1️⃣

Bài 2: JSTL – Jakarta Standard Tag Library

1. Giới thiệu JSTL

JSTL (Jakarta Standard Tag Library) là tập hợp các thẻ chuẩn dành cho JSP, giúp:

- Loại bỏ việc dùng Scriptlet (`<% %>`)
- Tách biệt **logic hiển thị** khỏi **logic xử lý**
- Code dễ đọc, dễ bảo trì, thân thiện với HTML
- Phù hợp với mô hình **MVC (Servlet + JSP)**

👉 JSTL thường được sử dụng kết hợp với **Expression Language (EL)**.

---

2. Cài đặt JSTL

2.1. Tải thư viện

Bạn có thể tải file JSTL (`jstl.jar` , `standard.jar`) từ:

- Maven Central
- Google Drive (giảng viên cung cấp)
- Thư mục `WEB-INF/lib`

📌 Với Maven:

```
<dependency>
  <groupId>jakarta.servlet.jsp.jstl</groupId>
  <artifactId>jakarta.servlet.jsp.jstl-api</artifactId>
  <version>2.0.0</version>
</dependency>
```

---

3. Cấu hình Taglib

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

📌 `prefix="c"` là tiền tố khi dùng JSTL Core Tag.

---

4. JSTL Core Tags

4.1. c:out – Xuất dữ liệu (thay thế `<%= %>`)

```
<c:out value="customer.address.street"/>
<c:out value="${'Hello'}"/>
```

📌 Tự động escape XML/HTML → an toàn hơn `out.print()`.

4.2. c:set – Gán giá trị cho biến

```
<c:set var="x" value="10" />
<c:set var="salary" scope="session" value="${2000 * 2}" />
<c:out value="${salary}" />
```

📌 `scope`: page | request | session | application

4.3. c:if – Câu điều kiện

```
<c:set var="x" value="10" />

<c:if test="${x > 5}">
  <p>x is greater than 5</p>
</c:if>
```

📌 Không có else → dùng `c:choose` khi cần.

4.4. c:choose, c:when, c:otherwise – If / Else nâng cao

```
<c:set var="day" value="${1}" />
```

```
<p>
  <c:choose>
    <c:when test="${day == 0}">Sunday</c:when>
    <c:when test="${day == 1}">Monday</c:when>
    <c:when test="${day == 2}">Tuesday</c:when>
    <c:when test="${day == 3}">Wednesday</c:when>
    <c:when test="${day == 4}">Thursday</c:when>
    <c:when test="${day == 5}">Friday</c:when>
    <c:otherwise>Saturday</c:otherwise>
  </c:choose>
</p>
```

📌 Tương đương `switch-case` trong Java.

4.5. c:catch – Bắt lỗi (Exception)

```
<c:catch var="catchException">
  <% int x = 5 / 0; %>
</c:catch>

<c:if test="${catchException != null}">
  <p>Error: ${catchException.message}</p>
</c:if>
```

📌 Hữu ích khi render view nhưng không làm sập trang.

4.6. c:forEach – Duyệt mảng / collection

```
<ul>
  <c:forEach var="color" items="${colors}">
    <li>${color}</li>
  </c:forEach>
</ul>
```

📌 Hỗ trợ `begin`, `end`, `step`, `varStatus`.

```
<c:forEach var="c" items="${colors}" varStatus="st">
  ${st.index} - ${c}<br/>
```

```
</c:forEach>
```

4.7. c:forTokens – Duyệt chuỗi phân tách

```
<c:forTokens items="apple,banana,orange" delims="," var="fruit">
  <c:out value="${fruit}" /><br />
</c:forTokens>
```

📌 Dùng khi không cần convert String → List.

5.EL (Expression Language)

là ngôn ngữ biểu thức trong JSP, cho phép truy cập dữ liệu của các bean, đối tượng ngầm định (implicit objects), thuộc tính trong `request`, `session`, `application`, hay `pageScope`, mà không cần viết mã Java thuần.

