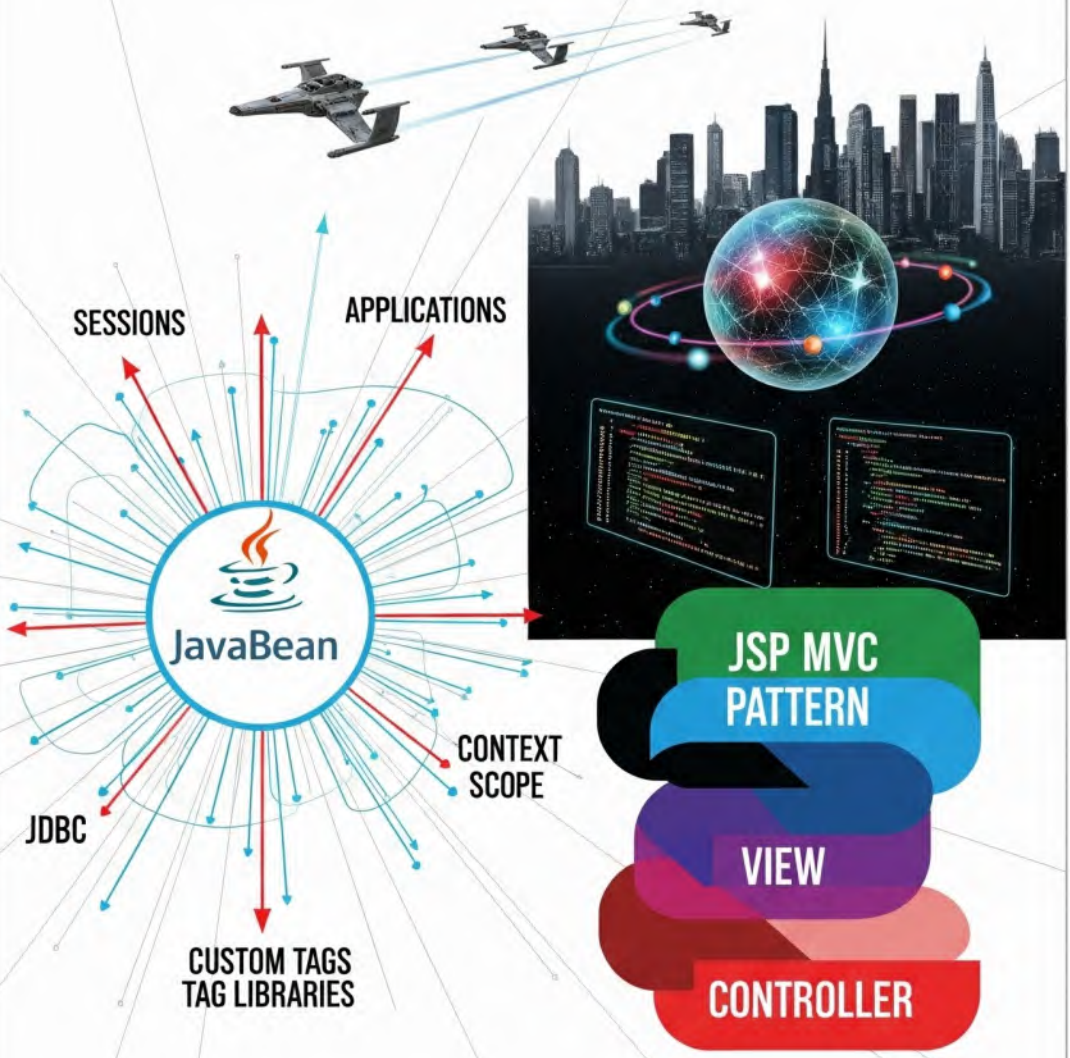


JSP WEB PROGRAMMING: INTERMEDIATE
(INTEGRATIVE-GENERATIVE AI EDITION)

Student Price Book Center

คำนำ

หนังสือ *JSP Web Programming: Intermediate* เล่มนี้จัดทำขึ้นเพื่อเป็นคู่มือฉบับสมบูรณ์สำหรับผู้ที่มีพื้นฐาน JSP เบื้องต้นแล้ว และต้องการพัฒนาทักษะการเขียนเว็บแอปพลิเคชันด้วย Java ให้ลึกซึ้งและเป็นระบบมากยิ่งขึ้น โดยมุ่งเน้นการประยุกต์ใช้เทคนิคสำคัญระดับกลางที่จำเป็นต่อการพัฒนาเว็บในระดับมืออาชีพ ทั้งในด้านการจัดการข้อมูล การแยกส่วนของระบบ และการประมวลผลที่ปลอดภัยและยืดหยุ่น

เนื้อหาในเล่มนี้เริ่มต้นจาก **บทที่ 5** ซึ่งอธิบายการใช้ *JavaBeans* ร่วมกับ JSP เพื่อแยกตรรกะของระบบออกจากส่วนแสดงผล โดยใช้แท็ก <jsp:useBean>, <jsp:setProperty>, และ <jsp:getProperty> อย่างมีประสิทธิภาพ พร้อมตัวอย่างการสร้าง JavaBean Class ที่ชัดเจน ช่วยให้นักพัฒนาสามารถจัดการข้อมูลแบบเป็นระบบและลดการใช้ Scriptlet ลงได้อย่างมาก

ใน **บทที่ 6** ผู้อ่านจะได้เรียนรู้วิธีการเชื่อมต่อฐานข้อมูลผ่าน *JDBC* ซึ่งเป็นหัวใจของเว็บแอปพลิเคชันที่ต้องติดต่อกับข้อมูลจริง บทนี้จะพาไปตั้งค่าการเชื่อมต่อกับ MySQL อย่างละเอียด ตั้งแต่การติดตั้ง MySQL Workbench การสร้างตาราง จนถึงการเขียน JSP เพื่อเพิ่ม แก้ไข ลบ และแสดงผลข้อมูล พร้อมเน้นการจัดการ Resource เช่น Connection และ Statement อย่างถูกต้อง

บทที่ 7 เจาะลึกการจัดการข้อมูลใน *Scope ต่าง ๆ* ของ JSP ได้แก่ Page, Request, Session และ Application โดยเน้นความแตกต่างด้านช่วงชีวิตและการมองเห็นตัวแปรในแต่ละระดับ พร้อมแนะนำการใช้ HttpSession และ ServletContext อย่างถูกวิธี เพื่อเพิ่มความปลอดภัยและประสิทธิภาพในการจัดเก็บข้อมูลชั่วคราวและถาวร

ต่อมาใน **บทที่ 8** ผู้อ่านจะได้เรียนรู้การใช้งาน *JSP Tag Libraries* ทั้ง JSTL Core Tags และแท็กเฉพาะด้าน เช่น SQL, XML และ Formatting รวมถึงวิธีการสร้าง *Custom Tag Library* ของตนเอง โดยใช้ TLD (Tag Library Descriptor) และ Tag Handler Class เพื่อช่วยลดความซ้ำซ้อนของโค้ด และเพิ่มความสามารถในการนำแท็กกลับมาใช้ซ้ำได้ในหลายหน้า JSP

