

JSP의 기초

JSP 문서의 기본 구조

- JSP 기술에서 웹 애플리케이션을 구현할 때 작성하는 코드를 JSP 페이지라고 한다.
- JSP 페이지는 HTML 문서의 사이에 Java 문법의 코드가 삽입되는 형태로 작성된다.

```
<% @page contentType= "text/html; charset=euc-kr" %>
<HTML>
  <HEAD><TITLE>1부터 100까지의 합</TITLE></HEAD>
  <BODY>
    <%
      int total = 0;
      for (int cnt = 1; cnt <= 100; cnt++)
        total += cnt;

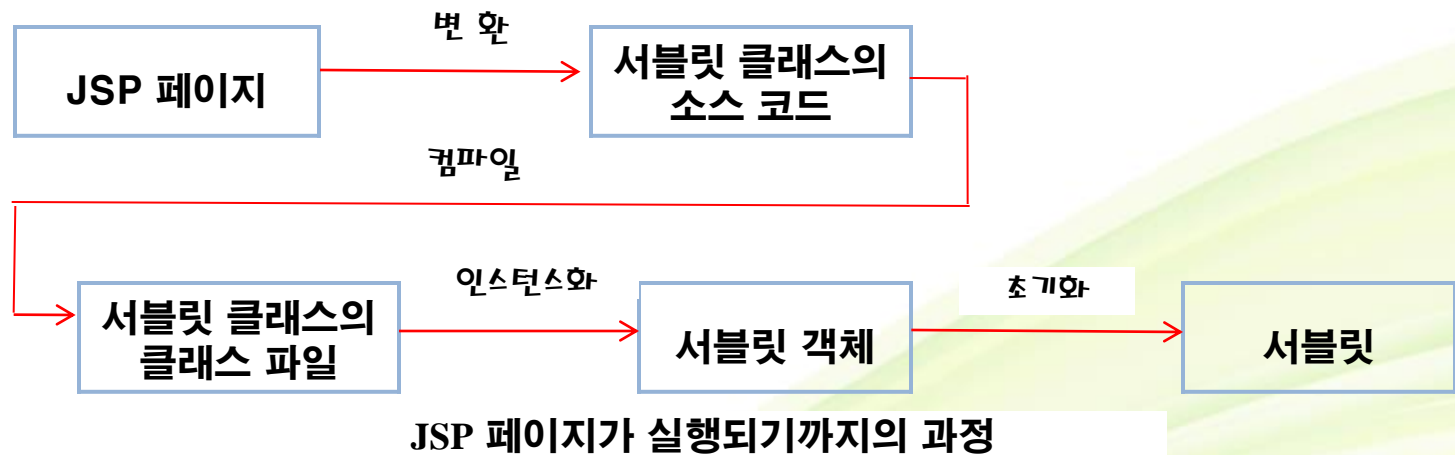
      %>
    1부터 100까지 더한 값은? <%= total %>
  </BODY>
</HTML>
```

HTML 문서의 사이사이에 JSP 문법의 코드가 삽입됩니다

JSP 페이지의 형태

- JSP 페이지에 있는 HTML 코드는 웹 브라우저로 그대로 전송되지만, JSP 문법의 코드는 웹 컨테이너 쪽에서 실행되고 그 결과만 웹 브라우저로 전송된다.

- 웹 컨테이너는 JSP 페이지 전체를 서블릿 클래스의 소스 코드로 변환한 다음에, 그 소스 코드를 컴파일해서 그 결과를 가지고 서블릿 객체를 만들고, 그 서블릿 객체를 초기화해서 서블릿을 만든다.
- 웹 브라우저로부터 URL이 왔을 때 실행되는 것은 서블릿이다.



JSP 문서의 기본 구조

```
1 <%@ page language="java" contentType="text/html; charset=EUC-KR"
2   pageEncoding="EUC-KR"%>
3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
4 <html>
5 <head>
6 <meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
7 <title>Insert title here</title>
8 </head>
9 <body>
10
11 </body>
12 </html>
```

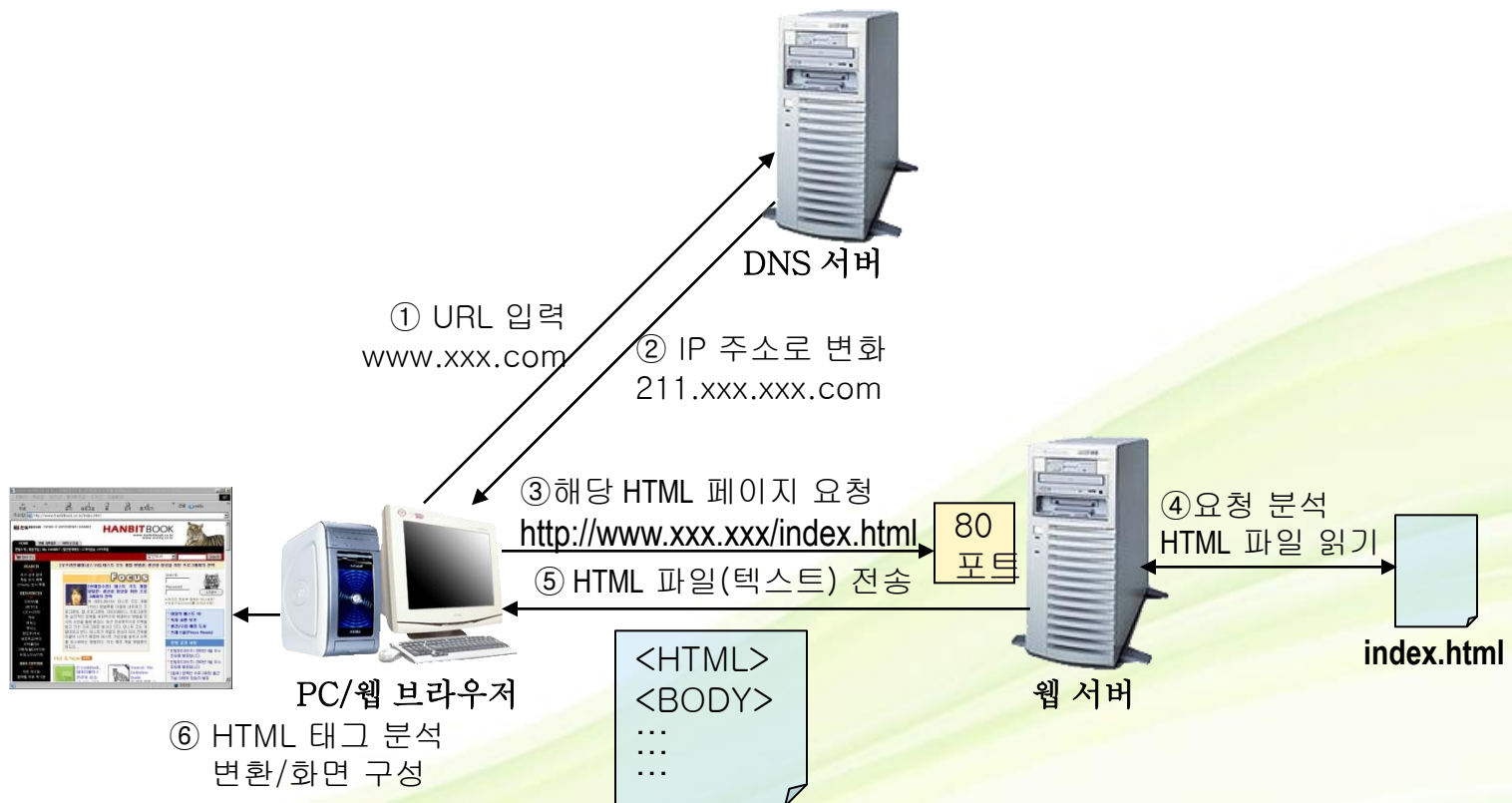
- ❖ <!DOCTYPE ...> 이전까지는 JSP 페이지에 대한 정보를 이용하는 설정부분
 - ❖ JSP 페이지가 생성하는 문서의 타입 및 사용할 커스텀 태그, 사용할 자바 클래스를 지정하는 부분
- ❖ <!DOCTYPE ..> 이후 부분은 문서를 생성하는 생성부분
 - ❖ 문서의 데이터와 문서를 생성하는데 필요한 스크립트 코드를 작성하는 부분
 - ❖ <% ... %> 부분이 스크립트 코드

JSP 문서의 구성 요소

- ❖ Directive(디렉티브)
- ❖ Script : Scriptlet, Expression, Declaration
- ❖ Expression Language(표현 언어)
- ❖ Implicit Object(기본 객체)
- ❖ Static Data
- ❖ Action Tag
- ❖ Custom Tag & JSTL

