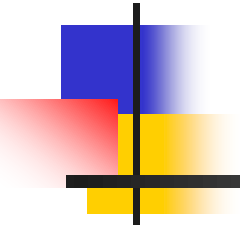


웹 프로그래밍

JSP





JSP 기초

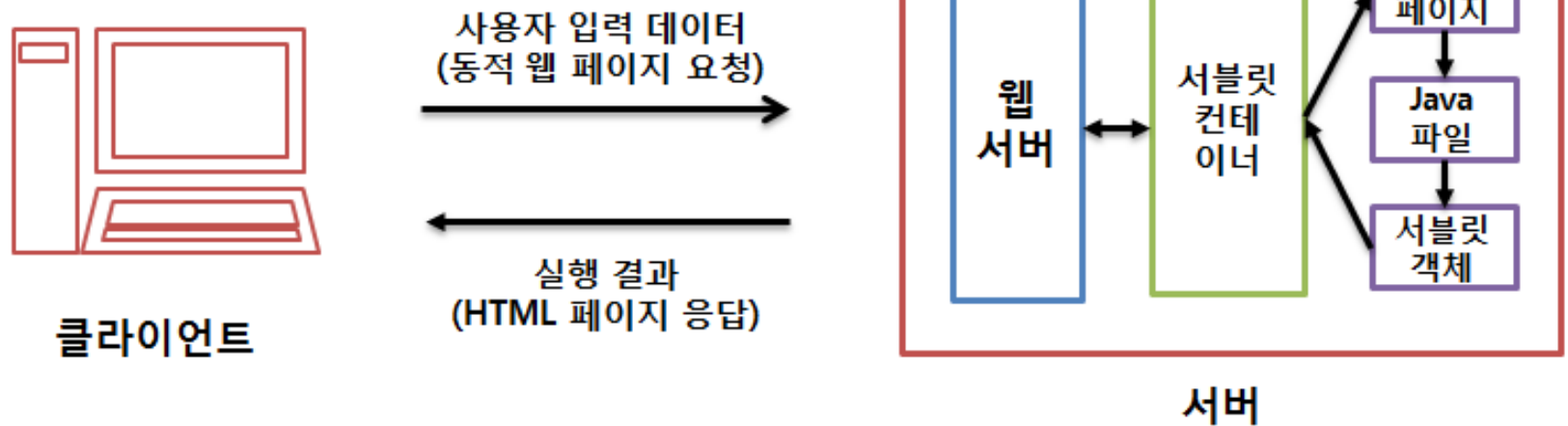
- JSP 개요
- JSP 처리 방식



1. JSP 개요

- 자바를 기반으로 하는 동적 웹 사이트 구축 언어
- 서버 사이드 스크립트(server side script) 언어
- JSP 특징
 - 플랫폼에 독립적
 - 서버 자원의 효율적 관리
 - 컴포넌트 기반 개발
 - 비즈니스 로직과 프리젠테이션 로직의 분리

2. JSP 처리 방식





2.1 JSP 기본 요소

- 지시문
- 스크립팅 요소
- 액션 태그
- 내장 객체
- 간단한 예제 작성



2.1 지시문

- 페이지의 처리와 관련된 정보 기술

형 식 : <%@ 지시문_이름 속성_정의_리스트 %>

속성_정의 : 속성 = "속성_값"



2.1 지시문

- page 지시문

사용 예 : `<%@ page contentType="text/html; charset=utf-8" %>`



2.1 지시문

[표 6.1] page 지시문의 속성 및 기본값

속성	설명
contentType	MIME 타입과 문자 셋
import	임포트할 패키지
pageEncoding	서블릿으로 변환할 때의 문자 인코딩 방식
session	HTTP 세션 사용 여부
buffer	클라이언트로 전송되는 out 객체의 버퍼 크기
autoFlush	버퍼가 찼을 때 클라이언트로 자동 출력 여부
extends	현재 페이지가 상속받을 수퍼 클래스
errorpage	에러를 출력할 페이지의 url
isErrorpage	현재 페이지를 에러 페이지로 사용 여부
isELIgnored	식 언어(expression language) 사용 여부
isThreadSafe	다중 스레드 사용 여부
info	JSP 페이지에 대한 정보
language	JSP 페이지에서 사용하는 언어



2.1 지시문

- include 지시문

형 식 : <%@ include file="파일의 url" %>



2.1 지시문

- 예제 : include-directive.jsp

```
4 : <body>
5 :     include 지시문 사용하기
6 :     <br><hr>
7 :     <%@ include file="include-test.jsp" %>
8 : </body>
```

- 예제 : include-test.jsp

```
4 : <body>
5 :     included text
6 : </body>
```

2.1 지시문

- 실행 결과





2.2 스크립팅 요소

■ 선언문

형 식 (변 수) : <%!
자료형 변수_리스트
%>

사용 예 : <%!
var a;
var b, c;
var d = 1;
%>



2.2 스크립팅 요소

■ 스크립트릿

형 식 : <%
 자바_코드
 %>

사용 예 : <%
 int x = 2;
 int y = 3;
 int z;

 z = add(x, y);
 %>



2.2 스크립팅 요소

■ 식

형 식 : $\langle \% = \text{자바_식} \% \rangle$

사용 예 (변수) : $\langle \% = x \% \rangle + \langle \% = y \% \rangle = \langle \% = z \% \rangle$

사용 예 (수식) : $\langle \% = x \% \rangle + \langle \% = y \% \rangle = \langle \% = x+y \% \rangle$



2.3 액션 태그

- 어떠한 액션이 발생하는 시점에 실행
- XML을 기반으로 하기 때문에 시작 태그가 있으면 반드시 종료 태그가 있어야 함
- 태그는 접두어 'jsp:'로 시작하여야 함



2.3 액션 태그

- <jsp:include> 태그

사용 예 : <jsp:include page="a.jsp" flush="false"/>



2.3 액션 태그

- `<jsp:forward>` 태그

사용 예 : `<jsp:forward page="a.jsp"/>`



2.3 액션 태그

- <jsp:param> 태그

사용 예 : <jsp:forward page="a.jsp">
 <jsp:param name="param_name" value="value1"/>
 </jsp:forward>



2.3 액션 태그

- `<jsp:plug-in>` 태그

사용 예 : `<jsp:plug-in type="applet" code="applet_exe.class" codebase="/class_code"/>`



2.3 액션 태그

- `<jsp:useBean>` 태그

사용 예 : `<jsp:useBean id="useBean" class="Bean.class" scope="page"/>`

- `<jsp:setProperty>` 태그

사용 예 : `<jsp:useBean id="bean">
 <jsp:setProperty name="bean" property="title"
 value="web programming"/>
</jsp:useBean>`



2.3 액션 태그

■ <jsp:getProperty> 태그

사용 예 :

```
<jsp:useBean id="bean">  
    <jsp:setProperty name="bean" property="title"  
        value="web programming"/>  
    책 이름은 <jsp:getProperty name="bean" property="title"/>  
    입니다.  
</jsp:useBean>
```



2.4 내장 객체

- 정의할 필요없이 자동으로 제공되는 객체
- `<% ... %>` 또는 `<%= ... %>` 내에서 사용



2.4 내장 객체

[표 6.4] JSP 내장 객체 및 기능

내장 객체	설명
request	클라이언트의 요청 처리
response	클라이언트의 요청에 대한 응답
out	출력 스트림 처리
session	클라이언트의 세션 정보 처리
application	애플리케이션 정보 처리
pageContext	JSP 페이지의 내용(context)
config	JSP 페이지의 초기화(initialization) 매개변수 저장
page	현재의 JSP 페이지
exception	예외 처리



2.5 간단한 예제 작성

- 예제 : welcome.jsp

```
6 :      <%  
7 :          out.println("Welcome to JSP !");  
8 :      %>  
9 :      <br> <hr>  
10 :     <%  
11 :         String text = "Welcome to JSP !";  
12 :     %>  
13 :     <%= text%>
```


