프로젝트로 배우는

자바 웹 프로그래밍

Servlet, JSP, JDBC

황희정 지음

Chapter 05. JSP 기본 문법

한빛아카데미 Hanbit Academy, Inc.

목차

학습목표

- JSP의 구조적 특징과 기본 문법을 익힌다.
- 지시어의 종류를 살펴보고 page 지시어의 주요 속성의 의미를 알아본다.
- 선언문, 표현식, 스크립트릿과 같은 스크립팅 요소를 배운다.

01. 주석

주석 이란 ?

- 주석은 프로그램 소스에 텍스트로 된 간단한 설명문을 넣는 것을 말한다.
- C 언어를 비롯한 대부분의 프로그램 언어가 주석을 사용하기 위한 문법을 제공하고 있다.
- JSP는 특성상 자바, HTML, JSP 코드가 섞여 있으므로 주석도 혼용해서 사용한다.
- **HTML 주석 : 클라이언트로 전달되는 주석**
  - 일반적인 HTML 문서에서 사용 가능한 주석으로 화면에는 보이지 않지만 브라우저 소스보기를 하면 내용이 노출됨.

```
<!-- 주석입니다. -->
```

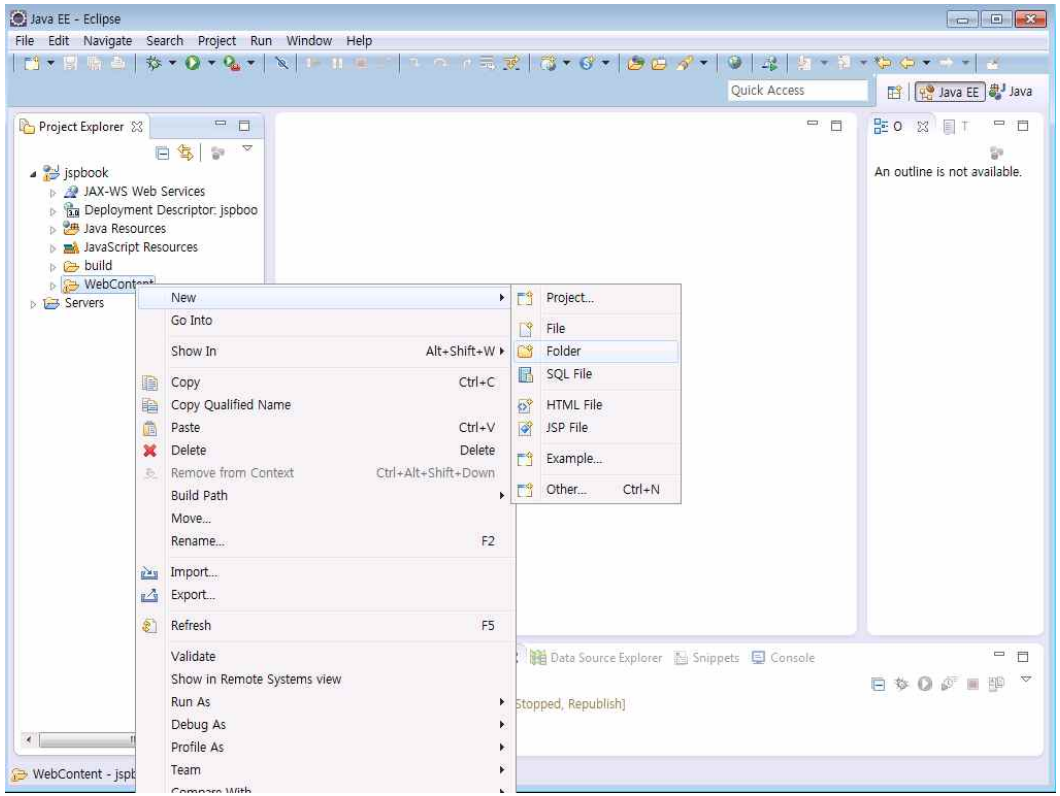
- **JSP 주석 : 클라이언트로 전달되지 않는 주석**
  - JSP 파일에서만 사용 가능한 주석으로 브라우저 소스보기를 해도 내용이 노출되지 않음.

```
<%-- 주석  --%>
```

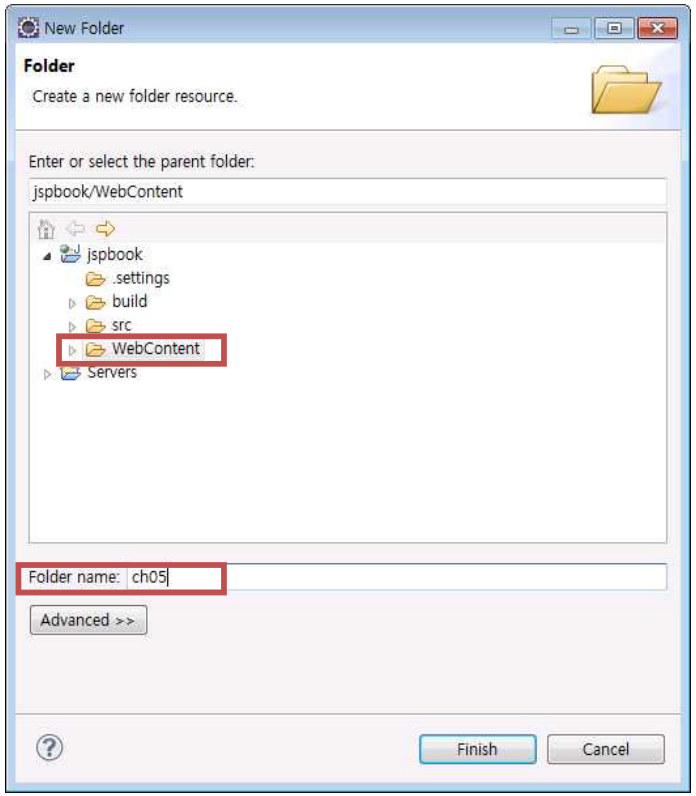
1. 이클립스를 이용한 소스 작성과 실행

■ 이클립스에서 작업 준비하기

❶ [WebContent] → [ch05] 폴더를 만든다.