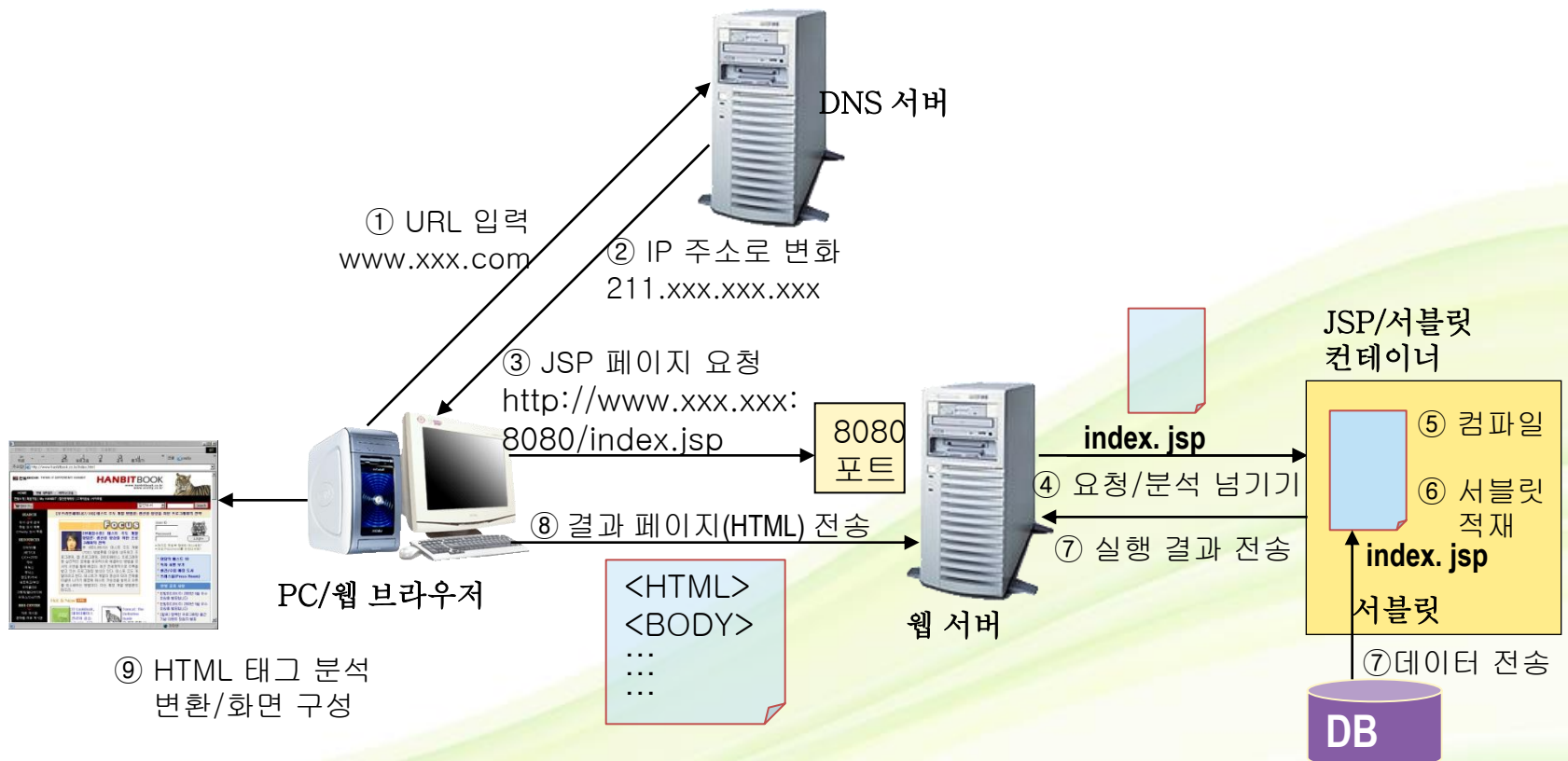
❖ JSP 동작원리

■ 일반적인 웹(www) 서비스 동작과정



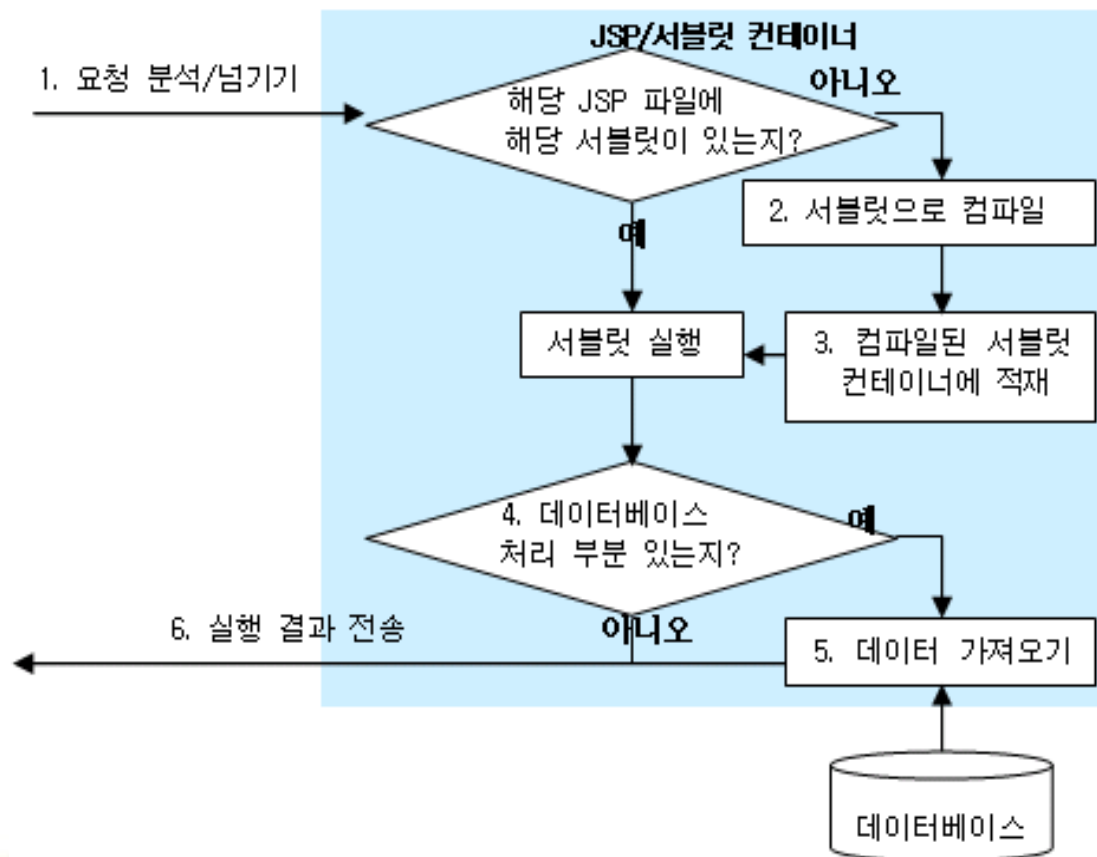
❖ JSP 동작원리

■ JSP 동작과정



❖ JSP 동작원리

■ JSP 서블릿 변환 처리 및 데이터 연동 과정



❖ JSP를 배우기 위한 필수 기술

필수 기술	프로그램 경험	비 고
자바	<ul style="list-style-type: none"> · 자바 언어 기본 · 객체지향 개념 · 상속, 오버로딩, 오버라이딩 · 인터페이스 구현 · java.util, java.io 패키지 · 스레드 · 예외 핸들링 	<ul style="list-style-type: none"> · 패키지와 클래스 이해 · 클 래 스 DOC 을 참 조 하 여 프로그래밍이 가능한 수준 · 자바 개발환경 설치 및 사용
JDBC	<ul style="list-style-type: none"> · JDBC 드라이버 세팅 · ResultSet · PreparedStatement · 데이터 핸들링 · 기초 SQL문 	<ul style="list-style-type: none"> · 오라클, MySQL 등 원격지 데이터 베이스 연결 처리 경험
서블릿	<ul style="list-style-type: none"> · 서블릿 구조 이해 · 간단한 서블릿 프로그래밍 · request, response 처리 · GET/POST 처리 	<ul style="list-style-type: none"> · 서블릿 생명주기 이해

❖ JSP를 배우는 데 도움이 되는 관련 기술

관련 기술	프로그램 경험	최소 요구사항
HTML	<ul style="list-style-type: none"> · HTML 기초 태그 사용 · FORM 관련 태그 사용 	<ul style="list-style-type: none"> · 전용 편집기가 아닌 수작업으로 코딩이 가능한 수준 · CSS, 레이어 이해
자바스크립트	<ul style="list-style-type: none"> · 함수(Function) 만들기 · FORM 연계 · 이벤트 처리 	<ul style="list-style-type: none"> · 자바스크립트 문법 이해 · 브라우저 객체 모델 이해
데이터베이스	<ul style="list-style-type: none"> · 다양한 SQL문의 사용 · 데이터베이스 연계 프로그래밍 경험 · 데이터베이스 함수 및 내장프로시저 	<ul style="list-style-type: none"> · 테이블 생성 및 키에 대한 이해와 관계 설정
웹 프로그래밍	<ul style="list-style-type: none"> · 웹 서버 세팅 · CGI, ASP, PHP 등 웹 프로그래밍 경험 	<ul style="list-style-type: none"> · 유닉스에서 웹 서버 세팅 경험
XML	<ul style="list-style-type: none"> · XML 스키마 및 DTD 이해 · XML DOM 개요 	<ul style="list-style-type: none"> · 스키마와 DTD 기반의 XML 문서 작성 및 파싱 능력

JSP의 기초 문법(스크립팅 요소, 지시자, 주석)

❖ JSP의 다양한 문법

■ JSP의 문법에는 세가지 형태가 있다.

- <%로 시작해서 %>로 끝나는 형태
- \${로 시작해서}로 끝나는 형태 : 익스프레션 언어(expression language)
- <jsp:forward>와 같은 XML : 태그 형태 액션(action)

■ 지시자와 스크립팅 요소

- <%로 시작해서 %>로 끝난다.

```
<%@page contentType= "text/html; charset=euc-kr" %>
<HTML>
  <HEAD><TITLE>1부터 100까지의 합</TITLE></HEAD>
  <BODY>
    <%
      int total = 0;
      for (int cnt = 1; cnt <= 100; cnt++)
        total += cnt;
    %>
    1부터 100까지 더한 값은? <%= total %>
  </BODY>
</HTML>
```

지시자(directive)

스크립틀릿(scriptlet)

익스프레션(expression)

JSP의 문법 – 지시자와 스크립팅 요소

- 스크립팅 요소(scripting elements)란 다음 세가지 문법을 말한다.
 - 스크립틀릿(scriptlet)
 - 익스프레션(expression)
 - 선언부(declaration)
- 스크립틀릿(scriptlet)은 <%로 시작해서 %> 로 끝나고, 그 사이 자바 명령문이 들어갈 수 있다.

<%

```
int total = 0;  
for (int cnt = 1; cnt <= 100; cnt++)  
total += cnt;
```

%>

자바 명령문들

- 이 명령문은 웹 브라우저로 전송되는 것이 아니라 웹 서버 쪽에 실행 된다.

- 익스프레션(expression)은 <%=로 시작해서 %>로 끝나고 그 사이에 자바 식이 들어갈 수 있다.
 - 이 식은 상수나 변수 이름 하나로 구성될 수도 있고, 연산자를 포함할 수도 있으며, 리턴 값이 있는 메서드 호출식이 될 수도 있다.

<%= total %>

자바 식

<%= total + 101 %>

자바 식

<%= Math.sqrt(num) %>

자바 식

- 이 식은 웹 서버 쪽에서 실행되고 그 결과만 웹 브라우저로 전송된다.

test.jsp

```
<%@ page contentType="text/html; charset=euc-kr" %>
<html>
<body>
    <%    // 스크립틀릿
        String strName = "홍길동";
    %>
    <!-- expression -->
    내 이름은 <%= strName %>   입니다!!!
</body>
</html>
```

Korea.jsp

```
<h1>Declaration Example1</h1>
```

```
<%
```

```
    String name = team + " Fighting!!!";
```

```
%>
```

```
<%! // declaration
```

```
    String team = "Korea";
```

```
%>
```

출력되는 결과는 ? <%=name%>

JSP 문서의 구성 요소

❖스크립트 요소: 자바 코드를 작성하고 코드에서 작성한 변수나 계산식 및 메소드의 결과를 출력하기 위해서 사용

- ❖Expression(표현식): 값을 출력

- ❖Scriptlet: 자바코드

- ❖Declaration: jsp 파일의 멤버변수 및 메서드 선언

❖Implicit Object: 웹 애플리케이션을 작성하는데 필요한 기능을 제공해주는 객체로 별도의 생성과정 없이 사용할 수 있는 객체

- ❖request

- ❖response

- ❖session

- ❖application

- ❖page

- ❖기타

❖ 스크립팅 요소의 문법

JSP 페이지의 코드

```
<HTML>
  <HEAD><TITLE>1부터 100까지의 합</TITLE></HEAD>
  <BODY>
    <%
      int total = 0;
      for (int cnt = 1; cnt <= 100; cnt++)
        total += cnt;
    %>
    1부터 100까지 더한 값은? <%= total %>
  </BODY>
</HTML>
```



서블릿 클래스의 코드

```
out.println( "<HTML> ");
out.println( "<HEAD><TITLE>1부터 100까지의 합</TITLE></HEAD> ");
out.println( "<BODY> ");
int total = 0;
for (int cnt = 1; cnt <= 100; cnt++)
  total += cnt;
out.print( "1부터 100까지 더한 값은? " );
out.println(total);
out.println( "</BODY> ");
out.println( "</HTML> ");
```

스크립틀릿 안에 있던
자바 명령문

익스프레션 안에 있던 자바 식

[그림 3-6] JSP 페이지의 코드가 서블릿 클래스의 코드로 변환되는 방법

- JSP 페이지의 스크립틀릿은 모두 `_jspService` 메서드 안에 들어가는 명령문이 되므로, 한 스크립틀릿 안에서 선언한 변수를 그 뒤에 나오는 다른 스크립틀릿 안에 사용하는 것이 가능하다.

여러 개의 스크립틀릿이 있는 JSP 페이지

```
<% @page contentType= "text/html; charset=euc-kr" %>
<HTML>
  <HEAD><TITLE>1부터 200까지의 합</TITLE></HEAD>
  <BODY>
    <%
      int total = 0;
      for (int cnt = 1; cnt <= 100; cnt++)
        total += cnt;
    %>
    1부터 100까지의 합 = <%= total %> <BR>
    <%
      for (int cnt = 101; cnt <= 200; cnt++)
        total += cnt;
    %>
    1부터 200까지의 합 = <%= total %> <BR>
  </BODY>
</HTML>
```

total 변수를 선언한다

total 변수를 사용한다

- 선언부(declaration)는 <%!로 시작해서 %>로 끝나고, 그 사이에 변수 선언문이나 메서드 선언문을 쓸 수 있다.
- 여기에 선언한 변수, 메서드는 JSP 페이지로부터 변환된 서블릿 클래스의 멤버가 되므로, final, public, private, protected, static 등의 키워드를 붙이는 것도 가능하다.

```
<%! final static int MAX=10000; %>
```

변수 선언

```
<%!  
private int add(int num1, int num2) {  
    int sum = num1 + num2;  
    return sum;  
}  
%>
```

메서드 선언

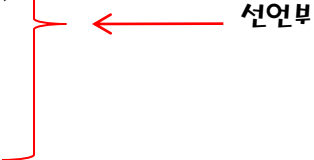
- 선언부 안에 변수를 선언할 때 주의해야 할 점

- JSP 페이지로부터 변환된 서블릿 클래스는 기본적으로 멀티-스레드 모델로 작동하고, 멀티-스레드 모델로 작동하는 서블릿 클래스 안에는 인스턴스 변수를 선언하면 안 된다.