ใน **บทที่ 9** ซึ่งเป็นบทส่งท้ายของหนังสือเล่มนี้ ผู้อ่านจะได้ศึกษา *แนวคิด MVC (Model-View-Controller)* ในการพัฒนาเว็บอย่างเป็นระบบ โดยแยกความรับผิดชอบระหว่าง Controller (Servlet), View (JSP) และ Model (Java Class) พร้อมตัวอย่างโปรเจกต์ที่ประยุกต์ใช้ MVC สำหรับการรับและประมวลผลฟอร์ม การส่งข้อมูลด้วย Request Attributes และการควบคุมการตอบกลับอย่างถูกต้อง

หนังสือเล่มนี้จึงเหมาะอย่างยิ่งสำหรับนักพัฒนาระดับกลางที่ต้องการยกระดับจากการเขียน JSP แบบพื้นฐาน สู่การพัฒนาแอปพลิเคชันเว็บที่มีโครงสร้างดี มีความปลอดภัย ใช้ทรัพยากรอย่างมีประสิทธิภาพ และสามารถดูแลรักษาโค้ดได้ง่ายในระยะยาว

ขอขอบคุณผู้อ่านทุกท่านที่เลือกใช้หนังสือเล่มนี้เป็นส่วนหนึ่งในการพัฒนาทักษะของตนเอง และหวังว่าเนื้อหาทั้งหมดในเล่มจะเป็นประโยชน์และเป็นแรงบันดาลใจในการต่อยอดสู่โปรเจกต์จริงและการเรียนรู้ขั้นสูงต่อไป.

ด้วยรักและปรารถนาดี

ศูนย์หนังสือราคานักเรียน

สารบัญ

หน้า

บทที่ 5
การใช้ JavaBeans กับ JSP
(JavaBeans and JSP)

เนื้อหา

- การใช้ JavaBeans กับ JSP
- รายละเอียดเชิงลึก: การใช้ JavaBeans กับ JSP
- การสร้าง JavaBean Class แบบง่าย (Properties, Getter/Setter)
- การใช้งาน <jsp:useBean> เพื่อสร้าง/เรียกใช้ JavaBean ใน JSP
- การกำหนด Properties ผ่าน <jsp:setProperty> และการดึงข้อมูล Properties ผ่าน <jsp:getProperty>
- การแยก Business Logic ออกจาก JSP เพื่อความสะอาดของโค้ด

บทนำ - บทที่ 5: การใช้ JavaBeans กับ JSP

ในกระบวนการพัฒนาแอปพลิเคชันเว็บด้วย JSP (JavaServer Pages) การรักษาความเป็นระเบียบของโค้ดและการแยกส่วนความรับผิดชอบ (separation of concerns) ถือเป็นแนวทางที่สำคัญอย่างยิ่ง เพื่อให้โค้ดสามารถดูแลรักษาได้ง่าย ขยายระบบในอนาคตได้สะดวก และสอดคล้องกับหลักการ MVC (Model-View-Controller) บทที่ 5 นี้จะพาผู้อ่านไปรู้จักและใช้งาน *JavaBeans* ซึ่งเป็นองค์ประกอบสำคัญในการช่วยจัดการข้อมูลและแยก business logic ออกจาก presentation layer อย่างมีประสิทธิภาพ

JavaBeans คือคลาส Java ทั่วไปที่มีรูปแบบมาตรฐาน ซึ่งประกอบด้วย properties (ตัวแปร), methods (getters/setters) และคอนสตรักเตอร์แบบไม่มีพารามิเตอร์ JavaBeans ถูกออกแบบมาให้สามารถนำมาใช้งานร่วมกับเทคโนโลยี JSP ได้โดยตรง ผ่านแท็กเฉพาะ เช่น <jsp:useBean>, <jsp:setProperty> และ <jsp:getProperty> สิ่งนี้ทำให้นักพัฒนาไม่ต้องเขียนโค้ด Java ซับซ้อนภายใน JSP ส่งผลให้หน้าเว็บสะอาดและดูแลได้ง่าย

ในช่วงแรกของบทนี้ ผู้อ่านจะได้ทำความเข้าใจกับนิยามและประโยชน์ของ JavaBeans อย่างชัดเจน โดยเน้นที่การนำมาใช้เป็นตัวแทนของข้อมูล (data model) เช่น ข้อมูลผู้ใช้ ผลิตภัณฑ์ หรือรายการคำสั่งซื้อ พร้อมอธิบายข้อดีเชิงโครงสร้างระบบเมื่อเทียบกับการเขียนโค้ด Java ตรงๆ ใน JSP

ถัดมา เราจะเริ่มต้นด้วยการสร้าง JavaBean class แบบง่ายๆ โดยใช้ properties ที่มีทั้ง private fields และ public getter/setter methods ซึ่งจะทำหน้าที่เป็น interface สำหรับ JSP ในการเข้าถึงข้อมูลภายใน JavaBean อย่างปลอดภัย และสอดคล้องกับหลักการ encapsulation ของ Java

เมื่อผู้อ่านสร้าง JavaBean เสร็จแล้ว บทเรียนจะอธิบายการนำมาใช้งานใน JSP ด้วยแท็ก <jsp:useBean> ซึ่งทำหน้าที่สร้างหรือเข้าถึง instance ของ JavaBean ตาม scope ที่กำหนด (page, request, session หรือ application) นอกจากนี้ยังแสดงวิธีตั้งค่าข้อมูลให้กับ properties ด้วยแท็ก <jsp:setProperty> ซึ่งสามารถรับค่าจาก request parameters ได้โดยอัตโนมัติ

ในขั้นตอนสุดท้าย ผู้อ่านจะได้เรียนรู้การดึงค่าข้อมูลจาก JavaBean ไปแสดงผลในหน้า JSP ด้วยแท็ก <jsp:getProperty> ซึ่งช่วยให้สามารถแสดงข้อมูลจาก backend ได้โดยไม่ต้องเขียนโค้ด Java ภายใน HTML และเพื่อให้เห็นภาพรวมของการนำ JavaBeans ไปใช้ในระดับที่ดีขึ้น บทนี้ยังกล่าวถึงแนวคิดของการแยก business logic ออกจาก JSP โดยย้ายส่วนประมวลผลหลักไปไว้ใน JavaBeans ซึ่งช่วยลดภาระของหน้า JSP ให้ทำหน้าที่เพียงการแสดงผลเท่านั้น

เมื่อจบบทนี้ ผู้อ่านจะสามารถสร้างและใช้งาน JavaBeans กับ JSP ได้อย่างคล่องแคล่ว เข้าใจรูปแบบการเชื่อมโยงข้อมูลจากผู้ใช้สู่ระบบ และสามารถออกแบบระบบที่มีโครงสร้างสะอาดและยืดหยุ่น โดยใช้เทคนิคที่เป็นมาตรฐานระดับมืออาชีพในการพัฒนาเว็บด้วย Java.

