demoCDI\_war\_exploded/sendmail.jsp

Người nhận:

Tiêu đề:

Nội dung:

File đính kèm:  Không có tệp nào được chọn

localhost:8088/demoCDI\_war\_exploded/sendMail

**Gửi mail thành công!**

<https://myaccount.google.com>

google app password:

## SendMailServlet.java

```
@WebServlet("/sendMail")
@MultipartConfig
public class SendMailServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private final String USERNAME = "abc@gmail.com"; // đổi thành email thật
    private final String PASSWORD = "yvnp dlla supw ckgk";
    // Bật App password xác thực password mail (Gmail thật)
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        String to = request.getParameter("to");
        String subject = request.getParameter("subject");
        String content = request.getParameter("content");
        Part filePart = request.getPart("file"); // file upload
        try {
            // 1. Cấu hình SMTP
            Properties props = new Properties();
            props.put("mail.smtp.host", "smtp.gmail.com");
            props.put("mail.smtp.port", "587");
            props.put("mail.smtp.auth", "true");
            props.put("mail.smtp.starttls.enable", "true");
            // 2. Tạo Session có xác thực
            Session session = Session.getInstance(props, new Authenticator() {
                protected PasswordAuthentication getPasswordAuthentication() {
                    return new PasswordAuthentication(USERNAME, PASSWORD);
                }
            });
```

```

// 3. Tạo message
MimeMessage message = new MimeMessage(session);
message.setFrom(new InternetAddress(USERNAME));
message.setRecipients(Message.RecipientType.TO, InternetAddress.parse(to));
message.setSubject(subject, "UTF-8");
// 4. Nội dung chính (text)
MimeBodyPart textPart = new MimeBodyPart();
textPart.setText(content, "UTF-8");
// 5. File đính kèm (nếu có)
Multipart multipart = new MimeMultipart();
multipart.addBodyPart(textPart);
if (filePart != null && filePart.getSize() > 0) {
    MimeBodyPart attachPart = new MimeBodyPart();
    String fileName = filePart.getSubmittedFileName();
    InputStream fileContent = filePart.getInputStream();
    attachPart.setFileName(fileName);
    attachPart.setDataHandler(new DataHandler(new ByteArrayDataSource(fileContent,
        getServletContext().getMimeType(fileName))));
    multipart.addBodyPart(attachPart); }
// 6. Gán multipart vào message
message.setContent(multipart);
// 7. Gửi mail
Transport.send(message);
response.setContentType("text/html; charset=UTF-8");
response.getWriter().println("<h3>Gửi mail thành công!</h3>");
} catch (Exception e) {
    throw new ServletException("Lỗi gửi mail: " + e.getMessage(), e);
}
}
}

```

# BÀI TẬP TUẦN 03-04-05 MÔN LẬP TRÌNH WWW JAVA

## Chương 3: Jakarta Server Pages - JSPs

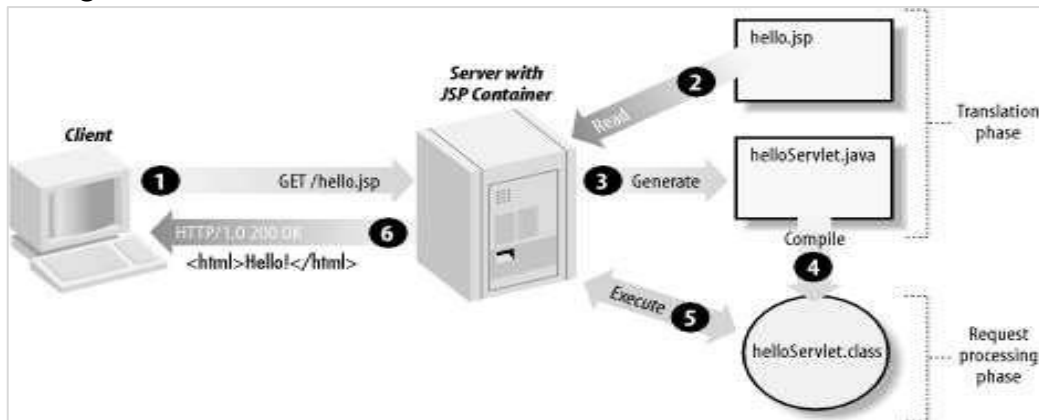
Mục tiêu:

- Hiểu và áp dụng được các cú pháp căn bản của JSPs trong việc xây dựng ứng dụng Web.
- Hiểu và áp dụng được các đối tượng ngầm định (implicit objects).
- Hiểu và áp dụng được JavaBean vào ứng dụng JSP.
- Hiểu và áp dụng được Expression Language vào ứng dụng Web.
- JSP Standard Tag Library (JSTL)

Yêu cầu:

- Tất cả các bài tập lưu trong thư mục: *T:\MaSV\_HoTen\_Tuan03*
- Tạo Project **MaSV\_HoTen\_Tuan03** trong thư mục trên trong IDE IntelliJ Jakarta EE.  
**Mỗi bài tập lưu trong từng project riêng biệt**
- SV phải nén (.rar hoặc .zip) thư mục làm bài và nộp LMS trong buổi đó.

JSP processing



1. Gửi một **yêu cầu HTTP** (HTTP request) đến Web server với trang **hello.jsp**.
2. Web server nhận ra rằng yêu cầu HTTP này là cho một trang JSP và chuyển tiếp nó đến **JSP engine**. Việc này được thực hiện vì URL hoặc tên trang kết thúc bằng **.jsp** thay vì **.html**.
3. **JSP engine** tải trang JSP từ đĩa và **chuyển đổi nó thành nội dung servlet**. Phần code sẽ hiện thực hành vi động (dynamic behavior) tương ứng của trang.
4. JSP engine biên dịch servlet này thành một **lớp thực thi (executable class)** và chuyển tiếp yêu cầu ban đầu đến **servlet engine**.
5. **Servlet engine** nạp lớp Servlet và thực thi nó. Trong quá trình thực thi, servlet tạo ra một đầu ra dưới dạng **HTML**. Đầu ra này sau đó được servlet engine gửi về Web server trong một **HTTP response**.
6. Web server chuyển tiếp **HTTP response** đó đến trình duyệt của bạn dưới dạng nội dung HTML tĩnh. Cuối cùng, trình duyệt web xử lý trang HTML được tạo ra động bên trong HTTP response giống hệt như khi xử lý một trang tĩnh.

## Standard Tag Library (JSTL) trong JSP phân loại

Functional Area	URI	Prefix
core	jakarta.tags.core	c
XML processing	jakarta.tags.xml	x
capable formatting	jakarta.tags.fmt	fmt
relational db access (SQL)	jakarta.tags.sql	sql
Functions	jakarta.tags.functions	fn

### Core Tags (JSTL)

```
<%@ taglib prefix="c" uri="jakarta.tags.core" %>
```

Thẻ	Miêu tả
<b>&lt;c:out &gt;</b>	Giống <%= ... >, nhưng cho các Expression
<b>&lt;c:set &gt;</b>	Thiết lập kết quả của 1 Expression trong một 'scope'
<b>&lt;c:remove &gt;</b>	Gỡ bỏ một biến mục tiêu (từ một biến scope cụ thể, nếu đã xác định)
<b>&lt;c:catch&gt;</b>	Bắt bất kỳ Throwable
<b>&lt;c:if&gt;</b>	Thẻ điều kiện đơn giản.
<b>&lt;c:choose&gt;</b>	Thẻ điều kiện đơn giản mà thiết lập một context cho các hoạt động điều kiện loại trừ, được đánh dấu bởi <when> và <otherwise>
<b>&lt;c:when&gt;</b>	Thẻ phụ của <choose> mà điều kiện được ước lượng là true
<b>&lt;c:otherwise &gt;</b>	Thẻ phụ của <choose> mà theo sau thẻ <when> và chỉ chạy nếu tất cả điều kiện là 'false'
<b>&lt;c:import&gt;</b>	Dùng để import.
<b>&lt;c:forEach &gt;</b>	Thẻ lặp cơ bản.
<b>&lt;c:forTokens&gt;</b>	Lặp qua các token, được phân biệt bởi các dấu phân tách (delimiter) đã cung cấp
<b>&lt;c:param&gt;</b>	Thêm một parameter tới một URL của thẻ đang chứa 'import'
<b>&lt;c:redirect &gt;</b>	Redirect tới một URL mới
<b>&lt;c:url&gt;</b>	Tạo một URL với các tham số truy vấn tùy ý

### Formatting Tags (JSTL)

Cú pháp

```
<%@ taglib prefix="fmt" uri=" jakarta.tags.fmt" %>
```

Thẻ	Miêu tả
<b>&lt;fmt:formatNumber&gt;</b>	Trả lại giá trị số với định dạng cụ thể

