Bài tập bắt buộc của Tuần XX GV sẽ công bố cuối giờ thực hành. Bài tập bắt buộc sẽ gửi với Subject và tập tin nén kèm theo:
LTWWW_JAVA_{NGAYTHANGNAM}_TUAN{XX}_HOTENSINHVIEN

(trong đó XX sẽ là 01, 02 ... 10).

VD: LTWWW_JAVA_21082025_TUAN01_HOTENSINHVIEN

Sinh viên nộp sai yêu cầu bài tập Tuần nào xem như không nộp bài tập Tuần đó.

BÀI TẬP TUẦN 02 MÔN LẬP TRÌNH WEB NÂNG CAO (JAVA)

1. ***Chương 1: Application Model - Web Application Architecture***
2. ***Chương 2: Jakarta 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 Web Server.
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:\MaSV_HoTen\Tuan01*
- *Tạo Project **MaSV_HoTen_Tuan01** trong thư mục T:\MaSV_HoTen\Tuan01\ trong IntelliJ/Eclipse Java EE (Java Platform Enterprise Edition). 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. *Java Servlet - Filter*

- Dùng Filter trong Servlet để chặn các Requests từ Client trước khi Client gửi yêu cầu truy cập dữ liệu Back-End.
- Dùng Filter trong Servlet để thao tác các Response Server trước khi gửi trở lại tới Client.
- *Hướng dẫn:*

  Các Filter thực hiện trong web.xml ánh xạ tới tên các Servlet và URL Pattern:

  - Authentication Filters
  - Image Conversion Filters
  - Logging Filters
  - MIME-TYPE Chain Filters.
  - …

Bài 2. *Java Servlet - Upload files*

Thực hiện upload nhiều file lưu trữ ở Server.

Bài 3. *Java Servlet - Upload hình, lưu CSDL*

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

- CSDL:

```
MSSQL

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)
)
```

```
MariaDB

CREATE TABLE `contacts` (
  `first_name` varchar(50) NOT NULL,
  `last_name` varchar(50) NOT NULL,
  `photo` longblob DEFAULT NULL,
  `contact_id` int(11) NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (`contact_id`)
) ENGINE=InnoDB AUTO_INCREMENT=0 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci;
```

■ Maven dependency:

```
<!--
https://mvnrepository.com/artifact/com.microsoft.sqlserver/mssql-
jdbc -->
<dependency>
     <groupId>com.microsoft.sqlserver</groupId>
     <artifactId>mssql-jdbc</artifactId>
     <version>11.2.3.jre17</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.mariadb.jdbc/mariadb-
java-client -->
<dependency>
    <groupId>org.mariadb.jdbc</groupId>
    <artifactId>mariadb-java-client</artifactId>
    <version>3.5.1</version>
</dependency>
```

■ Tạo kết nối đến Database:

```
public class DBConnection {
    // MSSQL
    // private static final String JDBC_URL =
"jdbc:sqlserver://localhost:1433;databaseName=UploadFileServlet
DB;encrypt=false;";
    // private static final String JDBC_USER = "sa";
    // private static final String JDBC_PASSWORD =
"sapassword";

    // MariaDB
    private static final String JDBC_URL =
"jdbc:mariadb://localhost:3306/UploadFileServletDB";
    private static final String JDBC_USER = "root";
    private static final String JDBC_PASSWORD = "123456";
    static {
        try {
           // Load driver chỉ một lần duy nhất
           //
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
            Class.forName("org.mariadb.jdbc.Driver");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
```

```
    public static Connection getConnection() throws
SQLException {
        return DriverManager.getConnection(JDBC_URL, JDBC_USER,
JDBC_PASSWORD);
    }
}
```

- HTML:

```
<form method="post" action="upload" enctype="multipart/form-data">
  <table border="0">
    <tr>
        <td>First Name: </td>
        <td><input type="text" name="firstName" size="50"/></td>
    </tr>
    <tr>
        <td>Last Name: </td>
        <td><input type="text" name="lastName" size="50"/></td>
    </tr>
    <tr>
      <td>Portrait Photo: </td>
      <td><input type="file" name="photo" size="50"/></td>
    </tr>
    <tr>
      <td colspan="2">
        <input type="submit" value="Save">
      </td>
    </tr>
  </table>
</form>
```