2.5 간단한 예제 작성

- 실행 결과





3. JSP 기본 문법

- 기본 태그
- 자료형 및 변수
- 연산자
- 문장
- 클래스
- 함수
- 예외 처리



3.1 기본 태그

- 식 태그

형 식 : <%= 자바_식 %>



3.1 기본 태그

- 스크립트릿 태그

형 식 : <% 자바_코드 %>



3.1 기본 태그

- 예제 : scriptlet-tag.jsp

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%
4
5     int a = 3;
6     int b = 5;
7   %>
8
9   3 + 5 = <%= a + b %>
10
```

3.1 기본 태그

- 실행 결과





3.1 기본 태그

■ 주석

[표 6.6] JSP 주석의 종류

형식	설명
<!-- 주석 -->	HTML 주석
<%-- 주석 --%>	원본 소스에서만 주석이 보이고 클라이언트의 소스 보기에서는 보이지 않으므로, 보안이 필요한 경우에 활용
// 주석	한 줄 이내의 주석
/* 주석 */	여러 줄의 주석



3.1 기본 태그

■ 예제 : comment.jsp

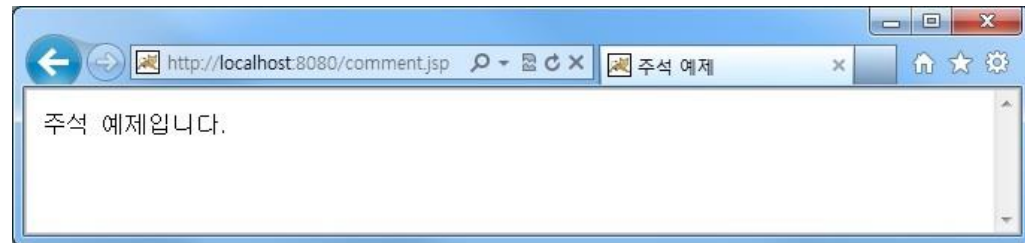
```
5 : <!-- HTML 주석입니다.-->
6 : <%-- 소스 보기에서는 주석이 안 보입니다.
7 :      이것은 블록 단위 주석입니다.
8 :      --%>
9 : <%
10 :      out.println("주석 예제입니다.");
11 : %>
```


3.1 기본 태그

■ 실행 결과



```
1
2 <html>
3     <head>
4         <title> 주석 예제 </title>
5     </head>
6     <body>
7         <!-- HTML 주석입니다.-->
8
9         주석 예제입니다.
10
11     </body>
12 </html>
```





3.2 자료형 및 변수

■ 변수

[표 6.7] JSP의 기본 자료형

자료형	설명
boolean	true 또는 false(1 bit)
char	자바의 유니코드 문자(2 bytes)
byte	-128 ~ 127 범위의 정수(1 byte)
short	-32,768 ~ 32,767 범위의 정수(2 bytes)
int	-2,147,483,648 ~ 2,147,483,647 범위의 정수(4 bytes)
long	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807 범위의 정수(8 bytes)
float	부동소수점 실수(4 bytes)
double	부동소수점 실수(8 bytes)



3.2 자료형 및 변수

- 변수(계속)

- 변수 선언

형 식 :

자료형 변수_선언_리스트;

변수_선언 : 변수_이름[= 값]

- 선언 예

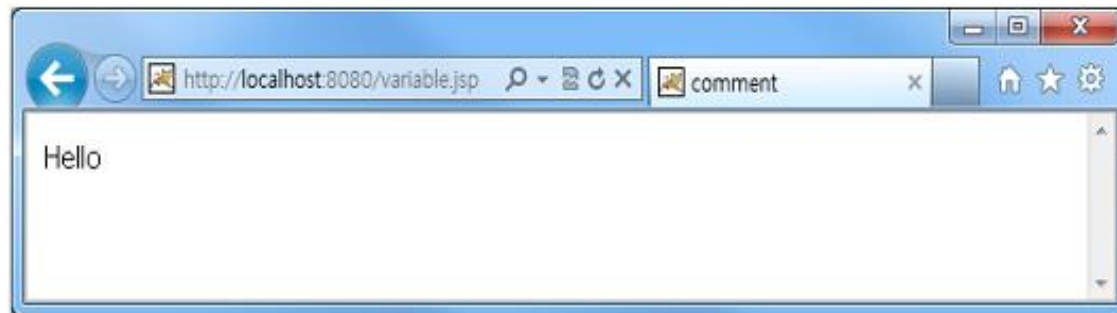
사용 예 : int a;
float b, c;
int d = 1;

3.2 자료형 및 변수

■ 예제 : variable.jsp

```
1 : <%  
2 :     String text;  
3 :     text = "Hello";  
4 : %>  
5 : <%= text %>
```

■ 실행 결과





3.2 자료형 및 변수

- 배열

- 하나의 변수에 같은 자료형의 하나 이상의 값을 저장하고자 할 때 사용



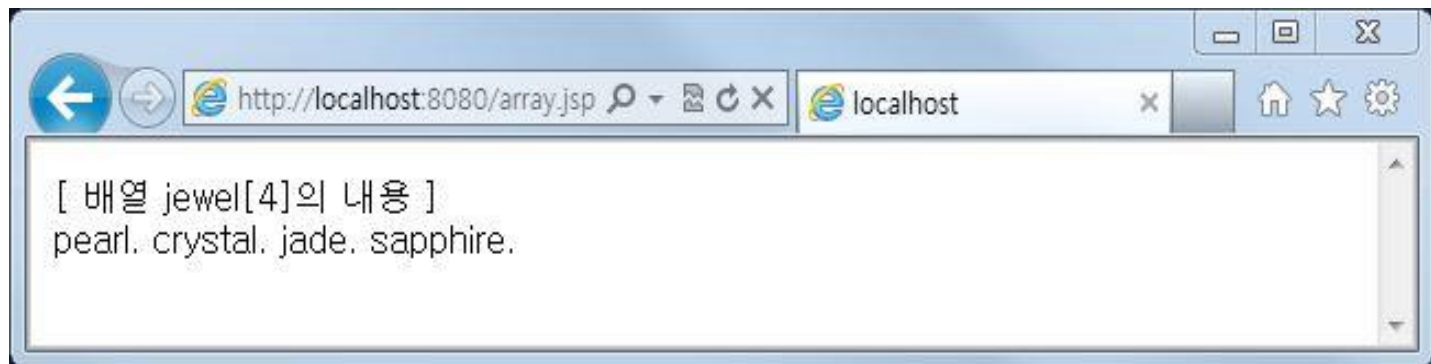
3.2 자료형 및 변수

■ 예제 : array.jsp

```
2 : <%  
3 :     String[] jewel = new String[4];  
4 :  
5 :     jewel[0] = "pearl";  
6 :     jewel[1] = "crystal";  
7 :     jewel[2] = "jade";  
8 :     jewel[3] = "sapphire";  
9 : %>  
10 : [ 배열 jewel[4]의 내용 ]<br>  
11 : <%= jewel[0]%>. <%= jewel[1]%>. <%= jewel[2]%>.  
    <%= jewel[3]%>.
```

3.2 자료형 및 변수

- 실행 결과





3.3 연산자

- 산술 연산자
 - 산술 연산

사용 예 :

- $a + b$
- $a - b$
- $a * b$
- a / b
- $a \% b$



3.3 연산자

- 증감 연산자

- ++, -- 기호를 변수 앞이나 뒤에 붙임
- 변수 값을 1씩 증가 또는 감소

사용 예 : ++a
 a++
 --a
 a--



3.3 연산자

- 관계 연산자
 - 값의 대소 관계를 판단
 - 참(true) 또는 거짓(false)을 구함

사용 예 :

- $a > b$
- $a \geq b$
- $a < b$
- $a \leq b$
- $a == b$
- $a != b$



3.3 연산자

- 논리 연산자
 - 참(true) 또는 거짓(false)을 구함

사용 예 :