■ 선언부의 사용 예를 보여주는 JSP 페이지

선언부를 포함한 JSP 페이지

```
<% @page contentType= "text/html; charset=euc-kr" %>
<HTML>
  <HEAD><TITLE>2의 거듭제곱</TITLE></HEAD>
  <BODY>
    2 ^ 1 = <%= power(2, 1) %> <BR>
    2 ^ 2 = <%= power(2, 2) %> <BR>
    2 ^ 3 = <%= power(2, 3) %> <BR>
    2 ^ 4 = <%= power(2, 4) %> <BR>
    2 ^ 5 = <%= power(2, 5) %> <BR>
  </BODY>
</HTML>
<%!
  private int power(int base, int exponent){
    int result= 1;
    for (int cnt = 0; cnt < exponent; cnt++)
      result *= base;
    return result;
  }
%>
```



선언부

page 디렉티브

- ❖ contentType 속성과 charset: 결과를 생성할 때의 문서 타입과 문자 인코딩 방식
 - ❖ TYPE 또는 TYPE; charset=캐릭터 셋
 - ❖ TYPE에는 MIME TYPE을 입력(text/html, text/xml, text/plain..)-기본값은 text/html
 - ❖ charset은 생략이 가능한데 생략하면 ISO-8859-1로 지정됩니다.
 - ❖ <%@ page contentType="text/html; charset=EUC-KR" %>
 - ❖ 서블릿으로 변환될 때 response.setContentType("text/html; charset=euc-kr")로 변환됩니다.

❖ 자바 언어가 제공하거나 직접 작성한 패키지의 클래스를 사용하고자 할 때 경로를 줄여서 사용하기 위한 방법으로 import 속성 사용

❖ `<%@ page import = "java.util.Calendar, java.util.Date...." %>`

❖ jsp 페이지는 javax.servlet, javax.servlet.jsp, javax.servlet.http 이 3개의 패키지는 자동으로 import되어 있습니다.

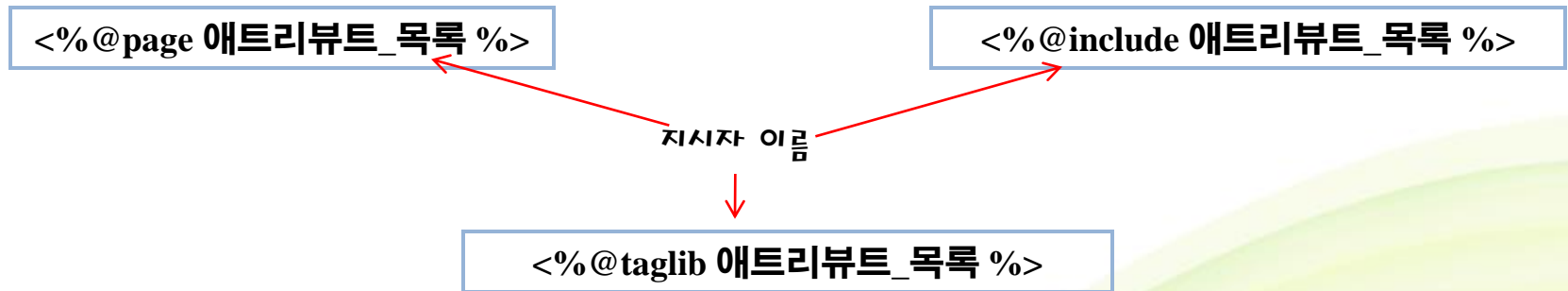
❖ include 지시자는 다른 jsp 페이지나 html 문서를 불러다가 현재 페이지의 일부로 만들기 위해서 사용하는 것으로 file 애트리뷰트를 이용하면 됩니다.

❖ `<%@include file="today.jsp">`, `<%@include file="sub/today.jsp">`

❖ html 문서로 변환될 때 불필요한 공백을 제거하기 위해서는 trimDirectiveWhitespaces 속성의 값을 true로 설정하면 됩니다.

❖ pageEncoding 속성에 jsp 페이지를 구현한 파일의 인코딩 방식이나 파일을 읽어올 때 사용할 인코딩을 지정

- 모든 지시자는 <%@으로 시작하고 %>로 끝나야 한다.
- <%@ 바로 다음에는 지시자 이름이 와야 하고, 지시자 이름 다음에는 여러 가지 애트리뷰트가 올 수 있다.



- page 지시자는 JSP 페이지 전체에 적용되는 정보를 기술하기 위해 사용된다.

애틀리뷰트 이름	기술하는 정보/애틀리뷰트의 역할
contentType	JSP 페이지가 생성하는 문서의 종류와 그 문서를 웹 브라우저로 전송할 때 사용되는 인코딩 타입
import	스크립팅 요소 안에서 사용할 자바 클래스와 인터페이스를 임포트하기 위해 사용하는 애틀리뷰트
buffer	출력 버퍼의 크기, default = 8Kb
autoFlush	출력 버퍼가 모두 찼을 때의 동작
isThreadSafe	JSP 페이지가 싱글-스레드 모드로 작동하도록 만들기 위해 필요한 애틀리뷰트
session	JSP 페이지의 세션 참여 여부
errorPage	에러를 처리할 JSP 페이지의 URL
isErrorPage	에러를 처리하는 JSP 페이지인지 여부
isELIgnored	익스프레션 언어의 무시/처리 여부
pageEncoding	JSP 페이지의 인코딩 타입
info	JSP 페이지에 대한 설명
extends	JSP 페이지로부터 생성되는 서블릿 클래스의 슈퍼클래스
language	스크립팅 요소 안에서 사용할 프로그래밍 언어. 현재는 ‘java’ 라는 값만 지정할 수 있음
deferredSyntaxAllowedAsLiteral	익스프레션 언어의 예약 문자열인 ‘#{’ 를 사용했을 때의 에러 발생 여부
trimDirectiveWhitespaces	지시자 바로 다음에 있는 공백 문자를 제거하기 위해 사용하는 애틀리뷰트

JSP 문서의 구성 요소

❖ Directive(디렉티브, 지시자)

- ❖ JSP 페이지에 대한 정보를 설정할 때 사용

- ❖ `<%@ 디렉티브이름 속성="값" 속성="값" ... %>` 의 형태로 작성

- ❖ `<%@ page language="java" contentType="text/html; charset=EUC-KR" pageEncoding="EUC-KR"%>`

- ❖ page가 디렉티브 이름이고 contentType과 pageEncoding이 속성

- ❖ "text/html; charset=EUC-KR" , "EUC-KR" 이 값

- ❖ 제공되는 디렉티브

- ❖ page: JSP 페이지에 대한 정보를 지정하며 문서의 타입, 출력 버퍼의 크기, 에러 페이지 등의 정보를 입력

- ❖ taglib: 태그 라이브러리를 지정

- ❖ include: 특정 영역에 다른 문서를 포함

PageDirectiveInfo.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
    Transitional//EN">
<html>
<head>
<%@ page language="java" contentType="text/html; charset=EUC-
    KR"
    pageEncoding="EUC-KR"%>
<%@ page info="copyright by Juni"%>
<meta http-equiv="Content-Type" content="text/html;
    charset=EUC-KR">
<title>page디렉티브 연습 - info 속성</title>
</head>
<body>
    <h2>page디렉티브 연습 - info 속성</h2>
    <%=getServletInfo() %>
</body>
</html>
```

- page 지시자의 import 애트리뷰트는 자바의 import 문과 마찬가지로 다른 패키지에 속하는 클래스나 인터페이스를 임포트하는 역할을 한다.

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

java.util 패키지의 GregorianCalendar 클래스를 임포트한다.

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

java.util 패키지의 모든 클래스와 인터페이스를 임포트한다

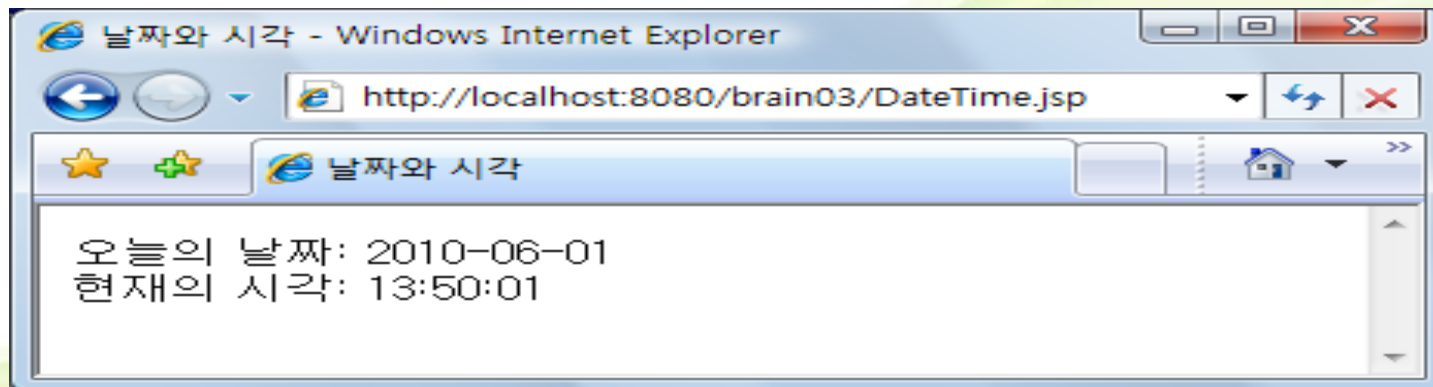
```
<%@page import="java.util.ArrayList, java.io.*" %>
```

java.util.ArrayList 클래스와 java.io 패키지의 모든 클래스, 인터페이스를 임포트한다

■ import 애트리뷰트의 사용 예를 보여주는 JSP 페이지

page 지시자의 import 애트리뷰트 사용 예

```
<% @page contentType= "text/html; charset=euc-kr" %>
<%@page import= "java.util.GregorianCalendar" %>
<HTML>
  <HEAD><TITLE>날짜와 시각</TITLE></HEAD>
  <BODY>
    <%
      GregorianCalendar now = new GregorianCalendar();
      String date = String.format( "%TF ", now);
      String time = String.format( "%TT ", now);
    %>
    오늘의 날짜: <%= date %> <BR>
    현재의 시각: <%= time %> <BR>
  </BODY>
</HTML>
```



- page 지시자에 여러 종류의 애트리뷰트를 함께 쓸 수도 있다.

```
<% @page contentType= "text/html; charset=euc-kr "
import= "java.util.GregorianCalendar " %>
```

여러 종류의 애트리뷰트를 함께 쓸 수 있다

- include 지시자는 다른 JSP 페이지 또는 HTML 문서를 불러다가 현재 JSP 페이지의 일부로 만들기 위해 사용한다.
- 불러올 대상은 file 애트리뷰트를 이용해서 지정할 수 있으며, 이 애트리뷰트의 값은 지시자가 속하는 JSP 페이지를 기준으로 한 상대적인 URL로 해석된다.

```
<%@include file= "Today.jsp " %>
```

현재 디렉터리에 있는 Today.jsp를
include한다

```
<%@include file= "sub1/Today.jsp " %>
```

sub1 디렉터리에 있는
Today.jsp를
include한다

❖ contentType속성

- type 또는 type; charset=캐릭터 셋
type은 생성할 응답문서의 MIME타입

❖ MIME

Multipurpose Internet Mail Extensions이 약자로 이메일 내용을 설명하기 위해 정의 되었음

❖ trimDirectiveWhitespces : html문서의 공백처리 지원

❖ BOM : Byte Order Mark의 약자로 UTF-16, UTF-32같은 유니코드에서 바이트 순서가 little endian인지 big endian인지 알려주는 값

❖ pageEncoding

웹 컨테이너가 캐릭터셋을 결정하는 방법

1. 파일이 BOM으로 시작하지 않을 때

- 기본 encoding으로 읽고 pageEncoding이 있으면
pageEncoding으로 읽고 없으면 contentType속성의
캐릭터셋으로 읽고 캐릭터셋이 없으면 ISO-8859-1을캐릭터
셋으로 사용

2. 파일이 BOM으로 시작할 경우

BOM을 이용해서 파일을 읽고 pageEncoding속성의 값과 BOM을
이용하여 결정된 인터딩이 다르면 에러 발생

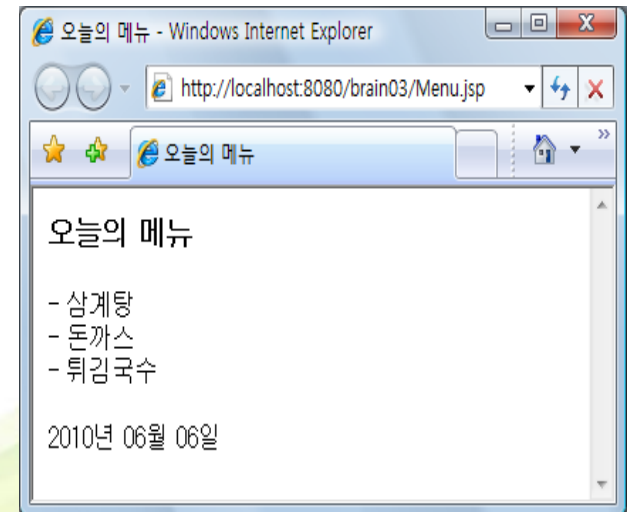
▪ include 지시자의 사용 예를 보여주는 JSP 페이지

include 지시자의 사용 예