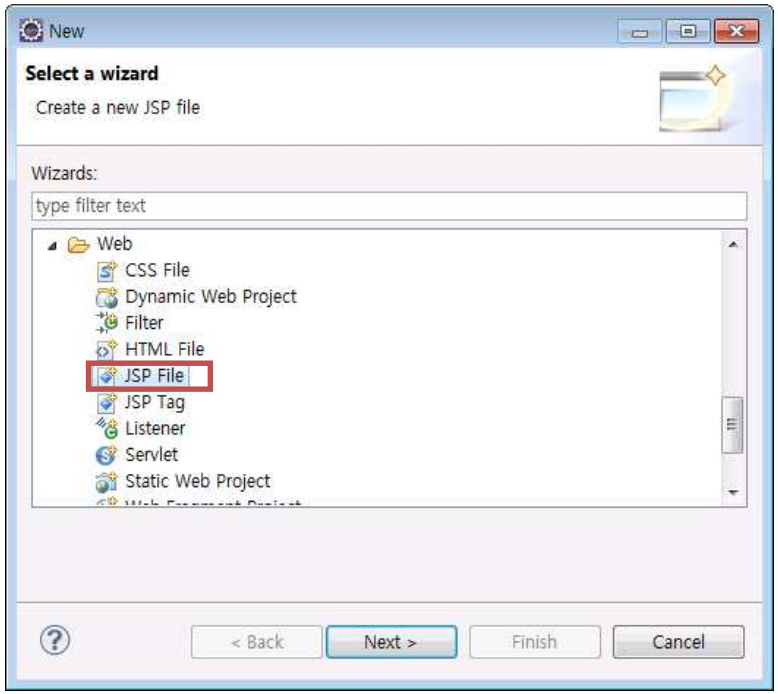
[그림 5-1] 기존 프로젝트에 새로운 폴더 생성



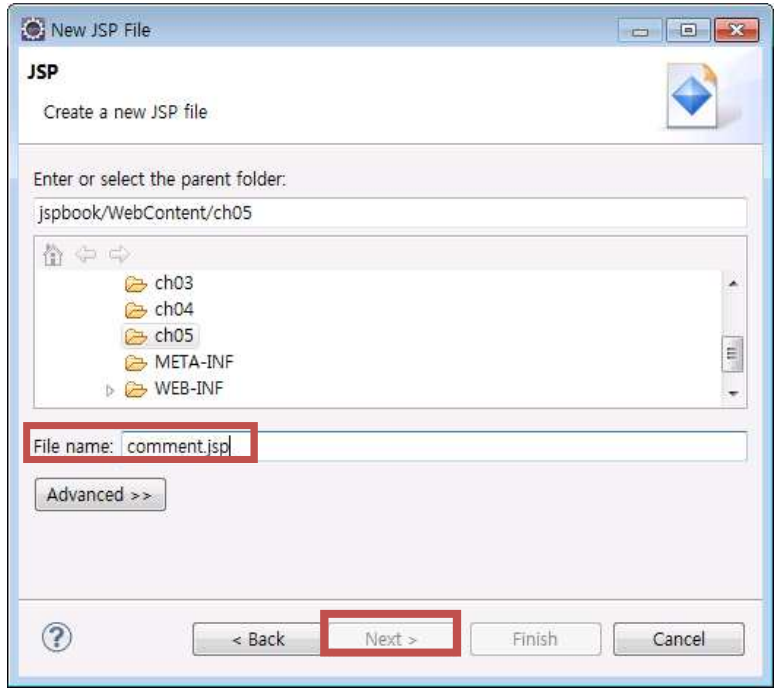
[그림 5-2] 폴더 생성 디렉터리 지정

01. 주석

❷ [File] → [New] → [Other]를 선택한 후 [Web] 아래의 [JSP File]를 선택한 뒤 <Next>버튼을 클릭한다. 그리고 [File name] 입력란에 comment.jsp를 입력하고 <Next> 버튼을 클릭한다.



[그림 5-3] JSP 마법사 선택



[그림 5-4] comment.jsp 파일 저장

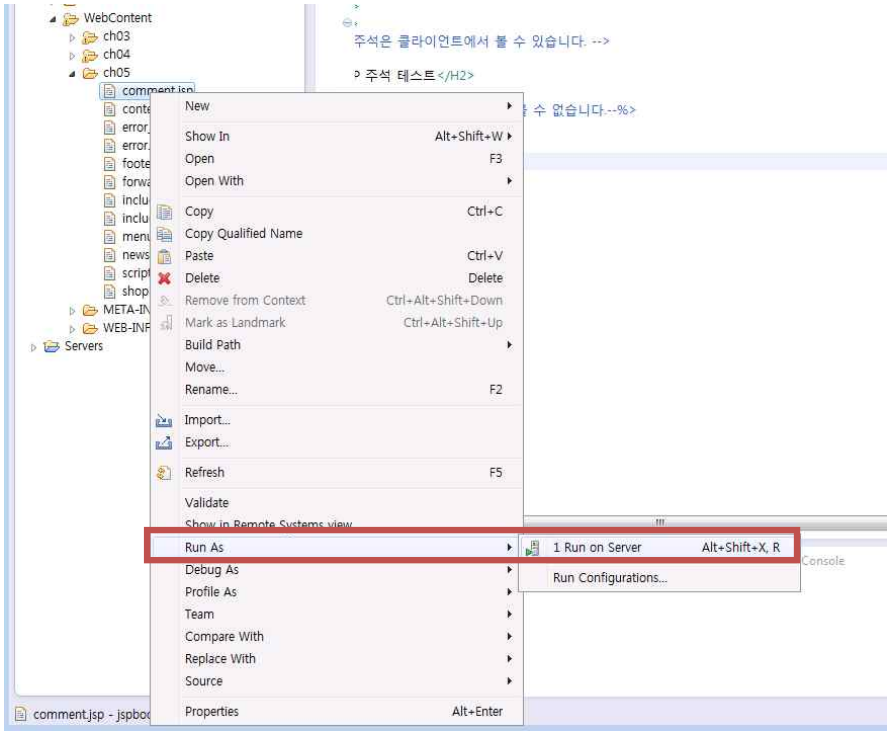
❸ JSP 템플릿을 선택하는 화면이 나오면 기본 선택 값을 사용하고 <Finish> 버튼을 눌러 종료한다.