```
a && b
a || b
!a

a > b && c == d
a > 7 || b <= 100
```



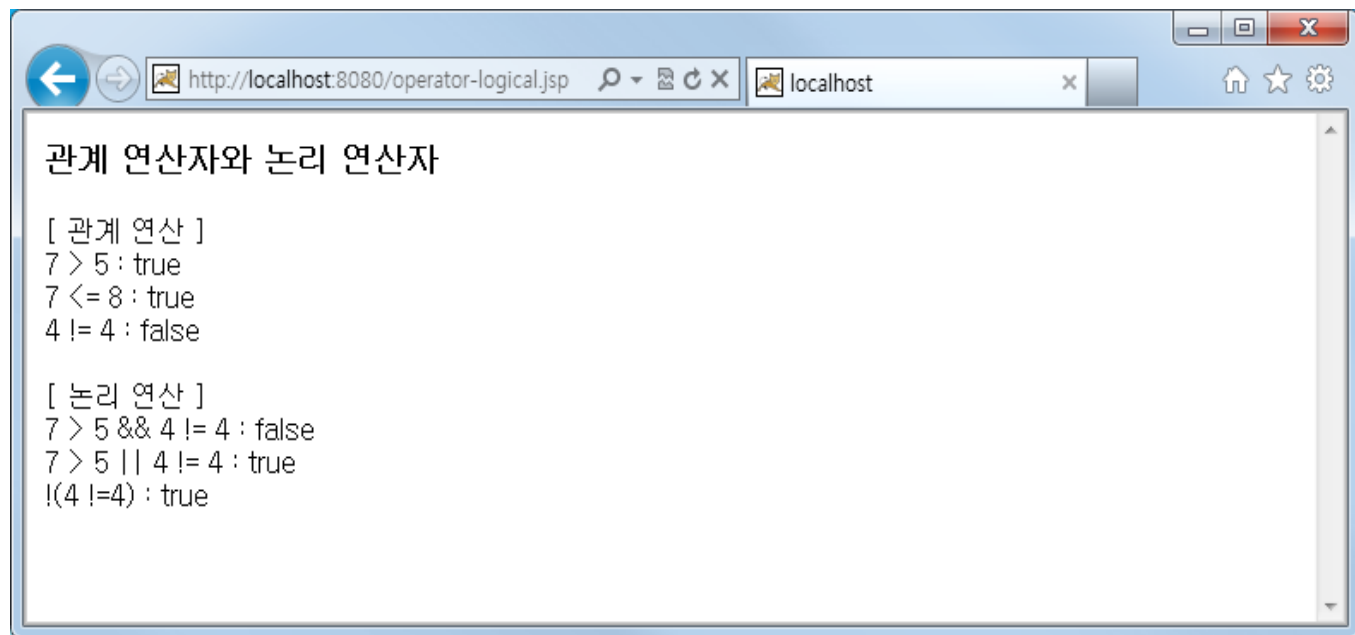
3.3 연산자

■ 예제 : operator-logical.jsp

```
2 : <h3> 관계 연산자와 논리 연산자 </h3>
3 :
4 : [ 관계 연산 ] <br>
5 : 7 > 5 : <%= 7 > 5 %> <br>
6 : 7 <= 8 : <%= 7 <= 8 %> <br>
7 : 4 != 4 : <%= 4 != 4 %> <br>
8 :
9 : <br> [ 논리 연산 ] <br>
10 : 7 > 5 && 4 != 4 : <%= 7 > 5 && 4 != 4 %> <br>
11 : 7 > 5 || 4 != 4 : <%= 7 > 5 || 4 != 4 %> <br>
12 : !(4 !=4) : <%= !(4 != 4) %>
```

3.3 연산자

■ 실행 결과



A screenshot of a web browser window. The address bar shows the URL `http://localhost:8080/operator-logical.jsp`. The page content is as follows:

```
관계 연산자와 논리 연산자

[ 관계 연산 ]
7 > 5 : true
7 <= 8 : true
4 != 4 : false

[ 논리 연산 ]
7 > 5 && 4 != 4 : false
7 > 5 || 4 != 4 : true
!(4 != 4) : true
```



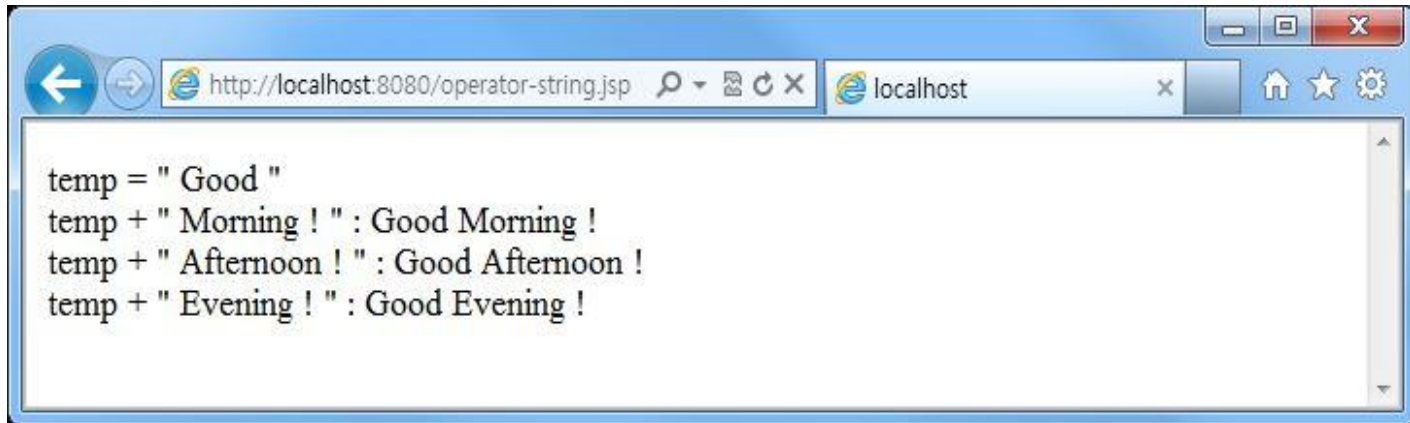
3.3 연산자

- 문자열 연산자
 - + 연산자 사용
- 예제 : operator-string.jsp

```
1 : <%  
2 :     String temp;  
3 :     temp = " Good ";  
4 : %>  
5 : temp = " Good " <br>  
6 : temp + " Morning ! " : <%= temp + " Morning ! " %> <br>  
7 : temp + " Afternoon ! " : <%= temp + " Afternoon ! " %> <br>  
8 : temp + " Evening ! " : <%= temp + " Evening ! " %>
```

3.3 연산자

■ 실행 결과



A screenshot of a web browser window. The address bar shows the URL `http://localhost:8080/operator-string.jsp`. The browser window displays the following text:

```
temp = " Good "  
temp + " Morning ! " : Good Morning !  
temp + " Afternoon ! " : Good Afternoon !  
temp + " Evening ! " : Good Evening !
```



3.3 연산자

- 배정 연산자
 - 변수에 값 저장
 - 할당 연산자 또는 대입 연산자

사용 예 :