การใช้ JavaBeans กับ JSP

---

1. ความหมายและประโยชน์ของ JavaBeans

- **JavaBeans** คือ คลาส Java ที่ถูกออกแบบตามมาตรฐานหนึ่ง เพื่อให้สามารถนำมาใช้ใน JSP และ Java EE ได้ง่าย
- มีลักษณะสำคัญคือ
  - มี **properties** (ตัวแปรข้อมูล) ที่เข้าถึงได้ผ่าน **getter/setter methods**
  - มี **constructor แบบไม่มีพารามิเตอร์ (default constructor)**
  - ต้องสามารถ **serializable** (implement Serializable) ได้ (ไม่บังคับเสมอไปแต่เป็นแนวทาง)
- **ประโยชน์**
  - ช่วยแยก Business Logic ออกจาก JSP ทำให้โค้ดสะอาดและดูแลรักษาง่ายขึ้น
  - ใช้ในการเก็บข้อมูลสถานะ (state) ระหว่างการทำงาน
  - ช่วยจัดการข้อมูลแบบ Object-Oriented ใน JSP
  - ทำงานร่วมกับ <jsp:useBean>, <jsp:setProperty>, <jsp:getProperty> ได้สะดวก

---

2. สร้าง JavaBean Class แบบง่าย

```
package com.example.beans;

import java.io.Serializable;
```

```
public class UserBean implements Serializable {
    private String username;
    private int age;

    // Constructor แบบไม่มีพารามิเตอร์
    public UserBean() {}

    // Getter และ Setter สำหรับ property username
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }

    // Getter และ Setter สำหรับ property age
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
}
```

---

3. การใช้งาน <jsp:useBean> ใน JSP

- ใช้เพื่อสร้างหรือเรียกใช้ JavaBean ใน JSP
- JSP จะตรวจสอบก่อนว่า bean ตัวนี้มีอยู่ใน scope ที่กำหนดหรือไม่ ถ้าไม่มีจะสร้างใหม่

ตัวอย่าง

```
<jsp:useBean id="user" class="com.example.beans.UserBean" scope="session" />
```

- id: ชื่ออ้างอิงตัวแปร JavaBean ใน JSP
- class: ชื่อคลาส JavaBean แบบเต็ม (package + class)
- scope: ขอบเขตการเก็บ JavaBean (page, request, session, application)

---

4. การกำหนด Properties ผ่าน <jsp:setProperty>

- ใช้กำหนดค่าให้ property ของ JavaBean
- มีรูปแบบหลัก ๆ คือ
    - กำหนดค่าแบบเจาะจง property และค่า
    - กำหนดค่าจาก parameter ของ request อัตโนมัติ

ตัวอย่างตั้งค่าค่าแบบเจาะจง

```
<jsp:setProperty name="user" property="username" value="chouvalit" />
<jsp:setProperty name="user" property="age" value="30" />
```

ตัวอย่างตั้งค่าจาก request parameter (matching ชื่อ property)

```
<jsp:setProperty name="user" property="*" />
```

- JSP จะอ่านค่าจาก request parameter ชื่อเดียวกับ property ของ JavaBean แล้วตั้งค่าให้โดยอัตโนมัติ

---

5. การดึงข้อมูล Properties ผ่าน <jsp:getProperty>

- ใช้แสดงค่าของ property ใน JavaBean

ตัวอย่าง

```
<p>Username: <jsp:getProperty name="user" property="username" /></p>
<p>Age: <jsp:getProperty name="user" property="age" /></p>
```

---

6. การแยก Business Logic ออกจาก JSP เพื่อความสะอาดของโค้ด

- การนำ JavaBeans มาใช้ช่วยให้ JSP มีหน้าที่แค่แสดงผล (View)
- Logic เช่น การคำนวณ ตรวจสอบข้อมูล จะย้ายไปอยู่ใน JavaBean หรือใน Servlet/Controller
- ส่ง JavaBean ที่ผ่านการประมวลผลแล้วมาให้ JSP แสดงผล
- ช่วยให้โค้ด JSP อ่านง่าย ดูแลรักษาง่าย และแยกหน้าที่ตามหลัก MVC

ตัวอย่างโปรเจกต์ง่าย ๆ

6.1 UserBean.java

```
package com.example.beans;

public class UserBean implements java.io.Serializable {
    private String username;
    private int age;

    public UserBean() {}
```

```
    public String getUsername() { return username; }
    public void setUsername(String username) { this.username = username; }

    public int getAge() { return age; }
    public void setAge(int age) { this.age = age; }
}
```

6.2 JSP - input.jsp

```
<form action="result.jsp" method="post">
    Username: <input type="text" name="username" /><br/>
    Age: <input type="text" name="age" /><br/>
    <input type="submit" value="Submit" />
</form>
```

6.3 JSP - result.jsp

```
<jsp:useBean id="user" class="com.example.beans.UserBean" scope="request" />
<jsp:setProperty name="user" property="*" />

<p>Username: <jsp:getProperty name="user" property="username" /></p>
<p>Age: <jsp:getProperty name="user" property="age" /></p>
```

สรุป

หัวข้อ	คำอธิบาย
JavaBeans คืออะไร	คลาส Java ที่มี getter/setter ตามมาตรฐาน ใช้เก็บข้อมูลและ logic
การสร้าง JavaBean	ต้องมี constructor ไม่มีพารามิเตอร์, getter/setter
<jsp:useBean>	สร้างหรือเรียกใช้ JavaBean ใน JSP
<jsp:setProperty>	กำหนดค่าคุณสมบัติ (property) ของ JavaBean
<jsp:getProperty>	ดึงค่าคุณสมบัติจาก JavaBean มาแสดง
ประโยชน์ของ JavaBeans	แยก Logic ออกจาก JSP ให้โค้ดสะอาดและดูแลง่าย

รายละเอียดเชิงลึก: การใช้ JavaBeans กับ JSP

---

1. JavaBeans คืออะไร? — แนวคิดเชิงลึก

- **JavaBeans** คือรูปแบบ (design pattern) ของ Java class ที่กำหนดมาตรฐานเพื่อรองรับการทำงานกับ Component-based frameworks เช่น JSP, JSF, IDE ต่าง ๆ
- JavaBeans ต้องมี
    - **Property**: ตัวแปรข้อมูลของคลาส ซึ่งสามารถเข้าถึงได้ผ่าน method ชื่อว่า **getter** และ **setter** ตามกฎ naming convention
    - **Default Constructor**: ต้องมี constructor แบบไม่มีพารามิเตอร์ เพื่อให้ framework (เช่น JSP) สามารถสร้าง instance ได้ง่าย
    - **Serializable**: ส่วนใหญ่ JavaBeans ควร implements java.io.Serializable เพื่อรองรับการเก็บสถานะ (state persistence) หรือการส่งผ่านเครือข่าย
- **Property Naming Convention**
    - Property ชื่อ foo ต้องมี method getFoo() และ setFoo() (ถ้าเป็น boolean ก็อาจใช้ isFoo())
    - ช่วยให้ JSP และ tool ต่าง ๆ สามารถเข้าถึงและจัดการ Property ได้โดยอัตโนมัติ

---

2. ทำไมต้องใช้ JavaBeans ใน JSP?

