

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>
<u>Bài 1.</u> Phân tích yêu cầu của Bài tập lớn	4
<u>Bài 2.</u> Layout Bài tập lớn - Bài tập cá nhân	6
<u>Bài 3.</u> Cài đặt 7.0 Server	7
<u>Bài 4.</u> Java Servlet - Thao tác với doGet(), doPost()	10
<u>Bài 5.</u> Java Servlet - Filter	12
<u>Bài 6.</u> Java Servlet - Upload files.....	14
<u>Bài 7.</u> Java Servlet - Upload hình, lưu CSDL	16
<u>Bài 8:</u> Java Servlet - JavaMail API	19
 BÀI TẬP TUẦN 03-04 MÔN LẬP TRÌNH WWW JAVA.....	 23
<i>Chương 3: Jakarta Server Pages - JSPs.....</i>	<i>23</i>
<u>Bài 1.</u> JSPs - Thao tác với Form	26
<u>Bài 2.</u> JSPs - Model View Controller	28

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

1. IDE IntelliJ Ultimate
2. Glassfish 7.0
3. JDK 21



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

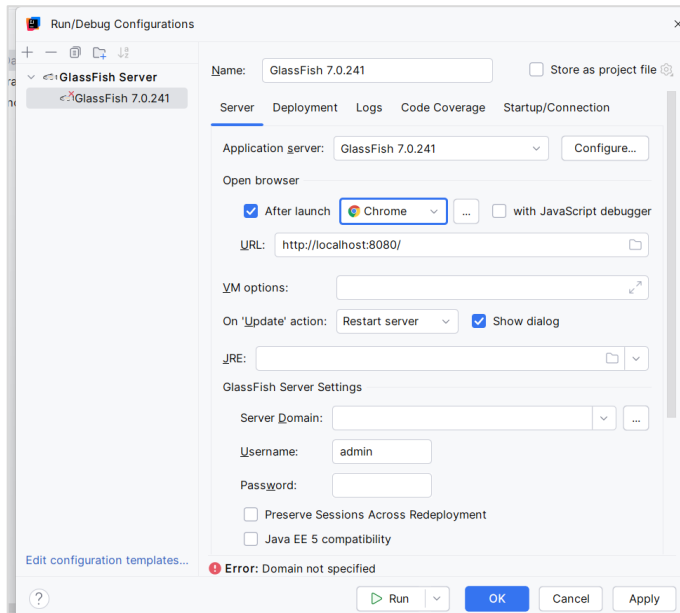
1. Cấu hình Application Server

- 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

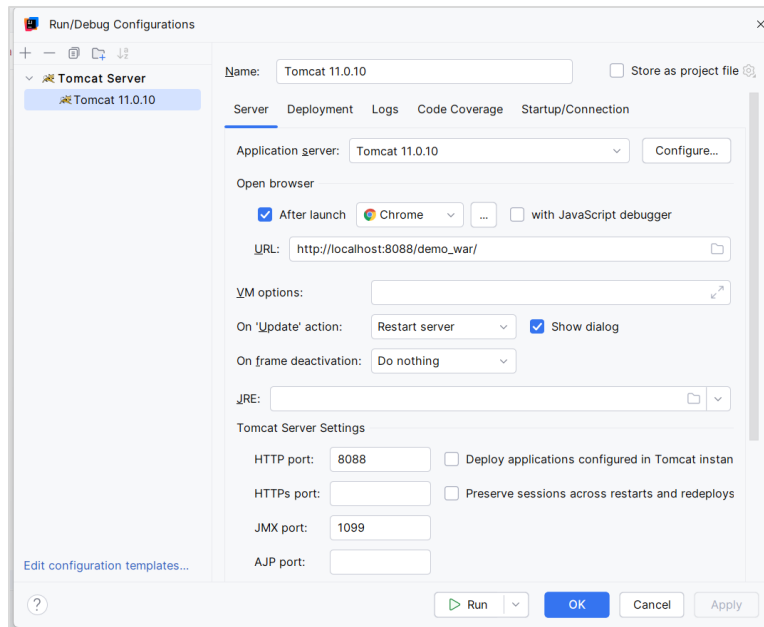
1. Mở IntelliJ → Vào menu File → Settings (hoặc Ctrl+Alt+S).
2. Chọn mục Build, Execution, Deployment → Application Servers.
3. Bấm dấu + → chọn Tomcat Server → Local.

