Java Server Page-JSP Web Programming

(Integrative-Generative AI Edition)

Fundamentals of JSP, Syntax and Directives, Variables and Scopes iſSP, JSP with CSS and Bopes JSP, JSP with MySQLL Database, Managing Cookies, Sessions, and Local ß JavaBeans Storage

Developing CRUD-Based Web Applications with Developng AJAX in JSP, Handling AAX in JSP Developing JSP Applications Using the MVC Patern

Author
Student Price Book Center

คำนำ

ในยุคปัจจุบันที่เทคโนโลยีสารสนเทศมีบทบาทสำคัญต่อการพัฒนาระบบสารสนเทศขององค์กร เว็บไซต์และแอปพลิเคชันบนเว็บได้กลายเป็นเครื่องมือหลักในการให้บริการ ตอบสนองความต้องการของผู้ใช้งาน และสนับสนุนกระบวนการทำงานต่าง ๆ อย่างมีประสิทธิภาพ เทคโนโลยีฝั่งเซิร์ฟเวอร์จึงมีความสำคัญอย่างยิ่ง โดยเฉพาะอย่างยิ่งเทคโนโลยี **JavaServer Pages (JSP)** ซึ่งเป็นหนึ่งในเครื่องมือที่ช่วยให้การพัฒนาเว็บแอปพลิเคชันสามารถผสานโค้ด Java เข้ากับ HTML ได้อย่างยืดหยุ่น เหมาะสำหรับผู้ที่ต้องการพัฒนาเว็บแบบไดนามิกและมีความสามารถในการติดต่อกับฐานข้อมูลได้อย่างสะดวก แม้ว่าจะมีเทคโนโลยีที่ใหม่กว่าเกิดขึ้น เช่น JavaServer Faces (JSF), Spring Boot หรือ Node.js แต่ JSP ยังคงมีความสำคัญทั้งในด้านการเรียนรู้พื้นฐานของการพัฒนาเว็บด้วยภาษา Java และการใช้งานในบางระบบที่ยังพึ่งพาโครงสร้างแบบเดิม การเรียนรู้ JSP จึงยังเป็นสิ่งจำเป็นสำหรับผู้เริ่มต้นและนักพัฒนาที่ต้องการเข้าใจโครงสร้างเบื้องหลังของเว็บแอปพลิเคชันแบบดั้งเดิม รวมถึงสามารถนำองค์ความรู้ไปต่อยอดสู่เทคโนโลยีอื่นได้อย่างมีประสิทธิภาพ

หนังสือเล่มนี้เริ่มต้นจากเนื้อหาพื้นฐานที่เข้าใจง่าย อธิบายตั้งแต่โครงสร้างของภาษา HTML ที่ใช้ร่วมกับ JSP การใช้งาน CSS และ Bootstrap เพื่อจัดการรูปแบบการแสดงผล ไปจนถึงการตั้งค่าสภาพแวดล้อมในโปรแกรม NetBeans และ GlassFish Server จากนั้นจึงต่อยอดด้วยหัวข้อที่ซับซ้อนขึ้น เช่น การใช้ตัวแปรใน JSP, การติดต่อกับฐานข้อมูลด้วย JDBC, การใช้งาน JavaBeans และการพัฒนาแอปพลิเคชันแบบ CRUD ซึ่งจำเป็นต่อการจัดการข้อมูลในระบบฐานข้อมูล

นอกจากนี้ ยังมีการอธิบายถึงเทคนิคในการจัดการการทำงานระหว่างหน้าด้วย Cookies, Session และ Local Storage ซึ่งล้วนเป็นองค์ประกอบที่สำคัญของเว็บยุคใหม่ อีกทั้งยังมีการอธิบายการใช้ AJAX เพื่อการดึงข้อมูลจากเซิร์ฟเวอร์แบบไม่ต้องโหลดหน้าใหม่ ซึ่งช่วยให้เว็บแอปพลิเคชันมีความทันสมัยและตอบสนองรวดเร็ว โดยในหนังสือได้นำเสนอทั้ง AJAX แบบใช้ JavaScript ธรรมดา, การใช้ fetch API, การใช้ GSON Library และการใช้งานร่วมกับ jQuery เพื่อให้ผู้อ่านสามารถเลือกแนวทางที่เหมาะสมกับตนเอง

ในช่วงท้ายของหนังสือ ผู้อ่านจะได้เรียนรู้แนวคิดการพัฒนาเว็บแอปพลิเคชันในรูปแบบ **MVC (Model-View-Controller)** ซึ่งช่วยให้การเขียนโค้ดมีความเป็นระบบ แยกส่วนการแสดงผล องค์ประกอบทางตรรกะ และการติดต่อกับฐานข้อมูลออกจากกันอย่างชัดเจน หนังสือได้นำเสนอการใช้งาน MVC แบบเต็มรูปแบบผ่านโปรเจกต์จริง เช่น ระบบ POS ระบบจัดการพนักงาน และระบบ Login ที่มีการใช้ CSS, CRUD, AJAX และ JSP ผสมผสานกันอย่างลงตัว

จุดเด่นของหนังสือเล่มนี้อยู่ที่การนำเสนอเนื้อหาอย่างเป็นลำดับขั้น พร้อมตัวอย่างโค้ดที่ทดลองได้จริงในสภาพแวดล้อมการพัฒนาแบบ NetBeans ทั้งในแบบ Ant Style และ Maven Style เพื่อให้ผู้อ่านไม่เพียงเข้าใจทฤษฎี แต่ยังสามารถนำความรู้ไปพัฒนาระบบจริงได้อย่างมั่นใจ หนังสือเล่มนี้จึงเหมาะสำหรับนิสิต นักศึกษา หรือผู้เริ่มต้นที่ต้องการศึกษาและฝึกฝนทักษะการพัฒนาเว็บด้วย Java และ JSP อย่างครบถ้วน

ศูนย์หนังสือราคานักเรียน ได้ตั้งใจเรียบเรียงเนื้อหาโดยอ้างอิงจากเอกสารทางวิชาการ มาตรฐานสากล และแหล่งข้อมูลจากการใช้งานจริง เพื่อให้ผู้อ่านได้รับความรู้ที่ถูกต้อง เป็นระบบ และสามารถต่อยอดสู่การพัฒนาเทคโนโลยีที่ซับซ้อนมากขึ้นได้ หากมีข้อผิดพลาดหรือความไม่สมบูรณ์ในเนื้อหาใด ทางศูนย์หนังสือฯ ขอน้อมรับคำแนะนำและข้อเสนอแนะทุกประการด้วยความยินดี

สุดท้ายนี้ **ศูนย์หนังสือราคานักเรียน** หวังเป็นอย่างยิ่งว่าหนังสือเล่มนี้จะสามารถตอบโจทย์ในการเรียนรู้ พัฒนา และต่อยอดองค์ความรู้ด้านการพัฒนาเว็บแอปพลิเคชันด้วยภาษา Java และ JSP ได้อย่างแท้จริง อันจะนำไปสู่การพัฒนาทักษะวิชาชีพและการสร้างสรรค์ระบบสารสนเทศที่มีคุณภาพในอนาคต

ด้วยความปรารถนาดี

ศูนย์หนังสือราคานักเรียน

สารบัญ

หน้า

บทที่ 1
พื้นฐานเจเอสพี
(JSP Basic)

เนื้อหา

- การพัฒนา JSP
- ประวัติความเป็นมาของ JSP
- สถานการณ์ปัจจุบันของ JSP
- เทคโนโลยีที่มาแทนที่ JSP
- ส่วนประกอบของ JSP
- ขั้นตอนการตั้งค่าและใช้งาน JSP ใน NetBeans
- ตัวอย่างการใช้งาน JSP แบบละเอียด
- ขั้นตอนการตั้งค่าและใช้งาน JSP ใน NetBeans โดยใช้ GlassFish Web Server
- ตัวอย่างการใช้งาน JSP ใน GlassFish
- รูปแบบการเขียนโค้ดของเจเอสพี
- การใช้คำสั่งของภาษาเจเอสพี
- การใช้งานฟอร์มร่วมกับเจเอสพี

JSP (Java Server Pages) คือเทคโนโลยีจาก Java ที่ใช้ในการพัฒนาเว็บแอปพลิเคชันเพื่อสร้างหน้าเว็บที่สามารถแสดงผลข้อมูลแบบ ไดนามิก ซึ่งเปลี่ยนแปลงได้ตามเวลาหรือสถานะต่าง ๆ โดยทำงานร่วมกับเทคโนโลยีอื่น ๆ เช่น Java Servlets และ JavaBeans เพื่อจัดการและประมวลผลข้อมูลฝั่งเซิร์ฟเวอร์ JSP มักถูกนำมาใช้ในเว็บไซต์ที่ต้องการการประมวลผลข้อมูลจากผู้ใช้ เช่น การกรอกฟอร์ม การเชื่อมต่อกับฐานข้อมูล หรือการแสดงผลที่เปลี่ยนแปลงตามข้อมูลจากเซิร์ฟเวอร์ โดยสามารถทำงานร่วมกับ Servlet ได้อย่างมีประสิทธิภาพ: Servlet ทำหน้าที่จัดการการประมวลผลข้อมูล และ JSP ทำหน้าที่ในการแสดงผล JSP ช่วยให้การพัฒนาเว็บแอปพลิเคชันด้วย Java มีความยืดหยุ่นและง่ายขึ้นด้วยการแยกส่วนการประมวลผลข้อมูลออกจากการแสดงผล ทำให้สามารถนำไปรวมกับเทคโนโลยีอื่น ๆ ได้อย่างสะดวก

JSP พัฒนาโดย Sun Microsystems (ปัจจุบันคือส่วนหนึ่งของ Oracle Corporation) เพื่อแก้ไขปัญหาข้อจำกัดของ CGI (Common Gateway Interface) และ SSI (Server-Side Includes) ซึ่งเป็นเทคโนโลยีเดิมที่ใช้ในการสร้างหน้าเว็บแบบไดนามิก แต่ขาดความยืดหยุ่นและไม่เหมาะสมกับการใช้

งานในระยะยาวสำหรับเว็บแอปพลิเคชันที่ซับซ้อน JSP เป็นภาษาสคริปต์ที่ประมวลผลฝั่งเซิร์ฟเวอร์ ซึ่งเป็นการขยายขีดความสามารถของ Java Servlet ให้ทำงานในรูปแบบของภาษาสคริปต์ โดยปกติจะอยู่ในแพ็คเกจ javax.servlet.jsp ซึ่งรวบรวมคลาสและเมธอดการทำงานต่าง ๆ รวมถึงแพ็คเกจ javax.servlet.jsp.tagtest สำหรับสร้างแท็กได้เอง ส่วนประกอบสำคัญของ JSP ได้แก่ ไดเรกทีฟ (Directive) สำหรับกำหนดค่าการทำงานของ JSP Container เช่น page, include, taglib; แอ็คชั่น (Action) สำหรับจัดเตรียมและดำเนินการบางอย่างในหน้าเพจโดยใช้แท็กที่กำหนดไว้ล่วงหน้า เช่น <jsp:include>, <jsp:forward> หรือการเรียกใช้ JavaBean; สคริปต์เล็ต (Scriptlet) ซึ่งเป็นส่วนของการเขียนโค้ดภาษา Java ภายใน JSP; และ แท็กไลบรารี (Tag Libraries) ที่ช่วยขยายความสามารถของภาษาโดยการกำหนดและเรียกใช้แท็กที่สร้างขึ้นเองได้

การพัฒนา JSP

1. จุดเริ่มต้นของ JSP

ในช่วงปลายปี 1990s **Sun Microsystems** ได้เห็นปัญหาของการใช้ CGI และ SSI ที่ไม่ได้มีความยืดหยุ่นและมีประสิทธิภาพต่ำ เมื่อเทียบกับเทคโนโลยีใหม่ๆ จึงได้พัฒนา **JSP** ขึ้นมาเพื่อตอบโจทย์ดังกล่าว โดยมีเป้าหมายในการพัฒนาเทคโนโลยีสำหรับการสร้างหน้าเว็บที่สามารถผสมผสานการเขียนโค้ด Java เข้ากับ HTML ได้อย่างง่ายดาย

2. การเปิดตัว JSP

JSP ได้รับการเปิดตัวครั้งแรกในปี **1999** ในเวอร์ชันแรกที่เป็นส่วนหนึ่งของ **Java 2 Enterprise Edition (J2EE)** ซึ่งเป็นสเปคที่ถูกออกแบบมาเพื่อให้สามารถพัฒนาแอปพลิเคชันด้านธุรกิจได้อย่างมีประสิทธิภาพ โดย JSP ถูกใช้ร่วมกับ **Java Servlets** ซึ่งเป็นเทคโนโลยีสำหรับการประมวลผลข้อมูลที่ทำงานบนเซิร์ฟเวอร์