■ Servlet:

```
@WebServlet(name = "uploadServlet", value = "/upload")
@MultipartConfig
public class UploadServlet extends HttpServlet {

    private String uploadPathToSource;

    @Override
    public void init(ServletConfig config) throws
ServletException {
        super.init(config);
        uploadPathToSource =
"T:\\UploadFileToDatabase\\src\\main\\webapp\\uploads\\";
    }

    @Override
    protected void doPost(HttpServletRequest req,
HttpServletResponse resp) throws ServletException, IOException
{
        String firstName = req.getParameter("firstName");
        String lastName = req.getParameter("lastName");
        InputStream inputStream = null;
        // luồng dữ liệu nhập của upload file
        // lấy thông tin tập tin upload trong form, form này
gồm nhiều phần dữ liệu text và file (multipart request)
        Part filePart = req.getPart("photo");

        String fileUploadName = "";
        if (filePart != null && filePart.getSubmittedFileName()
!= null && !filePart.getSubmittedFileName().isEmpty()) {
            fileUploadName = filePart.getSubmittedFileName();
            inputStream = filePart.getInputStream();
        }

        String message = null;




        // try-with-resources
        try (Connection conn = DBConnection.getConnection()) {

            // Insert dữ liệu vào CSDL UploadFileServletDB,
trường hợp này bảng contacts (khóa tự động tăng)
            String sqlInsert = "INSERT INTO contacts
```

```
(first_name, last_name, photo) values (?, ?, ?)";
            PreparedStatement statement =
conn.prepareStatement(sqlInsert);
            statement.setString(1, firstName);
            statement.setString(2, lastName);
            if (inputStream != null) {
                statement.setBlob(3, inputStream);
            }
            int row = statement.executeUpdate();
            // thực hiện lưu thông tin vào CSDL
            if (row > 0) {
                message = "File uploaded and saved into
database";
            }

            // đọc CSDL lưu file
            String filePath = uploadPathToSource +
fileUploadName;
            String sqlSelect = "SELECT photo FROM contacts
WHERE first_name=? AND last_name=?";
            statement = conn.prepareStatement(sqlSelect);
            statement.setString(1, firstName);
            statement.setString(2, lastName);
            ResultSet result = statement.executeQuery();
            if (result.next()) {
                Blob blob = result.getBlob("photo");
                inputStream = blob.getBinaryStream();
                OutputStream outputStream = new
FileOutputStream(filePath);
                int bytesRead = -1;
                byte[] buffer = new byte[1024];
                while ((bytesRead = inputStream.read(buffer))
!= -1) {
                    outputStream.write(buffer, 0, bytesRead);
                }
                inputStream.close();
                outputStream.close();
            }

        } catch (SQLException e) {
            message = "ERROR: " + e.getMessage();
            e.printStackTrace();
        }

        // Forward đến servlet MessageServlet
        req.setAttribute("message", message);
        getServletContext().getRequestDispatcher("/message").
```

```
forward(req, resp);
    }
}
```

```
@WebServlet(name = "messageServlet", value = "/message")
public class MessageServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req,
HttpServletResponse resp) throws ServletException, IOException
{
        PrintWriter out = resp.getWriter();
        out.println("<html>");

out.println("<head>");out.println("<title>MessageServlet</title
>");
        out.println("</head>");out.println("<body>");
        out.println("<h1>" + req.getAttribute("message")+
"</h1>");
        out.println("</body>");out.println("</html>");
        out.close();
    }

    @Override
    protected void doPost(HttpServletRequest req,
HttpServletResponse resp) throws ServletException, IOException
{
        this.doGet(req, resp);
    }
}
```

Bài 4. *Jakarta Servlet – CDI*

Làm lại bài tập: Jakarta Servlet - Thao tác với doGet() (Tuần 1- Bài 3)

- Response trả về dạng: application/json

Gợi ý:

- Tạo class User
- Sử dụng thư viện Jackson/Gson để chuyển từ Object sang json string.
- Dependencies

```
<!--
     https://mvnrepository.com/artifact/com.google.code.gson/gson --
>
<dependency>
```

```
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.11.0</version>
</dependency>
<!--
     https://mvnrepository.com/artifact/org.jboss.weld.servlet/weld-
servlet-core -->
<dependency>
     <groupId>org.jboss.weld.servlet</groupId>
     <artifactId>weld-servlet-core</artifactId>
     <version>5.1.2.Final</version>
</dependency>
```