```
<% @page contentType= "text/html; charset=euc-kr" %>
<HTML>
  <HEAD><TITLE>오늘의 메뉴</TITLE></HEAD>
  <BODY>
    <H3>오늘의 메뉴</H3>
    - 삼계탕 <BR>
    - 돈까스 <BR>
    - 튀김국수 <BR><BR>
    <%@include file= "Today.jsp" %>
  </BODY>
</HTML>
```

↑ Today.jsp를 include한다

```
<% @page contentType="text/html; charset=euc-kr"%>
<%@page import="java.util.*"%>
<% GregorianCalendar now = new GregorianCalendar(); %>
<%= String.format("%TY년 %Tm월 %Td일" , now, now, now) %>
```



❖ 지시자의 문법

- taglib 지시자는 JSP 문법 중 하나인 액션(action)을 사용할 때 필요하다.
- taglib 지시자는 액션이 속한 라이브러리를 설치해야만 사용할 수 있다.

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

액션이 속하는 라이브러리를 지정하는 지시자

스크립트 요소

❖ JSP페이지에서는 선언문(Declaration), 스크립트릿(Scriptlet), 표현식(Expression)이라는 3가지의 스크립트 요소를 제공한다.

- 선언문(Declaration) - `<%! %>` : 전역변수 선언 및 메소드 선언에 사용
- 스크립트릿(Scriptlet) - `<% %>` : 프로그래밍 코드 기술에 사용
- 표현식(Expression) - `<%= %>` : 화면에 출력할 내용 기술에 사용

선언문

선언문은 JSP 페이지 내에 안에서 필요한 멤버변수나 메소드가 필요할 때 쓰이는 요소
선언문에서 선언된 변수는 자바에서와 마찬가지로 전역변수 역할을 하는 멤버 변수가 된다.

선언문의 문법

`<%! 문장 %>`

❖ 선언문에서 변수 선언

❖ 선언문에서 선언된 변수는 JSP 페이지가 서블릿(Servlet) 으로 파싱(parsing)될 때 서블릿의 멤버변수가 된다.

■ 예시

```
<%!  
    private String name= HongGilDong";  
    private int year = 2007;  
%>
```

❖ 선언문에서 메소드 선언

■ 선언문에서 선언된 메소드는 JSP 페이지 내에서 일반적인 메소드로 사용된다

■ 예시

```
<%!  
    String id = "HongGilDong";  
    public String getId( ) {  
        return id;  
    }  
%>
```

scriptTest.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%>
<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>Script 예제</title> </head>
<body> <h2>선언문, 스크립트릿, 표현식의 쓰임을 알아보는 예제</h2>
    <%! //선언문 - 전역변수 선언
        String str = "전역변수 입니다";
    %>
    <%! //선언문 - 메소드 선언
        String getStr(){
            return str;
        }
    %>
    <% //스크립트릿
        String str2 = "지역변수 입니다.";
    %>
    스크립트릿에서 선언한 변수 str2는 <%=str2 %> <br> <!-- 표현식 -->
    선언문에서 선언한 변수 str1은 <%=getStr()%> <!-- 표현식 -->
</body></html>
```

declarationTest.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%>
<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>선언문 예제 - 변수선언</title>
</head>
<body>
    <h2>선언문 예제 - 변수선언</h2>
    <%
        String str1 = str2 + " Server Page";
    %>
    <%!
        String str2 = "Java"; // 반영
    %>
    출력결과 : <%=str1 %>
</body>
</html>
```

declarationTest2.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%>
<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>선언문 예제 - 메소드 선언</title>
</head>
<body>
    <h2>선언문 예제 - 메소드 선언</h2>
    <%!
        String id = "HongGilDong";

        public String getId( ) {
            return id;
        }
    %>
    id변수의 내용 : <%=id %><br>
    getId( )메소드의 실행결과 : <%=getId() %>
</body>
</html>
```

스크립트릿(Scriptlet)

- ❖ 스크립트릿(Scriptlet)은 JSP페이지에서 가장 일반적으로 많이 쓰이는 스크립트 요소로 주로 프로그래밍의 로직을 기술할 때 많이 쓰인다.
- ❖ JSP 페이지가 Servlet으로 변환되고 이 페이지가 호출 될 때 `_jspService` 메소드 안에 선언
- ❖ 스크립트릿(Scriptlet)에서 선언된 변수는 지역변수로 선언
- ❖ 스크립트릿의 문법
 - `<% 문장%>`
- ❖ 스크립트릿에서 선언한 변수는 지역 변수이므로 별도로 선언한 메소드 부분에서는 해당변수를 사용할 수 없다.
- ❖ 자동 초기화가 안되므로 반드시 초기화를 해 주어야 한다.
 - 예시

```
<%  
    String id = "HongGilDong";  
    String pass="";  
%>
```

Scriptlet2.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%>
<%@ page import="java.util.*" %>
<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>스크립트릿 예제 - 활용</title>
</head>
<body>
    <h2>스크립트릿 예제 - 활용</h2>
    <%
        Calendar getDate = Calendar.getInstance();
        Date nowTime = getDate.getTime();
    %>
    현재는 <%=nowTime.getHours()%>시 <%=nowTime.getMinutes()%>분 입니다.
</body>
</html>
```

표현식(Expression)

- ❖ 표현식(expression)은 JSP 페이지에서 웹 브라우저에 출력할 부분을 표현하기 위한 것이다.
- ❖ 표현식은 웹 브라우저에 출력을 목적으로 하는 변수의 값 및 메소드의 결과값도 출력할 수 있다.
 - 스크립트릿 코드 내에서 표현식을 쓸 수 없다. 대신 스크립트릿내에서 출력할 부분은 내장객체인 out객체의 print()또는 println()메소드를 사용해서 출력한다.
- ❖ 표현식의 일반적인 문법
 - <%=문장%>
- ❖ 표현식내에서는 세미콜론(;)은 생략하는데, 이는 JSP페이지가 서블릿으로 변환될 때 표현식부분은 out.print();메소드로 변환되어 자동적으로 세미콜론이 붙여지기 때문이다.
 - <%=name[i]%>

expressionTest2.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%>
<%@ page import="java.util.Date"%>
<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>표현식 예제 - 레퍼런스타입출력</title>
</head>
<body>
    <h2>표현식 예제 - 레퍼런스타입출력</h2>
    <%
        Date date = new Date();
    %>
    현재 날짜와 시간: <%=date.toString()%><p>
</body>
</html>
```

❖ 주석을 기술하는 방법

- JSP 페이지에 주석을 다는 방법은 다양하다.
- JSP 페이지의 HTML 코드 부분 : `<!--`로 시작해서 `-->`로 끝나는 HTML 주석을 쓸 수 있다.

`<!-- HTML의 주석 -->`

↑ 시작 표시 ↑ 끝 표시

- JSP 페이지의 스크립팅 요소 안 : 자바 문법을 따르는 주석을 쓸 수 있다.

`/* Java의 주석 */`

↑ 시작 표시 ↑ 끝 표시

`// Java의 주석`

↑ 시작 표시

- JSP 고유의 주석을 사용할 수 있다.

`<%-- JSP의 주석 --%>`

↑ 시작 표시 ↑ 끝 표시

❖ 주석을 기술하는 방법

■ 여러 가지 주석의 사용 예를 보여주는 JSP 페이지

여러 가지 주석을 포함하는 JSP 페이지

```
<% @page contentType= "text/html; charset=euc-kr" %>
```

```
<HTML>
```

```
<HEAD><TITLE>1부터 10까지의 곱</TITLE></HEAD>
```

```
<!-- 이것은 JSP에 의해 생성된 HTML 문서입니다. -->
```

```
<BODY>
```

```
<%-- 다음은 데이터를 처리하는 스크립틀릿입니다. --%>
```

```
<% int result = 1; // 곱을 저장할 변수
```

```
/* 1부터 10까지 곱하는 반복문 */
```

```
for (int cnt = 1; cnt <= 10; cnt++)
```

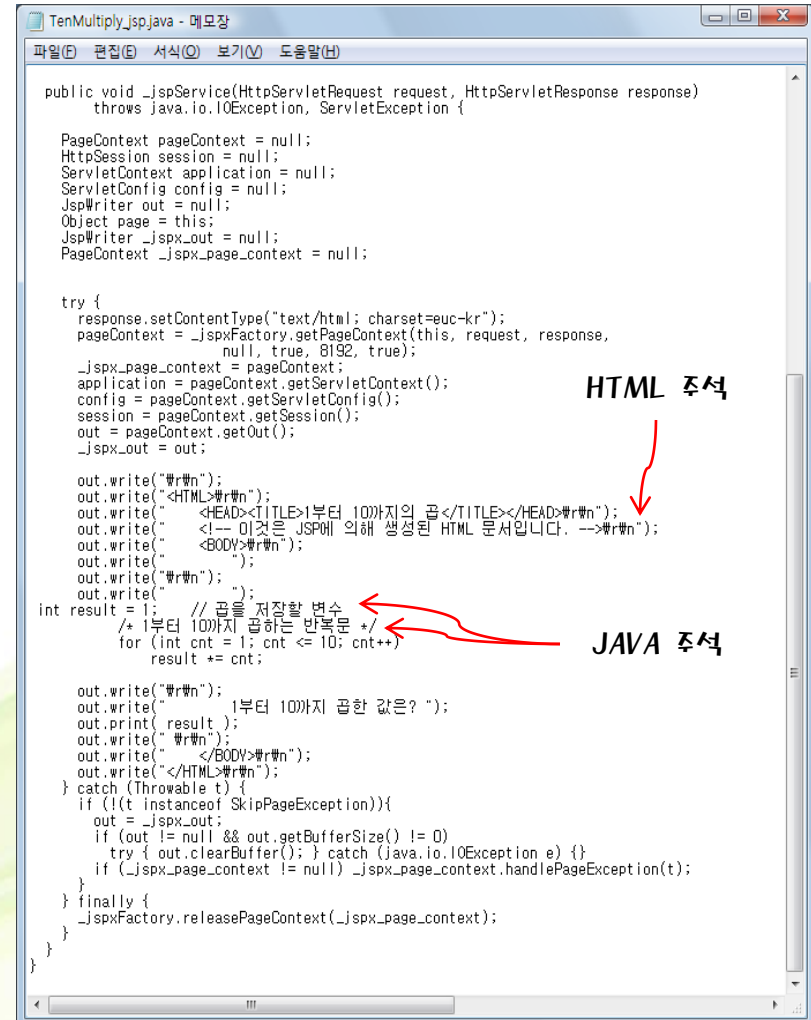
```
    result *= cnt;
```

```
%>
```

```
1부터 10까지 곱한 값은? <%= result %>
```

```
</BODY>
```

```
</HTML>
```



```
TenMultiply.jsp.java - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

public void _jspService(HttpServletRequest request, HttpServletResponse response)
    throws java.io.IOException, ServletException {

    PageContext pageContext = null;
    HttpSession session = null;
    ServletContext application = null;
    ServletConfig config = null;
    JspWriter out = null;
    Object page = this;
    JspWriter _jspx_out = null;
    PageContext _jspx_page_context = null;

    try {
        response.setContentType("text/html; charset=euc-kr");
        pageContext = _jspxFactory.getPageContext(this, request, response,
            null, true, 8192, true);
        _jspx_page_context = pageContext;
        application = pageContext.getServletContext();
        config = pageContext.getServletConfig();
        session = pageContext.getSession();
        out = pageContext.getOut();
        _jspx_out = out;

        out.write("\r\n");
        out.write("<HTML>\r\n");
        out.write("<HEAD><TITLE>1부터 10까지의 곱</TITLE></HEAD>\r\n");
        out.write("<!-- 이것은 JSP에 의해 생성된 HTML 문서입니다. -->\r\n");
        out.write("<BODY>\r\n");
        out.write(" ");
        out.write("\r\n");
        out.write(" ");
        int result = 1; // 곱을 저장할 변수
        /* 1부터 10까지 곱하는 반복문 */
        for (int cnt = 1; cnt <= 10; cnt++)
            result *= cnt;

        out.write("\r\n");
        out.write("    1부터 10까지 곱한 값은? ");
        out.print(result);
        out.write("\r\n");
        out.write("</BODY>\r\n");
        out.write("</HTML>\r\n");
    } catch (Throwable t) {
        if (!(t instanceof SkipPageException)){
            out = _jspx_out;
            if (out != null && out.getBufferSize() != 0)
                try { out.clearBuffer(); } catch (java.io.IOException e) {}
            if (_jspx_page_context != null) _jspx_page_context.handlePageException(t);
        }
    } finally {
        _jspxFactory.releasePageContext(_jspx_page_context);
    }
}
```

- JSP 페이지의 내장 변수(implicit variable) : JSP 페이지 안에 선언을 하지 않고도 사용할 수 있는 변수

```
<% @page contentType= "text/html; charset=euc-kr" %>
<HTML>
  <HEAD><TITLE>정수를 순서대로</TITLE></HEAD>
  <BODY>
    <H3>정수를 순서대로</H3>
    <%
      String str = request.getParameter( "MAX " );
      int max = Integer.parseInt(str);
      for (int cnt = 1; cnt <= max; cnt++)
        out.println(cnt + "<BR> ");
    %>
  </BODY>
</HTML>
```

내장 변수

내장 변수

JSP 페이지의 내장 변수의 예

- request 내장 변수는 서블릿 클래스의 doGet, doPost 메서드의 첫 번째 파라미터와 동일한 역할을 한다.
- out 내장 변수는 서블릿 클래스에서 getWriter 메서드를 호출해서 얻은 PrintWriter 객체와 마찬가지로의 역할을 한다.

