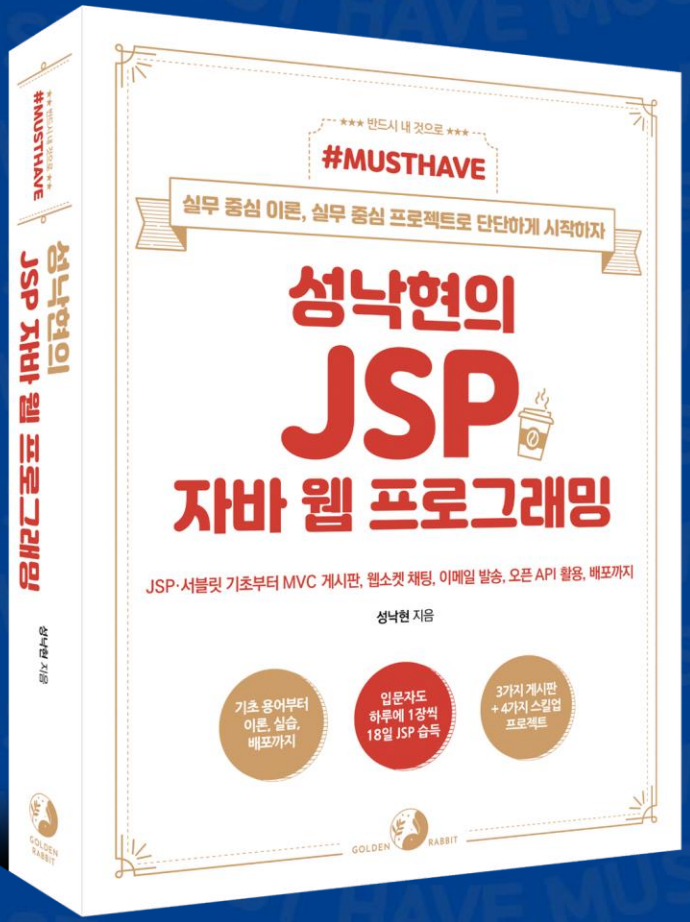


Chapter

01

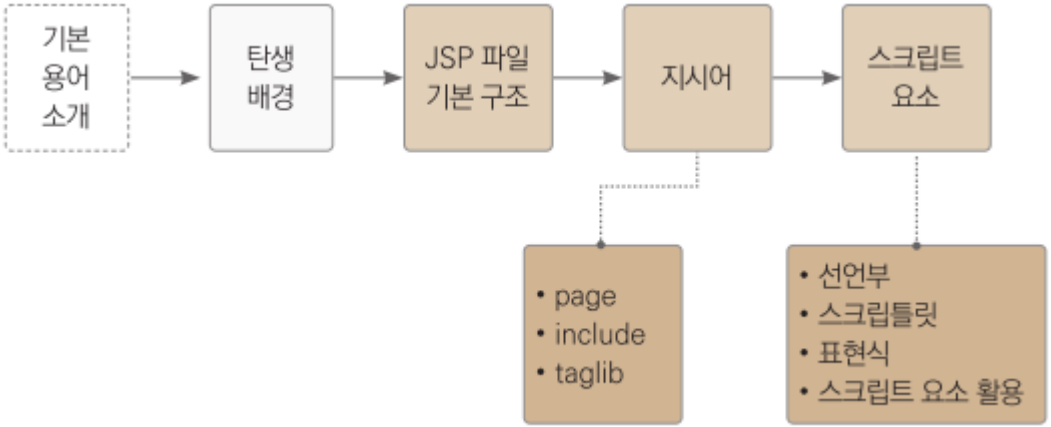
JSP 기본

학습 목표 및 순서

▣ 학습 목표

- JSP의 개념, 탄생 배경, 동작 원리를 이해하고 JSP 파일의 기본 구조와 핵심 요소를 익힙니다.