■ 소스코드 작성하기

- 주석 비교하기(comment.jsp) - 교재 p.158 참고

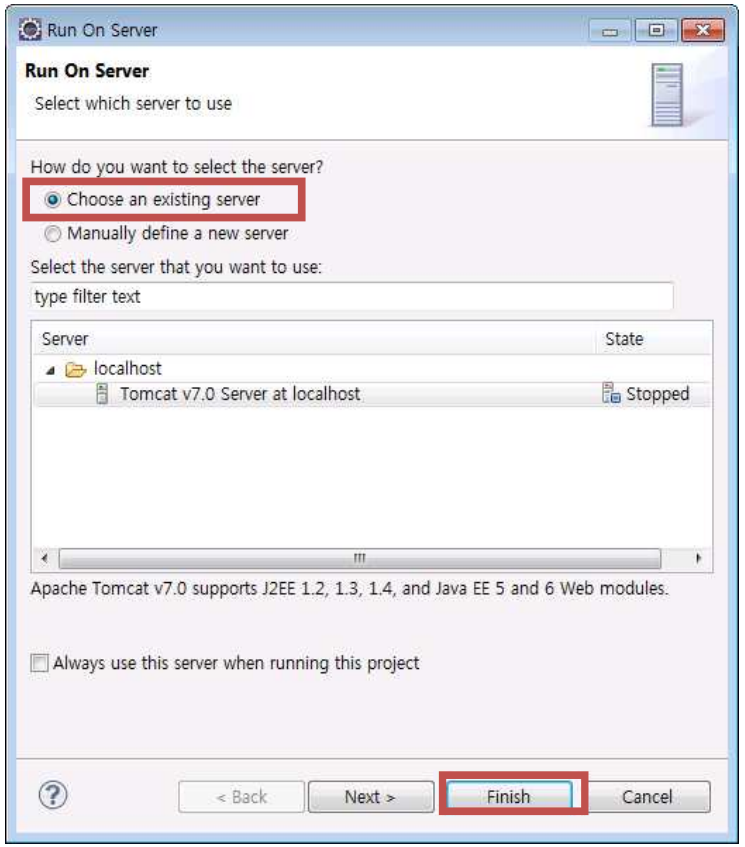
이클립스에서 실행하기

❶ 탐색기에서 comment.jsp 파일을 선택하고 <마우스 오른쪽> 버튼을 클릭해 [Run As] → [Run on Server] 를 선택한다.



[그림 5-5] 이클립스에서 comment.jsp 실행

❷ 서버를 선택하는 화면에서 [Choose an existing server] 를 선택하고, 3장에서 등록한 톰캣 서버가 기본으로 선택되어 있는지 확인한다.

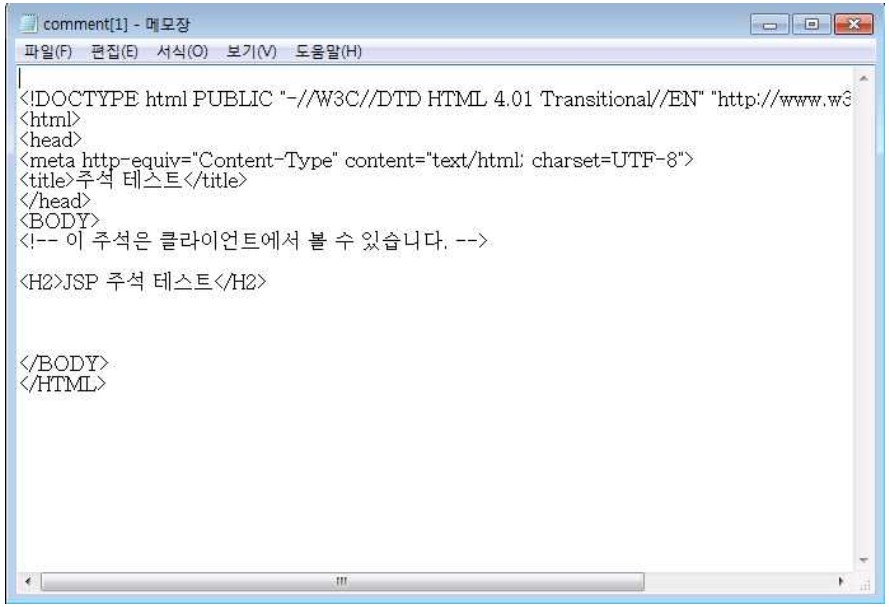


[그림 5-6] 톰캣 서버 선택

01. 주석

실행결과 확인하기

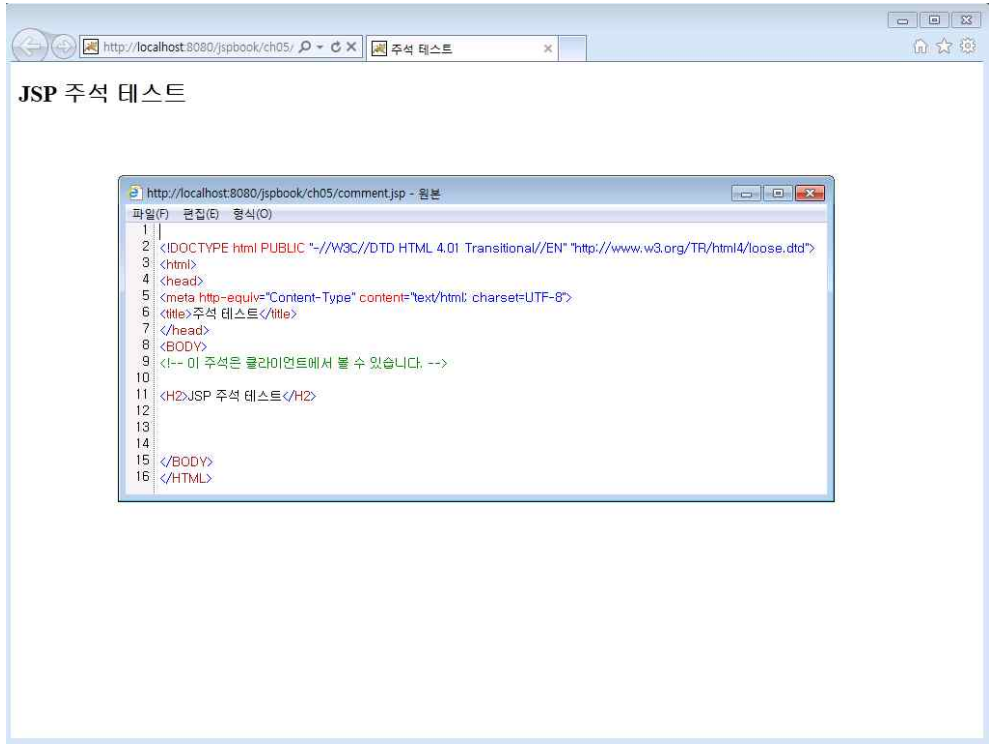
❶ 웹 브라우저에서 <마우스 오른쪽> 버튼을 클릭하고 [소스 보기] 명령을 선택



[그림 5-7] 실행 결과 화면

❷ 외부 브라우저를 통해 결과를 확인하려면 다음과 같은 URL을 입력

- http://localhost:8080/jspbook/ch05/comment.jsp



[그림 5-8] 인터넷 익스플로러에서 실행한 화면

02. 지시어

지시어(Directives)란?