- JSP ออกแบบมาให้แยก **Presentation Layer** กับ **Business Logic** ออกจากกัน
- JavaBeans ช่วยเก็บข้อมูลสถานะและทำงานทางธุรกิจได้ (เช่นคำนวณ หรือดึงข้อมูลจากฐานข้อมูล)
- JSP จะมีหน้าที่แค่แสดงข้อมูลที่ได้จาก JavaBeans เท่านั้น (Separation of Concerns)
- JavaBeans สามารถนำกลับมาใช้ซ้ำ (Reusable) และทำ Unit Testing ได้ง่าย
- IDE และ JSP Container ใช้ JavaBeans เพื่อจัดการกับ data binding, form data mapping อัตโนมัติ

---

3. การประกาศและจัดการ JavaBean ใน JSP — ความหมายเชิงลึกของ <jsp:useBean>

<jsp:useBean id="user" class="com.example.UserBean" scope="session" />

- JSP Container จะทำงานดังนี้
    1. ตรวจสอบว่ามี Bean ชื่อ user อยู่ใน scope ที่ระบุหรือไม่ (session ในที่นี้)
    2. ถ้ามีอยู่แล้ว ใช้ตัวนั้น (ไม่สร้างใหม่)
    3. ถ้าไม่มี สร้าง instance ใหม่ของคลาส com.example.UserBean และเก็บไว้ใน scope นั้น
- scope มี 4 ระดับ คือ
    - page — Bean อยู่ในขอบเขตของ JSP page ปัจจุบัน (default)
    - request — Bean อยู่ในขอบเขตของ request ปัจจุบัน
    - session — Bean อยู่ในขอบเขตของ session ของ user นั้น ๆ

- application — Bean อยู่ในขอบเขตของแอปพลิเคชันทั้งหมด (application scope)

ข้อควรระวัง: การใช้ session หรือ application scope ต้องระวังเรื่อง thread safety และ memory management

---

4. การตั้งค่า Properties ด้วย <jsp:setProperty>

4.1 แบบเจาะจง Property

<jsp:setProperty name="user" property="username" value="chouvalit" />

- กำหนดค่า "chouvalit" ให้กับ property username ผ่าน method setUsername(String)

4.2 แบบอัตโนมัติ (Wildcard)

<jsp:setProperty name="user" property="*" />

- JSP จะอ่านค่าจาก **request parameter** โดยอัตโนมัติ
- ถ้าพบ request parameter ที่ชื่อเดียวกับ property ของ Bean เช่น username, age จะเรียก setter ตามนั้นโดยอัตโนมัติ
- ช่วยลดโค้ดกำหนดค่าทีละ property และสะดวกกับฟอร์ม HTML ที่มีหลาย field

---

5. การดึงข้อมูล Properties ผ่าน <jsp:getProperty>

<jsp:getProperty name="user" property="username" />

- เรียกใช้ getter method เช่น getUsername() เพื่อดึงค่า property และแสดงผลใน JSP
- JSP จะเปลี่ยน <jsp:getProperty> เป็น out.print(bean.getProperty()) ใน Servlet ที่สร้างขึ้น

---

6. การแยก Business Logic ออกจาก JSP — แนวทางที่ลึกซึ้ง

- JSP ควรใช้ JavaBeans สำหรับ
  - การเก็บข้อมูล (data model)
  - การประมวลผลแบบธุรกิจ เช่น ตรวจสอบ, คำนวณ, แปลงข้อมูล
- JSP ควรแสดงผลอย่างเดียว (presentation) ไม่ควรมี business logic หรือ database access
- Business logic ที่ซับซ้อนควรเขียนใน **Servlet**, **EJB**, หรือ **Service class** และส่งผลลัพธ์ผ่าน JavaBeans มายัง JSP
- ตัวอย่างเช่น ใน Servlet
- UserBean user = new UserBean();
- user.setUsername(request.getParameter("username"));
- user.setAge(Integer.parseInt(request.getParameter("age")));
- // ทำงาน business logic ที่นี่
- request.setAttribute("user", user);
- request.getRequestDispatcher("result.jsp").forward(request, response);

- ใน JSP ใช้ <jsp:useBean id="user" scope="request" /> เพื่อดึงข้อมูลมาแสดง

---

7. Scope ของ JavaBeans และผลกระทบต่อการทำงาน

Scope	ระยะเวลาการอยู่ของ Bean	การใช้งานในกรณี	ข้อควรระวัง
page	ตลอดอายุการประมวลผล JSP page ปัจจุบัน	ข้อมูลที่ใช้แค่ใน JSP หน้านั้น ๆ	ข้อมูลไม่ถูกเก็บข้าม request
request	ตลอดอายุของ HTTP request นั้น ๆ	ส่งข้อมูลระหว่าง JSP/Servlet	ป้องกันข้อมูลซ้ำซ้อน
session	ตลอดอายุการใช้งานของ session (ผู้ใช้คนเดียวกัน)	เก็บข้อมูลผู้ใช้ เช่น login	ต้องจัดการ memory, thread safety
application	ตลอดอายุของเว็บแอปพลิเคชัน (ทุกผู้ใช้แชร์กัน)	เก็บข้อมูลร่วมเช่น config app	ระวัง thread safety และ memory leaks

---

8. เทคนิคและแนวทางปฏิบัติที่ดี

- **ใช้ scope ให้เหมาะสม** ตามวัตถุประสงค์ของข้อมูลและช่วงเวลาที่ต้องการเก็บ
- **หลีกเลี่ยงการเขียน logic มากใน JSP** ให้ JSP ทำหน้าที่แค่แสดงผล
- **ตรวจสอบชนิดข้อมูล (data validation)** ก่อนตั้งค่าค่า property ของ JavaBean
- **ใช้ try-catch ใน JavaBean หรือ Servlet** เพื่อจัดการข้อผิดพลาด
- **ใช้ jsp:setProperty แบบ wildcard ในกรณีฟอร์มง่าย** แต่ถ้าฟอร์มซับซ้อนควรกำหนด property ทีละตัวเพื่อควบคุมและ validate
- **ใช้ MVC pattern** โดยให้ Servlet/Controller จัดการ JavaBean และส่งข้อมูลไป JSP แสดงผล

---

9. การ Debugging และการตรวจสอบ

- ใช้ System.out.println() หรือ logging ใน JavaBean เพื่อดูค่า property หรือ flow การทำงาน
- ตรวจสอบว่า JavaBean ถูกสร้างและเก็บใน scope ที่ถูกต้อง
- ทดสอบการเข้าถึง property โดย jsp:getProperty ว่าคืนค่าถูกต้องหรือไม่
- ระวังการตั้งค่าซ้ำซ้อนใน scope เดียวกันที่อาจทำให้ข้อมูลผิดพลาด

สรุป

หัวข้อสำคัญ	รายละเอียดเชิงลึก
JavaBeans คือ	คลาส Java ตามมาตรฐาน มี properties พร้อม getter/setter, default constructor

หัวข้อสำคัญ	รายละเอียดเชิงลึก
การใช้งานใน JSP	ใช้ <jsp:useBean>, <jsp:setProperty>, <jsp:getProperty> จัดการ JavaBeans ใน JSP
Scope ของ JavaBean	page, request, session, application มีผลต่ออายุและการแชร์ข้อมูล
ประโยชน์หลัก	แยก logic กับ presentation, code reuse, data encapsulation
แนวทางปฏิบัติที่ดี	ใช้ MVC, แยก logic ออกจาก JSP, ใช้ scope ให้เหมาะสม, validate data

การสร้าง JavaBean Class แบบง่าย (Properties, Getter/Setter)