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

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

```

@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");
        }
    }
}

```

- *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");
}

}
```

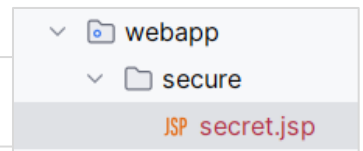
- home.jsp cho phép đăng nhập bị chặn theo AuthFilter.java

```
<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.

Upload multi-files

File #1:	<input type="button" value="Choose File"/>	No file chosen
File #2:	<input type="button" value="Choose File"/>	No file chosen
File #3:	<input type="button" value="Choose File"/>	No file chosen
File #4:	<input type="button" value="Choose File"/>	No file chosen
File #5:	<input type="button" value="Choose File"/>	No file chosen

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 Binary trong CSDL SQL Server.

File Upload to Database (multipart/form-data)

First Name:

Last Name:

Portrait Photo: No file chosen

Hướng dẫn: (Sinh viên cần tách các lớp thao tác với CSDL, không dùng chung)

▪ CSDL: (SQL Server)

```
CREATE DATABASE UploadFileServletDB; CREATE
TABLE contacts (
    contact_id int NOT NULL identity(1,1),
    first_name nvarchar(45) DEFAULT NULL,
    last_name nvarchar(45) DEFAULT NULL,
    photo image,
    PRIMARY KEY (contact_id)
)
```

Hướng dẫn:

Bước 1: Tạo Servlet hoặc trang 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
- Kết nối CSDL : jdbc kết nối CSDL

// Thay đổi kết nối theo SQL Server của bạn

```
private String dbURL = "jdbc:sqlserver://localhost\\MSSQLLocalDB;databaseName=QLThoiKhoaBieu;" + 1 usage
    "IntegratedSecurity=true;encrypt=false";
private String dbUser = "sa"; 1 usage
private String dbPass = "sapassword"; 1 usage
```

■ Insert dữ liệu vào bảng gồm cả dữ liệu Text và dữ liệu Image

```
protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    // super.doPost(req, resp);
    String firstName = req.getParameter(s: "firstName");
    String lastName = req.getParameter(s: "lastName");

    InputStream inputStream = null;
    Part filePart = req.getPart(s: "photo");
    if (filePart != null) {
        inputStream = filePart.getInputStream();
    }

    Connection conn = null;
    String message = null;

    try {
        DriverManager.registerDriver(new com.microsoft.sqlserver.jdbc.SQLServerDriver());
        conn = DriverManager.getConnection(dbURL, dbUser, dbPass);

        String sql = "INSERT INTO StudentPhotos (first_name, last_name, photo) values (?, ?, ?)";
        PreparedStatement statement = conn.prepareStatement(sql);
        statement.setString(parameterIndex: 1, firstName);
        statement.setString(parameterIndex: 2, lastName);

        if (inputStream != null) {
            statement.setBinaryStream(parameterIndex: 3, inputStream, (int) filePart.getSize());
        } else {
            statement.setNull(parameterIndex: 3, Types.BLOB);
        }

        int row = statement.executeUpdate();
        if (row > 0) {
            message = "Upload and save into database successful!";
        }
    } catch (Exception ex) {
        ex.printStackTrace();
        message = "ERROR: " + ex.getMessage();
    } finally {
        if (conn != null) try { conn.close(); } catch (SQLException ignore) {}
    }

    resp.setContentType("text/html;charset=UTF-8");
    resp.getWriter().println("<h3>" + message + "</h3>");
}
```

Bước 3: Trả về kết quả sau khi đã thực hiện xong.

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.

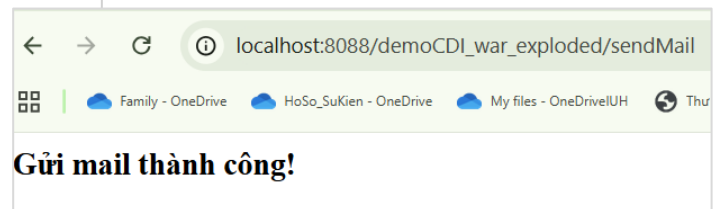
Luồng gửi mail trong Servlet

1. User nhập form JSP → to, subject, content, file.	1. Form HTML (multipart/form-data) chứa: To, Subject, Body, và <input type="file" multiple>.
2. Servlet nhận request , tạo Session với SMTP.	