<b>&lt;fmt:parseNumber&gt;</b>	Parse biểu diễn chuỗi của một số, tiền tệ, phần trăm
<b>&lt;fmt:formatDate&gt;</b>	Định dạng một date/time bởi sử dụng Style và Pattern đã cho
<b>&lt;fmt:parseDate&gt;</b>	Parse biểu diễn chuỗi của một date/time
<b>&lt;fmt:bundle&gt;</b>	Gán một Resource Bundle để được sử dụng bởi phần thân thẻ
<b>&lt;fmt:setLocale&gt;</b>	Lưu giữ Locale đã cho trong biến cấu hình locale
<b>&lt;fmt:setBundle&gt;</b>	Gán một Resource Bundle và lưu giữ trong biến scope đã đặt tên hoặc biến cấu hình bundle
<b>&lt;fmt:timeZone&gt;</b>	Xác định timezone cho bất kỳ định dạng time nào hoặc parse các action được lập trong phần thân của tag.
<b>&lt;fmt:setTimeZone&gt;</b>	Gán timezone đã cung cấp biến cấu hình time zone đó
<b>&lt;fmt:message&gt;</b>	Hiển thị một thông báo đa ngôn ngữ
<b>&lt;fmt:requestEncoding&gt;</b>	Thiết lập mã hóa ký tự cho request

## SQL Tags (JSTL)

Cú pháp

```
<%@ taglib prefix="sql" uri="jakarta.tags.sql" %>
```

Thẻ	Miêu tả
<b>&lt;sql:setDataSource&gt;</b>	Tạo một DataSource kết nối vào hệ quản trị cơ sở dữ liệu.
<b>&lt;sql:query&gt;</b>	Thực thi SQL query được định nghĩa trong tag đóng/mở hoặc thông qua thuộc tính sql
<b>&lt;sql:update&gt;</b>	Thực thi SQL update được định nghĩa trong tag đóng/mở hoặc thông qua thuộc tính sql
<b>&lt;sql:param&gt;</b>	Thiết lập một parameter trong một lệnh SQL
<b>&lt;sql:dateParam&gt;</b>	Thiết lập một parameter trong một lệnh SQL tới giá trị java.util.Date đã xác định
<b>&lt;sql:transaction &gt;</b>	Cung cấp các phần tử database action được lập với một Connection đã chia sẻ, thiết lập để thực thi tất cả các lệnh

## XML Tags (JSTL)

Cú pháp

```
<%@ taglib prefix="x" uri=" jakarta.tags.xml" %>
```

Thẻ	Miêu tả
<b>&lt;x:out&gt;</b>	Giống <%= ... >, nhưng dành cho các XPath Expression
<b>&lt;x:parse&gt;</b>	Sử dụng để <i>parse</i> XML data được xác định hoặc thông qua một thuộc tính hoặc trong phần thân thẻ
<b>&lt;x:set &gt;</b>	Thiết lập một biến tới giá trị của một XPath expression



<b>&lt;x:if &gt;</b>	Ước lượng XPath Expression và nếu nó là true, thì xử lý phần thân thẻ. Nếu điều kiện là false, phần thân bị bỏ qua
<b>&lt;x:forEach&gt;</b>	Lặp qua các node trong một tài liệu XML
<b>&lt;x:choose&gt;</b>	Điều kiện đơn giản mà thiết lập một context cho hoạt động điều kiện loại trừ, được đánh dấu bởi <when> và <otherwise> trong JSTL
<b>&lt;x:when &gt;</b>	Thẻ phụ của <choose>.
<b>&lt;x:otherwise &gt;</b>	Thẻ phụ của <choose>.
<b>&lt;x:transform &gt;</b>	Áp dụng một phép biến đổi XSL trên một tài liệu XML
<b>&lt;x:param &gt;</b>	Sử dụng cùng với thẻ transform để thiết lập một parameter trong XSLT stylesheet

## *JSTL Functions*

Cú pháp

```
<%@ taglib prefix="fn" uri=" jakarta.tags.functions" %>
```

Hàm	Miêu tả
<b>Hàm fn:contains()</b>	Kiểm tra nếu một chuỗi input chứa chuỗi phụ đã cho
<b>Hàm fn:containsIgnoreCase()</b>	Kiểm tra nếu một chuỗi input chứa chuỗi phụ đã cho trong trường hợp không phân biệt kiểu chữ
<b>Hàm fn:endsWith()</b>	Kiểm tra nếu một chuỗi input kết thúc với suffix đã cho
<b>Hàm fn:escapeXml()</b>	Các ký tự thoát mà có thể được phiên dịch như XML markup
<b>Hàm fn:indexOf()</b>	Trả về index bên trong một chuỗi về sự xuất hiện đầu tiên của chuỗi phụ
<b>Hàm fn:join()</b>	Kết hợp tất cả phần tử trong một mảng thành một chuỗi
<b>Hàm fn:length()</b>	Trả về số item trong một tập hợp, hoặc số ký tự trong một chuỗi
<b>Hàm fn:replace()</b>	Trả về một chuỗi là kết quả của việc thay thế một chuỗi input với một chuỗi đã cho
<b>Hàm fn:split()</b>	Chia một chuỗi thành một mảng các chuỗi phụ
<b>Hàm fn:startsWith()</b>	Kiểm tra nếu một chuỗi input bắt đầu với prefix đã cho
<b>Hàm fn:substring()</b>	Trả về một tập con của một chuỗi
<b>Hàm fn:substringAfter()</b>	Trả về một tập con của một chuỗi ở sau một chuỗi phụ đã cho
<b>Hàm fn:substringBefore()</b>	Trả về một tập con của một chuỗi ở trước một chuỗi phụ đã cho
<b>Hàm fn:toLowerCase()</b>	Biến đổi tất cả ký tự của một chuỗi thành chữ thường
<b>Hàm fn:toUpperCase()</b>	Biến đổi tất cả ký tự của một chuỗi thành chữ hoa
<b>Hàm fn:trim()</b>	Gỡ bỏ các khoảng trống trắng từ hai đầu của một chuỗi

## *Bài 1. JSPs - Thao tác với Form*

Các bài tập Java Servlets thực hiện bằng JSPs.

Thực hiện form đăng ký khóa học cho sinh viên:

### Student Registration Form

First Name

Last Name

Date of Birth

dd/mm/yyyy

Email

Mobile

Gender

☐ Male
☐ Female

Address

City

Pin Code

State

Country

Hobbies

☐ Drawing
☐ Singing
☐ Dancing
☐ Sketching
☐ Others

Qualification

Sl.No	Examination	Board	Percentage	Year of Passing
1	Class X			
2	Class XII			

Course applies for

☐ BCA
☐ B.Com
☐ B.Sc
☐ B.A

Submit

Reset

*Hướng dẫn:*

*Bước 1. Tạo class Student bao gồm các thuộc tính mô tả trên form, các thuộc tính khai báo private kèm theo các phương thức get/set + constructors.*

*Bước 2. Tạo form đăng ký .jsp có minh họa 1 số thành phần form như sau: (Theo form trên sinh viên thiết kế thêm các thành phần trên form)*

```

<form action="student" method="get">
<input type="text" name="firstName" class="form-control" maxLength="30" required>
<div class="form-check form-check-inline">
  <input type="checkbox" class="form-check-input" name="hobbies" value="Drawing"> Drawing
</div>
<div class="form-check form-check-inline">
  <input type="checkbox" class="form-check-input" name="hobbies" value="Singing"> Singing
</div>
<div class="form-check form-check-inline">
  <input type="checkbox" class="form-check-input" name="hobbies" value="Dancing"> Dancing
</div>
<div class="form-check form-check-inline">
  <input type="checkbox" class="form-check-input" name="hobbies" value="Sketching"> Sketching
</div>
<div class="form-check form-check-inline">
  <input type="checkbox" class="form-check-input" name="hobbies" value="Others"> Others
</div>

```

*Bước 3. Tạo Servlet lấy dữ liệu từ các thành phần của form từ trang **student.jsp**:  
Lấy dữ liệu từ các thành phần trên form như textbox, radio, checkbox, select...*

```
String fname = req.getParameter( name: "firstName");
student.setFirstName(fname);

String[] hobbies = req.getParameterValues( name: "hobbies");
student.setHobbies(hobbies);
```

*Minh họa Servlet như sau:*

```
@WebServlet("/student") new *
public class StudentServlet extends HttpServlet {

    public StudentServlet() { no usages new *
        super();
    }

    @Override 5 usages new *
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

        Student student = new Student();

        String fname = req.getParameter( name: "firstName");
        student.setFirstName(fname);

        .....
        req.setAttribute( name: "student", student);
        RequestDispatcher rd = req.getRequestDispatcher( path: "student-result.jsp");
        rd.forward(req, resp);
    }
}
```