1. JavaBean คืออะไรในเชิงเทคนิค?

- เป็น **Java class** ที่ออกแบบให้มี
  - **Properties** หรือ ตัวแปรข้อมูลที่ต้องการเก็บข้อมูล
  - **Getter methods** (getPropertyName()) เพื่ออ่านค่า Properties
  - **Setter methods** (setPropertyName(value)) เพื่อกำหนดค่า Properties
  - **Constructor แบบไม่มีพารามิเตอร์** (Default constructor) เพื่อให้ JSP หรือ framework สามารถสร้าง object ได้ง่าย
- เป็นไปตามมาตรฐาน JavaBeans Specification เพื่อให้ component ต่าง ๆ (เช่น JSP) สามารถทำงานร่วมกันได้

2. ตัวอย่างสร้าง JavaBean Class แบบง่าย

สมมติเราจะสร้าง JavaBean สำหรับเก็บข้อมูลผู้ใช้ (UserBean) มี 2 property คือ username กับ age

UserBean.java

```
package com.example.beans;

import java.io.Serializable;

public class UserBean implements Serializable {
    // Properties
    private String username;
    private int age;
```

```
    // Constructor แบบไม่มีพารามิเตอร์ (สำคัญมาก)
    public UserBean() {
    }

    // Getter สำหรับ username
    public String getUsername() {
        return username;
    }

    // Setter สำหรับ username
    public void setUsername(String username) {
        this.username = username;
    }

    // Getter สำหรับ age
    public int getAge() {
        return age;
    }

    // Setter สำหรับ age
    public void setAge(int age) {
        this.age = age;
    }
}
```

---

3. อธิบายโครงสร้างโค้ด

ส่วน	รายละเอียด
package com.example.beans;	กำหนดแพ็กเกจของคลาส (จัดการโครงสร้างโปรเจกต์)
implements Serializable	ทำให้ Bean สามารถถูกเก็บสถานะหรือส่งผ่านเครือข่ายได้
private String username;	ตัวแปร Property แบบ private เพื่อป้องกันการเข้าถึงโดยตรง
public String getUsername()	Getter method ใช้สำหรับอ่านค่าของ property username
public void setUsername(String	Setter method ใช้สำหรับกำหนดค่าของ property

ส่วน	รายละเอียด
username)	username
public UserBean() {}	Constructor แบบไม่มีพารามิเตอร์ จำเป็นต้องมีสำหรับ JavaBean

---

4. ทำไมต้องมี Getter และ Setter?

- **Encapsulation**: ซ่อนตัวแปรภายใน class และควบคุมการเข้าถึงผ่าน method
- JSP และ framework ใช้ naming convention ของ getter/setter ในการเชื่อมต่อข้อมูลกับ JavaBean (เช่น <jsp:setProperty> จะเรียก setter อัตโนมัติ)
- ช่วยให้สามารถตรวจสอบข้อมูลหรือปรับแต่งค่าก่อนเก็บได้ เช่น validate ใน setter

---

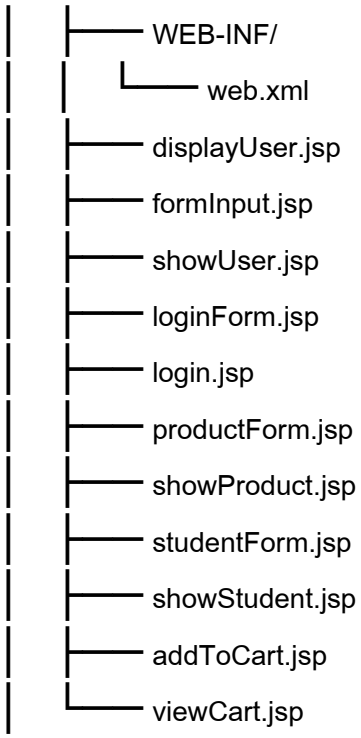
5. ข้อแนะนำ

- ชื่อ property ต้องสอดคล้องกับชื่อ getter/setter (ตามมาตรฐาน JavaBeans)
- หลีกเลี่ยงการเขียน logic ซับซ้อนใน getter/setter ควรให้ทำหน้าที่แค่อ่านและเขียนข้อมูล
- ต้องมี constructor ไม่มีพารามิเตอร์เสมอ เพื่อให้ container หรือ JSP สร้าง instance ได้ง่าย

โครงสร้างโปรเจกต์ JSP พร้อม **ไฟล์โค้ดแบบเต็ม** และ **คำอธิบายโครงสร้าง-โค้ด-ผลการรัน** สำหรับ 3 โปรแกรมพื้นฐาน และ 3 โปรแกรมแนวประยุกต์ ที่ใช้ JavaBean แบบง่าย (Properties + Getter/Setter) ดังนี้ครับ

โครงสร้างโปรเจกต์ JSP ตัวอย่าง

```
YourProject/
│
├── src/
│   └── com/
│       └── example/
│           └── beans/
│               ├── UserBean.java
│               ├── ProductBean.java
│               ├── StudentBean.java
│               └── CartItemBean.java
│
├── WebContent/
```



โปรแกรมพื้นฐาน 3 โปรแกรม

1. โปรแกรมแสดงข้อมูล User ผ่าน JSP

1.1 ไฟล์ JavaBean: src/com/example/beans/UserBean.java

```
package com.example.beans;

import java.io.Serializable;

public class UserBean implements Serializable {
    private String username;
    private int age;

    public UserBean() {}

    public String getUsername() { return username; }
    public void setUsername(String username) { this.username = username; }

    public int getAge() { return age; }
    public void setAge(int age) { this.age = age; }
```

```
}
```

1.2 ไฟล์ JSP: WebContent/displayUser.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-
8"%>
<jsp:useBean id="user" class="com.example.beans.UserBean" scope="page" />
<jsp:setProperty name="user" property="username" value="chouvalit" />
<jsp:setProperty name="user" property="age" value="28" />

<html>
<head><title>User Info</title></head>
<body>
<h2>User Information</h2>
<p>Username: <jsp:getProperty name="user" property="username" /></p>
<p>Age: <jsp:getProperty name="user" property="age" /></p>
</body>
</html>
```

คำอธิบาย

- สร้าง UserBean ที่เก็บ username และ age
- JSP สร้าง Bean และกำหนดค่า username, age
- แสดงค่าด้วย <jsp:getProperty>

ผลลัพธ์ (แสดงใน Browser)

User Information

Username: chouvalit

Age: 28

---

2. โปรแกรมรับข้อมูลจาก Form HTML (POST) แล้วเก็บใน JavaBean

2.1 ไฟล์ JSP: WebContent/formInput.jsp

```
<html>
<head><title>Input Form</title></head>
<body>
<form action="showUser.jsp" method="post">
    Username: <input type="text" name="username" /><br/>
    Age: <input type="text" name="age" /><br/>
    <input type="submit" value="Submit" />
```

```
</form>
</body>
</html>
```

2.2 ไฟล์ JSP: WebContent/showUser.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-
8"%>
<jsp:useBean id="user" class="com.example.beans.UserBean" scope="request" />
<jsp:setProperty name="user" property="*" />

<html>
<head><title>User Info</title></head>
<body>
<h2>Submitted User Data</h2>
<p>Username: <jsp:getProperty name="user" property="username" /></p>
<p>Age: <jsp:getProperty name="user" property="age" /></p>
</body>
</html>
```

คำอธิบาย

- formInput.jsp เป็นฟอร์มรับ username กับ age
- เมื่อ submit ไป showUser.jsp
- showUser.jsp สร้าง UserBean และตั้งค่าจาก request parameter อัตโนมัติ (property="*")
- แสดงค่าที่รับมา