- JSP 페이지 안에서 내장 변수를 사용할 수 있는 이유는 웹 컨테이너가 JSP 페이지를 서블릿 클래스로 변환할 때 자동으로 내장 변수를 선언하기 때문이다.
- JSP 페이지에서 사용할 수 있는 내장 변수들

변수 이름	제공하는 기능/변수의 역할	변수 타입
request	doGet, doPost 메서드의 첫 번째 파라미터와 동일한 역할	javax.servlet.http.HttpServletRequest
response	doGet, doPost 메서드의 두 번째 파라미터와 동일한 역할	javax.servlet.http.HttpServletResponse
out	웹 브라우저로 HTML 코드를 출력하는 기능	javax.servlet.jsp.JspWriter
application	JSP 페이지가 속하는 웹 애플리케이션에 관련된 기능	javax.servlet.ServletContext
config	JSP 페이지의 구성 정보를 가져오는 기능	javax.servlet.ServletConfig
pageContext	JSP 페이지 범위 내에서 사용할 수 있는 데이터 저장 기능 등	javax.servlet.jsp.PageContext
session	세션에 관련된 기능	javax.servlet.http.HttpSession
page	JSP 페이지로부터 생성된 서블릿	java.lang.Object
exception	익셉션 객체	java.lang.Throwable

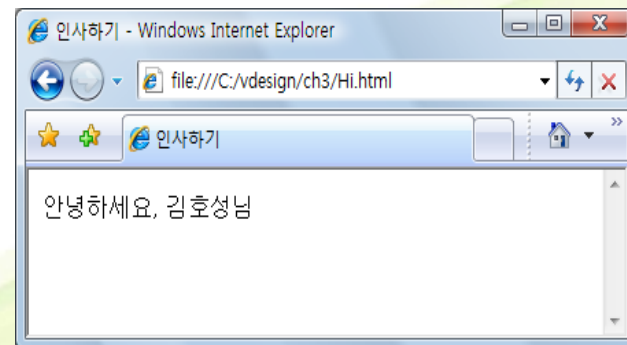
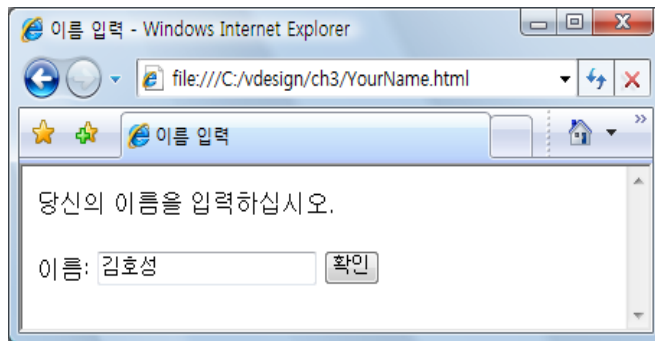
❖ request 내장 변수

- request 내장 변수는 서블릿 클래스에 있는 doGet, doPost 메서드의 첫 번째 파라미터와 동일한 역할을 하고, 타입도 동일하게 javax.servlet.http.HttpServletRequest이다.

```
String str = request.getParameter( "NAME " );
```

↑
데이터 이

- 웹 브라우저를 통해 입력된 데이터가 처리하는 애플리케이션을 작성해보자.



인사말을 출력하는 웹 애플리케이션의 화면 설계

JSP 페이지의 내장 변수

❖ request 내장 변수

- 둘 이상의 화면으로 구성된 애플리케이션은 먼저 URL을 정한 뒤 각 URL에 해당하는 코드를 작성하는 것이 좋다.

http://localhost:8181/ch03_web/YourName.html ←

왼쪽 화면 URL

http://localhost:8181/ch03_web/Hi.jsp ←

오른쪽 화면 URL

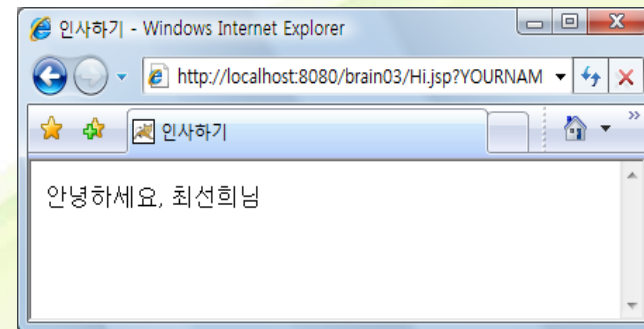
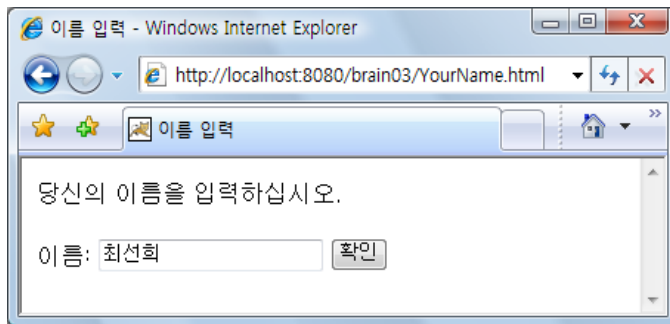
웹 브라우저로부터 이름을 입력받는 HTML 문서

```
<HTML>
  <HEAD>
    <META http-equiv= "Content-Type" content= "text/html; charset=euc-kr" >
    <TITLE>이름 입력</TITLE>
  </HEAD>
  <BODY>
    당신의 이름을 입력하세요.
    <FORM ACTION=/ch03_web/Hi.jsp METHOD=GET>
      이름: <INPUT TYPE=TEXT NAME=YOURNAME>
      <INPUT TYPE=SUBMIT VALUE= '확인' >
    </FORM>
  </BODY>
</HTML>
```

❖ request 내장 변수

입력된 이름을 가지고 인사말을 출력하는 JSP 페이지

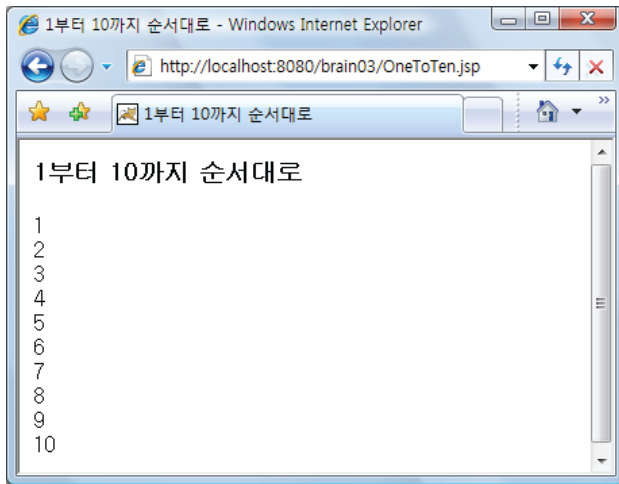
```
<%@page contentType="text/html; charset=euc-kr"%>
<HTML>
  <HEAD> <TITLE> 인사하기 </TITLE> </HEAD>
  <BODY>
    <%
      String str = request.getParameter("YOURNAME");
      String st= new String (str.getBytes("8859_1"), "euc-kr");
    %>
    안녕하세요, <%=st %> 님
  </BODY>
</HTML>
```



실행 결과

❖ out 내장 변수

- JSP 페이지에서는 HTML 코드와 익스프레션만 가지고도 원하는 HTML 문서를 만들어서 출력할 수 있기 때문에, 서블릿 클래스의 경우처럼 `println`, `print`, `printf` 메서드를 굳이 호출해야 할 필요가 없다.



1부터 10까지의 정수를 순서대로
출력하는 웹 페이지

입력된 이름을 가지고 인사말을 출력하는 JSP 페이지

```
<% @page contentType= "text/html; charset=euc-kr" %>

<HTML>

  <HEAD><TITLE>1부터 10까지 순서대로</TITLE></HEAD>

  <BODY>

    <H3>1부터 10까지 순서대로</H3>

    <% for (int cnt = 1; cnt <= 10; cnt++) { %>

      <%= cnt %> <BR>

    <% } %>

  </BODY>

</HTML>
```

❖ out 내장 변수

- out 내장 변수는 서블릿 클래스에서 `getWriter` 메서드를 호출해서 얻은 `PrintWriter` 객체와 비슷한 역할을 한다.

```
out.print( "<FONT SIZE=1>안녕하세요!</FONT> ");
```

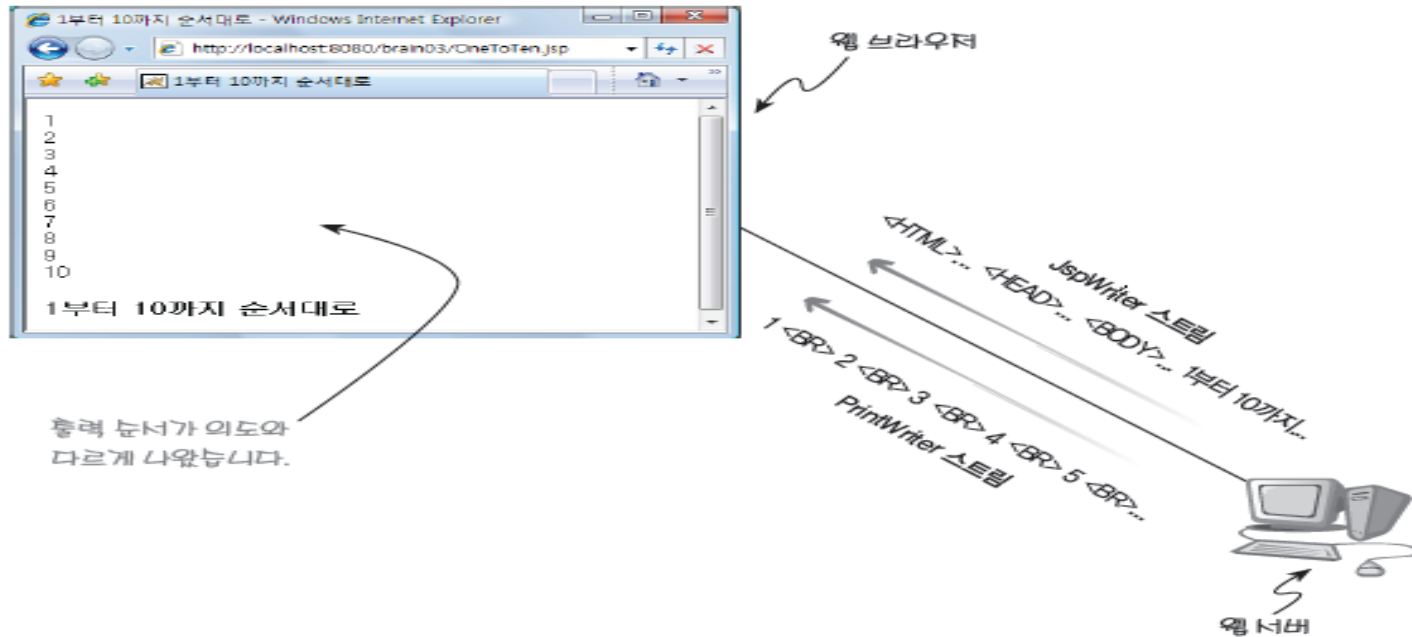
```
out.println( "<BR> ");
```

out 내장 변수를 사용하는 JSP 페이지

```
<% @page contentType= "text/html; charset=euc-kr" %>
<HTML>
  <HEAD><TITLE>1부터 10까지 순서대로</TITLE></HEAD>
  <BODY>
    <H3>1부터 10까지 순서대로</H3>
    <%
      for (int cnt = 1; cnt <= 10; cnt++)
        out.println(cnt + "<BR> ");
    %>
  </BODY>
</HTML>
```

