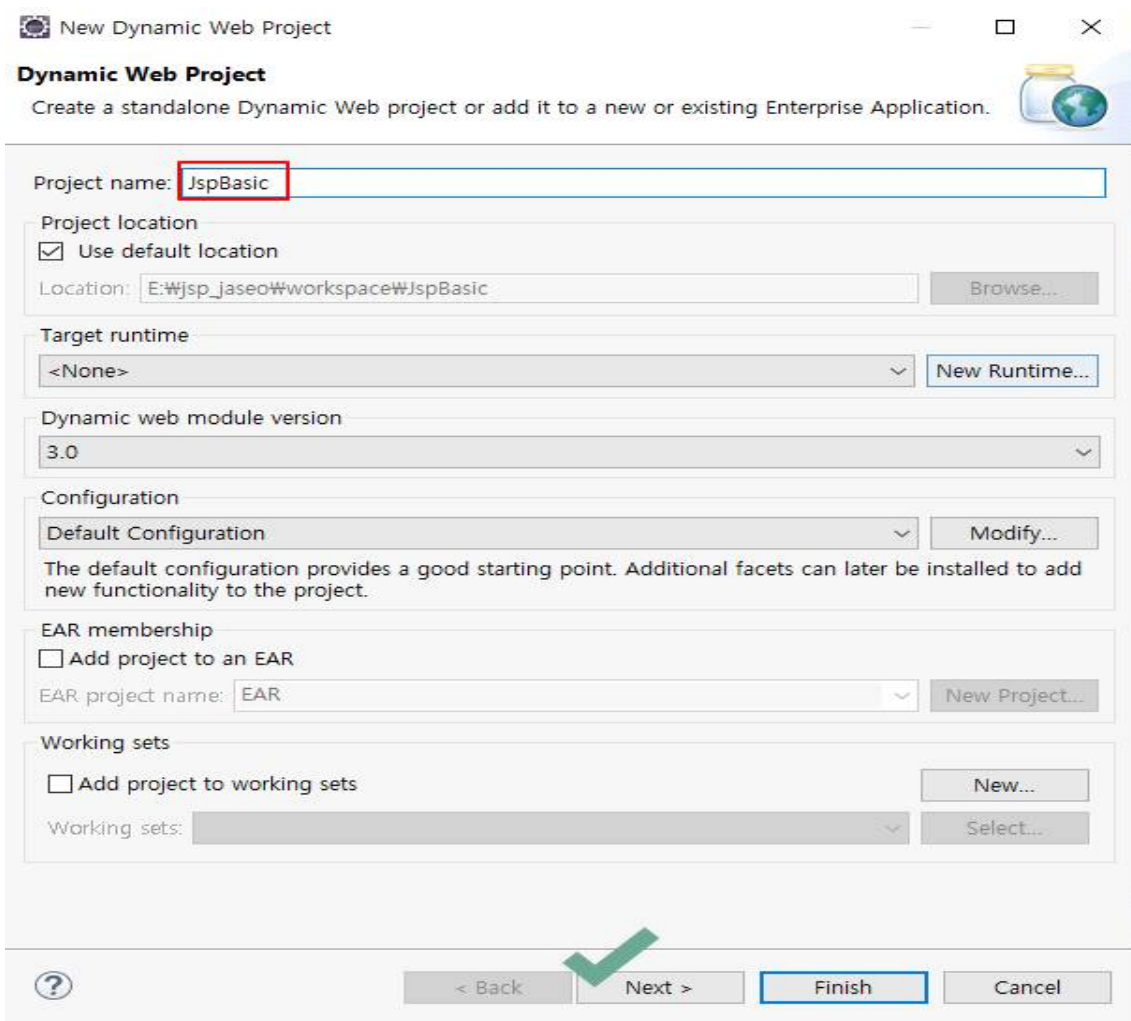
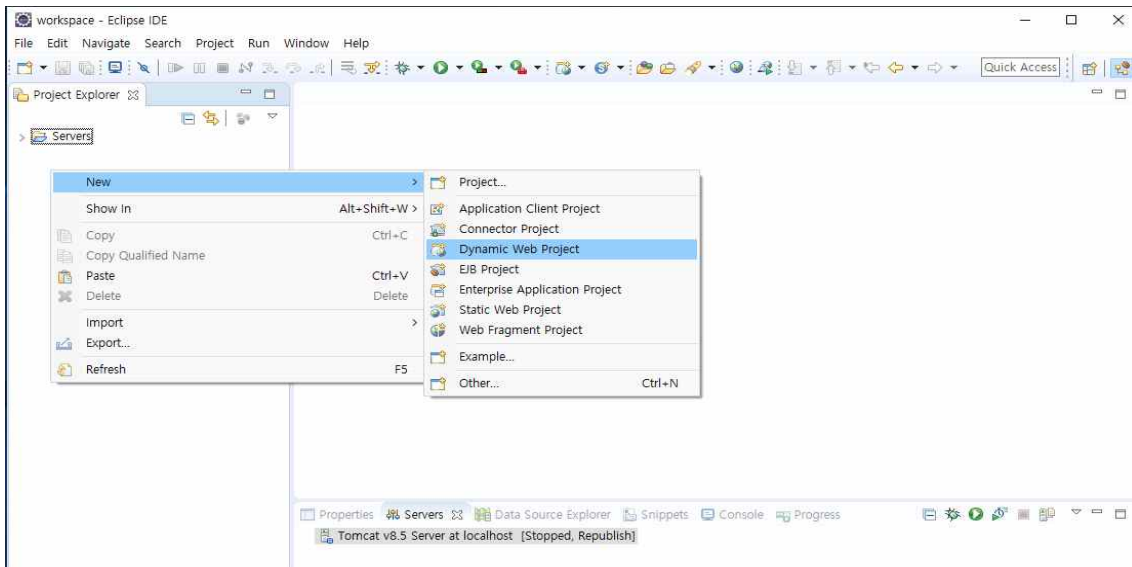


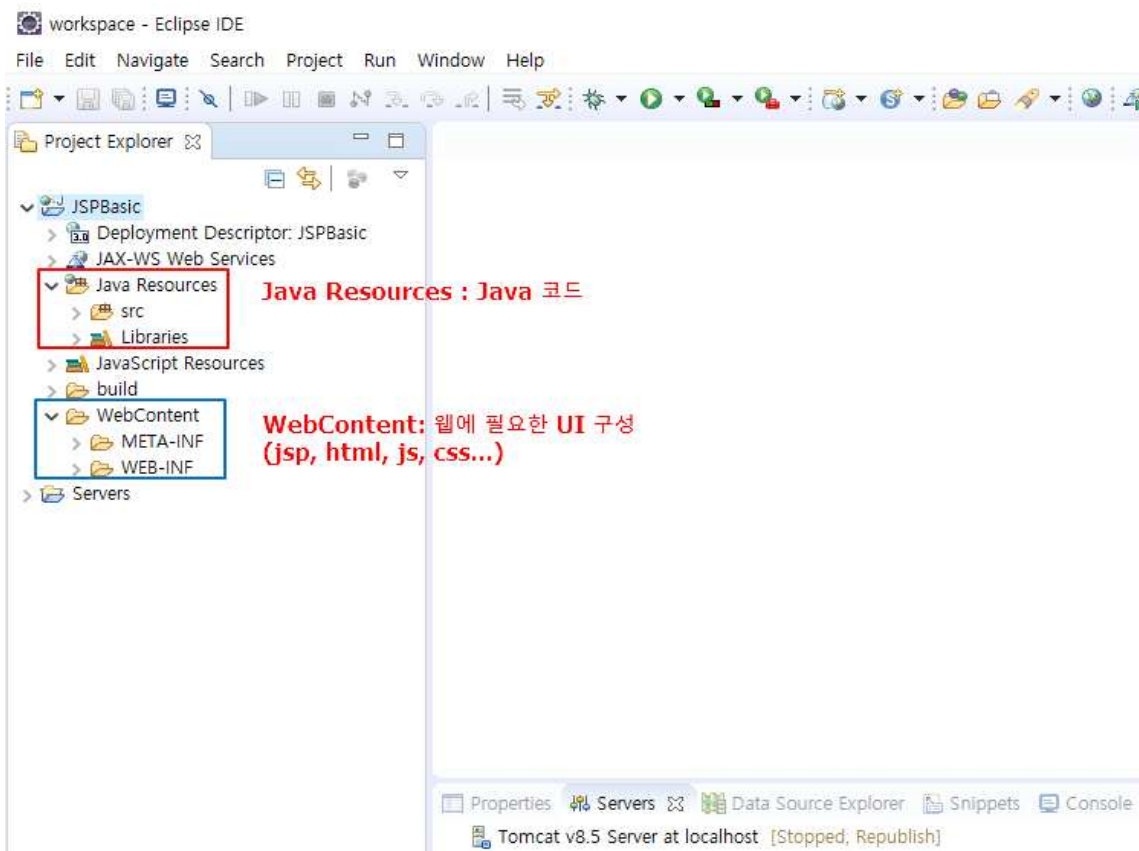
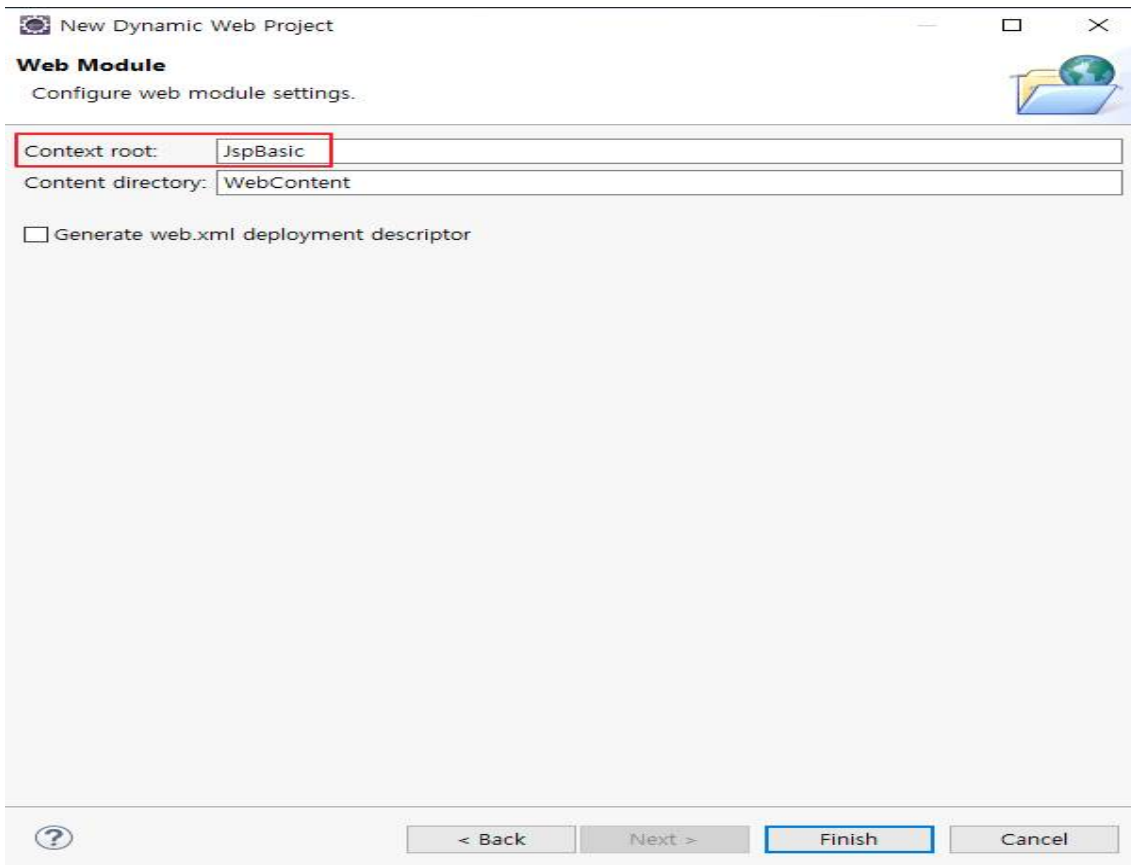
## 목차

<a href="#">1. Test</a>	... p2
<a href="#">2. Html</a>	... p9
<a href="#">3. JSP - 개념 / JSP tag</a>	... p14
<a href="#">4. JSP - 내장객체 (1. request)</a>	... p33
<a href="#">4-2. JSP - 내장객체 (2. response)</a>	... p50
<a href="#">4-3. JSP - 내장객체 (3. session, cookie)</a>	... p61
<a href="#">4-4. JSP - 내장객체 (3-2. application)</a>	... p76
<a href="#">4-5. JSP - 내장객체 (4. exception)</a>	... p77
<a href="#">5. JSP - Action Tag</a>	... p86
<a href="#">6. JSP - Java bean</a>	... p94
<a href="#">7. JSP - Servlet</a>	... p105
<a href="#">8. JDBC - DB연동 실습</a>	... p118
<a href="#">9. JDBC - MVC 패턴 실습</a>	... p130
<a href="#">10. Transaction/ Connection pool</a>	... p135
<a href="#">11. WebProject 1</a>	... p141
<a href="#">12. WebProject 2</a>	... p144
<a href="#">13. WebProject 3</a>	... p147
<a href="#">14. WebProject 4 / EL /JSTL</a>	... p149
<a href="#">15. JSTL - MVC2</a>	... p156 # MyFirstWeb5
<a href="#">16. MVC 2-2</a>	... p159 # MyFirstWeb6 - 게시판
<a href="#">17. MVC 2-3</a>	... p161 # MyFirstWeb7

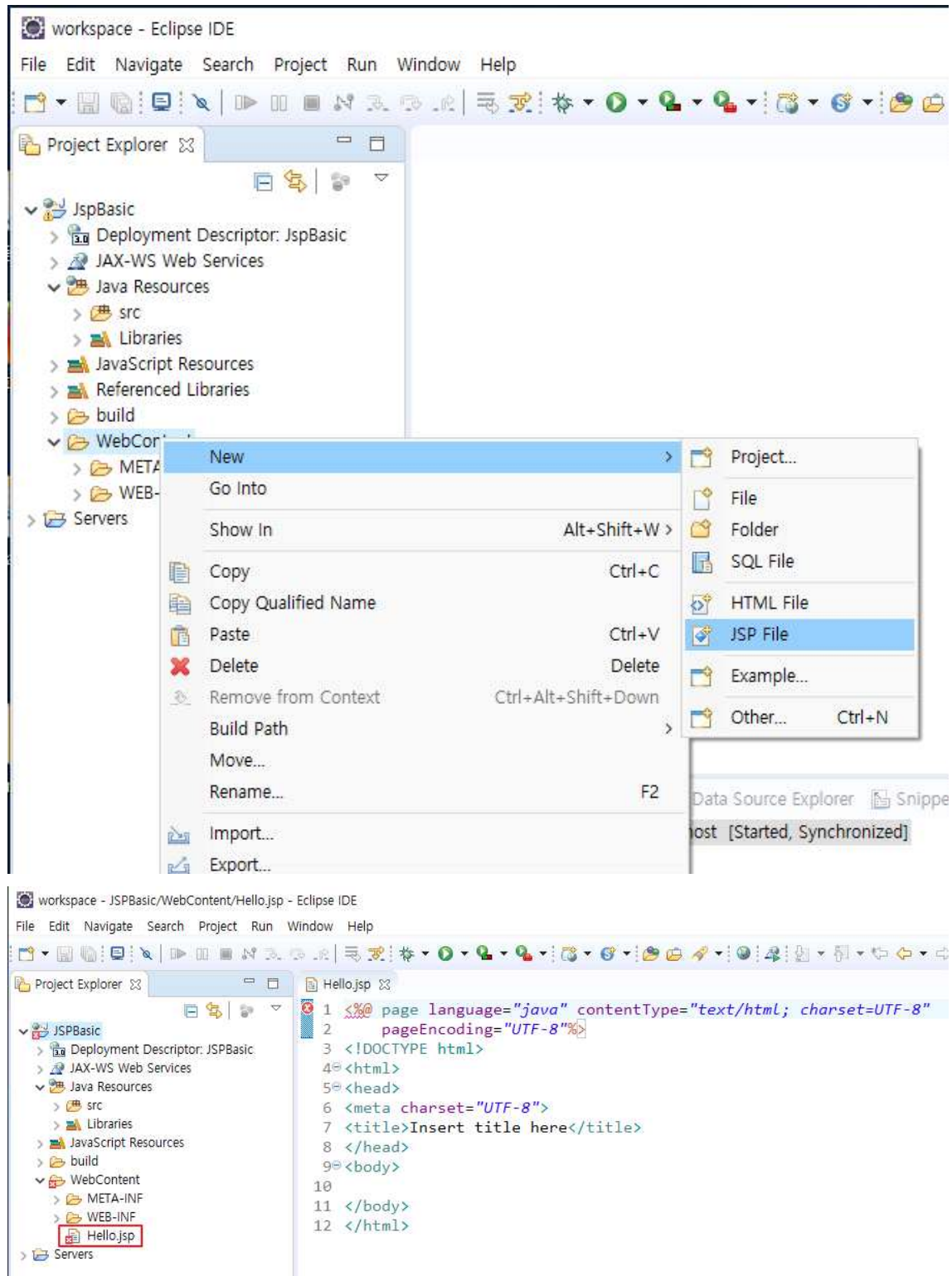
# 1. Test

## 1) 프로젝트 생성

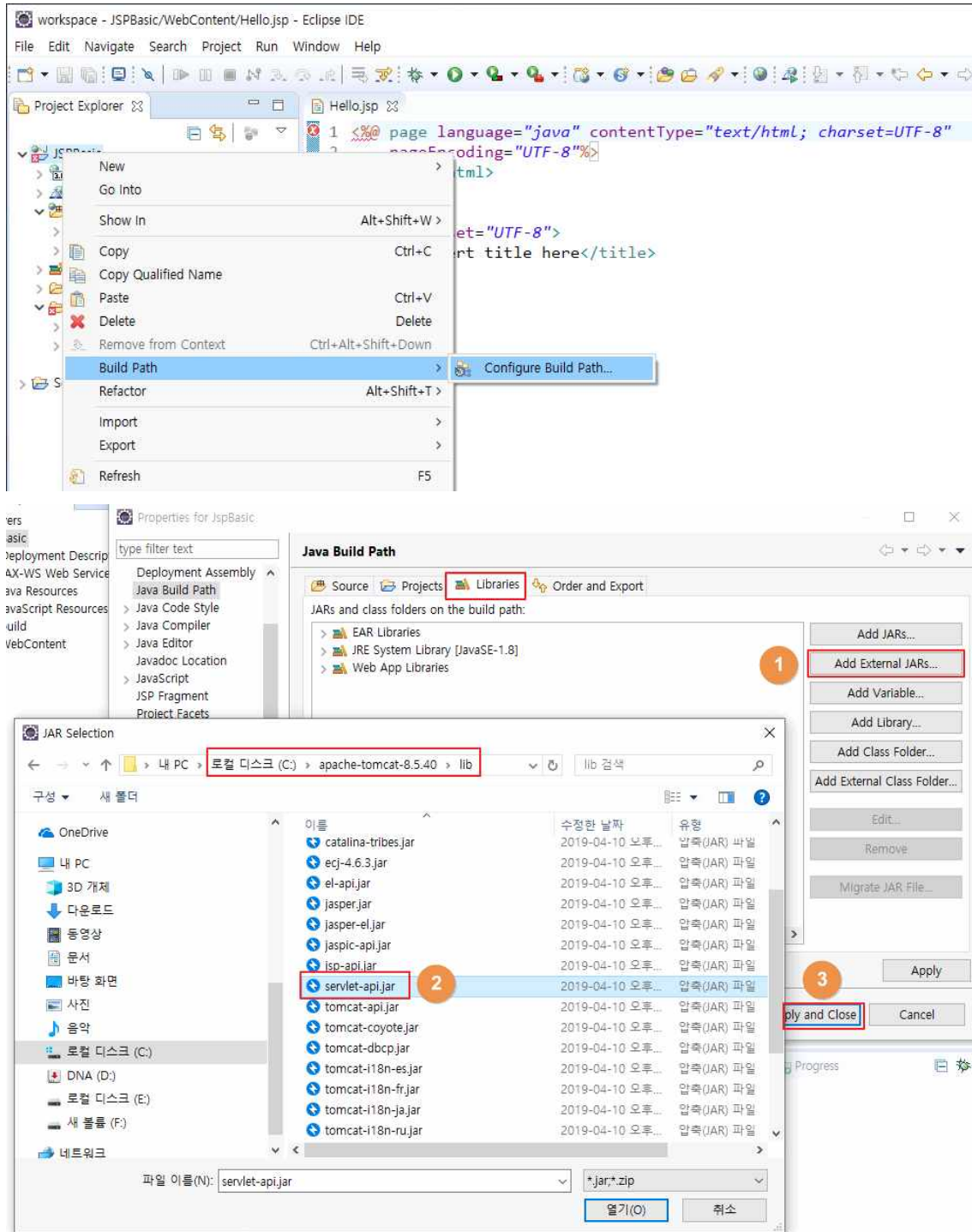




## 2) JSP 파일 생성

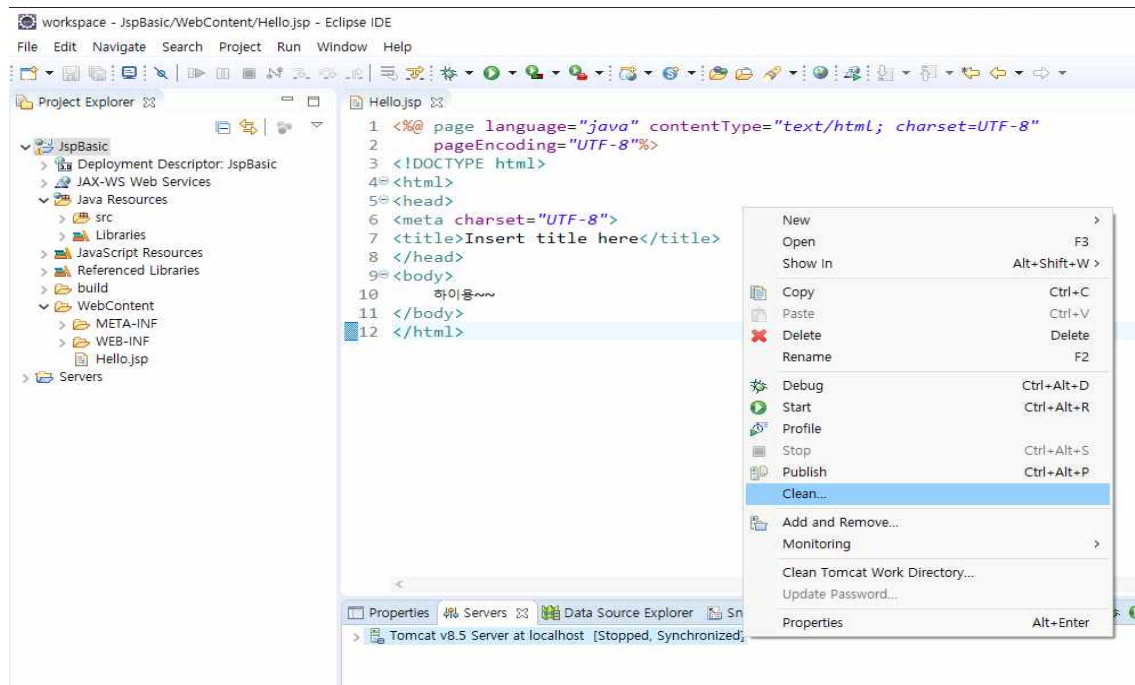


### 3) tomcat jar 파일 추가

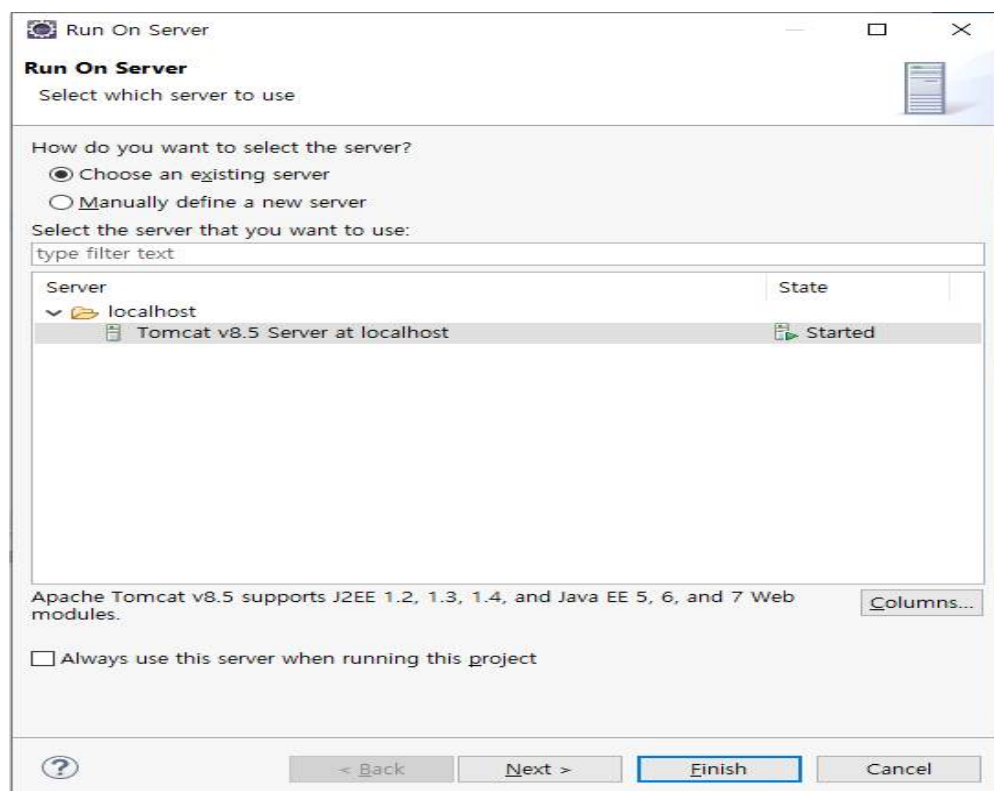


#### 4) Server 가동

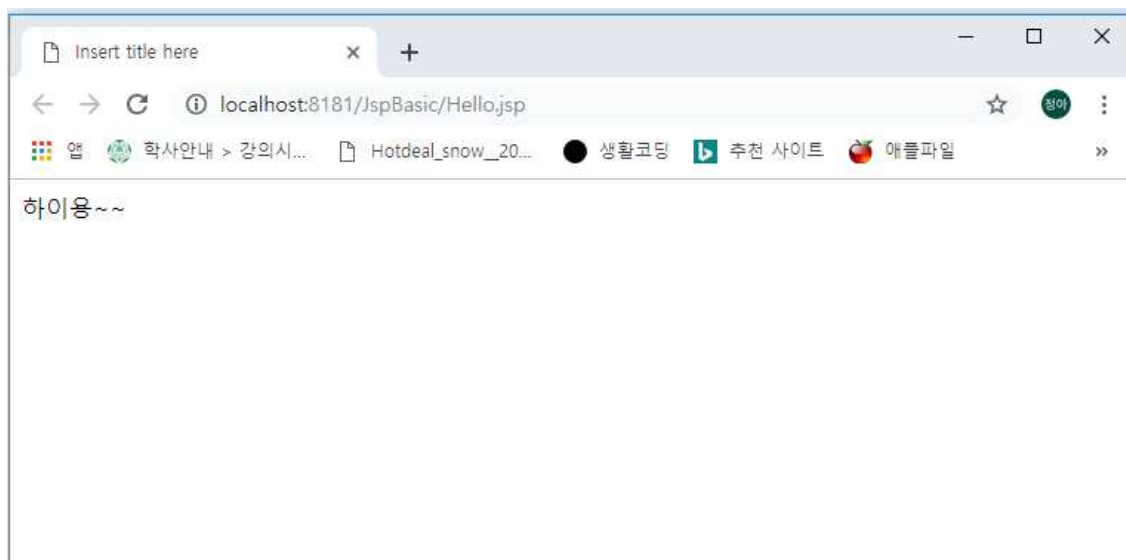
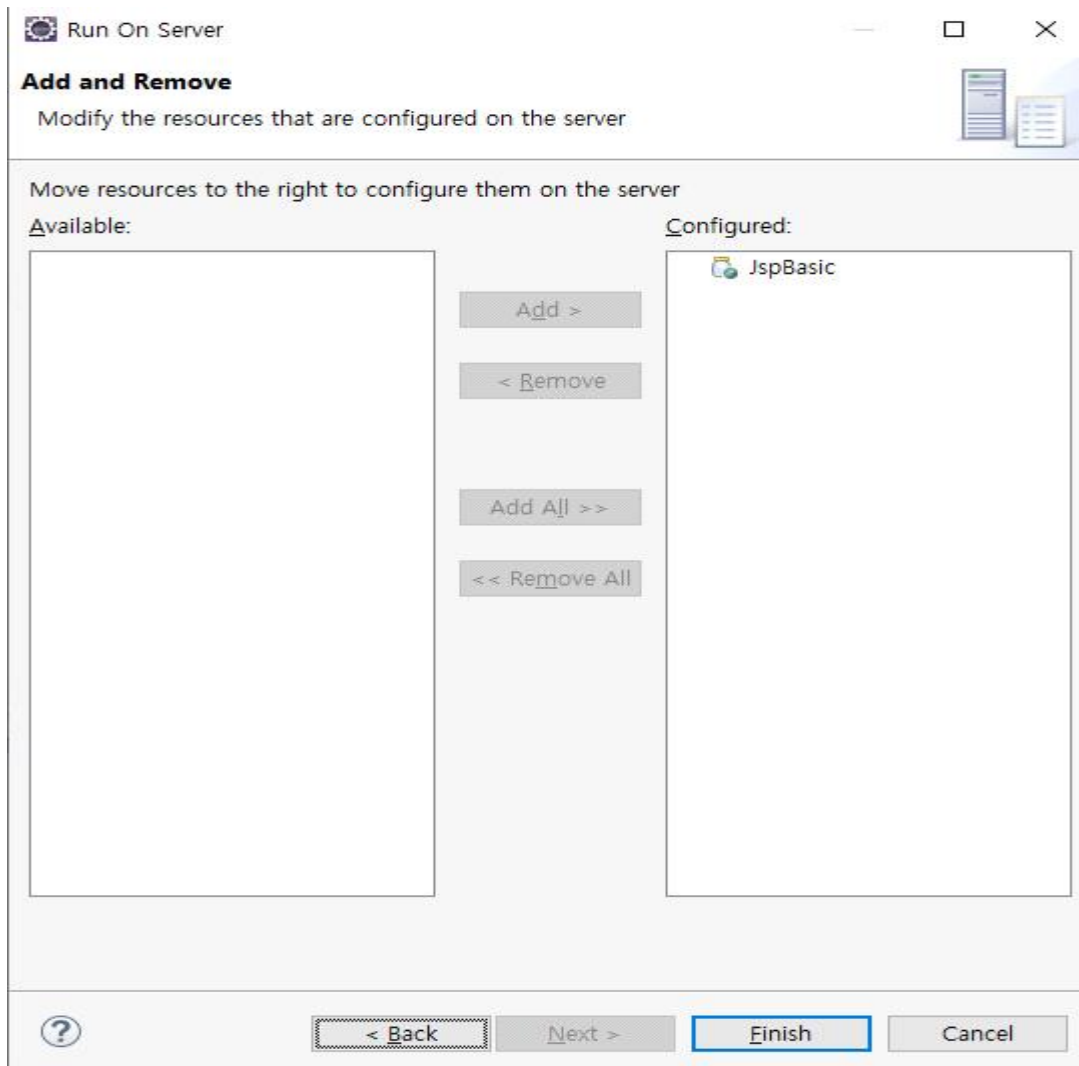
- ▶ clean
- ▶ start



- ▶ Ctrl + F11



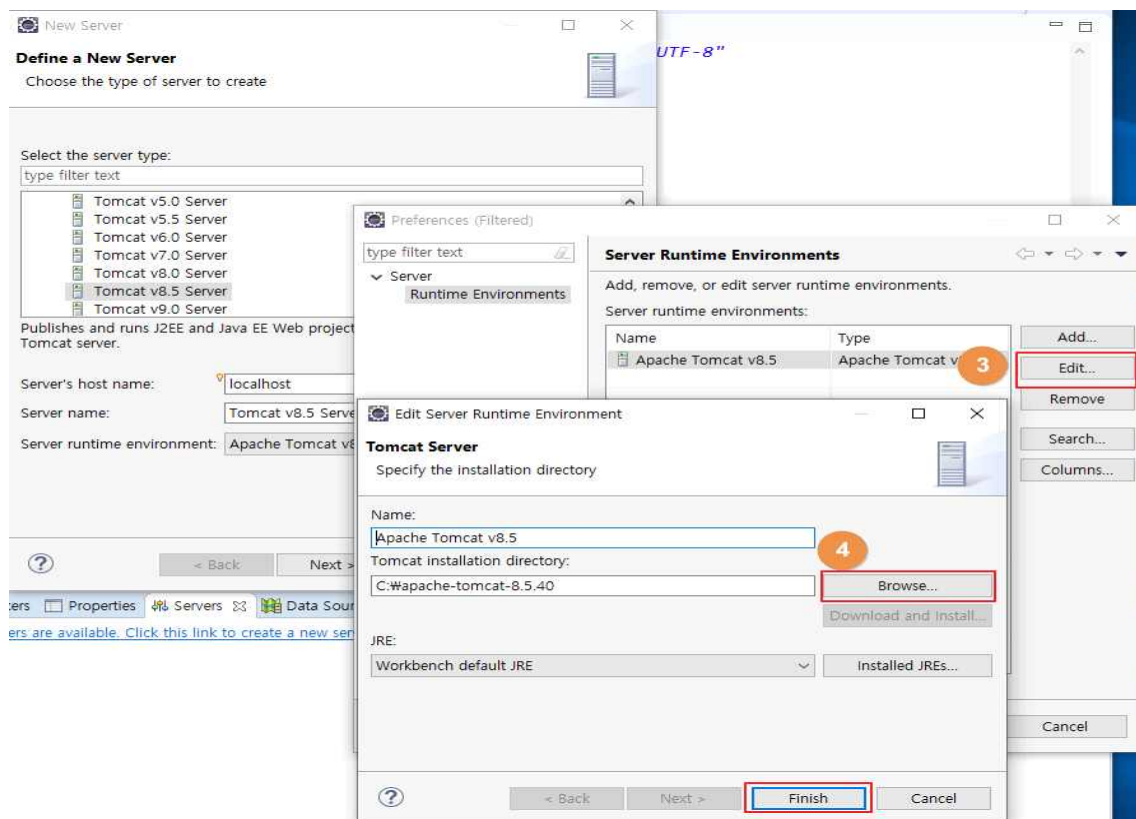
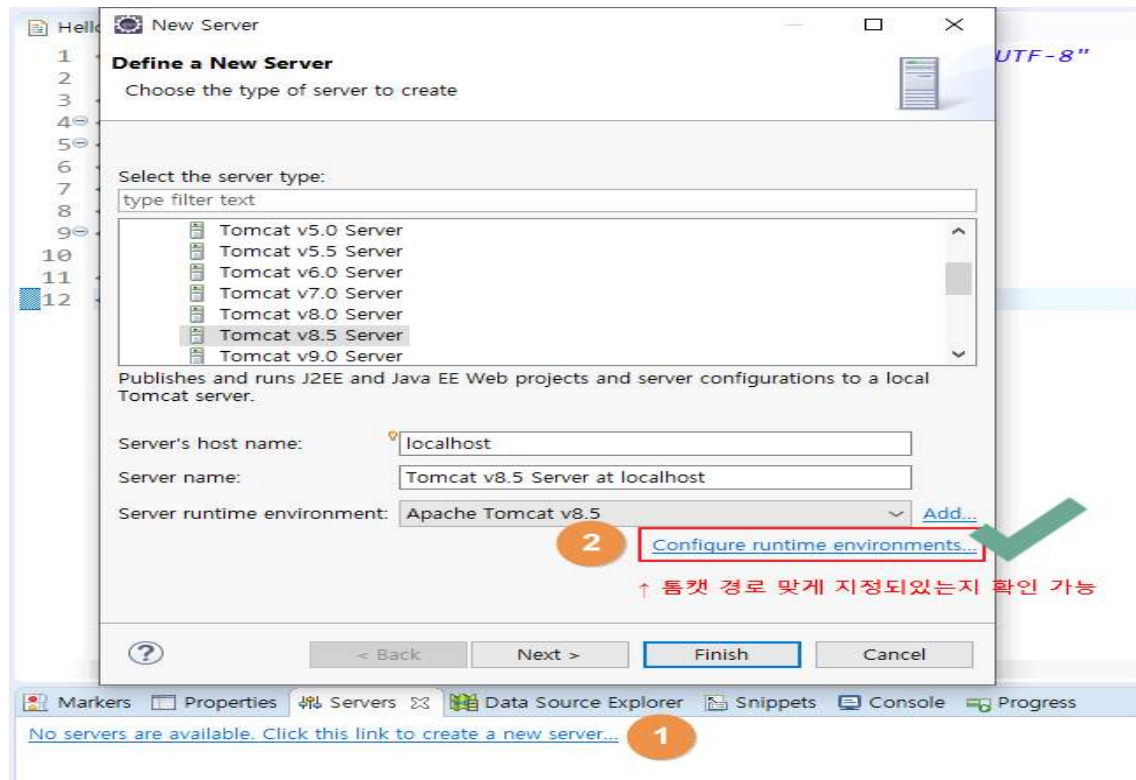




localhost:8181/JspBasic/Hello.jsp  
 도메인/context root/파일명

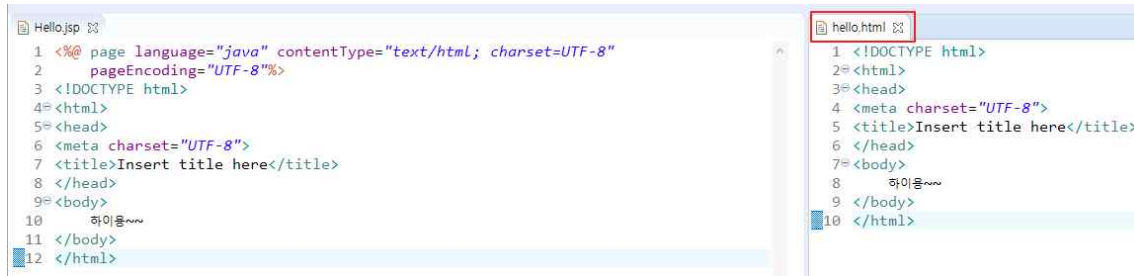
4-2) 서버 가동 에러시...