*Bước 4. Tạo trang **student-result.jsp** hiển thị kết quả*

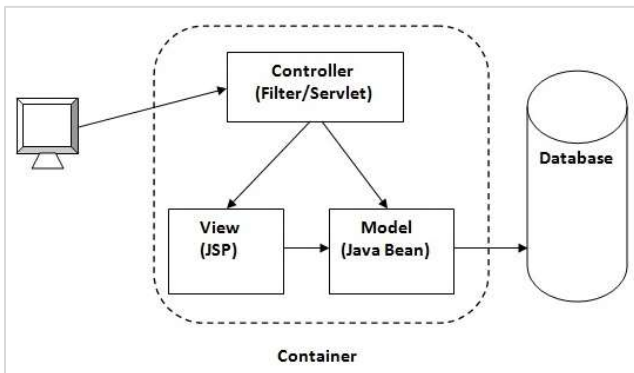
*Sử dụng các taglib xuất dữ liệu trong trang jsp, sử dụng Expression Language  $\${expression}$*

```
<h2>Student Information</h2>
<p><b>Name:</b>  $\${student.firstName}$   $\${student.lastName}$ </p>
<p><b>Date of Birth:</b>  $\${student.dob}$ </p>
<p><b>Email:</b>  $\${student.email}$ </p>
<p><b>Mobile:</b>  $\${student.mobile}$ </p>
<p><b>Gender:</b>  $\${student.gender}$ </p>
<p><b>Address:</b>  $\${student.address}$ ,  $\${student.city}$ ,  $\${student.state}$ ,  $\${student.country}$ </p>
<p><b>Course:</b>  $\${student.course}$ </p>

<h3>Hobbies:</h3>
<ul>
    <c:forEach var="h" items=" $\${student.hobbies}$ ">
        <li> $\${h}$ </li>
    </c:forEach>
</ul>
```

## Bài 2. JSPs - Model View Controller

- Thực hiện form đăng ký tài khoản với JSPs.
- Sau khi đăng ký thành công cập nhật danh sách tài khoản (không hiển thị password)
- Thực hiện bài tập theo mô hình MVC.
- Kết nối database MariaDB/SQL Server khi Sign Up **lưu thông tin đăng ký vào database**



Database *storedb* có table *accounts*

Name:

accounts

Comment:

Columns:

+

Add

✖


Remove

▲

Up

▼

Down

#	Name	Datatype	Length/S
 1	ID	INT	10
2	FIRSTNAME	VARCHAR	50
3	LASTNAME	VARCHAR	50
4	EMAIL	VARCHAR	50
5	PASSWORD	VARCHAR	50
6	DATEOFBIRTH	DATE	

### User Registration Form

Birthdate

Month▼

Day▼

Year▼

Gender

☐ Female ☐ Male

Sign Up

```
public class Account { 9 usages new *
    private int id; 4 usages
    private String firstname; 5 usages
    private String lastname; 5 usages
    private String email; 6 usages
    private String password; 5 usages
    private Date dateOfBirth; 5 usages
}
```

Hướng dẫn:

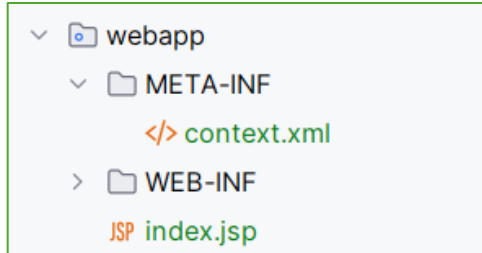
Bước 0: trong file pom.xml thêm Dependencies jstl, mariadb thực hiện kết nối Database

```
jakarta.servlet.jsp.jstl-api (3.0.2)
```

```
jakarta.servlet.jsp.jstl (3.0.1)
```

```
mariadb-java-client 3.5.5
```

Thiết lập connection Pool trong file webapp/META-INF/context.xml



Bước 1: Tạo class Account.java gồm các thuộc tính tương ứng thành phần như form minh họa (SV tự làm)

Bước 2: Tạo class AccountUtil.java: thực hiện việc truy cập database

Sử dụng DataSource cho việc kết nối với Connection Pool trong webapp\META-INF\context.xml

Các phương thức thực hiện lấy toàn bộ danh sách account – câu query select, thêm account mới – câu query insert into

### AccountUtil.java

```
public class AccountUtil {
9
10     private DataSource datasource;
11
12     public AccountUtil(DataSource datasource) throws Exception {
13         this.datasource = datasource;
14     }
15     // Lấy danh sách account
16     public List<Account> getAccounts() throws Exception {
17         List<Account> accounts = new ArrayList<>();
18
19         Connection conn = null;
20         Statement stmt = null;
21         ResultSet rs = null;
22
23         try {
24             conn = datasource.getConnection();
25             String sql = "SELECT * FROM accounts ORDER BY ID";
26             stmt = conn.createStatement();
27             rs = stmt.executeQuery(sql);
28             while (rs.next()) {
29                 int id = rs.getInt("ID");
30                 String fname = rs.getString("FIRSTNAME");
31                 String lname = rs.getString("LASTNAME");
```

```

32         String email = rs.getString("EMAIL");
33         String password = rs.getString("PASSWORD");
34         Date dateofbirth = rs.getDate("DATEOFBIRTH");
35         Account acc = new Account(fname, lname, email, password, (java.sql.Date) dateofbirth);
36         accounts.add(acc);
37     }
38 } catch (Exception e) {
39     throw new RuntimeException(e);
40 }
41 return accounts;
42 }
43 // Thêm account
44 public void addAccount(Account acc) throws Exception {
45     String sql = "INSERT INTO accounts (FIRSTNAME, LASTNAME, EMAIL, PASSWORD, DATEOFBIRTH) " +
46         "VALUES (?, ?, ?, ?, ?)";
47     Connection conn = null;
48     PreparedStatement ps = null;
49     ResultSet rs = null;
50
51     try {
52         conn = datasource.getConnection();
53         ps = conn.prepareStatement(sql);
54         ps.setString(1, acc.getFirstname());
55         ps.setString(2, acc.getLastname());
56         ps.setString(3, acc.getEmail());
57         ps.setString(4, acc.getPassword());
58         ps.setDate(5, acc.getDateOfBirth());
59         ps.executeUpdate();
60     } catch (SQLException e) {
61         throw new RuntimeException(e);
62     }
63 }
64 }
65

```

*Bước 3: Tạo RegisterForm.jsp như hình (SV tự làm)*

```
<form action="${pageContext.request.contextPath}/registerform" method="post">
```

*Bước 4: Tạo AccountFormServlet.java*

#### AccountRegisterServlet.java

```

@WebServlet("/registerform")
17 public class AccountRegisterServlet extends HttpServlet {
18     private static final Long serialVersionUID = 1L;
19     private AccountUtil accountUtil;
20     @Resource(name = "jdbc/storedb")
21     private DataSource dataSource;
22
23     @Override
24     public void init(ServletConfig config) throws ServletException {
25         try {
26             accountUtil = new AccountUtil(dataSource);
27         } catch (Exception e) {
28             throw new RuntimeException(e);
29         }
30     }
31 }
32

```

```

33     @Override
34     protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
35         String firstname = req.getParameter("firstname");
36         String lastname = req.getParameter("lastname");
37         String email = req.getParameter("email");
38         String password = req.getParameter("password");
39
40         int day = Integer.parseInt(req.getParameter("day"));
41         int month = Integer.parseInt(req.getParameter("month"));
42         int year = Integer.parseInt(req.getParameter("year"));
43
44         LocalDate localDate = LocalDate.of(year, month, day);
45         java.sql.Date dob = java.sql.Date.valueOf(localDate);
46
47         Account account = new Account(firstname, lastname, email, password, (java.sql.Date)
dob);
48
49         try {
50             accountUtil.addAccount(account);
51             List<Account> accounts = accountUtil.getAccounts();
52             req.setAttribute("accounts", accounts);
53             RequestDispatcher rd = req.getRequestDispatcher("account.jsp");
54             rd.forward(req, resp);
55         } catch (Exception e) {
56             throw new RuntimeException(e);
57         }
58     }
59
60     @Override
61     protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
62     }
63 }

```

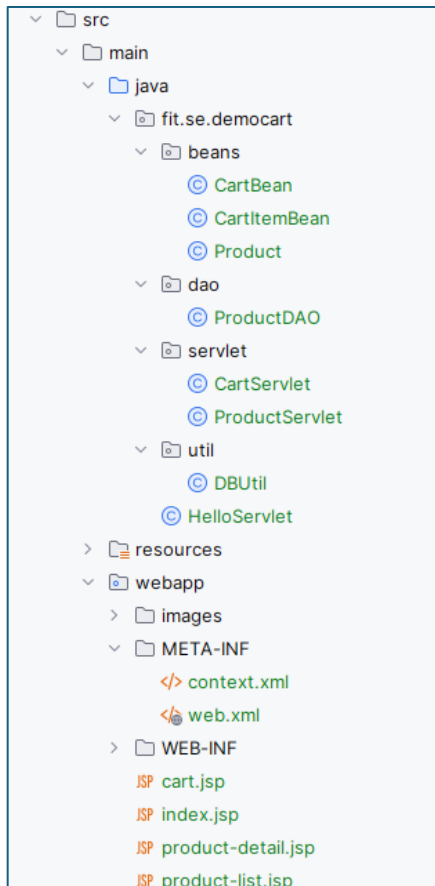
## Bài 3. JSPs – Thao tác với session

Thực hiện website mua bán sản phẩm đơn giản sử dụng:

- Session, *Standard Tag Library* và *Expression Language*
- Kiến trúc MVC, thực hiện theo kỹ thuật JSP, Servlet.
- Kết nối database mariadb thông qua DataSource, sử dụng cấu hình chuỗi kết nối connection pool trong file **webapp/META-INF/context.xml**(như bài 2)

Cấu trúc thư mục cho project minh họa như sau:

context.xml	
1	<Context>
2	<Resource
3	name="jdbc/shopdb"
4	auth="Container"
5	type="javax.sql.DataSource"
6	maxTotal="20"
7	maxIdle="10"
8	maxWaitMillis="10000"
9	driverClassName="org.mariadb.jdbc.Driver"
10	url="jdbc:mariadb://localhost:3306/shopdb"
11	username="root"
12	password="root"/>
13	</Context>



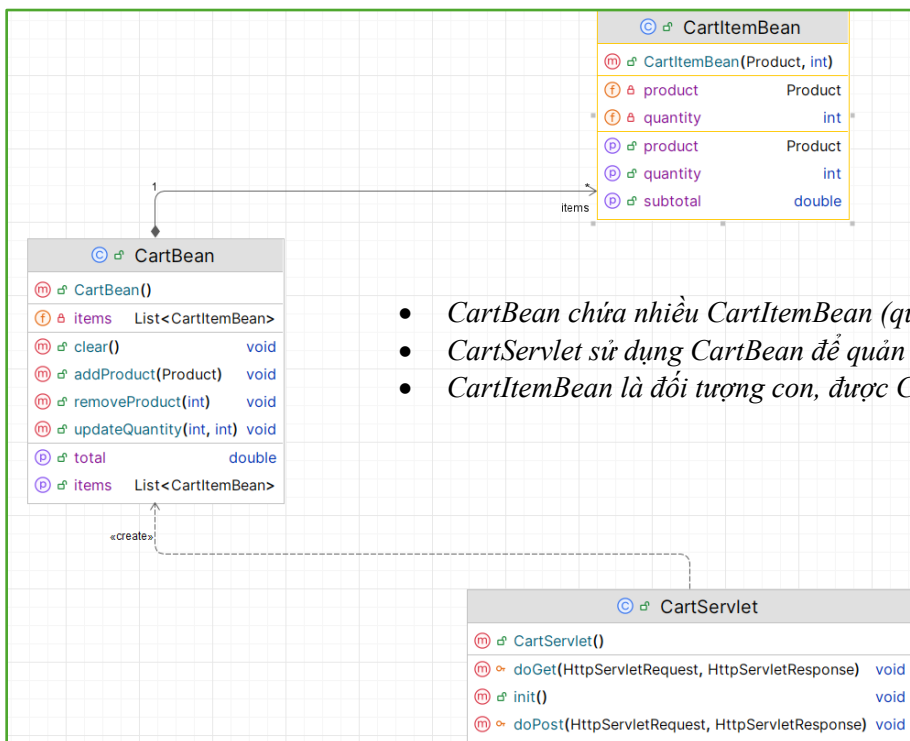
Database: *shopdb*

Name:

Comment:

Columns: + Add × Remove ▲ Up

#	Name	Datatype
1	ID	INT
2	MODEL	VARCHAR
3	DESCRIPTION	VARCHAR
4	QUANTITY	INT
5	PRICE	DOUBLE
6	IMGURL	LONGTEXT







- *CartBean* chứa nhiều *CartItemBean* (quan hệ 1 – n).
- *CartServlet* sử dụng *CartBean* để quản lý giỏ hàng.
- *CartItemBean* là đối tượng con, được *CartBean* quản lý.

Hiển thị danh sách sản phẩm và cho thêm sản phẩm vào giỏ hàng (Add To Cart), hoặc xem chi tiết giỏ hàng (Product Detail)



[View Cart](#)

Samsung	Sony	Iphone 15	Iphone 16
			
Price: 20000.0	Price: 23000.0	Price: 17000.0	Price: 18000.0
<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>
<button>Add To Cart</button>	<button>Add To Cart</button>	<button>Add To Cart</button>	<button>Add To Cart</button>
<a href="#">Product Detail</a>	<a href="#">Product Detail</a>	<a href="#">Product Detail</a>	<a href="#">Product Detail</a>

Khi giỏ hàng không có gì, hiển thị thông báo:

## Cart

Cart is empty!

[Tiếp tục mua](#)

Khi giỏ hàng có sản phẩm, cho phép xóa, cập nhật sản phẩm.

## Cart

Model	Quantity		Price	SubTotal	Action
Sony	<input type="text" value="1"/>	<button>Cập nhật</button>	23000.0	23000.0 VND	<button>Xóa</button>
Samsung	<input type="text" value="2"/>	<button>Cập nhật</button>	20000.0	40000.0 VND	<button>Xóa</button>
<b>Total:</b>				63000.0 VND	

Xóa hết giỏ hàng

[Tiếp tục mua](#)

Hướng dẫn:

Bước 1: Các Class – Beans trong hệ thống - **Model**

1. **Product:** sản phẩm

**Thuộc tính (Properties):**

- *id* : mã sản phẩm (duy nhất)
- *model*: mẫu sản phẩm
- *price* : giá sản phẩm

- *quantity*: số lượng sản phẩm
- *description*: mô tả sản phẩm

2. **CartItemBean**: Đại diện cho **một sản phẩm trong giỏ hàng** (sản phẩm được chọn mua trong danh sách các sản phẩm đang có trong table products)

**Thuộc tính (Properties):**

- *product* : sản phẩm được chọn mua
- *quantity*: số lượng

**Phương thức (Methods):**

- *getSubtotal()*: tính thành tiền cho 1 sản phẩm chọn mua

**CartItemBean.java**

```
public class CartItemBean implements Serializable {
7     private Product product;
8     private int quantity;
9
10    public CartItemBean(Product product, int quantity) {
11        this.product = product;
12        this.quantity = quantity;
13    }
14    //Getter & Setter
15    .....
33    public double getSubtotal() {
34        return product.getPrice() * quantity;
35    }
```

3. **CartBean**: Đại diện cho **giỏ hàng** gồm các *CartItemBean*

**Thuộc tính (Properties):**

- *items* : danh sách các *CartItemBean* (các sản phẩm đã chọn mua)

**Phương thức (Methods):**

- *addProduct*(Product p) → thêm sản phẩm vào giỏ
- *updateQuantity*(int productId, int quantity) → cập nhật số lượng
- *removeProduct*(int productId) → xóa sản phẩm đã mua trong giỏ hàng
- *clear*() → Xóa hết giỏ hàng
- *getTotal*() → tính tổng tiền đơn hàng

**CartBean.java**

```
public class CartBean {
9     private List<CartItemBean> items;
10
11    public CartBean() {
12        items = new ArrayList<>();
13    }
14
15    public List<CartItemBean> getItems() {
16        return items;
17    }
18
19    // thêm sản phẩm
20    public void addProduct(Product p) {
22        for (CartItemBean item : items) {
23            if (item.getProduct().getId() == p.getId()) {
```

```

24         item.setQuantity(item.getQuantity() + 1);
25         return;
26     }
27 }
28 items.add(new CartItemBean(p, 1));
29 }
30
31 // xóa sản phẩm
32 public void removeProduct(int productId) {
33     items.removeIf(item -> item.getProduct().getId() == productId);
34 }
35
36 // cập nhật số lượng
37 public void updateQuantity(int productId, int quantity) {
38     for (CartItemBean item : items) {
39         if (item.getProduct().getId() == productId) {
40             if (quantity > 0) {
41                 item.setQuantity(quantity);
42             } else {
43                 // nếu nhập <= 0 thì xóa luôn sản phẩm
44                 removeProduct(productId);
45             }
46             return;
47         }
48     }
49 }
50
51 // tính tổng tiền
52 public double getTotal() {
53     double total = 0;
54     for (CartItemBean item : items) {
55         total += item.getSubtotal();
56     }
57     return total;
58 }
59
60 // xóa hết giỏ hàng
61 public void clear() {
62     items.clear();
63 }
64 }

```

## Bước 2: Class DBUtil (Connection)

Thực hiện tạo kết nối thông qua đối tượng DataSource

### DBUtil.java

```

public class DBUtil {
7
8     private DataSource dataSource;
9     public DBUtil(DataSource dataSource) {
10         this.dataSource = dataSource;
11     }
12
13     public Connection getConnection() {
14         Connection conn;
15         try {
16             conn= dataSource.getConnection();

```

```

17         } catch (SQLException e) {
18             throw new RuntimeException(e);
19         }
20         return conn;
21     }
22 }

```

### Bước 3: Các Class DAO (Query Database)

Sử dụng class DBUtil thực hiện việc kết nối database và thực thi các câu truy vấn trực tiếp từ Database

1. **ProductDAO**: thực hiện các phương thức trực tiếp truy vấn table Product như select, insert, update, delete

#### **Phương thức (Methods):**

- *getAllProducts()* → thực hiện câu truy vấn select tất cả các sản phẩm trong table products

```

// READ ALL
public List<Product> getAllProducts() {
    List<Product> list = new ArrayList<>();
    String sql = "SELECT * FROM products";