```
a = 1  
a = b  
a += b  
a -= b  
a *= b  
a /= b  
a %= b
```



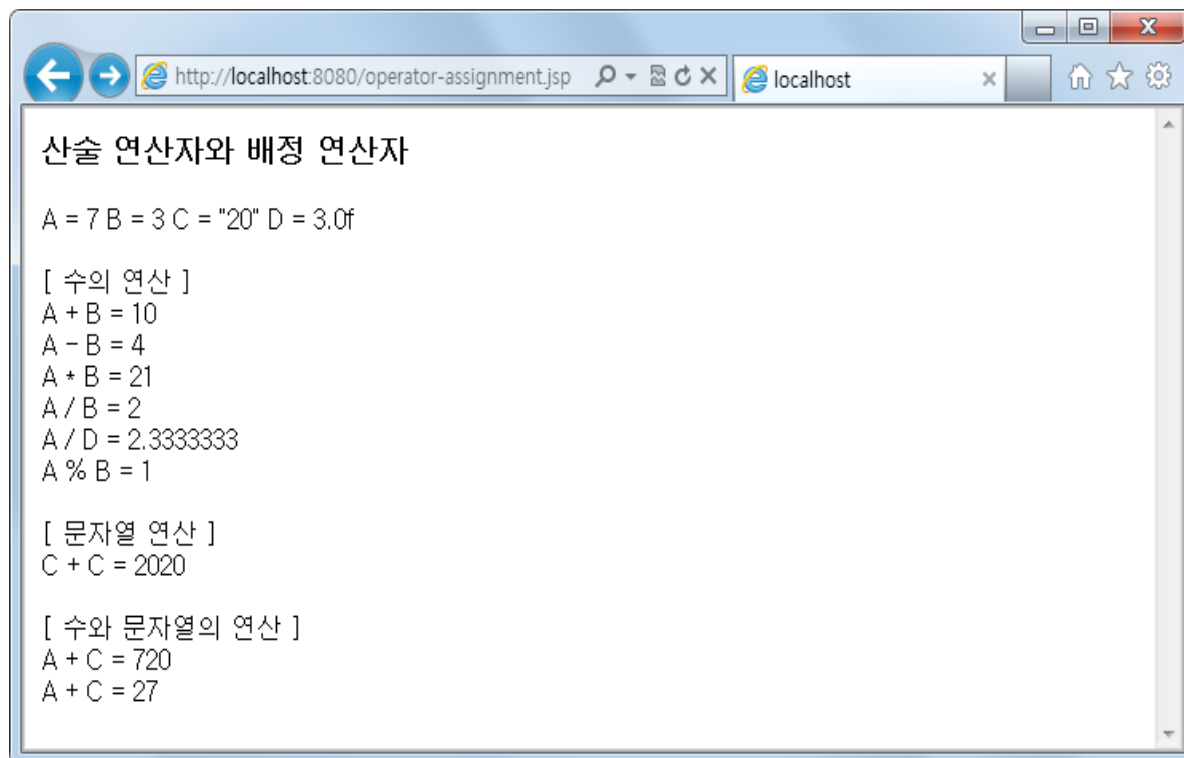

3.3 연산자

■ 예제 : operator-assignment.jsp

```
2: <h3> 산술 연산자와 배정 연산자 </h3>
3: <%
4:     int a = 7;
5:     int b = 3;
6:     String c = "20";
7:     float d = 3.0f;
8: %>
9:   A = 7   B = 3   C = "20"   D = 3.0f <br><br>
10:  [ 수의 연산 ] <br>
11:   A + B = <%= a + b %> <br>
12:   A - B = <%= a - b %> <br>
13:   A * B = <%= a * b %> <br>
14:   A / B = <%= a / b %> <br>
15:   A / D = <%= a / d %> <br>
16:   A % B = <%= a % b %> <br><br>
17:  [ 문자열 연산 ] <br>
18:   C + C = <%= c + c %> <br><br>
19:  [ 수와 문자열의 연산 ] <br>
20:   A + C = <%= a + c %> <br>
21:   A + C = <%= a + Integer.parseInt(c) %>
```

3.3 연산자

■ 실행 결과



The screenshot shows a web browser window with the address bar displaying `http://localhost:8080/operator-assignment.jsp`. The page content is as follows:

```
산술 연산자와 배정 연산자  
  
A = 7 B = 3 C = "20" D = 3.0f  
  
[ 수의 연산 ]  
A + B = 10  
A - B = 4  
A * B = 21  
A / B = 2  
A / D = 2.3333333  
A % B = 1  
  
[ 문자열 연산 ]  
C + C = 2020  
  
[ 수와 문자열의 연산 ]  
A + C = 720  
A + C = 27
```



3.4 문장

- 조건문
- 반복문



3.4.1 조건문

■ if 문

사용 예 :

(a) if 문

```
if(조건식1)
{
    문장_리스트1
}
```

(b) if-else 문

```
if(조건식1)
{
    문장_리스트1
}
else
{
    문장_리스트2
}
```

(c) if-else if-else 문

```
if(조건식1)
{
    문장_리스트1
}
else if(조건식2)
{
    문장_리스트2
}
else
{
    문장_리스트3
}
```



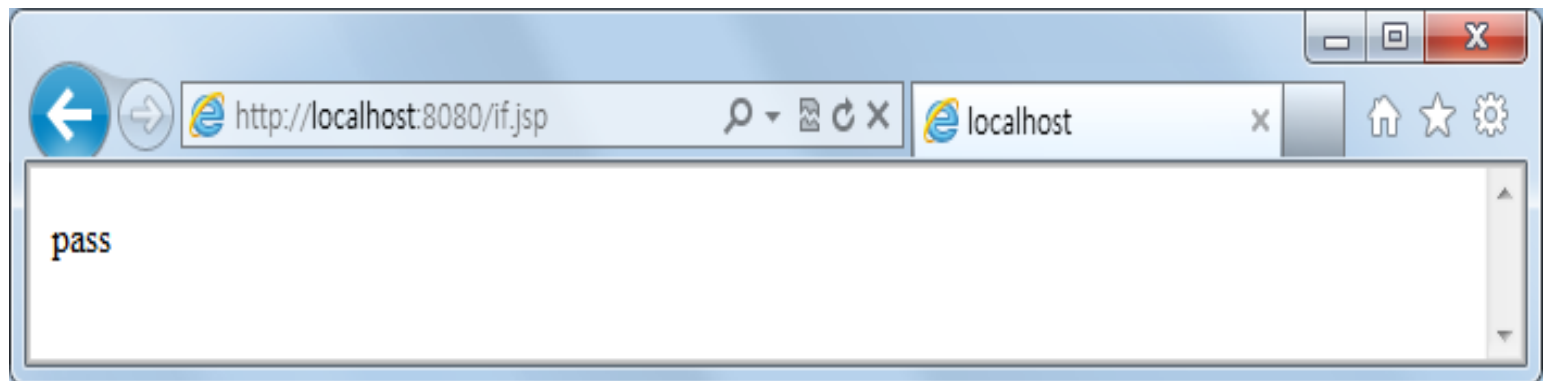
3.4.1 조건문

■ 예제 : if.jsp

```
1: <%  
2:     int score;  
3:     score = 95;  
4:     String temp;  
5:  
6:     if(score >= 90)  
7:         temp = "pass";  
8:     else  
9:         temp = "fail";  
10: %>  
11: <%= temp %>
```

3.4.1 조건문

- 실행 결과





3.4.1 조건문

■ switch 문

형 식 :

```
switch (식)
{
    case 값1 : 문장_리스트1
                break;
    case 값2 : 문장_리스트2
                break;
    ...
    case 값n : 문장_리스트n
                break;
    [ default : 문장_리스트n+1 ]
}
```



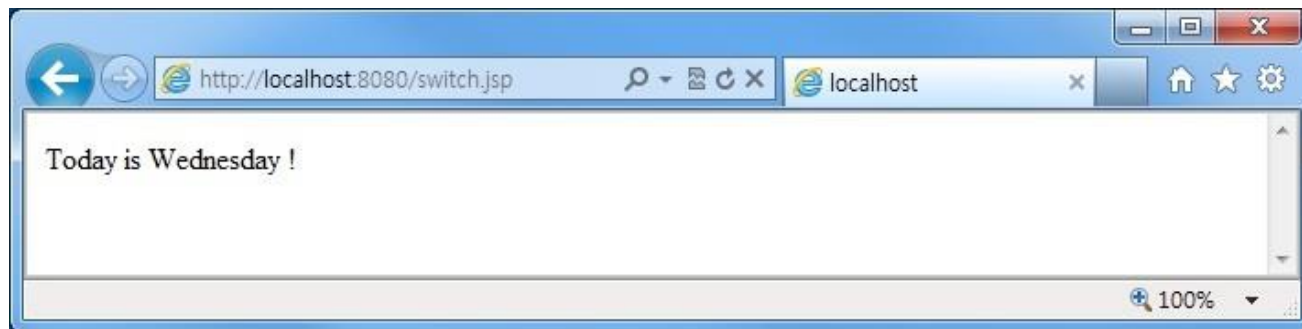
3.4.1 조건문

■ 예제 : switch.jsp