서버 지우고 다시 생성해서 테스트





## 2. HTML



### 1) 개념

- Hyper Text Markup Language의 약자
- 정적인 웹페이지를 만드는 언어
- 메모장, 한글, 워드... 문서작업이 가능한 프로그램에서는 모두 작성 가능하다.
- HTML 문서는 태그라는 명령어로 구성되어있다.
- 태그의 기본 형식

<태그이름>문서의 내용</태그이름>  
여는태그                      닫는태그

### 2) html 참고 사이트

<https://www.w3schools.com/>

### 3) HTML form 관련 태그

- input태그: 데이터를 입력하기 위해 사용되는 태그. 속성으로 태그의 종류를 지정합니다.
  - type 속성: 태그의 종류를 지정
- ① text: 일반적인 text데이터를 입력하기 위해 사용.
  - ② password: 로그인, 회원가입 페이지 등에서 비밀번호 입력을 하기 위해 사용.
  - ③ checkbox: 데이터 값을 여러 개 전송해야 할 때 사용.
  - ④ radio: 여러 개의 데이터 값 중에 한 개의 값만 전송해야 할 때 사용.
  - ⑤ select: 리스트 형태의 데이터를 사용.
  - ⑥ textarea: 여러 줄의 텍스트를 입력하기 위해 사용. 속성 cols와 rows로 행과 열의 크기를 지정.
  - ⑦ file: 파일 업로드를 위해 사용.
  - ⑧ button: 버튼 형태를 만들기 위해 사용. 속성으로 value를 지정하여 버튼 안에 들어갈 문자열을 입력.
  - ⑨ submit: form내의 데이터를 서버로 전송할 때 사용.
  - ⑩ reset: form내의 데이터를 초기화할 때 사용.

#### 4) sample

# hello.html

```
<!-- html주석은 이렇게 사용합니다. -->

<!-- 이 문서는 html 표준 규약을 준수하고 있습니다. -->
<!DOCTYPE html>
<html> <!-- html 문서의 시작태그 -->
<head> <!-- 현재 페이지의 속성과 정보를 설정 -->

<meta charset="EUC-KR"> <!-- 페이지 인코딩을 한글로 처리 -->

<!-- 페이지 제목을 설정하는 태그 -->
<title>html 기초 연습</title>

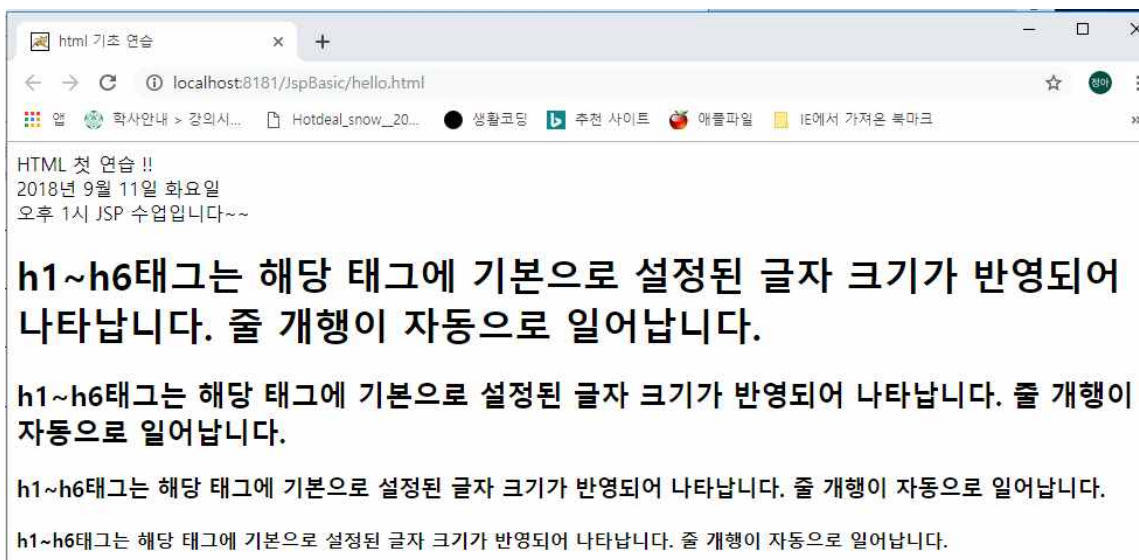
</head>
<body> <!-- 웹 브라우저에 실제로 보여질 내용 -->

HTML 첫 연습 !!
<br> <!-- 줄 개행 태그(\n) -->
2018년 9월 11일 화요일 <br>

오후 1시 JSP 수업입니다~~

<h1>h1~h6태그는 해당 태그에 기본으로 설정된 글자 크기가 반영되어 나타납니다. 줄 개행이 자동으로 일어납니다.</h1>
<h2>h1~h6태그는 해당 태그에 기본으로 설정된 글자 크기가 반영되어 나타납니다. 줄 개행이 자동으로 일어납니다.</h2>
<h3>h1~h6태그는 해당 태그에 기본으로 설정된 글자 크기가 반영되어 나타납니다. 줄 개행이 자동으로 일어납니다.</h3>
<h4>h1~h6태그는 해당 태그에 기본으로 설정된 글자 크기가 반영되어 나타납니다. 줄 개행이 자동으로 일어납니다.</h4>

</body>
</html>
```



## # img\_link.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="EUC-KR">
</head>
<body>

<!-- 페이지 내에 이미지 삽입은 img태그를 사용합니다. -->
<!-- img태그의 src속성에는 이미지파일의 경로를 지정합니다. -->

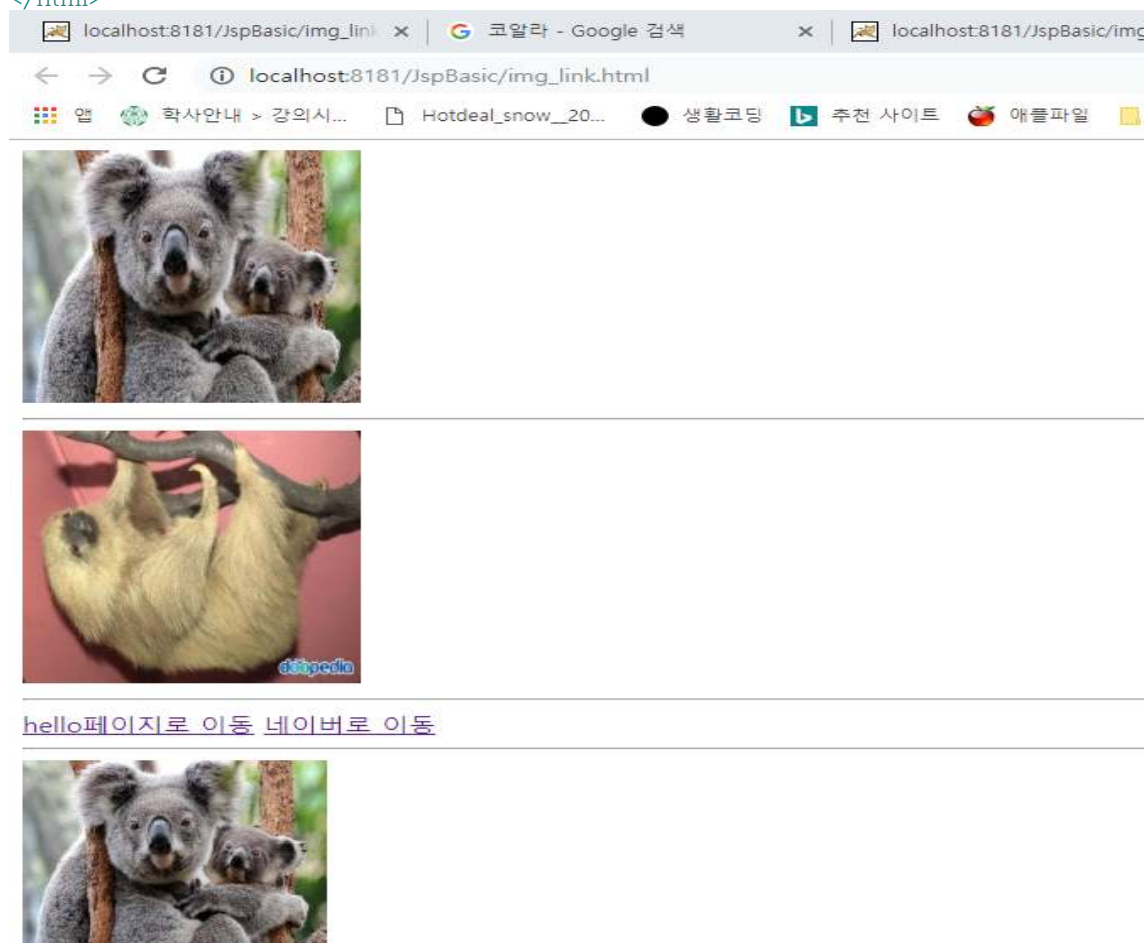

<hr> <!-- 구분선을 만들어주는 태그입니다. -->



<hr>
<!-- 페이지 링크를 걸려면 a태그를 사용합니다 -->
<!-- a태그의 href속성으로 연결할 페이지나 url주소를 적습니다. -->
<a href="hello.html">hello페이지로 이동</a>
<a href="https://www.naver.com">네이버로 이동</a>
<hr>

<a href="http://www.daum.net">
  
</a>

</body>
</html>
```



## # form.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="EUC-KR">
</head>
<body>

<!-- form태그는 서버에 데이터를 전송하기 위해 사용합니다. 태그 내부의 action속성에는 데이터를 전달할 페이지 url주소를 입력합니다. -->
<form action="">
  <h3>회원가입 양식</h3>
  <hr>
  아이디: <input type="text"><br>
  비밀번호: <input type="password"><br>
  주민번호: <input type="text"> - <input type="password"><br>
  자기소개: <br>
  <textarea cols="50" rows="5"></textarea> <br>

  지역:
  <select>
    <option>서울</option>
    <option>경기</option>
    <option>인천</option>
    <option>대전</option>
  </select> <br>

  전공:
  <!-- &nbsp;는 스페이스바를 치는 명령 -->
  <input type="radio" name="major">경영학 &nbsp;
  <input type="radio" name="major">기계공학 &nbsp;
  <input type="radio" name="major">국어국문학 &nbsp; <br>

  취미:
  <input type="checkbox" name="hobby">독서 &nbsp;
  <input type="checkbox" name="hobby">웹툰보기 &nbsp;
  <input type="checkbox" name="hobby">수면 &nbsp;
  <input type="checkbox" name="hobby">명때리기 &nbsp; <br>

  <!-- submit버튼은 폼데이터를 서버로 전송하는 버튼입니다. form태그의 action속성에 지정한 url로 데이터를 전송합니다. -->
  <!-- reset버튼은 폼에 입력한 값들을 초기화합니다. -->
  <input type="submit" value="확인" &nbsp;
  <input type="reset" value="초기화" &nbsp;

</form>

</body>
</html>
```

회원가입 양식

아이디:

비밀번호:

주민번호:  -

자기소개:

지역:

전공: ☐ 경영학 ☐ 기계공학 ☐ 국어국문학

취미: ☐ 독서 ☐ 웹툰보기 ☐ 수면 ☐ 명때리기

## # table.html

```

<!DOCTYPE html>
<html>
<head>
<meta charset="EUC-KR">
</head>
<body>

<!-- table태그는 표를 작성하는 태그입니다. -->
<!-- border속성은 테두리선의 유무를 지정. 1로 지정시 테두리선이 생깁니다. -->
<!-- align속성은 브라우저 내부에 표를 어느 위치로 정렬시킬지를 지정. -->
<table border="1" align="center" width="300px">
  <!-- table의 자식태그 tr은 표의 행을 나타내는 태그입니다.-->
  <tr>
    <!-- tr의 자식태그 td는 한 행에 들어갈 열들을 지정합니다. -->
    <td>이름</td>
    <td>성별</td>
    <td>주소</td>
    <td>나이</td>
  </tr>

  <tr>
    <td>홍길동</td>
    <td>남자</td>
    <td>서울</td>
    <td>44세</td>
  </tr>

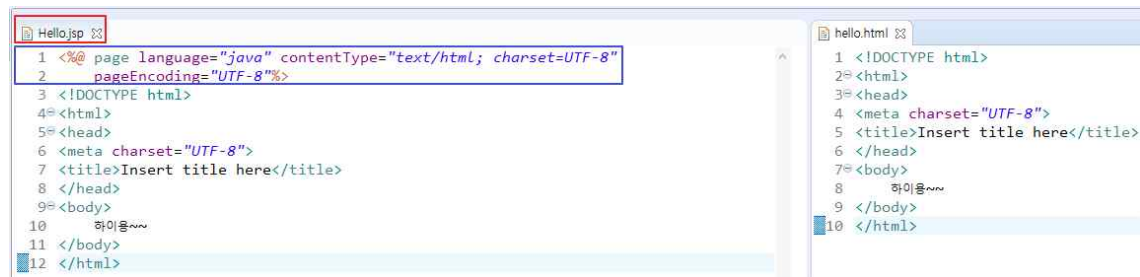
  <tr>
    <td>신사임당</td>
    <td>여자</td>
    <td>경기도</td>
    <td>33세</td>
  </tr>
</table>
</body>
</html>

```



이름	성별	주소	나이
홍길동	남자	서울	44세
신사임당	여자	경기도	33세

### 3.JSP - 개념 / JSP tag



#### 1) 개념

- ① 동적 웹어플리케이션 컴포넌트
- ② .jsp 확장자
- ③ 클라이언트의 요청에 동적으로 작동하고, 응답은 HTML을 이용.
- ④ JSP는 웹서버와 통신시에 자동으로 Servlet으로 변환됨.
- ⑤ MVC패턴에서 View로 이용됨.

cf. 면접 질문 유형: 객체 지향(oop), 상속, MVC 패턴??

#### 2) JSP 태그의 개념 이해

- Servlet은 java언어를 이용하여 문서를 작성하고, 출력객체(PrintWriter)를 이용하여 html 코드를 삽입합니다.
- jsp는 서블릿과 반대로 html코드에 java언어를 삽입하여 동적 문서를 만들 수 있습니다.
- html코드 안에 java코드를 삽입하기 위해서는 jsp태그를 이용해야 하며, 이러한 태그를 학습해야 합니다.

#### 3) JSP 태그의 종류

- ① 스크립트릿(scriptlet): <% %> 거의 모든 자바코드 기술 가능.

# 메서드 내 지역변수 선언

- ② 지시자(directive): <%@ %> 페이지 속성을 지정.

ex> <%@ page language="java" contentType="text/html; charset=UTF-8"  
pageEncoding="UTF-8"%>

- ③ 선언자(declaration): <%! %> 변수나 메서드 선언시 사용. # class 내 변수 선언

- ④ 표현식(expression): <%= %> 결과 값을 출력할 때 사용.

- ⑤ 주석(comments): <%-- --%> 코드 주석처리 시에 사용.



### 3-2) JSP 태그 - 상세

#### ① 스크립트릿(Scriptlet)

- JSP페이지에서 JAVA언어를 사용하기 위한 요소 중 가장 많이 사용하는 태그입니다.
  - 스크립트릿 안에는 우리가 알고 있는 거의 모든 JAVA코드를 사용할 수 있습니다.
- class 선언은 안됨?

#### ③ 선언(Declaration)

- JSP 페이지 내에서 사용되는 변수 또는 메서드를 선언할 때 사용합니다.
- 여기 선언된 변수 및 메서드는 전역의 의미로 사용됩니다.

#### ④ 표현식(Expression)

- JSP 페이지 내에서 사용되는 변수의 값 또는 메서드 호출 결과 값을 출력하기 위해 사용됩니다.
- 결과 값의 데이터 유형은 String이며, 세미콜론(;)을 사용할 수 없습니다.
- 표현식은 `out.println()`을 대체합니다.

#### ⑤ 주석(Comments)

- 실제 프로그램 실행에는 영향이 없고, 프로그램 설명 등의 목적으로 사용되는 태그입니다.
- HTML 및 JSP 주석이 각각 별도로 존재합니다.
- HTML 주석은 `<!-- -->` 표기하고, 웹 브라우저에서 페이지 소스보기를 하면 주석도 표기됩니다.
- JSP 주석은 `<%-- --%>` 표기하고, 웹 브라우저에서 소스보기를 해도 나타나지 않습니다.
- JAVA의 주석도 혼용 사용 가능합니다.(`//` , `/* ~~~ */`)

#### ② 지시자(directive)

- JSP페이지의 전체적인 속성을 지정할 때 사용합니다.

1. `page`: JSP페이지에 대한 정보를 지정한다. JSP가 생성하는 문서의 타입, 출력 버퍼의 크기, 에러 페이지 등 페이지에서 필요로 하는 정보를 설정합니다.
  - 페이지 디렉티브에 선언 될 수 있는 속성들
    - a. `language` - 언어를 지정합니다. java만 지정할 수 있습니다.
    - b. `import` - 패키지를 import할 때 사용합니다.
    - c. `errorPage` - 설정 된 페이지에서 에러가 발생했을 시에 미리 만들어 둔 에러 페이지를 호출합니다.
    - d. `contentType` - `text/html`이라고 쓰면 결과가 HTML문서로 만들어집니다.
    - e. `pageEncoding` - 출력 문자 인코딩입니다. 기본은 ISO-8859-1이며 한글을 사용하려면 `utf-8`로 지정합니다.
    - f. `trimDirectiveWhitespaces` - 디렉티브나 스크립트릿 코드로 인해서 만들어진 공백 문자를 제거하는 기능.
2. `include`: JSP 페이지의 특정 영역에 다른 문서를 포함시킵니다.
3. `taglib`: JSP 페이지에서 사용할 태그 라이브러리를 지정합니다.

#### 4) JSP 장점

- Java 코드를 첨가해 Html을 동적으로 만드는 기술
- Java 코드 노출 안되 보안상 좋음!(소스보기하면 html 코드만 보임)
- Html + Java

#### 5) JSP 아키텍처

- .jsp파일을 실행(요청)하면 웹 서버에서 우선 jsp파일을 java코드로 변환합니다. 그 이후 그 파일을 컴파일하고 html로 응답합니다.

ex) helloworld.jsp -> helloworld\_jsp.java(서버에서 servlet화)

-> helloworld\_jsp.class(서블릿 클래스 컴파일)

실행시 다시 html 코드로 변환

#### 6) 구글링: java api

Java API - Oracle Docs

<https://docs.oracle.com/javase/8/docs/api/>

<https://docs.oracle.com/javase/7/docs/api/>

## 7) sample

① 스크립트릿(scriptlet): <% %> 거의 모든 자바코드 기술 가능.

# scriptlet.jsp

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

<%
    int i = 0;
    while(true){
        i++;
        // out.println은 브라우저에 데이터를 출력할 수 있게 해주는 메서드임.
        out.println("2 x"+ i +"=" + (2*i) + "<br>");
    }

    <hr> <!-- html 코드가 시작되는 부분에서는 스크립트릿을 달아줘야 함. -->

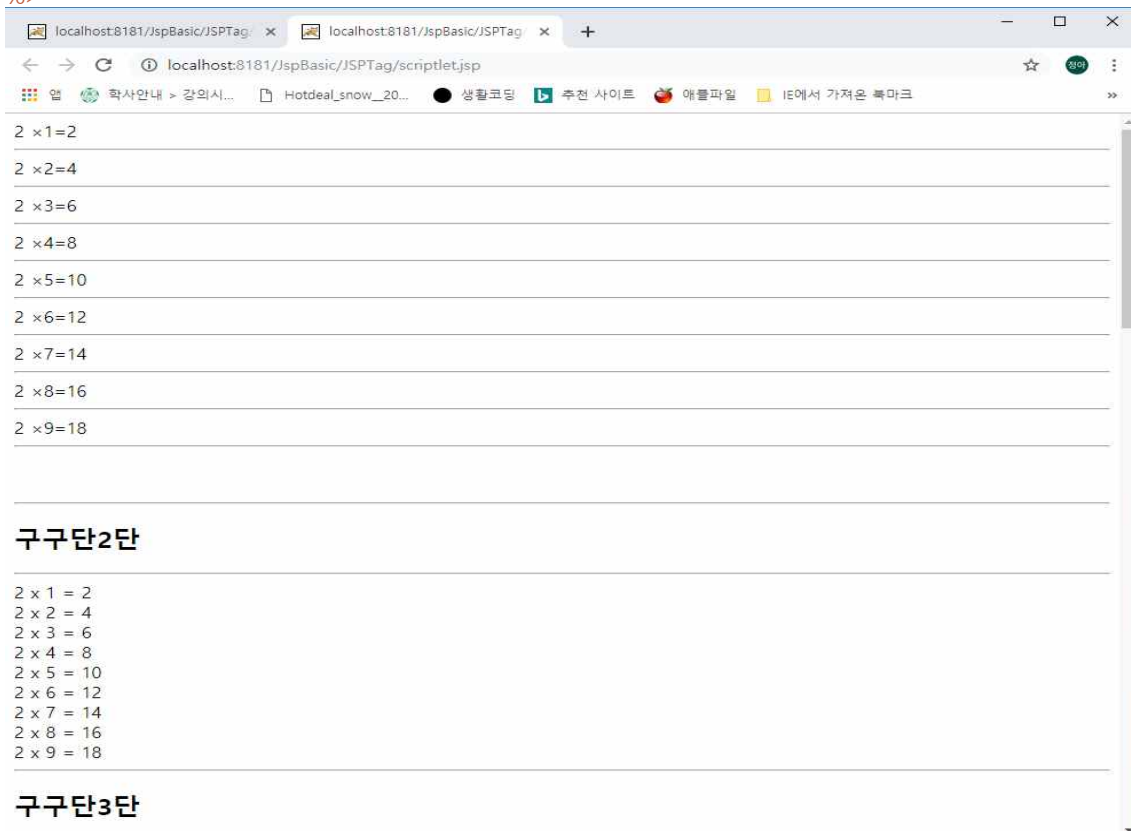
    if(i >= 9) break;
    } // close while

<br><br>

<%
    // 중첩 for문을 이용하여 2~9단까지의 구구단 출력

    for(int j=2; j<10; j++){
        out.println("<hr><h2>구구단" +j+"단</h2><hr>");
        for(int k=1; k<10; k++){
            out.println(j+" x "+k+" = "+(j*k)+"<br>");
        }
    }

%>
```



③ 선언자(declaration): <%! %> 변수나 메서드 선언시 사용. # class 내 변수 선언

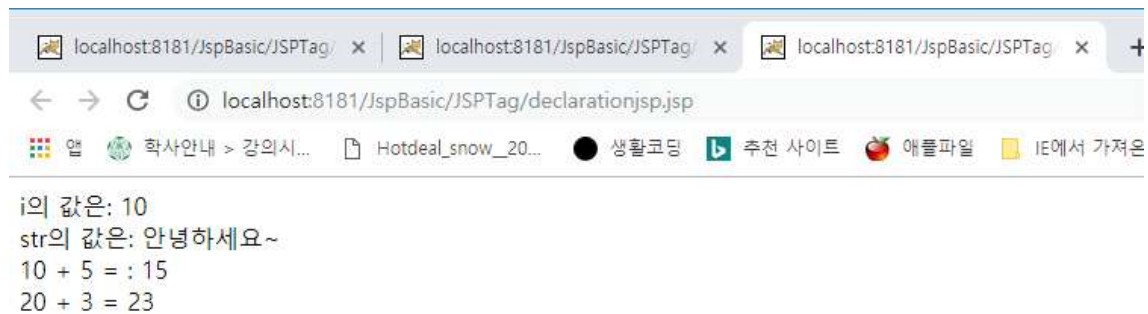
# declaration.jsp

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

```
<%!
    // 선언 (declaration) 영역
    int i = 10; // 전역변수
    String str = "안녕하세요~";

    public int sum(int a, int b){
        return a + b;
    }
%>

<%
    // 상제 코드 기술 (scriptlet)
    int result = sum(10, 5); //지역변수
    out.println("i의 값은: " + i + "<br>");
    out.println("str의 값은: " + str + "<br>");
    out.println("10 + 5 = : " + result + "<br>");
    out.println("20 + 3 = " + sum(20,3) + "<br>");
%>
```

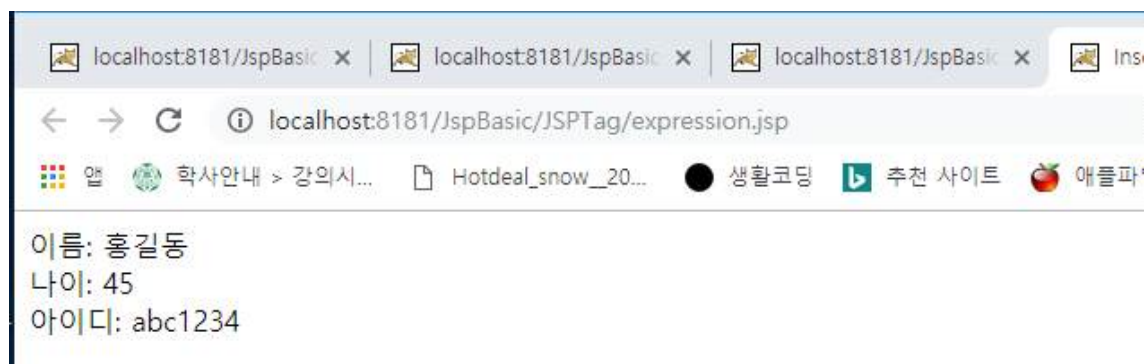


④ 표현식(expression): <%= %> 결과 값을 출력할 때 사용.

# expression.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%!
    // declaration(선언자)
    String name = "홍길동";    // class변수, member변수, field
    int age = 45;
%>

<%
    // scriptlet(스크립트릿)
    String id = "abc1234";
    int number = 123;
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<!-- expression(표현식) -->
<!-- 표현식에는 세미콜론(:)을 사용하지 않습니다. -->
이름: <%= name %> <br>
나이: <%= age %> <br>
아이디: <%= id %>
</body>
</html>
```



⑤ 주석(comments): <%-- --%> 코드 주석처리 시에 사용.

# comment.jsp

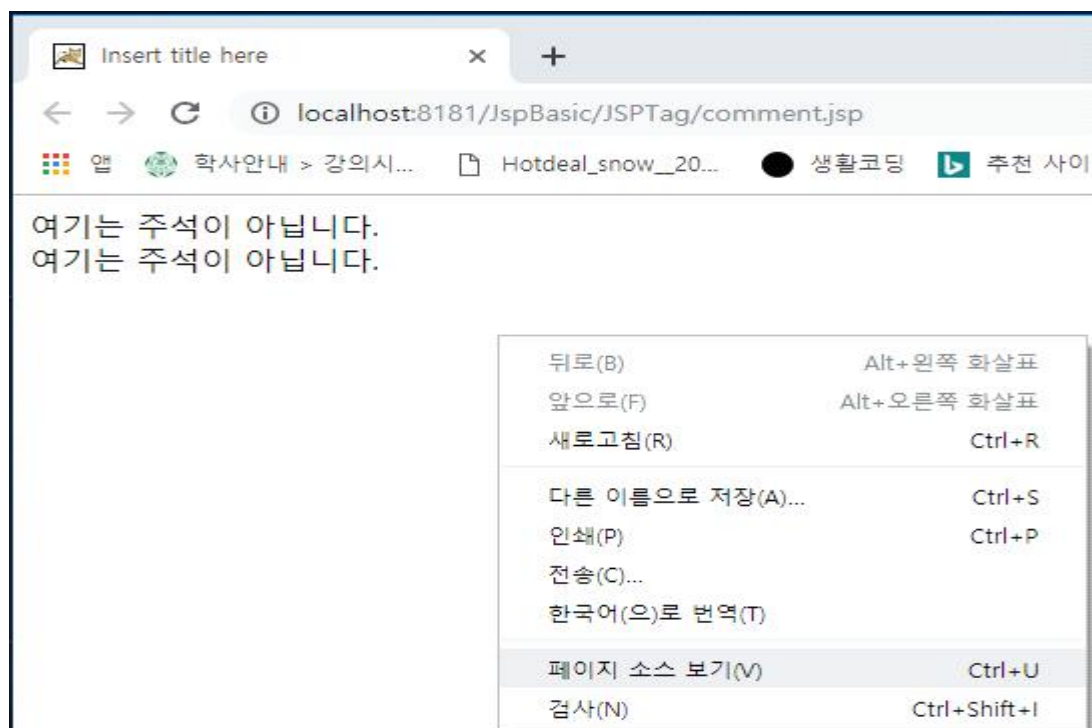
```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>

<%
    // int a = 100;
    /*
        여러 줄 주석
        히히히힃
        abab
    */

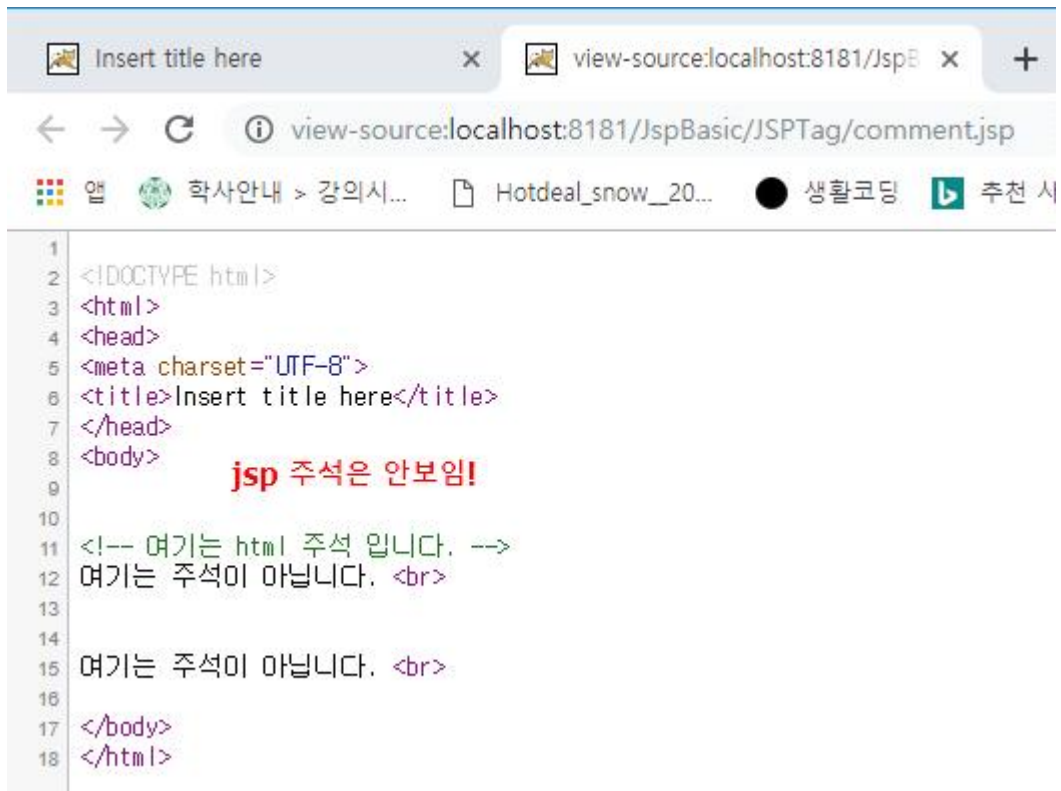
%>
<!-- 여기는 html 주석 입니다. -->
여기는 주석이 아닙니다. <br>

<%-- 여기는 jsp주석 입니다. --%>
여기는 주석이 아닙니다. <br>

</body>
</html>
```







```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Insert title here</title>
6 </head>
7 <body>
8     jsp 주석은 안보임!
9
10
11 <!-- 여기는 html 주석 입니다. -->
12 여기는 주석이 아닙니다. <br>
13
14
15 여기는 주석이 아닙니다. <br>
16
17 </body>
18 </html>
```

② 지시자(directive): <%@ %> 페이지 속성을 지정.