▣ 학습 순서



학습 목표 및 순서

▣ JSP란

- JSP(Java Server Pages)는 동적인 웹 페이지를 개발하기 위한 웹 프로그래밍 기술
- 자바(Java) 언어를 사용하여 서버 측에서 웹 페이지를 생성해 웹 브라우저로 전송

▣ 장점

- 짧은 코드로 동적인 웹 페이지를 생성
- 기본적인 예외는 자동으로 처리
- 많은 확장 라이브러리를 사용할 수 있음
- 스레드 기반으로 실행되어 시스템 자원을 절약해줌

▣ 활용 사례

- 대한민국 정부 표준 프레임워크의 근간
- 정부나 공기업 주도의 사업 등 대규모 기업용 시스템 구축에 주로 사용

기본 용어 소개

▣ 서버(Server)

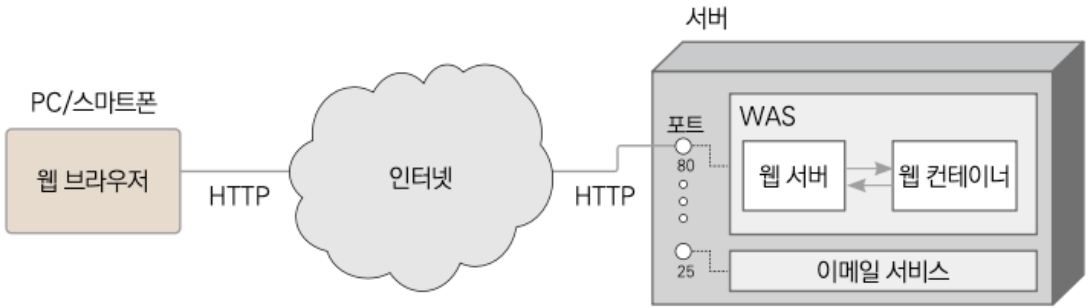
- 웹에서 서비스를 제공하는 컴퓨터

▣ 웹 서버(Web Server)

- 사용자로부터 HTTP를 통해 요청을 받거나, 웹 컨테이너가 전달해준 결과물을 정적인 페이지로 생성하여 사용자에게 응답해주는 소프트웨어

▣ 웹 컨테이너(Web Content)

- 웹 서버가 전송해준 요청을 기초로 동적인 페이지를 생성하여 웹 서버로 돌려줌, 동적인 페이지라고 표현하는 이유는 사용자마다 다른 결과로 응답할 수 있기 때문임



1.1 동적 웹 페이지로의 여정과 JSP

▣ 정적 웹 페이지

- 클라이언트의 요청에 상관없이 항상 동일한 내용을 출력

▣ 동적 웹 페이지

- 서버가 클라이언트의 요청을 해석하여 가장 적절한 웹 페이지를 생성해 웹 브라우저에 출력

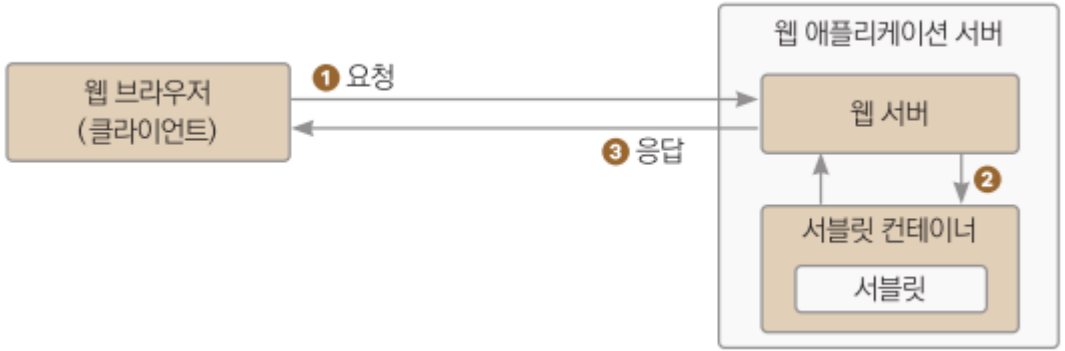
1.1 동적 웹 페이지로의 여정과 JSP

▣ 애플릿

- 웹을 동적으로 만들기 위한 고대의 자바 기술
- 웹에서 실행할 수 있는 자바 애플리케이션을 통째로 웹 브라우저로 전송한 후 실행하는 방식
- 속도, 보안, 유연성의 한계로 현재는 더 이상 지원되지 않음