3. Tạo Message → set From, To, Subject, Body. 4. Nếu có file đính kèm → dùng MimeBodyPart + Multipart. 5. Gọi Transport.send(message) . 6. Hiển thị kết quả cho người dùng.	2. Servlet gắn @MultipartConfig để đọc Part của tệp. 3. Tạo Session SMTP (host/port/TLS + Authenticator). 4. Tạo MimeMessage: <ul style="list-style-type: none"> Một MimeBodyPart cho body (plain text hoặc HTML). Với mỗi file: tạo MimeBodyPart + DataHandler (JAF) → addBodyPart. 5. Transport.send(msg) để gửi.
--	---

Add Dependencies:

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



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 MÔN LẬP TRÌNH WWW JAVA

Chương 3: Jakarta Server Pages - JSPs

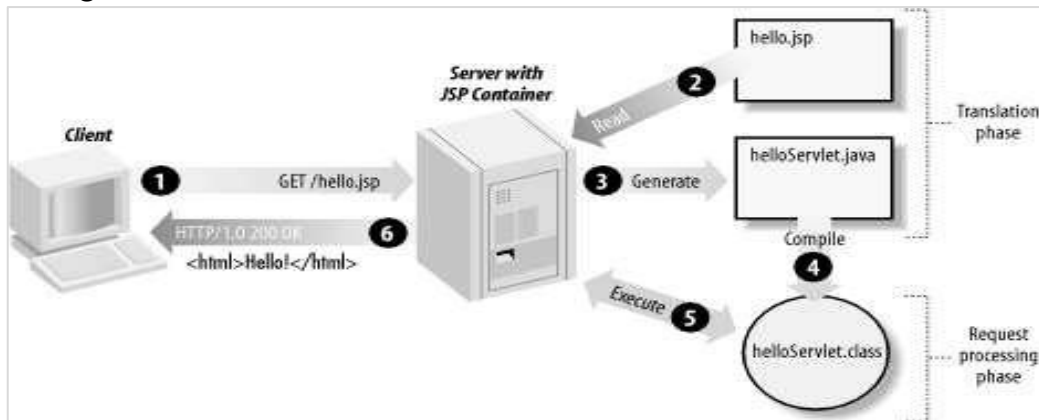
Mục tiêu:

- 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 có thể lưu trong từng package 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
<c:out >	Giống <%= ... >, nhưng cho các Expression
<c:set >	Thiết lập kết quả của 1 Expression trong một 'scope'
<c:remove >	Gỡ bỏ một biến mục tiêu (từ một biến scope cụ thể, nếu đã xác định)
<c:catch>	Bắt bất kỳ Throwable
<c:if>	Thẻ điều kiện đơn giản.
<c:choose>	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>
<c:when>	Thẻ phụ của <choose> mà điều kiện được ước lượng là true
<c:otherwise >	Thẻ phụ của <choose> mà theo sau thẻ <when> và chỉ chạy nếu tất cả điều kiện là 'false'
<c:import>	Dùng để import.
<c:forEach >	Thẻ lặp cơ bản.
<c:forTokens>	Lặp qua các token, được phân biệt bởi các dấu phân tách (delimiter) đã cung cấp
<c:param>	Thêm một parameter tới một URL của thẻ đang chứa 'import'
<c:redirect >	Redirect tới một URL mới
<c:url>	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
<fmt:formatNumber>	Trả lại giá trị số với định dạng cụ thể

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

SQL Tags (JSTL)

Cú pháp

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

Thẻ	Miêu tả
<sql:setDataSource>	Tạo một DataSource kết nối vào hệ quản trị cơ sở dữ liệu.
<sql:query>	Thực thi SQL query được định nghĩa trong tag đóng/mở hoặc thông qua thuộc tính sql
<sql:update>	Thực thi SQL update được định nghĩa trong tag đóng/mở hoặc thông qua thuộc tính sql
<sql:param>	Thiết lập một parameter trong một lệnh SQL
<sql:dateParam>	Thiết lập một parameter trong một lệnh SQL tới giá trị java.util.Date đã xác định
<sql:transaction >	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
<x:out>	Giống <%= ... >, nhưng dành cho các XPath Expression
<x:parse>	Sử dụng để <i>parse</i> XML data được xác định hoặc thông qua một thuộc tính hoặc trong phần thân thẻ
<x:set >	Thiết lập một biến tới giá trị của một XPath expression