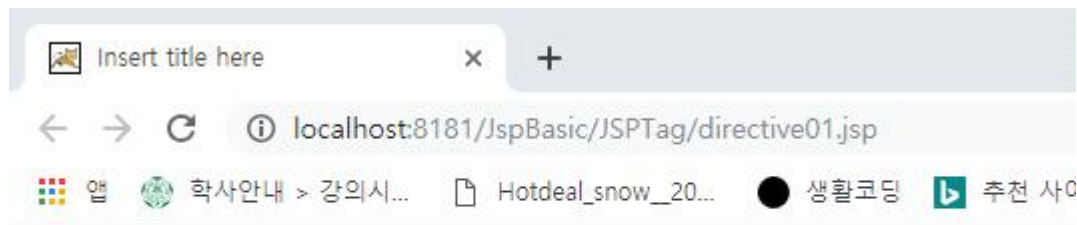
## # directive01.jsp

```
<%@page import="java.text.SimpleDateFormat"%>
<%@page import="java.util.*"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    Calendar cal = Calendar.getInstance();    // Calendar + [Ctrl+Space] => import
    int year = cal.get(Calendar.YEAR);
    int month = cal.get(Calendar.MONTH) + 1;
    int day = cal.get(Calendar.DAY_OF_MONTH);

    Date date = new Date();

    // Date클래스가 가져온 날짜정보가 우리나라 정서에 맞지 않기 때문에
    // 포맷을 변경하기 위해 사용하는 클래스가 SimpleDateFormat입니다.
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd a hh시 mm분 ss초");
    String time = sdf.format(date);
%>

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    오늘은 <%= year %>년 <%= month %>월 <%= day %>일 입니다. <br>
    <%= time %>
</body>
</html>
```



오늘은 2019년 5월 4일 입니다.  
2019-05-04 오후 05시 00분 04초

## # directive02\_main.jsp 애 실행 (Ctrl +F11)

```
<%@ 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>

<%@ include file="directive02_header.jsp" %>

<h2>방문자 수: <%=visit %>명</h2>
<h3>여기는 메인페이지 부분입니다 ~~</h3>
<%@ include file="directive02_footer.jsp" %>
<!--
</body>
</html>
-->
```

## # directive02\_header.jsp

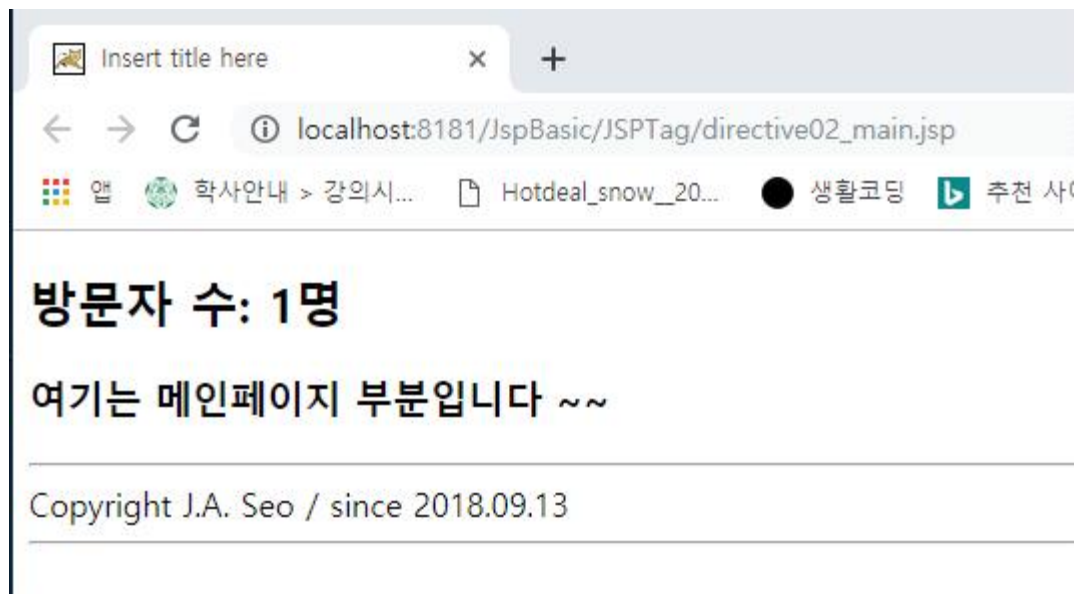
```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%!
    int visit = 0;    // 전역변수 선언
%>
%>
    visit++;
%>
```

## # directive02\_footer.jsp

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

<hr>

Copyright J.A. Seo / since 2018.09.13
<hr>
</body>
</html>
```



## # tag\_ex01.jsp

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

<!-- Declaration : JSP 파일에서 사용할 멤버변수, 메서드를 선언할 때 사용. -->
<%!
    int total = 0;

    int randomNumber(){
        // 1~10까지의 난수를 발생시켜서 해당 난수를 리턴하는 메서드를 만드세요.
        //return (int)(Math.random()*10) + 1;
        int n = (int) (Math.random()*10) + 1;
        // 0<=x<10 : 0~9 => 1~10
        return n;
    }

    String randomColor(){
        /*
            실수 0.0 ~ 1.0 미만의 난수를 발생시켜서
            난수값이 0.66이상이면 "빨강"을, 0.33이상이면 "노랑"을
            그 이외에는 "파랑"을 리턴하는 메서드를 구성하세요.
        */

        double d = Math.random();
        if(d > 0.66){
            return "빨강";
        }else if(d > 0.33){
            return "노랑";
        }
        else{
            return "초록";
        }
    }
}>

<!-- Scriptlet: 페이지 요청이 발생할 때 1번의 요청마다 실행해야할 로직을 작성. --%>
String str = "WEB";
int each = 0;
total++;
each++;

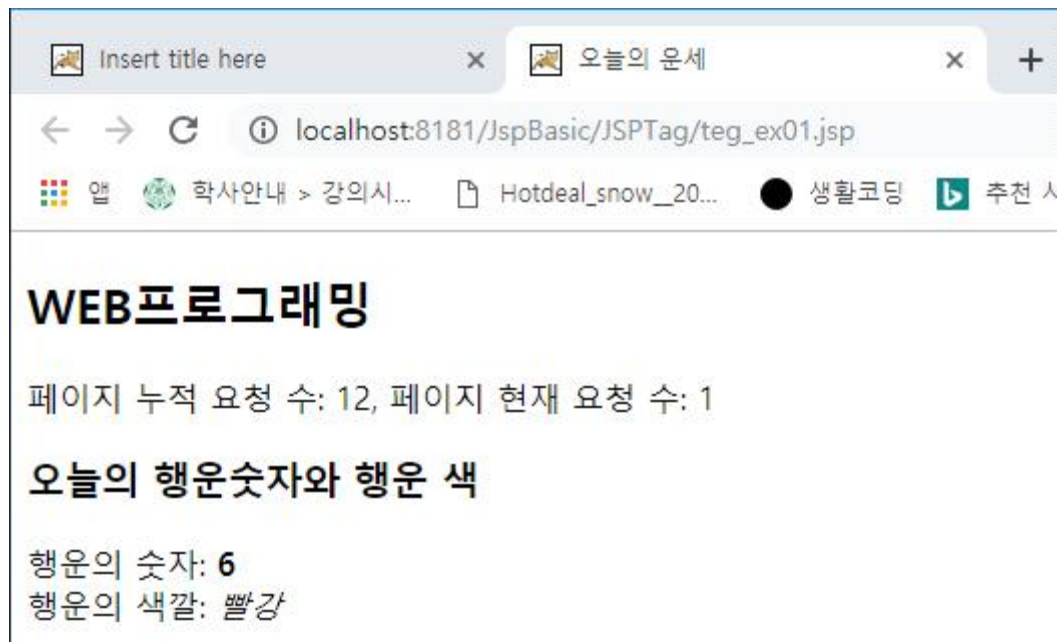
int number = randomNumber();
String color = randomColor();

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>오늘의 운세</title>
</head>
<body>
    <h2><%= str %>프로그래밍</h2>
    <!-- p태그는 문단을 나눌 때 사용하는 태그입니다. -->
    <p>
        <!--
            total변수와 each변수의 선언위치에 따라
            페이지 재 요청시(F5)의 변화를 확인해보세요.
        --%>
        페이지 누적 요청 수: <%= total %>,
        페이지 현재 요청 수: <%= each %> <br>
    </p>
    <p>
        <h3>오늘의 행운숫자와 행운 색</h3>
        행운의 숫자: <b><%=number %></b> <br>
        행운의 색깔: <i><%=color %></i>
    </p>
</body>
</html>
```

```

<%--
ranI = <%= randomNumber() %> <br>
ranS = <%= randomColor() %> <br>
--%>
</body>
</html>

```



## # tag\_ex02.jsp

```
<%@page import="java.util.Random"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>

<%! int total: %>
<%
    total++;
    Random r = new Random(); // Random + [Ctrl + Space] 자동완성
    int num = r.nextInt(8) + 2; // 0<=x<8 => 2<=x<10
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
페이지 누적 요청 수: <%= total %> <br>
매 10번째 방문자에게는 기프티콘을 드립니다. <br>
    <% if(total%10 ==0){ %><br>
        당첨되었습니다!!
    <%} %>

<%--
    // out.println()도 좋지만, 나중에 html 코드 입히기에는 위의 방식이 좋음!
    <%
        if(total%10 ==0){
            out.println("당첨되었습니다!! <br>");
        }
    %>
--%>

<br>
<hr>
<h2>랜덤 구구단</h2>
<br>

이번에 나온 구구단은 <%=num %>단 입니다.
<br><br>
    <% for(int i=1; i<=9; i++){ %>
        <%= num %> × <%= i %> = <%= num * i %> <br>
    <% } %>

<%--
    <%
        for(int hang=1; hang<10; hang++){
            out.println(<u>num</u>+ " x "+hang+ " = " + (<u>num</u>*hang) + "<br>");
        }
    %>
--%>

</body>
</html>
```





페이지 누적 요청 수: 4  
매 10번째 방문자에게는 기프티콘을 드립니다.

---

## 랜덤 구구단

이번에 나온 구구단은 7단 입니다.

7 × 1 = 7  
7 × 2 = 14  
7 × 3 = 21  
7 × 4 = 28  
7 × 5 = 35  
7 × 6 = 42  
7 × 7 = 49  
7 × 8 = 56  
7 × 9 = 63

# teg\_ex03.jsp

```
<%@page import="java.util.*"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>

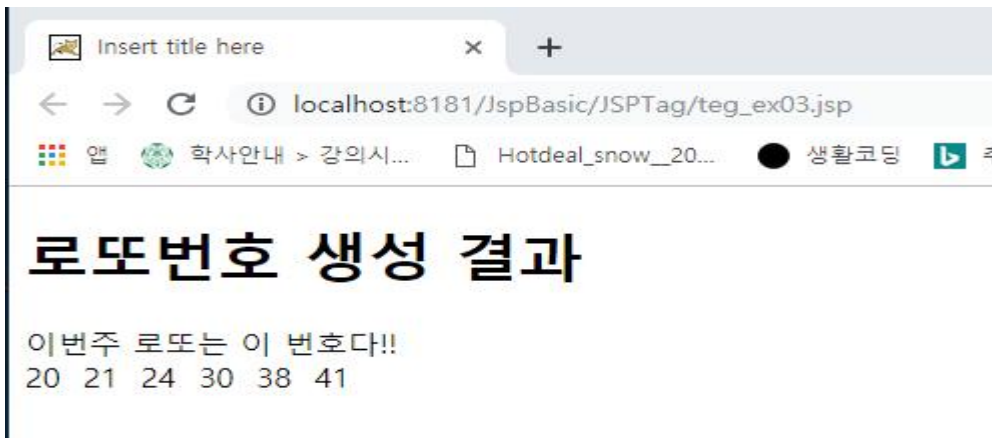
<%
    // TreeSet은 기존의 HashSet에 정렬기능과 검색기능을 강화한 클래스입니다.
    // 자동 오름차정렬을 지원합니다.

    // Set + [Ctrl + Space]
    //Set<Integer> lotto = new TreeSet<Integer>(); // ver 7.
    Set<Integer> lotto = new TreeSet(); // ver 7. 이상

    /*
        1. 1~45의 정수 난수를 발생시키세요.
        2. 무한루프를 사용하여 TreeSet에 해당 정수를 저장하세요.
        3. TreeSet내부에 6개의 정수가 저장되면 반복문을 탈출하세요.
        4. body태그안에 표현식으로 로또번호를 출력하세요.
        toString() 사용
    */

    while(true){ // 중복숫자를 자동으로 걸러주기 때문에 단순히 6번만 돌리면 안됨!
        Random rnd = new Random();
        int rNum = rnd.nextInt(45) + 1; // 0~45 => 1~46
        // Integer rr = new Integer(rNum); // 위와 같은 표현
        lotto.add(rNum); // autoboxing 기능(자동으로 integer로 인식)
        if(lotto.size() == 6){
            break;
        }
    }

%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h1>로또번호 생성 결과</h1>
    <p>
        이번주 로또는 이 번호다!! <br>
        <%
            for(Integer num : lotto){
                out.println(num + "&nbsp;");
                Thread.sleep(700); // CPU를 잠시 멈추는 메서드
                out.flush(); // 브라우저의 출력 버퍼를 비우는 메서드.(데이터 버퍼 40% 여도
전송)
            }
        %>
    <%= lotto.toString() %>
    </p>
</body>
</html>
```



### cf. Collection (Java Review)

1) 정의 : 모음 데이터 관리

2) 종류 :

① List : ArrayList, LinkedList

: 중복 객체 저장 가능(ArrayList)

```
List<String> list = new ArrayList<>();
```

```
list.add(2, 객체);
```

```
list.size()
```

```
list.remove(2)
```

```
list.get(2)
```

```
...
```

② Set : HashSet, TreeSet

: 중복 개체 저장불가(HashSet) -> 자동 오름차 정렬 기능 지원(TreeSet)

```
Set<> O = new 클래스생성자;
```

③ Map : HashMap

: Key, Value

```
Map<String, Integer> map = new HashMap<>();
```

```
map.put(key1, val1);
```

```
map.remove(key1);
```

```
map.get(key); # key에 해당하는 value get!
```

```
map.keySet(); # key들만 썩 꺼냄
```

## cf. Autoboxing

=> 참고링크: <http://hyeonstorage.tistory.com/168>

JDK 1.5 버전 이후에는 자동으로 Boxing과 UnBoxing을 처리하도록 AutoBoxing 과 AutoUnBoxing을 제공한다.

- AutoBoxing

```
Integer obj = 61;
```

숫자 61을 Integer 객체에 넣기 위해서는(Boxing) new Integer(61) 과 같이 객체를 생성해야 하지만, 위와 같이 대입하면 AutoBoxing이 자동으로 진행된다.

- AutoUnBoxing

```
Integer obj2 = new Integer(69);
```

```
int num1 = obj2;
```

Integer 객체에 있는 int 값을 가져오기 위해서는(UnBoxing) obj2.intValue() 메소드를 사용하여 가져와야 하지만, 위와 같이 int 형 변수에 Integer 객체를 대입하면 자동으로 UnBoxing이 진행된다.

## # tag\_ex04.jsp

```
<%@page import="java.util.*"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%!
    // scriptlet (전역변수 설정)
    List<String> party = new ArrayList<>(); //java.util.*

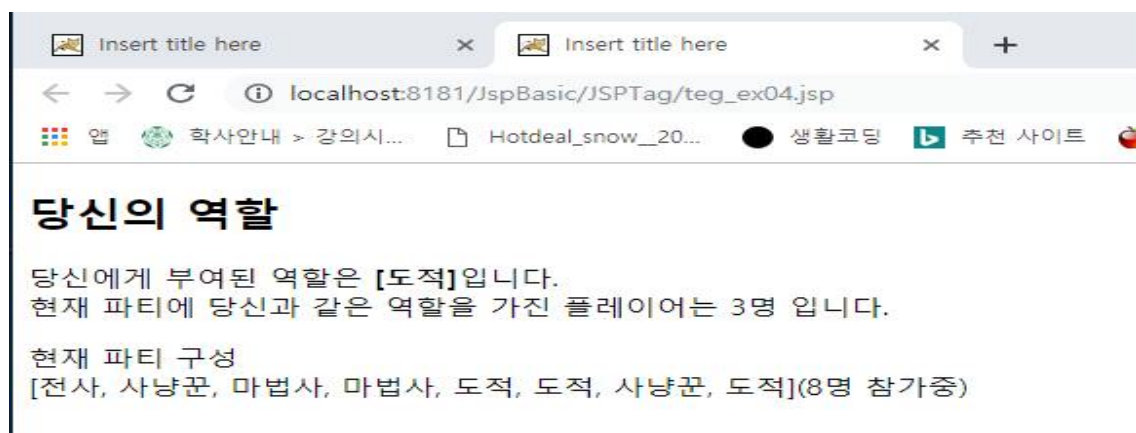
    String[] jobs = {"전사", "도적", "사냥꾼", "마법사", "사제"};
    int r = (int)(Math.random() * jobs.length);
    String job = jobs[r];

    party.add(job);

    // 저장된 list에 현재 선택된 직업이 몇 개 있는지 찾는 작업.
    int cnt = 0;

    // for문으로 list를 순회해서 현재 선택된 직업의 문자열과 리스트 내부에
    // 같은 문자열이 발견될 때마다 cnt수를 1 올립니다.
    for(String e : party){
        if(e.equals(job)){
            cnt++;
        }
    }

    /*
    for(int i=0; i<party.size(); i++){
        String listVal = party.get(i);
        if(listVal.equals(job)){
            cnt++;
        }
    }
    */
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h2>당신의 역할</h2>
    <p>
        당신에게 부여된 역할은 <b>[<%=job %>]</b>입니다. <br>
        현재 파티에 당신과 같은 역할을 가진 플레이어는 <%=cnt %>명 입니다.
    </p>
    <p>
        현재 파티 구성<br>
        <%=party.toString() %>(<%=party.size() %>명 참가중)
    </p>
%>
    if(party.size() == 10){
        party.clear();
    }
%>
</body>
</html>
```

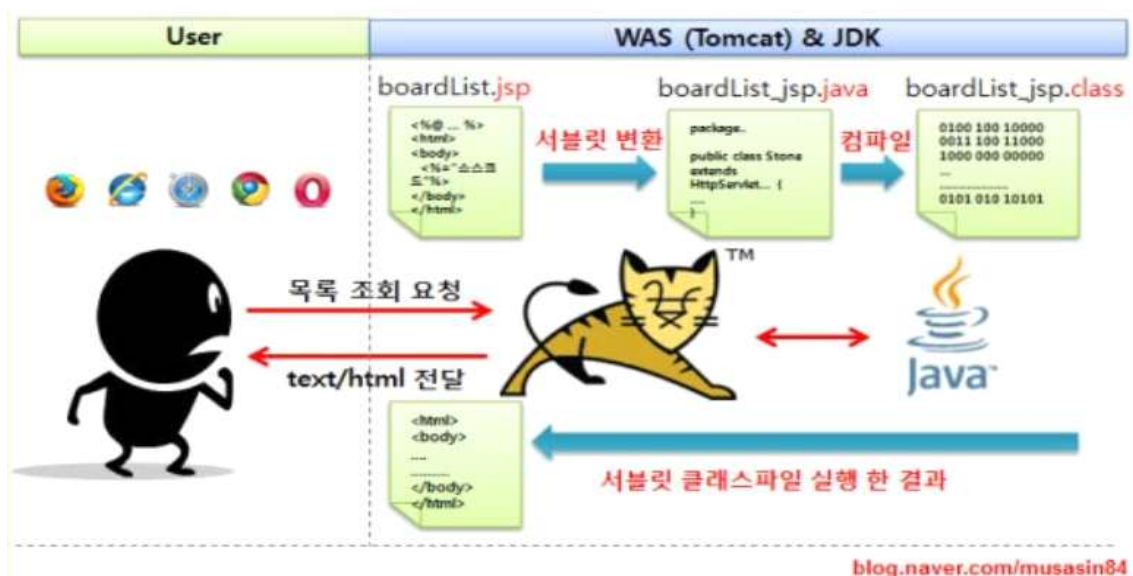
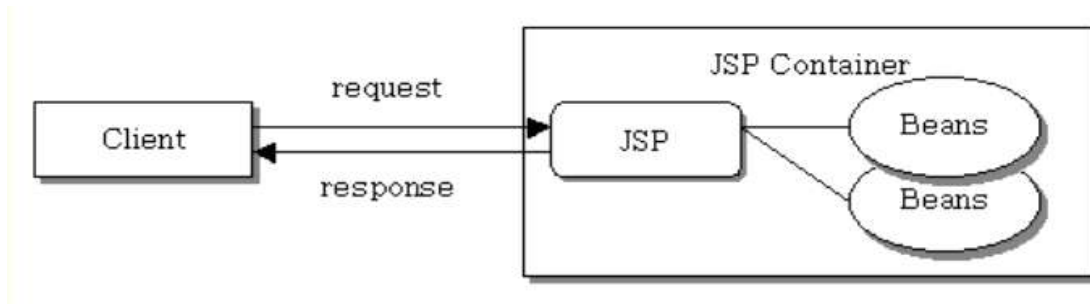
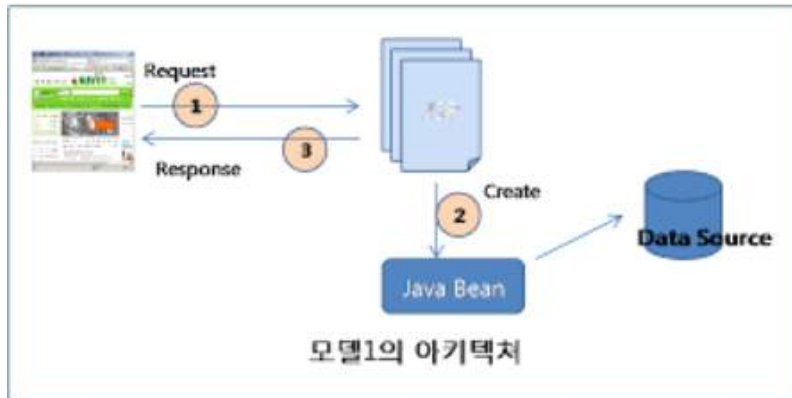


## 8) JSP 아키텍처

: .jsp파일을 실행(요청)하면 웹 서버에서 우선 jsp파일을 java코드로 변환합니다. 그 이후 그 파일을 컴파일하고 html로 응답합니다.

ex) helloworld.jsp -> helloworld\_jsp.java(서버에서 servlet화) -> helloworld\_jsp.class(서블릿 클래스 컴파일)

실행시 다시 html 코드로 변환



[blog.naver.com/musasin84](http://blog.naver.com/musasin84)



## 4. JSP - 내장객체 (1. request)

### 1) 개념

#### ★ JSP 내장 객체

- 개발자가 JSP 파일 내에 객체를 생성하지 않고 바로 사용할 수 있는 객체가 내장 객체입니다.
- JSP에서 제공되는 내장객체는 JSP 컨테이너에 의해 서블릿으로 변환될 때 자동으로 객체가 생성됩니다.

#### ★ JSP 내장 객체의 종류

- ★1. request `javax.servlet.http.HttpServletRequest`
- ★2. response `javax.servlet.http.HttpServletResponse`
- ★3. out `javax.servlet.jsp.JspWriter`
- 4. session `javax.servlet.http.HttpSession`
- 5. application `javax.servlet.ServletContext`
- 6. pageContext `javax.servlet.jsp.PageContext`
- 7. page `javax.servlet.jsp.HttpJspPage`
- 8. config `javax.servlet.ServletConfig`
- 9. exception `java.lang.Throwable`

#### ★ request 객체의 이해

- 웹 브라우저를 통해 서버에 어떤 정보를 요청하는 것을 request라고 합니다.
- 이러한 요청 정보가 담기고 관리되는 곳이 request객체입니다.
- request 객체가 제공하는 기능.
  - 1. 클라이언트(웹 브라우저)와 관련된 정보 읽기 기능.
  - 2. 서버와 관련된 정보 읽기 기능.
- ★3. 클라이언트가 전송한 요청 파라미터 읽기 기능.
  - 4. 클라이언트가 전송한 쿠키 읽기 기능.
- request 객체 관련 주요 메서드
  - 1. `getContextPath():String` - 웹 어플리케이션의 컨텍스트 루트의 경로를 얻습니다.
  - 2. `getMethod():String` - 웹 브라우저가 정보를 전송할 때 사용한 요청 방식을 구합니다.(get, post)
  - 3. `getServerName():String` - 연결할 때 사용한 서버 이름을 구합니다.
  - 4. `getServerPort():int` - 서버가 실행중인 포트 번호를 구합니다.
  - 5. `getRequestURL():String` - 요청 URL을 얻습니다.
  - 6. `getRequestURI():String` - 요청 URI를 얻습니다.
  - 7. `getRemoteAddr():String` - 웹 서버에 연결한 클라이언트의 IP주소를 구합니다.
  - 8. `getProtocol():String` - 해당 프로토콜을 얻습니다.
- 9. `getParameter(String name):String`
  - 이름이 name인 파라미터 값을 구합니다. 존재하지 않을 경우 null을 반환합니다.
- 10. `getParameterValues(String name):String[]`
  - 이름이 name인 모든 파라미터 값들을 배열로 구합니다. 존재하지 않을 경우 null을 반환합니다.
- 11. `getParameterNames():java.util.Enumeration`
  - 웹 브라우저 전송한 파라미터의 이름 목록을 구합니다.
- 12. `getParameterMap():java.util.Map`
  - 웹 브라우저가 전송한 파라미터의 맵을 구합니다. 맵은 <파라미터 이름, 파라미터 값> 쌍으로 구성됩니다.

## \* HttpRequest 방식

### - GET 방식

1. 서버에 데이터를 요청하는 용도.
2. 전송하는 데이터가 주소에 묻어서 감.
3. 전송했던 데이터는 브라우저의 히스토리에 접속했던 주소와 함께 남아 있어 보안성에 취약함.
4. 게시판 글 조회나 검색 같이 서버의 정보를 가져올 필요성이 있을 때 사용함.
5. 전송할 수 있는 최대 크기는 브라우저별로 다르지만 크기가 정해져있음.
6. HTML form태그가 반드시 필요하지는 않습니다.

### - POST 방식

1. 서버에 데이터를 전송하는 용도.
2. 전송되는 데이터가 URL에 묻어나가지 않고 전송 객체의 메시지 바디를 통해 전달됨.
3. 브라우저에 전달되는 데이터가 남지 않기 때문에 보안성에 강함.
4. 비밀번호나 주민번호 등 private한 데이터를 서버에 전송해야 할 때 사용함.
5. 반드시 HTML form태그가 필요합니다.
6. 데이터 양의 제한이 없기 때문에 대량의 데이터를 전송할 수 있습니다.

## \* GET/ POST 방식 브라우저 한글처리

- 톰캣서버의 기본 문자처리 방식은 IOS-8859-1 방식입니다.
- 따라서 개발자가 별도의 한글 인코딩을 하지 않으면 서버로 전송된 데이터의 한글들이 깨져보이는 현상이 발생합니다.

### 1. GET 방식의 한글처리

- server.xml 파일 수정
- <connector> 에 속성 값으로 URIEncoding="utf-8"

### 2. POST 방식의 한글처리

- post 방식을 처리하는 메서드에

request.setCharacterEncoding("utf-8");

## # server.xml

```
req_form_post.jsp  req_take_post.jsp  server.xml
55
56  <!-- A "Connector" represents an endpoint by which requests are received
57       and responses are returned. Documentation at :
58       Java HTTP Connector: /docs/config/http.html
59       Java AJP  Connector: /docs/config/ajp.html
60       APR (HTTP/AJP) Connector: /docs/apr.html
61       Define a non-SSL/TLS HTTP/1.1 Connector on port 8080
62  -->
63  <Connector connectionTimeout="20000" port="8181" protocol="HTTP/1.1" redirectPort="8443"/>
64  <!-- A "Connector" using the shared thread pool-->
65  <!--
66  <Connector executor="tomcatThreadPool"
67              port="8080" protocol="HTTP/1.1"
68              connectionTimeout="20000"
69              uriEncoding="utf-8" />
70  -->
71  </-->
```

★ out 객체의 이해

- JSP 페이지가 생성하는 모든 내용은 out 기본 객체를 통해 전송됩니다.
- JSP 페이지 내에서 사용하는 비스크립트 요소들(HTML코드와 텍스트)이 out 객체에 전달됩니다.
- 값을 출력하는 표현식(expression)의 결과값도 out객체에 전달됩니다.

★ Servlet 특징

1. 동적 웹어플리케이션 컴포넌트
2. .java 확장자
3. 클라이언트의 요청에 동적으로 작동하고, 응답은 html을 이용.
4. java thread를 이용하여 동작.
5. MVC패턴에서 Controller로 이용됨.

★ URL-Mapping

- URL 매핑을 하지 않으면 URL주소가 너무 길어지고, 경로가 노출되어 보안에 위험이 생기기 때문에 URL 매핑을 사용하여 그 문제들을 해결합니다.

- http://localhost:8181/JSPBasic/servlet/kr.co.koo.HelloWorld

---->> http://localhost:8181/JSPBasic/HelloWorld

- 사용 방법

1. 아노테이션 이용, 클래스 선언부 바로 위에 작성.  
ex) @WebServlet("/HelloWorld")

2. web.xml 설정파일 수정.

ex)

```
<servlet>
    <servlet-name>helloworld</servlet-name>
    <servlet-class>kr.co.koo.HelloWorld</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>helloworld</servlet-name>
    <url-pattern>/HWorld</url-pattern>
</servlet-mapping>
```

★ Servlet 작동 순서

- 클라이언트에서 요청(request)이 들어오면 서버에서는 servlet 컨테이너를 만들고, 요청이 있을때 thread와 Servlet 객체가 생성됩니다.

★ Servlet의 생명주기(LifeCycle)

- Servlet의 장점은 빠른 응답 속도입니다.
- Servlet은 최초 요청시에 객체가 만들어져 메모리에 로딩되고, 이후 추가 요청시에는 기존의 객체를 재활용하게 됩니다. 따라서 동작속도가 매우 빠릅니다.

1. Servlet 객체를 생성 (최초 한번)
2. Init() 메서드 호출(최초 한번)
3. doGet(), doPost(), service() 호출 (요청시 매번)
4. destroy() 호출 (마지막 한번) - 자원이 해제될 시 호출(Servlet코드를 수정, 서버 재가동할 시)

★ 웹 어플리케이션 생명주기 (ServletContextListener)

- 웹 어플리케이션에는 프로그램의 생명주기를 감시하는 리스너가 있습니다.
  - 리스너의 해당 메서드가 웹 어플리케이션의 시작과 종료시에 호출됩니다.
1. 시작시에는 contextInitialized()
  2. 종료시에는 contextDestroyed()

★ 서블릿 초기화 파라미터 (ServletConfig)

- 특정한 서블릿이 생성될 때 초기에 필요한 데이터들이 있습니다.
- 이러한 데이터들을 초기화 파라미터라고 하며, 아노테이션으로 지정하는 방법과, web.xml파일에 기술하는 방법이 있습니다.

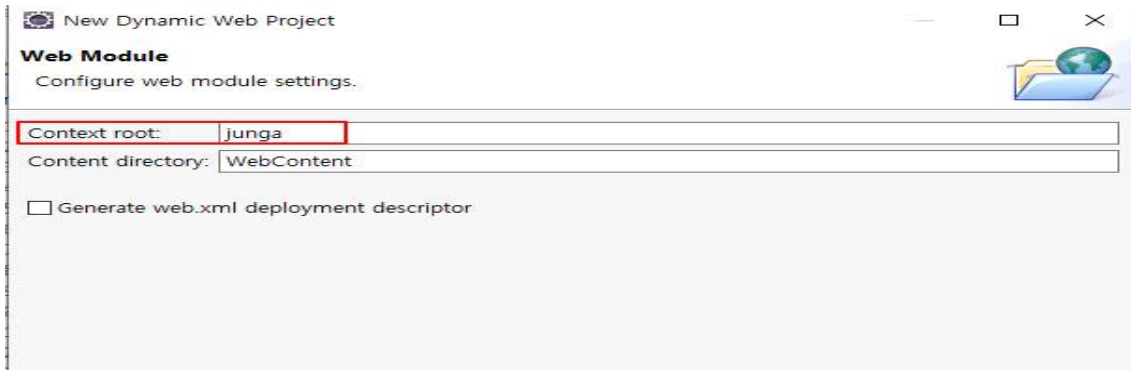
★ 데이터 공유(ServletContext)

- 여러 서블릿에서 특정 데이터를 공유해야 할 경우 Context Parameter를 이용하여 web.xml파일에 데이터를 기술하고, 여러 서블릿에서 공유하면서 사용할 수 있습니다.