3. การพัฒนาและการขยายตัวของ JSP

หลังจากการเปิดตัวครั้งแรก JSP ก็ได้รับความนิยมจากนักพัฒนาที่ใช้ Java ในการพัฒนาเว็บแอปพลิเคชัน เพราะมันช่วยให้การพัฒนาเว็บแอปพลิเคชันง่ายขึ้น โดยสามารถแยกแยะระหว่างการแสดงผล HTML และการประมวลผลข้อมูลด้วย Java ได้อย่างชัดเจน

ในปี 2001 JSP 1.2 ถูกปล่อยออกมาและมีการปรับปรุงหลายประการ โดยการเพิ่มการสนับสนุนการใช้ **JavaBeans** และ **custom tags** ที่ช่วยให้สามารถใช้ JSP ในการพัฒนาเว็บแอปพลิเคชันได้อย่างยืดหยุ่นมากขึ้น

4. การพัฒนาและการสนับสนุน JSTL

ในปี 2002 ได้มีการเปิดตัว **JSTL (JavaServer Pages Standard Tag Library)** ซึ่งเป็นการนำเสนอชุดของแท็กมาตรฐานที่ช่วยในการทำงานกับข้อมูล เช่น การวนลูป การจัดการเงื่อนไขต่างๆ และการแสดงผลข้อมูลที่ได้จากฐานข้อมูล โดยทำให้การเขียนโค้ด JSP ง่ายขึ้นมาก เนื่องจากไม่จำเป็นต้องใช้สคริปต์ Java มากมาย

5. การเปลี่ยนแปลงใน Java EE และการรวมกับเทคโนโลยีอื่น ๆ

JSP เป็นส่วนหนึ่งของ **Java EE** (Enterprise Edition) ซึ่งได้รับการพัฒนาอย่างต่อเนื่อง โดยในปี 2006 Java EE 5 ได้รวมเอา JSP และเทคโนโลยีที่เกี่ยวข้อง เช่น **EJB (Enterprise JavaBeans)** และ **JPA (Java Persistence API)** มาพัฒนาร่วมกัน ทำให้สามารถสร้างแอปพลิเคชันทางธุรกิจที่มีความซับซ้อนได้มากขึ้น ในปี 2009 Java EE 6 ได้ทำการปรับปรุงเทคโนโลยี JSP โดยการทำให้สามารถใช้ **annotations** ในการกำหนดค่าและลดความซับซ้อนในการเขียนโค้ด

6. การพัฒนาในยุคปัจจุบัน

ถึงแม้ว่า JSP จะได้รับความนิยมในช่วงแรกๆ แต่ในปัจจุบันได้มีเทคโนโลยีใหม่ๆ ที่มีประสิทธิภาพมากขึ้น เช่น **JavaServer Faces (JSF)**, **Spring MVC**, และ **Thymeleaf** ที่ใช้ในการพัฒนาเว็บแอปพลิเคชันในระบบที่มีความซับซ้อนสูงขึ้น โดย JSP ยังคงใช้ในบางกรณีที่ต้องการความยืดหยุ่นในการพัฒนาและแสดงผล HTML ที่เปลี่ยนแปลงได้

จุดเด่นของ JSP

- **แยกการแสดงผลกับการประมวลผล**: ทำให้สามารถจัดการการแสดงผลและการประมวลผลข้อมูลได้แยกกันอย่างชัดเจน
- **ง่ายต่อการใช้งาน**: สามารถเขียนโค้ด Java ร่วมกับ HTML ได้โดยตรง
- **สามารถขยายการทำงานด้วย custom tags และ JSTL**: ทำให้สามารถสร้างโค้ดที่สามารถนำกลับมาใช้ใหม่ได้
- **รองรับการทำงานกับ JavaBeans และฐานข้อมูล**: ทำให้สามารถสร้างเว็บแอปพลิเคชันที่มีฟังก์ชันเชื่อมต่อกับฐานข้อมูลได้

JSP เป็นเทคโนโลยีที่เกิดขึ้นเพื่อแก้ปัญหาของการพัฒนาเว็บแอปพลิเคชันในยุคก่อน โดยช่วยให้การสร้างหน้าเว็บที่มีการประมวลผลข้อมูลบนเซิร์ฟเวอร์ทำได้ง่ายขึ้น และมีการพัฒนามาอย่างต่อเนื่องจนถึงปัจจุบัน ถึงแม้จะมีเทคโนโลยีใหม่ๆ เข้ามาแทนที่ในบางกรณี แต่ JSP ยังคงเป็นเครื่องมือที่สำคัญในโลกของการพัฒนาเว็บแอปพลิเคชันด้วย Java JSP (JavaServer Pages) เป็นเทคโนโลยีที่พัฒนาโดย **Sun Microsystems** (ปัจจุบันเป็นส่วนหนึ่งของ Oracle Corporation) เพื่อเพิ่มประสิทธิภาพในการสร้างหน้าเว็บที่มีการประมวลผลข้อมูลบนเซิร์ฟเวอร์ โดยการผสมผสานโค้ด Java เข้ากับ HTML ทำให้สามารถสร้างเว็บเพจที่มีความยืดหยุ่นและตอบสนองต่อผู้ใช้ได้ดีขึ้น

ประวัติความเป็นมาของ JSP

- **ปี 1999:** Sun Microsystems เปิดตัว JSP ครั้งแรกในฐานะส่วนหนึ่งของ **Java 2 Enterprise Edition (J2EE)** ซึ่งเป็นสเปคที่ออกแบบมาเพื่อพัฒนาแอปพลิเคชันทางธุรกิจ
- **ปี 2001:** JSP 1.2 ถูกปล่อยออกมา โดยมีการปรับปรุงหลายประการ เช่น การเพิ่มการสนับสนุนการใช้ **JavaBeans** และ **custom tags** ที่ช่วยให้การพัฒนาเว็บแอปพลิเคชันมีความยืดหยุ่นมากขึ้น

- **ปี 2002:** เปิดตัว **JSTL (JavaServer Pages Standard Tag Library)** ซึ่งเป็นชุดของแท็กมาตรฐานที่ช่วยในการจัดการข้อมูล เช่น การวนลูป การจัดการเงื่อนไข และการแสดงผลข้อมูลจากฐานข้อมูล ทำให้การเขียนโค้ด JSP ง่ายขึ้น
- **ปี 2006:** Java EE 5 รวมเอา JSP และเทคโนโลยีที่เกี่ยวข้อง เช่น **EJB (Enterprise JavaBeans)** และ **JPA (Java Persistence API)** มาพัฒนาร่วมกัน ทำให้สามารถสร้างแอปพลิเคชันทางธุรกิจที่มีความซับซ้อนได้มากขึ้น
- **ปี 2009:** Java EE 6 ได้ปรับปรุงเทคโนโลยี JSP โดยการทำให้สามารถใช้ **annotations** ในการกำหนดค่าและลดความซับซ้อนในการเขียนโค้ด

JSP ยังคงเป็นเครื่องมือที่สำคัญในการพัฒนาเว็บแอปพลิเคชันด้วย Java แม้ว่าในปัจจุบันจะมีเทคโนโลยีใหม่ๆ เข้ามาแทนที่ในบางกรณี แต่ JSP ยังคงมีบทบาทสำคัญในวงการพัฒนาเว็บแอปพลิเคชัน ในปัจจุบัน, **JSP (JavaServer Pages)** ยังเป็นเครื่องมือที่มีความสำคัญในบางกรณี แต่ก็ไม่ได้เป็นเทคโนโลยีหลักสำหรับการพัฒนาเว็บแอปพลิเคชันใน Java เนื่องจากมีการพัฒนาเทคโนโลยีใหม่ๆ ที่มุ่งเน้นไปที่การพัฒนาเว็บแอปพลิเคชันในลักษณะที่มีประสิทธิภาพสูงขึ้นและง่ายต่อการใช้งานมากกว่า JSP

สถานการณ์ปัจจุบันของ JSP

1. **ลดความนิยมในปัจจุบัน**
   ถึงแม้ว่า JSP จะเป็นเครื่องมือที่มีประสิทธิภาพในช่วงแรกๆ แต่ในปัจจุบันมีเทคโนโลยีใหม่ๆ ที่มีความยืดหยุ่นมากขึ้น เช่น **JavaServer Faces (JSF)**, **Spring MVC**, และ **Thymeleaf** ที่ได้รับความนิยมมากขึ้น เนื่องจากช่วยให้การพัฒนาเว็บแอปพลิเคชันเป็นไปได้ง่ายกว่า โดยไม่ต้องฝังโค้ด Java ในหน้า HTML โดยตรง
2. **การแทนที่ด้วยเทคโนโลยีใหม่ๆ**
   เทคโนโลยีอย่าง **Spring MVC** และ **Thymeleaf** ได้กลายเป็นทางเลือกยอดนิยมในการพัฒนาเว็บแอปพลิเคชันในสภาพแวดล้อมของ Java เนื่องจากสามารถแยกส่วนการประมวลผลจากการแสดงผลได้ดีขึ้น และมีการสนับสนุนการใช้งานที่เป็นมิตรต่อผู้พัฒนา เช่น การใช้ **Model-View-Controller (MVC)** ที่ช่วยให้การจัดการการแสดงผลและการประมวลผลข้อมูลทำได้อย่างมีระเบียบ
3. **JSP ใน Java EE (Jakarta EE)**
   ถึงแม้ว่า JSP จะไม่ได้รับความนิยมเหมือนเมื่อก่อน แต่ยังคงถูกใช้ในบางส่วนของ **Jakarta EE (เดิมคือ Java EE)** ซึ่งเป็นสเปคหลักสำหรับการพัฒนาเว็บแอปพลิเคชันใน Java องค์กรที่มีระบบที่ใช้ JSP อยู่แล้วก็ยังคงใช้ต่อไป แต่แนวโน้มการใช้งานลดลงเมื่อเทียบกับการใช้เทคโนโลยีที่ทันสมัยกว่า
4. **การใช้งานในบางกรณี**

JSP ยังเหมาะสมกับการใช้งานในโปรเจกต์ที่ต้องการโครงสร้างง่ายๆ หรือไม่ต้องการฟังก์ชันการทำงานที่ซับซ้อนเกินไป ซึ่งมันยังคงให้ประโยชน์ในกรณีที่มีการสร้างเว็บแอปพลิเคชันแบบไดนามิกที่ไม่ซับซ้อนหรือใช้สำหรับการเรียนการสอนการพัฒนาเว็บแอปพลิเคชัน

5. **การใช้งานในเครื่องมือเก่า**
   JSP ยังคงถูกใช้อยู่ในเครื่องมือหรือระบบที่ถูกพัฒนามาแล้วก่อนที่จะมีการแนะนำเทคโนโลยีใหม่ เช่น ในระบบ **Apache Tomcat** ซึ่งยังคงรองรับ JSP เพื่อความเข้ากันได้กับระบบเก่าและโปรเจกต์ที่ยังใช้ JSP อยู่

เทคโนโลยีที่มาแทนที่ JSP

- **Spring MVC**: ใช้แนวคิด **Model-View-Controller** (MVC) ซึ่งทำให้การแยกการประมวลผลและการแสดงผลข้อมูลเป็นระเบียบ และการทำงานกับ Spring Framework มีความยืดหยุ่นมากขึ้น
- **Thymeleaf**: เป็นเครื่องมือที่นิยมใช้ในการสร้างเทมเพลตสำหรับแสดงผลหน้าเว็บใน Spring Boot โดยให้การทำงานที่สะดวกและเป็นธรรมชาติมากขึ้น

อย่างไรก็ตาม JSP ยังสามารถใช้ได้ในบางกรณี โดยเฉพาะในโปรเจกต์ที่ต้องการความเรียบง่ายและไม่ได้ใช้เทคโนโลยีใหม่ๆ ที่ซับซ้อนมากนัก แต่ในภาพรวม, JSP ถูกแทนที่ด้วยเทคโนโลยีที่มีความยืดหยุ่นและสะดวกในการใช้งานมากกว่า เช่น Spring MVC และ Thymeleaf ซึ่งทำให้การพัฒนาเว็บแอปพลิเคชันในสภาพแวดล้อมของ Java มีความยืดหยุ่นและสะดวกมากขึ้น
การใช้งาน JSP (JavaServer Pages) ใน NetBeans IDE เป็นกระบวนการที่ไม่ซับซ้อน และสามารถทำได้ตามขั้นตอนที่แนะนำด้านล่างนี้ โดยใช้ Apache Tomcat เป็นเซิร์ฟเวอร์เพื่อรันแอปพลิเคชัน JSP