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

JSP Expression Language (EL là một phần của đặc tả của JSTL 1.0, đi kèm với JSP 1.2 và có thể được sử dụng như các attribute của thẻ JSTL)

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:

The registration form includes the following fields and sections:

- First name:** Text input (max 30 characters a-z and A-Z)
- Last name:** Text input (max 30 characters a-z and A-Z)
- Date of birth:** Day, Month, and Year dropdown menus
- Email:** Text input
- Mobile number:** Text input (10 digit number)
- Gender:** Radio buttons for Male and Female
- Address:** Text area
- City:** Text input (max 30 characters a-z and A-Z)
- Pin code:** Text input (6 digit number)
- State:** Text input (max 30 characters a-z and A-Z)
- Country:** Text input (pre-filled with India)
- Hobbies:** Checkboxes for Drawing, Singing, Dancing, Sketching, Others, and a text input for others
- Qualification:** Table with 4 rows (Class X, Class XII, Graduation, Masters) and 4 columns (Sl.No., Examination, Board, Percentage, Year of Passing)
- Course applies for:** Radio buttons for BCA, B.Com, B.Sc, B.A
- Buttons:** Submit and Reset

Hướng dẫn:

B1. Tạo lớp *Student* bao gồm các thuộc tính mô tả trên form, các thuộc tính khai báo private kèm theo các phương thức *get/set* + *constructors*.

B2. Tạo form đăng ký `<form action="RegistrationForm" name="formDangKy" method="GET">`

B3. Tạo Servlet xử lý thao tác nhập dữ liệu của form:

Minh họa Servlet như sau:

```
@WebServlet("/RegistrationForm")
public class RegistrationForm extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter dt = response.getWriter();

        String fname = request.getParameter("txtFName");
        String lname = request.getParameter("txtLName");
        String day = request.getParameter("day");
        .....

        request.setAttribute("student", sv);

        RequestDispatcher rd = request.getRequestDispatcher("ResultForm.jsp");
        rd.forward(request, response);
    }
}
```

B4. Tạo trang *ResultForm.jsp* hiển thị kết quả

Sử dụng các taglib xuất dữ liệu trong trang *jsp*

```

<%@ taglib prefix="c" uri="jakarta.tags.core" %>
<%@ taglib prefix="fmt" uri="jakarta.tags.fmt" %>
<%@page import="se.iuh.edu.vn.Student"%>
<body>
    <%
        Student svt= new Student();
        svt = (Student)request.getAttribute("student");
        out.println("First name:" + svt.getFirstName()
            + "<br/> Last name: " + svt.getLastName()
            + "<br/> Email : " + svt.getEmail()
            + "<br/> Gender: " + svt.getGender()
            + "<br/> Hobbies: " + svt.getHobbies()
            + "<br/> Birthday: " + svt.getBirthday()
            );
    %>
</body>