## 2) sample (request)

[작업순서]

1. 프로젝트 생성
  - 프로젝트명: JspObj
  - context root: 내 이름
2. WebContent 하위에 폴더 생성
  - 폴더명: request
3. WebContent/request 하위에 jsp 파일 생성
  - req\_info.jsp
4. BuildPath (tomcat jar 파일 추가)

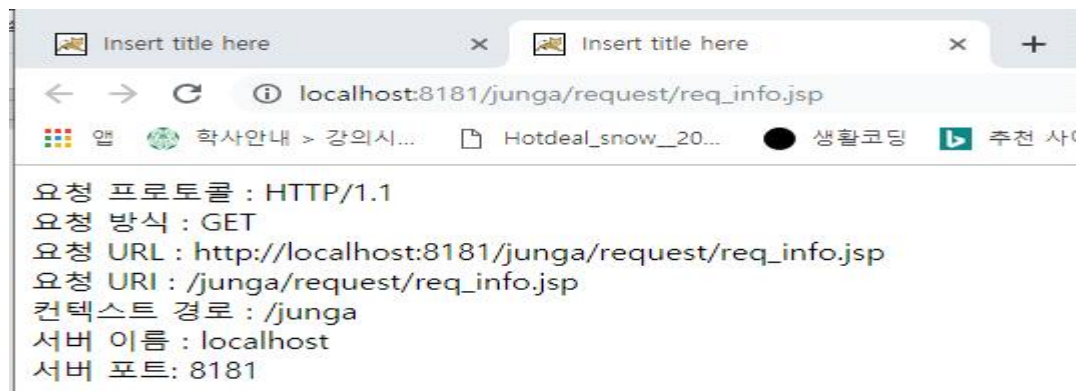


# req\_info.jsp

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

<%
    String protocol = request.getProtocol();
    String method = request.getMethod();
    StringBuffer reqUrl = request.getRequestURL();
    String reqUri = request.getRequestURI();
    String conPath = request.getContextPath();
    String serverName = request.getServerName();
    int serverPort = request.getServerPort();
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    요청 프로토콜 : <%= protocol %> <br>
    요청 방식 : <%= method %> <br>
    요청 URL : <%= reqUrl %> <br>
    요청 URI : <%= reqUri %> <br>
    컨텍스트 경로 : <%= conPath %> <br>
    서버 이름 : <%= serverName %> <br>
    서버 포트: <%= serverPort %>

</body>
</html>
```



[localhost:8181/junga/request/req\\_info.jsp](http://localhost:8181/junga/request/req_info.jsp)

도메인 /context root/파일명

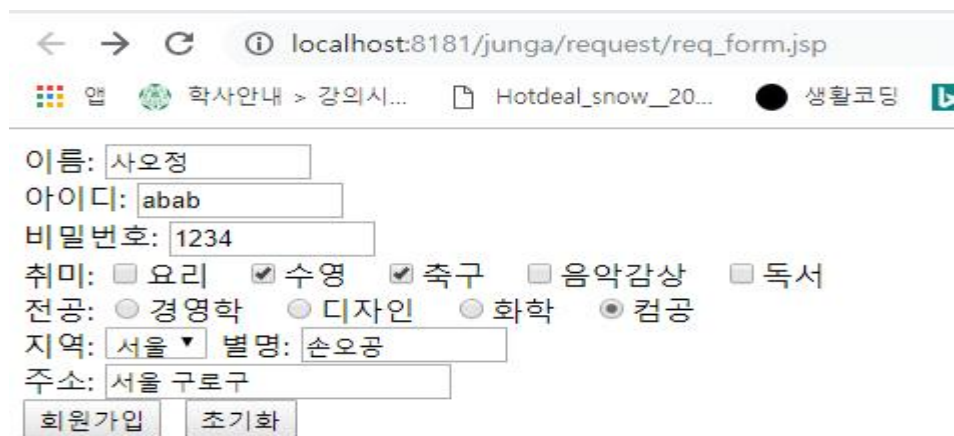
```
# req_form.jsp
```

```
<%@ 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>  
  
    <!--  
        form에 작성된 데이터들을 서버로 넘길 때 해당 데이터를 받아서 처리할 페이지의  
        경로를 form태그의 action속성에 적어줍니다.  
    --%>  
    <form action="req_parameter.jsp">  
  
    <!--태그의 name속성은 서버로 전송될 파라미터 변수의 이름이 됩니다. --%>  
        이름: <input type="text" name="name" size="10"><br>  
        아이디: <input type="text" name="id" size="10"><br>  
        비밀번호: <input type="text" name="pw" size="10"><br>  
  
        취미:  
        <input type="checkbox" name="hobby" value="요리">요리 &nbsp;   <br>  
        <input type="checkbox" name="hobby" value="수영">수영 &nbsp;   <br>  
        <input type="checkbox" name="hobby" value="축구">축구 &nbsp;   <br>  
        <input type="checkbox" name="hobby" value="음악감상">음악감상 &nbsp;   <br>  
        <input type="checkbox" name="hobby" value="독서">독서 &nbsp;   <br>  
  
        전공:  
        <input type="radio" name="major" value="경영학">경영학 &nbsp;   <br>  
        <input type="radio" name="major" value="디자인">디자인 &nbsp;   <br>  
        <input type="radio" name="major" value="화학">화학 &nbsp;   <br>  
        <input type="radio" name="major" value="컴공">컴공 &nbsp;   <br>  
  
        지역:  
        <select name="region">  
            <option>서울</option>  
            <option>경기</option>  
            <option>인천</option>  
            <option>제주</option>  
        </select>  
    <!--  
        회원의 별명(nickname)과 주소(address)를 입력받아 서버로 전송하여  
        req_parameter.jsp 페이지에서 입력한 별명과 주소를 출력하세요.  
    --%>  
        별명: <input type="text" name="nickname" size="10"><br>  
        주소: <input type="text" name="address" size="20"><br>  
        <input type="submit" value="회원가입">&nbsp;   <br>  
        <input type="reset" value="초기화">  
  
    </form>  
  
</body>  
</html>
```

## # req\_parameter.jsp

```
<%@page import="java.util.*"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    // getParameter의 매개값으로 내가 읽어오고 싶은 파라미터 변수의 이름을 씁니다.
    String userId = request.getParameter("id");
    String userPw = request.getParameter("pw");
    String userName = request.getParameter("name");
    String major = request.getParameter("major");
    String region = request.getParameter("region");
    String[] hobbies = request.getParameterValues("hobby");
    String nick = request.getParameter("nickname");
    String addr = request.getParameter("address");
    System.out.println(userId);
    System.out.println(major);
%>

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    이름: <%= userName %> <br>
    Id: <%= userId %> <br>
    Pw: <%= userPw %> <br>
    취미: <%= Arrays.toString(hobbys) %> <br>
    전공: <%= major %> <br>
    지역: <%= region %> <br>
    별명: <%= nick %> <br>
    주소: <%= addr %> <br>
</body>
</html>
```



이름: 사오정  
 아이디: abab  
 비밀번호: 1234  
 취미: ☐ 요리 ☒ 수영 ☒ 축구 ☐ 음악감상 ☐ 독서  
 전공: ☐ 경영학 ☐ 디자인 ☐ 화학 ☒ 컴공  
 지역: 서울 ▼ 별명: 손오공  
 주소: 서울 구로구  
 회원가입 초기화



이름: 사오정  
 Id: abab  
 Pw: 1234  
 취미: [수영, 축구]  
 전공: 컴공  
 지역: 서울  
 별명: 손오공  
 주소: 서울 구로구



## # req\_bmi\_form.jsp

```
<%@ 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>
<!--
* 1. form태그를 활용하여 사용자의 키와 몸무게를 입력받으세요.
2. 입력받은 데이터를 서버로 전송하여 req_bmi.jsp 페이지에서
   bmi지수를 계산처리하세요.
--%>
    <form action="req_bmi01.jsp">
        신장: <input type="text" name="cm" size="10">cm<br>
        체중: <input type="text" name="kg" size="10">kg<br><br>
        <input type="submit" value="확인"> &nbsp; <input type="submit" value="초기화">
    </form>
</body>
</html>
```

## # req\_bmi01.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    String strCm = request.getParameter("cm");
    String strKg = request.getParameter("kg");

    /*
     문자열 내부에 들어있는 데이터를 기본데이터 타입으로 변환하는 방법
     : 포장 클래스에 들어있는 parse+기본타입이름() 메서드 활용.
     ex) String -> int
           Integer.parseInt(문자열);
           String -> double
           Double.parseDouble(문자열);
    */
    // String은 계산 불가! 데이터 변환
    double cm = Double.parseDouble(strCm);
    double kg = Double.parseDouble(strKg);

    // bmi지수 계산 공식[체중 / 신장(m) * 신장(m)]
    double bmi = kg / (cm/100 * cm/100);
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h2>체질량 지수(BMI) 계산</h2>
    <hr>
    <p>
        - 신장: <%= strCm%>cm <br>
        - 체중: <%= strKg%>kg <br>
    </p>
    <p>
        BMI 지수: <b><%=bmi %></b> <br>
        <!--
            * bmi지수가 23을 초과한다면 "당신은 과체중 입니다"를 출력.
              16.5미만이라면 "당신은 저체중 입니다"를 출력.
              나머지는 "당신은 정상체중 입니다"를 출력하세요.
        --%>
        <%if(bmi > 23){ %>
            당신은 과체중 입니다
        <%}else if(bmi < 16.5){ %>
            당신은 저체중 입니다
        <%}else { %>
            당신은 정상체중 입니다
        <%} %>
    </p>
</body>
</html>
```

Insert title here x | Insert title here

localhost:8181/junga/request/req\_bmi\_form.jsp

앱 학사안내 > 강의시... Hotdeal\_snow\_20... 생활코딩

신장: 180 cm  
체중: 65 kg

확인 초기화

localhost:8181/junga/request/req\_bmi01.jsp?cm=180&kg=65

앱 학사안내 > 강의시... Hotdeal\_snow\_20... 생활코딩 추천 사이트

## 체질량 지수(BMI) 계산

---

- 신장: 180cm  
- 체중: 65kg

BMI 지수: **20.061728395061728**  
당신은 정상체중 입니다

## # req\_album01.jsp

```
<%@ 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>
    <div align="center">
        <form action="req_album02.jsp">
            <table border="1" width="700">
                <tr>
                    <td></td>
                    <td>앨범 커버</td>
                    <td>가수</td>
                    <td>앨범 제목</td>
                    <td>발매일</td>
                </tr>
                <tr>
                    <td align="center"><input type="radio" name="title" value="lovelyz4"></td>
                    <td align="center"></td>
                    <td>러블리즈</td>
                    <td>러블리즈 (Lovelyz)-미니앨범 4집 : [治癒] (치유)</td>
                    <td>2018-04-23</td>
                </tr>
                <tr>
                    <td align="center"><input type="radio" name="title" value="secret"></td>
                    <td align="center"></td>
                    <td>오마이걸</td>
                    <td>비밀정원</td>
                    <td>2018-01-09</td>
                </tr>
                <tr>
                    <td align="center" colspan="5">
                        <!-- td의 colspan속성은 열을 병합하는 기능입니다. -->
                        <td colspan="5">
                            <input type="submit" value="확인"/>
                        </td>
                    </td>
                </tr>
            </table>
        </form>
    </div>
</body>
</html>
```

## # req\_album02.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    String title = request.getParameter("title");
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <%if(title.equals("lovelyz4")) {%>
    <div align="center">
        <h2>선택하신 앨범 정보</h2>
        <p>당신이 선택한 앨범은 러블리즈 미니앨범 4집 '치유'입니다.
        </p>
        <hr>
        <h3>타이틀 곡 MV</h3>
        <%--유투브 동영상 광고 끝나고 오버 소스코드 복사 --%>
        <%--src="" 뒤 ?rel=0:apm;autoplay=1 복붙 --%>
        <iframe width="800" height="600" src="https://www.youtube.com/embed/40Z9-kr504w?rel=0:apm:autoplay=1"
frameborder="0" allow="autoplay; encrypted-media" allowfullscreen></iframe>
    </div>
    <%}else if(title.equals("secret")) {%>
    <div align="center">
        <h2>선택하신 앨범 정보</h2>
        <p>당신이 선택한 앨범은 오마이걸의 '비밀정원' 입니다.
        </p>
        <hr>
        <h3>타이틀 곡 MV</h3>
        <iframe width="800" height="600" src="https://www.youtube.com/embed/QIN5_tIRiyY?rel=0:apm:autoplay=1"
frameborder="0" allow="autoplay; encrypted-media" allowfullscreen></iframe>
    </div>
    <%} %>
</body>
</html>
```

Insert title here				
← → ↻ ⓘ localhost:8181/junga/request/req_album01.jsp				
열 확장안내 > 강의서... Hotdeal_snow_20... 생활코딩 추천 사이트 애플파일 IE에서 가져온 북마크 신일SW인력을 위... KG아이티				
앨범 커버	가수	앨범 제목	발매일	
	러블리즈	러블리즈 (Lovelyz)-미니앨범 4집 : [治愈] (치유)	2018-04-23	
	오마이걸	비밀정원	2018-01-09	
확인				

Insert title here

localhost:8181/junga/request/req\_album02.jsp?title=lovelyz4

선택하신 앨범 정보


당신이 선택한 앨범은 러블리즈 미니앨범 4집 '치유'입니다.

타이틀 곡 MV

[MV] Lovelyz(러블리즈) \_ That day(그날의 너)

나중에 시청하기 공유 정보

12



Music  
Brand  
Unit

Mnet 2018.04.18

0:01 / 3:13

1theK YouTube

각 링크 클릭시 넘어가는 파라미터 값

[http://localhost:8181/junga/request/req\\_album02.jsp?title=lovelyz4](http://localhost:8181/junga/request/req_album02.jsp?title=lovelyz4)

[http://localhost:8181/junga/request/req\\_album02.jsp?title=secret](http://localhost:8181/junga/request/req_album02.jsp?title=secret)

## # req\_album01.jsp 코드 변경

=> 이미지, 제목에 parameter값 넘겨서 링크 처리

```
<%@ 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>
    <div align="center">
        <form action="req_album02.jsp">
            <table border="1" width="700">
                <tr>
                    <td>앨범 커버</td>
                    <td>가수</td>
                    <td>앨범 제목</td>
                    <td>발매일</td>
                </tr>
                <tr>
                    <td align="center">
                        <!-- <input type="radio" name="title" value="lovelyz4"> -->
                        <a href="req_album02.jsp?title=lovelyz4">
                            
                        </a>
                    </td>
                    <td>러블리즈</td>
                    <td>
                        <a href="req_album02.jsp?title=lovelyz4">
                            러블리즈 (Lovelyz)-미니앨범 4집 : [治癒] (치유)
                        </a>
                    </td>
                    <td>2018-04-23</td>
                </tr>
                <tr>
                    <td align="center">
                        <!-- <input type="radio" name="title" value="secret"> -->
                        <a href="req_album02.jsp?title=secret">
                            
                        </a>
                    </td>
                    <td>오마이걸</td>
                    <td>
                        <a href="req_album02.jsp?title=secret">
                            비밀정원
                        </a>
                    </td>
                    <td>2018-01-09</td>
                </tr>
                <tr>
                    <td colspan="4">
                        <!-- td의 colspan속성은 열을 병합하는 기능입니다. -->
                        <td colspan="5">
                            <input type="submit" value="확인">
                        </td>
                    </tr>
            </table>
        </form>
    </div>
</body>
</html>
```

localhost:8181/junga/request/req\_album01.jsp

> 강의시... Hotdeal\_snow\_\_20... 생활코딩 추천 사이트 애플파일 IE에서 가져온 북마크 신입SW인력을 위... KG아이

앨범 커버	가수	앨범 제목	발매일
	러블리즈	<u>러블리즈 (Lovelyz)-미니앨범 4집 : [治愈](치유)</u>	2018-04-23
	오마이걸	<u>비밀정원</u>	2018-01-09
<input type="button" value="확인"/>			

## # req\_form\_post.jsp

Ex 1> get방식으로 처리한 경우 (default)

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    //request.setCharacterEncoding("utf-8");

    String id = request.getParameter("id");
    String pw = request.getParameter("pw");
    String name = request.getParameter("name");
%>

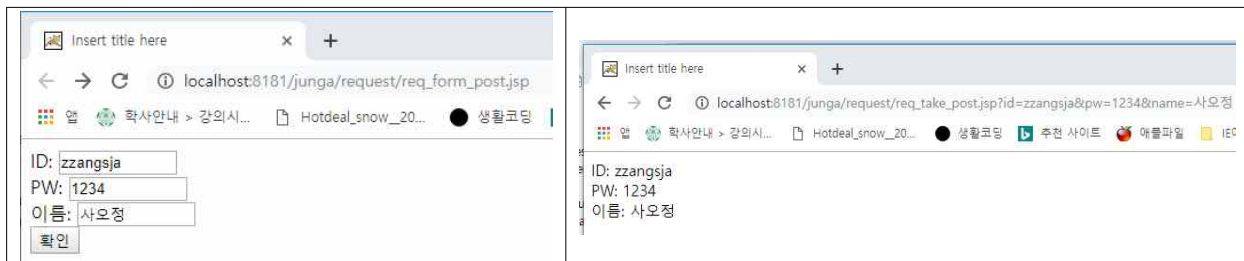
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <form action="req_take_post.jsp">
        ID: <input type="text" name="id" size="10"><br>
        PW: <input type="text" name="pw" size="10"><br>
        이름: <input type="text" name="name" size="10"><br>
        <input type="submit" value="확인">
    </form>
</body>
</html>
```

## # req\_take\_post.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    //request.setCharacterEncoding("utf-8");

    String id = request.getParameter("id");
    String pw = request.getParameter("pw");
    String name = request.getParameter("name");
%>

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    ID: <%=id %><br>
    PW: <%=pw %><br>
    이름: <%=name %><br>
</body>
</html>
```



=> 주요 정보 다 넘어감



## # req\_form\_post.jsp

Ex 2> post방식으로 처리한 경우 (한글 깨짐현상 처리, form태그 사용 필수)

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    request.setCharacterEncoding("utf-8");

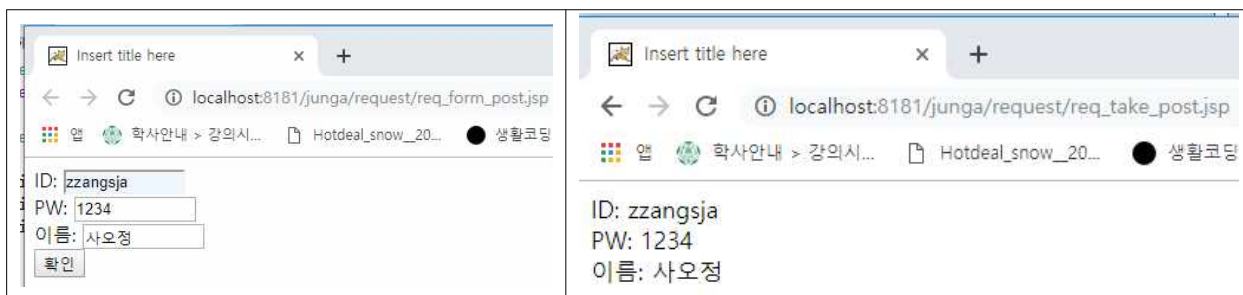
    String id = request.getParameter("id");
    String pw = request.getParameter("pw");
    String name = request.getParameter("name");
%>
```

## # req\_take\_post.jsp

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <!--Get방식과 달리 POST 방식은 반드시 HTML form 태그가 필요 --%>
    <!--POST방식을 사용하려면 form태그의 method라는 속성값을 "post"로 적습니다. --%>
    <form action="req_take_post.jsp" method="post">
        ID: <input type="text" name="id" size="10"><br>
        PW: <input type="text" name="pw" size="10"><br>
        이름: <input type="text" name="name" size="10"><br>
        <input type="submit" value="확인">
    </form>
</body>
</html>

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    request.setCharacterEncoding("utf-8");

    String id = request.getParameter("id");
    String pw = request.getParameter("pw");
    String name = request.getParameter("name");
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    ID: <%=id %><br>
    PW: <%=pw %><br>
    이름: <%=name %><br>
</body>
</html>
```



## 4-2. JSP - 내장객체 (2. response)

### 1) 개념

#### \* response 객체의 이해

- 웹 브라우저의 요청에 응답하는 것을 response라고 합니다.
- 이러한 응답의 정보를 가지고 있는 객체를 response객체라고 합니다.

- response 객체 주요 메서드

1. `getCharacterEncoding()`: 응답할 때의 문자의 인코딩 형태를 구합니다.
2. `addCookie(Cookie c)`: 쿠키를 지정합니다.
3. `sendRedirect(URL)`: 지정한 URL로 이동합니다.

#### \* 상대경로, 절대경로

- 절대경로 : 같은 폴더 내에 있는 jsp 파일의 경우 (같은 폴더 내에 있으면 써도 그만 안 써도 그만)
- 상대경로 : 다른 폴더 내에 있는 jsp 파일의 경우 (다른 폴더로 이동해야 하므로 무조건 써야 함)

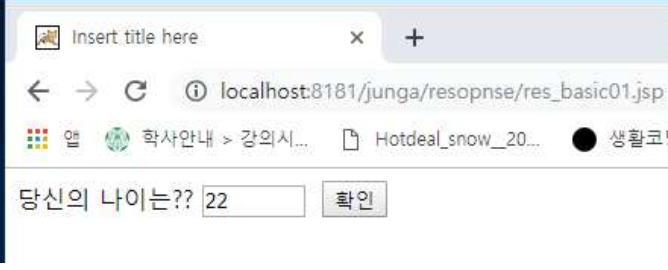
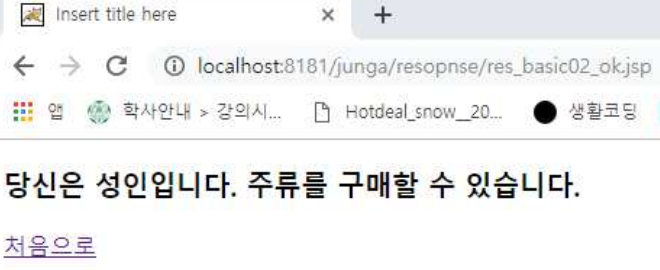
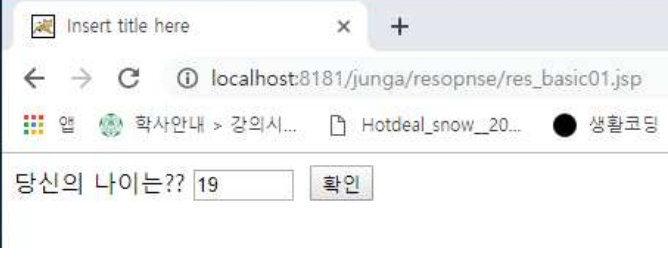
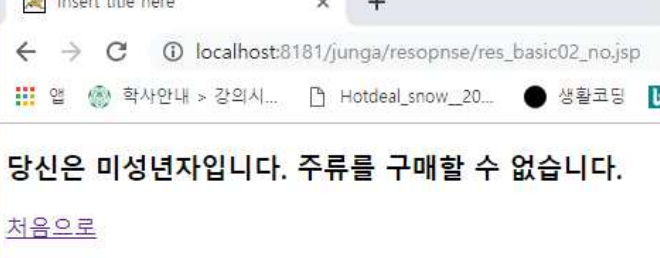
[관련 실습]

=> 링크: [Sample 3>](#)



## # res\_basic02\_no.jsp

```
<%@ 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>
    <h3>당신은 미성년자입니다. 주류를 구매할 수 없습니다.</h3>
    <a href="res_basic01.jsp">처음으로</a>
</body>
</html>
```

sample 2>

# res\_login\_form.jsp 🖱 [Ctrl + F11]

# res\_login\_ok.jsp

# res\_welcome.jsp

# res\_pw\_fail.jsp

# res\_id\_fail.jsp

# res\_login\_form.jsp 🖱 [Ctrl + F11]

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

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<%--
```

스타일 태그 내부에 적용하고 싶은 디자인을 class이름 또는 id이름으로 접근하여 지정합니다.  
그러면 해당 class이름이나 id이름을 가진 모든 태그에 디자인이 일괄 적용됩니다.

클래스 이름 선택시=> .클래스이름  
아이디 선택시=> #아이디이름

```
--%>
```

```
<style>
```

```
.login_btn{
    width:60px;
    height:50px;
    background-color:#CA73FF;
}
```

```
</style>
```

```
<meta charset="UTF-8">
```

```
<title>Insert title here</title>
```

```
</head>
```

```
<body>
```

```
<div align="center">
```

```
<h3>SendRedirect 로그인</h3>
```

```
<hr>
```

```
<form action="/res_login_ok.jsp" method="get">
```

```
<table border="1" cellpadding="0" cellspacing="0">
```

```
<tr>
```

```
<td>
```

```
<input type="text" name="id" size="10" placeholder="아이디">
```

```
</td>
```

```
<td rowspan="2">
```

```
<%--
```

HTML 태그에 디자인요소를 적영하는 언어는 css입니다.  
css스타일 적용방식은

1. 인라인 스타일 시트 - 태그 내부에 스타일을 지정  
2. 내부 스타일 시트 - 페이지 속성으로 스타일을 지정(id:js,

3. 외부 스타일 시트 - 외부 css파일을 페이지에 로딩하여 스타일

class:디자인)

지정 : 페이지 많은 경우 일괄로 적용 위함

height:50px; width:60px:">

```
--%>
```

```
<!-- 인라인 스타일시트 방식.
```

```
<input type="submit" value="로그인" style="background-color: #F6358A;
```

```
-->
```

```
<input type="submit" value="로그인" class="login_btn">
```

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td>
```

```
<input type="password" name="pw" size="10" placeholder="비밀번호">
```

```
</td>
```

```
</tr>
```

```
</table>
```

```
</form>
```

```
</div>
```

```
</body>
```

```
</html>
```

## # res\_login\_ok.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!--
이 페이지에서는 넘어온 아이디값과 비밀번호값을
받아서 처리합니다.
만약 아이디가 'abc1234'이고 비밀번호가 'xyz9876'이라면
1) 아이디가 일치하지 않았을 경우
   'res_id_fail.jsp'로 리다이렉팅해서
   '존재하지 않는 회원입니다'를 브라우저에 출력하세요.
2) 아이디가 일치할 경우 비밀번호도 확인해서 틀렸을 경우
   'res_pw_fail.jsp'로 리다이렉팅해서
   '비밀번호가 틀렸습니다'를 브라우저에 출력하세요.
3) ID와 PW가 전부 일치했을 경우 'res_welcome.jsp'로
   리다이렉팅하여 '회원님 반갑습니다'를 출력하세요.
-->
<%
    String id = request.getParameter("id");
    String pw = request.getParameter("pw");
    if(id.equals("abcd")){
        // 아이디 일치
        // response 내장객체가 지원하는 sendRedirect() 메서드는 매개값으로
        // 페이지의 URL을 지정하면 해당 페이지로 강제이동 시킵니다.

        if(pw.equals("1234")){
            // 비밀번호도 일치
            response.sendRedirect("./res_welcome.jsp");
            // response.sendRedirect("../request/req_album01.jsp");
        }else{
            // 비밀번호 불일치
            response.sendRedirect("./res_pw_fail.jsp");
        }
    }else{
        // 아이디 불일치
        response.sendRedirect("./res_id_fail.jsp");
    }
}%>
```

## # res\_welcome.jsp

```
<%@ 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>
    <p><b>로그인에 성공했습니다. 반갑습니다 회원님 :)</b></p>
    <a href="res_login_form.jsp">로그인 페이지로 이동</a>
</body>
</html>
```

## # res\_pw\_fail.jsp

```
<%@ 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>
    <p style="color:red;"><b>비밀번호가 틀렸습니다.</b></p>
    <a href="res_login_form.jsp">로그인 페이지로 이동</a>
</body>
</html>
```

## # res\_id\_fail.jsp

```
<%@ 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>
    <p style="color:red;">존재하지 않는 회원입니다.</p>
    <a href="res_login_form.jsp">로그인 페이지로 이동</a>
</body>
</html>
```

<p>localhost:8181/junga/resopnse/res_login_form.jsp</p> <p>추천 사이트 &gt; 강의시... Hotdeal_snow_20... 생활코딩 추천 사이트 애플파일 IE에서 가져온 북마크</p> <p><b>SendRedirect 로그인</b></p> <div> <input type="text" value="abcd"/> <input type="password" value="...."/> <input type="button" value="로그인"/> </div>	<p>localhost:8181/junga/resopnse/res_welcome.jsp</p> <p>앱 학사안내 &gt; 강의시... Hotdeal_snow_20... 생활코딩</p> <p><b>로그인에 성공했습니다. 반갑습니다 회원님 :)</b></p> <p><a href="#">로그인 페이지로 이동</a></p>
<p>추천 사이트 애플파일 IE에서 가져온 북마크</p> <p><b>SendRedirect 로그인</b></p> <div> <input type="text" value="abcc"/> <input type="password" value="...."/> <input type="button" value="로그인"/> </div>	<p>Insert title here</p> <p>localhost:8181/junga/resopnse/res_id_fail.jsp</p> <p>앱 학사안내 &gt; 강의시... Hotdeal_snow_20... 생활코딩</p> <p><b>존재하지 않는 회원입니다.</b></p> <p><a href="#">로그인 페이지로 이동</a></p>
<p><b>SendRedirect 로그인</b></p> <div> <input type="text" value="abcd"/> <input type="password" value="...."/> <input type="button" value="로그인"/> </div>	<p>localhost:8181/junga/resopnse/res_pw_fail.jsp</p> <p>앱 학사안내 &gt; 강의시... Hotdeal_snow_20... 생활코딩</p> <p><b>비밀번호가 틀렸습니다.</b></p> <p><a href="#">로그인 페이지로 이동</a></p>

- ★ 필요 jsp 파일 (5개)
- 로그인 메인화면 1
  - 로직처리 1
  - sendRedirect 3

### sample 3> 실습

```
// 링크주소에 물어있는 파라미터 변수를 처리한 후
// 해당 파라미터에 값에 맞게 각각 다른 영상이 나오도록 리다이렉팅하세요.
//1번째 영상 페이지(response폴더에 res_mv01.jsp)
//2번째 영상 페이지(response폴더에 res_mv02.jsp)
```

[response]

```
# res_login_form.jsp      ! [Ctrl + F11] login_btn에 style 적용값 변경
# res_login_ok.jsp        ! 로그인 성공시 파라미터값 request 폴더의 req_album01.jsp로 redirect
[request]
```

```
# => req_album01.jsp      ! a태그에 실어보낼 값 response 폴더로 변경
```

[response]

```
# res_album.jsp           ! req_album에서 받은 파라미터값으로 원하는 화면으로 redirect 처리
# => res_mv01.jsp         ! req_album01.jsp의 첫 번째 <div>영역 가져옴
# => res_mv02.jsp         ! req_album01.jsp의 두 번째 <div>영역 가져옴
```

# req\_login\_form.jsp

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

스타일 태그 내부에 적용하고 싶은 디자인을 class이름 또는 id이름으로 접근하여 지정합니다.  
그러면 해당 class이름이나 id이름을 가진 모든 태그에 디자인이 일괄 적용됩니다.

클래스 이름 선택시=> .클래스이름  
아이디 선택시=> #아이디이름

```
--%>
<style>
.login_btn {
border: 1px solid #4B546A;
-webkit-box-shadow: #B7B8B8 0px 1px 0px inset;
-moz-box-shadow: #B7B8B8 0px 1px 0px inset;
box-shadow: #B7B8B8 0px 1px 0px inset;
-webkit-border-radius: 3px;
-moz-border-radius: 3px;
border-radius: 3px;
font-size: 12px;
font-family: arial, helvetica, sans-serif;
padding: 10px 10px 10px 10px;
text-decoration: none;
display: inline-block;
text-shadow: -1px -1px 0 rgba(0, 0, 0, 0.3);
font-weight: bold;
color: #FFFFFF;
height: 50px;
background-color: #fcfac0;
background-image: -webkit-gradient(linear, left top, left bottom, from(#fcfac0),
to(#f6f283));
background-image: -webkit-linear-gradient(top, #fcfac0, #f6f283);
background-image: -moz-linear-gradient(top, #fcfac0, #f6f283);
background-image: -ms-linear-gradient(top, #fcfac0, #f6f283);
background-image: -o-linear-gradient(top, #fcfac0, #f6f283);
background-image: linear-gradient(to bottom, #fcfac0, #f6f283);
filter: progid:DXImageTransform.Microsoft.gradient(GradientType=0,
startColorstr=#fcfac0, endColorstr=#f6f283);
}

.login_btn:hover {
border: 1px solid #4B546A;
background-color: #faf68f;
background-image: -webkit-gradient(linear, left top, left bottom, from(#faf68f),
to(#f3ed53));
background-image: -webkit-linear-gradient(top, #faf68f, #f3ed53);
background-image: -moz-linear-gradient(top, #faf68f, #f3ed53);
background-image: -ms-linear-gradient(top, #faf68f, #f3ed53);
background-image: -o-linear-gradient(top, #faf68f, #f3ed53);
background-image: linear-gradient(to bottom, #faf68f, #f3ed53);
filter: progid:DXImageTransform.Microsoft.gradient(GradientType=0,
startColorstr=#faf68f, endColorstr=#f3ed53);
}
</style>
```



```

<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <div align="center">
        <h3>SendRedirect 로그인</h3>
        <hr>
        <form action="/res_login_ok.jsp" method="get">
            <table border="1" cellpadding="0" cellspacing="0">
                <tr>
                    <td>
                        <input type="text" name="id" size="10" placeholder="아이디">
                    </td>
                    <td rowspan="2">
                        <input type="submit" value="로그인" style="background-color: #F6358A;
                        --%>
                        <!-- 인라인 스타일시트 방식.
                        <input type="submit" value="로그인" style="background-color: #F6358A;
                        --%>
                        <input type="submit" value="로그인" class="login_btn">
                    </td>
                </tr>
                <tr>
                    <td>
                        <input type="password" name="pw" size="10" placeholder="비밀번호">
                    </td>
                </tr>
            </table>
        </form>
    </div>
</body>
</html>

```

class:디자인)

지정 : 페이지 많은 경우 일괄로 적용 위함

height:50px; width:60px:">

## # res\_login\_ok.jsp

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%--
이 페이지에서는 넘어온 아이디값과 비밀번호값을
받아서 처리합니다.
만약 아이디가 'abc1234'이고 비밀번호가 'xyz9876'이라면
1) 아이디가 일치하지 않았을 경우
   'res_id_fail.jsp'로 리다이렉팅해서
   '존재하지 않는 회원입니다'를 브라우저에 출력하세요.
2) 아이디가 일치할 경우 비밀번호도 확인해서 틀렸을 경우
   'res_pw_fail.jsp'로 리다이렉팅해서
   '비밀번호가 틀렸습니다'를 브라우저에 출력하세요.
3) ID와 PW가 전부 일치했을 경우 'res_welcome.jsp'로
   리다이렉팅하여 '회원님 반갑습니다'를 출력하세요.
--%>
<%
String id = request.getParameter("id");
String pw = request.getParameter("pw");

if(id.equals("abcd")){
    // 아이디 일치
    // response 내장객체가 지원하는 sendRedirect() 메서드는 매개값으로
    // 페이지의 URL을 지정하면 해당 페이지로 강제이동 시킵니다.

    if(pw.equals("1234")){
        // 비밀번호도 일치
        //response.sendRedirect("/res_welcome.jsp");
        response.sendRedirect("../request/req_album01.jsp");
    }else{
        // 비밀번호 불일치
        response.sendRedirect("/res_pw_fail.jsp");
    }
}else{
    // 아이디 불일치
    response.sendRedirect("/res_id_fail.jsp");
}
%>

```

## # req\_album01.jsp

```
<%@ 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>
    <div align="center">
        <form action="req_album02.jsp">
            <table border="1" width="700">
                <tr>
                    <td>앨범 커버</td>
                    <td>가수</td>
                    <td>앨범 제목</td>
                    <td>발매일</td>
                </tr>
                <tr>
                    <td align="center">
                        <!-- <a href="req_album02.jsp?title=lovelyz4"> -->
                        <a href=" ../response/res_album.jsp?title=lovelyz4">
                            
                        </a>
                    </td>
                    <td>러블리즈</td>
                    <td>
                        <a href=" ../response/res_album.jsp?title=lovelyz4">
                            러블리즈 (Lovelyz)-미니앨범 4집 : [治癒] (치유)
                        </a>
                    </td>
                    <td>2018-04-23</td>
                </tr>
                <tr>
                    <td align="center">
                        <!-- <a href="req_album02.jsp?title=secret"> -->
                        <a href=" ../response/res_album.jsp?title=secret">
                            
                        </a>
                    </td>
                    <td>오마이걸</td>
                    <td>
                        <a href=" ../response/res_album.jsp?title=secret">
                            비밀정원
                        </a>
                    </td>
                    <td>2018-01-09</td>
                </tr>
                <tr>
                    <td colspan="4">
                        <!-- td의 colspan속성은 열을 병합하는 기능입니다. -->
                        <td colspan="5">
                            <input type="submit" value="확인">
                        </td>
                    </tr>
            </table>
        </form>
    </div>
</body>
</html>
```

## # res\_album.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    // 링크주소에 들어있는 파라미터 변수를 처리한 후
    // 해당 파라미터에 값에 맞게 각각 다른 영상이 나오도록 리다이렉팅하세요.
    // 1번째 영상 페이지(response폴더에 res_mv01.jsp)
    // 2번째 영상 페이지(response폴더에 res_mv02.jsp)

    String title = request.getParameter("title");

    if(title.equals("lovelyz4")){
        response.sendRedirect("res_mv01.jsp");
    }else{
        response.sendRedirect("res_mv02.jsp");
    }
%>
```

## # res\_mv01.jsp

```
<%@ 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>
    <div align="center">
        <h2>선택하신 앨범 정보</h2>
        <p>
            당신이 선택한 앨범은 러블리즈 미니앨범 4집 '치유'입니다.
        </p>
        <hr>
        <h3>타이틀 곡 MV</h3>
        <!--유튜브> 동영상 광고 끝나고 오.버> 소스코드 복사 --%>
        <!--src="" 뒤> ?rel=0:apm:autoplay=1 복.불 --%>
        <iframe width="800" height="600"
src="https://www.youtube.com/embed/40Z9-kr504w?rel=0:apm:autoplay=1" frameborder="0" allow="autoplay;
encrypted-media" allowfullscreen></iframe>

    </div>
</body>
</html>
```

## # res\_mv02.jsp

```
<%@ 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>
    <div align="center">
        <h2>선택하신 앨범 정보</h2>
        <p>
            당신이 선택한 앨범은 오마이걸의 '비밀정원' 입니다.
        </p>
        <hr>
        <h3>타이틀 곡 MV</h3>
        <iframe width="800" height="600"
src="https://www.youtube.com/embed/QIN5_tJRiyY?rel=0:apm:autoplay=1" frameborder="0" allow="autoplay;
encrypted-media" allowfullscreen></iframe>

    </div>
</body>
</html>
```

localhost:8181/junga/response/res\_login\_form.jsp

사안내 > 강의시... Hotdeal\_snow\_20... 생활코딩 추천 사이트 애플파일 IE에서 가져온 북마크

## SendRedirect 로그인

abcd	로그인
....	

localhost:8181/junga/request/req\_album01.jsp

사안내 > 강의시... Hotdeal\_snow\_20... 생활코딩 추천 사이트 애플파일 IE에서 가져온 북마크 신임SW인력을 위... KG아이

앨범 커버	가수	앨범 제목	발매일
	러블리즈	<a href="#">러블리즈 (Lovelyz)-미니앨범 4집 : [治愈](치유)</a>	2018-04-23
	오마이걸	<a href="#">비밀정원</a>	2018-01-09
<input type="button" value="확인"/>			


localhost:8181/junga/response/res\_mv01.jsp


사안내 > 강의시... Hotdeal\_snow\_20... 생활코딩 추천 사이트 애플파일 IE에서 가져온 북마크 신임SW인력을 위... KG아이티뱅크 평생.