- WEB-INF/beans.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="https://jakarta.ee/xml/ns/jakartaee"
xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
https://jakarta.ee/xml/ns/jakartaee/beans_4_0.xsd"
version="4.0" bean-discovery-mode="all">
</beans>
```

Bài 5. *Jakarta - RestAPI*

Link tham khảo: https://restfulapi.net/resource-naming/

- Làm lại VD trong Slide lý thuyết.
- Tạo table users: id, first_name, last_name, dob
- Viết API:
  - Lấy danh sách users: **HTTP GET /api/users**
  - Lấy thông tin của 1 user: **HTTP GET /api/users/{id}**
  - Thêm mới 1 user: **HTTP POST /api/users**
  - Cập nhật 1 user: **HTTP PUT /api/users/{id}**
  - Xoá 1 user: **HTTP DELETE /api/users/{id}**

Bài 6. *WebSocket – Làm lại VD trong slide lý thuyết.*

Bài 7. *JavaMail*

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 mail dùng:

- **SMTP (Simple Mail Transfer Protocol)** dùng để gửi email trên Internet.
- **POP3 (Post Office Protocol phiên bản 3)** dùng để nhận email từ Server.
- **IMAP (Internet Message Access Protocol)**

Lớp trong JavaMail

- Session: đại diện cho một lần gửi nhận mail
- Message: gồm các phương thức hỗ trợ việc gửi nhận mail. Các thành phần message cơ bản bao gồm địa chỉ người gửi và địa chỉ người nhận, tiêu đề mail (subject) và nội dung (body gồm nhiều parts)
- Transport: hỗ trợ việc gửi nhận mail qua Internet
- Address: Internet Address là đối tượng thực thi tạo các địa chỉ cho việc gửi nhận email trên Internet.
- Transport: gửi message trên Internet thông qua thông tin được xác định trên session
- Authenticator: xác thực tài khoản

B1. Dependency:

```
<!-- https://mvnrepository.com/artifact/jakarta.mail/jakarta.mail-api -->
<dependency>
    <groupId>jakarta.mail</groupId>
    <artifactId>jakarta.mail-api</artifactId>
    <version>2.1.3</version>
</dependency>
```

B2. Chuẩn bị thông tin tài khoản để gửi mail. VD dùng Google Mail

Incoming Mail (POP3) Server - requires SSL:	pop.gmail.com **Use SSL**: Yes **Port**: 995 1
Outgoing Mail (SMTP) Server - requires TLS:	smtp.gmail.com (use authentication) **Use Authentication**: Yes **Use STARTTLS**: Yes (some clients call this SSL) **Port**: 465 or 587 2 3
Account Name:	your full email address (including @gmail.com or @your_domain.com)
Email Address:	your email address (username@gmail.com or username@your_domain.com)
Password:	your Gmail password 4

B3. Thực hiện gửi mail

```
String recipient = "recipient@gmai.com";
String subject = "Subject Of Email";
String body = "Hello,\nThis is a test email.";
String attachmentPath = "path_to_your_attachment_file";
String senderEmail = "your-email@gmail.com";
String password = "your-email-password";
String host = "smtp.gmail.com";
int port = 587;

Properties properties = new Properties();
properties.put("mail.smtp.host", host);
properties.put("mail.smtp.port", port);
properties.put("mail.smtp.auth", "true");
properties.put("mail.smtp.starttls.enable", "true");
Session session = Session.getInstance(properties, new Authenticator()
{
    protected PasswordAuthentication getPasswordAuthentication() {
        return new PasswordAuthentication(senderEmail, password);
    }
});
```

```
try {
  // Tạo đối tượng mặc định MimeMessage.
  MimeMessage message = new MimeMessage(session);
  message.setFrom(new InternetAddress(senderEmail));
  message.addRecipient(Message.RecipientType.TO, new
InternetAddress(recipient));
  message.setSubject(subject);

 // Tạo đối tượng BodyPart của email.
 BodyPart messageBodyPart = new MimeBodyPart();

 // Nội dung của email.
 messageBodyPart.setText(body);

 // Email sẽ gồm 2 part (text, file attached)
 Multipart multipart = new MimeMultipart();
 multipart.addBodyPart(messageBodyPart);

 // Phần xử lý với file attached
 if (attachmentPath != null && !attachmentPath.isEmpty()) {
     MimeBodyPart attachPart = new MimeBodyPart();
     DataSource source = new FileDataSource(attachmentPath);
     attachPart.setDataHandler(new DataHandler(source));
     attachPart.setFileName(source.getName());
     multipart.addBodyPart(attachPart);
  }
  message.setContent(multipart);

  // Gửi mail
  Transport.send(message);
  System.out.println("Email with attachment sent successfully to: " +
recipient);
}
catch (MessagingException e) {
   e.printStackTrace();
}
```