ส่วนประกอบของ JSP

1. **Directives**: ใช้ในการกำหนดพารามิเตอร์เกี่ยวกับการทำงานของ JSP เช่น การนำเข้าไลบรารี หรือการกำหนด encoding ของไฟล์
   - ตัวอย่าง:
   - <%@ page language="java" contentType="text/html; charset=ISO-8859-1"%>
2. **Declarations**: ใช้ในการประกาศตัวแปรหรือเมธอดที่สามารถใช้ได้ทั่วทั้งหน้า JSP
   - ตัวอย่าง:
   - <%! int counter = 0; %>
3. **Scriplets**: ส่วนที่ฝังโค้ด Java ลงใน HTML ใช้สำหรับการเขียนคำสั่ง Java ที่จะถูกประมวลผลเมื่อหน้าเว็บถูกเรียกใช้งาน
   - ตัวอย่าง:
   - <% out.println("Hello, world!"); %>

4. **Expressions**: ใช้ในการแสดงค่าของตัวแปรหรือการคำนวณใน HTML
    - ตัวอย่าง:
    - <%= 2 + 2 %> <!-- แสดงผลเป็น 4 -->
5. **Actions**: ใช้ในการเรียกใช้คำสั่งหรือฟังก์ชันจากภายนอก เช่น การใช้งาน JavaBeans หรือการเชื่อมต่อกับฐานข้อมูล
    - ตัวอย่าง:
    - <jsp:useBean id="user" class="com.example.User" />
6. **Standard Tag Library (JSTL)**: เป็นคอลเลกชันของแท็กที่ใช้ในการจัดการข้อมูล เช่น การวนลูปหรือการแสดงผลที่มีเงื่อนไข
    - ตัวอย่าง:
    - <c:forEach var="item" items="${list}">
    - ${item}
    - </c:forEach>

ขั้นตอนการตั้งค่าและใช้งาน JSP ใน NetBeans

1. ติดตั้ง NetBeans IDE และ Apache Tomcat

- ดาวน์โหลดและติดตั้ง **NetBeans IDE** จากเว็บไซต์ NetBeans
- ดาวน์โหลดและติดตั้ง **Apache Tomcat** (เวอร์ชันที่รองรับ JSP และ Servlets) จากเว็บไซต์ Tomcat

2. ตั้งค่า Apache Tomcat ใน NetBeans

1. เปิด **NetBeans IDE** ขึ้นมา
2. ไปที่ **Tools** > **Servers** > คลิกที่ **Add Server...**
3. เลือก **Apache Tomcat** จากรายการเซิร์ฟเวอร์ที่รองรับ
4. เลือกตำแหน่งที่ติดตั้ง **Apache Tomcat** และกด **Next** ตามด้วยการตั้งค่าต่างๆ ที่จำเป็นจนเสร็จสิ้น

3. สร้างโปรเจกต์ JSP

1. ใน **NetBeans IDE**, ไปที่ **File** > **New Project...**
2. เลือก **Java Web** > **Web Application** แล้วกด **Next**
3. กำหนดชื่อโปรเจกต์และตำแหน่งการบันทึก
4. เลือก **Apache Tomcat** เป็นเซิร์ฟเวอร์ที่ใช้ในการรันโปรเจกต์
5. กด **Finish** เพื่อสร้างโปรเจกต์ใหม่

4. สร้างไฟล์ JSP

1. ใน **Project Explorer**, คลิกขวาที่ **Web Pages** > เลือก **New** > **JSP** (JavaServer Page)

2. กำหนดชื่อไฟล์ JSP เช่น index.jsp แล้วกด **Finish**

5. เขียนโค้ด JSP

- เปิดไฟล์ index.jsp แล้วเพิ่มโค้ดตัวอย่าง JSP ลงไป:
- <%-- index.jsp --%>
- <html>
- <head>
- <title>Welcome to JSP</title>
- </head>
- <body>
- <h1>Welcome to JavaServer Pages (JSP)</h1>
- <%
- String message = "Hello, World!";
- out.println(message); // ใช้คำสั่ง Java เพื่อแสดงข้อความ
- %>
- </body>
- </html>

6. รันโปรเจกต์

- คลิกขวาที่โปรเจกต์ใน **Project Explorer** และเลือก **Run** (หรือกด **F6**) เพื่อรันโปรเจกต์
- เว็บเบราว์เซอร์จะเปิดขึ้น และแสดงผลลัพธ์จาก index.jsp

7. ผลลัพธ์ในเว็บเบราว์เซอร์

- หน้าเว็บจะถูกแสดงผลตามโค้ดที่เขียนใน index.jsp โดยจะแสดงข้อความ **"Welcome to JavaServer Pages (JSP)"** และ **"Hello, World!"**
-

ตัวอย่างการใช้งาน JSP แบบละเอียด

สมมุติว่าเราต้องการสร้างหน้าเว็บที่รับค่าจากผู้ใช้และแสดงผลข้อความจากแบบฟอร์ม:

1. **สร้างไฟล์ JSP สำหรับรับค่าผู้ใช้:**
    - สร้างไฟล์ **form.jsp**:
    - <%-- form.jsp --%>
    - <html>
    - <head>
    - <title>Form Input</title>
    - </head>
    - <body>
    - <h1>Please Enter Your Name</h1>

```
        <form action="greeting.jsp" method="POST">
            Name: <input type="text" name="name" />
            <input type="submit" value="Submit" />
        </form>
    </body>
</html>
```

2. **สร้างไฟล์ JSP สำหรับแสดงผล:**
    - สร้างไฟล์ **greeting.jsp**:

```
<%-- greeting.jsp --%>
<html>
    <head>
        <title>Greeting</title>
    </head>
    <body>
        <h1>Welcome,
            <%
                String name = request.getParameter("name");  // รับค่าจาก form
                if (name != null) {
                    out.println(name);  // แสดงชื่อที่กรอก
                } else {
                    out.println("Guest");
                }
            %>!
        </h1>
    </body>
</html>
```

3. **ทดสอบการทำงาน**:
    - รันโปรเจกต์ใน **NetBeans IDE** และเปิดหน้า form.jsp ผ่านเบราว์เซอร์
    - กรอกชื่อในฟอร์มแล้วส่งข้อมูลไปยัง greeting.jsp ซึ่งจะทำการแสดงผลการต้อนรับตามชื่อที่กรอก

การใช้งาน **JSP (JavaServer Pages)** ใน **NetBeans IDE** และรันบน **GlassFish Web Server** มีขั้นตอนที่คล้ายกับการใช้งานใน Apache Tomcat เพียงแค่ต้องตั้งค่า **GlassFish** เป็น

เซิร์ฟเวอร์ที่ใช้งานในโปรเจกต์ ต่อไปนี้คือขั้นตอนและตัวอย่างการใช้งาน JSP บน GlassFish Web Server:

ขั้นตอนการตั้งค่าและใช้งาน JSP ใน NetBeans โดยใช้ GlassFish Web Server

1. ติดตั้ง NetBeans IDE และ GlassFish Server

- ดาวน์โหลดและติดตั้ง **NetBeans IDE** จากเว็บไซต์ NetBeans
- **GlassFish Server** มักจะมาพร้อมกับ **NetBeans IDE** อยู่แล้ว ถ้าไม่ได้ติดตั้ง GlassFish Server ให้ทำการดาวน์โหลดจากเว็บไซต์ GlassFish และติดตั้ง

2. ตั้งค่า GlassFish Server ใน NetBeans

1. เปิด **NetBeans IDE** ขึ้นมา
2. ไปที่ **Tools** > **Servers** > คลิกที่ **Add Server...**
3. เลือก **GlassFish Server** จากรายการเซิร์ฟเวอร์ที่รองรับ
4. กำหนดที่อยู่และตำแหน่งของการติดตั้ง **GlassFish Server** หากยังไม่ได้ติดตั้ง GlassFish ให้ทำการติดตั้งและระบุที่ตั้ง
5. กด **Next** แล้ว **Finish** หลังจากตั้งค่าเสร็จ

3. สร้างโปรเจกต์ JSP (Web Application) ใน NetBeans

1. ใน **NetBeans IDE**, ไปที่ **File** > **New Project...**
2. เลือก **Java Web** > **Web Application** แล้วกด **Next**
3. กำหนดชื่อโปรเจกต์และตำแหน่งการบันทึก
4. เลือก **GlassFish** เป็นเซิร์ฟเวอร์ที่ใช้ในการรันโปรเจกต์
5. กด **Finish** เพื่อสร้างโปรเจกต์ใหม่

4. สร้างไฟล์ JSP

1. ใน **Project Explorer**, คลิกขวาที่ **Web Pages** > เลือก **New** > **JSP** (JavaServer Page)
2. กำหนดชื่อไฟล์ JSP เช่น index.jsp แล้วกด **Finish**

5. เขียนโค้ด JSP

- เปิดไฟล์ index.jsp แล้วเพิ่มโค้ดตัวอย่าง JSP ลงไป:
- <%-- index.jsp --%>
- <html>
- <head>
- <title>Welcome to JSP</title>
- </head>
- <body>
- <h1>Welcome to JavaServer Pages (JSP)</h1>
- <%

```
            String message = "Hello, World!";
            out.println(message);  // ใช้คำสั่ง Java เพื่อแสดงข้อความ
        %>
    </body>
</html>
```

6. รันโปรเจกต์บน GlassFish Server

1. คลิกขวาที่โปรเจกต์ใน **Project Explorer** แล้วเลือก **Run** (หรือกด **F6**) เพื่อรันโปรเจกต์
2. **NetBeans** จะเริ่ม **GlassFish Server** และเปิดเบราว์เซอร์ที่แสดงหน้าเว็บจาก index.jsp
3. ถ้าไม่มีเบราว์เซอร์เปิดขึ้น, คุณสามารถเปิดเบราว์เซอร์เองและไปที่ **http://localhost:8080/your-project-name/index.jsp** เพื่อดูผลลัพธ์

ตัวอย่างการใช้งาน JSP ใน GlassFish

สมมุติว่าต้องการสร้างหน้าเว็บที่รับค่าจากผู้ใช้และแสดงผลข้อความจากแบบฟอร์ม:

1. **สร้างไฟล์ JSP สำหรับรับค่าผู้ใช้**:
    - สร้างไฟล์ **form.jsp**:
    - <%-- form.jsp --%>
    - <html>
    - <head>
    - <title>Form Input</title>
    - </head>
    - <body>
    - <h1>Please Enter Your Name</h1>
    - <form action="greeting.jsp" method="POST">
    - Name: <input type="text" name="name" />
    - <input type="submit" value="Submit" />
    - </form>
    - </body>
    - </html>
2. **สร้างไฟล์ JSP สำหรับแสดงผล**:
    - สร้างไฟล์ **greeting.jsp**:
    - <%-- greeting.jsp --%>
    - <html>
    - <head>
    - <title>Greeting</title>

```
    </head>
    <body>
        <h1>Welcome,
            <%
                String name = request.getParameter("name");  // รับค่าจาก form
                if (name != null) {
                    out.println(name);  // แสดงชื่อที่กรอก
                } else {
                    out.println("Guest");
                }
            %>!
        </h1>
    </body>
</html>
```

3. **ทดสอบการทำงาน**:
   - รันโปรเจกต์ใน **NetBeans IDE** และเปิดหน้า form.jsp ผ่านเบราว์เซอร์
   - กรอกชื่อในฟอร์มแล้วส่งข้อมูลไปยัง greeting.jsp ซึ่งจะทำการแสดงผลการต้อนรับตามชื่อที่กรอก

รูปแบบการเขียนโค้ดของเจเอสพี

รูปแบบการเขียนภาษาเจเอสพี เขียนได้ในสามลักษณะดังนี้

1. สคริปต์เล็ต

เขียนภายใต้เครื่องหมาย <% code %> ซึ่งเป็นรูปแบบการเขียนหลักที่สามารถเปิดและปิดในโค้ดของภาษาเอชทีเอ็มแอลได้ตลอดเวลา ตัวอย่างแสดงดังนี้

```
<%
        int x=10;
        double 10.50;
        String s=”Testing JSP String”
        int sum=x+100;
         out.println(sum);
%>
```

2. เดคลอเรชัน (Declarations)

เดคคลอเรชันเป็นการกำหนดเมธอดใช้งานด้วยโค้ดภาษาจาวาเพื่อเรียกใช้ในส่วนอื่นในหน้าเพจ มีรูปแบบที่ไป 2 แบบ คือ กำหนดด้วยรูปแบบเจเอสพี และแบบเอ็กซ์เอ็มแอลดังนี้

แบบเจเอสพี : <%! code %>

แบบเอ็กซ์เอ็มแอล : <jsp:declaration> code </jsp:declaration>

ตัวอย่างเช่น

```
<%!
    public String showName(String s){
        Return “สวัสดีคุณ:”+s;
    }
%>
```