## 선택하신 앨범 정보


당신이 선택한 앨범은 러블리즈 미니앨범 4집 '치유'입니다.

## 타이틀 곡 MV

 [MV] Lovelyz(러블리즈) \_ That day(그날의 너)

 나중에 시청하기

 공유

 정보



12

### 4-3. JSP - 내장객체 (3. session, cookie)

#### 1) 개념

##### \* JSP 내장 객체의 종류

- ★1. request javax.servlet.http.HttpServletRequest
- ★2. response javax.servlet.http.HttpServletResponse
- ★3. out javax.servlet.jsp.JspWriter
- ★4. session javax.servlet.http.HttpSession
- 5. application javax.servlet.ServletContext
- 6. pageContext javax.servlet.jsp.PageContext
- 7. page javax.servlet.jsp.HttpJspPage
- 8. config javax.servlet.ServletConfig
- 9. exception java.lang.Throwable

##### \* 쿠키(Cookie)

- 웹 브라우저에서 서버로 어떤 데이터를 요청하면, 서버측에서는 알맞은 로직을 수행한 후 데이터를 웹 브라우저에 응답합니다.
- 그리고 Http 프로토콜은 응답 후에 웹 브라우저와의 관계를 종료합니다.
- 연결이 끊겼을 때, 어떤 정보를 지속적으로 유지하기 위한 수단으로 쿠키라는 방식을 사용합니다.
- 쿠키는 서버에서 생성하여, 서버가 아닌 클라이언트측(local)에 정보를 저장합니다.
- 서버에서 요청할 때마다 쿠키의 속성값을 참조 또는 변경할 수 있습니다.
- 쿠키는 개당 4kb로 용량이 제한적이며, 300개까지(1.2MB) 데이터 정보를 가질 수 있습니다.
- 쿠키문법: 쿠키클래스에서 쿠키 생성 -> setter메서드로 쿠키의 속성 설정 -> response객체에 쿠키 탑재 -> 로컬 환경에 저장
- Cookie 객체 관련 메서드

1. setMaxAge(): 쿠키의 유효시간을 설정합니다.
2. setPath(): 쿠키사용의 유효디렉토리를 설정합니다.
3. setValue(): 쿠키의 값을 설정합니다.
4. setVersion(): 쿠키 버전을 설정합니다.
5. getMaxAge(): 쿠키 유효기간 정보를 얻습니다.
6. getName(): 쿠키의 이름을 얻습니다.
7. getPath(): 쿠키사용의 유효디렉토리 정보를 얻습니다.
8. getValue(): 쿠키의 값을 얻습니다.
9. getVersion(): 쿠키 버전을 얻습니다.

## \* 세션(Session)

- 세션도 쿠키와 마찬가지로 서버와의 관계를 유지하기 위한 수단입니다.
- 단, 쿠키와 달리 클라이언트의 특정 위치에 저장되는 것이 아니라, 서버상에 객체형태로 존재합니다.
- 서버당 하나의 세션 객체를 가질 수 있습니다.
- 세션 객체는 브라우저 창을 종료하면 삭제됩니다. ★
- 따라서 세션은 서버에서만 접근이 가능하여 보안이 좋고, 저장할 수 있는 데이터에 한계가 없습니다. ★
- 세션은 클라이언트의 요청이 발생하면 자동생성되어 고유한 ID값을 클라이언트에 넘겨주며 이것은 쿠키에 저장됩니다.
- JSP에서는 session이라는 내장 객체를 지원하여 세션의 속성을 설정할 수 있습니다.

### - session 객체 관련 메서드

1. setAttribute() - 세션에 데이터를 저장합니다.
2. getAttribute() - 세션에 저장되어 있는 데이터를 얻습니다.
3. getAttributeNames() - 세션에 저장되어 있는 모든 데이터의 세션 이름(key)을 얻습니다.
4. getId() - 자동생성된 세션의 유니크한 아이디를 얻습니다.
5. getCreationTime() - 세션이 생성된 시간을 구합니다.
6. getLastAccessedTime() - 웹 브라우저가 가장 마지막에 세션에 접근한 시간을 구합니다.
7. setMaxInactiveInterval() - 세션의 유효시간을 설정합니다. 초 단위로 기록합니다.
8. getMaxInactiveInterval() - 세션의 유효시간을 얻습니다. 가장 최근 요청시점을 기준으로 카운트됩니다.  
<session-timeout>60</session-timeout>
9. removeAttribute() - 특정 세션을 삭제합니다.
10. invalidate() - 모든 세션을 삭제합니다.

## \* 쿠키 vs 세션 ==> 어느게 더 나은가 ?

- 쿠키 대신에 세션을 사용하는 가장 큰 이유는 세션이 쿠키보다 보안에서 앞서기 때문입니다.
- 쿠키의 이름이나 데이터는 네트워크를 통해 전달되기 때문에 HTTP 프로토콜을 사용하는 경우 중간에서 누군가가 쿠키의 값을 읽어올 수 있습니다.
- 그러나 세션은 오직 서버에만 저장되기 때문에 중요한 데이터를 저장하기에 좋습니다.
- 세션을 사용하는 또 다른 이유는 웹 브라우저가 쿠키를 지원하지 않거나 강제로 사용자가 쿠키를 차단한 경우에도 사용할 수 있다는 점입니다.
- JSP에서는 쿠키를 사용할 수 없는 경우 세션ID를 쿠키에 저장하지 않고 URL 재작성 방식을 통해 세션ID를 URL로 웹 서버에 전달합니다.

- 세션은 여러 서버에서 공유할 수 없는 단점이 있습니다. 그러나 쿠키는 도메인을 이용해 쿠키를 여러 도메인에서 공유할 수 있기 때문에 Naver, Daum과 같은 포털사이트들은 쿠키에 로그인 방식을 저장하는 것을 선호합니다.  
ex) www.naver.com과 mail.naver.com, blog.naver.com의 서버는 각각 다름.

### => 1. 세션 쓰기

- 장점: 보안 강함. / 단점: 공유가 안됨.

### => 2. 쿠키 쓰기

- 장점: 공유가 됨. 따라서 다른데서 많이 쓰임. / 단점: 보안이 약함. 암호화 처리해야...

## 2) sample

sample 1> cookie 1

# cookie\_set.jsp ! 📄 [Ctrl + F11]

# cookie\_get.jsp

# cookie\_all.jsp

# cookie\_set.jsp ! 📄 [Ctrl + F11]

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    // 쿠키 생성 방법
    // 1. 쿠키객체를 생성 - 생성자의 매개값으로 쿠키의 이름과 저장할 데이터를 입력.
    Cookie nameCookie = new Cookie("user_name", "홍길동");
    Cookie idCookie = new Cookie("user_id", "abc1234");

    // 2. 쿠키 클래스의 setter메서드를 이용하여 쿠키의 속성을 설정.
    nameCookie.setMaxAge(60 * 60); //쿠키 유효시간 지정 60초 × 60초 = 1시간
    idCookie.setMaxAge(20);

    // 3.http응답시 response객체에 생성된 쿠키를 탑재하여 클라이언트로 전송.
    response.addCookie(nameCookie);
    response.addCookie(idCookie);
%>
```

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <a href="cookie_get.jsp">특정 쿠키 사용하기</a>
</body>
</html>
```

# cookie\_get.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    Cookie[] cookies = request.getCookies(); // 클라이언트측에 저장되어 있는 쿠키를 가져오는 메서드.

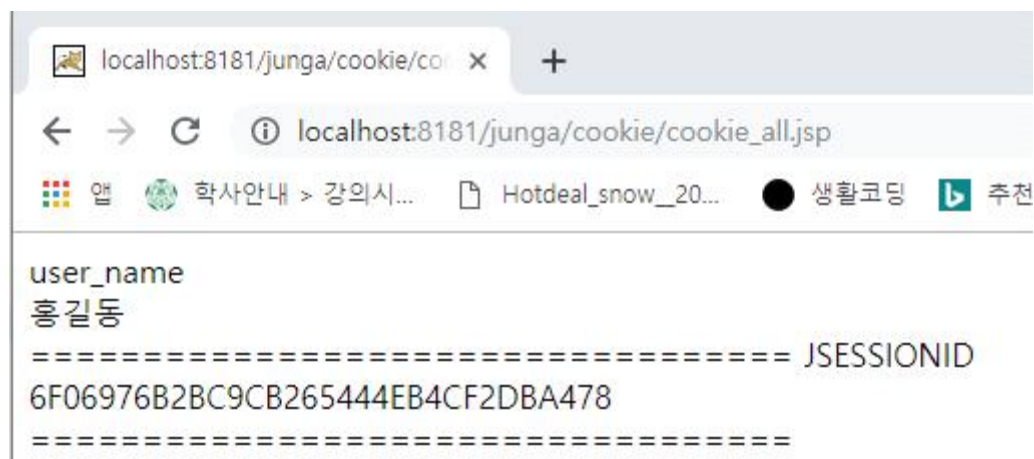
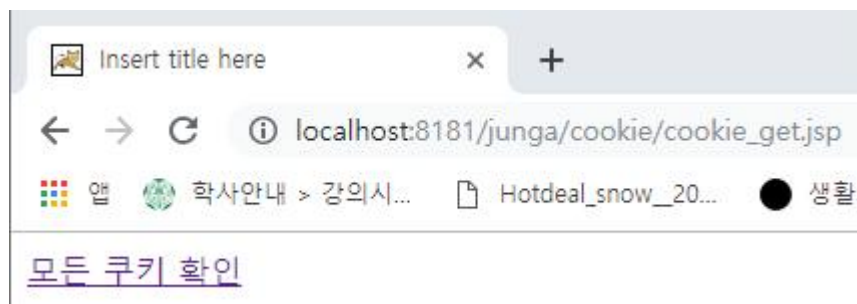
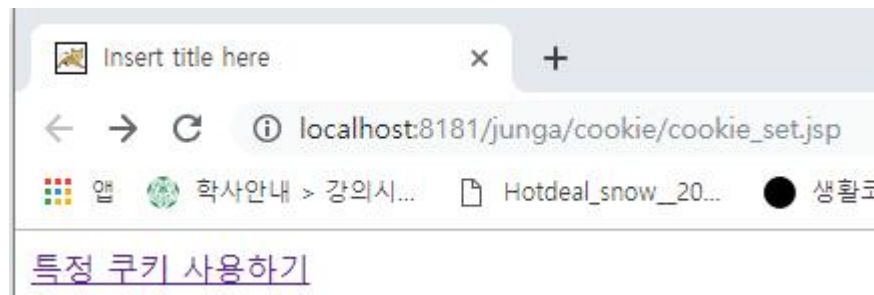
    String str = null;

    for(int i=0; i<cookies.length; i++){
        str = cookies[i].getName();
        if(str.equals("user_id")){
            String id = cookies[i].getValue();
            out.println(id + "님은 로그인 중입니다.<br>");
        }
    }
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <a href="cookie_all.jsp">모든 쿠키 확인</a>
</body>
</html>
```

## # cookie\_all.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    Cookie[] cookies = request.getCookies();

    if(cookies != null){
        for(int i=0; i<cookies.length; i++){
            out.println(cookies[i].getName() + "<br>");
            out.println(cookies[i].getValue() + "<br>");
            out.println("=====");
        }
    }
%>
```





sample 2> cookie 실전 적용(주로 js, 프론트단에서 처리)

# cookie\_login.jsp !  [Ctrl + F11]

# cookie\_login\_ok.jsp

# cookie\_login\_welcome.jsp

# cookie\_logout.jsp

# cookie\_all.jsp

# cookie\_login.jsp !  [Ctrl + F11]

```
<%@ 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>
    <form action="cookie_login_ok.jsp" method="post">
        <input type="text" name="id" size="10" placeholder="ID"><br>
        <input type="text" name="pw" size="10" placeholder="PW"><br>
        <input type="submit" value="로그인">
    </form>
</body>
</html>
```

# cookie\_login\_ok.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    request.setCharacterEncoding("utf-8");

    String id = request.getParameter("id");
    String pw = request.getParameter("pw");

    if(id.equals("abc1234") && pw.equals("1234")){
        Cookie idCookie = new Cookie("user_id", id);
        idCookie.setMaxAge(30);
        response.addCookie(idCookie);
        response.sendRedirect("cookie_login_welcome.jsp");
    }else{
        response.sendRedirect("cookie_login.jsp");
    }
%>
```

# cookie\_login\_welcome.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    Cookie[] cookies = request.getCookies();

    for(int i=0; i<cookies.length; i++){
        String id = cookies[i].getValue();
        if(id.equals("abc1234")){
            out.println(id + "님 안녕하세요.<br>");
        }
    }
%>

<a href="cookie_logout.jsp">로그아웃</a> &nbsp;
<a href="cookie_all.jsp">모든 쿠키 보기</a>
```

## # cookie\_logout.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    Cookie[] cookies = request.getCookies();
    if(cookies != null){
        for(int i=0; i<cookies.length; i++){
            cookies[i].setMaxAge(0); //쿠키의 삭제
            response.addCookie(cookies[i]);
        }
        //response.sendRedirect("cookie_login.jsp");
        response.sendRedirect("cookie_all.jsp");
    %>
```

## # cookie\_all.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    Cookie[] cookies = request.getCookies();

    if(cookies != null){
        for(int i=0; i<cookies.length; i++){
            out.println(cookies[i].getName() + "<br>");
            out.println(cookies[i].getValue() + "<br>");
            out.println("=====");
        }
    }
    %>
```


sample 3> session 1 (백에서 처리 집중!)

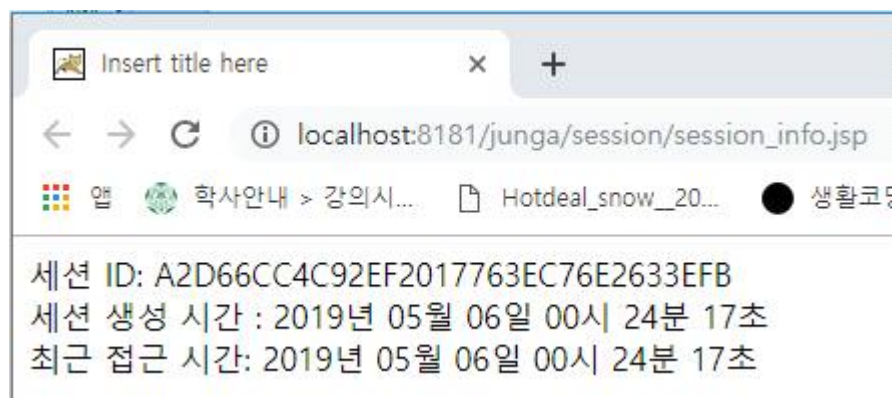
=> session 폴더 생성!

### # session\_info.jsp

```
<%@page import="java.util.Date"%>
<%@page import="java.text.SimpleDateFormat"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>

<%
    Date date = new Date();
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy년 MM월 dd일 HH시 mm분 ss초");
%>

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    세션 ID: <%=session.getId() %> <br>
    <%
        /*
            getCreationTime()은 세션의 생성 시간은 long타입의 밀리세컨드로
            리턴하기 때문에 date객체의 setTime() 메서드를 통해 해당 밀리초를
            날짜형식으로 반환해야 합니다.
        */
        date.setTime(session.getCreationTime());
        String createTime = sdf.format(date);
    %>
    세션 생성 시간 : <%=createTime %> <br>
    <%
        date.setTime(session.getLastAccessedTime());
        String accessTime = sdf.format(date);
    %>
    최근 접근 시간: <%= accessTime %>
</body>
</html>
```



sample 4> session 2

# session\_set.jsp ! 🖱 [Ctrl + F11] 세션 개념,

# session\_test.jsp

! session 객체 관련 메서드(유효시간, 특정 세션 데이터 삭제/확인, 모든 세션 데이터 삭제/확인)

# session\_set.jsp ! 🖱 [Ctrl + F11] 세션 개념,

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    /*
        세션에 데이터를 저장할 때는 내장객체 session이 지원하는
        setAttribute()메서드를 사용합니다.
        해당 메서드의 첫번째 매개값으로 세션의 이름을 정해주고, 두번째 매개값으로
        세션에 저장할 데이터를 정해줍니다.
    */
    session.setAttribute("user_id", "zzz1234");
    session.setAttribute("user_name", "박철수");
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <a href="session_test.jsp">세션 테스트</a>
</body>
</html>
```

# session\_test.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    /*
        세션객체에 저장된 데이터는 브라우저 창이 종료되거나 혹은 세션의 유효시간이
        만료되기 전까지는 어느 페이지에서나 해당 데이터를 사용할 수 있습니다.
        (통신이 끊어져도 날라가지 않음?)
        session객체의 getAttribute("세션의 이름")를 사용합니다.
    */
    String id = (String) session.getAttribute("user_id");
    // Typecasting 조건: 1.상속관계, 2.인터페이스 구현관계 있어야...
    String name = (String) session.getAttribute("user_name");
    out.println(name + "<br>");
    out.println(id + "<br>");
    out.println("=====<br>");
    // 세션의 유효시간은 Servers 폴더에 web.xml에 기술되어 있습니다.

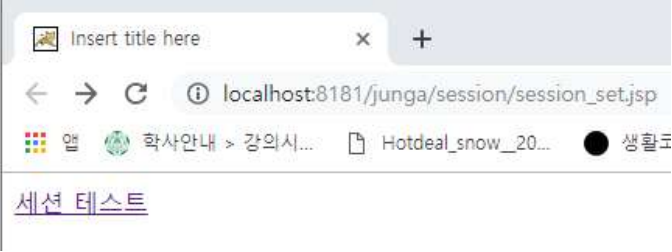
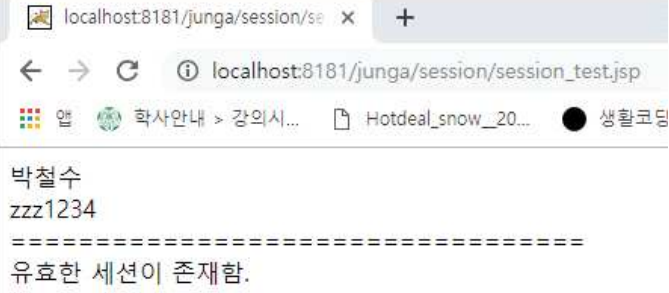
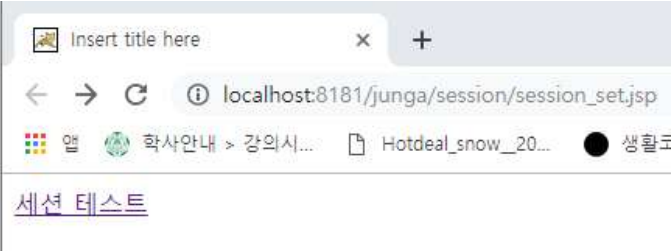
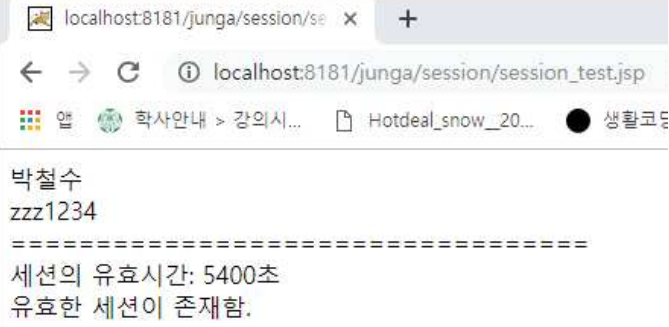
    // setter메서드로 유효시간을 설정하면 web.xml 설정보다 우선적용됩니다.
    session.setMaxInactiveInterval(5400);

    int time = session.getMaxInactiveInterval(); //세션의 유효시간을 초단위로 지정
    out.println("세션의 유효시간: " + time + "초<br>");

    // 특정 세션 데이터 삭제
    session.removeAttribute("user_name");

    // isRequestedSessionIdValid() 메서드는
    // 유효한 세션이 존재하는지의 여부를 boolean타입으로 리턴.

    if(request.isRequestedSessionIdValid()){
        out.println("유효한 세션이 존재함.");
    }else{
        out.println("유효한 세션이 존재하지 않음.");
    }
%>
```

sample 5> session 3

# session\_login.jsp !  [Ctrl + F11]

# session\_login\_ok.jsp

# session\_login\_welcome.jsp

# session\_logout.jsp

# session\_login.jsp !  [Ctrl + F11]

```
<%@ 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>
    <%
        // 로그인한사람은 보지 못하게 처리
        if(session.getAttribute("user_id") != null){
            String nick = (String) session.getAttribute("nick");
    %>
    <%= nick %>님이 현재 로그인중입니다. <br>
    <a href="session_login_welcome.jsp">메인으로</a>
    <%
        }else{
    %>
    <!--로그인 안 한사람 뷰 --%>
    <div align="center">
        <form action="session_login_ok.jsp" method="post">
            ID: <input type="text" name="id" size="10"><br>
            PW: <input type="text" name="pw" size="10"><br>
            별명: <input type="text" name="nick" size="10"><br>
            <input type="submit" value="로그인">
        </form>
    </div>
    <%
        }
    %>
</body>
</html>
```

## # session\_login\_ok.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    request.setCharacterEncoding("utf-8");

    String id = request.getParameter("id");
    String pw = request.getParameter("pw");
    String nickname = request.getParameter("nick");

    if(id.equals("abc1234") && pw.equals("1234")){
        session.setAttribute("user_id", id);
        session.setAttribute("nick", nickname);
        response.sendRedirect("session_login_welcome.jsp");
    }else{
%>
        <%--
            JS(Javascript)
            HTML내부에 자바스크립트 코드를 사용하려면 <script>라는 태그를 사용합니다.
            자바스크립트 내장함수 alert()은 브라우저에 경고창을 띄웁니다.
            자바스크립트 내장객체 history가 지원하는 go()함수는 매개값으로 -1dmf wnaus
            뒤로가기 기능을 실행합니다.
            --%>
            <script>
                alert("로그인에 실패했습니다.")
                history.go(-1)
            </script>
        <%
    }
%>
```

## # session\_login\_welcome.jsp

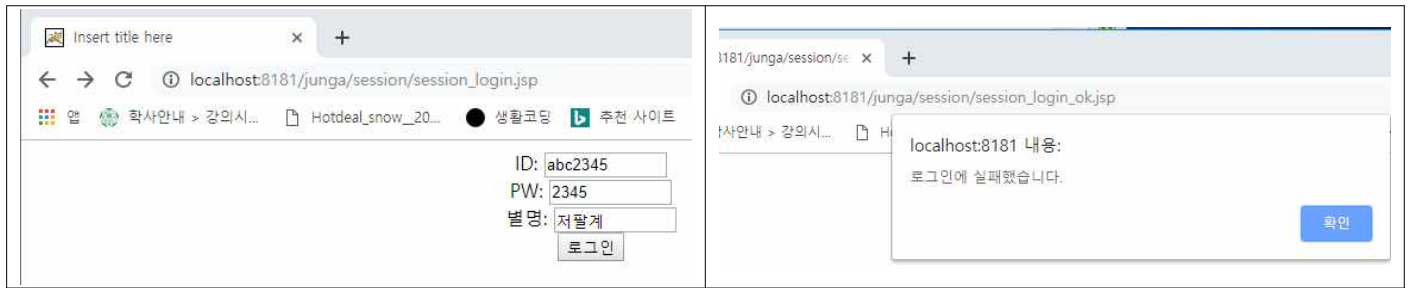
```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    // 로그인인 인가되지 않은 사람은 이 페이지를 볼 수 없도록 세션을 사용합니다.
    if (session.getAttribute("user_id") == null){
        response.sendRedirect("session_login.jsp");
    }
    String id = (String) session.getAttribute("user_id");
    String nick = (String) session.getAttribute("nick");
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h4><%=nick %>(<%=id %>)님 안녕하세요.</h4>

    <a href="session_login.jsp">로그인 페이지로</a>
    <a href="session_logout.jsp">로그아웃</a>
</body>
</html>
```

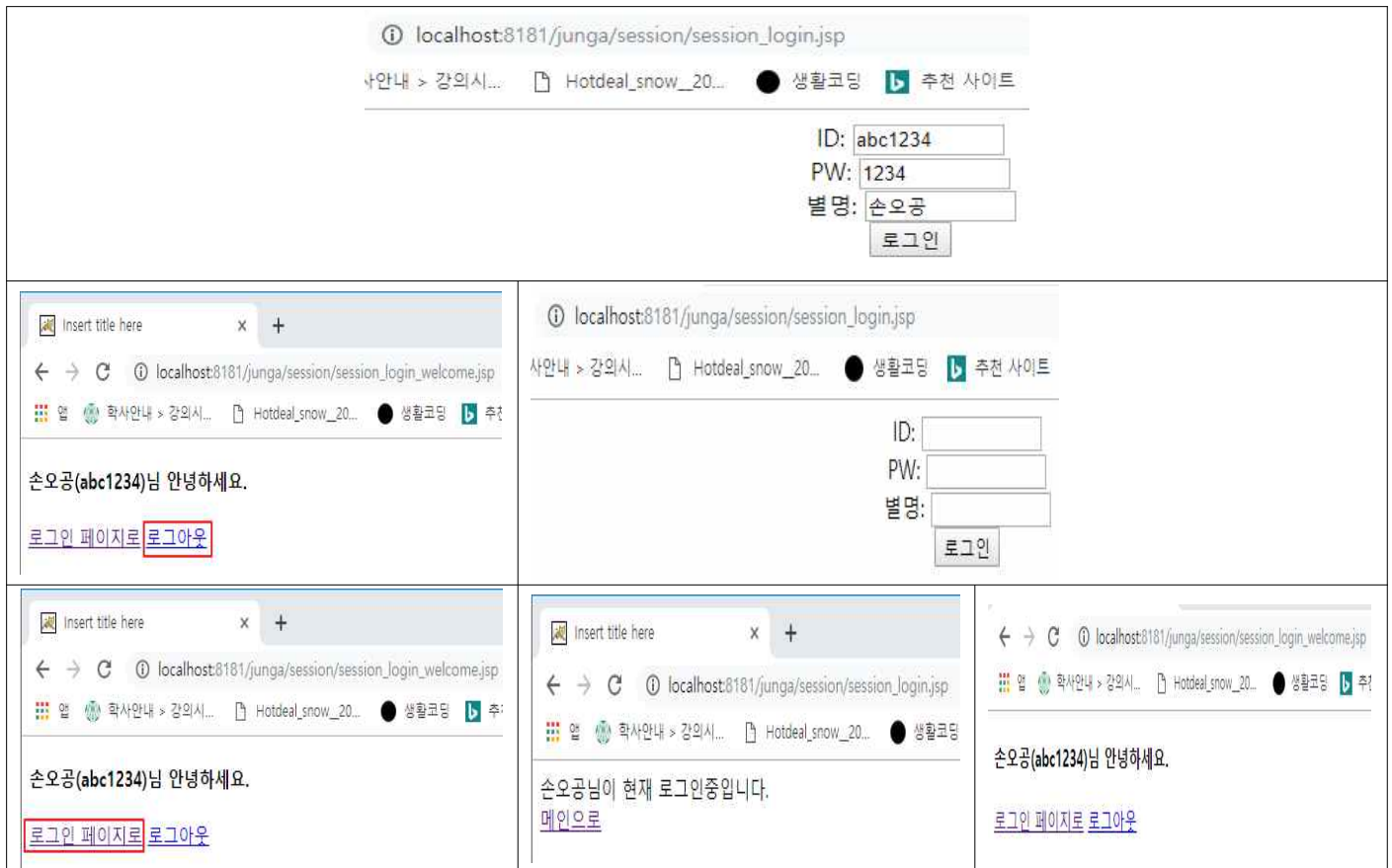
## # session\_logout.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    session.invalidate();
    response.sendRedirect("session_login.jsp");
%>
```

## 로그인 실패시



## 로그인 성공시





sample 6> 실습 1

- 로그인하면 앨범 볼수 있게 웰컴 페이지에서 연결...
- 로그인 안하면 못보게 설정해보기!!!

case 1> session으로 처리

```
# session_login.jsp    ! [Ctrl + F11]
# session_login_ok.jsp
# session_login_welcome.jsp // 수정 ○
# req_album01.jsp      // 수정 ○
# session_logout.jsp
```

case 2> cookie로 처리

=> case 1>과 비슷하게 처리하면 될 듯...

# session\_login\_welcome.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    // 로그인이 인가되지 않은 사람은 이 페이지를 볼 수 없도록 세션을 사용합니다.
    if (session.getAttribute("user_id") == null){
        response.sendRedirect("session_login.jsp");
    }
    // String id = (String) session.getAttribute("user_id");
    // String nick = (String) session.getAttribute("nick");
    response.sendRedirect("../request/req_album01.jsp");
%>

<%--
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h4><%=nick %>(<%=id %>)님 안녕하세요.</h4>

    <a href="session_login.jsp">로그인 페이지로</a>
    <a href="session_logout.jsp">로그아웃</a>
</body>
</html>
--%>
```

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    String id = (String) session.getAttribute("user_id");
    String nick = (String) session.getAttribute("nick");
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <div align="center">
        <form action="../session/session_login.jsp" method="post">
            <h4><%=nick %>(<%=id %>)님 안녕하세요.</h4>
            <table border="1" width="700">
                <tr>
                    <td>앨범 커버</td>
                    <td>가수</td>
                    <td>앨범 제목</td>
                    <td>발매일</td>
                </tr>
                <tr>
                    <td align="center">
                        <a href="../response/res_album.jsp?title=lovelyz4">
                            
                        </a>
                    </td>
                    <td>러블리즈</td>
                    <td>
                        <a href="../response/res_album.jsp?title=lovelyz4">
                            러블리즈 (Lovelyz)-미니앨범 4집 : [治癒] (치유)
                        </a>
                    </td>
                    <td>2018-04-23</td>
                </tr>
                <tr>
                    <td align="center">
                        <a href="../response/res_album.jsp?title=secret">
                            
                        </a>
                    </td>
                    <td>오마이걸</td>
                    <td>
                        <a href="../response/res_album.jsp?title=secret">
                            비밀정원
                        </a>
                    </td>
                    <td>2018-01-09</td>
                </tr>
                <tr>
                    <td align="center" colspan="4">
                        <!-- td의 colspan 속성은 열을 병합하는 기능입니다. -->
                        <td colspan="5">
                            <input type="submit" value="확인">&nbsp;  
                            <a href="../session/session_logout.jsp">로그아웃</a>
                        </td>
                    </tr>
                </tr>
            </table>
        </form>
    </div>
</body>
</html>
```

localhost:8181/junga/session/session\_login.jsp



[학사안내 > 강의시...](#)
[Hotdeal\\_snow\\_20...](#)
● 생활코딩
추천 사이트

ID: 
PW: 
별명:

localhost:8181/junga/request/req\_album01.jsp

[학사안내 > 강의시...](#)
[Hotdeal\\_snow\\_20...](#)
● 생활코딩
추천 사이트
애플파워
IE에서 가져온 북마크

사오정(abc1234)님 안녕하세요.



앨범 커버	가수	앨범 제목	발매일
	러블리즈	러블리즈 (Lovelyz)-미니앨범 4집 : [洛麗] (치유)	2018-04-23
	오마이걸	비밀정원	2018-01-09

localhost:8181/junga/session/session\_login.jsp

[학사안내 > 강의시...](#)
[Hotdeal\\_snow\\_20...](#)
● 생활코딩
추천 사이트

ID: 
PW: 
별명:

사오정(abc1234)님 안녕하세요.

앨범 커버	가수	앨범 제목	발매일
	러블리즈	러블리즈 (Lovelyz)-미니앨범 4집 : [洛麗] (치유)	2018-04-23
	오마이걸	비밀정원	2018-01-09

Insert title here



[←](#)
[→](#)
[↻](#)
localhost:8181/junga/session/session\_login.jsp

앱
학사안내 > 강의시...
Hotdeal\_snow\_20...
● 생활코딩

사오정님이 현재 로그인중입니다.