```

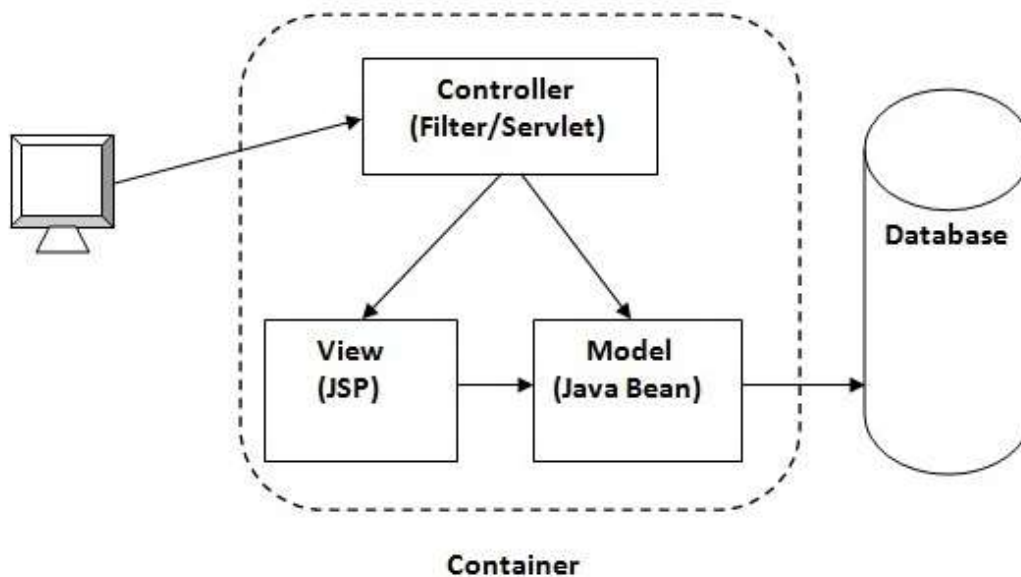
Bài 2. JSPs - Model View Controller

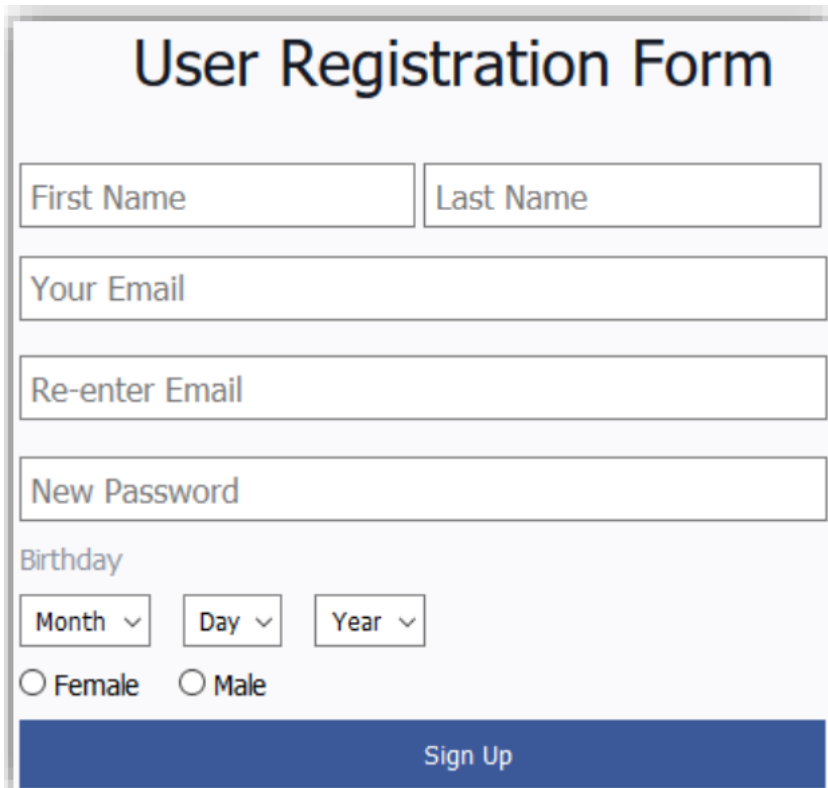
Thực hiện form đăng ký tài khoản với JSPs.

Sau khi đăng ký thành công cập nhật danh sách tài khoản (không hiển thị password),

Thực hiện theo mô hình MVC.

Kết nối CSDL MariaDB/SQL Server khi Sign Up lưu thông tin vào table





The image shows a web form titled "User Registration Form". It contains several input fields: "First Name" and "Last Name" (side-by-side), "Your Email", "Re-enter Email", and "New Password". Below these is a "Birthday" section with three dropdown menus for "Month", "Day", and "Year". At the bottom of the birthday section are two radio buttons labeled "Female" and "Male". A blue "Sign Up" button is at the very bottom of the form.

User Registration Form	
First Name	Last Name
Your Email	
Re-enter Email	
New Password	
Birthday	
Month ▾	Day ▾
Year ▾	
<input type="radio"/> Female <input type="radio"/> Male	
Sign Up	

Hướng dẫn:

B1: Tạo class Account.java gồm các thành phần như form minh họa

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

B2: Tạo RegisterForm.jsp như hình

B3: Tạo RegisterFormServlet.java