ผลลัพธ์

- กรอก username: chouvalit
- กรอก age: 30

  แสดง

Submitted User Data

Username: chouvalit

Age: 30

---

3. โปรแกรมเก็บข้อมูล User ใน Session Scope

3.1 ไฟล์ JSP: WebContent/loginForm.jsp

```
<html>
<head><title>Login Form</title></head>
```

```
<body>
<form action="login.jsp" method="post">
    Username: <input type="text" name="username" /><br/>
    Age: <input type="text" name="age" /><br/>
    <input type="submit" value="Login" />
</form>
</body>
</html>
```

3.2 ไฟล์ JSP: WebContent/login.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<jsp:useBean id="user" class="com.example.beans.UserBean" scope="session" />
<jsp:setProperty name="user" property="*" />

<html>
<head><title>Welcome</title></head>
<body>
<h2>Welcome, <jsp:getProperty name="user" property="username" />!</h2>
<p>Your age is: <jsp:getProperty name="user" property="age" /></p>
</body>
</html>
```

คำอธิบาย

- เก็บ UserBean ใน session scope เพื่อให้ข้อมูลคงอยู่ข้ามหลายหน้า
- loginForm.jsp รับข้อมูลผู้ใช้
- login.jsp สร้าง/อัพเดท UserBean ใน session พร้อมแสดงข้อมูล

ผลลัพธ์

เมื่อกรอก username = chouvalit และ age = 30 แสดง

Welcome, chouvalit!

Your age is: 30

โปรแกรมแนวประยุกต์ 3 โปรแกรม

---

4. JavaBean คำนวณราคาสินค้าในตัว Bean

4.1 JavaBean: src/com/example/beans/ProductBean.java

```
package com.example.beans;

import java.io.Serializable;

public class ProductBean implements Serializable {
    private String name;
    private double price;
    private int quantity;

    public ProductBean() {}

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    public double getPrice() { return price; }
    public void setPrice(double price) { this.price = price; }

    public int getQuantity() { return quantity; }
    public void setQuantity(int quantity) { this.quantity = quantity; }

    // Method คำนวณราคารวม
    public double getTotal() {
        return price * quantity;
    }
}
```

4.2 JSP: WebContent/productForm.jsp

```
<html>
<head><title>Product Input</title></head>
<body>
<form action="showProduct.jsp" method="post">
    Product Name: <input type="text" name="name" /><br/>
    Price: <input type="text" name="price" /><br/>
    Quantity: <input type="text" name="quantity" /><br/>
    <input type="submit" value="Submit" />
```

```
</form>
</body>
</html>
```

4.3 JSP: WebContent/showProduct.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<jsp:useBean id="product" class="com.example.beans.ProductBean" scope="request" />
<jsp:setProperty name="product" property="*" />

<html>
<head><title>Product Detail</title></head>
<body>
<h2>Product Details</h2>
<p>Name: <jsp:getProperty name="product" property="name" /></p>
<p>Price: <jsp:getProperty name="product" property="price" /></p>
<p>Quantity: <jsp:getProperty name="product" property="quantity" /></p>
<p>Total Price: <%= product.getTotal() %></p>
</body>
</html>
```

คำอธิบาย

- Bean มี method getTotal() คำนวณราคารวม
- JSP แสดงข้อมูลและใช้ Expression <%= %> เรียก method ภายใน Bean

ผลลัพธ์

เมื่อกรอกชื่อสินค้า: "Notebook", ราคา: 1500, จำนวน: 2

แสดง

Product Details

Name: Notebook

Price: 1500.0

Quantity: 2

Total Price: 3000.0

---

5. JavaBean ตรวจสอบข้อมูลใน Setter (Validate)

5.1 JavaBean: src/com/example/beans/StudentBean.java

```
package com.example.beans;
```

```
import java.io.Serializable;

public class StudentBean implements Serializable {
    private String name;
    private int score;

    public StudentBean() {}

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    public int getScore() { return score; }
    public void setScore(int score) {
        if(score < 0) {
            this.score = 0;
        } else if(score > 100) {
            this.score = 100;
        } else {
            this.score = score;
        }
    }
}
```

5.2 JSP: WebContent/studentForm.jsp

```
<html>
<head><title>Student Form</title></head>
<body>
<form action="showStudent.jsp" method="post">
    Name: <input type="text" name="name" /><br/>
    Score: <input type="text" name="score" /><br/>
    <input type="submit" value="Submit" />
</form>
</body>
</html>
```

5.3 JSP: WebContent/showStudent.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<jsp:useBean id="student" class="com.example.beans.StudentBean" scope="request" />
<jsp:setProperty name="student" property="*" />

<html>
<head><title>Student Info</title></head>
<body>
<h2>Student Details</h2>
<p>Name: <jsp:getProperty name="student" property="name" /></p>
<p>Score: <jsp:getProperty name="student" property="score" /></p>
</body>
</html>
```

คำอธิบาย

- Setter ของ score ตรวจสอบค่าให้ไม่ต่ำกว่า 0 และไม่เกิน 100
- JSP แสดงคะแนนที่ผ่านการตรวจสอบแล้ว

---

6. JavaBean ใช้กับ Session เพื่อเก็บข้อมูลตะกร้าสินค้า

6.1 JavaBean: src/com/example/beans/CartItemBean.java

```
package com.example.beans;

import java.io.Serializable;

public class CartItemBean implements Serializable {
    private String productName;
    private int quantity;

    public CartItemBean() {}

    public String getProductName() { return productName; }
    public void setProductName(String productName) { this.productName = productName; }

    public int getQuantity() { return quantity; }
```

```
    public void setQuantity(int quantity) { this.quantity = quantity; }
}
```

6.2 JSP: WebContent/addToCart.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<jsp:useBean id="cartItem" class="com.example.beans.CartItemBean" scope="session" />
<jsp:setProperty name="cartItem" property="*" />

<html>
<head><title>Cart Updated</title></head>
<body>
<h2>Cart Updated</h2>
<p>Product: <jsp:getProperty name="cartItem" property="productName" /></p>
<p>Quantity: <jsp:getProperty name="cartItem" property="quantity" /></p>
<a href="viewCart.jsp">View Cart</a>
</body>
</html>
```

6.3 JSP: WebContent/viewCart.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<jsp:useBean id="cartItem" class="com.example.beans.CartItemBean" scope="session" />