[메인으로](#)

사오정(abc1234)님 안녕하세요.

앨범 커버	가수	앨범 제목	발매일
	러블리즈	러블리즈 (Lovelyz)-미니앨범 4집 : [洛麗] (치유)	2018-04-23
	오마이걸	비밀정원	2018-01-09

#### 4-4. JSP - 내장객체 (3-2. application)

##### 1) 개념

##### \* JSP 내장 객체의 종류 > Review

----- 핵심~!!!

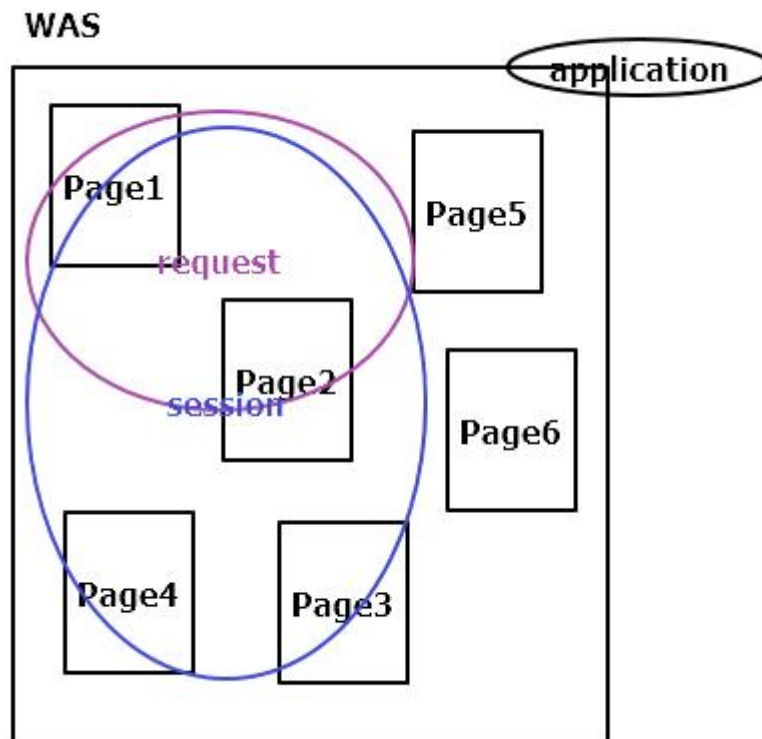
1. request javax.servlet.http.HttpServletRequest ★★★
2. response javax.servlet.http.HttpServletResponse ★★★
3. out javax.servlet.jsp.JspWriter
4. session javax.servlet.http.HttpSession ★★★
5. application javax.servlet.ServletContext ★ (DB에서 유지되므로 잘 안쓰임..)

- 
6. pageContext javax.servlet.jsp.PageContext # 도태...
  7. page javax.servlet.jsp.HttpJspPage # 잘 안씀...
  8. config javax.servlet.ServletConfig # Servlet에서...
  9. exception java.lang.Throwable # 예외

##### \* application 기본 객체 (서버 다운 전까지는 남아있음)

- 특정 웹 어플리케이션에 포함된 모든 JSP페이지는 하나의 application 기본 객체를 공유합니다.
- application 객체는 웹 어플리케이션 전반에 걸쳐서 사용되는 정보를 담고 있습니다.
- request 객체는 요청영역마다 생성되고, session 객체는 브라우저별로 생성되고, application은 프로그램 전체에서 딱 한번 최초 가동시 생성됩니다.

request <<session <<application



## 4-5. JSP - 내장객체 (4. exception)

### 1) 개념

#### \* JSP 내장 객체의 종류 > Review

----- 핵심~!!!

1. request javax.servlet.http.HttpServletRequest ★★★
  2. response javax.servlet.http.HttpServletResponse ★★★
  3. out javax.servlet.jsp.JspWriter
  4. session javax.servlet.http.HttpSession ★★★
  5. application javax.servlet.ServletContext ★ (DB에서 유지되므로 잘 안쓰임..)
- 
6. pageContext javax.servlet.jsp.PageContext # 도태...
  7. page javax.servlet.jsp.HttpJspPage # 잘 안씀...
  8. config javax.servlet.ServletConfig # Servlet에서...
  9. exception java.lang.Throwable # 예외

#### \* 예외 페이지

- 예외 상황이 발생했을 경우 웹 컨테이너(톰캣)에서 제공되는 기본적인 예외페이지가 보여집니다.
- 개발 과정에서는 이러한 예외 페이지를 보고 어떤 에러가 발생했는지 알 수 있기 때문에 오류를 수정하는 데 도움이 됩니다.
- 그러나 사용자에게 상용 서비스를 제공하고 있는데 이러한 딱딱한 페이지가 보여진다면 사용자로 하여금 불쾌감을 일으키고, 해당 사이트에 대한 신뢰도가 하락하게 됩니다.
- 또한 코드의 일부가 노출되어 보안 측면에도 좋지 않습니다.
- 그래서 개발자가 따로 만들어 둔 에러 페이지로 유도하여 사용자에게 친숙한 페이지를 보여줍니다.

#### \* HTTP 주요 응답 상태 코드

1. 404: 요청한 URL을 찾을 수 없는 경우.
2. 500: 서버측 내부 오류로 인해 페이지가 나타나지 않는 경우(java, JSP 페이지 내의 코드오류)
3. 200: 요청을 성공적으로 처리함.
4. 307: 임시로 페이지를 리다이렉트함.
5. 400: 클라이언트의 요청이 잘못된 구문으로 작성됨.
6. 405: 요청 방식을 허용하지 않음(GET, POST 등)
7. 503: 서버가 일시적으로 서비스를 제공할 수 없음(일시적 서버과부하, 서버 임시 보수 등)

#### \* JSP에서 에러를 처리하는 방법

1. 직접 예외를 처리하기
  - 자바의 키워드인 try ~ catch를 사용하여 직접 개발자가 예외를 처리합니다.

#### 2. 에러를 처리할 페이지를 따로 지정하기

- JSP는 실행 도중 예외가 발생할 때 톰캣 기본 에러화면 대신 개발자가 지정한 JSP페이지를 보여줄 수 있는 기능을 제공하고 있습니다.
  - 에러가 발생하면 보여줄 JSP페이지는 페이지 지시자(directive)의 `errorPage` 속성을 사용하여 지정합니다.
  - 예외가 발생할 것으로 예상되는 페이지에 예외가 발생했을 때 보여줄 페이지를 지정합니다.
- ex) `<%@ page errorPage="예외가 발생했을 시 보여줄 페이지" %>`

- 에러 발생시 유도된 페이지에는 페이지 지시자태그로 isErrorPage 속성을 사용하여 true로 값을 설정합니다.  
ex) <%@ page isErrorPage="true" %>

- 에러 페이지로 지정된 JSP파일 내에서는 예외를 처리해주는 exception 내장객체를 사용할 수 있습니다.

- 인터넷 익스플로러 브라우저의 경우 전체 응답 결과의 데이터가 512바이트보다 작을 경우 마이크로소프트에서 자체 제공하는 에러페이지를 출력합니다.

### 3. 응답 상태 코드별로 에러 페이지 지정하기

- JSP는 에러 코드별로 사용할 에러 페이지를 web.xml파일 수정을 통해 지정할 수 있습니다.

- 이렇게 지정한 에러페이지는 일반 JSP파일과 동일하게 작성하면 됩니다.

### 4. 예외 타입별로 에러 페이지 지정하기

- 발생하는 예외의 종류별로도 에러 페이지를 지정할 수 있습니다. web.xml파일을 수정합니다.

## \* 에러 페이지의 우선순위

- 에러 페이지를 여러 방법으로 지정한 경우 다음의 우선순위에 따라 사용할 에러 페이지를 선택합니다.

1. 페이지 지시자 태그의 errorPage속성에 지정한 페이지.
2. web.xml에 지정한 에러 타입에 따른 페이지.
3. web.xml에 지정한 응답 상태 코드에 따른 페이지.
4. 위 3항목에 해당하지 않을 경우 톰캣이 제공하는 에러 페이지.

## 2) sample

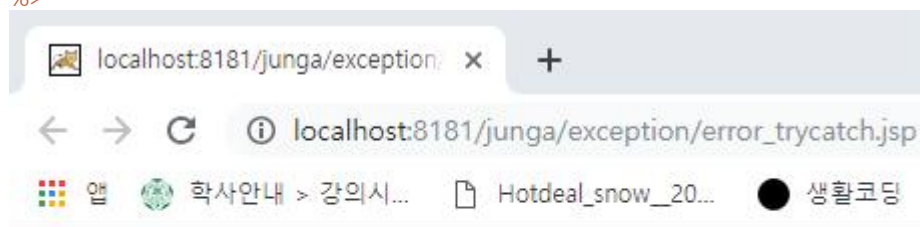
=> New> Folder> exception

sample 1> try~catch

# error\_trycatch.jsp

# error\_trycatch.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    try{
        String id = request.getParameter("id");
        id = id.toLowerCase(); // Nullpoint Exception
        out.println("<h4>" + id + "</h4>");
    }catch(Exception e){
        <h4>죄송합니다. 서버측 오류가 발생했습니다.</h4>
        <h5>빠른 시일 내에 문제를 해결하겠습니다.</h5>
    }
%>
```



죄송합니다. 서버측 오류가 발생했습니다.

빠른 시일 내에 문제를 해결하겠습니다.

sample 2> 준비된 페이지로 보내주기

# error\_page01.jsp !  [Ctrl + F11]

# error\_page02.jsp

# error\_page01.jsp !  [Ctrl + F11]

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page errorPage="error_page02.jsp" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <%= request.getParameter("id").toUpperCase() %> <!--에러 발생 --%>
</body>
</html>
```

# error\_page02.jsp

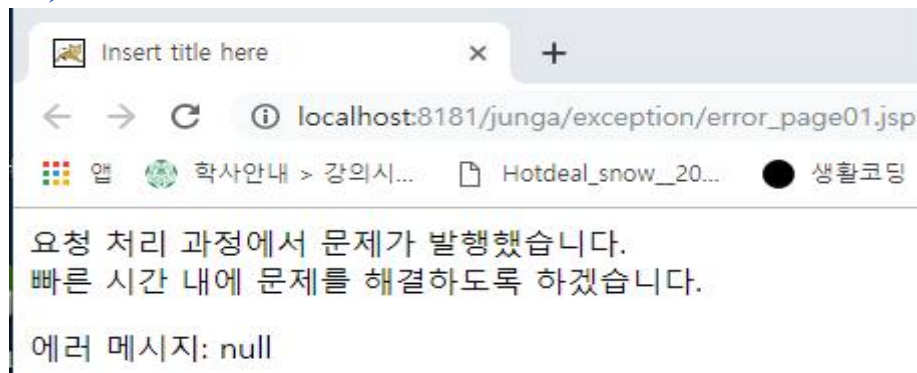
```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!--
    jsp 페이지 내에서 내장객체 exception을 사용하기 위해서는 페이지 디렉티브의 속성
    isErrorPage의 속성값을 true로 지정해야 합니다.
--%>
<%@ page isErrorPage="true" %>

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    요청 처리 과정에서 문제가 발행했습니다. <br>
    빠른 시간 내에 문제를 해결하도록 하겠습니다.
    <p>
        에러 메시지: <%= exception.getMessage() %>
    </p>
</body>
</html>
<!--
```

만약에 에러 페이지의 길이가 512바이트보다 작다면 인터넷 익스플로러는 이 페이지가 출력하는 여러 페이지를 출력하지 않고 자체적으로 제공하는 화면을 출력합니다.  
인터넷 익스플로러에서도 에러 페이지 내용을 올바르게 출력하려면 응답 결과에 512바이트를 넘기는 데이터들을 포함시키면 됩니다.

만약에 에러 페이지의 길이가 512바이트보다 작다면 인터넷 익스플로러는 이 페이지가 출력하는 여러 페이지를 출력하지 않고 자체적으로 제공하는 화면을 출력합니다.  
인터넷 익스플로러에서도 에러 페이지 내용을 올바르게 출력하려면 응답 결과에 512바이트를 넘기는 데이터들을 포함시키면 됩니다.

-->





sample 3> web.xml에 에러 코드 지정

# web.xml(WEB-INF)

# error\_404.jsp, error500.jsp (exception)

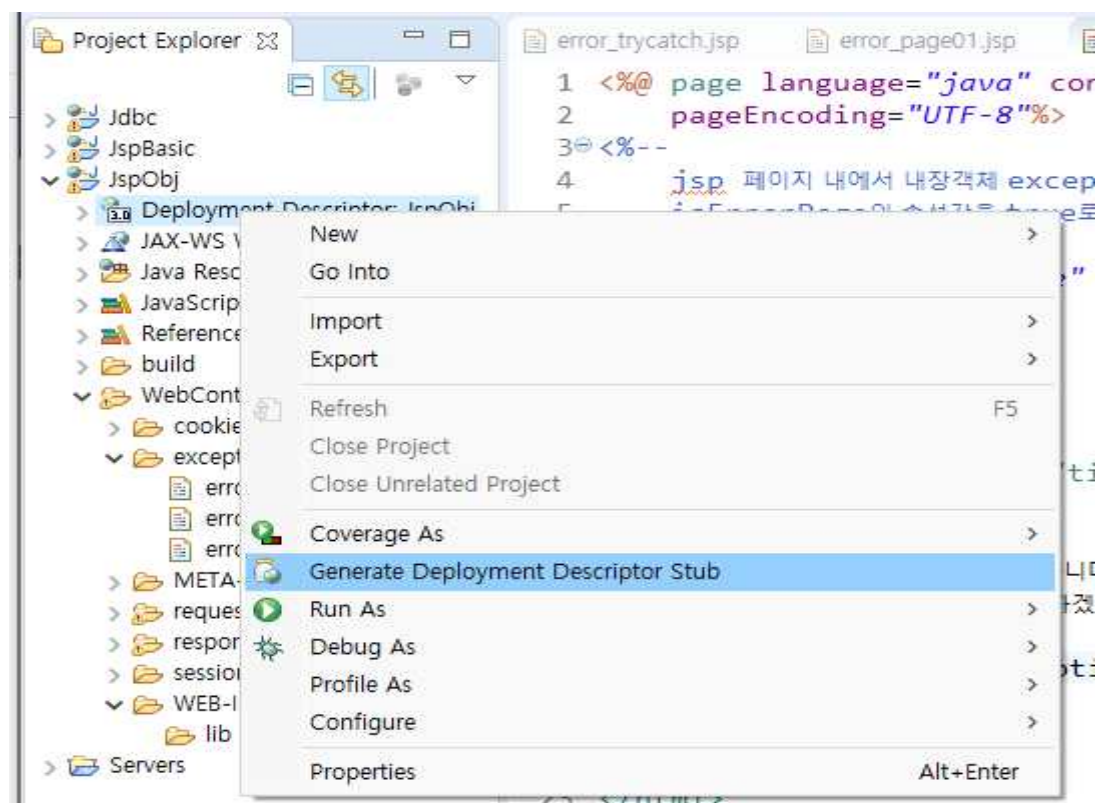
# error\_get404.jsp : 404 error 발생 페이지 ! 👁 [Ctrl + F11]

# error\_get500.jsp : 500 error 발생 페이지 ! 👁 [Ctrl + F11]

# web.xml(WEB-INF)

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
version="3.0">
  <display-name>JspObj</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
  <error-page>
    <error-code>404</error-code>
    <location>/exception/error_404.jsp</location>
  </error-page>
  <error-page>
    <error-code>500</error-code>
    <location>/exception/error_500.jsp</location>
  </error-page>
</web-app>
```

cf. web.xml 생성 방법



### # error\_404.jsp

```
<%@ 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>

    <h3>요청한 페이지를 찾을 수 없는 404에러 입니다.</h3>
    요청한 페이지는 존재하지 않습니다. <br>
    주소를 올바르게 입력했는지 확인해보세요~

</body>
</html>
```

### # error\_get404.jsp : 404 error 발생 페이지 ! 🖱️ [Ctrl + F11]

```
<%@ 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>

    <a href="hello.jsp">404 유발!</a>

</body>
</html>
```

### # error500.jsp (exception)

```
<%@ 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>

    <a href="hello.jsp">404 유발!</a>

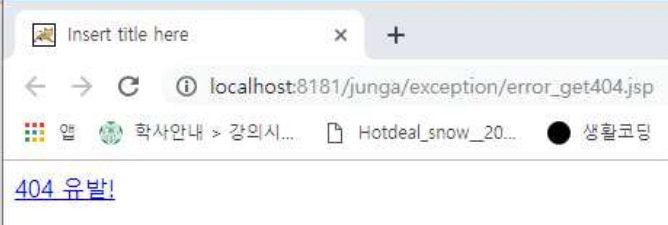
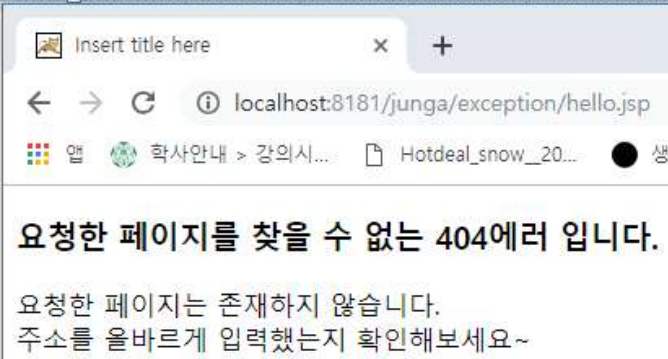

</body>
</html>
```

### # error\_get500.jsp : 500 error 발생 페이지 ! 🖱️ [Ctrl + F11]

```
<%@ 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>

    <%
        int i = 10/0;
    %>

</body>
</html>
```

sample 4> web.xml에 에러 타입 지정

# web.xml(WEB-INF)

# error\_null.jsp(exception)

# error\_get\_null.jsp ! 🐞 [Ctrl + F11] 원래 애는 지금 500이면서 NullPointerException인데,  
후자로 갔음 => 에러 페이지에 우선순위가 있다!!!

# web.xml(WEB-INF)

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
version="3.0">
  <display-name>JspObj</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
  <error-page>
    <error-code>404</error-code>
    <location>/exception/error_404.jsp</location>
  </error-page>
  <error-page>
    <error-code>500</error-code>
    <location>/exception/error_500.jsp</location>
  </error-page>
  <error-page>
    <exception-type>java.lang.NullPointerException</exception-type>
    <location>/exception/error_null.jsp</location>
  </error-page>
</web-app>
```

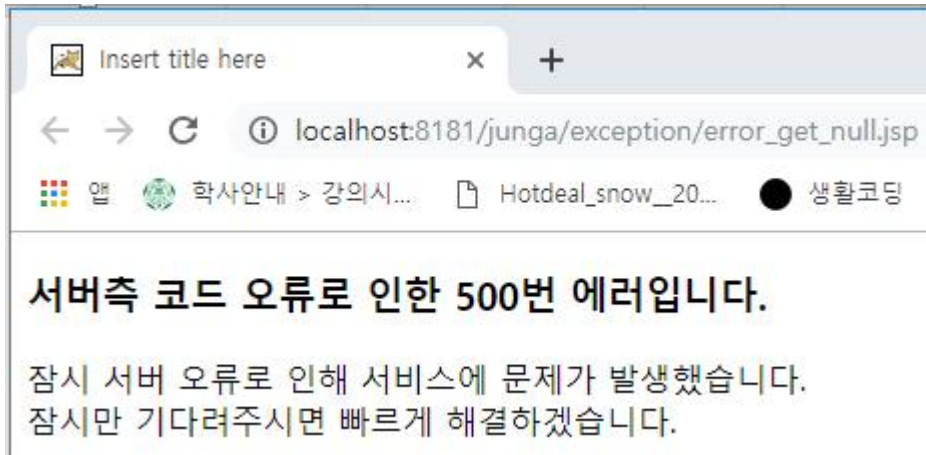
# error\_null.jsp

```
<%@ 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>
    <h3>서비스 처리 과정에서 널(null)에러가 발생했습니다.</h3>
</body>
</html>
```

# error\_get\_null.jsp ! 🐞 [Ctrl + F11] 원래 애는 지금 500이면서 NullPointerException인데,  
후자로 갔음 => 에러 페이지에 우선순위가 있다!!!

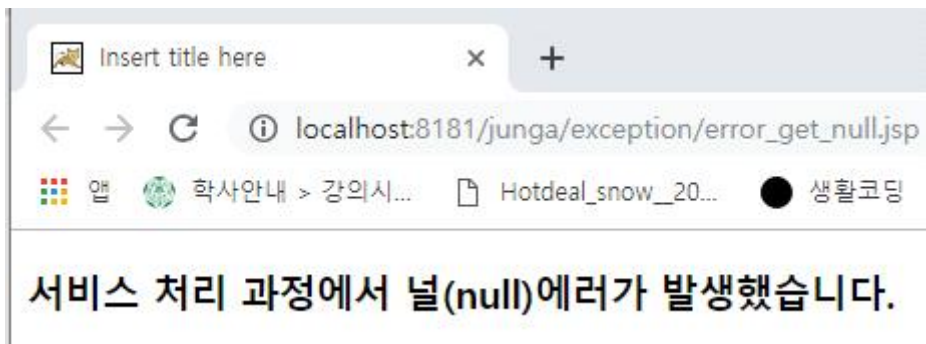
```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <%
        String str = null;
        str.charAt(0);
    %>
</body>
</html>
```

! web.xml에 타입 지정 전



=> 기존에 web.xml에 설정해놓은 500에러 페이지로 넘어감.

! web.xml에 타입 지정 후



=> web.xml에 새로 지정한 exception-type 페이지로 넘어감.  
<exception-type> 이 <error-code> 보다 우선 적용되는 것을 확인 할 수 있다.

## 5. JSP - Action Tag

### 1) 개념

#### \* JSP Action Tag

##### ==> JSP 태그 가독성 보완(JSP 태그의 HTML화)

- JSP 페이지 내에서 어떤 동작을 하도록 지시하는 태그입니다.

- 종류에는 페이지 이동을 강제하는 **forward**, 페이지를 삽입하는 **include**, **forward**나 **include**를 할 때 값을 지정하는 **param**, 자바의 클래스와 연동하는 useBean 등이 있습니다.

1. forward: 현재의 페이지에서 다른 특정 페이지로 전환할 때 사용합니다.

2. include: 현재 페이지에 다른 페이지를 삽입할 때 사용합니다.

==> 코드의 가독성을 위해 디렉티브 태그의 include는 java관련 파일을 추가할때 (scriptlet위에 선언) 사용.

ex) <%... %>

액션태그의 include는 HTML 코드 가운데에 UI추가시 사용.

ex) <jsp:xxx 속성... ></jsp>

3. param: forward 및 include 태그에 데이터를 전달할 목적으로 사용되는 태그입니다. name과 value 속성으로 이루어져 있습니다. ==> 추가로 parameter 전달 가능!

#### \* forward vs sendRedirect

1. forward

- 요청 받은 요청객체(request)를 위임하는 컴포넌트에 요청 객체값을 동일하게 전달할 수 있습니다.

- 요청을 처리함과 동시에 해당 결과를 바로 보여줘야 하는 경우

ex) 게시판 목록에서 글 제목을 클릭했을 때 바로 내용을 보여줘야 하는 경우.

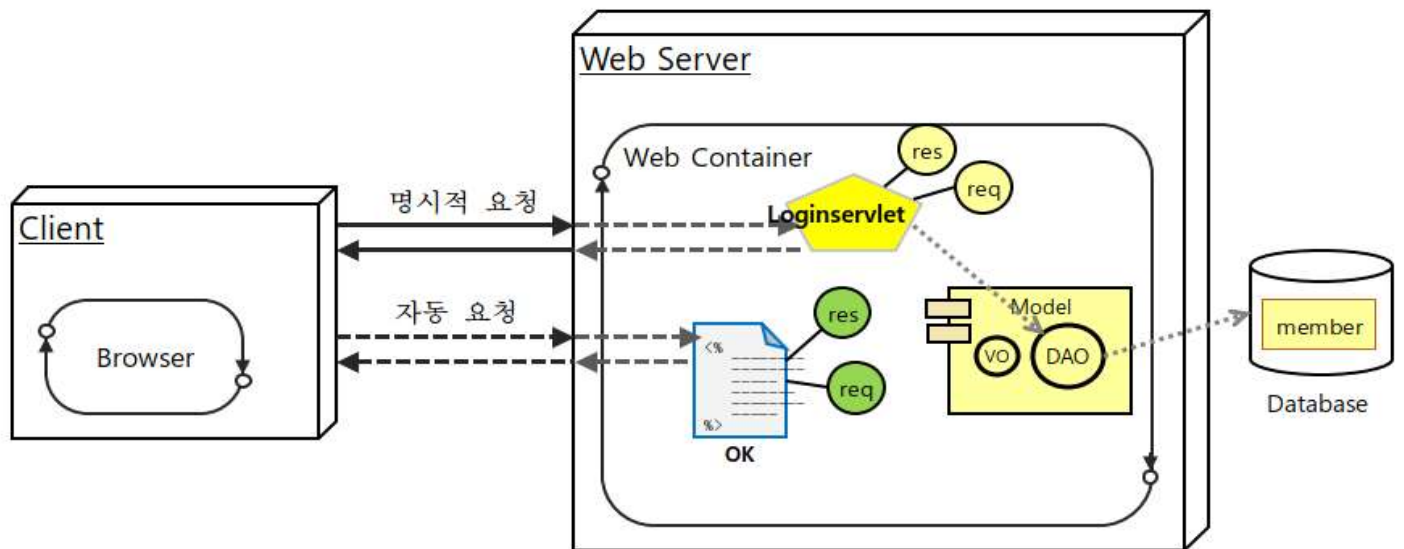
2. sendRedirect

- 요청 받은 요청객체를(request) 위임하는 컴포넌트에 전달하는 것이 아닌, 새로운 요청객체를 생성합니다.

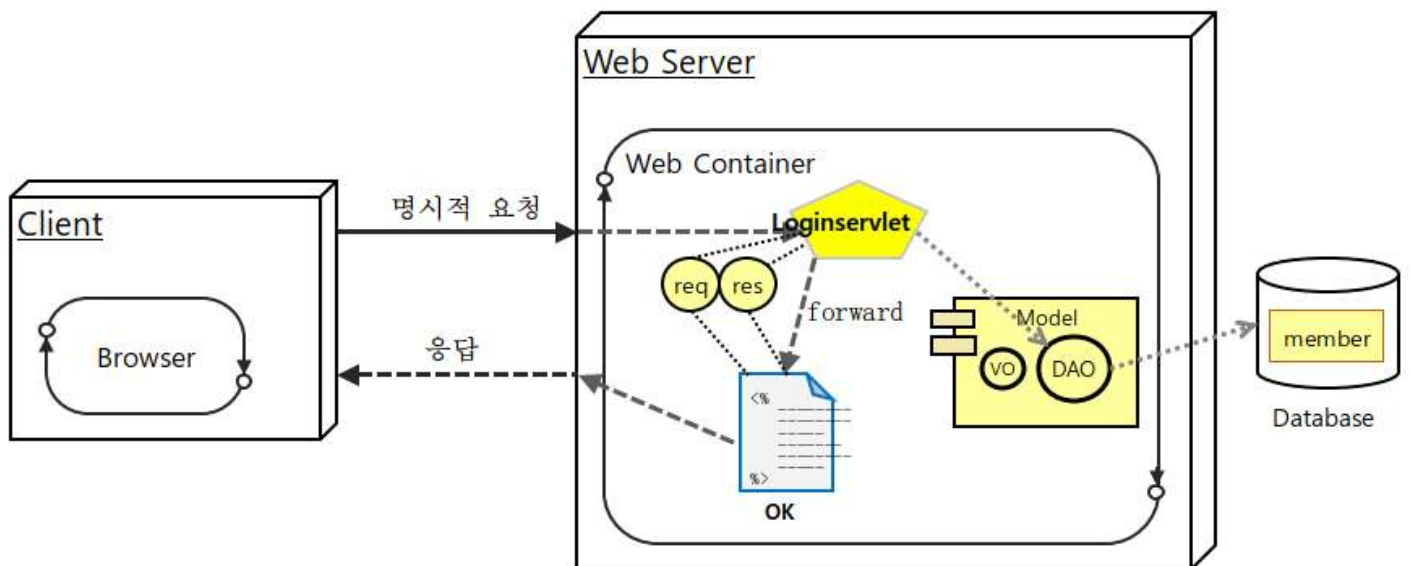
- 요청을 처리한 다음 새로운 요청으로 작업을 해야할 경우에 사용합니다.

ex) 회원가입 요청 처리 후 단순히 메인페이지로 안내해야 할 경우.

response



forward



## 2) sample

=> WebContent > ActionTag 폴더 생성

sample 1) 강제 페이지 이동(response.sendRedirect)

# forward\_ex01.jsp ! 📄 [Ctrl + F11]

# forward\_ex02.jsp

# forward\_ex01.jsp ! 📄 [Ctrl + F11]

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!--
    * jsp action tag 사용법
    - <jsp: 사용할 가능 속성=""...></jsp:사용할 기능>
    - 닫는 태그를 사용할 때 태그 내부에 추가적으로 종속된 태그가 없다면
      닫는 태그를 생략하고 열린 태그 끝부분에 />로 마감할 수 있습니다.
    ex) <jsp:xxx 속성... />

--%>

<jsp:forward page="forward_ex02.jsp"/>

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h3>forward_ex01페이지입니다.</h3>
    페이지 강제 이동에 의해 이 페이지의 텍스트는 보실 수 없습니다.
</body>
</html>
```

# forward\_ex02.jsp

```
<%@ 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>
    <h3>강제 이동된 페이지 forward_ex02.jsp입니다.</h3>
</body>
</html>
```

## // JSP Tag

```
<%
    response.sendRedirect("forward_ex02.jsp");
%>
```

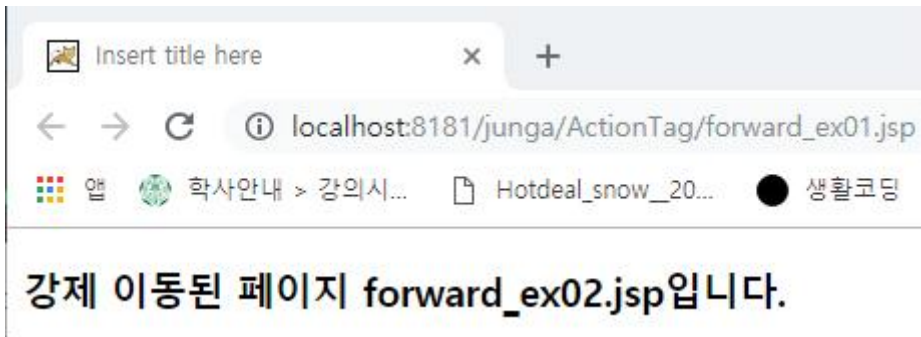
## // JSP Action Tag

```
<jsp:forward page="forward_ex02.jsp"></jsp:forward>
```

or

```
<jsp:forward page="forward_ex02.jsp"/> # 종속태그 없으면,
```





sample 2) include

# include\_ex01.jsp !  [Ctrl + F11]

# include\_ex02.jsp

sample 2) include

# include\_ex01.jsp !  [Ctrl + F11]

```
<%@ 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>
    <%--
        # 디렉티브 태그의 include:
        ex) <%@ include file="삽입할 페이지" %>
        - 페이지 내부에서 공유하면서 써야할 전역적인 변수나 메서드를
        미리 다른 파일에 선언해두고 디렉티브 인클루드로 삽입하여 사용합니다.