- 지시어(Directives)는 JSP 파일의 속성을 기술하는 JSP 문법.
- JSP 컨테이너에게 해당 페이지를 어떻게 처리해야 하는지 전달하기 위한 내용을 담고 있다.
- 지시어는 크게 page, include, taglib으로 나눌 수 있으며, 각각에서 다루는 속성이 다르다.

1. page 지시어

- page 지시어는 현재 JSP 페이지를 컨테이너에서 처리하는 데 필요한 각종 속성을 기술하는 부분.
- 보통 JSP 페이지 맨 앞에 위치함.

```
<%@ page 속성1="속성값1" 속성2="속성값2" ... %>
```

- 여러 줄에 나누어 작성할 수도 있음.

```
<%@ page contentType="text/html;charset=UTF-8"
    import="javax.sql.*, java.util.*" errorPage="error.jsp"%>
<%@ page import="java.util.*" %>
```

02. 지시어

- 이클립스 개발 도구를 이용해 jsp 파일을 생성하는 경우 기본적인 page 지시어는 자동 생성됨.
- 필요에 따라 속성을 추가해 사용

[표 5-1] page 지시어 속성

속성	설명	기본 설정 값
language	스크립트 언어의 유형을 정한다.	java
import	JSP 내에서 사용할 외부 자바 패키지나 클래스의 불러오기(import)를 정한다.	–
session	세션의 사용 유무를 정한다.	true
buffer	버퍼의 크기를 정한다.	8KB
autoFlush	버퍼의 내용을 자동으로 비운다.	true
isThreadSafe	단일 스레드 모델을 사용함으로써 동시성 제어 여부를 정한다.	true
info	JSP 페이지에 대한 설명이다.	–
errorPage	현재 페이지에서 오류가 발생할 경우 호출될 페이지를 지정한다.	–
isErrorPage	오류만을 처리하는 페이지로 지정한다.	false
contentType	MIME 형식 지정 및 캐릭터셋을 설정한다.	text/html;charset=ISO-8859-1
pageEncoding	contentType과 동일한 기능을 한다.	ISO-8859-1
extends	현재 JSP 페이지를 특정 클래스를 상속한 클래스로 작성한다.	[표준스펙] javax.servlet.jsp.HttpJspPage [톰캣구현] org.apache.jasper.runtime.HttpJspBase

02. 지시어

Page 지시어와 JSP의 한글 처리

- page 지시어에서 중요한 부분 중 하나는 한글 처리 부분임.
- JSP 에서는 다음과 같이 3단계로 캐릿터셋을 설정함.

[표 5-2] JSP 에서의 캐릭터셋 관련 설정 항목

우선순위	설정 방법	비고
1	server.xml 파일에서 정의한다.	Connector 설정에 URI 요청에 대한 캐릭터셋을 지정할 수 있다. 작성한 소스를 다른 서버에서 실행할 경우, 해당 서버 설정에 따라 한글이 깨질 수도 있다.
2	각 JSP 파일, page 지시어의 pageEncoding 속성에서 정의한다.	페이지 단위의 설정이므로 애플리케이션 설정과 무관하게 한글을 처리할 수 있다. 단, 서버가 JSP 2.0 스펙을 지원하지 않는 예전 버전이라면 pageEncoding 속성을 인식하지 못해 한글이 깨질 수 있다.
3	각 JSP 파일, page 지시어의 contentType 속성에서 정의한다.	클라이언트에 대한 응답 결과를 지정하는 부분으로 브라우저에게 전달할 캐릭터셋을 지정할 수 있다.

- 위 설정에서 캐릭터셋 설정을 찾지 못할 경우 ISO8859-1을 적용한다.
- 페이지 지시어에 다음과 같이 한글 속성 설정
  - pageEncoding="UTF-8", contentType="text/html;chatsert=UTF-8"

02. 지시어

import

- import 는 JSP 스크립트 부분에서 자바 클래스를 사용하는 경우 해당 클래스의 패키지에 대한 import 설정으로 기본적으로 자바에서와 동일하다.
- 다만 패키지 import 구분을"," 을 이용하거나 라인 단위로 작성해야 한다.
- 다음은 page 지시어에서 import 속성의 다양한 사용 예이다.

```
<%@ page import="java.sql.*,java.util.*" %>
```

```
<%@ page import="java.sql.*" %>
<%@ page import="java.util.*" %>
```

```
<%@ page
        import="java.sql.*,
        java.util.*"
%>
```

02. 지시어

session

- 세션은 웹 브라우저와 웹 서버가 지속적인 클라이언트 인식을 위해 필요한 정보를 임시로 저장해두는 방법
- 주로 웹 사이트에 로그인하거나 쇼핑몰에서 장바구니 등을 구현할 때 사용된다.
- 기본 값이 true(세션을 사용한다)이므로, 일부러 사용을 제한할 목적이 아니라면 별도로 설정하지 않아도 됨.
- 세션과 관련한 자세한 내용은 6장에서 다룸.

```
<%@ page session="true" %>
```

buffer

- JSP 페이지 데이터를 출력하기 위한 JspWriter 즉 out 내장객체의 버퍼 크기를 지정.
- 기본값은 8KB 이고 JSP 페이지에 동적으로 많은 내용이 포함될 경우 버퍼 크기 조정이 필요할 수도 있으나 일반적으로는 변경하지 않아도 됨.

02. 지시어

autoFlush

- autoFlush는 버퍼를 자동으로 비울 것인지를 지정하는 속성으로, 기본 값은 true다.
- 버퍼 속성에 지정되어 있는 크기만큼 버퍼를 유지하고 있다가 버퍼가 다 차면 자동으로 전송한다.

```
<%@ page autoFlush="true" %>
```

isThreadSafe

- 기본적으로 서블릿은 스레드로 동작하기 때문에 스레드로 인한 동기화 문제를 해결하기 위한 옵션임.
- 기본값은 true로, 일반적으로 false로 설정하는 경우는 거의 없다.

```
<%@ page ifThreadSafe="true" %>
```

info

- 해당 JSP에 대한 간단한 설명으로 저작권이나 작성일 등 간단한 정보 기술에 사용.

```
<%@ page info="JSP Example" %>
```

02. 지시어

errorPage, isErrorPage

- 두 속성은 jsp파일의 오류 처리를 위한 것으로, errorPage는 현재 페이지에 오류 발생시 호출할 페이지를 지정하는 속성이고 isErrorPage는 오류 처리를 위한 전용 페이지임을 알리는 속성이다.