❖ out 내장 변수

- JspWriter 나 PrintWriter처럼 스트림 형태로 데이터를 출력하는 클래스는 송신측과 수신측 사이에 가상의 통로를 만든다.
- JSP 페이지에서 PrintWriter 객체를 새로 만들면 기존의 out 내장 변수가 관리하던 통로와 더불어 두 개의 통로가 공존하게 된다.



response.getWriter 메서드를 호출하면 안 되는 이유

❖ out 내장 변수

- out.print, out.println 메서드를 통해 출력되는 내용뿐만 아니라, JSP 페이지 안에 있는 HTML 코드와 익스프레션을 통해 출력되는 내용도 모두 out 내장 변수를 통해 웹 브라우저로 출력된다.
- page 지시자의 buffer 애트리뷰트를 이용하면 출력 버퍼의 크기를 바꿀 수 있다. buffer 애트리뷰트에는 버퍼의 크기를 킬로바이트 단위의 정수로 써야 하며, 뒤에 kb라는 단위 표시를 붙여 써야 한다.

```
<%@page buffer="2kb" %>
```

출력 버퍼의 크기

- 버퍼의 실제 크기를 알고 싶을 경우에는 out 내장 변수에 대해 getBufferSize라는 메서드를 호출하면 된다.

```
int bsize = out.getBufferSize();
```

출력 버퍼의 크기를 바이트 단위로 리턴하는 메서드

❖ out 내장 변수

- page 지시자의 buffer 애트리뷰트에 none이라는 값을 지정하면 out 내장 변수를 통해 출력되는 내용이 출력 버퍼를 거치지 않고 웹 브라우저로 바로 전송되도록 만들 수 있다.

```
<%@page buffer= "none" %>
```

출력 버퍼를 사용하지 않겠다는 표시

출력 버퍼의 크기를 바꾸는 JSP 페이지

```
<%@page contentType= "text/html; charset=euc-kr" %>
```

```
<%@page buffer= "4kb" %>
```

```
<HTML>
```

```
<HEAD><TITLE>출력 버퍼의 크기
```

```
지정</TITLE></HEAD>
```

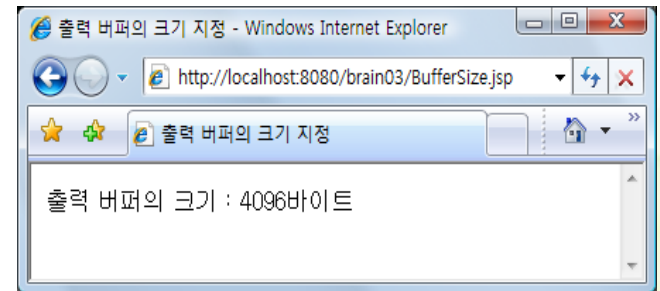
```
<BODY>
```

```
출력 버퍼의 크기 : <%=
```

```
out.getBufferSize() %>바이트
```

```
</BODY>
```


```
</HTML>
```



❖ response 내장 변수

- response 내장 변수는 서블릿 클래스에 있는 doGet, doPost 메서드의 두 번째 파라미터와 동일한 역할을 한다. 이 변수는 javax.servlet.http.HttpServletResponse 타입이기 때문에 이 인터페이스에 속하는 여러 가지 메서드들을 호출 할 수 있다.

```
response.sendRedirect( "http://www.daum.net/ ");
```

 호출할 웹 자원의 URL

- sendRedirect 메서드를 호출할 때 주의할 점: 이 메서드를 호출하기 전과 후에 웹 브라우저로 데이터를 출력하면 안 된다.

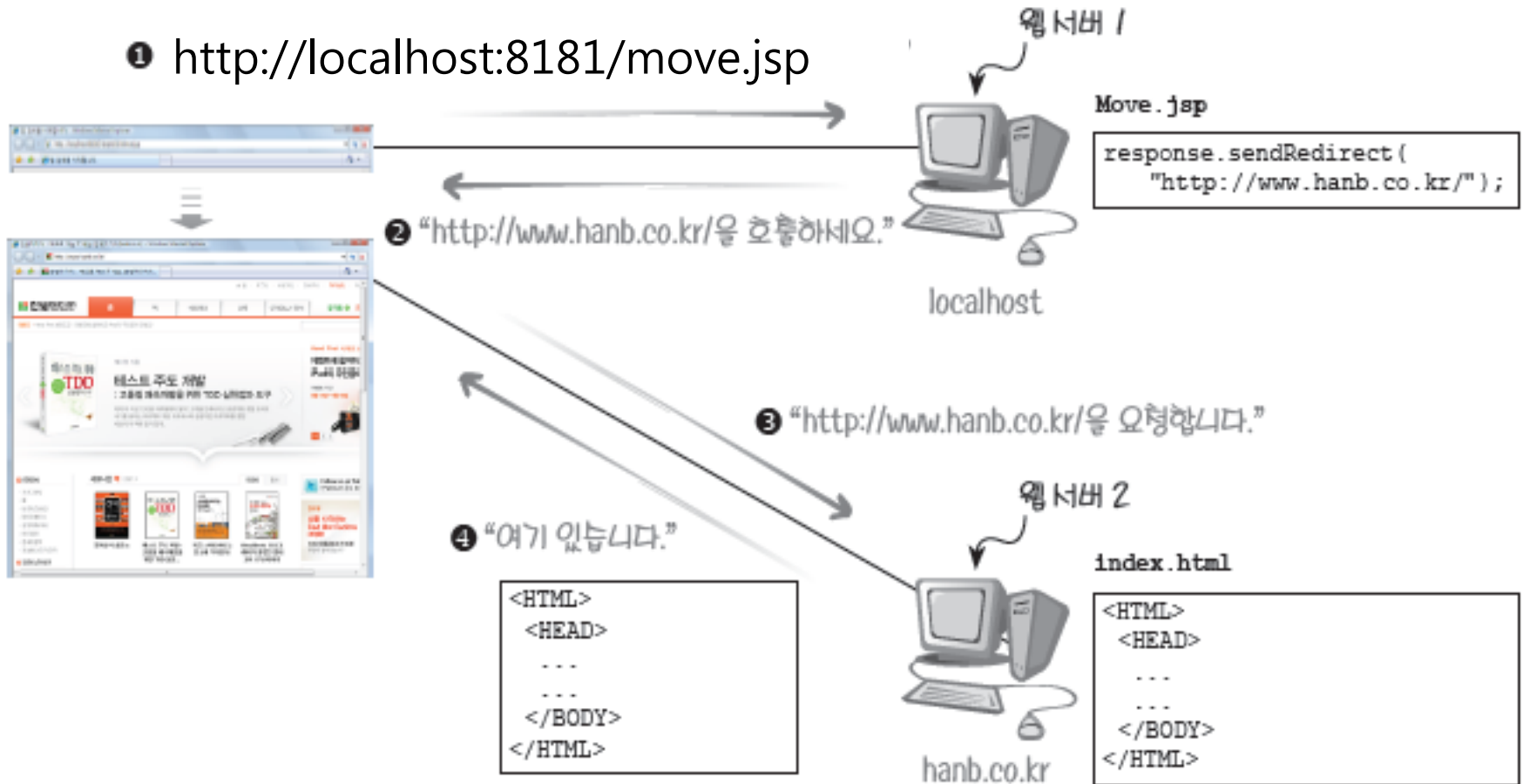
다른 웹 페이지로 이동하는 JSP 페이지

```
<% response.sendRedirect( "http://www.daum.net/ "); %>
```

❖ response 내장 변수

- sendRedirect 메서드는 파라미터로 지정한 URL을 직접 호출하는 것이 아니라 그 URL을 이용해서 다시 웹 자원을 호출하라는 메시지를 웹 브라우저로 보낼 뿐이다.

❶ http://localhost:8181/move.jsp



sendRedirect 메서드의 작동 원리

❖ application 내장 변수

- application 내장 변수는 웹 애플리케이션에 관련된 여러 가지 기능을 제공한다.
- application 내장 변수에 대해 호출할 수 있는 `getContextPath` 메서드는 웹 애플리케이션의 URL 경로명을 리턴하는 메서드이다.

```
String appPath = application.getContextPath();
```

↑
웹 애플리케이션의 URL 경로명을 리턴하는 메서드

- application 내장 변수에 대해 호출할 수 있는 `getRealPath` 메서드는 웹 애플리케이션 내에서의 파일 경로명을 파일시스템 전체에 대한 절대 경로명으로 바꾸는 메서드이다.

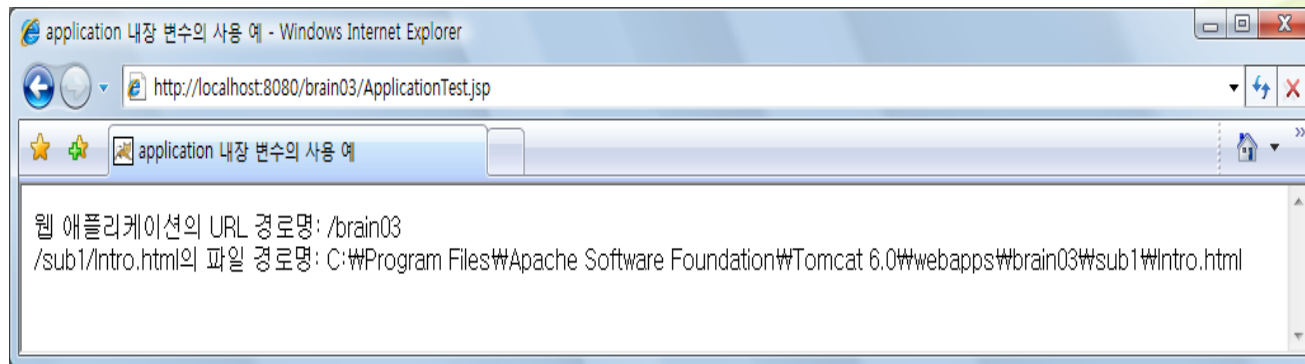
```
String absolutePath = application.getRealPath( "/sub1/Intro.html " );
```

↑
웹 애플리케이션 내에서의 파일의 경로명

❖ Application 내장 변수

[application 내장 변수의 사용 예]

```
<% @page contentType= "text/html; charset=euc-kr" %>
<HTML>
  <HEAD><TITLE>application 내장 변수의 사용 예</TITLE></HEAD>
  <BODY>
    <%
      String appPath = application.getContextPath();
      String filePath = application.getRealPath( "/sub1/Intro.html " );
    %>
    웹 애플리케이션의 URL 경로명: <%= appPath %> <BR>
    /sub1/Intro.html의 파일 경로명: <%= filePath %> <BR>
  </BODY>
</HTML>
```