        # 액션태그의 include:
        ex) <jsp:include page="삽입할 페이지" />
        - 일반적인 UI만 복사하고 싶을 때 주로 사용합니다.
    --%>
    <h3>안녕하세요~~ 여기는 1번 페이지입니다!</h3>

    <jsp:include page="include_ex02.jsp" />

    <h3>다시 여기는 1번 페이지 입니다.</h3>
</body>
</html>
```

# include\_ex02.jsp

```
<%@ 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>
    <h4>반갑습니다. 여기는 2번 페이지 입니다. :)</h4>
</body>
</html>
```

← → ↺ ⓘ localhost8181/junga/ActionTag/include\_ex01.jsp

 앱  학사안내 > 강의시...  Hotdeal\_snow\_20...  생활코딩

안녕하세요~~ 여기는 1번 페이지입니다!

반갑습니다. 여기는 2번 페이지 입니다. :)

다시 여기는 1번 페이지 입니다.

sample 3) param

# param\_ex01.jsp !  [Ctrl + F11]

# param\_ex02.jsp

# param\_ex03.jsp ==> 추가로 parameter 전달 가능!

# param\_ex01.jsp !  [Ctrl + F11]

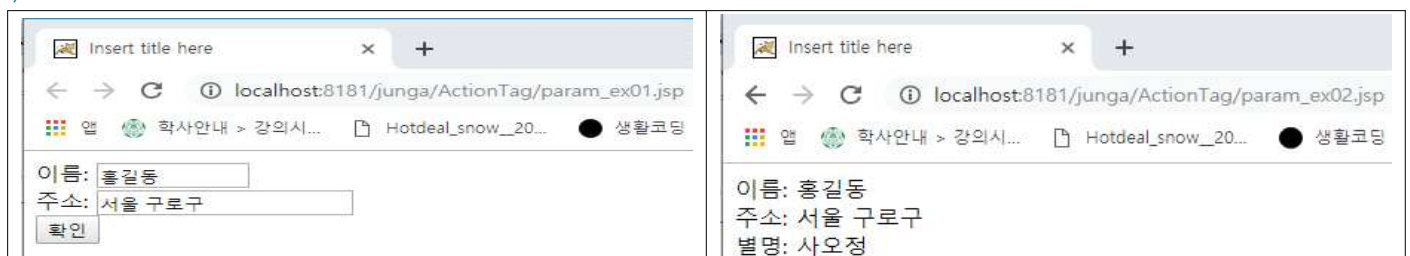
```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <form action="param_ex02.jsp" method="post">
        이름: <input type="text" name="name" size="10"> <br>
        주소: <input type="text" name="addr" size="20"> <br>
        <input type="submit" value="확인"/>
    </form>
</body>
</html>
```

# param\_ex02.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<% request.setCharacterEncoding("utf-8");%>
<jsp:forward page="param_ex03.jsp">
    <jsp:param name="nick" value="사오정"/>
</jsp:forward>
```

# param\_ex03.jsp ==> 추가로 parameter 전달 가능!

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    String name = request.getParameter("name");
    String address = request.getParameter("addr");
    String nickName = request.getParameter("nick");
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    이름: <%= name %><br>
    주소: <%= address %><br>
    별명: <%= nickName %><br>
</body>
</html>
```



sample 4) sendRedirect (req값 유지 ×)

=> Send vs Forward 폴더 생성

# send\_ex01.jsp ! 📄 [Ctrl + F11]

# send\_ex02.jsp

# send\_ex03.jsp

# send\_ex01.jsp ! 📄 [Ctrl + F11]

```
<%@ 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>
    <form action="send_ex02.jsp" method="post">
        ID: <input type="text" name="id" size="10"><br>
        PW: <input type="text" name="pw" size="10"><br>
        <input type="submit" value="로그인">
    </form>
</body>
</html>
```

# send\_ex02.jsp

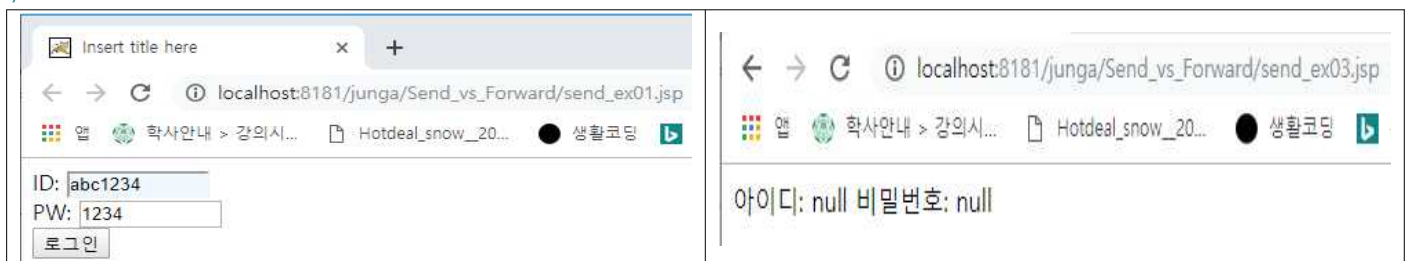
```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    response.sendRedirect("send_ex03.jsp");
%>
```

# send\_ex03.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    request.setCharacterEncoding("utf-8");

    String id = request.getParameter("id");
    String pw = request.getParameter("pw");

%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    아이디: <%= id %>
    비밀번호: <%= pw %>
</body>
</html>
```



sample 5) forward (req값 유지 ○)

=> Send vs Forward 폴더 생성

# forward\_ex01.jsp !  [Ctrl + F11]

# forward\_ex02.jsp

# forward\_ex03.jsp

# forward\_ex01.jsp !  [Ctrl + F11]

```
<%@ 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>
    <form action="forward_ex02.jsp" method="post">
        ID: <input type="text" name="id" size="10"><br>
        PW: <input type="password" name="pw" size="10"><br>
        <input type="submit" value="로그인">
    </form>
</body>
</html>
```

# forward\_ex02.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<jsp:forward page="forward_ex03.jsp"/>
```

# forward\_ex03.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    request.setCharacterEncoding("utf-8");

    String id = request.getParameter("id");
    String pw = request.getParameter("pw");

%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    아이디: <%= id %>
    비밀번호: <%= pw %>
</body>
</html>
```



## 6. JSP - Java bean

### 1) 개념

#### \* 자바빈 (JAVA Bean)

- 자바빈이란 JAVA언어의 데이터(변수)와 기능(메서드)으로 이루어진 클래스입니다.
- 자바빈은 데이터를 저장하는 변수, 데이터를 읽어오는 메서드(getter), 데이터를 저장할 때 사용되는 메서드(setter)로 이루어져 있습니다.
- 자바빈은 데이터베이스와의 반복적인 작업을 효율적으로 처리하기 위해 사용됩니다.
- JSP에서는 액션태그를 사용하여 자바빈을 사용할 수 있는 방법이 있습니다.
- JSP 액션태그로 자바빈 사용하는 방법

ex) <jsp:useBean id="자바빈 객체 이름" class="자바빈이 위치한 실제 경로" scope="객체를 사용할 범위" />

1. id - JSP 페이지에서 자바빈 객체에 접근할 때 사용할 이름을 지정함.
2. class - 패키지 이름을 포함한 자바빈 클래스의 완전한 경로를 입력함.
3. scope - 자바빈 객체를 저장할 영역을 지정함.

- a. page: 하나의 JSP페이지를 처리할 때 사용되는 영역.
- b. request: 하나의 요청을 처리할 때 사용되는 영역.
- c. session: 하나의 웹 브라우저와 관련된 영역.
- d. application: 하나의 웹 어플리케이션과 관련된 영역.

#### - JSP 액션태그로 setter와 getter메서드를 사용할 수 있습니다.

##### 1. setter 사용 방법

ex) <jsp:setProperty name="자바빈 id" property="자바빈 클래스의 변수명" value="할당할 값" />

- a. name: 값을 변경할 자바빈 객체의 이름을 지정합니다. useBean태그에서 id속성에 지정한 값을 그대로 사용합니다.
- b. property: 값을 지정할 프로퍼티의 이름을 지정합니다. 자바빈 클래스의 변수명을 적어줍니다.
- c. value: 프로퍼티의 값을 지정합니다. 표현식이나 EL도 사용할 수 있습니다.

##### 2. getter 사용 방법

ex) <jsp:getProperty name="자바빈 id" property="변수명" />

- a. name: useBean태그에서 id속성에 지정한 값을 사용합니다.
- b. property: 출력할 프로퍼티의 이름을 지정합니다. 자바빈 클래스의 변수명을 적습니다.

## 2) Sample

### Sample 1>

```
=> JspObj> java_bean 폴더 생성
# User.java      ! 자바 파일 생성. 패키지명: kr.co.koo
# bean_make.jsp ! 🖱️ [Ctrl + F11]
# bean_use.jsp
```

Case 1> 일반적인 java 클래스 활용 방법 => new 객체 생성

```
# User.java      ! 자바 파일 생성. 패키지명: kr.co.koo
```

```
package kr.co.koo;
```

```
public class User {
```

```
/*
 * 1. 자바빈 클래스는 데이터베이스와의 반복작업을 처리하기 위한 클래스입니다. 2. 보통 데이터베이스의 컬럼(변수)과 1:1로 매칭하여
 * 필드(멤버변수)를 선언합니다. 3. 데이터베이스 정보보호를 위해 변수들을 은닉(캡슐화)시킵니다.
 */
```

```
private String id;
private String pw;
private String name;
private String email;
```

```
// 자바빈 클래스는 일반적으로 기본생성자 1개와
// 모든 멤버변수를 초기화하는 생성자를 1개 선언합니다.
```

```
// 기본 생성자
public User() { // Ctrl + Space> user()
    // TODO Auto-generated constructor stub
}
```

```
// 모든멤버변수 초기화 생성자
public User(String id, String pw, String name, String email) {
    // Source탐> Generate constructor Using field
    super();
    this.id = id;
    this.pw = pw;
    this.name = name;
    this.email = email;
}
```

```
// getters(), setters() 함수 생성
// Source탐> Generate Getters and Setters> Select All 체크
public String getId() {
    return id;
}
```

```
public void setId(String id) {
    if(id.length() < 6) {
        System.out.println("아이디는 6글자이상으로 해주세요.");
    }else {
        this.id = id;
    }
}
```

```
public String getPw() {
    return pw;
}
```

```
public void setPw(String pw) {
    this.pw = pw;
}
```

```
public String getName() {
    return name;
}
```

```
public void setName(String name) {
```

```

        this.name = name;
    }

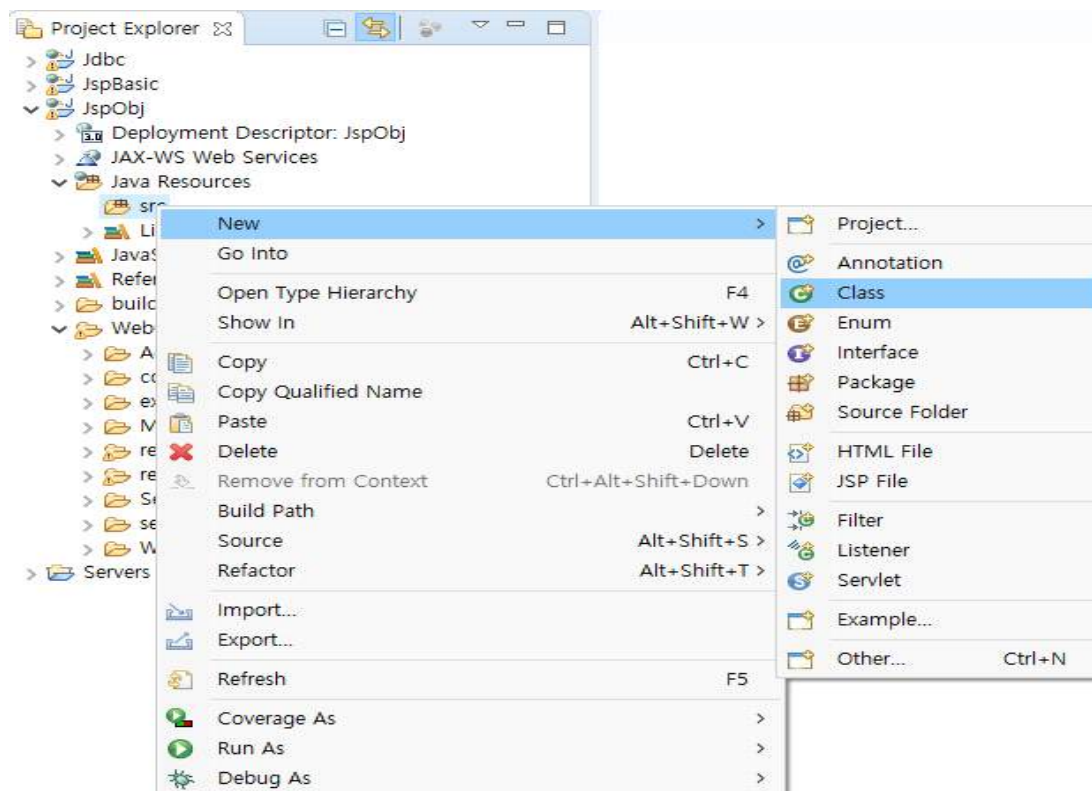
    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}

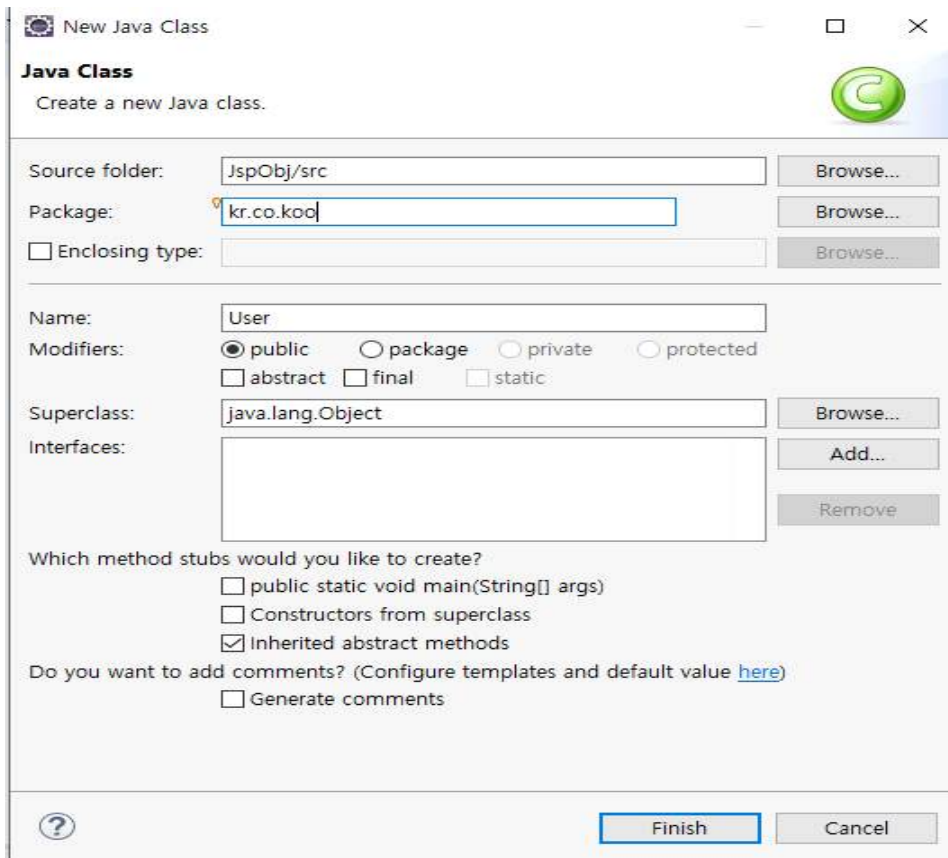
```

cf. User.java 만드는 방법

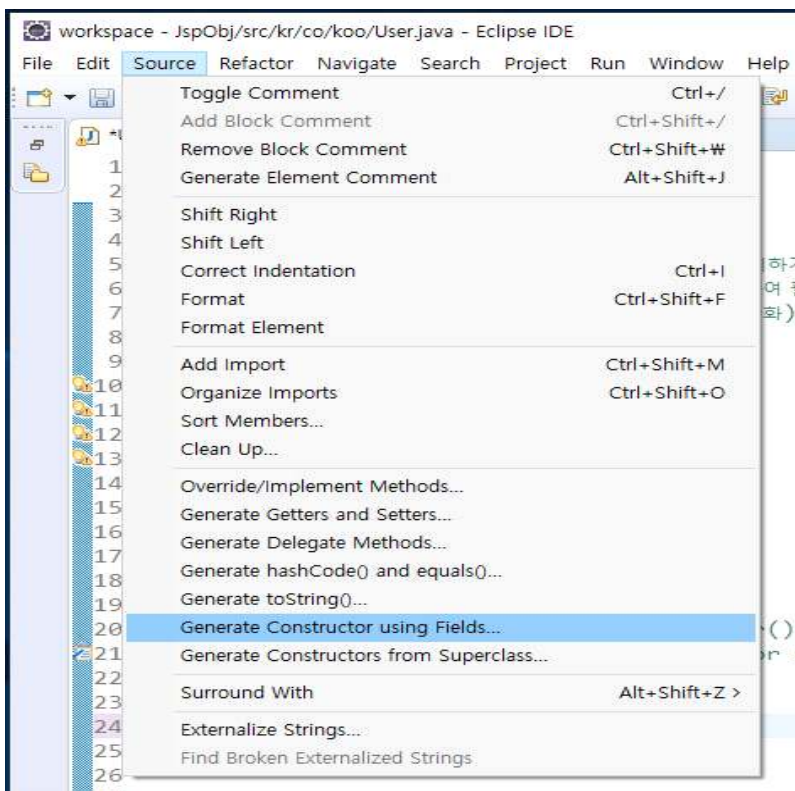
### ① class 파일 생성



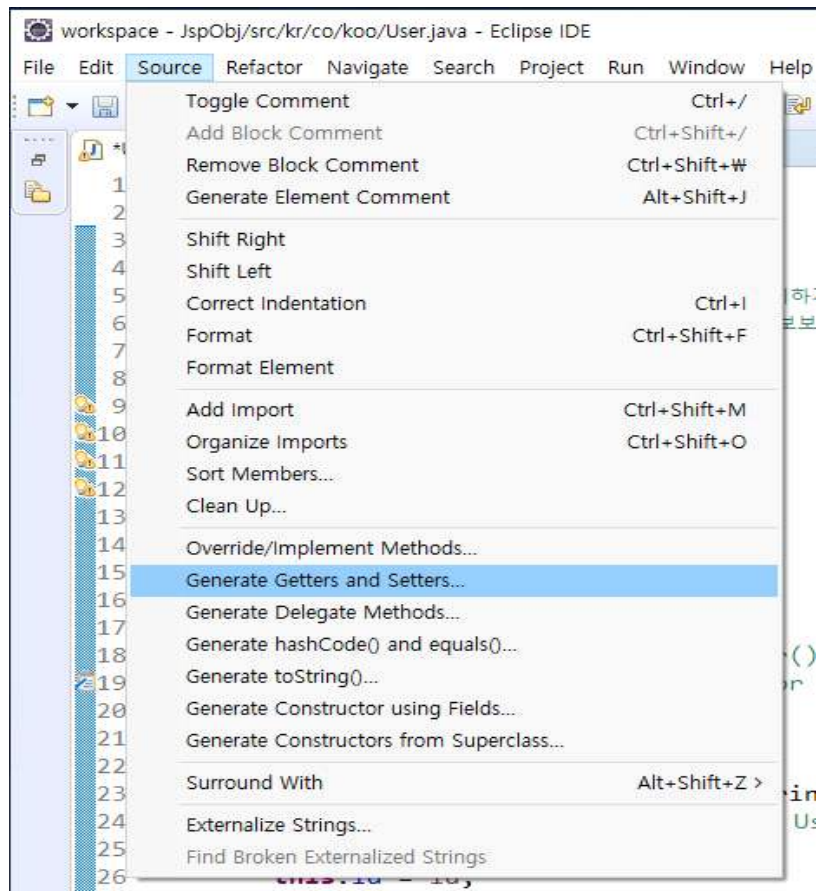




## ② 생성자 만들기



## ②-2 getter(), setter() 생성



### # bean\_make.jsp ! 📄 [Ctrl + F11]

```
<%@page import="kr.co.koo.User"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>

<%
    User user = new User();

    user.setId("abc1234");
    user.setName("이순신");

    request.setAttribute("user", user);
%>
<jsp:forward page="bean_use.jsp" />
<%-- forward이므로 전달 페이지에 객체 유지됨. --%>
```

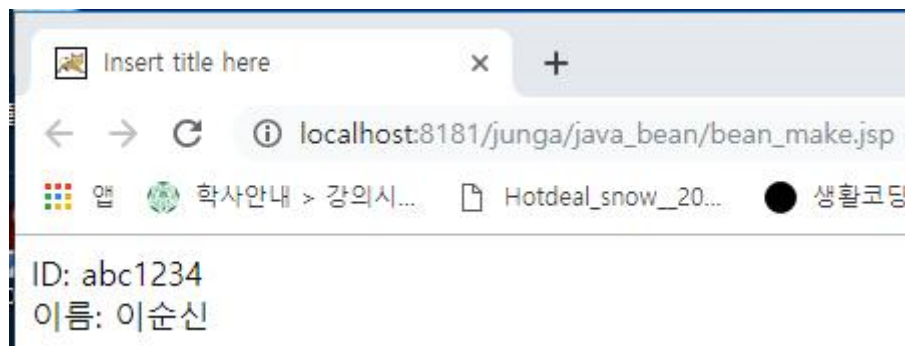
## # bean\_use.jsp

```
<%@page import="kr.co.koo.User"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    // User user = new User(); // 이렇게 하면 null만 뜸! 다시 객체를 또 생성했으므로... request객체에
    담아줘야...

    User user = (User) request.getAttribute("user");

    String id = user.getId();
    String name = user.getName();

%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    ID: <%= id %> <br>
    이름: <%= name %> <br>
</body>
</html>
```



## Case 2> JSP Action 태그로 javaBean 이용

### # bean\_make.jsp ! [Ctrl + F11]

```
<%-- <%@page import="kr.co.koo.User"%> --%>
<jsp:useBean id="user" class="kr.co.koo.User" scope="request" />
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>

<%
    // javaBean으로 대체
    User user = new User();

    user.setId("abc1234");
    user.setName("이순신");

    // request.setAttribute("user", user);
%>
<jsp:forward page="bean_use.jsp" />
<%-- forward이므로 전달 페이지에 객체 유지됨. --%>
```

=> 주석 내용이 JSP Action 태그를 이용해 1줄로 바뀜.

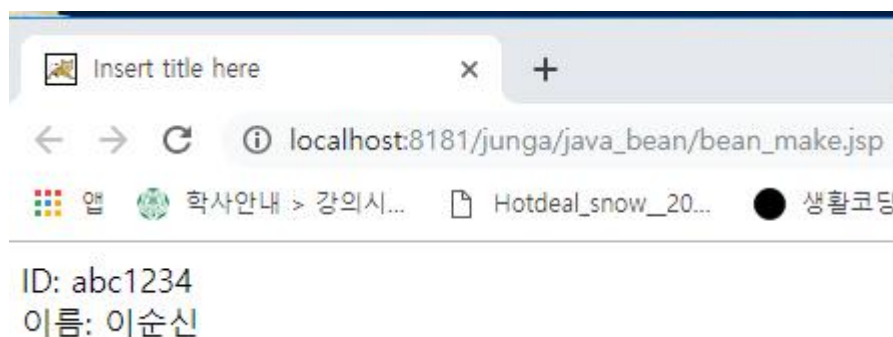
### # bean\_use.jsp

```
<%-- <%@page import="kr.co.koo.User"%> --%>
<jsp:useBean id="user" class="kr.co.koo.User" scope="request" />
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    // User user = new User(); // 이렇게 하면 null만 뜸! 다시 객체를 또 생성했으므로... request객체에
    // User user = (User) request.getAttribute("user");

    String id = user.getId();
    String name = user.getName();

%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    ID: <%= id %> <br>
    이름: <%= name %> <br>
</body>
</html>
```

=> 주석 내용이 JSP Action 태그를 이용해 1줄로 바뀜.



## Sample 2> javaBean 2

```
# User.java      ! DB 활용 클래스
# bean_form.jsp ! 🖱 [Ctrl + F11]
# bean_join.jsp  ! setter(), getter()
```

Case 1> 가장 일반적인 방법 => new 객체 생성

```
# bean_form.jsp ! 🖱 [Ctrl + F11]
```

```
<%@ 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>
<form action="bean_join.jsp" method="post">
    <!-- 'input type name의 값' == 'java bean 클래스 변수 값'
        private String id;
        private String pw;
        private String name;
        private String email;
    --%>
    <table border="1" cellpadding="0" cellspacing="0">
        <tr>
            <td>아이디</td>
            <td><input type="text" name="id" size="10"></td>
            <td>비밀번호</td>
            <td><input type="password" name="pw" size="10"></td>
        </tr>
        <tr>
            <td>이름</td>
            <td><input type="text" name="name" size="10"></td>
            <td>이메일</td>
            <td><input type="text" name="email" size="10"></td>
        </tr>
        <tr align="center">
            <td colspan="4"><input type="submit" value="회원가입"></td>
        </tr>
    </table>
</form>
</body>
</html>
```

# bean\_join.jsp ! setter(), getter()

```
<%@page import="kr.co.koo.User"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<% request.setCharacterEncoding("utf-8"); %>

<%
    String id = request.getParameter("id");
    String pw = request.getParameter("pw");
    String name = request.getParameter("name");
    String email = request.getParameter("email");

    User user = new User();
    user.setId(id);
    user.setPw(pw);
    user.setName(name);
    user.setEmail(email);
%>

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
아이디: <%= user.getId() %> <br>
비밀번호: <%= user.getPw() %> <br>
이름: <%= user.getName() %> <br>
이메일: <%= user.getEmail() %> <br>
</body>
</html>
```

## Case 2> Jsp - ActionTag 사용 - javaBean 1

### # bean\_join.jsp

```
<%-- <%@page import="kr.co.koo.User"%> --%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" trimDirectiveWhitespaces="true"
    errorPage=" ../exception/error_page02.jsp"%>
<% request.setCharacterEncoding("utf-8"); %>

<%-- trimDirectiveWhitespaces="true" : 소스 보거나 빈 공간 없애주는 기능--%>
<%
    String id = request.getParameter("id");
    String pw = request.getParameter("pw");
    String name = request.getParameter("name");
    String email = request.getParameter("email");
    /*
    User user = new User();
    user.setId(id);
    user.setPw(pw);
    user.setName(name);
    user.setEmail(email);
    */
%>
<%-- 위 /**/ => jsp Tag로 대체 --%>

<jsp:useBean id="user" class="kr.co.koo.User" />

<jsp:setProperty name="user" property="id" value="<%=id %>" />
<jsp:setProperty name="user" property="pw" value="<%=pw %>" />
<jsp:setProperty name="user" property="name" value="<%=name %>" />
<jsp:setProperty name="user" property="email" value="<%=email %>" />

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
아이디: <%= user.getId() %> <br>
비밀번호: <%= user.getPw() %> <br>
이름: <jsp:getProperty name="user" property="name" /> <br>
이메일: <jsp:getProperty name="user" property="email" /> <br>
</body>
</html>
```

### Case 3> Jsp - ActionTag 사용 - javaBean 2

#### # bean\_join.jsp

```
<%-- <%@page import="kr.co.koo.User"%> --%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" trimDirectiveWhitespaces="true"
    errorPage=" ../exception/error_page02.jsp"%>
<% request.setCharacterEncoding("utf-8"); %>

<%
    /*
    (**)
    String id = request.getParameter("id");
    String pw = request.getParameter("pw");
    String name = request.getParameter("name");
    String email = request.getParameter("email");
    */
    /*
    User user = new User();
    user.setId(id);
    user.setPw(pw);
    user.setName(name);
    user.setEmail(email);
    */
%>
<%-- 위 /**/ => jsp Tag로 대체 --%>

<%--
    파라미터의 name값과 자바빈클래스의 변수명이 일치할 경우
    액션태그 setProperty의 속성 property값을 "*"로 지정할 경우
    자동으로 파라미터를 읽어서 자바빈 클래스에 저장합니다.
--%>

<%-- (**) 대신 한줄로! --%>
<jsp:useBean id="user" class="kr.co.koo.User" />
<%--scope 생략시 default로 page가 입력됨 --%>
<jsp:setProperty name="user" property="*" />

<%-- (**)
<jsp:setProperty name="user" property="id" value="<%=id %>" />
<jsp:setProperty name="user" property="pw" value="<%=pw %>" />
<jsp:setProperty name="user" property="name" value="<%=name %>" />
<jsp:setProperty name="user" property="email" value="<%=email %>" />
--%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
아이디: <%= user.getId() %> <br>
비밀번호: <%= user.getPw() %> <br>
이름: <jsp:getProperty name="user" property="name" /> <br>
이메일: <jsp:getProperty name="user" property="email" /> <br>
</body>
</html>
```



## 7. JSP - Servlet

### 1) 개념

#### \* Servlet 특징

=> 사용목적 : jsp context루트 이하 파일 링크 주소 숨기기 위함

1. 동적 웹어플리케이션 컴포넌트
2. .java 확장자
3. 클라이언트의 요청에 동적으로 작동하고, 응답은 html을 이용.
4. java thread를 이용하여 동작.
5. MVC패턴에서 Controller로 이용됨.

#### \* URL-Mapping

- URL 매핑을 하지 않으면 URL주소가 너무 길어지고, 경로가 노출되어 보안에 위험이 생기기 때문에 URL 매핑을 사용하여 그 문제들을 해결합니다.

- `http://localhost:8181/JSPBasic/servlet/kr.co.koo.HelloWorld`

----> `http://localhost:8181/JSPBasic/HelloWorld`

- 사용 방법

1. 아노테이션 이용, 클래스 선언부 바로 위에 작성.

ex) `@WebServlet("/HelloWorld")`

2. web.xml 설정파일 수정.

ex)

```
<servlet>
    <servlet-name>helloworld</servlet-name>
    <servlet-class>kr.co.koo.HelloWorld</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>helloworld</servlet-name>
    <url-pattern>/HWorld</url-pattern>
</servlet-mapping>
```

#### \* Servlet 작동 순서

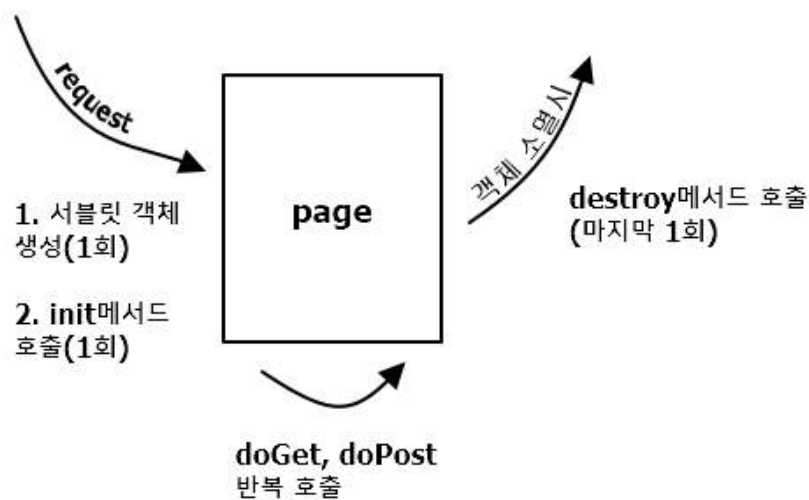
- 클라이언트에서 요청(request)이 들어오면 서버에서는 servlet 컨테이너를 만들고, 요청이 있을때 thread와 Servlet 객체가 생성됩니다.

## \* Servlet의 생명주기(LifeCycle)

- Servlet의 장점은 빠른 응답 속도입니다.
- Servlet은 최초 요청시에 객체가 만들어져 메모리에 로딩되고, 이후 추가 요청시에는 기존의 객체를 재활용하게 됩니다. 따라서 동작속도가 매우 빠릅니다.

=> jsp 페이지 요청시 1번만 만들어짐. heap에 유지 재활용. 멤버변수 초기화 안됨!

1. Servlet 객체를 생성 (최초 한번)
2. Init() 메서드 호출(최초 한번)
3. doGet(), doPost(), service() 호출 (요청시 매번)
4. destroy() 호출 (마지막 한번) - 자원이 해제될 시 호출(Servlet코드를 수정, 서버 재가동할 시)



## \* 웹 어플리케이션 생명주기 (ServletContextListener)

- 웹 어플리케이션에는 프로그램의 생명주기를 감시하는 리스너가 있습니다.
  - 리스너의 해당 메서드가 웹 어플리케이션의 시작과 종료시에 호출됩니다.
1. 시작시에는 contextInitialized()
  2. 종료시에는 contextDestroyed()

## \* 서블릿 초기화 파라미터 (ServletConfig)

- 특정한 서블릿이 생성될 때 초기에 필요한 데이터들이 있습니다.
- 이러한 데이터들을 초기화 파라미터라고 하며, 아노테이션으로 지정하는 방법과, web.xml파일에 기술하는 방법이 있습니다.

## \* 데이터 공유(ServletContext)

- 여러 서블릿에서 특정 데이터를 공유해야 할 경우 Context Parameter를 이용하여 web.xml파일에 데이터를 기술하고, 여러 서블릿에서 공유하면서 사용할 수 있습니다.

## 2) Sample

Sample 1> 어노테이션 방법

```
# ServletBasic.java (kr.co.koo) ! [Ctrl + F11]
http://localhost:8181/jung/servlet/kr.co.koo.ServletBasic
=> http://localhost:8181/jung/apple # @WebServlet("/apple")
=> http://localhost:8181/jung/fruit/apple # @WebServlet("/fruit/apple") 서버릿 객체 생성!
# res_basic01.jsp (response) ! [Ctrl + F11] ServletBasic.java> doPost에 소스 연결 후
```

Case 1> ServletBasic.java에 어노테이션 설정 후 실행해보기

=> 설정한 annotation에 따라 링크가 다르게 나옴

# ServletBasic.java (kr.co.koo) ! [Ctrl + F11] / File> New> class (아직 Servlet 안 씬.)

```
package kr.co.koo;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

// 서버릿 클래스 생성 방법: HttpServlet 클래스를 상속받습니다.

// 서버릿 url매핑시 어노테이션 방법과 web.xml 수정 방식을 동시에 적용시킬 경우
// web.xml에 적용한 url로 인식합니다.

// 1. 어노테이션 이용 방법: 클래스 선언부 바로 위에 작성.
@WebServlet("/apple")
@WebServlet("/fruit/banana")
public class ServletBasic extends HttpServlet { // ctrl + 1 > Add generate serial version ID

    private static final long serialVersionUID = 5195390950896920855L;

    // 기본 생성자 선언.
    public ServletBasic() { // Ctrl + Space > ServletBasic
        System.out.println("서블릿 객체 생성!");
    }

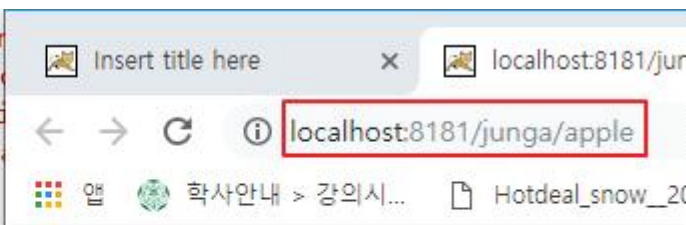
    // 일반적으로 2개의 메서드를 오버라이딩 합니다. (doGet, doPost)
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException { // doGet> Ctrl+Space
        System.out.println("get 요청시 이 메서드가 호출됩니다.");
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException { // doPost> Ctrl+Space
        System.out.println("post 요청시 이 메서드가 호출됩니다.");
    }
}
```

```

정보: Starting ProtocolHandler ["http-...
5월 06, 2019 9:29:40 오후 org.apache.coy
정보: Starting ProtocolHandler ["ajp-n...
5월 06, 2019 9:29:40 오후 org.apache.cata
정보: Server startup in 776 ms
서블릿 객체 생성!
get 요청시 이 메서드가 호출됩니다.

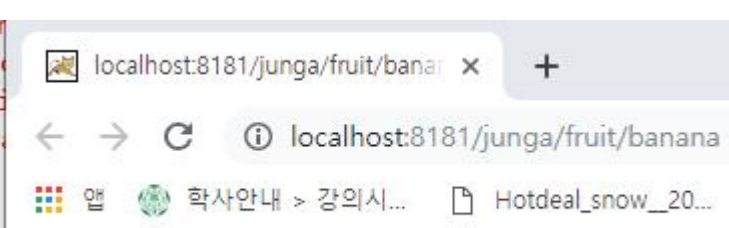
```



```

정보: Starting ProtocolHandler [ http-...
5월 06, 2019 9:30:18 오후 org.apache.coy
정보: Starting ProtocolHandler ["ajp-n...
5월 06, 2019 9:30:18 오후 org.apache.cata
정보: Server startup in 759 ms
서블릿 객체 생성!
get 요청시 이 메서드가 호출됩니다.