หรือ

```
<jsp:declaration>
    public String showName(String s){
        Return “สวัสดีคุณ:”+s;
    }
</jsp:declaration>
```

3. เอ็กซ์เพรสชัน (Expressions)

การใช้งานแบบนี้ คือ ใช้งานคำสั่งแสดงตัวแปรที่ต้องการให้แสดงค่า ณ ตำแหน่งใดตำแหน่งหนึ่งในหน้าเพจ สามารถเขียนได้ทั้งรูปแบบของเจเอสพีและแบบเอ็กซ์เอ็มแอล ดังนี้

แบบเจเอสพี : <%= code %>

แบบเจเอสพี : <jsp:expression> code </jsp:expression>

```
<%=x%>
```

หรือ

```
<jsp:expression> x </jsp:expression>
```

ตัวอย่างหน้าเพจที่เขียนด้วยภาษาเจเอสพี

```
<%@page contentType="text/html" pageEncoding="UTF-8"%> //ส่วนหัวของเพจ
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<html>                    //โครงสร้างภาษา HTML
```

```
    <head>              //ส่วนหัวของภาษา HTML
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>  // ส่วนชื่อเรื่องของเพจ
    </head>
    <body>                  //ส่วนการเริ่มต้น ของการเขียนโปแกรม บอดี้
<%
          int x=10;
          double 10.50;
          int sum=x+100;
         out.println(sum);
%>
<%!
     public String showName(String s){
        Return “สวัสดีคุณ:”+s;
     }
%>
<%=showName()>

    </body>                  //จบส่วนบอดี้
</html>                     //ส่วนการจบแฟ้ม HTML
```

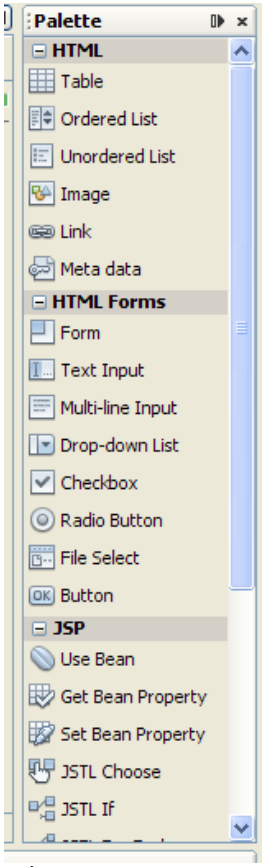
การใช้คำสั่งของภาษาเจเอสพี

1) คำสั่งแสดงผล

การใช้งานคำสั่งแสดงผลโดยทั่วไปจะใช้ out.println() เป็นหลัก โดยเขียนผ่านแท็ก เอ็กซ์เพรสชันเป็นต้น

2) คำสั่งรับข้อมูล

โดยทั่วไปจะไม่มีคำสั่งรับข้อมูลจากภาษาเจเอสพีโดยตรง แต่จะรับข้อมูลผ่านฟอร์มของภาษา HTML ซึ่งจะกล่าวในส่วนต่อไป แสดงดังภาพต่อไปนี้ โดยในการออกแบบโปรแกรมสามารถลากมาวางแล้วเขียนรหัสโปรแกรมตามแนวทางของการใช้งานโปรแกรมเน็ตบีนได้ทันที ซึ่งจะแสดงในหัวข้อต่อไป



ภาพที่ 1.1 แสดงเครื่องมือสำหรับพัฒนาโปรแกรม

3) คำสั่งเงื่อนไข

คำสั่งประเภทนี้จะใช้คำสั่งของภาษาจาวาแทรกเข้าไปในภาษา HTML ได้ ทั้งคำสั่ง if และเงื่อนไขแบบ switch

ตัวอย่างการใช้งานเงื่อนไข if

```
<%
int x = 5;
if (x > 10) {
  out.println(“OK.”);
} else {
  out.println(“Not OK.”);
}
%>
```

ตัวอย่างการใช้งานเงื่อนไขแบบ switch

```
<%
int m = 5;
```

```
switch (m) {
case 1 : {out.println("Banana");  break;}
case 2 : {out.println("Apple");  break;}
case 3 : {out.println("Lemon");  break;}
case 4 : {out.println("Orange"); break;}
default : {out.println("Should be 1-4");}
}
%>
```

คำสั่งวนซ้ำ

ตัวอย่างการใช้งานคำสั่ง for

```
<%
for (int i=1;i<=100;i++){
  out.println(i);
}
%>
```

ตัวอย่างการใช้งานลูป while

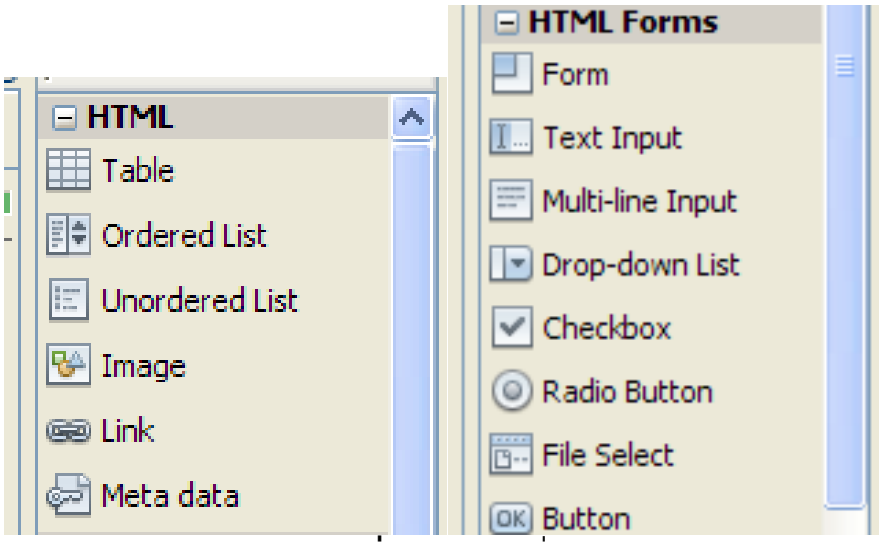
```
<%
int i=0;
while (i<= 10) {
  out.println(i);
  i++;
}
%>
```

ตัวอย่างการใช้งานลูป do…while

```
<%
int i =0;
do {
  out.println(i);
  i++;
} while (i <= 100);
%>
```

การใช้งานฟอร์มร่วมกับเจเอสพี

การใช้งานฟอร์มนั้นโดยทั่วไปมีวัตถุประสงค์ในการรับค่าจากผู้ใช้เพื่อประมวลผลอย่างใดอย่างหนึ่ง เช่น การส่งค่าเพื่อรายงานผล ส่งค่าเพื่อบันทึกลงฐานข้อมูล ส่งค่าฟอร์มจากภาษา HTML ไปยังฟอร์มตนเองหรือต่างฟอร์มก็ได้ การใช้งานฟอร์มจะใช้ส่วนดังภาพ

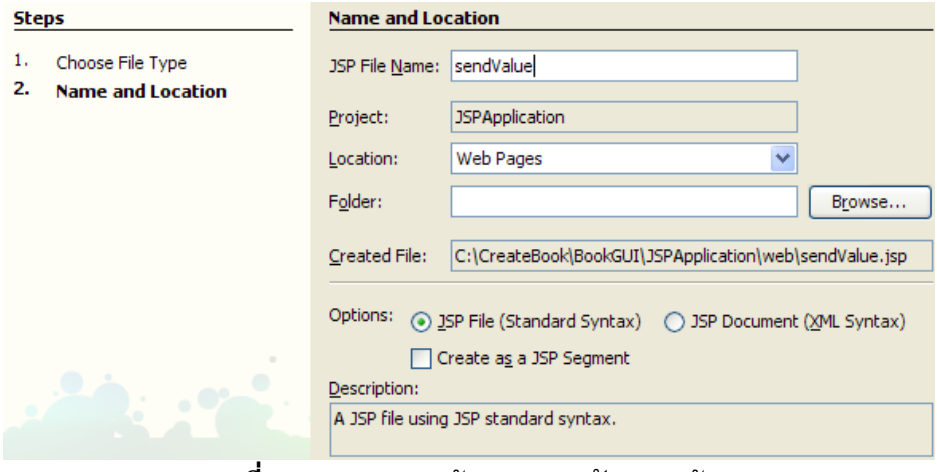


ภาพที่ 1.2 แสดงเครื่องมือจัดการ HTML และฟอร์ม

ตัวอย่างการใช้งานฟอร์มเพียงฟอร์มเดียวด้วยการส่งค่าให้กับฟอร์มของตนเอง เช่น การประกาศฟอร์มแล้วสร้างส่วนรับข้อมูลสองปุ่มแล้วรับข้อมูลจากผู้ใช้ หลังจากนั้น คลิกเพื่อส่งค่าแสดงให้กับฟอร์มตนเอง และแสดงผล โดยทำตามตัวอย่างต่อไปนี้

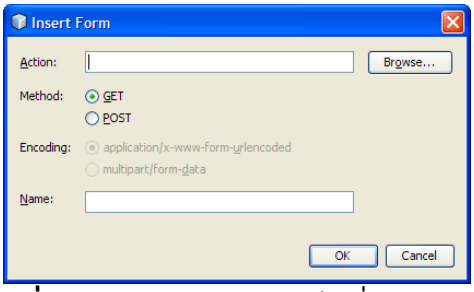
ตัวอย่างที่ 1.1 จงออกแบบฟอร์มรับชื่อและสกุล เมื่อคลิกปุ่มแสดงผล ให้แสดงชื่อและนามสกุลที่ป้อนให้แสดงบนเพจเดิม

1. สร้างแฟ้มเจเอสพี โดยการคลิกปุ่มขวาของเมาส์ที่ WebPage->JSP แล้วกำหนดชื่อ ดังภาพ



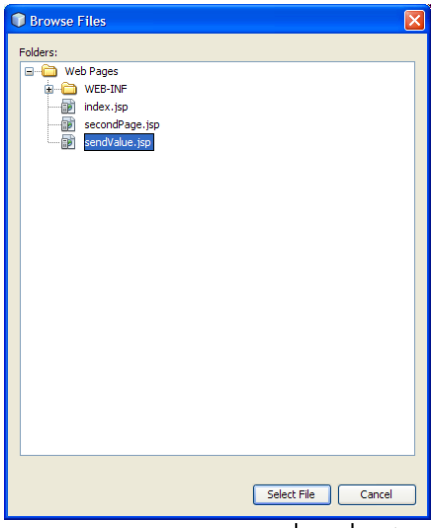
ภาพที่ 1.3 แสดงการสร้างเพจสำหรับส่งค่าข้อมูล

2. ออกแบบฟอร์มในส่วน <body> โดยการคลิกลาก Form มาวางดังภาพ



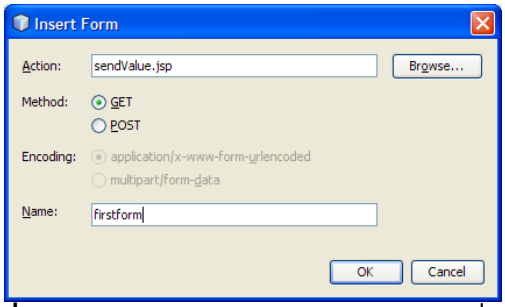
ภาพที่ 1.4 แสดงการสร้างฟอร์มเมื่อลากมาวาง

3. เลือก Browse เลือกชื่อตนเอง แล้วคลิก OK จะได้นี้



ภาพที่ 1.5 แสดงการเลือกฟอร์มที่จะเชื่อมโยง

4. Select File แล้วกำหนดชื่อฟอร์ม ดังภาพ

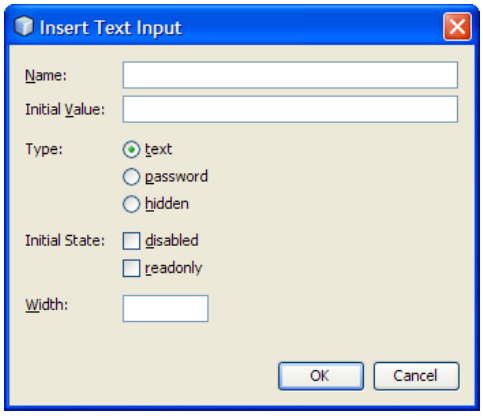


ภาพที่ 1.6 แสดงการกำหนดแอ็คชันของการเชื่อมโยง

5. คลิกปุ่ม OK จะได้รหัสโปรแกรมที่โปรแกรมเน็ตบีนสร้างให้ดังนี้

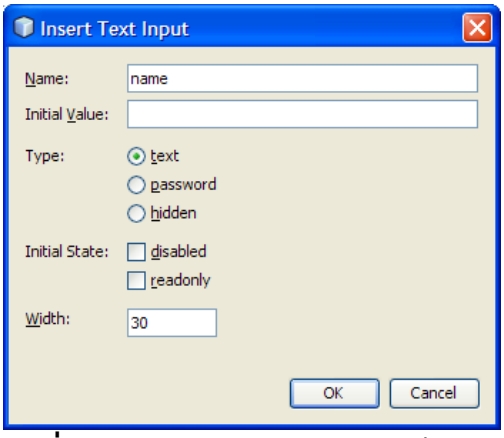
```
<form name="firstform" action="sendValue.jsp">
</form>
```