- errorPage지정을 통해 보다 효과적으로 페이지 오류를 관리할 수 있다.
  - errorPage : 일반적인 JSP 파일에 사용

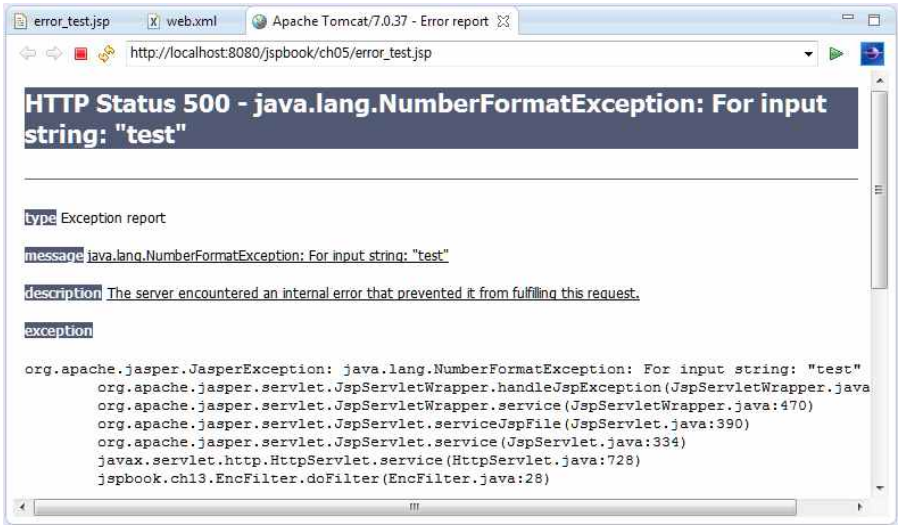
```
<%@ page errorPage="오류_처리_파일.jsp" %>
```

  - isErrorPage : 오류 처리 파일에만 사용

```
<%@ page isErrorPage="true" %>
```

02. 지시어

[실습] 런타임 오류를 발생시키는 파일(error_test.jsp) - 교재 p.168 참고



[그림 5-9] 일반적인 오류 화면

[실습] 오류 처리 전용 파일(error.jsp) - 교재 p.169 ~ 170 참고

[그림 5-10] 사용자 정의 오류 페이지 사용

contentType

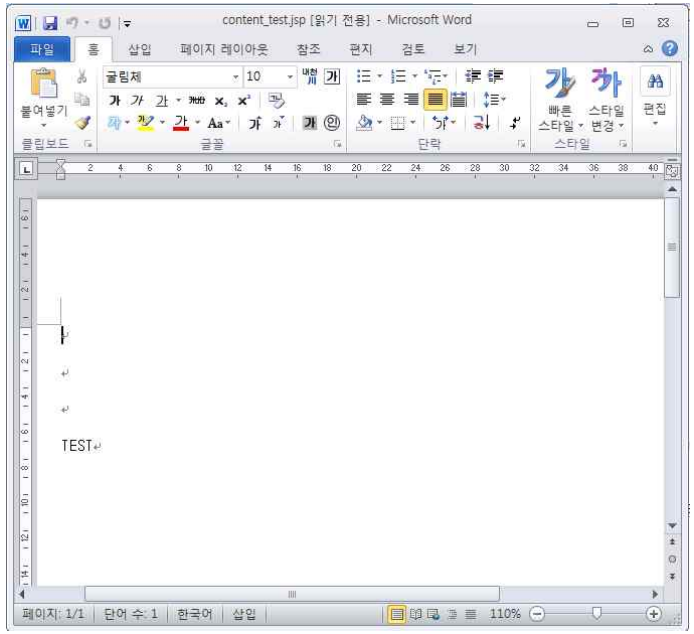
- 현재 JSP 페이지를 클라이언트에서 처리하기 위한 콘텐츠 유형을 지정하는 부분.
- 윈도우에서 파일 확장자(.doc, .hwp등)에 따라 연결 프로그램이 동작하는 것과 마찬가지로 웹 브라우저에서도 contentType에 따라 전달되는 내용을 어떻게 처리할지 결정할 수 있다.

```
<%@ page contentType="text/html" %>
```

- text/html 이 아니라 application/msword 로 지정할 경우 브라우저는 서버가 전달하는 콘텐츠를 ms word 문서로 인식해 처리할 것은 사용자에게 요청함.

```
<%@ page contentType="application/msword" %>
TEST
```

[그림 5-11] 파일 저장을 묻는 화면



[그림 5-12] MS 워드 실행 화면

02. 지시어

[실습] Content Type 테스트를 위한 JSP 파일(content_test.jsp) – 교재 p.171 참고

pageEncoding

- pageEncoding은 컨테이너에서 처리할 JSP 파일의 인코딩을 설정.

```
<%@ page pageEncoding="UTF-8" %>
```

- JSP 2.0 스펙에 추가된 속성으로, 이전 버전을 지원하는 컨테이너의 경우에는 사용할 수 없다.

extends

- JSP가 서블릿으로 변환될 때 상속받을 슈퍼클래스를 지정할 수 있는 부분이다.
- 일반적으로는 사용할 일이 없으며 톰캣의 경우 org.apache.jasper.runtime.HttpJspBase 클래스를 상속받게 된다.

2. include 지시어

- include 지시어는 현재 JSP 파일에 다른 HTML이나 JSP 문서를 포함하기 위한 기능을 제공.
- include 지시어는 다음 절에서 살펴볼 include 액션과 비슷한 기능을 한다.

```
<%@ include file="포함할 파일_이름" %>
```

- 네이버와 같은 인터넷 포털사이트의 화면처럼 여러 정보의 조합으로 한 화면을 구성할 때 유용하게 사용됨.
- include 지시어를 사용하면 기능 혹은 화면을 모듈화할 수 있어 화면 구성이나 재활용이 용이하다.

❶ 배너 화면 구성 : 광고 등 관련 사이트를 배너로 구성한다.

❷ 네이버 서비스 : 디렉터리 데이터베이스 연동 후 등록된 목록을 가져온다.

❸ 뉴스캐스트 : 뉴스 데이터베이스 연동 후 조회 수가 높은 최신 기사 제목들을 가져 온다.

❹ 쇼핑 : 쇼핑몰 데이터베이스 연동 후 인기 상품 이미지와 제품 정보를 가져 온다.