▣ 서블릿

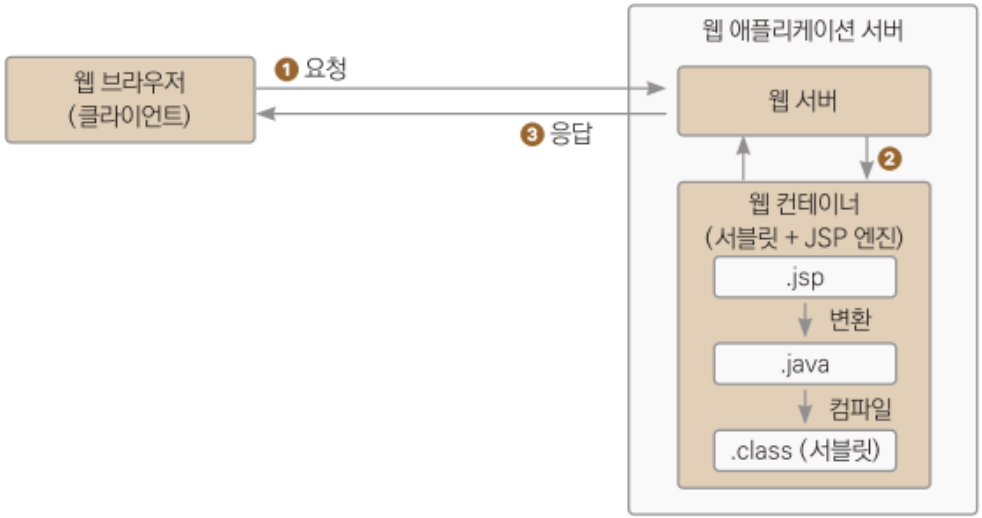
- 클라이언트의 요청을 받으면 서버에서 처리한 후, 응답으로 결괏값만 보내주는 구조
- 자바(.java) 파일을 컴파일 한 클래스(.class) 파일의 형태를 가짐
- 대표적인 서블릿 컨테이너로 아파치 톰캣(Apache Tomcat)이 있음



1.1 동적 웹 페이지로의 여정과 JSP

▣ JSP - 자바 웹 기술의 최종 진화

- 자바코드를 HTML로 변환하는 과정에서 너무 많은 코드가 필요했던 서블릿의 단점을 보완
- 기본을 HTML로 작성한 후 필요한 부분만 Java를 사용하는 형태
- JSP 파일을 서블릿으로 변환한 후 실행하는 방식
- JSP는 클라이언트에게 보여지는 FrontEnd를 담당, 서블릿은 제어를 위한 BackEnd를 담당



Servlet vs jsp

Servlet 코드

```
protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws
ServletException, IOException {
// TODO Auto-generated method stub
response.setContentType("text/html; charset=UTF-8");

// 출력 스트림 가져오기
PrintWriter out = response.getWriter();

// HTML과 "Hello World" 출력
out.println("<!DOCTYPE html>");
out.println("<html>");
out.println("<head><meta charset='UTF-
8'><title>Hello Servlet</title></head>");
out.println("<body>");
out.println("<h1>Hello World</h1>");
out.println("</body>");
out.println("</html>");
}
```