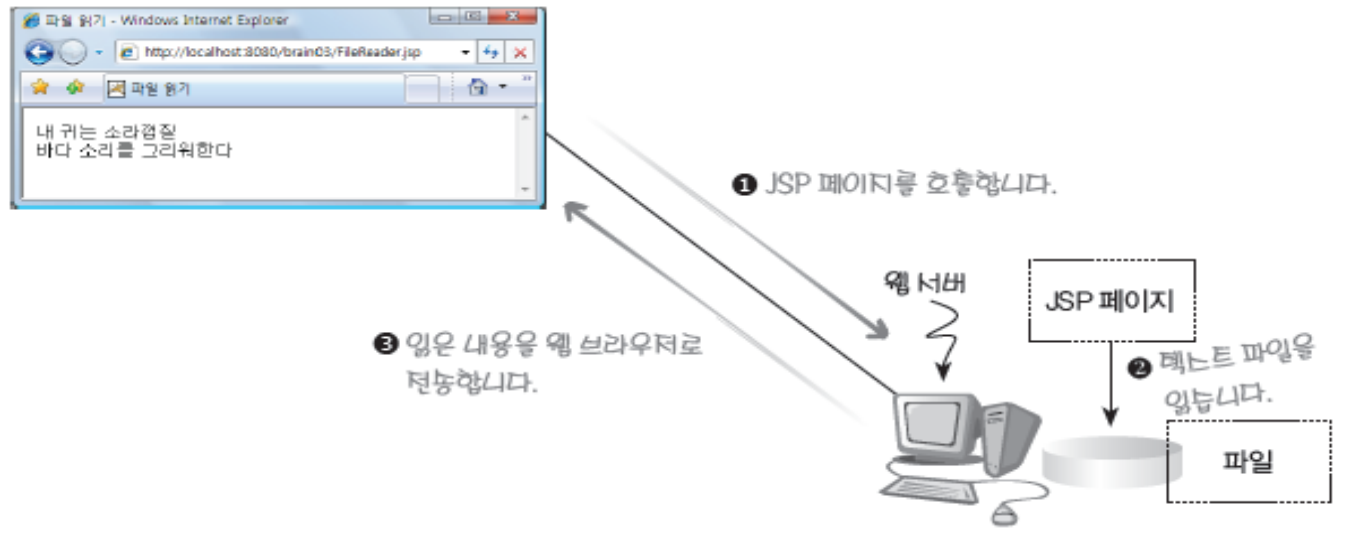


실행 결과

파일 입출력

❖ 파일로부터 데이터를 읽는 방법

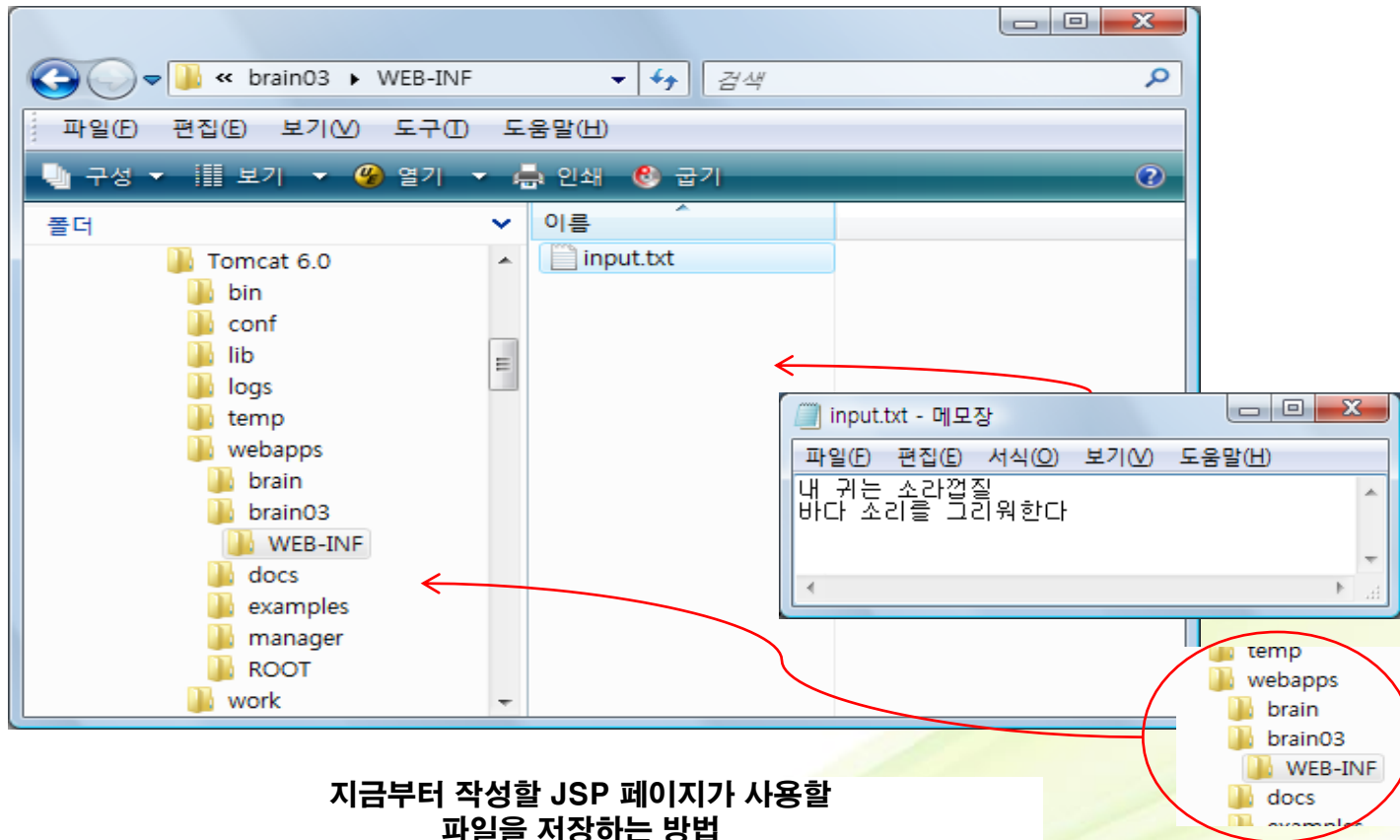
- 파일을 읽는 웹 애플리케이션의 구성도



[그림 3-25] 텍스트 파일을 읽어서 출력하는 웹 애플리케이션의 구성도

- JSP 페이지나 서블릿 클래스는 스탠드얼론 프로그램과 달리 웹 서버의 일부가 되어서 실행된다.

- 웹 브라우저에서 해당 파일의 URL을 통해 파일의 내용을 직접 읽을 수 없도록 만들려면 WEB-INF 디렉터리에 저장해야 한다.



- JSP 페이지 안에서 파일을 읽기 위해서는 기본적으로 파일의 절대 경로를 사용하거나 톰캣의 설치 디렉터리로부터 상대 경로명을 사용해야 한다.
- 하지만 더 좋은 방법은 `getRealPath` 메서드를 이용해서 웹 애플리케이션 내에서의 경로명을 절대 경로명으로 바꾸어 사용하는 방법이다.

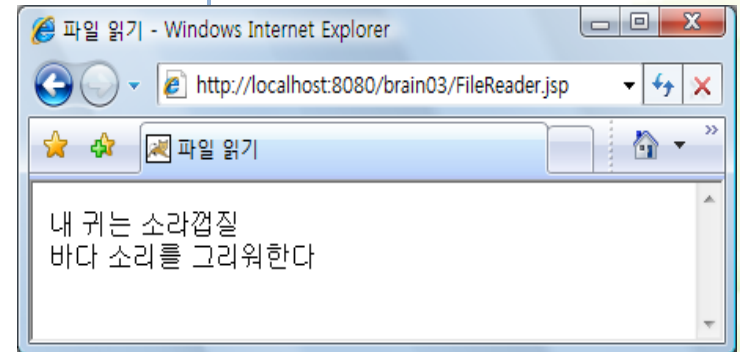
```
String filePath = application.getRealPath( “/WEB-INF/input.txt ”);
```



웹 애플리케이션 디렉터리 내에서의 파일의 경로명

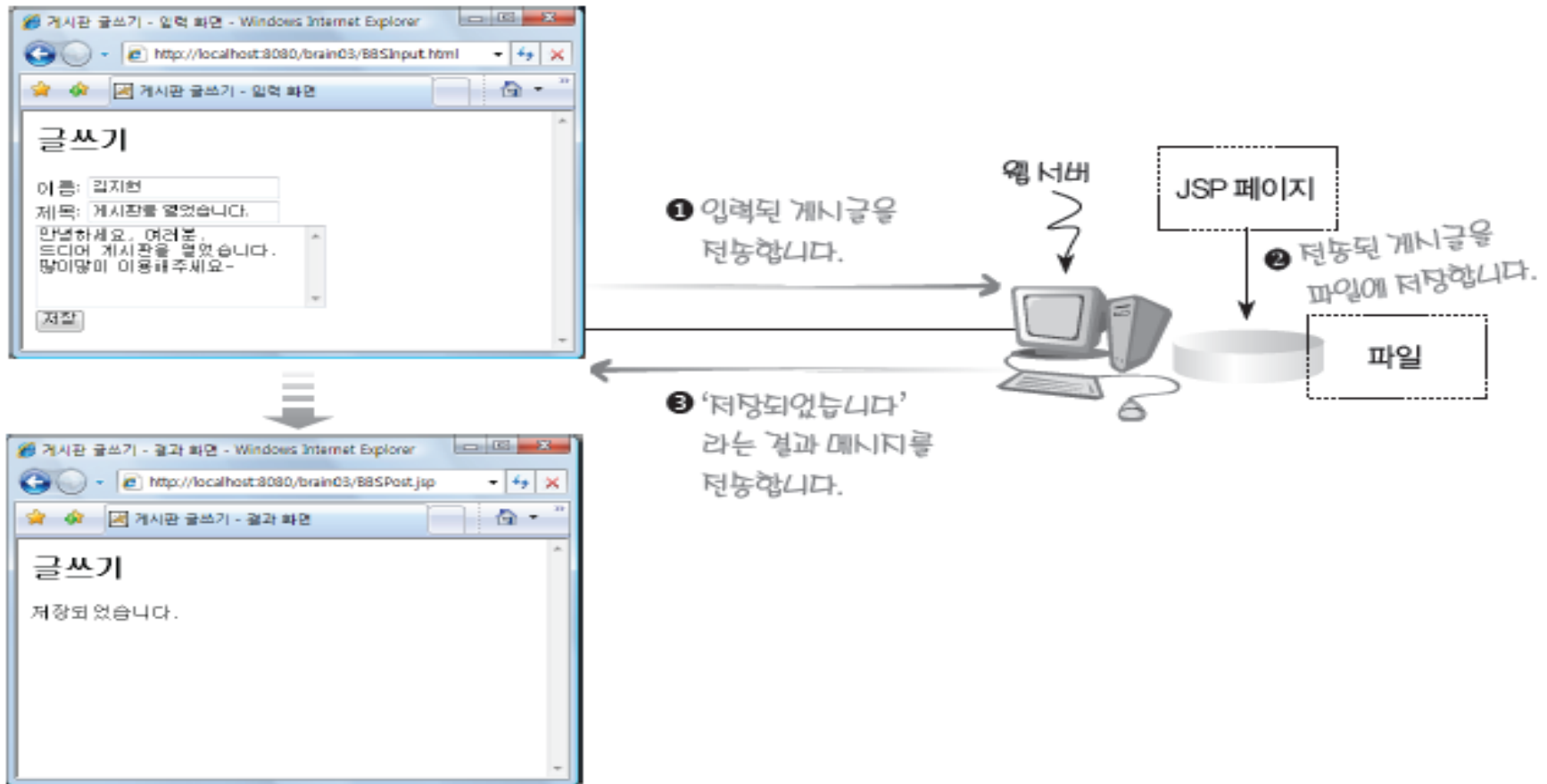
텍스트 파일의 내용을 읽어서 웹 브라우저로 출력하는 JSP 페이지

```
<%@page contentType= "text/html; charset=euc-kr " %>
<%@page import= "java.io.* " %>
<HTML>
  <HEAD><TITLE> 파일 읽기</TITLE></HEAD>
  <BODY>
    <%
      BufferedReader reader = null;
      try {
        String filePath = application.getRealPath( "/WEB-INF/input.txt ");
        reader = new BufferedReader(new FileReader(filePath));
        while (true) {
          String str = reader.readLine();
          if (str == null)
            break;
          out.println(str + "<BR> ");
        }
      }
      catch (FileNotFoundException fnfe) {
        out.println( "파일이 존재하지 않습니다." );
      }
      catch (IOException ioe) {
        out.println( "파일을 읽을 수 없습니다." );
      }
      finally {
        try {
          reader.close();
        }
        catch (Exception e) {
        }
      }
    %>
  </BODY>
</HTML>
```



❖ 파일에 데이터를 쓰는 방법

- JSP 페이지에서도 스탠드얼론 프로그램에서와 마찬가지로 `java.io.PrintWriter`, `java.io.PrintWriter`, `java.io.FileOutputStream` 등의 클래스를 이용하면 된다.