6. หลังจากนั้นคลิกที่ Text Input ลากมาวางภายในฟอร์ม (ระหว่าง <form> </form>) จะได้ดังนี้



ภาพที่ 1.7 แสดงกำหนดค่าการสร้างเท็กซ์ฟิลด์

7. กำหนดชื่อและความกว้างหรือกำหนดค่าเริ่มต้นด้วยก็ได้ ดังนี้



ภาพที่ 1.8 แสดงการกำหนดค่าเท็กซ์

8. จะได้รหัสโปรแกรมดังนี้

```
<input type="text" name="name" value="" size="30" />
```

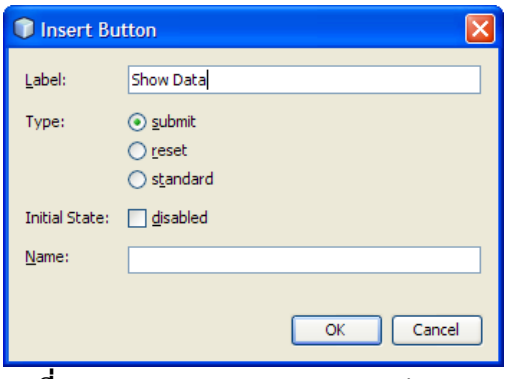
9. ดำเนินการเช่นกันนั้นกับ นามสกุลจะได้รหัสโปรแกรมดังนี้

```
<input type="text" name="surname" value="" size="30" />
```

ให้เขียนข้อความและแท็ก
 เพื่อขึ้นบรรทัดใหม่เพิ่มต่อท้ายส่วนของ Text Input ทั้งสอง ซึ่งเมื่อสมบูรณ์แล้วจะได้ดังนี้

```
Name: <input type="text" name="name" value="" size="30" /><br>
Surname:<input type="text" name="surname" value="" size="30" /><br>
```

10. หลังจากลากส่วนของ Button มาวางเพื่อทำปุ่ม submit เพื่อแสดงผล เมื่อลากมาวางต่อท้ายส่วนของ Text Input จะได้ดังนี้



ภาพที่ 1.9 แสดงการกำหนดค่าของปุ่ม

11. ให้กำหนดเฉพาะส่วนของ Label ดังภาพด้านบน ซึ่งจะได้รหัสโปรแกรมดังนี้

```
<input type="submit" value="Show Data" />
```

12. หลังจากนั้นให้เขียนส่วนของเจเอสพีแท็ก หลังจาก </form> ดังนี้

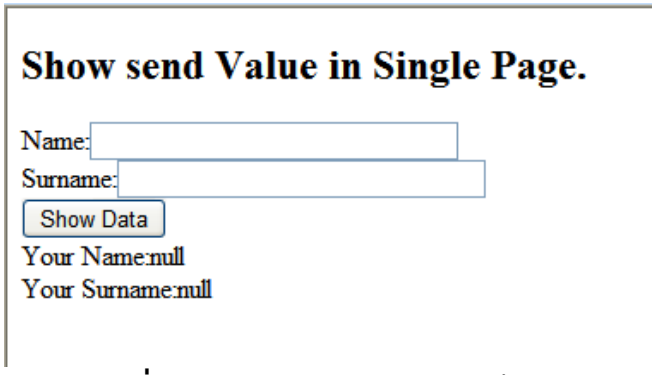
```
<%
    String name=request.getParameter("name");
    String surname=request.getParameter("surname");
%>
<%  out.println("Your Name:"+name); %> <br>
<%  out.println("Your Surname:"+surname); %> <br>
```

รหัสโปรแกรมทั้งหมดจะได้ดังนี้

```
1.  <%@page contentType="text/html" pageEncoding="UTF-8"%>
2.  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
3.  <html>
4.  <head>
5.      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
6.      <title>JSP Page</title>
7.  </head>
8.  <body>
9.       <h2>Show send Value in Single Page.</h2>
10.     <form name="firstform" action="sendValue.jsp">
11.          Name:<input type="text" name="name" value="" size="30" /><br>
12.          Surname:<input type="text" name="surname" value="" size="30" /><br>
13.         <input type="submit" value="Show Data" />
14.   </form>
```

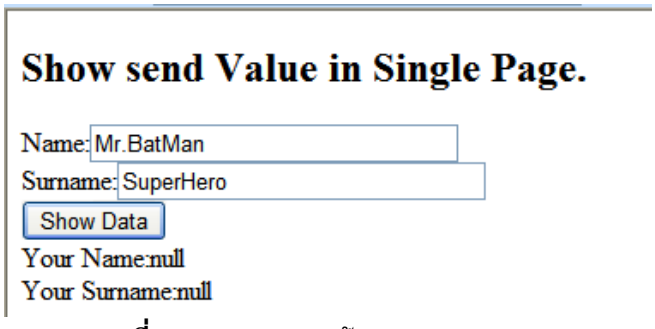
```
15.    <%
16.       String name=request.getParameter("name");
17.       String surname=request.getParameter("surname");
18.    %>
19.    <%  out.println("Your Name:"+name); %> <br>
20.    <%  out.println("Your Surname:"+surname); %> <br>
21. </body>
22. </html>
```

13. ผลการรันโปรแกรมจะได้ดังนี้



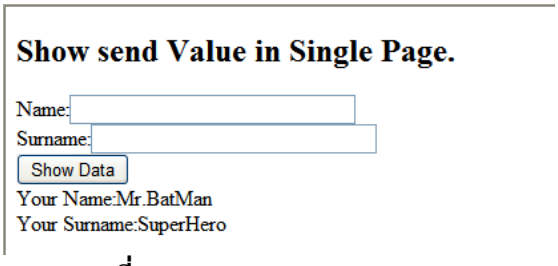
ภาพที่ 1.10 แสดงผลการรันของฟอร์มส่งค่า

เมื่อป้อนข้อมูลดังภาพ



ภาพที่ 1.11 แสดงการป้อนค่าก่อนการส่งค่า

หลังจากนั้นคลิกปุ่ม Show Data จะได้ดังนี้

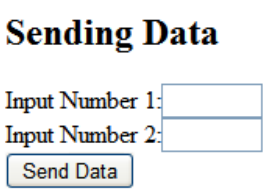


ภาพที่ 1.12 แสดงการส่งค่าเพจตนเอง

การส่งค่าข้ามฟอร์มก็จะมีลักษณะเช่นเดียวกันกับการส่งค่าในเพจ เพียงแต่เลือกแฟ้มที่ลิงค์เชื่อมโยงไปเป็นอีกแฟ้มหนึ่ง โดยสร้างฟอร์มไว้ที่แฟ้มหลักแล้ว หลังจากนั้นสร้างแฟ้มรับค่าที่จะส่งไปแล้วนำข้อมูลออกมาด้วย request.getParameter() ตามค่าที่ส่งข้ามมา ตัวอย่างต่อไปนี้เป็นการสร้างแฟ้มส่งค่าตัวเลขข้ามฟอร์มไปประมวลผลอีกฟอร์มหนึ่ง โดยฟอร์มแรกจะส่งค่าไปแล้วให้อีกฟอร์ม ที่รับค่านำไปคำนวณหาค่า ผลบวก ลบ คูณ และหาร แล้วแสดงผล

ตัวอย่างที่ 1.2 จงเขียนโปรแกรมส่งค่าข้ามเพจโดยเพจแรกเป็นตัวป้อนข้อมูล ส่งค่าให้เพจที่สอง นำไปแสดงผลบวก ลบ คูณ และหาร

1. สร้างฟอร์มป้อนข้อมูลดังนี้



ภาพที่ 1.13 แสดงการออกแบบก่อนการส่งค่าข้ามเพจ

รหัสโปรแกรมของแฟ้มนี้คือ

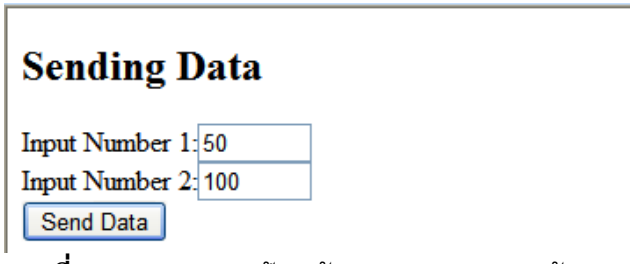
```
1.  <%@page contentType="text/html" pageEncoding="UTF-8"%>
2.  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
3.  <html>
4.  <head>
5.      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
6.      <title>JSP Page</title>
7.  </head>
8.  <body>
9.      <h2>Sending Data</h2>
10.     <form action="resultPage.jsp">
11.      Input Number 1:<input type="text" name="numberone" value="" size="5" /><br>
12.      Input Number 2:<input type="text" name="numbertwo" value="" size="5" /><br>
13.      <input type="submit" value="Send Data" />
14.   </form>
15. </body>
16. </html>
```

2. เขียนส่วนของเพจรับข้อมูล resultPage ดังนี้

```
1.  <%@page contentType="text/html" pageEncoding="UTF-8"%>
```

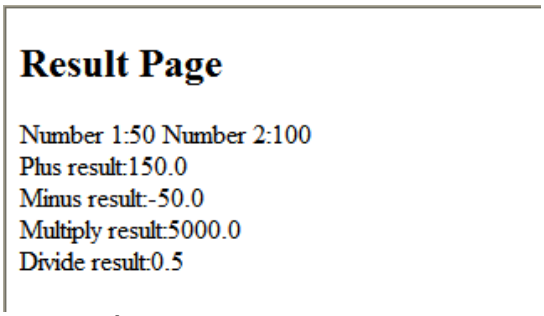
```
2.  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
3.  <html>
4.  <head>
5.      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
6.      <title>JSP Page</title>
7.  </head>
8.  <body>
9.      <h2>Result Page</h2>
10.     <%
11.         String n1=request.getParameter("numberone");
12.         String n2=request.getParameter("numbertwo");
13.         double result1=Double.parseDouble(n1)+Double.parseDouble(n2);
14.         double result2=Double.parseDouble(n1)-Double.parseDouble(n2);
15.         double result3=Double.parseDouble(n1)*Double.parseDouble(n2);
16.         double result4=Double.parseDouble(n1)/Double.parseDouble(n2);
17.     %>
18.      <% out.println("Number 1:"+n1+"  Number 2:"+n2);%> <br>
19.      <% out.println("Plus result:"+result1); %> <br>
20.      <% out.println("Minus result:"+result2); %> <br>
21.      <% out.println("Multiply result:"+result3); %> <br>
22.      <% out.println("Divide result:"+result4); %> <br>
23.  </body>
24. </html>
```

3. ผลการรันเริ่มจากเพจรับข้อมูลดังนี้



ภาพที่ 1.14 แสดงการป้อนข้อมูลก่อนการส่งค่าข้ามเพจ

4. เมื่อป้อนข้อมูลแล้วคลิกปุ่ม Send Data จะให้ผลดังนี้



ภาพที่ 1.15 แสดงผลการส่งค่าข้ามเพจ

สรุป

JSP (Java Server Pages) คือเทคโนโลยีจาก Java สำหรับพัฒนาเว็บแอปพลิเคชันที่สร้างหน้าเว็บแบบ ไดนามิก ซึ่งทำงานร่วมกับ Java Servlets และ JavaBeans เพื่อจัดการข้อมูลและการประมวลผลฝั่งเซิร์ฟเวอร์ โดยเฉพาะอย่างยิ่งในการประมวลผลข้อมูลจากผู้ใช้ เช่น ฟอร์มกรอกข้อมูล การเชื่อมต่อฐานข้อมูล หรือการแสดงผลที่เปลี่ยนแปลงตามเซิร์ฟเวอร์ ทั้งยังสามารถทำงานร่วมกับ Servlet ได้ โดย Servlet ทำหน้าที่ประมวลผลข้อมูลและ JSP ทำหน้าที่แสดงผล เทคโนโลยีนี้ถูกพัฒนาโดย Sun Microsystems (ปัจจุบันคือ Oracle Corporation) เพื่อแก้ไขปัญหาข้อจำกัดของ CGI และ SSI ในการสร้างเว็บแบบไดนามิกที่ซับซ้อน ทำให้การพัฒนาเว็บแอปพลิเคชันที่ใช้ Java มีความยืดหยุ่นและง่ายขึ้นด้วยการแยกการประมวลผลข้อมูลออกจากการแสดงผล โครงสร้างหลักของ JSP ซึ่งเป็นภาษาสคริปต์ที่ประมวลผลฝั่งเซิร์ฟเวอร์และขยายความสามารถของ Java Servlet นั้น ประกอบด้วย ไดเรกทีฟ (Directive) สำหรับกำหนดค่าการทำงาน เช่น page, include, taglib; แอ็คชั่น (Action) สำหรับจัดเตรียมและดำเนินการในหน้าเพจผ่านแท็กที่กำหนดไว้ล่วงหน้า เช่น <jsp:include>, <jsp:forward> หรือการเรียกใช้ JavaBean; สคริปต์เล็ต (Scriptlet) ซึ่งเป็นส่วนของการเขียนโค้ด Java ภายใน JSP; และ แท็กไลบรารี (Tag Libraries) สำหรับขยายความสามารถของภาษาโดยการกำหนดและเรียกใช้แท็กที่สร้างขึ้นเองได้