jsp 코드

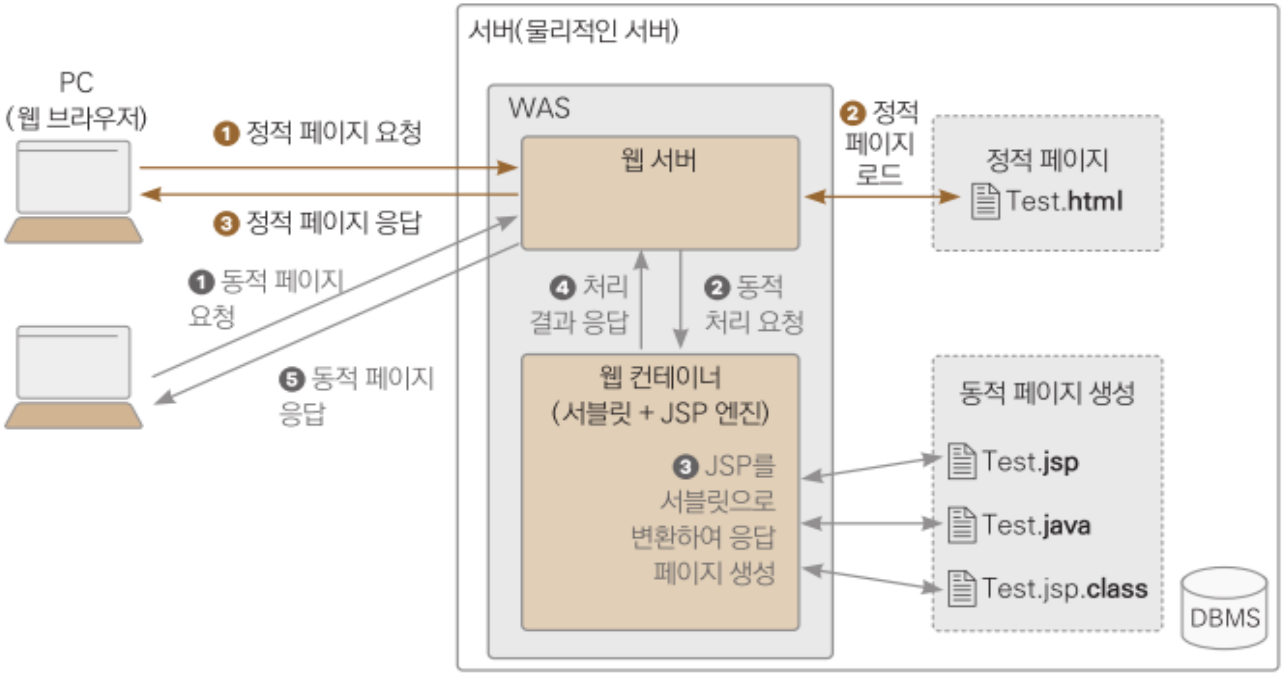
```
<%@ page language="java"
contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<h1><%= "Hello World" %></h1>
</body>
</html>
```

클래스 파일 확인 경로
~\2025_jsp\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\work\Catalina\localhost\JspPro\org\apache\jsp

1.1 동적 웹 페이지로의 여정과 JSP

▣ 오늘날의 웹 사이트

- 동적 웹 페이지와 정적 웹 페이지가 혼합된 형태로 구성됨



1.2 JSP 파일 기본 구조

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%!
String str1 = "JSP";
String str2 = "안녕하세요..";
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>HelloJSP</title>
</head>
<body>
    <h2>처음 만들어보는 <%= str1 %></h2>
    <p>
        <%
        out.println(str2 + str1 + "입니다. 열공합시다^^*");
        %>
    </p>
</body>
</html>
```

지시어

스크립트 요소(선언부)

스크립트 요소(표현식)

스크립트 요소
(스크립틀릿)

지시어(Directive)

- 해당 JSP 페이지의 처리 방법을 JSP 엔진에 '지시'해주는 역할

스크립트 요소(Scripting Elements)

- 선언부 : 멤버변수나 메서드 선언
- 표현식 : 변수 출력, 메서드 호출
- 스크립틀릿 : Java코드 작성

1.3 지시어(Directive)

지시어(Directive)란..??

- JSP 페이지를 자바(서블릿) 코드로 변환할때 필요한 정보를 JSP 엔진에 알려줌
- 스크립트 언어나 인코딩 방식 등을 설정