게시판 글쓰기 기능을 구현하는 웹 애플리케이션의 구성도

- 앞 페이지 두 화면의 URL을 정하고, URL에 해당하는 HTML 문서를 작성한다.

http://localhost:8181/BBSInput.html



위쪽 화면 URL

http://localhost:8181/BBSPost.jsp



아래쪽 화면 URL

게시판 글쓰기 기능의 입력 화면을 구현하는 HTML 문서

```
<HTML>
  <HEAD>
    <META http-equiv= "Content-Type" content= "text/html; charset=euc-kr" >
    <TITLE>게시판 글쓰기 - 입력 화면</TITLE>
  </HEAD>
  <BODY>
    <H2>글쓰기</H2>
    <FORM ACTION=/ch03_web/BBSPost.jsp METHOD=POST>
      이름: <INPUT TYPE=TEXT NAME=NAME><BR>
      제목: <INPUT TYPE=TEXT NAME=TITLE><BR>
      <TEXTAREA COLS=30 ROWS=5 NAME=CONTENT></TEXTAREA><BR>
      <INPUT TYPE=SUBMIT VALUE= '저장' >
    </FORM>
  </BODY>
</HTML>
```

- 밀리세컨드 단위의 현재 시각을 파일 이름으로 사용하기로 한다. 이 값은 `java.util.Date` 클래스의 객체를 만든 다음 `getTime` 메서드를 호출해서 얻을 수 있다.

```
Date date = new Date();
```

```
long time = date.getTime();
```

현재 시각을 밀리세컨드 단위로 가져오는
메서드

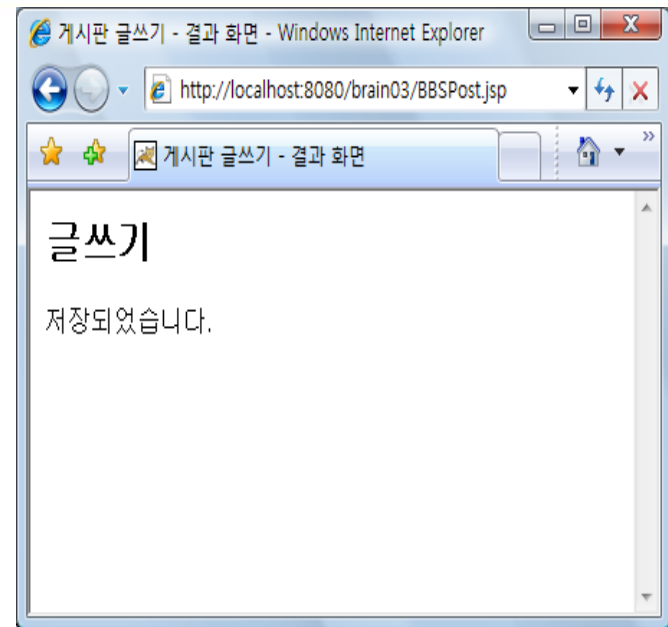
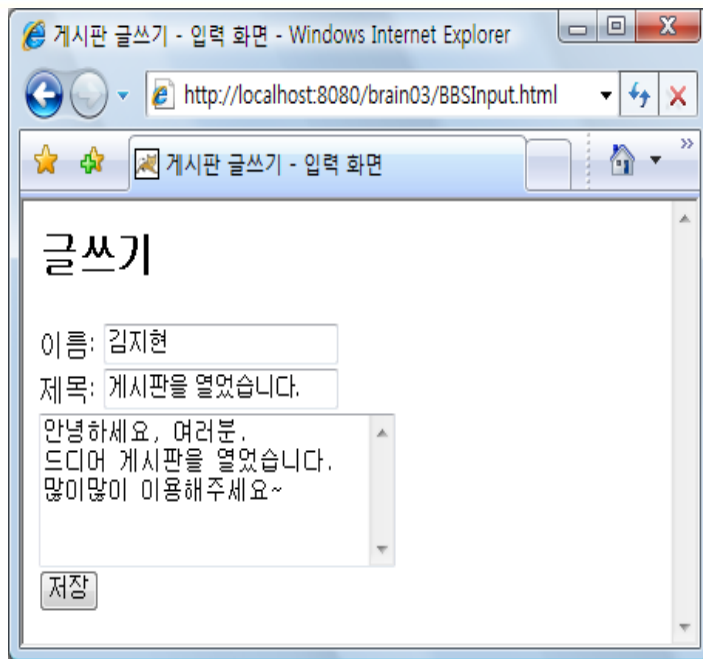
- `getTime` 메서드가 리턴하는 값은 1212414054907처럼 아주 긴 정수이다.

게시글을 저장하는 JSP 페이지

```
<% @page contentType= "text/html; charset=euc-kr" %>
<%@page import= "java.io.*, java.util.Date" %>
<HTML>
    <HEAD><TITLE>게시판 글쓰기 - 결과
화면</TITLE></HEAD>
    <BODY>
        <H2>글쓰기</H2>
        <% request.setCharacterEncoding( "euc-kr" );
            String name = request.getParameter( "NAME" );
            String title = request.getParameter( "TITLE" );
            String content =
                request.getParameter( "CONTENT" );
            Date date = new Date();
            Long time = date.getTime();
            String filename = time + ".txt";
            String filePath =
application.getRealPath("/WEB-INF/bbs/" + filename);
            FileWriter writer = new FileWriter(filePath);
            try {
                String str = "제목: " + title + "\n";
                str += "글쓴이: " + name + "\n";
                str += content;
                writer.write(str);
                out.println("저장되었습니다.");
            }
            catch (IOException ioe) {
```

게시글을 저장하는 JSP 페이지

```
                out.println( "파일에 데이터를 쓸 수
없습니다." );
            }
            finally {
                try {
                    writer.close();
                }
                catch (Exception e) {
                }
            }
        %>
    </BODY>
</HTML>
```



실행 방법

- 이 애플리케이션의 문제점 : 결과 화면에서 새로 고침 버튼을 누를 때마다 게시글 디렉터리에 똑같은 내용의 게시글 파일이 하나씩 더 생긴다.
 - 웹 브라우저의 새로 고침 버튼을 누를 때 마다 [예제 3-16]의 JSP 페이지가 다시 호출되기 때문이다.

다른 JSP 페이지 호출하기

❖ forward 메서드의 사용 방법

- forward 메서드는 JSP 페이지 안에서 다른 JSP 페이지를 호출할 때 사용하는 메서드이다.
- 이 메서드는 `javax.servlet.RequestDispatcher` 인터페이스에 속하기 때문에 이 타입의 객체가 있어야 호출할 수 있다.

```
RequestDispatcher dispatcher = request.getRequestDispatcher( "Result.jsp ");
```

↑
호출할 JSP 페이지의 URL 경로명

- forward 메서드를 호출할 때에는 request 내장 변수와 response 내장 변수를 파라미터로 넘겨줘야 한다.

```
dispatcher.forward(request, response);
```

request 내장 변수

response 내장 변수

- forward 메서드를 통해 호출하는 JSP 페이지로 데이터를 넘겨주려면 이 메서드보다 먼저 request.setAttribute 메서드를 호출해서 request 내장 변수 안에 데이터를 저장해 놓아야 한다.

```
request.setAttribute( "HEIGHT ", new Integer(178));
```

↑
데이터 이름

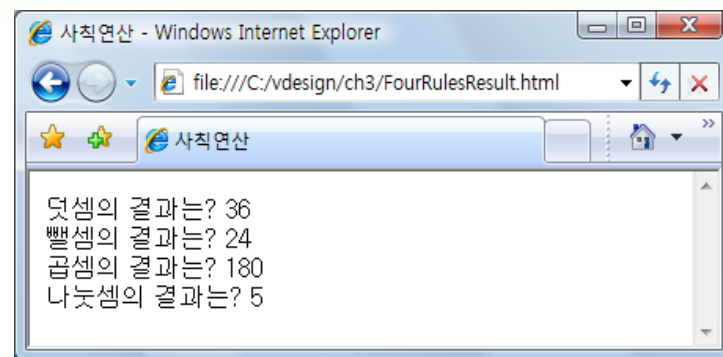
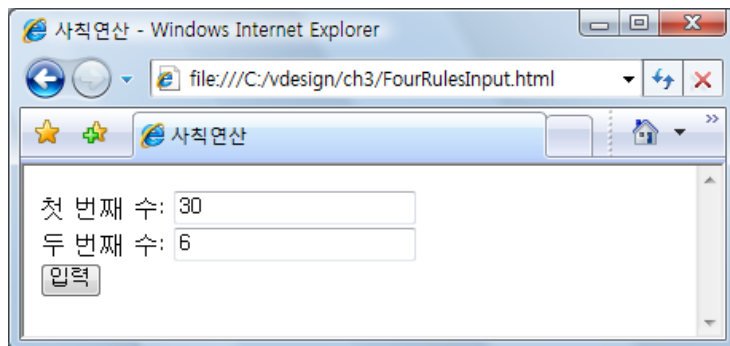
↑
데이터 값

- 호출된 JSP 페이지 안에서 request 내장 변수 안의 데이터를 가져오려면 request.getAttribute 메서드를 호출하면 된다.

```
Integer height = (Integer) request.getAttribute( "HEIGHT ");
```

↑
캐스트 연산자

↑
데이터 이름



사칙 연산을 수행하는 웹 애플리케이션 화면 설계

왼쪽 화면의 URL

<http://localhost:8181/ch03/FourRules.html>

오른쪽 화면의 URL

<http://localhost:8181/ch03/FourRules.jsp>

사칙 연산을 수행하는 JSP 페이지의 URL

<http://localhost:8181/ch03/FourRuelsResult.jsp>

사칙 연산의 결과를 출력하는 JSP 페이지의 URL

두 개의 수를 입력받는 HTML 문서

<HTML>

<HEAD>

<META http-equiv= "Content-Type " content= "text/html;charset=euc-kr ">

<TITLE>**사칙 연산**</TITLE>

</HEAD>

<BODY>

<FORM ACTION=FourRules.jsp>

첫 번째 수: <INPUT TYPE=TEXT NAME=NUM1>

두 번째 수: <INPUT TYPE=TEXT NAME=NUM2>

<INPUT TYPE=SUBMIT VALUE= '**입력**' >

</FORM>

</BODY>

</HTML>

```
<%  
    String str1 = request.getParameter("NUM1");  
    String str2 = request.getParameter("NUM2");  
    int num1 = Integer.parseInt(str1);  
    int num2 = Integer.parseInt(str2);  
    request.setAttribute("SUM", new Integer(num1 + num2));  
    request.setAttribute("DIFFERENCE", new Integer(num1 - num2));  
    request.setAttribute("PRODUCT", new Integer(num1 * num2));  
    request.setAttribute("QUOTIENT", new Integer(num1 / num2));  
    RequestDispatcher dispatcher = request.getRequestDispatcher("FourRulesResult.jsp");  
    dispatcher.forward(request, response);  
%>
```

```
<%@page contentType="text/html; charset=euc-kr"%>  
<HTML>  
    <HEAD> <TITLE> 사칙연산 </TITLE> </HEAD>  
    <BODY>  
        입력된 수: <%= request.getParameter("NUM1") %>, <%=  
request.getParameter("NUM2") %> <BR> <BR>  
        덧셈의 결과는? <%= request.getAttribute("SUM") %> <BR>  
        뺄셈의 결과는? <%= request.getAttribute("DIFFERENCE") %> <BR>  
        곱셈의 결과는? <%= request.getAttribute("PRODUCT") %> <BR>  
        나눗셈의 결과는? <%= request.getAttribute("QUOTIENT") %> <BR>  
    </BODY>  
</HTML>
```

❖ include 메서드의 사용 방법

- include 메서드도 forward 메서드처럼 다른 JSP 페이지를 호출하지만, 호출된 JSP 페이지가 끝나고 나면 실행 흐름의 제어가 본래의 JSP 페이지로 돌아온다.
- include 메서드는 forward 메서드와 마찬가지로 javax.servlet.RequestDispatcher 인터페이스에 속하므로 먼저 RequestDispatcher 객체를 구해야 한다.

```
RequestDispatcher dispatcher = request.getRequestDispatcher( "Today.jsp " );
```

호출할 JSP 페이지의 URL 경로명

- include 메서드를 호출할 때는 request 내장 변수와 response 내장 변수를 파라미터로 넘겨줘야 한다.

```
dispatcher.include(request, response);
```

request 내장 변수

response 내장 변수

- include 메서드를 통해 호출되는 JSP 페이지로 데이터를 넘겨주기 위해서는 forward 메서드의 경우와 마찬가지로 request 내장 변수에 대해 setAttribute 메서드와 getAttribute 메서드를 호출하면 된다.

```
request.setAttribute( "WEIGHT ", new Double(72.5));
```

request 내장 변수에
데이터를 저장하는 메서드

```
Double weight = (Double) request.getAttribute( "WEIGHT ");
```

request 내장 변수에 저장되
어 있는 데이터를 가져오는 메
서드

include 메서드의 사용 예를 보여주는 JSP 페이지

include 메서드의 사용 예

```
<% @page contentType= "text/html; charset=euc-kr" %>
<HTML>
  <HEAD><TITLE>오늘의 메뉴</TITLE></HEAD>
  <BODY>
    <H3>오늘의 메뉴</H3>
    - 짜장면 <BR>
    - 볶음밥 <BR>
    - 짬뽕 <BR><BR>
    <%
      out.flush();
      RequestDispatcher dispatcher =
request.getRequestDispatcher( "Now.jsp ");
      dispatcher.include(request, response);
    %>
  </BODY>
</HTML>
```

Now.jsp를 include합니다

```
<% @page contentType="text/html; charset=euc-kr"%>
<%@page import="java.util.*"%>
<% GregorianCalendar now = new GregorianCalendar(); %>
[현재의 시각] <%= String.format("%TF %TT", now,
now) %>
```