Truy cập thuộc tính

Lấy giá trị của biến `message` (có thể nằm trong `request`, `session`, `application`...):

```
<p>${message}</p>
```

Lấy thuộc tính `name` của bean `user` (tương đương với `user.getName()` trong Java):

```
<p>${user.name}</p>
```

Cách viết tương đương với `${user.name}`, cho phép dùng khi tên thuộc tính là biến hoặc chứa ký tự đặc biệt:

```
<p>${user["name"]}</p>
```

Biểu thức điều kiện (Toán tử 3 ngôi)

Nếu `user.name` khác `null`, thì hiển thị tên; ngược lại hiển thị `'Guest'`:

```
<p>${user.name != null ? user.name : 'Guest'}</p>
```

Toán tử logic

Thường kết hợp với `c:if`

Trả về `true` nếu tuổi của người dùng lớn hơn 18:

```
<p>${user.age > 18}</p>
```

Trả về `true` nếu `user.name` là `null` hoặc chuỗi rỗng (`""`):

```
<p>${empty user.name}</p>
```

Trả về `true` nếu `user.name` tồn tại:

```
<p>${not empty user.name}</p>
```

Làm việc với các Scope (phạm vi)

Truy cập thuộc tính `name` trong các phạm vi tương ứng:

```


<p>${pageScope.name}</p>
<p>${requestScope.name}</p>
<p>${sessionScope.name}</p>
<p>${applicationScope.name}</p>
```

Sử dụng trong vòng lặp (kết hợp JSTL)

Duyệt qua danh sách `todoList`, hiển thị tiêu đề và trạng thái của từng công việc:

```
<c:forEach var="todo" items="${todoList}">
  <p>${todo.title} - ${todo.completed ? 'Done' : 'Pending'}</p>
</c:forEach>
```

CALCULATOR CRUD (Servlet + JSP + JSTL)

Mô tả

Xây dựng ứng dụng web cho phép người dùng **tính toán** (+, -, *, /) và **lưu lịch sử phép tính**. Người dùng có thể **CRUD** (thêm/xem/sửa/xóa) các bản ghi phép tính.

Yêu cầu chức năng

1) Create (Thực hiện phép tính & lưu)

- Trang calculator.jsp có form nhập:
  - number1, number2
  - operator (+, -, *, /)
- Submit gửi tới Servlet:
  - Tính result
  - Nếu chia cho 0 → báo lỗi: **"Không thể chia cho 0"**, **không lưu**
  - Nếu hợp lệ → **lưu 1 record** vào danh sách lịch sử

Record gồm: id, number1, number2, operator, result, createdAt

---

2) Read (Xem danh sách lịch sử)

- Trang history.jsp hiển thị bảng lịch sử phép tính:
  - Cột: ID | Biểu thức | Kết quả | Thời gian | Thao tác
- Dùng **JSTL c:forEach** để render list

---

3) Update (Sửa bản ghi)

- Mỗi dòng có nút **Edit**
- Khi Edit:
  - Hiển thị form với dữ liệu cũ
  - Cho phép sửa number1, number2, operator
  - Submit → Servlet tính lại result và cập nhật record
  - Nếu chia cho 0 → báo lỗi, không cập nhật

---

4) Delete (Xóa bản ghi)

- Mỗi dòng có nút **Delete**
- Xóa record theo id và quay lại danh sách

Yêu cầu kỹ thuật

- Mô hình **MVC**
  - Servlet: xử lý nghiệp vụ + điều hướng
  - JSP: hiển thị (bắt buộc **JSTL + EL**, **không dùng scriptlet**)
- Lưu dữ liệu:
  - **Tối thiểu:** dùng List trong application scope hoặc session scope
  - **Khuyến khích:** DAO + DB (nếu lớp đã học)

Output mong đợi

- Trang tính toán nhập liệu
- Trang lịch sử hiển thị:
  - Ví dụ: 5 + 3 = 8
- CRUD hoạt động:
  - Add mới sau khi tính
  - Edit cập nhật đúng kết quả
  - Delete xóa khỏi danh sách
- Chia cho 0 → hiển thị **“Không thể chia cho 0”**