```
<%@ 지시어종류 속성1="값1" 속성2="값2" ... %>
```

지시어의 종류

- page 지시어 : JSP 페이지에 대한 정보를 설정
- include 지시어 : 외부 파일을 현재 JSP 페이지에 포함시킴
- taglib 지시어 : 표현 언어에서 사용할 자바 클래스나 JSTL을 선언

1.3 지시어(Directive) - page

page 지시어

- 문서의 타입, 에러 페이지, MIME 타입과 같은 JSP 페이지에 대한 정보를 설정

속성	내용	기본값
info	페이지에 대한 설명을 입력	없음
language	페이지에서 사용할 스크립팅 언어를 지정	java
contentType	페이지에서 생성할 MIME 타입을 지정	없음
pageEncoding	charset과 같이 인코딩을 지정	ISO-8859-1
import	페이지에서 사용할 패키지와 클래스를 지정	없음
session	세션 사용 여부를 지정	true
buffer	출력 버퍼의 크기를 지정. 사용하지 않으려면 “none”으로 지정	8Kb
autoFlush	출력 버퍼가 모두 채워졌을 때 자동으로 비울지를 결정	true
trimDirectiveWhitespaces	지시어 선언으로 인한 공백을 제거할지 여부를 지정	false
errorPage	에러 발생시 에러처리를 위한 페이지를 지정	없음
isErrorPage	해당 페이지가 에러를 처리할지 여부를 지정	false

1.3 지시어(Directive) - page

language, contentType, pageEncoding 속성

- JSP 페이지 생성시 기본적으로 삽입되는 지시어
  - language : 스크립팅 언어를 java로 지정
  - contentType : 문서의 MIME타입과 캐릭터셋 지정
  - pageEncoding : 소스 코드의 인코딩 방식 지정
- import 속성
  - java.lang 패키지에 속하지 않은 클래스를 JSP 문서에서 사용하기 위해 선언
  - 이클립스에서는 자동완성(Content assist) 기능으로 임포트 하면 됨

예제 1-1] 01DirectiveScript/Import.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page import="java.text.SimpleDateFormat"%>  ❶ 필요한 외부 클래스 임포트
<%@ page import="java.util.Date"%>
<!DOCTYPE html>
```

1.3 지시어(Directive) - page

errorPage, isErrorPage 속성

- JSP에서 에러 발생 시 “HTTP Status 500”과 같은 에러페이지를 출력
- 웹서버의 버전, 소스코드의 일부가 노출되므로 보안 측면에서 좋지 않음

예제 1-2] 01DirectiveScript/Error500.jsp

```
<body>
<%
int myAge = Integer.parseInt(request.getParameter("age")) + 10;  // 에러 발생 ❶
out.println("10년 후 당신의 나이는 " + myAge + "입니다.");  // 실행되지 않음
%>
</body>
```

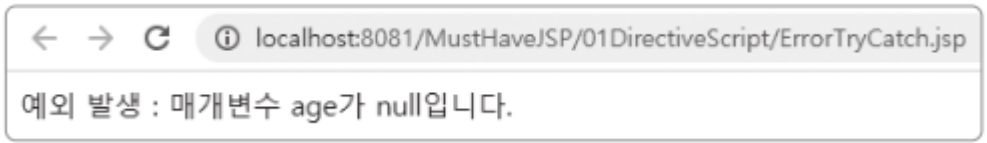
1.3 지시어(Directive) - page

errorPage, isErrorPage 속성 - 방법1 : try/catch 사용

- 에러 발생이 예상되는 코드를 try ~ catch로 묶어줌

예제 1-3] 01DirectiveScript/ErrorTryCatch.jsp

```
<body>
<%
try {  // 예외 발생 부분을 try/catch로 감쌉니다. ❶
    int myAge = Integer.parseInt(request.getParameter("age")) + 10;
    out.println("10년 후 당신의 나이는 " + myAge + "입니다.");
}
catch (Exception e) {
    out.println("예외 발생 : 매개변수 age가 null입니다.");
}
%>
</body>
```