<html>
<head><title>View Cart</title></head>
<body>
<h2>Your Cart</h2>
<p>Product: <jsp:getProperty name="cartItem" property="productName" /></p>
<p>Quantity: <jsp:getProperty name="cartItem" property="quantity" /></p>
</body>
</html>
```

คำอธิบาย

- เก็บข้อมูลสินค้าใน session Bean ข้ามหลายหน้าได้
- addToCart.jsp รับข้อมูลจากฟอร์มและเก็บใน session Bean
- viewCart.jsp แสดงข้อมูลในตะกร้า

ยกตัวอย่างการตั้งค่า **web.xml** รวมถึงแนวทางการเตรียมเครื่องมือ (library) และการตั้งค่า Tomcat เพื่อให้โปรเจกต์ JSP + JavaBeans ที่ผมส่งไปก่อนหน้านี้ สามารถรันได้อย่างสมบูรณ์บน Apache Tomcat

1. ตัวอย่างไฟล์ web.xml (Deployment Descriptor)

ไฟล์นี้อยู่ในโฟลเดอร์ WEB-INF/web.xml ของโปรเจกต์ JSP และทำหน้าที่กำหนด servlet, JSP mapping รวมถึง config ต่าง ๆ ของเว็บแอป

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
                             https://jakarta.ee/xml/ns/jakartaee/web-app_5_0.xsd"
         version="5.0">

    <!-- Welcome file (หน้าแรก) -->
    <welcome-file-list>
        <welcome-file>formInput.jsp</welcome-file> <!-- ตัวอย่างหน้าเริ่มต้น -->
    </welcome-file-list>

    <!-- ตัวอย่าง Servlet mapping ถ้าจำเป็น (โปรเจกต์ JSP เราอาจไม่ต้องใช้) -->
    <!--
    <servlet>
        <servlet-name>ExampleServlet</servlet-name>
        <servlet-class>com.example.servlets.ExampleServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>ExampleServlet</servlet-name>
        <url-pattern>/example</url-pattern>
    </servlet-mapping>
    -->

</web-app>
```

2. การเตรียม Library (JARs)

- สำหรับ JSP + JavaBeans พื้นฐาน **ไม่จำเป็นต้องเพิ่มไลบรารีเพิ่มเติม** เพราะ Tomcat จะจัดการ runtime libraries ให้เอง (เช่น JSP API, Servlet API)
- แต่ถ้าต้องการใช้ library เสริม เช่น JSTL หรือ framework อื่น ๆ สามารถดาวน์โหลด JAR มาใส่ใน

YourProject/WebContent/WEB-INF/lib/

- ตัวอย่าง JAR สำคัญ:
  - javax.servlet-api.jar (แต่ Tomcat มักมีให้ใน runtime แล้ว)
  - javax.servlet.jsp-api.jar
  - jstl.jar (สำหรับ JSTL tag library)

---

3. การติดตั้งและตั้งค่า Apache Tomcat

3.1 ดาวน์โหลด Tomcat

- ไปที่ Apache Tomcat Official Download (หรือเวอร์ชัน 10+ ตามต้องการ)
- ดาวน์โหลดไฟล์ zip หรือ tar.gz ตาม OS ของคุณ
- แตกไฟล์ไปยังโฟลเดอร์ เช่น C:\apache-tomcat-9.0.XX หรือ /usr/local/apache-tomcat-9.0.XX

3.2 ตั้งค่า Environment Variables (Windows ตัวอย่าง)

- ตั้งค่า JAVA_HOME ให้ชี้ไปยัง JDK ที่ติดตั้งไว้
- เพิ่ม %JAVA_HOME%\bin ลงใน PATH

3.3 การ Deploy Project JSP ไปยัง Tomcat

วิธีง่ายสุด: Copy โฟลเดอร์โปรเจกต์ (ไฟล์ JSP + WEB-INF) เข้าไปใน

apache-tomcat/webapps/YourProjectName/

- จากนั้นรัน Tomcat (เปิด cmd แล้วรัน bin\startup.bat หรือ bin/startup.sh บน Linux/Mac)
- เปิดเบราเซอร์เข้า URL:

http://localhost:8080/YourProjectName/

โดยหน้าแรกจะเป็นไฟล์ที่กำหนดใน welcome-file-list เช่น formInput.jsp

---

4. การตั้งค่าใน IDE ให้เชื่อมกับ Tomcat

Eclipse

- ติดตั้ง Tomcat Server ผ่าน Eclipse Server Runtime
- เพิ่ม Tomcat Server ใน Server Tab
- สร้าง Dynamic Web Project เลือก Target Runtime เป็น Tomcat

- รัน Project ผ่าน Eclipse จะ deploy อัตโนมัติ

IntelliJ IDEA

- สร้าง Project แบบ Java Enterprise Web Application
- ตั้งค่า Tomcat Server ใน Run/Debug Configurations
- รันผ่าน IntelliJ เพื่อ deploy และรันบน Tomcat

สรุป

ส่วนประกอบ	การตั้งค่า/ขั้นตอน
web.xml	กำหนด welcome file และ servlet mapping (ถ้ามี)
Libraries	ไม่ต้องเพิ่มถ้าใช้ Tomcat มาตรฐาน
Tomcat	ดาวน์โหลด ติดตั้ง ตั้งค่า JAVA_HOME และรัน
Deploy	Copy โฟลเดอร์โปรเจกต์เข้า webapps/ หรือผ่าน IDE
รันเว็บ	เปิด URL: http://localhost:8080/YourProjectName/

การใช้งาน <jsp:useBean> เพื่อสร้าง/เรียกใช้ JavaBean ใน JSP

การใช้งาน <jsp:useBean>

<jsp:useBean> เป็น JSP action ที่ใช้สำหรับ **สร้าง (instantiate) JavaBean หรือเรียกใช้ JavaBean ที่มีอยู่แล้วใน scope** (page, request, session, application) โดย JSP จะเช็คก่อนว่า JavaBean ตัวนั้นมีอยู่หรือยัง ถ้าไม่มีก็จะสร้างใหม่ให้อัตโนมัติ

โครงสร้างคำสั่งพื้นฐาน

<jsp:useBean id="beanName" class="fully.qualified.BeanClassName" scope="scopeName" />

- id = ชื่อที่ใช้เรียก JavaBean ใน JSP (เหมือนตัวแปร)
- class = ชื่อคลาส JavaBean แบบเต็ม เช่น com.example.beans.UserBean
- scope = ขอบเขตของ Bean มีได้ 4 แบบ คือ
    - page (default) — ใช้ได้แค่ใน JSP หน้านั้น
    - request — ใช้ได้ภายใน request เดียวกัน
    - session — เก็บไว้ใน session ของผู้ใช้ข้ามหลาย request
    - application — เก็บไว้ในระดับ application (ทุกคนใช้ร่วมกัน)

ตัวอย่างการใช้ <jsp:useBean>

ตัวอย่าง 1: สร้าง Bean ใหม่ใน Page Scope

```
<%@ page language="java" contentType="text/html; charset=UTF-8" %>
<jsp:useBean id="user" class="com.example.beans.UserBean" scope="page" />
<jsp:setProperty name="user" property="username" value="chouvalit" />
<jsp:setProperty name="user" property="age" value="28" />

<html>
<head><title>User Info</title></head>
<body>
<p>Username: <jsp:getProperty name="user" property="username" /></p>
<p>Age: <jsp:getProperty name="user" property="age" /></p>
</body>
</html>
```

อธิบาย:

- <jsp:useBean> สร้าง UserBean ขึ้นใหม่ใน **page scope** (สำหรับหน้านี้เท่านั้น)
- ใช้ <jsp:setProperty> กำหนดค่าคุณสมบัติ (properties) ของ Bean
- แสดงค่าด้วย <jsp:getProperty>

ตัวอย่าง 2: ใช้ Bean ที่ถูกสร้างไว้ใน Session Scope

(1) หน้าแรก สร้าง Bean และเก็บใน session

```
<jsp:useBean id="user" class="com.example.beans.UserBean" scope="session" />
<jsp:setProperty name="user" property="username" value="chouvalit" />
<jsp:setProperty name="user" property="age" value="28" />
<p>User created and stored in session.</p>
```