```

- *getProductById(int id)* → thực hiện câu truy vấn lấy 1 sản phẩm theo mã id

```

// READ BY ID
public Product getProductById(int id) {
    String sql = "SELECT * FROM products WHERE ID=?";

```

#### ProductDAO.java

```

public class ProductDAO {
9     private DBUtil dbUtil;
10
11     public ProductDAO(DataSource dataSource) {
12         dbUtil= new DBUtil(dataSource);
13     }
14
15     // READ ALL
16     public List<Product> getAllProducts() {
17         List<Product> list = new ArrayList<>();
18         String sql = "SELECT * FROM products";
19         try (Connection conn = dbUtil.getConnection();
20             Statement stmt = conn.createStatement();
21             ResultSet rs = stmt.executeQuery(sql)) {
22             while (rs.next()) {
23                 Integer id = rs.getInt("ID");
24                 String model = rs.getString("MODEL");
25                 Double price = rs.getDouble("PRICE");
26                 Integer quantity = rs.getInt("QUANTITY");
27                 String image = rs.getString("IMGURL");
28                 String description = rs.getString("DESCRIPTION");
29                 Product p = new Product(id,model,description,quantity,price,image);
30                 list.add(p);
31             }
32         } catch (SQLException e) {
33             e.printStackTrace();
34         }
35         return list;
36     }

```

```

55     }
56
57     // READ BY ID
58     public Product getProductById(int id) {
59         String sql = "SELECT * FROM products WHERE ID=?";
60         try (Connection conn = dbUtil.getConnection();
61             PreparedStatement ps = conn.prepareStatement(sql)) {
62             ps.setInt(1, id);
63             try (ResultSet rs = ps.executeQuery()) {
64                 if (rs.next()) {
65                     Integer proid = rs.getInt("ID");
66                     String model = rs.getString("MODEL");
67                     Double price = rs.getDouble("PRICE");
68                     Integer quantity = rs.getInt("QUANTITY");
69                     String image = rs.getString("IMGURL");
70                     String description = rs.getString("DESCRIPTION");
71
72                     Product p = new Product(proid,model,description,quantity,price,image);
73                     return p;
74                 }
75             }
76         } catch (Exception e) {
77             e.printStackTrace();
78         }
79         return null;
80     }
81 }

```

#### Bước 4: Các Servlet xử lý - **Controller**:

1. **ProductServlet: (WebServlet):** Sử dụng class ProductDAO xử lý chức năng lấy tất cả thông tin sản phẩm, xem chi tiết 1 sản phẩm theo mã sản phẩm. (doGet: lấy thông tin, hiển thị danh sách)

#### ProductServlet.java

```

@WebServlet("/{products", "/product"})
17 public class ProductServlet extends HttpServlet {
18     private ProductDAO productDAO;
19     @Resource(name="jdbc/shopdb")
20     private DataSource dataSource;
21
22     @Override
23     public void init() {
24         productDAO = new ProductDAO(dataSource);
25     }
26
27     @Override
28     protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
29         String idstr = req.getParameter("id");
30
31         if(idstr!=null)
32         {
33             int id = Integer.parseInt(idstr);
34             Product product = productDAO.getProductById(id);
35             if(product!=null)
36             {
37                 req.setAttribute("product", product);
38                 RequestDispatcher dispatcher = getServletContext().getRequestDispatcher("/product-
detail.jsp");

```

```

42         dispatcher.forward(req, resp);
43     }else {
44         resp.sendError(HttpServletResponse.SC_NOT_FOUND, "Product not found");
45         return;
46     }
47 }
48 List<Product> products = productDAO.getAllProducts();
49 req.setAttribute("products", products);
50 RequestDispatcher dispatcher = getServletContext().getRequestDispatcher("/product-
list.jsp");
51 dispatcher.forward(req, resp);
52
53 }
54
55 @Override
56     protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
57     //     super.doPost(req, resp);
58
59 }
60 }
61

```

## 2. *CartServlet*: Xử lý chức năng cập nhật, xóa giỏ hàng. (doPost: cập nhật, xóa)

### CartServlet.java

```

@WebServlet("/cart")
17 public class CartServlet extends HttpServlet {
18
19     private ProductDAO productDAO;
20
21     @Resource(name="jdbc/shopdb")
22     private DataSource dataSource;
23
24     @Override
25     public void init() throws ServletException {
26         try {
27             productDAO = new ProductDAO(dataSource);
28         } catch (Exception e) {
29             throw new ServletException(e);
30         }
31     }
32     @Override
33     protected void doGet(HttpServletRequest req, HttpServletResponse resp)
34         throws ServletException, IOException {
35         req.getRequestDispatcher("cart.jsp").forward(req, resp);
36     }
37     @Override
38     protected void doPost(HttpServletRequest req, HttpServletResponse resp)
39         throws ServletException, IOException {
40         HttpSession session = req.getSession();
41         CartBean cart = (CartBean) session.getAttribute("cart");
42         if (cart == null) {
43             cart = new CartBean();
44             session.setAttribute("cart", cart);
45         }
46
47         String action = req.getParameter("action");
48

```

```

49     try {
50         if ("add".equals(action)) {
51             int id = Integer.parseInt(req.getParameter("id"));
52             Product p = productDAO.getProductById(id);
53             cart.addProduct(p);
54         } else if ("update".equals(action)) {
55             int id = Integer.parseInt(req.getParameter("productId"));
56             int quantity = Integer.parseInt(req.getParameter("quantity"));
57             cart.updateQuantity(id, quantity);
58         } else if ("remove".equals(action)) {
59             int id = Integer.parseInt(req.getParameter("productId"));
60             cart.removeProduct(id);
61         } else if ("clear".equals(action)) {
62             cart.clear();
63         }
64     } catch (Exception e) {
65         throw new ServletException(e);
66     }
67
68     resp.sendRedirect("cart");
69 }
72 }

```

## Bước 5: Các trang jsp – View

1. *product-list.jsp* → Giao diện hiển thị danh sách các sản phẩm.

### product-list.jsp

```

<body>
32 <p>
33     <a href="cart">View Cart</a>
34 </p>
35 <c:forEach items="${products}" var="p">
36     <div class="product-class">
37         <b> ${p.model}</b>
38         <br/>
39          <br/>
40         Price: ${p.price}<br/>
41         <form action="${pageContext.request.contextPath}/cart" method="post">
42             <input type="text" size="2" value="1" name="quantity"> <br/>
43             <input type="hidden" name="id" value="${p.id}">
44             <input type="hidden" name="price" value="${p.price}">
45             <input type="hidden" name="model" value="${p.model}">
46             <input type="hidden" name="action" value="add"><br/>
47             <input type="submit" name="addToCart" value="Add To Cart"><br/>
48         </form>
49         <a href="${pageContext.request.contextPath}/product?id=${p.id}">Product Detail</a><br/>
50     </div>
51 </c:forEach>
52 </body>

```

2. *cart.jsp* → Giao diện hiển thị trang giỏ hàng

### cart.jsp

```

<body>
22 <div class="container">
23     <h2>Cart</h2>

```

```

24     <c:if test="${empty cart.items}">
25         <p>Cart is empty!</p>
26     </c:if>
27
28     <c:if test="${not empty cart.items}">
29         <table class="table table-border">
30             <tr>
31                 <th>Model</th>
32                 <th>Quantity</th>
33                 <th>Price</th>
34                 <th>Total</th>
35                 <th>Actions</th>
36             </tr>
37             <c:forEach var="item" items="${cart.items}">
38                 <tr>
39                     <td>${item.product.model}</td>
40                     <td>
41                         <form action="${pageContext.request.contextPath}/cart" method="post"
42                             style="display:inline;">
43                             <input type="hidden" name="action" value="update"/>
44                             <input type="hidden" name="productId" value="${item.product.id}"/>
45                             <input type="number" name="quantity" value="${item.quantity}"
46                                 min="1"/>
47                             <input type="submit" value="Update"/>
48                         </form>
49                     </td>
50                     <td>${item.product.price}</td>
51                     <td>${item.product.price * item.quantity}</td>
52                     <td>
53                         <form action="${pageContext.request.contextPath}/cart" method="post"
54                             style="display:inline;">
55                             <input type="hidden" name="action" value="remove"/>
56                             <input type="hidden" name="productId" value="${item.product.id}"/>
57                             <input type="submit" value="Remove"/>
58                         </form>
59                     </td>
60                 </tr>
61             </c:forEach>
62             </table>
63             <p><strong>Total: </strong> ${cart.total}</p>
64         </c:if>
65
66         <a href="product">Continue Shopping</a>
67     </div>
68 </body>

```

### 1. product-detail.jsp → Xem chi tiết sản phẩm theo mã

#### product-detail.jsp

```

<body>
15 <h2>Product Detail</h2>
16 <c:if test="${not empty product}">
17     <ul>
18         <li>Id: ${product.id}</li>
19         <li>Model: ${product.model}</li>
20         <li>Description: ${product.description}</li>
21         <li>Quantity: ${product.quantity}</li>
22         <li>Price: ${product.price}</li>