1.3 지시어(Directive) - page

errorPage, isErrorPage 속성 - 방법2 : errorPage, isErrorPage 사용

- 에러 발생이 예상되는 페이지에 errorPage 속성 추가
- 에러를 처리할 페이지에 isErrorPage 속성을 추가

예제 1-4] 01DirectiveScript/ErrorPage.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
      pageEncoding="UTF-8"
      errorPage="IsErrorPage.jsp"%>  ❶ 에러 페이지 지정
... 생략 ...
```

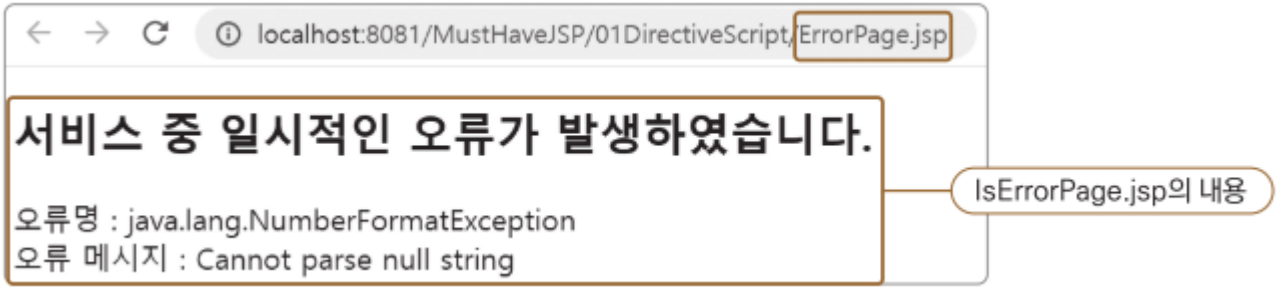
예제 1-5] 01DirectiveScript/IsErrorPage.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"
    isErrorPage="true"%>  ❶ isErrorPage 속성에 true를 지정
<!DOCTYPE html>
<html>
```

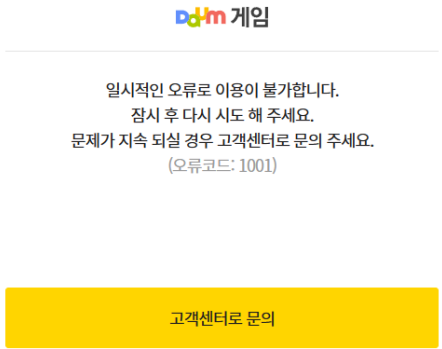
1.3 지시어(Directive) - page

errorPage, isErrorPage 속성 - 방법2 : errorPage, isErrorPage 사용

- ErrorPage.jsp 실행 시 그림과 같이 IsErrorPage.jsp 내용이 출력됨



- 실제 서비스에서는 오류메세지 대신 사용자에게 친근한 UI로 출력함



1.3 지시어(Directive) - page

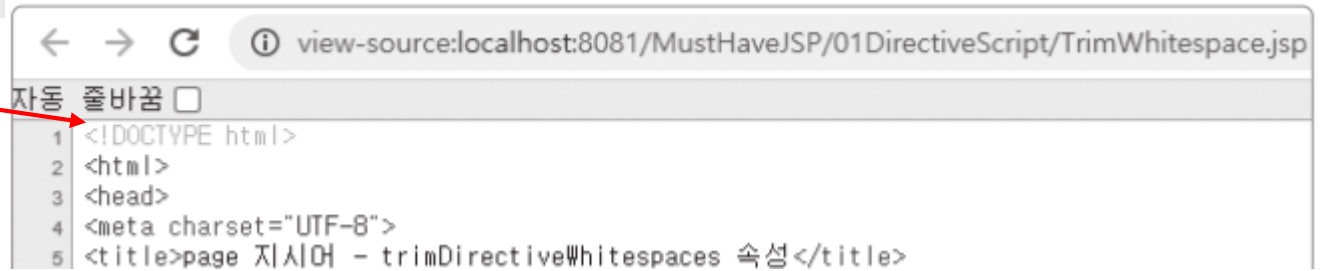
trimDirectiveWhitespaces 속성

- page 지시어가 웹 서버에 서 처리된 후 공백으로 남게 되는 것을 제거함
- 공백(Space)도 엄연한 문자이므로, 외부기기와 연동시 문제를 일으키는 경우가 있음

예제 1-6] 01DirectiveScript/TrimWhitespace.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"
    trimDirectiveWhitespaces="true"%>  ❶
<!DOCTYPE html>
<html>
```

공백제거됨



1.3 지시어(Directive) - page

buffer, autoFlush 속성

- JSP는 응답결과를 즉시 출력하지 않고 **버퍼(Buffer)**에 먼저 저장한 후 일정량이 되었을때 출력
- 작은 단위로 여러번 전송하는것 보다, 큰 단위로 묶어서 한번에 보내는 것이 훨씬 효율적
- **버퍼**라는 임시 저장소를 두어 데이터들이 충분히 쌓일 때까지 기다렸다가 출력

- JSP는 버퍼를 사용함으로써 포워드(forward: 페이지 전달)와 에러 페이지 처리를 할 수 있음

예제 1-7] 01DirectiveScript/AutoFlushTest.jsp

1.3 지시어(Directive) - include

include 지시어

- 웹 사이트에서 상단 메뉴나 하단의 정보가 여러페이지에 반복 사용되는 경우에 필요함
- JSP페이지에 또 다른 JSP페이지를 삽입할 때 사용