[그림 5-13] 네이버 화면 구성

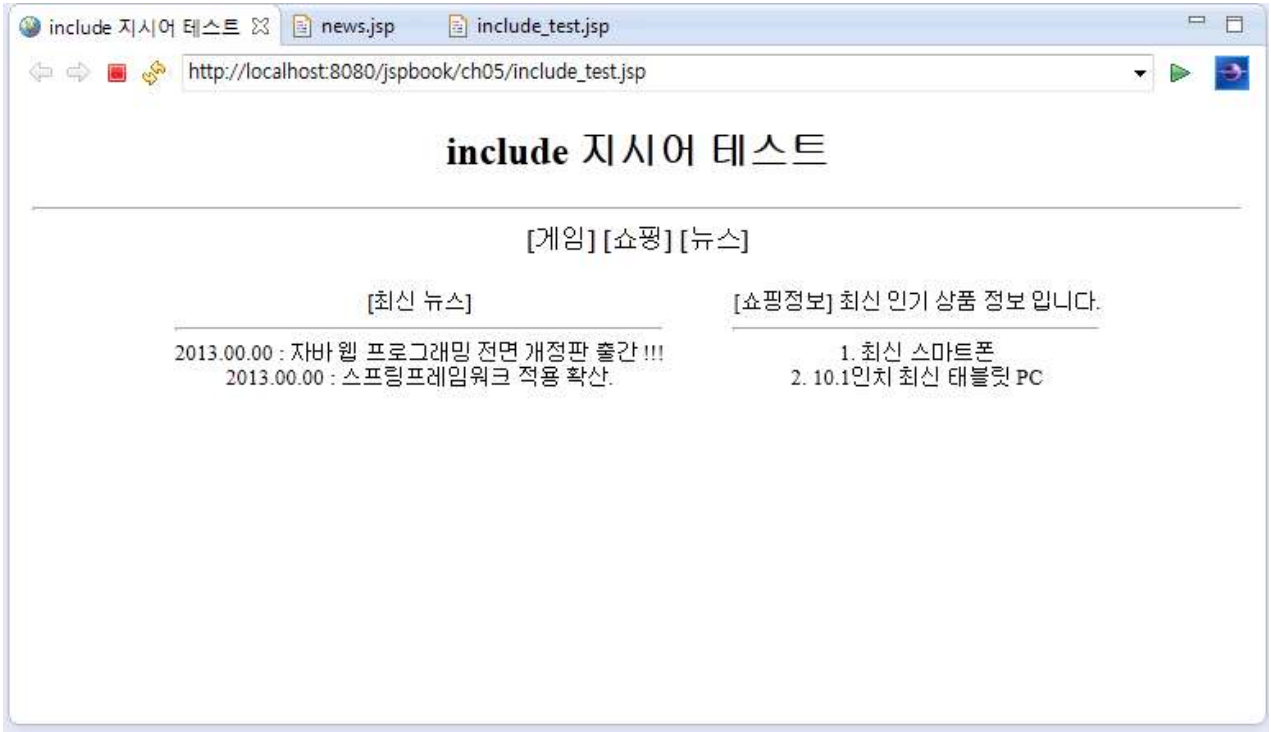
02. 지시어

- **[실습] 배너 화면 구성(include_test.jsp)**

  - 교재 p.175 참고

- **[실습] 메인 메뉴(menu.jsp), 뉴스캐스트(news.jsp), 쇼핑(shopping.jsp)**

  - 교재 p.176 참고



[그림 5-14] 결과 확인

02. 지시어

3. taglib 지시어

- JSP 기능을 확장하기 위해 만들어진 커스텀 태그 라이브러리를 사용하기 위한 지시어.
- 태그 라이브러리는 10장에서 자세히 살펴볼 것이고 여기서는 간단한 문법만 참조.

```
<%@ taglib uri="/META-INF/mytag.tld" prefix="mytag" %>
```

- 커스텀 태그는 공통으로 활용하거나 특정 기능을 HTML 태그 형태로 모듈화 하는 기술임
- 다음 예제에서는 GetInfo 라는 커스텀 태그를 사용하는 예로 실제 기능은 별도로 구현되어 있고 실행 결과는 user1 이라는 사용자의 정보를 DB에서 가져와 출력하는 형태가 될 수 있다.

```
01 <%@ page contentType="text/html;charset=UTF-8" %>
02 <%@ taglib uri="/WEB-INF/tlds/mytag.tld" prefix="mytag" %>
03 <HTML>
04 <BODY>
05 <mytag:GetInfo name="user1" />
06 </BODY>
07 </HTML>
```

taglib 지시어

커스텀 태그 사용

02. 지시어

저자 한마디

대규모 사이트에서의 분할 관리

include 지시어는 분명 편리한 페이지 분할 관리 기법을 제공하지만, 성능적인 측면에서는 그다지 권장할 만한 기법은 아니다. 또한 대규모 사이트의 구성에는 단순한 페이지 분할뿐만 아니라, 레이아웃 관리를 비롯한 고급 기능이 요구되므로 JSP의 include 지시어보다는 다음과 같은 별도의 전용 프레임워크를 사용하는 것이 더 적절한데, 스프링 프레임워크와 함께 사용하는 경우가 많다. 이들 프레임워크는 비교적 복잡한 설정과정이 필요하고 구조를 잘 설계해야 하므로 간단한 사이트 구성 등에는 include가 유용하게 사용될 수 있다.

- 타일즈(http://tiles.apache.org)
- 사이트 메시(http://www.sitemesh.org)

---

include 내부 실행 구조

[예제 5-5]의 실행결과 화면에서 [소스보기]를 하면 jsp 파일 4개가 모두 하나로 합쳐져 있다. 실제 java로 변환된 소스를 보더라도 include_test.jsp 이외의 파일은 자바 소스와 클래스가 생성되어 있지 않다. 즉 include는 각각의 파일을 포함한 뒤 하나의 컴파일된 클래스를 생성하므로 include만을 목적으로 사용할 파일이라면 소스에서 page 지시어 이외의 HTML 기본 태그 관련 부분들(<html>, <head>, <body> 등)은 생략할 수 있다.

URI와 URL의 차이점

URI는 'Uniform Resource Identifier'의 약자로, 필요로 하는 자원(파일 등)의 위치 정보를 기술하는데 사용한다. 일반적으로 알고 있는 URL과의 차이점은 프로토콜, 주소, 포트 번호를 제외한 경로만을 표시한다는 것이다.

- URL 예 http://www.hanbitbook.com:8080/jsp/book/index.jsp
- URI 예 /jsp/book/index.jsp

03. 액션

1. JSP 액션의 종류

- JSP 액션은 JSP 고유 기능으로 빈즈 클래스 연동 및 동적 페이지 관리를 위한 기능을 제공함.
- <jsp:action_name attribute="value" /> 형태를 가짐.
- 주로 사용하는 액션은 useBean, get/setProperty 이며 자바 클래스와의 연동을 위해 사용함.