```



```

23
24         
26     <br/>
27 </ul>
28 </c:if>
29
30 <p>
31     <a href="${pageContext.request.contextPath}/product">Back to Product List</a>
32 </p>
33 </body>

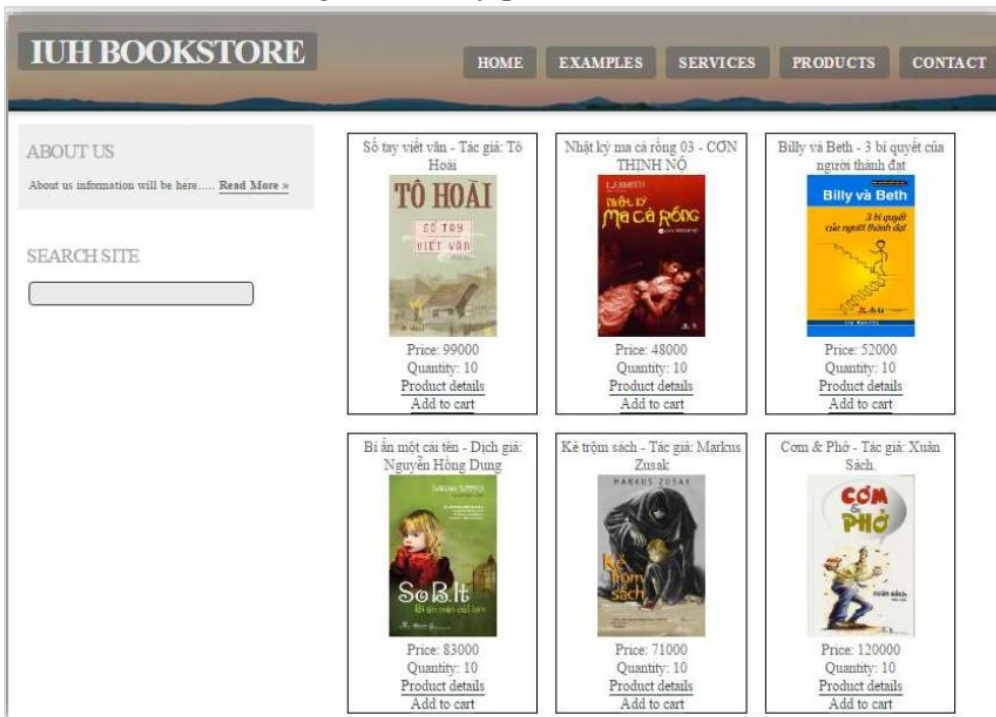
```

## Bài 4. JSPs – Thao tác với session

Thực hiện website quản lý nhà sách (mua bán, tìm kiếm sách,...) sử dụng:

- Session, *Standard Tag Library* và *Expression Language*
- Kiến trúc MVC, thực hiện theo kỹ thuật JSP, Servlet.
- Kết nối database mariadb thông qua DataSource, sử dụng cấu hình chuỗi kết nối connection pool trong file **webapp/META-INF/context.xml**(như bài 2,3).
- Sử dụng CSS, Bootstrap thiết kế trang như hình minh họa.

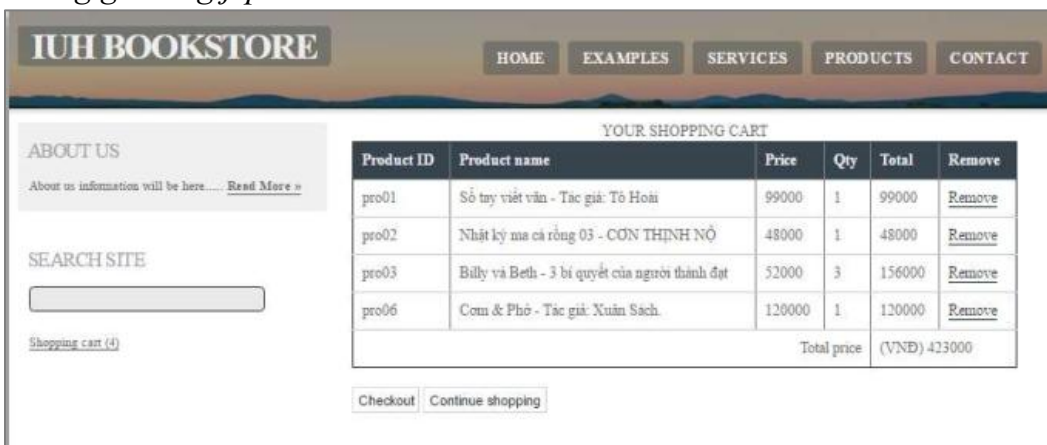
Thực hiện các trang *danhsach.jsp*



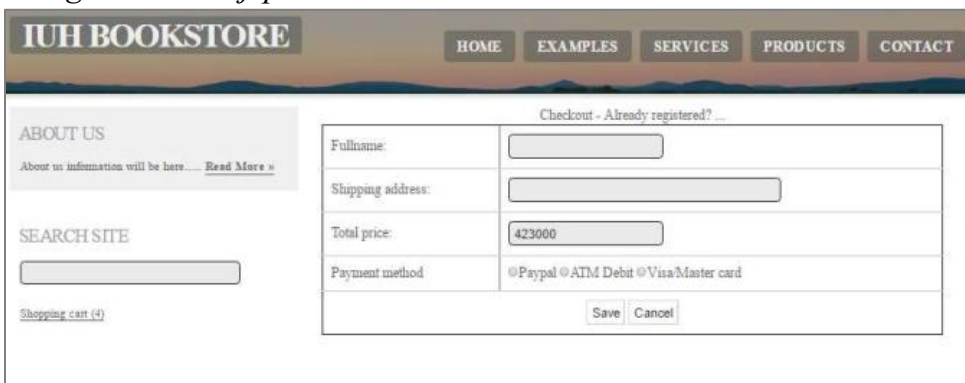
Trang *chitietsach.jsp*



Trang giohang.jsp

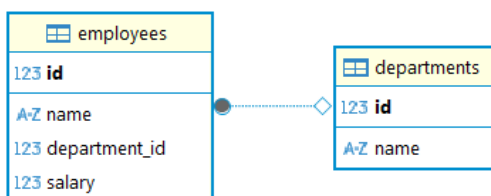


Trang thanhtoan.jsp



## Bài 5: Bài tập tổng hợp 1

Thực hiện website quản lý nhân viên được mô tả như sau:




Website thực hiện các chức năng:

- Hiển thị danh sách phòng ban, CRUD phòng ban
- Hiển thị danh sách nhân viên theo từng phòng ban, CRUD nhân viên

Kỹ thuật thực hiện

- *Standard Tag Library* và *Expression Language*
- Kiến trúc MVC, thực hiện theo kỹ thuật JSP, Servlet.
- Kết nối database mariadb thông qua DataSource, sử dụng cấu hình chuỗi kết nối connection pool trong file **webapp/META-INF/context.xml**(như bài 2,3,4).
- Sử dụng CSS, Bootstrap thiết kế trang như hình minh họa

Trang hiển thị danh sách phòng ban và chức năng CRUD phòng ban:




## Departments List

[Add Department](#)

Tìm phòng ban:

DEPT ID	Name Department	Action
1	Giám Đốc	<a href="#">Edit</a>   <a href="#">Delete</a>   <a href="#">Employees</a>
2	Kế Toán	<a href="#">Edit</a>   <a href="#">Delete</a>   <a href="#">Employees</a>
3	Phát triển	<a href="#">Edit</a>   <a href="#">Delete</a>   <a href="#">Employees</a>
4	Kế hoạch Vật tư	<a href="#">Edit</a>   <a href="#">Delete</a>   <a href="#">Employees</a>
7	Phòng Quản Trị	<a href="#">Edit</a>   <a href="#">Delete</a>   <a href="#">Employees</a>

Trang hiển thị tìm kiếm phòng ban: Tìm kiếm tương đối theo tên phòng ban (LIKE)



## Departments List

[Add Department](#)

Tìm phòng ban:

DEPT ID	Name Department	Action
2	Kế Toán	<a href="#">Edit</a>   <a href="#">Delete</a>   <a href="#">Employees</a>
4	Kế hoạch Vật tư	<a href="#">Edit</a>   <a href="#">Delete</a>   <a href="#">Employees</a>

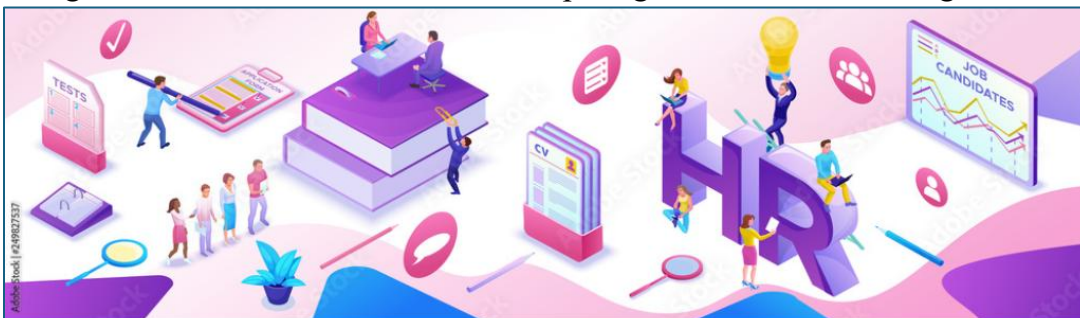
Trang thêm phòng ban mới:



## Department Information

Name:

Trang hiển thị danh sách nhân viên theo phòng ban và các chức năng CRUD



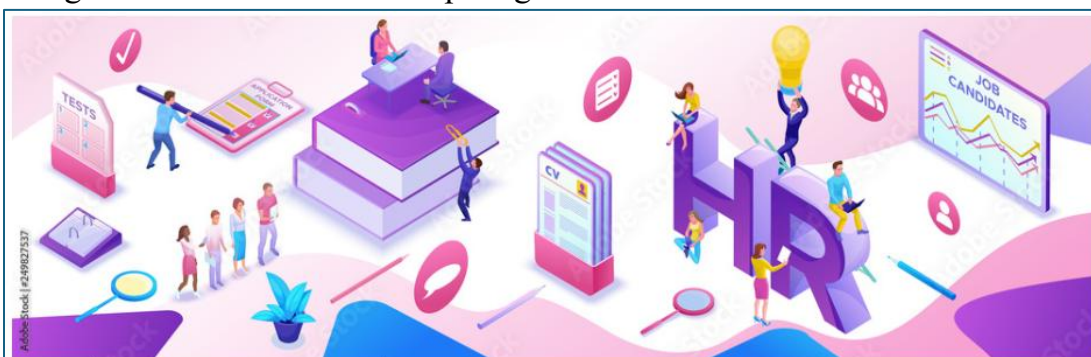
## Employees List

[Add Employee](#)

ID	Name Employee	Salary	Dept	Action
6	Admin	1212.0	2	<a href="#">Edit</a>   <a href="#">Delete</a>

[Department](#)

Trang thêm nhân viên mới theo phòng ban:



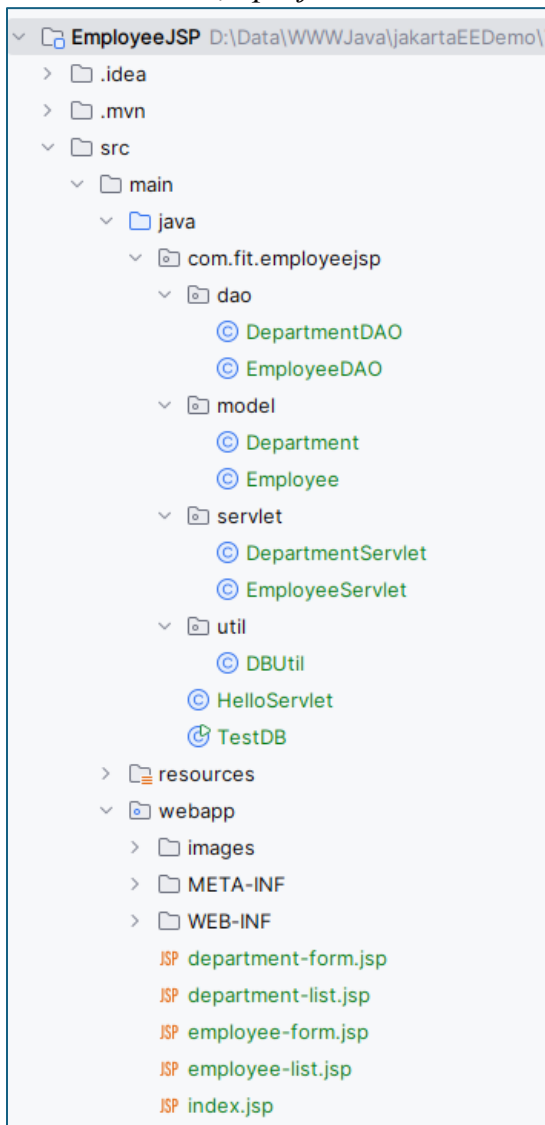
Name:

Salary:

Department:



Cấu trúc thư mục project theo kiến trúc như sau:



Bước 0: Add dependencies tương ứng (bài 2,3,4)

Bước 1: (Model) Tạo các **Model** tương ứng với database

Bước 2: (Connection) Tạo **DBUtil** kết nối connection

Tạo file webapp\META-INF\context.xml thiết lập connection pool. Trong DBUtil sử dụng DataSource lấy chuỗi kết nối

Bước 3: (Connnection – Query data) Tạo các class **DepartmentDAO**, chứa các phương thức thực hiện các **EmployeeDAO** câu truy vấn đến database và trả về kết quả như yêu cầu.

Bước 4: (Controller) Tạo các class **DepartmentServlet**, **EmployeeServlet** thực hiện các phương thức doGet, doPost từ client xử lý

Bước 5: (View) Tạo các view là các file **employee-list.jsp**, **employee-form.jsp**, **department-list.jsp**, **department-form.jsp** tương ứng cho các request từ client.

### EmployeeDAO.java

```
1 package com.fit.employeejsp.dao;
2
3 import com.fit.employeejsp.model.Employee;
4 import com.fit.employeejsp.util.DBUtil;
5
6 import javax.sql.DataSource;
7 import java.sql.Connection;
8 import java.sql.PreparedStatement;
9 import java.sql.ResultSet;
10 import java.util.ArrayList;
11 import java.util.List;
12
13 public class EmployeeDAO {
14     private DBUtil dbutil;
15
16     public EmployeeDAO(DataSource dataSource) {
17         dbutil = new DBUtil(dataSource);
18     }
19 }
```

```

20 public List<Employee> getAllEmployees() {
21     List<Employee> emplist = new ArrayList<>();
22     String sql = "select * from employees";
23     try {
24         Connection con = dbutil.getConnection();
25         PreparedStatement ps = con.prepareStatement(sql);
26         ResultSet rs = ps.executeQuery();
27         while (rs.next()) {
28             Employee emp = new Employee();
29             emp.setId(rs.getInt("id"));
30             emp.setName(rs.getString("name"));
31             emp.setSalary(rs.getDouble("salary"));
32             emp.setDepartmentId(rs.getInt("department_id"));
33
34             emplist.add(emp);
35
36         }
37     } catch (Exception e) {
38         throw new RuntimeException(e);
39     }
40     return emplist;
41 }
42
43
44
45 public List<Employee> getAllByDepartment(int deptId) {
46     List<Employee> list = new ArrayList<>();
47     String sql = "SELECT * FROM employees WHERE department_id=?";
48     try (Connection conn = dbutil.getConnection();
49         PreparedStatement ps = conn.prepareStatement(sql)) {
50         ps.setInt(1, deptId);
51         try (ResultSet rs = ps.executeQuery()) {
52             while (rs.next()) {
53                 list.add(new Employee(
54                     rs.getInt("id"),
55                     rs.getString("name"),
56                     rs.getInt("department_id"),
57                     rs.getDouble("salary")
58                 ));
59             }
60         }
61     } catch (Exception e) {
62         e.printStackTrace();
63     }
64     return list;
65 }
66
67 public void save(Employee emp) {
68     String sql = "INSERT INTO employees(name, salary, department_id) VALUES (?, ?, ?)";
69     try (Connection conn = dbutil.getConnection();
70         PreparedStatement ps = conn.prepareStatement(sql)) {
71         ps.setString(1, emp.getName());
72         ps.setDouble(2, emp.getSalary());
73         ps.setInt(3, emp.getDepartmentId());
74         ps.executeUpdate();
75     } catch (Exception e) {
76         e.printStackTrace();
77     }

```

```

78     }
79
80     public void update(Employee emp) {
81         String sql = "UPDATE employees SET name=?, salary=?, department_id=? WHERE id=?";
82         try (Connection conn = dbutil.getConnection();
83             PreparedStatement ps = conn.prepareStatement(sql)) {
84             ps.setString(1, emp.getName());
85             ps.setDouble(2, emp.getSalary());
86             ps.setInt(3, emp.getDepartmentId());
87             ps.setInt(4, emp.getId());
88             ps.executeUpdate();
89         } catch (Exception e) {
90             e.printStackTrace();
91         }
92     }
93
94     public void delete(int id) {
95         String sql = "DELETE FROM employees WHERE id=?";
96         try (Connection conn = dbutil.getConnection();
97             PreparedStatement ps = conn.prepareStatement(sql)) {
98             ps.setInt(1, id);
99             ps.executeUpdate();
100        } catch (Exception e) {
101            e.printStackTrace();
102        }
103    }
104 }
105