```
1:  <%
2:      int today = 3;
3:      String day;
4:
5:      switch ( today ){
6:          case 1 :
7:              day = "Monday"; break;
8:          case 2 :
9:              day = "Tuesday"; break;
10:         case 3 :
11:             day = "Wednesday"; break;
12:         case 4 :
13:             day = "Thursday"; break;
14:         case 5 :
15:             day = "Friday"; break;
16:         case 6 :
17:             day = "Saturday"; break;
18:         default :
19:             day = "Sunday";
20:     }
21:  %>
22:  Today is <%= day%> !
```


3.4.1 조건문

- 실행 결과





3.4.2 반복문

- for 문

형 식 :

```
for(초기화식; 조건식; 증감식) {  
    문장_리스트  
}
```



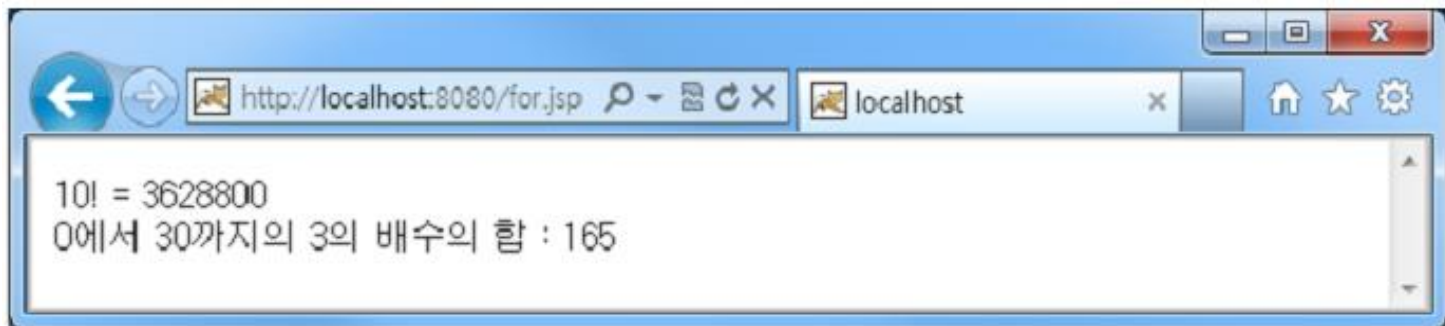
3.4.2 반복문

■ 예제 : for.jsp

```
2 : <%
3 :     int i, fac;
4 :     fac = 1;
5 :     for(i = 1; i <= 10; i++){
6 :         fac = fac * i;
7 :     }
8 : %>
9 : 10! = <%= fac %> <br>
10 : <%
11 :     int temp = 0;
12 :     for(i = 0; i <= 30; i = i + 3){
13 :         temp = temp + i;
14 :     }
15 : %>
16 : 0에서 30까지의 3의 배수의 합 : <%= temp %>
```

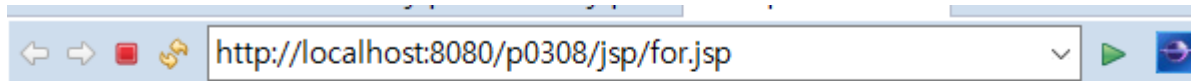
3.4.2 반복문

- 실행 결과



3.4.2 반복문

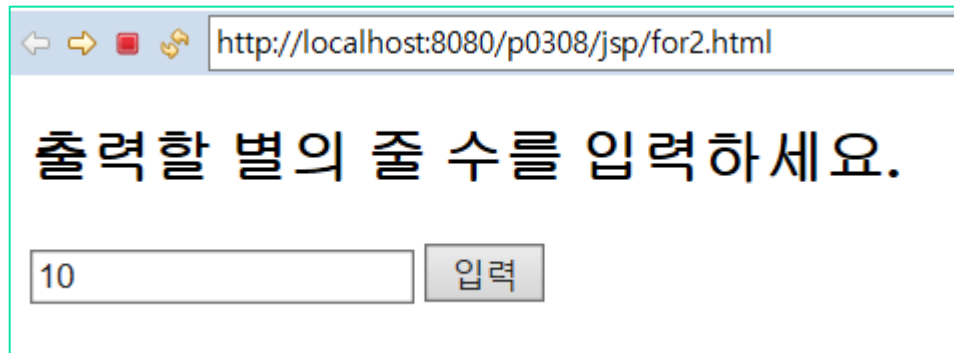
- [실습]for문을 사용하여 1~1000까지의 합을 구하는 jsp프로그램을 작성하시오.



0에서 1,000까지의 합은 500500 입니다. (for 문을 이용한 예제)

3.4.2 반복문

- [실습] 다중 for문을 사용하여 입력한 수만큼 별표를 출력하는 jsp프로그램을 작성하시오.



A screenshot of a web browser window. The address bar shows the URL `http://localhost:8080/p0308/jsp/for2.html`. The main content area displays the text "출력할 별의 줄 수를 입력하세요." (Enter the number of lines of asterisks to output). Below this text is a text input field containing the number "10" and a button labeled "입력" (Input).



A screenshot of a web browser window showing the output of the JSP program. The address bar shows the URL `http://localhost:8080/p0308`. The main content area displays a pattern of asterisks arranged in 10 rows, where each row contains a number of asterisks equal to the row number (1 to 10).

```
*  
* *  
* * *  
* * * *  
* * * * *  
* * * * * *  
* * * * * * *  
* * * * * * * *  
* * * * * * * * *  
* * * * * * * * * *  
* * * * * * * * * *
```



3.4.2 반복문

- while 문

형 식 :

```
while (조건식) {  
    문장_리스트  
    증감식;  
}
```



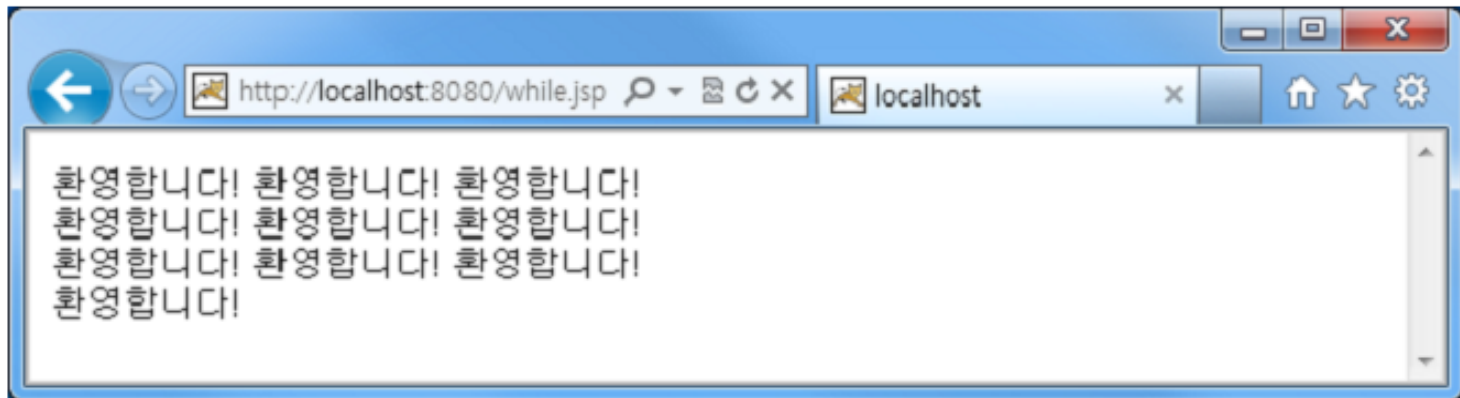
3.4.2 반복문

■ 예제 : while.jsp