บทที่ 2

คำสั่งของเจเอสพี

(JSP Commands)

เนื้อหา

- องค์ประกอบหลักของไฟล์ JSP
- คำสั่งใน JSP (JavaServer Pages)
- Directives (คำสั่งกำกับไฟล์ JSP)
- Declarations (ตัวแปรและเมธอดของ Java)
- Scriptlets (โค้ด Java ฝังใน JSP)
- Expressions (แสดงค่าของตัวแปรหรือเมธอด)
- Implicit Objects (อ็อบเจ็กต์ที่สามารถเรียกใช้ได้โดยตรง)
- Standard Actions (คำสั่งมาตรฐานของ JSP)
- JSTL (JavaServer Pages Standard Tag Library)
- Expression Language (EL)
- Custom Tags (แท็กที่สร้างขึ้นเอง)
- ตัวอย่างเพิ่มเติม ของคำสั่งต่างๆ ใน JSP
- คำสั่ง JSP ที่สามารถใช้งานใน NetBeans IDE รันบน Glassfish Web Server
- JSTL (JavaServer Pages Standard Tag Library)
- JavaBeans, JSTL, EL (Expression Language) และ การจัดการกับข้อผิดพลาด
- คำสั่งรับข้อมูลใน JSP (JSP Input Handling) และรายละเอียดแต่ละประเภท
- เพิ่มวิธีการรับข้อมูลใน JSP อย่างละเอียดเพิ่มเติม
- โปรเจ็กต์ JSP ที่มี Homepage ใช้ NetBeans + GlassFish Web Server

ในยุคที่เทคโนโลยีเว็บมีการพัฒนาอย่างรวดเร็ว การสร้างเว็บไซต์แบบไดนามิกและตอบสนองความต้องการของผู้ใช้อย่างมีประสิทธิภาพเป็นสิ่งจำเป็น JSP (JavaServer Pages) จึงกลายเป็นเครื่องมือสำคัญในการพัฒนาเว็บแอปพลิเคชันบนพื้นฐานของภาษา Java โดยเฉพาะเมื่อใช้งานร่วมกับเครื่องมือพัฒนาอย่าง NetBeans IDE และ GlassFish Web Server ผู้พัฒนาเว็บสามารถสร้างระบบที่มีโครงสร้างดี มีความปลอดภัย และสามารถขยายตัวได้ในระยะยาว หนังสือเล่มนี้จึงมุ่งเน้นการอธิบายองค์ประกอบหลักของ JSP อย่างเป็นระบบ พร้อมตัวอย่างที่สามารถประยุกต์ใช้ได้จริงในโปรเจกต์

เนื้อหาในเล่มจะเริ่มต้นด้วยการแนะนำ **องค์ประกอบหลักของไฟล์ JSP** ซึ่งรวมถึง Directives หรือคำสั่งกำกับไฟล์ที่ใช้กำหนดลักษณะของหน้า JSP, Declarations ที่ใช้สร้างตัวแปรและเมธอด Java ภายในหน้า JSP, Scriptlets ที่ฝังโค้ด Java ไว้ในส่วนของ HTML, และ Expressions ที่ใช้แสดงค่าผลลัพธ์จากตัวแปรหรือเมธอดโดยตรง โดยเนื้อหาทั้งหมดนี้จะถูกอธิบายอย่างละเอียดและเสริมด้วยตัวอย่างที่สามารถทดสอบได้ใน NetBeans

นอกจากนี้ ยังได้กล่าวถึงการใช้งาน **Implicit Objects** ที่ JSP เตรียมไว้ให้ใช้งานโดยอัตโนมัติ เช่น request, response, session, application เป็นต้น รวมถึง **Standard Actions** อย่าง <jsp:include> และ <jsp:forward> ที่ช่วยในการควบคุมโฟลว์ของเว็บเพจได้อย่างยืดหยุ่น เนื้อหาจะขยายความไปถึงการใช้ **JSTL (JavaServer Pages Standard Tag Library)** และ **Expression Language (EL)** เพื่อเขียนโค้ดที่อ่านง่าย แยกส่วนการประมวลผลออกจากส่วนแสดงผล และลดการใช้ Scriptlets ที่มักทำให้โค้ดยุ่งยากในการบำรุงรักษา

อีกส่วนสำคัญที่ไม่อาจมองข้ามคือ **Custom Tags** หรือการสร้างแท็ก JSP ขึ้นเอง ซึ่งช่วยให้สามารถกำหนดพฤติกรรมซ้ำ ๆ ในเว็บเพจอย่างเป็นระบบ รวมถึงการใช้ **JavaBeans** ร่วมกับ JSTL และ EL เพื่อแยกตรรกะธุรกิจออกจากการแสดงผล และยังมีเนื้อหาเกี่ยวกับการจัดการข้อผิดพลาด (Exception Handling) ใน JSP เพื่อให้ระบบมีความเสถียรและปลอดภัยยิ่งขึ้น ทั้งยังกล่าวถึงเทคนิคการรับข้อมูลจากผู้ใช้ด้วยวิธีต่างๆ ทั้งแบบ GET, POST, ผ่าน Form Elements หรือแม้แต่ Ajax บน JSP

นอกจากนั้นยังได้รวมตัวอย่างโปรเจกต์ JSP ที่มี **Homepage แบบครบวงจร** พัฒนาด้วย **NetBeans IDE** และ **GlassFish Web Server** พร้อมทั้งคำอธิบายแบบ Step-by-Step ตั้งแต่การสร้างหน้าแรก การรับข้อมูล การประมวลผล ไปจนถึงการแสดงผลแบบมีเงื่อนไขและมีโครงสร้าง MVC ครบถ้วน โดยหวังว่าผู้อ่านจะสามารถนำความรู้จากหนังสือเล่มนี้ไปประยุกต์ใช้ในการพัฒนาเว็บแอปพลิเคชัน JSP ได้อย่างมีประสิทธิภาพ

องค์ประกอบหลักของไฟล์ JSP

ไฟล์ .jsp ประกอบไปด้วยส่วนสำคัญดังต่อไปนี้:

1. **Directives (คำสั่งกำกับไฟล์ JSP)**
2. **Declarations (ตัวแปรและเมธอดของ Java)**
3. **Scriptlets (โค้ด Java ฝังใน JSP)**
4. **Expressions (แสดงค่าของตัวแปรหรือเมธอด)**
5. **Implicit Objects (อ็อบเจ็กต์ที่สามารถเรียกใช้ได้โดยตรง)**
6. **Standard Actions (คำสั่งมาตรฐานของ JSP)**
7. **JSTL (JavaServer Pages Standard Tag Library)**
8. **Expression Language (EL)**

9. **Custom Tags (แท็กที่สร้างขึ้นเอง)**

---

1. Directives (คำสั่งกำกับไฟล์ JSP)

Directives เป็นคำสั่งที่ใช้กำหนดคุณสมบัติของไฟล์ JSP เช่น การกำหนดประเภทของไฟล์, การนำเข้าแพ็กเกจ Java, และการตั้งค่าต่าง ๆ

รูปแบบของ Directives

<%@ directive attribute="value" %>

ประเภทของ Directives

1. **Page Directive** – กำหนดคุณสมบัติของไฟล์ JSP
2. <%@ page language="java" contentType="text/html; charset=UTF-8" %>
3. **Include Directive** – ใช้รวมไฟล์ JSP หรือ HTML อื่นเข้ามา
4. <%@ include file="header.jsp" %>
5. **Taglib Directive** – ใช้กำหนดไลบรารีแท็กที่ต้องการใช้งาน
6. <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

---

2. Declarations (ตัวแปรและเมธอดของ Java)

ใช้สำหรับประกาศตัวแปรและเมธอดใน JSP ซึ่งจะถูกสร้างเป็นฟิลด์หรือเมธอดในคลาสที่ถูกคอมไพล์

รูปแบบของ Declarations

<%! dataType variableName; %>

<%! returnType methodName() { ... } %>

ตัวอย่าง

<%! int count = 0; %>

<%! int increment() { return ++count; } %>

---

3. Scriptlets (โค้ด Java ฝังใน JSP)

ใช้เขียนโค้ด **Java** ที่ต้องการให้รันภายในไฟล์ JSP

รูปแบบของ Scriptlets

<% Java code %>

ตัวอย่าง

```
<%
    String name = "John";
    int age = 25;
%>
<p>Name: <%= name %></p>
```

Age: <%= age %>

---

4. Expressions (แสดงค่าของตัวแปรหรือเมธอด)

ใช้แสดงผลค่าของตัวแปรหรือเมธอดโดยตรงในหน้าเว็บ

รูปแบบของ Expressions

<%= expression %>

ตัวอย่าง

Current Time: <%= new java.util.Date() %>

---

5. Implicit Objects (อ็อบเจ็กต์ที่ใช้ได้โดยตรง)

JSP มี **implicit objects** ที่สามารถเรียกใช้ได้โดยไม่ต้องประกาศ ซึ่งรวมถึง:

- request ⟶ ใช้ดึงข้อมูลที่ส่งมาจาก **client**
- response ⟶ ใช้กำหนดค่าที่จะส่งกลับไปยัง **client**
- session ⟶ ใช้จัดเก็บข้อมูลของผู้ใช้ขณะยังอยู่ใน session
- application ⟶ ใช้จัดเก็บข้อมูลของแอปพลิเคชัน
- out ⟶ ใช้แสดงผลข้อมูลไปยัง **client**

ตัวอย่าง

Client IP: <%= request.getRemoteAddr() %>

---

6. Standard Actions (คำสั่งมาตรฐานของ JSP)

JSP มีชุดคำสั่งที่ช่วยให้สามารถทำงานร่วมกับ **JavaBeans**, **Forward Requests**, และ **รวมไฟล์ JSP อื่น ๆ**

ตัวอย่างคำสั่ง Standard Actions

1. **ใช้ JavaBean**
2. <jsp:useBean id="user" class="com.example.User" scope="session"/>
3. <jsp:setProperty name="user" property="name" value="John Doe"/>
4. <jsp:getProperty name="user" property="name"/>
5. **Redirect ไปยังไฟล์อื่น**
6. <jsp:forward page="home.jsp"/>

---

7. JSTL (JavaServer Pages Standard Tag Library)

JSTL คือไลบรารีแท็กมาตรฐานของ JSP ที่ช่วยลดการใช้ **Java code** ภายใน JSP และทำให้โค้ดสะอาดขึ้น

ตัวอย่าง

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

<c:if test="${param.username == 'admin'}">
    <p>Welcome, Admin!</p>
</c:if>
```

---

8. Expression Language (EL)

Expression Language หรือ **EL** เป็นวิธีที่ง่ายกว่าในการเข้าถึงข้อมูลใน **request, session, application** โดยไม่ต้องใช้ Java code

ตัวอย่าง

```
<p>Username: ${sessionScope.user}</p>
<p>Product Price: ${product.price}</p>
```

---

9. Custom Tags (แท็กที่สร้างขึ้นเอง)

เราสามารถสร้าง **Custom Tag** ขึ้นมาเองได้โดยใช้ **Tag Library Descriptor (TLD)** และ **Java Class**

ตัวอย่าง