```

## EmployeeServlet.java

```

1  package com.fit.employeejsp.servlet;
2
3  import com.fit.employeejsp.dao.DepartmentDAO;
4  import com.fit.employeejsp.dao.EmployeeDAO;
5  import com.fit.employeejsp.model.Employee;
6  import jakarta.annotation.Resource;
7  import jakarta.servlet.ServletConfig;
8  import jakarta.servlet.ServletException;
9  import jakarta.servlet.annotation.WebServlet;
10 import jakarta.servlet.http.HttpServlet;
11 import jakarta.servlet.http.HttpServletRequest;
12 import jakarta.servlet.http.HttpServletResponse;
13
14 import javax.sql.DataSource;
15 import java.io.IOException;
16 import java.util.List;
17
18 @WebServlet("/employees")
19 public class EmployeeServlet extends HttpServlet {
20
21     @Resource(name = "jdbc/hrdb")
22     private DataSource dataSource;
23
24     private EmployeeDAO empDao;
25     private DepartmentDAO deptDao;
26

```

```

27     @Override
28     public void init(ServletConfig servletConfig) throws ServletException {
29         try {
30             empDao = new EmployeeDAO(dataSource);
31             deptDao = new DepartmentDAO(dataSource);
32         } catch (Exception e) {
33             throw new RuntimeException(e);
34         }
35     }
36 }
37
38
39     @Override
40     protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
41         String action = req.getParameter("action");
42         if (action == null) action = "list";
43
44         switch (action) {
45             case "list":
46                 // Load toàn bộ employees (không quan tâm deptId)
47                 List<Employee> allEmployees = empDao.getAllEmployees();
48                 req.setAttribute("employees", allEmployees);
49                 req.getRequestDispatcher("employee-list.jsp").forward(req, resp);
50                 break;
51
52             case "new":
53                 req.setAttribute("departments", deptDao.getAll());
54                 req.getRequestDispatcher("employee-form.jsp").forward(req, resp);
55                 break;
56             case "edit":
57                 int id = Integer.parseInt(req.getParameter("id"));
58                 // TODO: thêm method getById cho EmployeeDAO nếu muốn edit cụ thể
59                 break;
60             case "delete":
61                 empDao.delete(Integer.parseInt(req.getParameter("id")));
62                 resp.sendRedirect("employees");
63                 break;
64             case "viewbyid": // list all employees (or by department)
65                 String deptId = req.getParameter("deptId");
66                 List<Employee> list;
67
68                 if (deptId != null) {
69                     list = empDao.getAllByDepartment(Integer.parseInt(deptId));
70                 } else {
71                     list = empDao.getAllByDepartment(1); // mặc định dept 1
72                 }
73                 req.setAttribute("employees", list);
74                 req.setAttribute("departments", deptDao.getAll());
75                 req.getRequestDispatcher("employee-list.jsp").forward(req, resp);
76             }
77         }
78
79     @Override
80     protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
81
82         String idParam = req.getParameter("id"); // lấy tham số "id" từ request

```



```

83         int id;
84
85         if (idParam != null && !idParam.isEmpty()) {
86             id = Integer.parseInt(idParam); // có giá trị thì parse sang int
87         } else {
88             id = 0; // nếu null hoặc rỗng thì gán mặc định = 0
89         }
90
91         String name = req.getParameter("name");
92         double salary = Double.parseDouble(req.getParameter("salary"));
93         int deptId = Integer.parseInt(req.getParameter("departmentId"));
94
95         Employee emp = new Employee(id, name, deptId, salary);
96         if (id > 0) {
97             empDao.update(emp);
98         } else {
99             empDao.save(emp);
100        }
101        resp.sendRedirect("employees?deptId=" + deptId);
102    }
103 }
104

```

### employee-list.jsp

```

1  <%@ page import="com.fit.employeejsp.model.Employee" %>
2  <%@ page import="java.util.List" %><!--
3      Created by IntelliJ IDEA.
4      User: ThinkPad
5      Date: 19/08/2025
6      Time: 12:25 AM
7      To change this template use File | Settings | File Templates.
8  --%>
9
10 <%@ page contentType="text/html; charset=UTF-8" language="java" %>
11 <%@ taglib prefix="c" uri="jakarta.tags.core" %>
12 <html>
13 <head><title>Employees</title>
14     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.8/dist/css/bootstrap.min.css"
15     rel="stylesheet">
16 </head>
17 <body>
18 <div class="container">
19     
21     <h2>Employees List</h2>
22     <a href="{pageContext.request.contextPath}/employees?action=new">Add Employee</a>
23     <br/>
24     <table class="table table-primary">
25         <tr>
26             <th>ID</th>
27             <th>Name Employee</th>
28             <th>Salary</th>
29             <th>Dept</th>
30             <th>Action</th>
31         </tr>
32         <c:forEach var="emp" items="{employees}">
33             <tr>
34                 <td>{emp.id}</td>

```

```

34         <td>${emp.name}</td>
35         <td>${emp.salary}</td>
36         <td>${emp.departmentId}</td>
37         <td>
38         <a href="${pageContext.request.contextPath}/employees?action=edit&id=${emp.id}">Edit</a> |
39 <a href="${pageContext.request.contextPath}/employees?action=delete&id=${emp.id}">Delete</a>
40         </td>
41     </tr>
42 </tr>
43 </c:forEach>
44 </table>
45 <a href="${pageContext.request.contextPath}/departments"> Department</a>
46 </div>
47 </body>
48 </html>
49

```

## employee-form.jsp

```

1  <%@ taglib prefix="c" uri="jakarta.tags.core" %>
2  <%@ page import="com.fit.employeejsp.model.Department" %>
3  <%@ page import="java.util.List" %><!--
4  Created by IntelliJ IDEA.
5  User: ThinkPad
6  Date: 19/08/2025
7  Time: 12:25 AM
8  To change this template use File | Settings | File Templates.
9  --%>
10 <%@ page contentType="text/html; charset=UTF-8" language="java" %>
11 <html>
12 <head><title>Employee Information</title></head>
13 <body>
14 <div class="container">
15     
16
17     <form action="${pageContext.request.contextPath}/employees" method="post">
18         <input type="hidden" name="id"/>
19         Name: <input type="text" name="name"/><br/>
20         Salary: <input type="text" name="salary"/><br/>
21         Department:
22         <select name="departmentId">
23             <c:forEach var="dep" items="${departments}">
24                 <option value="${dep.department_id}">${dep.department_name}</option>
25             </c:forEach>
26         </select>
27
28         <br/>
29         <input type="submit" value="Save"/>
30     </form>
31 </div>
32 </body>
33 </html>

```

## Bài 6. JSPs - Bài tập tổng hợp 2

Xây dựng ứng dụng quản lý tin tức cho một website trực tuyến, các trang Web cần thực hiện các chức năng: *xem tin tức theo từng danh mục, thêm mới tin tức và xóa tin tức không cần thiết.*

Một phần của cơ sở dữ liệu được mô tả như sau:

DANHMUC(**MADM**, TENDANHMUC, NGUOIQUANLY, GHICHU)

TINTUC(**MATT**, TIEUDE, NOIDUNGTT, LIENKET, **MADM**)

Xây dựng ứng dụng dùng Servlet/JSPs kết hợp với Mariadb/SQL Server và Web server Tomcat thực hiện các yêu cầu với mô hình MVC (Các Servlet đóng vai trò Controller):

- Tạo cơ sở dữ liệu với các quan hệ như mô tả trên và nhập dữ liệu cần thiết để kiểm tra chương trình. Cơ sở dữ liệu trong lưu tên: QUANLYDANHMUC

- Dùng HTML/CSS tạo Layout chung cho các *trang Web quản lý tin tức trực tuyến*
- Tạo lớp mô tả thông tin *TinTuc.java* (bao gồm các get/set cho các thuộc tính và các constructors) dùng để thao tác với tin tức trực tuyến.

- Tạo lớp *DanhSachTinTucQuanLy.java* (bao gồm các phương thức thao tác với CSDL: lấy tin tức theo danh mục tin tức, thêm mới tin tức và xóa tin tức).

- Tạo giao diện Web cho phép thực hiện thao tác liệt kê dữ liệu dùng JSPs:  
(*DanhSachTinTucServlet.java - DanhSachTinTuc.jsp*) Liệt kê dữ liệu danh sách tin tức theo từng danh mục.

- Tạo giao diện Web cho phép thực hiện giao diện thao tác với việc thêm dữ liệu
  - o Tạo Form cho phép thêm nội dung tin tức với đầy đủ thông tin phù hợp với CSDL  
(*TinTucFormServlet.java, TinTucForm.jsp*)

- o Kiểm tra dữ liệu nhập phía Client trên Form:

- Mã TT, Tiêu đề, Liên kết, Nội dung, Thông tin danh mục của tin là bắt buộc nhập.
    - Liên kết bắt đầu bởi “http://” (dùng RegularExpression).
    - Nội dung không quá 255 ký tự (dùng RegularExpression).
    - Thông tin nhập vào hợp lệ sau khi nhấn nút “Thêm” sẽ được thêm vào cơ sở dữ liệu và hiển thị dữ liệu trên màn hình (*KetQua.jsp hoặc trả về DanhSachTinTuc.jsp*).

- ▪ Tạo giao diện Web cho phép thực hiện chức năng quản lý: thao tác hủy dữ liệu. Khi chọn chức năng này, hiển thị danh sách và kèm theo chức năng hủy tin tức ra khỏi cơ sở dữ liệu  
(*QuanLyFormServlet.java, QuanLyForm.jsp*).