```
2 : <%
3 :     int i = 0;
4 :     while( i < 10 ) {
5 :         i = i + 1;
6 :     %>
7 :
8 :         환영합니다!
9 :
10 : <%     if(( i % 3 ) == 0){ %>
11 :
12 :         <br>
13 :
14 : <%     }
15 :     } %>
```

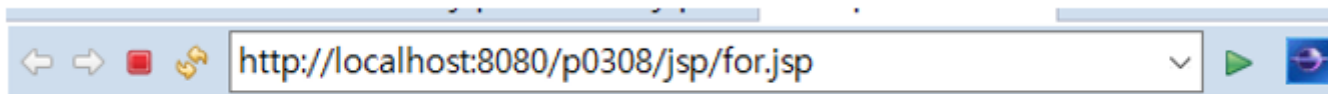

3.4.2 반복문

- 실행 결과



3.4.2 반복문

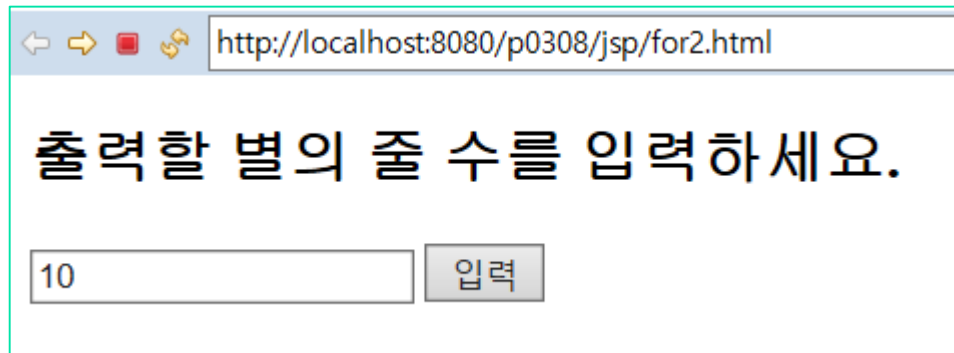
- [실습] while문을 사용하여 1~1000까지의 합을 구하는 jsp프로그램을 작성하시오.



0에서 1,000까지의 합은 500500 입니다.

3.4.2 반복문

- [실습] 다중 while문을 사용하여 입력한 수만큼 별표를 출력하는 jsp프로그램을 작성하시오.



A screenshot of a web browser window. The address bar shows the URL `http://localhost:8080/p0308/jsp/for2.html`. The main content area displays the text "출력할 별의 줄 수를 입력하세요." (Enter the number of lines of asterisks to output). Below this text is a text input field containing the number "10" and a button labeled "입력" (Input).



A screenshot of a web browser window showing the output of the program. The address bar shows the URL `http://localhost:8080/p0308`. The main content area displays a pattern of asterisks arranged in 10 lines, with the number of asterisks increasing by one in each successive line, forming a right-angled triangle shape.

```
*  
* *  
* * *  
* * * *  
* * * * *  
* * * * * *  
* * * * * * *  
* * * * * * * *  
* * * * * * * * *  
* * * * * * * * * *
```



3.4.2 반복문

- do while 문

형 식 :

```
do {   문장_리스트  
      증감식;  
} while (조건식);
```



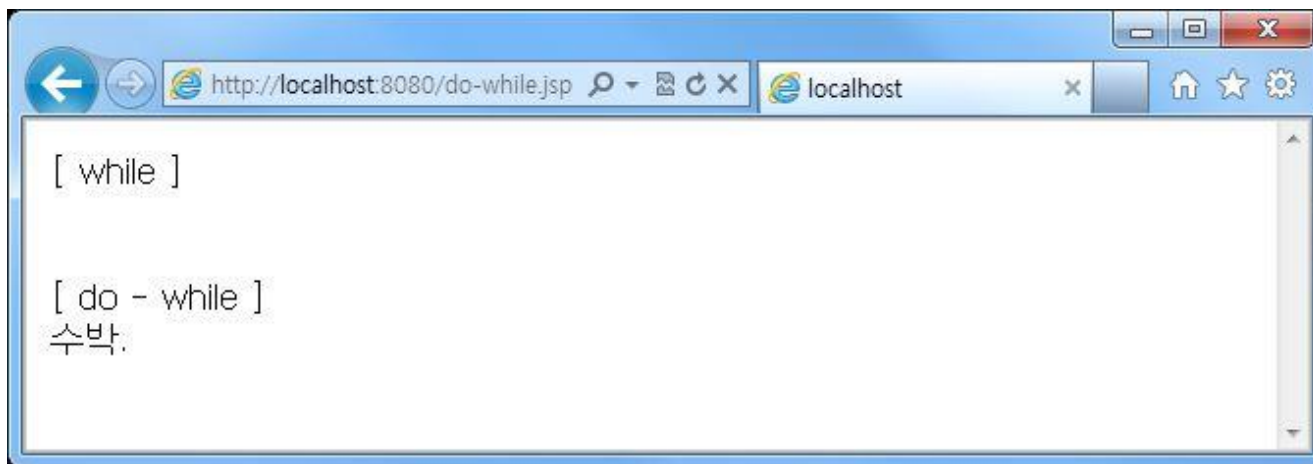
3.4.2 반복문

■ 예제 : do-while.jsp

```
2 : [ while ] <br><br>
3 : <%
4 :     int i;
5 :     i = 0;
6 :     while(i < 0) {
7 :         i = i + 1;
8 :         out.println("수박.");
9 :     }
10 : %>
11 :
12 : <br> [ do - while ] <br>
13 : <%
14 :     i = 0;
15 :     do {
16 :         i = i + 1;
17 :         out.println("수박.");
18 :     } while(i < 0);
19 : %>
```

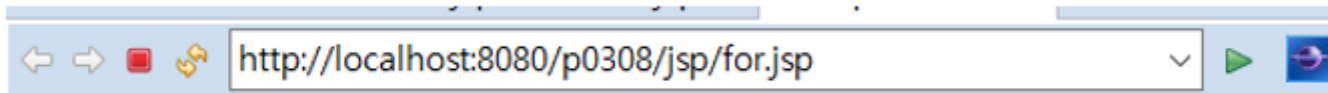
3.4.2 반복문

- 실행 결과



3.4.2 반복문

- [실습] do~while문을 사용하여 1~1000까지의 합을 구하는 jsp 프로그램을 작성하시오.



0에서 1,000까지의 합은 500500 입니다.



3.5 클래스

- 자바에서 정의된 클래스의 메소드들을 사용할 수 있음
- Integer 클래스, String 클래스, Date 클래스, Math 클래스, File 클래스의 주요 메소드들을 살펴봄



3.5 클래스

- Integer 클래스
 - 정수 처리

[표 6.8] Integer 클래스의 주요 메소드

메소드	설명
toString()	데이터를 문자열로 변환
parseInt("문자열")	문자열 데이터를 정수형으로 변환
valueOf("문자열")	문자열 데이터를 정수형 객체로 반환
equals(객체)	객체의 정수값이 같으면 true, 아니면 false 반환



3.5 클래스

- String 클래스
 - 문자열 다룸

[표 6.9] String 클래스의 주요 메소드

메소드	설명
toUpperCase()	알파벳 글자를 대문자로 변환
toLowerCase()	알파벳 글자를 소문자로 변환
equals(객체)	지정한 객체의 두 문자열이 같으면 true, 아니면 false 반환
concat("문자열")	문자열 연결
substring(인덱스1, 인덱스2)	인덱스1에서 인덱스2 전까지 문자열 반환, 인덱스2 생략되면 끝까지 문자열 반환
replace(기존 문자, 새로운 문자)	기존 문자를 새로운 문자로 대체
trim()	공백 제거



3.5 클래스

- Date 클래스
 - 날짜와 시각 처리

[표 6.10] Date 클래스의 주요 메소드

메소드	설명
getDate()	날짜
getYear()	연
getMonth()	월
getDay()	일
getHours()	시
getMinutes()	분
toLocaleString()	현재 지역의 날짜



3.5 클래스

- Math 클래스
 - 수학 계산

[표 6.11] Math 클래스의 주요 메소드

메소드	설명
abs(n)	n의 절대값
sqrt(n)	n의 제곱근
sin(n)	n의 사인값
cos(n)	n의 코사인값
tan(n)	n의 탄젠트값
max(n1, n2)	n1과 n2 중의 최대값
min(n1, n2)	n1과 n2 중의 최소값



3.5 클래스

- File 클래스
 - 파일이나 디렉토리 처리

[표 6.12] File 클래스의 주요 메소드

메소드	설명
canRead()	파일이 읽기가 가능하면 true, 아니면 false
canWrite()	파일이 쓰기가 가능하면 true, 아니면 false
delete()	파일 삭제
getPath()	파일 경로 반환
getName()	파일 이름 반환
length()	파일 길이 반환
mkdir()	디렉토리 생성



3.5 클래스

■ 예제 : class-method.jsp