(2) หน้าอื่นเรียกใช้ Bean เดิมจาก session

```
<jsp:useBean id="user" class="com.example.beans.UserBean" scope="session" />
<p>Username from session: <jsp:getProperty name="user" property="username" /></p>
<p>Age from session: <jsp:getProperty name="user" property="age" /></p>
```

อธิบาย:

- ครั้งแรกที่ใช้ <jsp:useBean> กับ scope="session" จะสร้าง Bean ใหม่และเก็บใน session
- ครั้งถัดไป JSP จะไม่สร้างใหม่ แต่เรียกใช้ Bean เดิมที่เก็บใน session
- ทำให้ข้อมูลคงอยู่ข้ามหลายหน้า

ตัวอย่าง 3: ใช้ <jsp:useBean> ร่วมกับ <jsp:setProperty property="*">

```
<!-- form.jsp -->
<form action="process.jsp" method="post">
    Username: <input type="text" name="username" /><br/>
    Age: <input type="text" name="age" /><br/>
    <input type="submit" value="Submit" />
</form>
<!-- process.jsp -->
<jsp:useBean id="user" class="com.example.beans.UserBean" scope="request" />
<jsp:setProperty name="user" property="*" />

Username: <jsp:getProperty name="user" property="username" /><br/>
Age: <jsp:getProperty name="user" property="age" />
```

อธิบาย:

- ฟอร์มส่งข้อมูลแบบ POST
- JSP ใช้ <jsp:setProperty name="user" property="*"/> เพื่ออ่านค่าจาก request.getParameter() ตามชื่อฟิลด์อัตโนมัติ แล้วตั้งเป็น property ของ Bean
- สะดวกไม่ต้องเขียนตั้งค่าแต่ละตัวทีละบรรทัด

สรุปข้อควรรู้ <jsp:useBean>

จุดสำคัญ	คำอธิบาย
สร้าง Bean	สร้าง JavaBean ขึ้นถ้ายังไม่มีใน scope นั้น ๆ
ใช้ Bean ที่มีอยู่	ถ้ามี Bean ใน scope เดิมแล้ว จะไม่สร้างใหม่
กำหนด Scope	page, request, session, application
ตั้งค่า Property	ใช้ <jsp:setProperty> กำหนดค่าจาก JSP หรือ request
ดึงข้อมูล Property	ใช้ <jsp:getProperty> แสดงค่าจาก Bean

ต่อไปนี้เป็นตัวอย่างโปรแกรม JSP แบบเต็มไฟล์ รวมโครงสร้างโปรเจกต์ พร้อมคำอธิบายโค้ดและผลการรัน ทั้งแบบพื้นฐาน 3 โปรแกรม และแบบประยุกต์ 3 โปรแกรม ที่ใช้ <jsp:useBean> เพื่อสร้างและเรียกใช้ JavaBean ใน JSP

โครงสร้างโปรเจกต์ตัวอย่าง

```
YourProject/
 ├─ WebContent/
```

```
│   ├── WEB-INF/
│   │   └── web.xml
│   ├── basic1.jsp
│   ├── basic2.jsp
│   ├── basic3.jsp
│   ├── app1.jsp
│   ├── app2.jsp
│   └── app3.jsp
└── src/
    └── com/example/beans/
        ├── UserBean.java
        ├── ProductBean.java
        └── CalculatorBean.java
```

ส่วนที่ 1: JavaBeans ตัวอย่าง (ไฟล์ Java)

```
// src/com/example/beans/UserBean.java
package com.example.beans;

public class UserBean {
    private String username;
    private int age;

    public UserBean() { }

    public String getUsername() { return username; }
    public void setUsername(String username) { this.username = username; }

    public int getAge() { return age; }
    public void setAge(int age) { this.age = age; }
}
// src/com/example/beans/ProductBean.java
package com.example.beans;
```

```
public class ProductBean {
    private String name;
    private double price;

    public ProductBean() { }

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    public double getPrice() { return price; }
    public void setPrice(double price) { this.price = price; }
}
// src/com/example/beans/CalculatorBean.java
package com.example.beans;

public class CalculatorBean {
    private int a;
    private int b;
    private int result;

    public CalculatorBean() { }

    public int getA() { return a; }
    public void setA(int a) { this.a = a; }

    public int getB() { return b; }
    public void setB(int b) { this.b = b; }

    public int getResult() { return result; }
    public void setResult(int result) { this.result = result; }

    public void add() {
        result = a + b;
    }
```

```
}
```

ส่วนที่ 2: ตัวอย่าง JSP พื้นฐาน 3 โปรแกรม

ตัวอย่าง 1: basic1.jsp — สร้าง Bean ใน page scope และกำหนด property ด้วย setProperty แบบ value ตรง ๆ

```
<%@ page contentType="text/html; charset=UTF-8" %>
<jsp:useBean id="user" class="com.example.beans.UserBean" scope="page" />
<jsp:setProperty name="user" property="username" value="chouvalit" />
<jsp:setProperty name="user" property="age" value="30" />

<html>
<head><title>User Info</title></head>
<body>
<h2>Basic1: User Info from Bean</h2>
<p>Username: <jsp:getProperty name="user" property="username" /></p>
<p>Age: <jsp:getProperty name="user" property="age" /></p>
</body>
</html>
```

ผลการรัน:

แสดงข้อความ

Username: chouvalit

Age: 30

ตัวอย่าง 2: basic2.jsp — สร้าง Bean ใน request scope และกำหนด property จาก request parameters

```
<%@ page contentType="text/html; charset=UTF-8" %>
<jsp:useBean id="user" class="com.example.beans.UserBean" scope="request" />
<jsp:setProperty name="user" property="*" />

<html>
<head><title>User Info From Form</title></head>
<body>
<h2>Basic2: User Info from Form Parameters</h2>
```

```
<p>Username: <jsp:getProperty name="user" property="username" /></p>
<p>Age: <jsp:getProperty name="user" property="age" /></p>
</body>
</html>
```

คำอธิบาย:

เวลารันไฟล์นี้ ต้องส่งข้อมูลผ่าน URL หรือ form เช่น

basic2.jsp?username=chouvalit&age=25

ผลการรัน:

Username: chouvalit

Age: 25

ตัวอย่าง 3: basic3.jsp — สร้าง Bean ใน session scope และใช้ Bean คงสถานะผู้ใช้ข้ามหลายหน้า

```
<%@ page contentType="text/html; charset=UTF-8" %>
<jsp:useBean id="user" class="com.example.beans.UserBean" scope="session" />
<jsp:setProperty name="user" property="username" value="sessionUser" />
<jsp:setProperty name="user" property="age" value="40" />

<html>
<head><title>Session User</title></head>
<body>
<h2>Basic3: User info stored in session</h2>
<p>Username: <jsp:getProperty name="user" property="username" /></p>
<p>Age: <jsp:getProperty name="user" property="age" /></p>
</body>
</html>
```

ผลการรัน:

แสดงข้อความ Username และ Age ที่ถูกเก็บใน session

ส่วนที่ 3: ตัวอย่าง JSP แนวประยุกต์ 3 โปรแกรม

ตัวอย่าง app1.jsp — รับข้อมูลฟอร์มสินค้า สร้าง ProductBean แล้วแสดงข้อมูล

สร้างฟอร์มใน productForm.jsp

```
<html>
```