```
<%@ taglib prefix="myTag" uri="WEB-INF/myTags.tld" %>
<myTag:hello name="John"/>
```

สรุป

องค์ประกอบ	คำอธิบาย	ตัวอย่าง
Directives	คำสั่งกำกับไฟล์ JSP	<%@ page language="java" %>
Declarations	ประกาศตัวแปร/เมธอด	<%! int count = 0; %>
Scriptlets	ฝังโค้ด Java	<% int x = 10; %>
Expressions	แสดงค่าตัวแปร	<%= new java.util.Date() %>
Implicit Objects	อ็อบเจ็กต์ใน JSP	<%= request.getRemoteAddr() %>
Standard Actions	คำสั่งมาตรฐาน JSP	<jsp:forward page="home.jsp"/>
JSTL	ไลบรารีแท็กของ JSP	<c:if test="${user == 'admin'}">
Expression Language (EL)	ใช้แสดงค่าตัวแปร	${sessionScope.user}
Custom Tags	แท็กที่สร้างขึ้นเอง	<myTag:hello name="John"/>

องค์ประกอบทั้งหมดนี้เป็นพื้นฐานของ **JSP** ที่สามารถนำไปใช้พัฒนา **Web Application** ที่มีความซับซ้อนขึ้นได้

คำสั่งใน JSP (JavaServer Pages)

คำสั่งใน **JSP (JavaServer Pages)** สามารถแบ่งออกเป็น 3 ประเภทหลักๆ ได้แก่ **Directives**, **Declarations**, และ **Scriplets** ซึ่งแต่ละประเภทมีการใช้งานและจุดประสงค์ที่แตกต่างกันไป เพื่อช่วยในการเขียนโค้ด JSP ที่มีประสิทธิภาพและสามารถทำงานได้ตามความต้องการ ต่อไปนี้จะอธิบายแต่ละประเภทพร้อมตัวอย่างโดยละเอียด:

1. Directives (คำสั่งกำหนดทิศทาง)

Directives ใช้สำหรับการกำหนดค่าคอนฟิกหรือการตั้งค่าต่างๆ ที่จะมีผลในระดับของทั้งหน้า JSP โดยทั่วไปจะใช้ในการกำหนดค่า เช่น การตั้งค่า **content type**, การกำหนด **import** class, หรือการตั้งค่า **tag libraries** เป็นต้น

ประเภทของ Directives:

- **Page Directive** (<%@ page ... %>)
- **Include Directive** (<%@ include ... %>)
- **Taglib Directive** (<%@ taglib ... %>)

1.1 Page Directive

คำสั่ง **Page Directive** ใช้ในการตั้งค่าหรือปรับแต่งพฤติกรรมของหน้า JSP เช่น กำหนด **content type**, **language**, หรือ **import** ไฟล์ Java

ตัวอย่าง:

```
<%@ page contentType="text/html;charset=UTF-8" language="java" import="java.util.*" %>
<html>
    <head>
        <title>Page Directive Example</title>
    </head>
    <body>
        <h1>Welcome to the JSP page with Page Directive</h1>
    </body>
</html>
```

ในตัวอย่างนี้:

- **contentType** กำหนดรูปแบบของเนื้อหาของหน้าเป็น **HTML** พร้อมด้วย **charset=UTF-8**
- **language** กำหนดว่าใช้ภาษา **Java** ในการเขียนโค้ด
- **import** ใช้เพื่อ **import** คลาสจาก **Java API** ที่ต้องการใช้งานในหน้า JSP

1.2 Include Directive

คำสั่ง **Include Directive** ใช้สำหรับการรวมไฟล์อื่นเข้ามาในหน้า JSP ตัวนี้จะช่วยให้สามารถรวมไฟล์ที่ต้องการซ้ำในหลายๆ หน้า เช่น ฟุตเตอร์หรือเฮดเดอร์

ตัวอย่าง:

```
<%@ include file="header.jsp" %>
<html>
    <body>
        <h2>This is the body of the page</h2>
    </body>
</html>
<%@ include file="footer.jsp" %>
```

ในตัวอย่างนี้จะรวม **header.jsp** และ **footer.jsp** เข้าไปในไฟล์ JSP หลัก โดยที่โค้ดในไฟล์ที่ถูก **include** จะถูกนำมาวางตรงจุดที่คำสั่ง **include** ถูกเขียน

1.3 Taglib Directive

คำสั่ง **Taglib Directive** ใช้สำหรับการประกาศ **JSP tag libraries** ซึ่งจะช่วยให้เราสามารถใช้งานแท็กที่กำหนดเองได้ใน JSP

ตัวอย่าง:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
    <body>
        <h2>Item List:</h2>
        <c:forEach var="item" items="${items}">
            <p>${item}</p>
        </c:forEach>
    </body>
</html>
```

ในตัวอย่างนี้:

- **taglib** จะประกาศการใช้งาน **JSTL core library** โดยที่ **prefix** เป็น **c**
- ใช้ **<c:forEach>** ในการวนลูปแสดงรายการของ **items**

2. Declarations (คำสั่งการประกาศ)

Declarations ใช้ในการประกาศตัวแปรหรือเมธอดใน JSP ซึ่งการประกาศในส่วนนี้จะมีผลในระดับของทั้งหน้า JSP และสามารถใช้ในการเขียนโค้ดที่ซับซ้อน เช่น การประกาศฟังก์ชันหรือคลาสในหน้า JSP

ตัวอย่าง:

```
<%!
```

```
    public String greet(String name) {
        return "Hello, " + name + "!";
    }
%>

<html>
    <body>
        <h2><%= greet("John") %></h2>
    </body>
</html>
```

ในตัวอย่างนี้:

- **<%! %>** ใช้ในการประกาศ **Java method** ที่สามารถเรียกใช้ในหน้า JSP ได้
- เมธอด **greet(String name)** จะรับชื่อผู้ใช้และแสดงข้อความทักทาย
- ในส่วนของ **<%= %>** จะใช้เรียกเมธอด **greet** และแสดงผลในหน้า JSP

3. Scriplets (คำสั่งสคริปต์)

Scriplets ใช้ในการเขียนโค้ด **Java** ภายใน JSP โดยโค้ดใน **scriplet** จะถูกประมวลผลในระหว่างการทำงานของ JSP และสามารถใช้ในการคำนวณหรือประมวลผลต่างๆ ที่จำเป็นต้องทำในหน้าดังกล่าว

ตัวอย่าง:

```
<html>
    <head>
        <title>Scriplet Example</title>
    </head>
    <body>
        <h2>
            <%
                int a = 5;
                int b = 10;
                int sum = a + b;
                out.println("The sum of " + a + " and " + b + " is " + sum);
            %>
        </h2>
    </body>
</html>
```

ในตัวอย่างนี้:

- ใช้ **scriplet** ในการคำนวณผลรวมของตัวเลขสองตัว และแสดงผลลัพธ์ออกมาบนหน้าเว็บ
- โค้ดที่อยู่ใน **<% %>** จะถูกประมวลผลก่อนที่จะส่งไปยังผู้ใช้

สรุป

1. **Directives** (คำสั่งกำหนดทิศทาง): ใช้สำหรับการกำหนดค่าต่างๆ ที่มีผลในระดับของทั้งหน้า JSP เช่น **page**, **include**, และ **taglib**
2. **Declarations** (คำสั่งการประกาศ): ใช้ในการประกาศตัวแปรหรือเมธอดที่สามารถใช้ในหน้า JSP
3. **Scriplets** (คำสั่งสคริปต์): ใช้ในการเขียนโค้ด **Java** ภายใน JSP เพื่อประมวลผลหรือคำนวณค่าต่างๆ ที่ต้องการในหน้าเว็บ

แต่ละประเภทมีบทบาทและจุดประสงค์ที่แตกต่างกันไปในการช่วยพัฒนาแอปพลิเคชัน JSP ให้ง่ายและมีประสิทธิภาพมากขึ้น

ใน **JSP (JavaServer Pages)** มีคำสั่ง (commands) ที่สามารถใช้ได้หลากหลาย ซึ่งแต่ละคำสั่งมีประโยชน์ในการจัดการกับข้อมูล, แสดงผล, และการทำงานต่างๆ ภายในหน้า JSP ต่อไปนี้คือ **command list** ของ JSP พร้อมประโยชน์และตัวอย่างการใช้งาน:

1. Directive Commands

คำสั่งในประเภทนี้ใช้สำหรับการกำหนดค่าคอนฟิกหรือการตั้งค่าพฤติกรรมของหน้า JSP โดยจะมีผลในระดับของทั้งไฟล์ JSP

1.1 Page Directive

ประโยชน์: ใช้ในการตั้งค่าคอนฟิกทั่วไปของหน้า JSP เช่น **contentType**, **language**, **import**, หรือ **buffering**

ตัวอย่าง:

```
<%@ page contentType="text/html;charset=UTF-8" language="java" import="java.util.*" %>
<html>
    <head>
        <title>Page Directive Example</title>
    </head>
    <body>
        <h1>Page Directive Example</h1>
    </body>
</html>
```

- **contentType**: กำหนดประเภทของข้อมูล (MIME type) ของหน้า เช่น **text/html** และ **charset=UTF-8**

- **language**: กำหนดภาษาที่ใช้ในหน้า JSP (ในที่นี้คือ **Java**)
- **import**: ใช้ในการนำเข้า **Java classes** ที่ต้องการใช้งานใน JSP

1.2 Include Directive

ประโยชน์: ใช้สำหรับการรวมไฟล์ JSP อื่นเข้ามาในหน้า JSP นี้ ซึ่งมักใช้สำหรับการรวมฟุตเตอร์หรือเฮดเดอร์ที่ซ้ำกันในหลายๆ หน้า

ตัวอย่าง:

```
<%@ include file="header.jsp" %>
<html>
    <body>
        <h1>Main Content</h1>
    </body>
</html>
<%@ include file="footer.jsp" %>
```

ในตัวอย่างนี้จะรวมไฟล์ **header.jsp** และ **footer.jsp** เข้ามาในหน้า JSP หลัก

1.3 Taglib Directive

ประโยชน์: ใช้เพื่อประกาศการใช้งาน **JSP tag libraries** (เช่น **JSTL**) ที่ช่วยให้สามารถใช้แท็กที่กำหนดเองได้ใน JSP

ตัวอย่าง:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
    <body>
        <c:set var="items" value="${['Apple', 'Banana', 'Cherry']}" />
        <c:forEach var="item" items="${items}">
            <p>${item}</p>
        </c:forEach>
    </body>
</html>
```

- **taglib** ประกาศการใช้งาน **JSTL** core library โดยที่ **prefix** เป็น **c**
- ใช้แท็ก **<c:forEach>** เพื่อวนลูปรายการผลไม้

---

2. Scripting Elements

ประโยชน์: ใช้ในการเขียนโค้ด **Java** ภายในหน้า JSP เพื่อทำการประมวลผลข้อมูลที่ต้องการหรือการแสดงผล

2.1 Scriplet

ประโยชน์: ใช้ในการเขียนโค้ด **Java** ที่จะประมวลผลและแสดงผลในหน้า JSP

ตัวอย่าง:

```
<html>
    <body>
        <h2>
            <%
                int a = 5;
                int b = 10;
                int sum = a + b;
                out.println("The sum of " + a + " and " + b + " is " + sum);
            %>
        </h2>
    </body>
</html>
```

- **Scriplet** ใช้ในการคำนวณผลรวมของ **a** และ **b** และแสดงผลในหน้า JSP

2.2 Expression

ประโยชน์: ใช้สำหรับแสดงค่าผลลัพธ์จากการประมวลผลในหน้า JSP โดยไม่ต้องใช้คำสั่ง **out.println()**

ตัวอย่าง:

```
<html>
    <body>
        <h2>The sum is: <%= 5 + 10 %></h2>
    </body>
</html>
```

- ใช้ **expression** เพื่อแสดงผลลัพธ์ของ **5 + 10** บนหน้าเว็บ

2.3 Declaration

ประโยชน์: ใช้ในการประกาศตัวแปรหรือเมธอดที่สามารถใช้ในหน้า JSP ได้

ตัวอย่าง:

```
<%!
    public String greet(String name) {
        return "Hello, " + name + "!";
    }
%>
<html>
    <body>
```

```
    <h2><%= greet("John") %></h2>
  </body>
</html>
```

- **Declaration** ใช้ในการประกาศเมธอด **greet** และเรียกใช้งานภายในหน้า JSP

---

3. Action Elements

ประโยชน์: ใช้ในการส่งคำสั่งให้กับ JSP engine เพื่อทำงานบางอย่าง เช่น การเปลี่ยนหน้า, ส่งข้อมูลไปยัง Servlet, หรือใช้ tag library

3.1 jsp:include

ประโยชน์: ใช้ในการรวมเนื้อหาจากไฟล์อื่นไปยังหน้า JSP (คล้ายกับ **include directive** แต่ใช้ในรูปแบบของ action element)

ตัวอย่าง:

```
<jsp:include page="footer.jsp" />
```

- คำสั่งนี้จะรวมเนื้อหาจาก **footer.jsp** เข้ามาในหน้า JSP ที่ใช้งาน

3.2 jsp:forward

ประโยชน์: ใช้ในการเปลี่ยนเส้นทาง (redirect) ไปยังหน้าอื่นโดยที่ไม่ต้องส่งคำตอบกลับไปยังลูกค้าก่อน

ตัวอย่าง:

```
<jsp:forward page="nextPage.jsp" />
```

- คำสั่งนี้จะเปลี่ยนเส้นทางการทำงานไปยัง **nextPage.jsp**

3.3 jsp:param

ประโยชน์: ใช้ในการส่งพารามิเตอร์ไปยัง **jsp:include** หรือ **jsp:forward**

ตัวอย่าง:

```
<jsp:forward page="nextPage.jsp">
  <jsp:param name="userName" value="John" />
</jsp:forward>
```

- คำสั่งนี้จะส่งพารามิเตอร์ **userName** ไปยัง **nextPage.jsp** พร้อมค่าที่เป็น **John**

---

4. Error Handling Elements

ประโยชน์: ใช้ในการจัดการกับข้อผิดพลาดในหน้า JSP

4.1

ประโยชน์: ใช้ในการกำหนดหน้าเว็บที่จะแสดงเมื่อเกิดข้อผิดพลาดขึ้นในแอปพลิเคชัน

ตัวอย่าง:

```
<error-page>
    <exception-type>java.lang.Exception</exception-type>
    <location>/error.jsp</location>
</error-page>
```

- หากเกิดข้อผิดพลาด **Exception** จะส่งไปที่ **error.jsp**

สรุป

Command List ของ JSP รวมถึงคำสั่งต่างๆ ที่ใช้ในการตั้งค่าการทำงานใน JSP โดยสามารถแบ่งได้เป็นหลายประเภท เช่น:

- **Directives**: ใช้ในการกำหนดค่าทั่วไปของหน้า JSP
- **Scripting Elements**: ใช้ในการเขียนโค้ด Java ในหน้า JSP
- **Action Elements**: ใช้ในการส่งคำสั่งพิเศษเช่นการรวมหน้า หรือการเปลี่ยนเส้นทาง
- **Error Handling**: ใช้ในการจัดการข้อผิดพลาดที่เกิดขึ้นในหน้า JSP

ทุกคำสั่งมีประโยชน์และสามารถช่วยเพิ่มความยืดหยุ่นในการพัฒนาเว็บแอปพลิเคชันด้วย JSP

ตัวอย่างเพิ่มเติม ของคำสั่งต่าง ๆ ใน JSP

ต่อไปนี้คือ **ตัวอย่างเพิ่มเติม** ของคำสั่งต่างๆ ใน **JSP** ที่สามารถใช้ในการพัฒนาแอปพลิเคชัน โดยทุกคำสั่งมีประโยชน์ในการจัดการเนื้อหาและการทำงานของหน้า JSP:

1. jsp:useBean

ประโยชน์: ใช้สำหรับการสร้างหรือเข้าถึง **JavaBean** ที่กำหนดไว้ เพื่อใช้งานในหน้า JSP

ตัวอย่าง:

```
<jsp:useBean id="person" class="com.example.Person" scope="session" />
<html>
    <body>
        <h2>Name: <jsp:getProperty name="person" property="name" /></h2>
    </body>
</html>
```

- คำสั่งนี้จะสร้าง **JavaBean** ที่ชื่อว่า **person** และแสดงค่า **name** จาก Bean ในหน้าเว็บ

2. jsp:setProperty

ประโยชน์: ใช้ในการตั้งค่าคุณสมบัติของ **JavaBean** ที่กำหนดไว้

ตัวอย่าง:

```
<jsp:useBean id="person" class="com.example.Person" scope="session" />
<jsp:setProperty name="person" property="name" value="John Doe" />
```

```
<html>
    <body>
        <h2>Name: <jsp:getProperty name="person" property="name" /></h2>
    </body>
</html>
```

- ใช้ในการตั้งค่าคุณสมบัติ **name** ของ **JavaBean person** เป็น **John Doe**

3. jsp:getProperty

ประโยชน์: ใช้ในการดึงค่าของคุณสมบัติจาก **JavaBean** ที่กำหนดไว้

ตัวอย่าง:

```
<jsp:useBean id="person" class="com.example.Person" />
<jsp:getProperty name="person" property="name" />
```

- ใช้ในการดึงข้อมูลจาก **JavaBean** เช่น **name** และแสดงในหน้า JSP

4. <c:out> (JSTL)

ประโยชน์: ใช้ในการแสดงค่าผลลัพธ์ที่มีการประมวลผลหรือไม่ต้องการ escape HTML (ป้องกัน Cross-site scripting)

ตัวอย่าง:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
    <body>
        <c:out value="${userInput}" />
    </body>
</html>
```

- คำสั่ง **<c:out>** จะแสดงผลค่าที่อยู่ใน **userInput** พร้อมการ escape อัตโนมัติ

5. <c:if> (JSTL)

ประโยชน์: ใช้ในการตรวจสอบเงื่อนไขและแสดงผลลัพธ์หากเงื่อนไขเป็นจริง

ตัวอย่าง:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
    <body>
        <c:if test="${userLoggedIn}">
            <p>Welcome back, user!</p>
        </c:if>
    </body>
</html>
```

- **<c:if>** ตรวจสอบว่า **userLoggedIn** เป็นจริงหรือไม่ ถ้าเป็นจริงจะถูกแสดงข้อความทักทาย

6. <c:choose>, <c:when>, <c:otherwise> (JSTL)

ประโยชน์: ใช้ในการจัดการเงื่อนไขหลายๆ อันอย่างมีประสิทธิภาพ

ตัวอย่าง:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
    <body>
        <c:choose>
            <c:when test="${userRole == 'admin'}">
                <p>Welcome, admin!</p>
            </c:when>
            <c:when test="${userRole == 'user'}">
                <p>Welcome, user!</p>
            </c:when>
            <c:otherwise>
                <p>Welcome, guest!</p>
            </c:otherwise>
        </c:choose>
    </body>
</html>
```

- **<c:choose>** ใช้ตรวจสอบเงื่อนไขหลายๆ อย่าง เช่น แสดงข้อความต่างๆ ตามบทบาทผู้ใช้

7. <c:forEach> (JSTL)

ประโยชน์: ใช้ในการวนลูปแสดงรายการข้อมูลที่เป็นคอลเล็กชัน

ตัวอย่าง:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
    <body>
        <ul>
            <c:forEach var="item" items="${itemList}">
                <li>${item}</li>
            </c:forEach>
        </ul>
    </body>
</html>
```

- **<c:forEach>** ใช้ในการวนลูปแสดงรายการ **itemList**

8. jsp:plugin

ประโยชน์: ใช้ในการสร้าง **Java applet** หรือ **plugin** ในหน้า JSP

ตัวอย่าง:

```
<jsp:plugin type="applet" code="com.example.MyApplet.class" width="300" height="300">
    <param name="param1" value="value1" />
</jsp:plugin>
```

- ใช้ในการฝัง **Java applet** ในหน้าเว็บ

9. jsp:body

ประโยชน์: ใช้ในการสร้างส่วนที่สามารถแทนที่ได้ใน **custom tag** เมื่อมีการใช้งาน **tag handler**

ตัวอย่าง:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
    <body>
        <c:forEach var="item" items="${items}">
            <jsp:body>
                <p>${item}</p>
            </jsp:body>
        </c:forEach>
    </body>
</html>
```

- ใช้ในการจัดการเนื้อหาภายในส่วนของแท็กที่สามารถแทนที่ได้

10. jsp:throw

ประโยชน์: ใช้ในการโยน **exception** เพื่อจัดการข้อผิดพลาดในหน้า JSP

ตัวอย่าง:

```
<%@ page errorPage="error.jsp" %>
<jsp:throw exception="java.lang.Exception" />
```

- ใช้ในการโยน **exception** เพื่อให้ **errorPage** รับผิดชอบการจัดการข้อผิดพลาด

สรุป

คำสั่งต่างๆ ที่ใช้ใน JSP นั้นมีหลากหลายประเภทและใช้เพื่อการจัดการข้อมูลในหน้า JSP เช่น:

- การจัดการกับ **JavaBeans** เช่น **jsp:useBean**, **jsp:setProperty**, **jsp:getProperty**
- การแสดงข้อมูลและตรวจสอบเงื่อนไขต่างๆ ด้วย **JSTL** เช่น **<c:out>**, **<c:forEach>**, **<c:if>**
- การสร้าง **plugin** หรือ **applet** เช่น **jsp:plugin**

- การจัดการข้อผิดพลาดด้วย **jsp:throw**

ทุกคำสั่งมีบทบาทในการทำให้การพัฒนาเว็บด้วย JSP ง่ายขึ้นและยืดหยุ่นมากขึ้น

คำสั่ง JSP ที่สามารถใช้งานใน NetBeans IDE รันบน Glassfish Web Server

ต่อไปนี้คือตัวอย่าง **คำสั่ง JSP** ที่สามารถใช้งานใน **NetBeans IDE** รันบน **Glassfish Web Server** อีก 10 ตัวอย่างที่สามารถนำไปใช้ในโครงการของคุณ:

1. jsp:forward

ประโยชน์: ใช้ในการเปลี่ยนเส้นทาง (forward) ไปยังหน้าต่างๆ โดยไม่ต้องส่งคำตอบกลับไปยังผู้ใช้ก่อน

ตัวอย่าง:

```
<jsp:forward page="nextPage.jsp" />
```

- คำสั่งนี้จะเปลี่ยนเส้นทางไปยังหน้า **nextPage.jsp**

2. jsp:include

ประโยชน์: ใช้ในการรวมเนื้อหาของหน้า JSP อื่นเข้ามาในหน้า JSP นี้

ตัวอย่าง:

```
<jsp:include page="header.jsp" />
```

- คำสั่งนี้จะรวมเนื้อหาของ **header.jsp** เข้ามาในหน้า JSP หลัก

3. jsp:useBean

ประโยชน์: ใช้ในการสร้างและเข้าถึง **JavaBean** เพื่อให้สามารถใช้งานในหน้า JSP ได้

ตัวอย่าง:

```
<jsp:useBean id="user" class="com.example.User" scope="session" />
<jsp:setProperty name="user" property="username" value="JohnDoe" />
<p>User name: <jsp:getProperty name="user" property="username" /></p>
```

- คำสั่งนี้จะสร้าง **JavaBean** ที่ชื่อ **user** และตั้งค่า **username** เป็น "JohnDoe"

4. jsp:setProperty

ประโยชน์: ใช้ในการตั้งค่าคุณสมบัติของ **JavaBean**

ตัวอย่าง:

```
<jsp:useBean id="person" class="com.example.Person" scope="session" />
<jsp:setProperty name="person" property="name" value="Alice" />
<p>Person's Name: <jsp:getProperty name="person" property="name" /></p>
```

- คำสั่งนี้จะตั้งค่าคุณสมบัติ **name** ของ **person** เป็น "Alice"

5. jsp:getProperty

ประโยชน์: ใช้ในการดึงค่าคุณสมบัติจาก **JavaBean**

ตัวอย่าง:

```
<jsp:useBean id="product" class="com.example.Product" />
<p>Product Name: <jsp:getProperty name="product" property="name" /></p>
```

- คำสั่งนี้จะดึงค่าของ **name** จาก **JavaBean** ที่ชื่อว่า **product**

6. jsp:plugin

ประโยชน์: ใช้ในการฝัง **Java applet** หรือ **plugin** ในหน้า JSP

ตัวอย่าง:

```
<jsp:plugin type="applet" code="com.example.MyApplet.class" width="300" height="300">
    <param name="param1" value="value1" />
</jsp:plugin>
```

- ใช้ฝัง **Java applet** ลงในหน้า JSP

7. <c:out> (JSTL)

ประโยชน์: ใช้ในการแสดงค่าผลลัพธ์จากการประมวลผลหรือค่าจากตัวแปรต่างๆ

ตัวอย่าง:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<p>Welcome, <c:out value="${userName}" />!</p>
```

- คำสั่ง **<c:out>** ใช้ในการแสดงผลค่าของ **userName**

8. <c:forEach> (JSTL)

ประโยชน์: ใช้ในการวนลูปแสดงรายการข้อมูลจาก **List** หรือ **Array**

ตัวอย่าง:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<ul>
    <c:forEach var="item" items="${itemList}">
        <li>${item}</li>
    </c:forEach>
</ul>
```

- คำสั่ง **<c:forEach>** จะวนลูปรายการใน **itemList** และแสดงในรูปแบบของรายการ

9. <c:if> (JSTL)

ประโยชน์: ใช้ในการแสดงเนื้อหาขึ้นอยู่กับเงื่อนไขที่กำหนด

ตัวอย่าง:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<c:if test="${userLoggedIn}">
```