```
4 : <b> Integer 클래스의 메소드 </b> <hr>
5 : <%
6 :     out.println("Integer.toString(56) => (string)" + Integer.toString(56) + "<br>");
7 :     out.println("Integer.parseInt(W"56W") => (int)" + Integer.parseInt("56") + "<br>");
8 : %>
9 : <br>
10 : <b> String 클래스의 메소드 </b> <hr>
11 : <%
12 :     String str = "milk";
13 :     String str2 = "MILK";
14 : %>
15 : <%= "sting : milk => 대문자 : " + str.toUpperCase()+" => 소문자 : " +
      str2.toLowerCase() %>
16 : <br> <br>
17 : <%
18 :     out.println("HOUSE".equals("House"));
19 : %>
```



3.5 클래스

■ 예제 : class-method.jsp(계속)

```
20 : <br><br>
21 : <b> Date 클래스의 메소드 </b><hr>
22 : <%!
23 :     Date d = new Date();
24 : %>
25 : <%= d.getYear() + 1900 %>-<%= d.getMonth() + 1 %>-<%= d.getDate() %>
26 : <br><br>
27 : <b> Math 클래스의 메소드</b><hr>
28 : <%= "-5의 절대값 : " + Math.abs(-5) + "<br> 4의 제곱근 : " + Math.sqrt(4)%>
```

3.5 클래스

■ 실행 결과



The screenshot shows a web browser window with the address bar displaying `http://localhost:8080/class-method.jsp`. The page content is organized into sections for different Java classes, each with a title and a horizontal line separator. The sections are: **Integer 클래스의 메소드** (Integer class methods), **String 클래스의 메소드** (String class methods), **Date 클래스의 메소드** (Date class methods), and **Math 클래스의 메소드** (Math class methods). Each section contains the results of specific method calls, such as `Integer.toString(56)` returning `(string)56` and `Integer.parseInt("56")` returning `(int)56`.

```
Integer 클래스의 메소드
Integer.toString(56) => (string)56
Integer.parseInt("56") => (int)56

String 클래스의 메소드
sting : milk => 대문자 : MILK => 소문자 : milk
false

Date 클래스의 메소드
2012-12-7

Math 클래스의 메소드
-5의 절댓값 : 5
4의 제곱근 : 2.0
```




3.6 함수

- 사용자가 특정 기능을 수행하는 부분을 함수로 정의하여 사용함
- 함수 선언

```
형식 :  
<%  
    [리턴_타입] 함수명( [매개변수_리스트] ) {  
        문장_리스트  
        [ return 결과값; ]  
    }  
%>
```

- 함수 호출

```
형식 :  
<%  
    함수명( [매개변수_리스트] );  
%>
```



3.6 함수

- 예제 : function.jsp
 - 함수 선언

```
2 :  <%!  
3 :      int p, q;  
4 :      int square(int i,int j){  
5 :          int value = 1;  
6 :          int k = 0;  
7 :          while (k < j){  
8 :              value = value * i;  
9 :              k = k + 1;  
10 :          }  
11 :          return value;  
12 :      }  
13 : %>
```



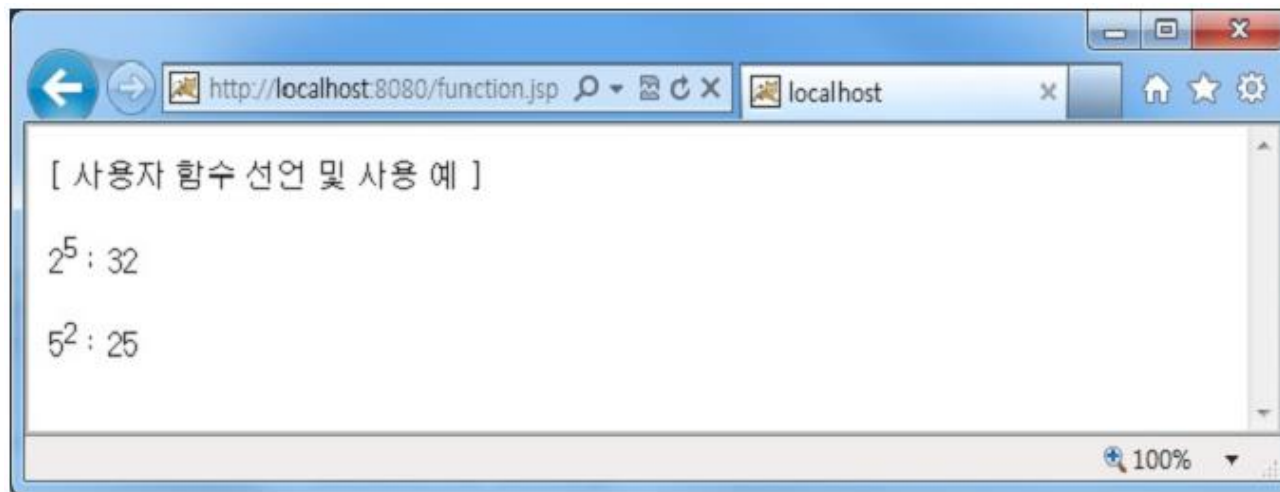
3.6 함수

- 예제 : function.jsp(계속)
 - 함수 호출

```
14 :  
15 : <%  
16 :             out.println("[ 사용자 함수 선언 및 사용 예 ]");  
17 :             p = square(2, 5);  
18 : %>  
19 :  
20 : <br><br> 2<sup>5</sup> : <%= p %>  
21 : <%  
22 :             q = square(5, 2);  
23 : %>  
24 : <br><br> 5<sup>2</sup> : <%= q %>
```

3.6 함수

- 실행 결과





3.7 예외 처리

■ try-catch-finally 문