  → 7장. JSP와 자바빈즈

- 액션(Action)은 JSP 주요 구성요소 중 하나로 다음과 같은 기능을 지원한다.
  - JSP 페이지간 흐름 제어
  - 자바 애플릿 지원
  - 자바 빈즈 컴포넌트와 JSP 상호작용 지원
- 특히 useBean 액션은 JSP에서 자바 빈즈 클래스와의 연동을 지원해주는 액션으로 잘 알아둘 필요가 있다.
- include 액션은 단순히 페이지를 포함하는 것 뿐만 아니라 파라미터를 포함될 페이지로 전달하는 것이 가능함.
  - 사용 예) <jsp:param name="user" value="홍길동" />

03. 액션

- 대표적인 액션은 표5-3과 같음.

[표 5-3] 대표적인 액션의 종류

액션	사용 예	기능
include	<jsp:include page="xx.jsp" />	다른 페이지를 현재 페이지에 포함시킨다.
forward	<jsp:forward page="xx.jsp" />	현재 페이지의 제어를 다른 페이지로 전달한다.
useBean	<jsp:useBean scope="page" id="cls" class="xx.MyBean" />	xx패키지의 MyBean 클래스를 cls라는 이름으로 page 범위에서 사용할 것을 선언한다.
setProperty	<jsp:setProperty name="cls" property="xxx" />	useBean으로 선언된 빈즈 클래스의 setxxx( ) 메서드를 호출한다.
액션	**사용 예**	**기능**
getProperty	<jsp:getProperty name="cls" property="xxx" />	useBean으로 선언된 빈즈 클래스의 getxxx( ) 메서드를 호출한다.
plugin	<jsp:plugin type="applet/bean" code="class"> </jsp:plugin>	애플릿이나 빈즈 클래스를 플러그인 형태로 로딩한다.
param	<jsp:param name="user" value="홍길동" />	include, forward 액션에서 사용할 수 있는 파라미터를 설정한다.

03. 액션

2. include 액션

- include 액션은 다른 파일을 불러온다는 측면에서 include 지시어와 개념이 유사.
- include 지시어는 해당 파일을 포함시킨 후 컴파일하는 것에 비해, include 액션은 실행 시점에서 해당 파일을 호출하여 그 결과를 포함한다는 점에서 차이가 있음.
- 동적으로 파일들을 핸들링 하기 때문에 과도한 사용은 성능상에 문제를 줄 수 있음.
- include 액션은 동적인 페이지를 포함시킬 경우에 사용하는 것이 좋고, include 지시어는 잘 바뀌지 않는 정적인 페이지를 포함 할 때 사용하는 것이 좋다.

```
<jsp:include page="포함할 파일_이름" />
```

- 사용 예

```
<jsp:include page="footer.jsp">
    <jsp:param name="email" value="test@test.net" />
    <jsp:param name="tel" value="000-000-0000" />
</jsp:include>
```

03. 액션

[실습] include 액션 사용하기(include_action.jsp)

- 교재 p.180 ~ 181 참고

[실습] include 액션에서 footer.jsp 호출하기(footer.jsp)

- 교재 p.181 참고

[그림 5-15] 실행 결과

3. forward 액션

- forward 액션은 include 액션과 사용법은 유사하지만 요청 페이지를 다른 페이지로 전환할 때 사용한다.
- response 내장객체의 sendRedirect()와 유사 하지만 포워드된 페이지에 파라미터를 전달할 수 있다는 점에서 차이가 있다.
- 브라우저 URL 창에는 최초 요청 페이지가 표시 되기 때문에 처리 페이지 정보를 숨기거나 MVC 패턴의 컨트롤러와 같이 특정 기능 수행 후 다른 페이지로 이동해야 하는 경우 유용하게 사용할 수 있다.

```
<jsp:forward page="포워딩할 파일_이름" />
```

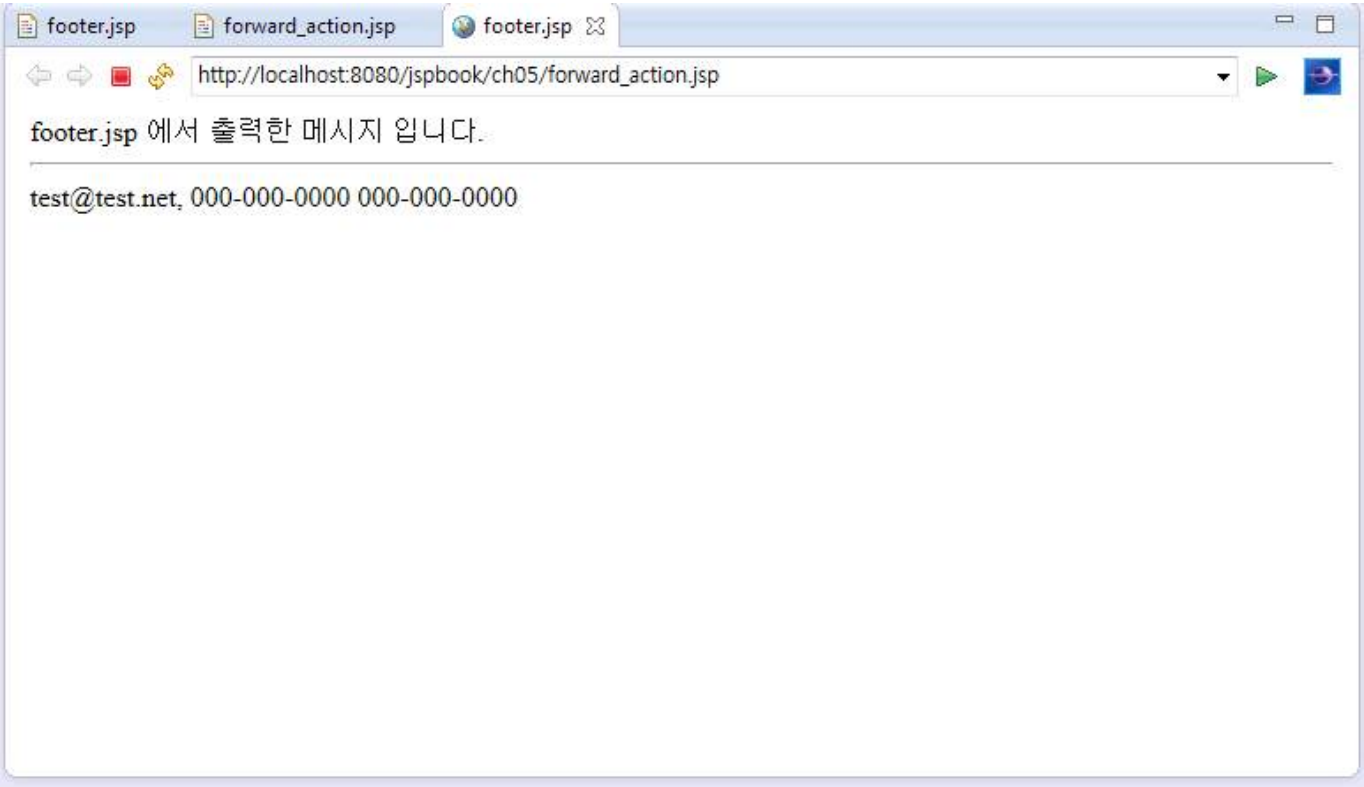
- 사용 예)

```
<jsp:forward page="footer.jsp" />
<jsp:param name="email" value="test@test.net" />
<jsp:param name="tel" value="000-000-0000" />
</jsp:forward>
```