```
<%@ include file="포함할 파일의 경로"%>
```

예제 1-8] 01DirectiveScript/IncludeFile.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
LocalDate today = LocalDate.now();  // 오늘 날짜
LocalDateTime tomorrow = LocalDateTime.now().plusDays(1);  // 내일 날짜
%>
```

예제 1-9] 01DirectiveScript/IncludeMain.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ include file="IncludeFile.jsp" %>  ❶ 다른 JSP 파일(IncludeFile.jsp) 포함
<!DOCTYPE html>
```

1.4 스크립트 요소(Script Elements)

스크립트 요소란..??

- JSP에서 자바 코드를 직접 작성할 수 있게 해줌
- 용도에 따라 선언부, 스크립틀릿, 표현식 3가지가 있음

▶ 선언부(Declaration)

- 스크립틀릿이나 표현식에서 사용할 멤버 변수나 메서드를 선언
- 서블릿으로 변환 시 _jspService( ) 메서드 '외부'에 선언

```
<%! 메서드 선언 %>
```

▶ 스크립틀릿(Scriptlet)

- 실행할 자바 코드를 작성하는 영역
- 서블릿으로 변환 시 _jspService( ) 메서드 '내부'에 선언

```
<% 자바 코드 %>
```

1.4 스크립트 요소(Script Elements)

▶ 표현식(Expression)

- 변수의 값을 웹 브라우저 화면에 출력
- 반환값이 있는 메서드를 호출
- out.print( ) 를 대체하여 좀 더 단순한 방법으로 출력

```
<%= 자바 표현식 %>
```

예제 1-10] 01DirectiveScript/ScriptElements.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%!  ❶ 선언부(메서드 선언)
public int add(int num1 , int num2) {
    return num1 + num2;
}
%>
<html>
```

1.4 스크립트 요소(Script Elements)

예제 1-10] 01DirectiveScript/ScriptElements.jsp

```
<%  ❷ 스크립틀릿(자바 코드)
int result = add(10, 20);
%>
덧셈 결과 1 : <%= result %> <br />  ❸ 표현식(변수)
덧셈 결과 2 : <%= add(30, 40) %>    ❹ 표현식(메서드 호출)
</body>
</html>
```

JSP 파일이 서블릿으로 변환된 결과 확인하기

- C:\02Workspaces\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\work\Catalina\localhost\MustHaveJSP\org\apache\jsp_01DirectiveScript

- 선언부에 정의한 add( ) 메서드는 _jspService( ) 외부에 선언되어 있음
- 스크립틀릿에 작성한 코드는 _jspService( ) 내부에 기술되어 있음
- 따라서 스크립틀릿에 메서드를 정의하면 에러발생됨

학습 마무리

- JSP는 지시어와 스크립트 요소로 구성
- 지시어는 JSP에 대한 가장 기본적인 설정을 하는 요소
  - page, include, taglib 3가지가 있음
- 스크립트 요소는 자바 코드를 삽입할 수 있게해줌
  - 선언부, 스크립틀릿, 표현식 3가지가 있음

▣ 핵심요약

- 지시어
  - page 지시어 : JSP 페이지에 대한 기본정보 설정
  - include 지시어 : JSP나 HTML 페이지를 포함시킬 때 사용
  - taglib 지시어 : EL(표현 언어)에서 자바 클래스의 메서드를 호출하거나, JSTL(JSP 표준 태그 라이브러리)을 사용하기 위해 선언(10,11장에서 학습)
- 스크립트 요소
  - 선언부 : 멤버 변수나 메서드를 선언할 때 사용하는 영역
  - 스크립틀릿 : 선언부에서 선언된 메서드를 호출하거나 자바 코드를 작성하는 영역
  - 표현식 : 주로 변수의 값을 간단하게 출력할 때 사용