```
형 식 : try {  
        문장_리스트  
    }  
    catch(예외타입1, 식별자1) {  
        문장_리스트  
    }  
    [ catch(예외타입2, 식별자2) {  
        문장_리스트  
    } ]*  
    [ finally {  
        예외 발생과 관계없이 무조건 수행되는 코드  
    } ]
```



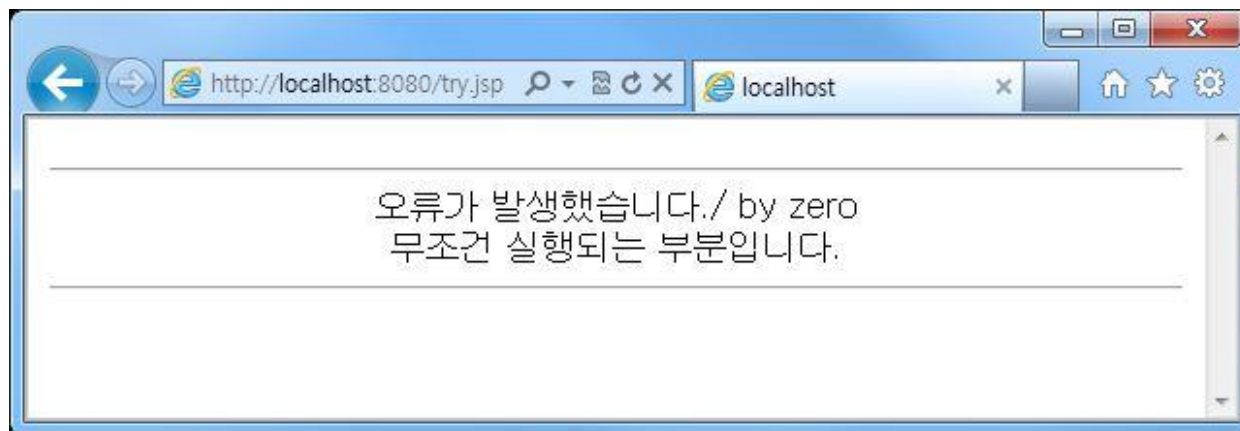
3.7 예외 처리

■ 예제 : try.jsp

```
6 : <%
7 :     try{
8 :         int a = 0;
9 :         int num = 100 / a;
10 :
11 :         out.println(num);
12 :     }
13 :     catch(Exception e){
14 :         out.println("오류가 발생했습니다." + e.getMessage() + "<br>");
15 :     }
16 :     finally{
17 :         out.println("무조건 실행되는 부분입니다.");
18 :     }
19 : %>
```

3.7 예외 처리

- 실행 결과





4 JSP 내장 객체

- request 객체
- response 객체
- out 객체
- session 객체
- application 객체



4.1 request 객체

- 서버에서 클라이언트에게 데이터를 요청할 때 사용하는 객체

형 식 :

```
request.메소드( [변수] );
```

[표 6.13] request 객체의 주요 메소드

메소드	내용
String getParameter(name)	매개변수 name의 값을 반환하며, 지정된 매개변수가 없으면 null 반환
String getCharacterEncoding()	전송된 데이터의 문자 인코딩 방식 반환

4.1 request 객체

■ 예제 : post.html

```
1 <!DOCTYPE html>
2 <html>
3   <head><title> Request 객체를 이용한 post 방식 예제 </title>
4   <meta charset="UTF-8">
5   </head>
6   <body>
7     <hr>
8     <center>
9       Post 방식을 통한 데이터의 전송 <br><br>
10      <form action="post.jsp" method="post">
11        이름 : <input type="text" name="name"><br><br>
12        학번 : <input type="text" name="id"><br><br>
13        소속 : <input type="text" name="dept"><br><br>
14        <input type="submit" value="입력">
15      </form>
16    </center>
17    <hr>
18  </body>
19 </html>
```

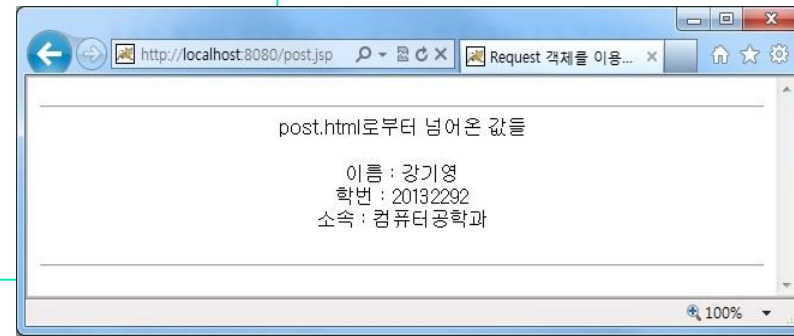
■ 실행 결과



4.1 request 객체

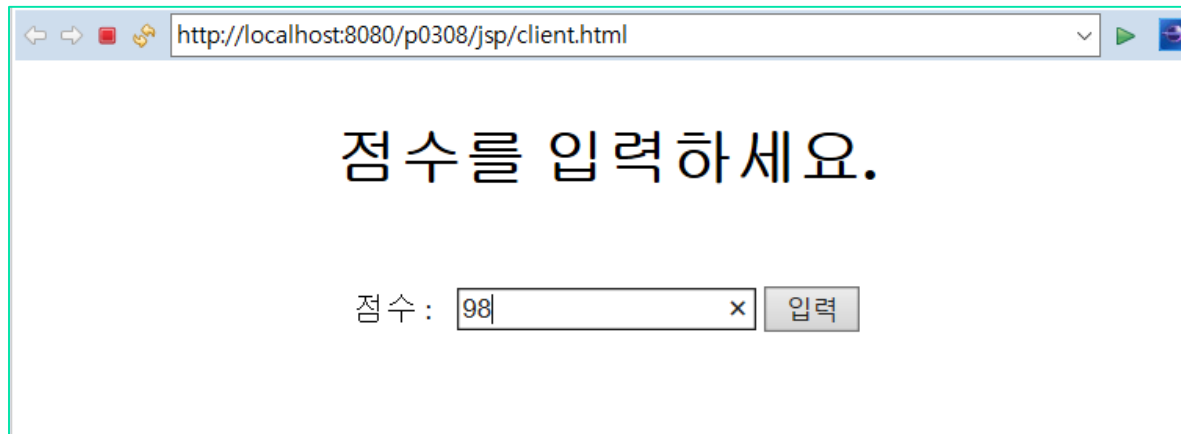
■ 예제 : post.jsp

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <% request.setCharacterEncoding("utf-8"); %>
4 <html>
5   <head><title> Request 객체를 이용한 post 방식 예제 </title></head>
6   <body>
7     <center>
8       <hr>
9       post.html로부터 넘어온 값들 <br><br>
10    <%
11      String name = request.getParameter("name");
12      String id = request.getParameter("id");
13      String dept = request.getParameter("dept");
14
15      out.println("이름 : " + name + "<br>");
16      out.println("학번 : " + id + "<br>");
17      out.println("소속 : " + dept + "<br>");
18    %><br><br>
19    <hr>
20  </center>
21 </body>
22 </html>
```

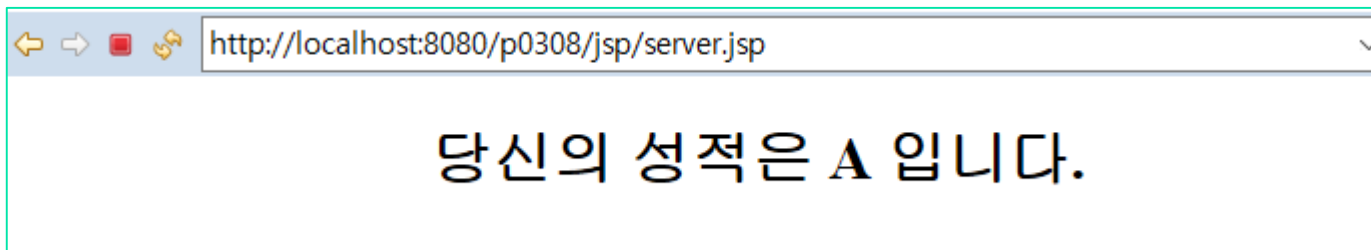


4.1 request 객체

- [실습] 다음과 같이 성적을 입력받아 학점을 처리하는 request 객체를 이용한 jsp 프로그램을 작성하시오.
- 실행 결과



A screenshot of a web browser window. The address bar shows the URL `http://localhost:8080/p0308/jsp/client.html`. The main content area displays the text "점수를 입력하세요." (Please enter your score.) Below this text is a form consisting of a label "점수 :" followed by a text input field containing the value "98", a close button (x), and a button labeled "입력" (Input).



A screenshot of a web browser window. The address bar shows the URL `http://localhost:8080/p0308/jsp/server.jsp`. The main content area displays the text "당신의 성적은 A 입니다." (Your score is A.)