03. 액션

[실습] forward 액션 사용하기(forward_action.jsp)

- 교재 p.182 ~ 183 참고



[그림 5-16] forward_action.jsp 실행 결과

4. plugin 액션

- plugin 액션은 웹 브라우저에서 자바 플러그인을 사용하여 자바 애플릿이나 자바 빈즈 컴포넌트를 실행 할 수 있게 한다.
- plugin 액션을 사용하면 자동으로 <OBJECT> 혹은 <EMBED>와 같은 태그를 통해 애플릿 등을 실행하도록 한다. 일반적으로 애플릿을 사용하는 경우가 드물기 때문에 참고만 하기 바란다.
- 사용법

```
<jsp:plugin type="bean|applet" code="objectCode" codebase="objectCodeBase"
{ align = alignment }
{ archive = archiveList }
{ height = height }
{ hspace = hspace }
{ name = name }
{ vspace = vspace }
{ width = width }
{ nspluginurl = url }
{ iepluginurl = url }
{ <jsp:params>
{<jsp:params name="paramName" value="paramValue" />}
</jsp:params> }
{<jsp:fallback> Plugin S/W를 지원하지 못하는 경우의 설명</jsp:fallback> }
</jsp:plugin>
```

5. useBean 액션

- 액션에서 가장 중요한 부분으로, JSP 빈즈를 다루는 7장에서 자세히 살펴볼 것이다.
- 여기서는 기본 표기 방법만을 살펴보기로 한다.
- 사용법

```
<jsp:useBean id="변수_이름" class="빈즈 클래스_이름"/>
<jsp:setProperty name="변수_이름" property="속성_이름"/>
<jsp:getProperty name="변수_이름" property="속성_이름"/>
```

  - useBean 액션은 빈즈 클래스를 사용하기 위한 구문이며 class 에 지정된 자바 빈즈 클래스를 id 라는 이름으로 사용할 수 있도록 해준다.
  - get/setProperty 액션은 브라우저에서 빈즈 클래스의 멤버 변수로 값을 저장하거나 가져오기 위한 구문이다.
  - get/setProperty는 빈즈 클래스의 getter/setter 메서드와 연동된다.

04. 선언과 표현식

1. 선언

- JSP 페이지에서 메서드나 멤버변수를 선언하기 위한 구문.
- JSP 가 서블릿으로 변환된 자바 코드에서는 모든 내용이 _jspService() 메서드에 들어가기 때문에 JSP 에서 선언한 변수는 로컬변수가 되고 메서드 안에서 다른 메서드를 선언하는 자바 문법상 잘못된 것이므로 컴파일 에러가 발생하게 됨.
- <%! %>는 JSP 페이지에서 이러한 제약 사항 없이 멤버변수와 메서드 선언을 가능하게 함.
- 구조적으로 JSP 에서 자바 코드를 사용하는 것은 권장되지 않기 때문에 선언문의 사용 역시 권장되지 않음.
- 사용 예)

```
<%!
    // 멤버변수 선언이나 메서드 선언이 올 수 있다.
    String str = "test";
    public boolean check() {
    return false;
    }
%>
```

[실습] 메서드를 이용한 계산기 프로그램(calc3.jsp)

- 교재 p.186 ~ 187 참고

04. 선언과 표현식

2. 표현식

- 표현식(Expression)은 이미 여러 소스를 통해 많이 살펴본 것처럼 <%= %>를 사용해서 간단한 데이터 출력이나 메서드 호출 등에 이용한다.
- 코드 마지막에 ';(세미콜론)'을 사용하지 않는다는 것에 주의하도록 한다.
- 표현식은 결국 out.println() 으로 변환되기 때문에 out.println()의 인자로서 적합한 형태로 사용해야 함.
- 메서드 호출, 변수출력, 사칙연산 및 문자열 결합 등이 가능하다.
- 표현식 보다는 표현언어(Expression Language) 사용을 권장한다. → 10장
- 사용 예)

```
메서드 호출 : <%= calculator() %>
변수 출력 : <%= result %>
사칙 연산과 문자열 결합 : <%= "i+2="+(i+2)+"입니다" %>
```

05. 스크립트릿

1. 스크립트릿이란?

- 스크립트릿(Scriptlet) 은 JSP 문서 내에서 자바 코드를 기술할 수 있는 부분으로 JSP의 가장 큰 특징 중 하나.
- 초기 JSP 발전에 큰 기여를 하였으나 지금은 JSP내에서 스크립트릿 사용은 권장되지 않음.
- JSP 내에서 화면과 프로그램이 섞여 있으면 유지보수가 힘들고 디자이너들과의 협업이 어려움.
- MVC 패턴에 따라 웹 프로그램을 개발하게 되면 JSP는 View의 역할을 하게 되고 표현언어, JSTL, 커스텀 태그 라이브러리, JSP 빈즈가 주로 사용된다.
- 그러나 JSP의 가장 큰 특징이며 유용한 부분이기도 하므로 자유자재로 사용할 수 있어야 함.
- UI(AWT, Swing 등) 을 제외한 거의 모든 자바 클래스 라이브러리를 사용한 프로그래밍이 가능함.
- 사용 예)

```
<%
    // 로컬 변수 선언이나 프로그램 로직이 올 수 있다.
    String str = "test";
    for (int i=0; i < 10; i++) {
    out.println(i);
    }
%>
```

■ **데이터 출력 프로그램 ❶(scriptlet1.jsp), 데이터 출력 프로그램 ❷(scriptlet2.jsp)**

- 교재 p.190 ~ 191 참고