```



## Case 2> WebContent> res\_basic01.jsp 소스와 Servlet 연동

# res\_basic01.jsp ! [Ctrl + F11]

```

<%@ 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>
    <!-- <form action="res_basic02.jsp" method="post"> -->
    <form action=" ../apple" method="post">
        당신의 나이는?? <input type="text" name="age" size="5">&nbsp;  
        <input type="submit" value="확인">
    </form>
</body>
</html>

```

# ServletBasic.java

```

package kr.co.koo;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

```

```

// 서블릿 클래스 생성 방법: HttpServlet 클래스를 상속받습니다.

// 서블릿 url매핑시 어노테이션 방법과 web.xml 수정 방식을 동시에 적용시킬 경우
// web.xml에 적용한 url로 인식합니다.

// 1. 어노테이션 이용 방법: 클래스 선언부 바로 위에 작성.
@WebServlet("/apple")
@WebServlet("/banana")
public class ServletBasic extends HttpServlet{ // ctrl + 1 > Add generate serial version ID

    private static final long serialVersionUID = 5195390950896920855L;

    // 기본 생성자 선언.
    public ServletBasic() { // Ctrl + Space > ServletBasic
        System.out.println("서블릿 객체 생성!");
    }

    // 일반적으로 2개의 메서드를 오버라이딩 합니다. (doGet, doPost)
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
    IOException { // doGet> Ctrl+Space
        System.out.println("get 요청시 이 메서드가 호출됩니다.");

        response.setContentType("text/html; charset=utf-8");

        // jsp => servlet (html 코드) 변환 과정
        PrintWriter out = response.getWriter();

        out.println("<html>");
        out.println("<head>");
        out.println("<title>서블릿 연습</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h4>get요청시 이 텍스트가 출력됩니다.</h4>");
        out.println("</body>");
        out.println("</html>");

    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
    IOException { // doPost> Ctrl+Space
        System.out.println("post 요청시 이 메서드가 호출됩니다.");

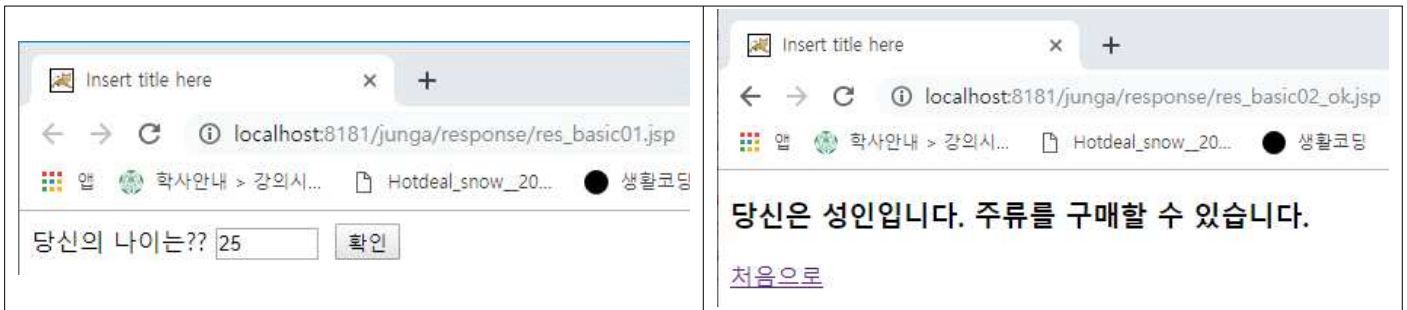
        // response> res_basic02.jsp 내용 복.붙

        String strAge = request.getParameter("age");

        int age = Integer.parseInt(strAge);

        if(age >= 20){
            // response 내장객체가 지원하는 sendRedirect() 메서드는 매개값으로
            // 페이지의 URL을 지정하면 해당 페이지로 강제이동 시킵니다.
            response.sendRedirect("response/res_basic02_ok.jsp");
        } else {
            response.sendRedirect("response/res_basic02_no.jsp");
        }
    }
}

```



## Sample 2> web.xml 설정파일 수정

### # web.xml

```
<servlet>
  <servlet-name>sb</servlet-name>
  <servlet-class>kr.co.koo.ServletBasic</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>sb</servlet-name>
  <url-pattern>/banana</url-pattern>
</servlet-mapping>
```

### # ServletBasic.java (kr.co.koo)

=> http://localhost:8181/jung/banana

### # res\_basic01.jsp (response) ! 🖱️ [Ctrl + F11]

=> <form action="../banana" method="get">

OR

=> <form action="../banana" method="post">

### # web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
version="3.0">
  <display-name>JspObj</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
  <error-page>
    <error-code>404</error-code>
    <location>/exception/error_404.jsp</location>
  </error-page>
  <error-page>
    <error-code>500</error-code>
    <location>/exception/error_500.jsp</location>
  </error-page>
  <exception-type>java.lang.NullPointerException</exception-type>
    <location>/exception/error_null.jsp</location>
  </error-page>
  <servlet>
    <servlet-name>sb</servlet-name>
    <servlet-class>kr.co.koo.ServletBasic</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>sb</servlet-name>
```

```
    <url-pattern>/banana</url-pattern>  
  </servlet-mapping>  
</web-app>
```

**# ServletBasic.java**

=> annotation 전부 주석 처리!

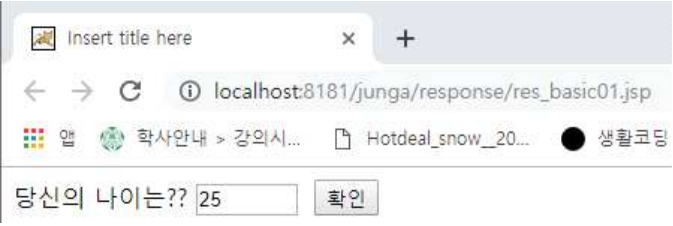
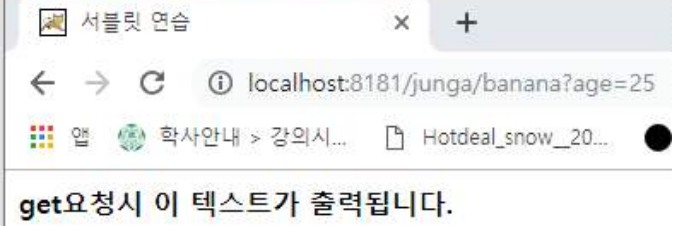
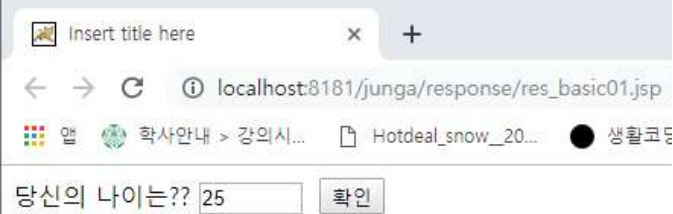
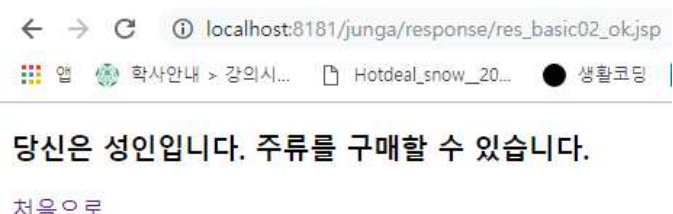
# res\_basic01.jsp (response) ! 🖱️ [Ctrl + F11]

=> <form action="../banana" method="get">

OR

=> <form action="../banana" method="post">

get, post 돌아가면서 실행해보기~!

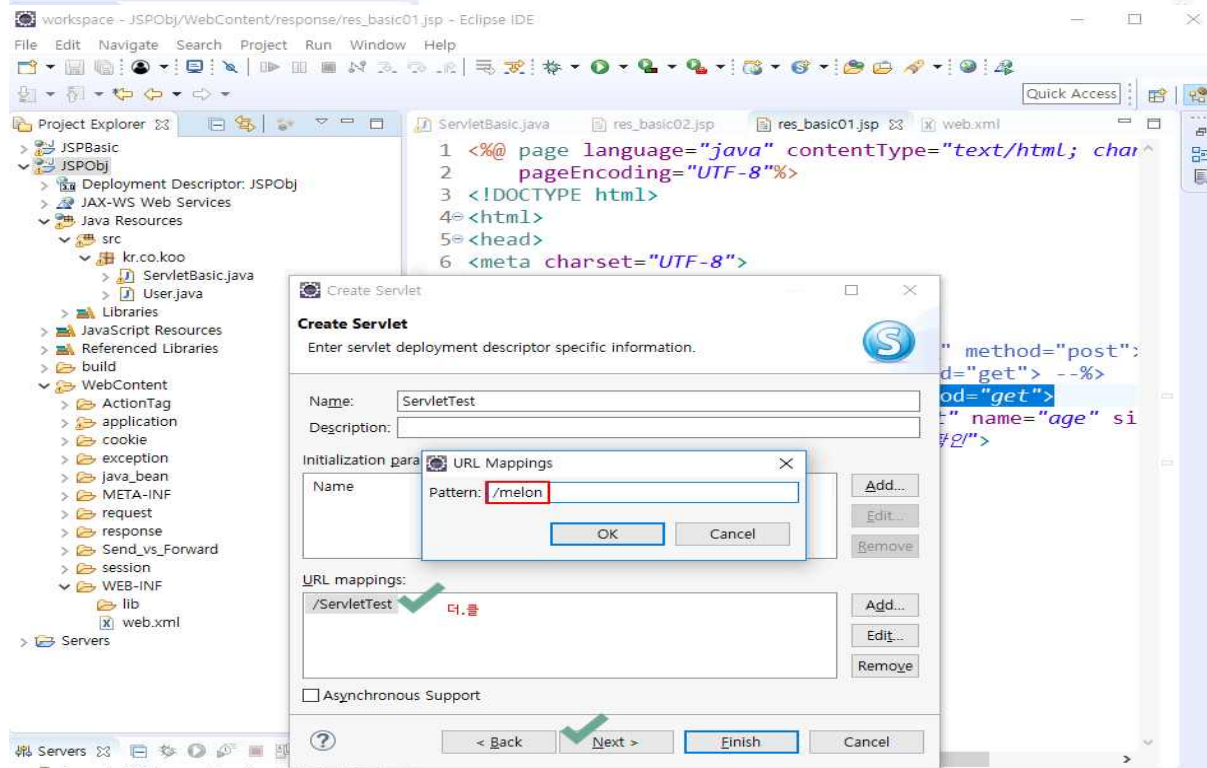
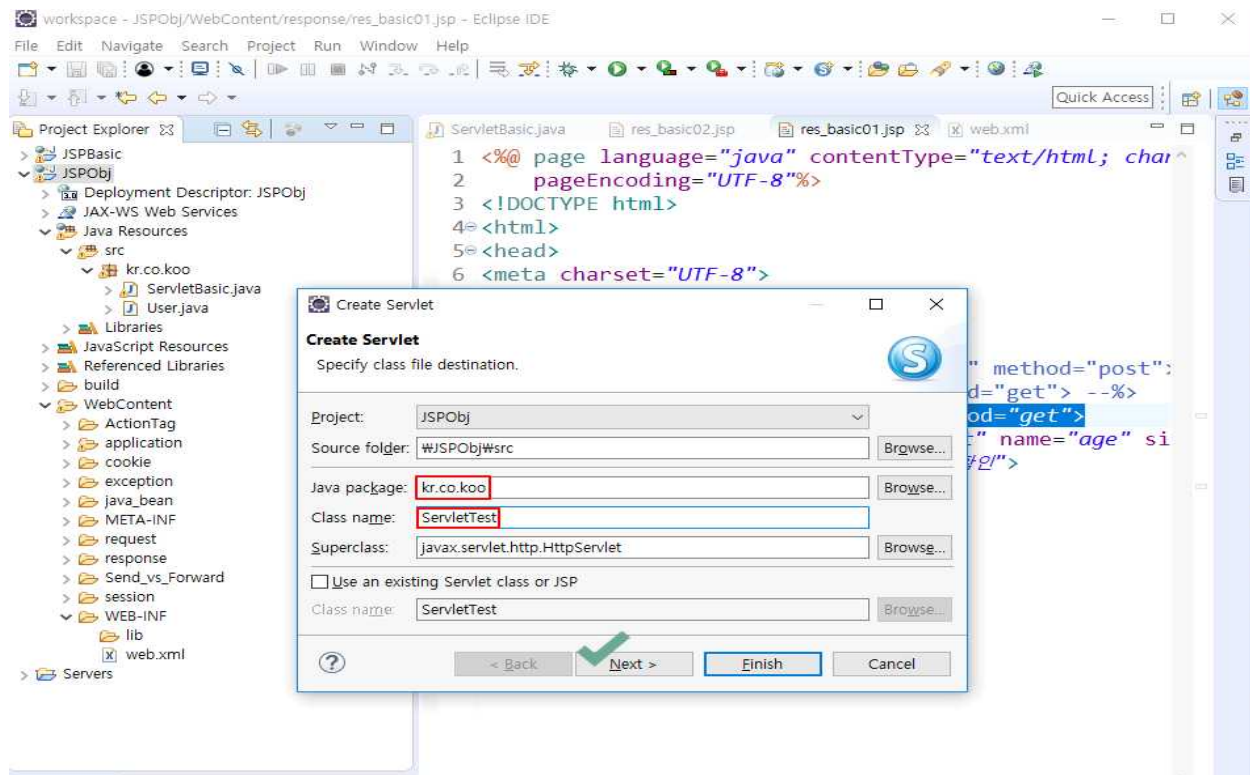
	
	

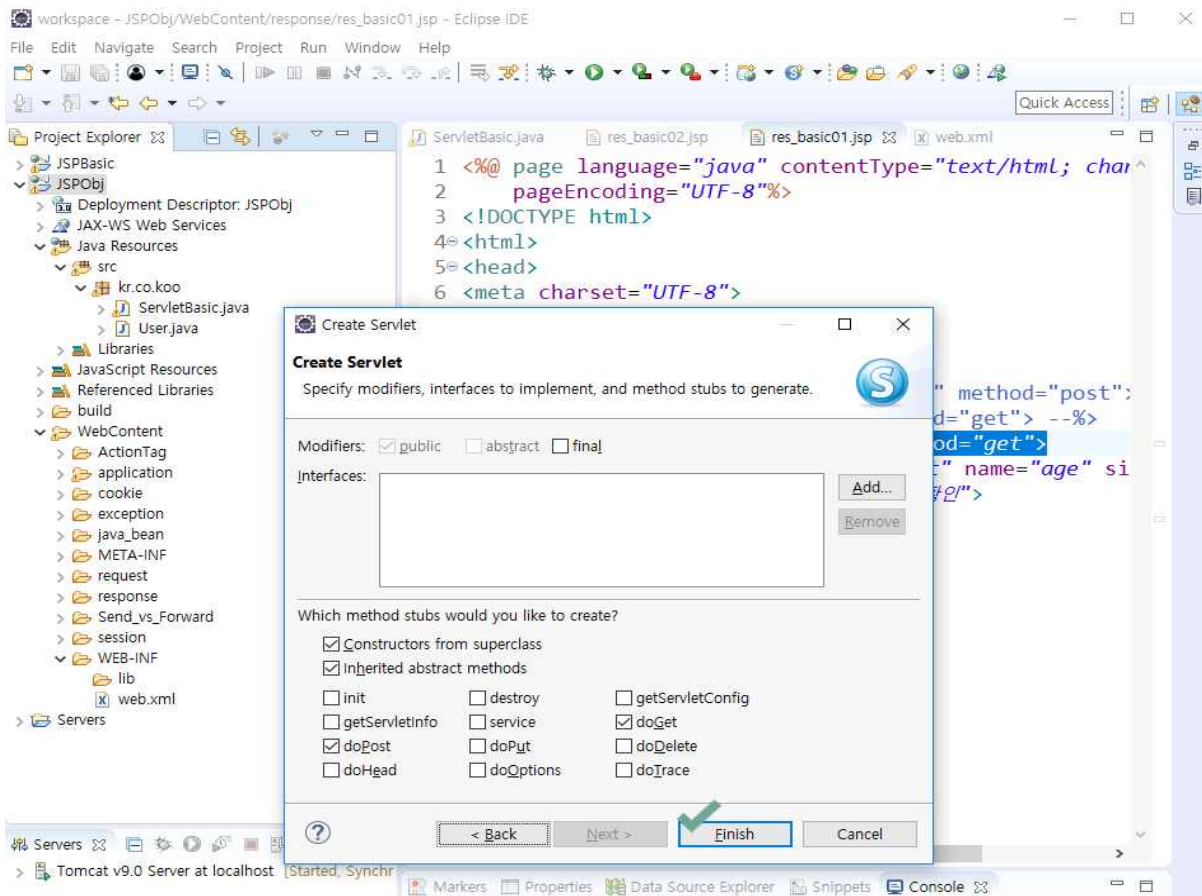


## Sample 3> Servlet 파일 만드는 방법

### # ServletTest.java

new> Servlet





```
# lifecycle_test.jsp (WebContent 밑임! 폴더 안 만듦)
```

workspace - JSPObj/src/kr/co/koo/ServletBasic.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer

- JSPObj
  - Deployment Descriptor: JSPObj
  - JAX-WS Web Services
  - Java Resources
    - src
      - kr.co.koo
        - ServletBasic.java
        - ServletTest.java
        - User.java
  - Libraries
  - JavaScript Resources
  - Referenced Libraries
  - build
  - WebContent
    - ActionTag
    - application
    - cookie
    - exception
    - java Bean
    - META-INF
    - request
    - response
    - Send\_vs\_Forward
    - session
    - WEB-INF
      - lib
      - web.xml
  - Servers

\*ServletBasic.java

```

1 package kr.co.koo;
2
3 import java.io.IOException;
4 import java.io.PrintWriter;
5
6 import javax.servlet.ServletException;

```

Create Servlet

Specify modifiers, interfaces to implement, and method stubs to generate.

Modifiers: ☒ public ☐ abstract ☐ final

Interfaces:

Which method stubs would you like to create?

☒ Constructors from superclass

☒ Inherited abstract methods

☒ init ☒ destroy

☐ getServletConfig

☐ getServletInfo

☒ doGet

☒ doPost

☐ doPut

☐ doDelete

☐ doHead

☐ doOptions

☐ doTrace

Finish

## # LifeCycle.java

```
package kr.co.koo;

import java.io.IOException;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class LifeCycle
 */
@WebServlet("/LifeCycle")
public class LifeCycle extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public LifeCycle() {
        System.out.println("서블릿 객체 생성!!");
        System.out.println("서블릿 객체는 페이지 요청시에 자동 생성 됩니다.");
    }

    /**
     * @see Servlet#init(ServletConfig)
     */
    public void init(ServletConfig config) throws ServletException {
        System.out.println("init 메서드 호출!");
        System.out.println("[init은 서블릿 객체 생성시 최초 1회만 실행됩니다.]");
        System.out.println("해당 페이지가 실행될 때 초기에 실행할 로직이 있다면 이곳에 기술합니다.");
    }

    /**
     * @see Servlet#destroy()
     */
    public void destroy() {
        System.out.println("destroy 메서드 호출!");
        System.out.println("[destroy 메서드는 서블릿 객체가 소멸할 때 마지막 1회 호출합니다.]");
        System.out.println("페이지가 종료되어 객체가 소멸될 때 자원을 해제해야 한다면 이곳에 코드를 기술합니
다.");
    }
}
```

```

/**
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
    System.out.println("doGet 호출!!");
    System.out.println("[get요청이 왔을 때 요청마다 반복호출 됩니다.]");
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
    System.out.println("doPost호출!!");
    System.out.println("[post요청이 왔을 때 요청마다 반복호출 됩니다.]");
}
}

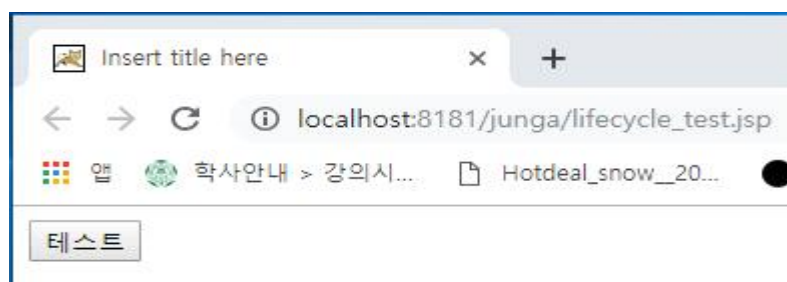
```

#### # lifeCycle\_test.jsp

```

<%@ 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>
    <form action="/junga/LifeCycle" method="post">
        <input type="submit" value="테스트">
    </form>
</body>
</html>

```



## 8. JDBC - DB연동 실습

### 1) SQL

#### \* 테이블과 레코드

- RDBMS에서 데이터를 저장하는 장소를 테이블이라고 합니다.
- 테이블을 데이터들을 저장하며 데이터들의 정보를 갖고 있습니다.
- 테이블의 구조와 관련된 정보를 테이블 스키마(Schema)라고 부릅니다.
- 테이블의 구조는 각각의 정보를 저장하는 컬럼(column)과 컬럼 타입 그리고 컬럼의 길이로 구성됩니다.
- 각 컬럼에 저장된 데이터 값을 레코드(record)라고 부릅니다.
- 레코드, 컬럼 그리고 테이블을 사용해서 데이터를 저장하고 조회하는 등의 작업을 수행하는 것을 데이터베이스 프로그래밍이라고 부릅니다.

#### \* SQL 문법

- SQL문의 특징은 대/소문자를 구분하지 않습니다.

#### # DDL(Data Definition Language)

- CREATE : 테이블이나 인덱스, 뷰 등 데이터베이스 객체를 생성
- DROP : 생성된 데이터베이스 객체를 삭제
- ALTER : 이미 생성된 데이터베이스 객체를 수정
- TRUNCATE : 테이블이나 클러스터의 데이터를 통째로 삭제

☆☆

#### # DML(Data Manipulation Language)

- SELECT : 테이블이나 뷰에 있는 데이터 조회
- INSERT : 데이터를 신규로 생성
- UPDATE : 이미 생성된 데이터를 수정
- DELETE : 데이터를 삭제
- COMMIT : 트랜잭션 처리, 변경된 데이터를 최종 적용
- ROLLBACK : 트랜잭션 처리, 변경된 데이터를 적용하지 않고 이전으로 되돌림

☆☆

#### # DCL(Data Control Language)

- GRANT : 사용자에게 특정 권한을 부여
- REVOKE : 사용자에게 부여된 권한을 회수

#### 1. 데이터베이스 생성

- create database [db이름] default character set utf8;

#### 2. 사용자 계정 생성과 권한 부여

- 관리자 계정의 이름은 'root'입니다.

##### a. 사용자 계정 생성

- create user '[계정명]' identified by '[암호]';

##### b. 권한 부여

- grant [권한 목록] on [데이터베이스이름].[대상] to '[계정]';

#### 3. 테이블 생성 문법

- create table [테이블 이름] (  
[컬럼명1] [컬럼데이터 타입(byte)],  
[컬럼명2] [컬럼데이터 타입(byte)],  
....  
);

- 테이블 생성시 필요한 SQL 데이터 타입

## A. MYSQL 문법

- 문자형 컬럼일 경우 CHAR, VARCHAR
- 정수형 컬럼일 경우 INT
- 날짜형 컬럼일 경우 DATETIME

## B. ORACLE 문법

- 문자형 컬럼일 경우 VARCHAR2
- 정수형 컬럼일 경우 NUMBER
- 날짜형 컬럼일 경우 DATE
- 주요 키(primary key)

- 테이블에 있는 레코드를 조회할 때 특정 값을 이용해서 조회한다면 좀 더 빠르게 레코드를 검색할 수 있습니다.
- 주요 키로 지정된 컬럼은 하나의 테이블에 저장된 모든 레코드가 중복된 값 없이 모두 다른 값을 갖는 레코드를 말합니다.
- 주로 회원 아이디나 게시판 글 번호 등이 주요 키로 지정됩니다.

## 4. INSERT 문

- insert문은 데이터를 저장하는 용도의 쿼리문입니다.
- insert into [테이블명] ([컬럼1], [컬럼2], ... ) values ([값1], [값2], ... );
- 테이블명 뒤에 컬럼명을 명시하지 않으면 전체 컬럼에 대해 값을 지정해야 합니다.

## 5. UPDATE 문

- update 쿼리는 데이터를 수정하는 용도입니다.
- update [테이블명] set [수정할 컬럼]=[수정할 값] where [조건];

## 6. DELETE 문

- delete 쿼리는 데이터를 삭제하는 용도입니다.
- delete from [테이블명] where [조건];

## 7. SELECT 문

- select 쿼리는 데이터베이스로부터 정보를 조회하는 문장입니다.
  - select [컬럼1], [컬럼2], ... from [테이블명] where [조건];
  - 모든 컬럼을 조회하고 싶으면 조회할 컬럼 이름에 \* 를 써줍니다.
- ex) select \* from member;
- where절로 조건을 지정할 때 like를 사용하면 특정 문장을 포함하고 있는지 확인할 수 있습니다.
- ex) select \* from member where name like '홍%';
- > name 컬럼의 값이 '홍\*\*\*\*\*'의 형태를 갖는지 확인

## 8. SELECT 문의 정렬

- 게시판이나 회원목록 등을 출력할 때 이름 순서나 아이디 오름차순 혹은 글 번호순으로 정렬하는 것이 일반적입니다.
- sql 쿼리문에서는 where 조건절 뒤에 order by절을 사용하여 데이터를 정렬합니다.
- 오름차순이면 asc, 내림차순이면 desc를 사용합니다.

## 9. DROP 문

- 테이블을 삭제할 수 있는 쿼리문입니다.
- drop table [테이블 이름];

## 10. commit

- 서버에 입력한 쿼리문들의 상태지점을 저장하는 용도.
  - commit과 rollback은 DML문에서만 작동합니다.
- 모든 작업이 올바르게 된 시점에 commit!

## 11. rollback

- commit된 위치를 기준으로 commit 이후에 작성 쿼리문을 commit한 시점으로 되돌리는 문장.

\* 트랜잭션: 저장 시점 결정

=> commit, rollback

ex>

DML : S, I, U, D => (O) 애나만 commit, rollback 가능!!!

DDL => D, A (x)



## 2) JDBC 이론

### \* JDBC(Java Database Connectivity) 프로그래밍

- JDBC란? 자바 프로그램에서 SQL문을 실행하여 데이터를 관리하기 위한 JAVA API입니다.
- 특징은 다양한 데이터베이스에 대해서 별도의 프로그램을 만들 필요 없이, 해당 데이터베이스의 JDBC를 이용하면 하나의 프로그램으로 데이터베이스를 관리할 수 있습니다.
- 우리는 MySQL을 사용하므로 MySQL용 JDBC를 사용합니다.

☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆

### \* 데이터베이스와 JAVA의 연결 순서

1. JDBC 드라이버 로드.
2. 데이터베이스 Connection 객체 생성.
3. 쿼리문 실행을 위한 Statement 객체 생성.
4. 쿼리문을 실행.
5. ResultSet 객체를 통해 쿼리문 실행 결과값을 소비.
6. Statement 객체 종료.
7. 데이터베이스 Connection 객체 종료.

☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆

### \* DBMS와의 통신을 위한 JDBC 드라이버

- JDBC 드라이버는 DBMS와의 통신을 담당하는 자바 클래스로서 각 DBMS마다 별도의 드라이버가 필요합니다.
- 주요 DBMS의 JDBC 드라이버 클래스
- 1. MySQL: "com.mysql.jdbc.Driver"
- 2. ORACLE: "oracle.jdbc.driver.OracleDriver"

### \* 데이터베이스 식별을 위한 JDBC URL

- 웹이 주소를 구분할 때 URL을 사용하는 것처럼 데이터베이스도 URL을 통해 데이터베이스들을 구분합니다.
- 주요 DBMS의 JDBC URL 패턴
- 1. MySQL: "jdbc:mysql://호스트이름:포트번호/DB이름"
- 2. ORACLE: "jdbc:oracle:thin:호스트이름:포트번호:DB이름"

### \* 데이터베이스 연결을 위한 Connection 객체

- JDBC를 이용해서 데이터베이스를 사용하려면 데이터베이스와 연결된 커넥션을 구해야 합니다.
- java.sql 패키지에 있는 Connection 클래스가 데이터베이스 커넥션을 지원하며 DriverManager 클래스가 제공하는 getConnection() 메서드를 사용하여 커넥션을 구할 수 있습니다.
- getConnection() 메서드에 파라미터 값으로 JDBC URL, DB 사용자 계정명, DB 사용자 암호를 전달하면 메서드는 DB와 연결된 커넥션 객체를 리턴합니다.
- 만일 제대로 객체를 생성하지 못하면 SQLException이 발생하므로 getConnection() 메서드를 사용할 때는 반드시 try ~ catch 구문으로 예외처리를 해줘야 합니다.
- Connection 객체를 다 사용한 뒤에는 반드시 close() 메서드를 호출하여 Connection 객체가 사용한 시스템 자원을 반환해야 합니다. 그렇지 않으면 시스템 자원이 불필요하게 소모되어 커넥션을 구할 수 없는 상황이 발생할 수도 있습니다. ★★★

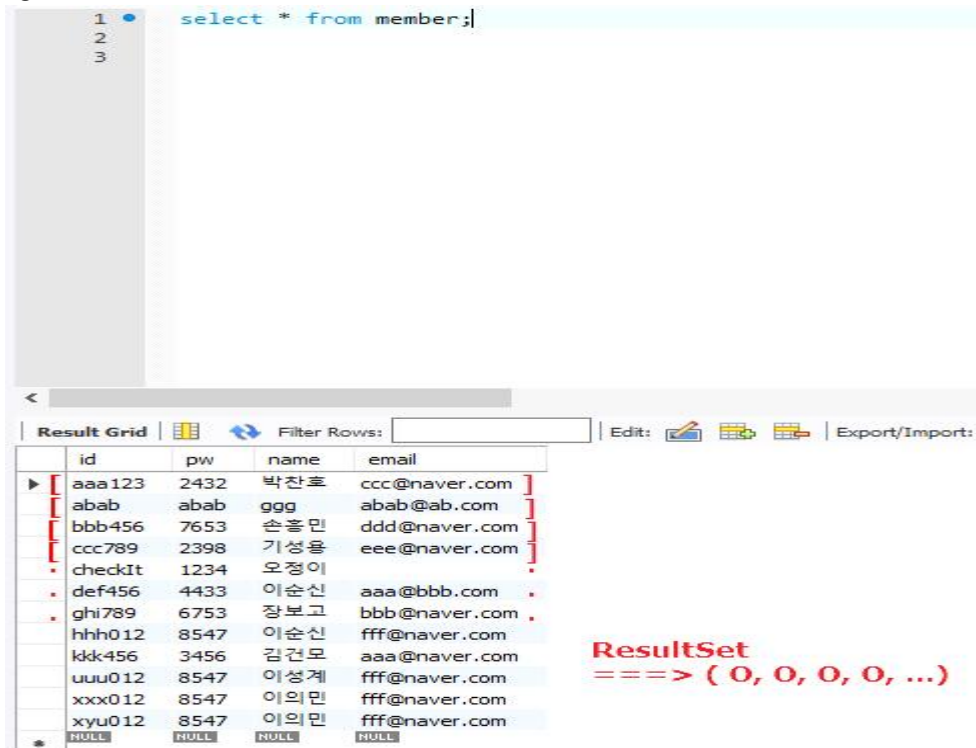


★ 쿼리문을 실행하기 위한 Statement 객체

- Connection 객체를 생성한 후에는 Connection 객체로부터 Statement를 생성하고 쿼리문을 실행할 수 있습니다.
- Statement 객체는 Connection객체의 **createStatement()** 메서드를 이용하여 생성합니다.
- Statement 객체를 사용하면 쿼리문을 실행시킬 수 있습니다.
- 1. **executeQuery(String query):ResultSet** - Select 쿼리문을 실행합니다.
- 2. **executeUpdate(String query):int** - Insert, Update, Delete 쿼리문을 실행합니다.

★ 쿼리 실행 결과 값을 읽어오는 ResultSet 객체

- Statement 객체의 **executeQuery()** 메서드는 Select 쿼리문의 결과를 ResultSet객체에 담아서 리턴합니다.
- 따라서 데이터 조회의 결과값을 ResultSet이 제공하는 메서드를 통해 읽어올 수 있습니다.
- ResultSet 객체가 제공하는 next() 메서드는 Select 쿼리문의 결과값의 존재 여부를 확인하는 메서드입니다.
- ResultSet 주요 메서드
- 1. **getString(String name):String** - 지정한 컬럼 값을 String으로 읽어옴. 파라미터 변수 name에는 DB 테이블의 컬럼이름을 적습니다.
- 2. **getInt(String name):int** - 지정한 컬럼 값을 int 타입으로 읽어옴.
- 3. **getDouble(String name):double** - 지정한 컬럼 값을 double 타입으로 읽어옴



★ Statement 객체를 대신하는 PreparedStatement 객체

- Statement 객체와 PreparedStatement객체는 쿼리문을 실행하는 동일한 기능을 제공합니다.
- 그런데 PreparedStatement 객체를 사용하는 이유는 이 객체가 값 변환을 자동으로 해주는 기능을 제공하고, 간결한 코드를 만들 수 있기 때문입니다.
- Statement객체는 지정할 값이 많아질 경우 따옴표가 복잡하게 얽히기 때문에 코드 작성에서 오류가 발생할 수도 있고, 코드 수정시에도 어려움이 발생합니다.
- 그러나 PreparedStatement 객체는 값을 지정할 때 값 부분을 물음표(?)로 처리하기 때문에 간단히 값을 지정할 수 있습니다. 이 때 첫번째 물음표의 인덱스는 1이며, 이후 물음표의 인덱스는 나오는 순서대로 인덱스 값이 1씩 증가합니다.

### 3) sample

[작업순서]

#### 1. sql 실행

=> 관리자계정 jsp\_system.sql

=> 사용자계정 jsp\_jspuser.sql

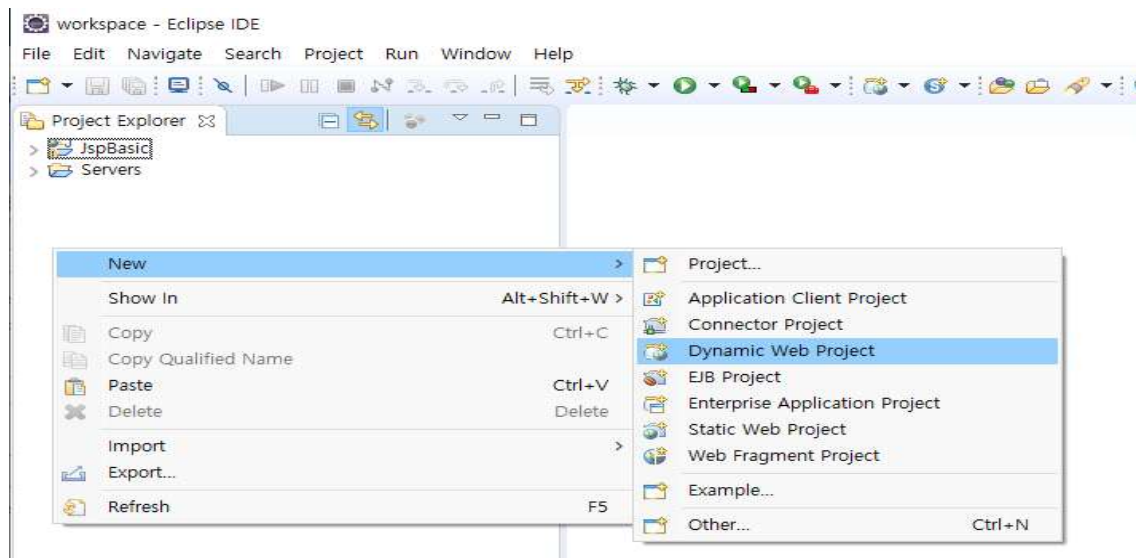
#### 2. eclipse 프로젝트 생성/ class 파일 생성

\* 프로젝트명: Jdbc

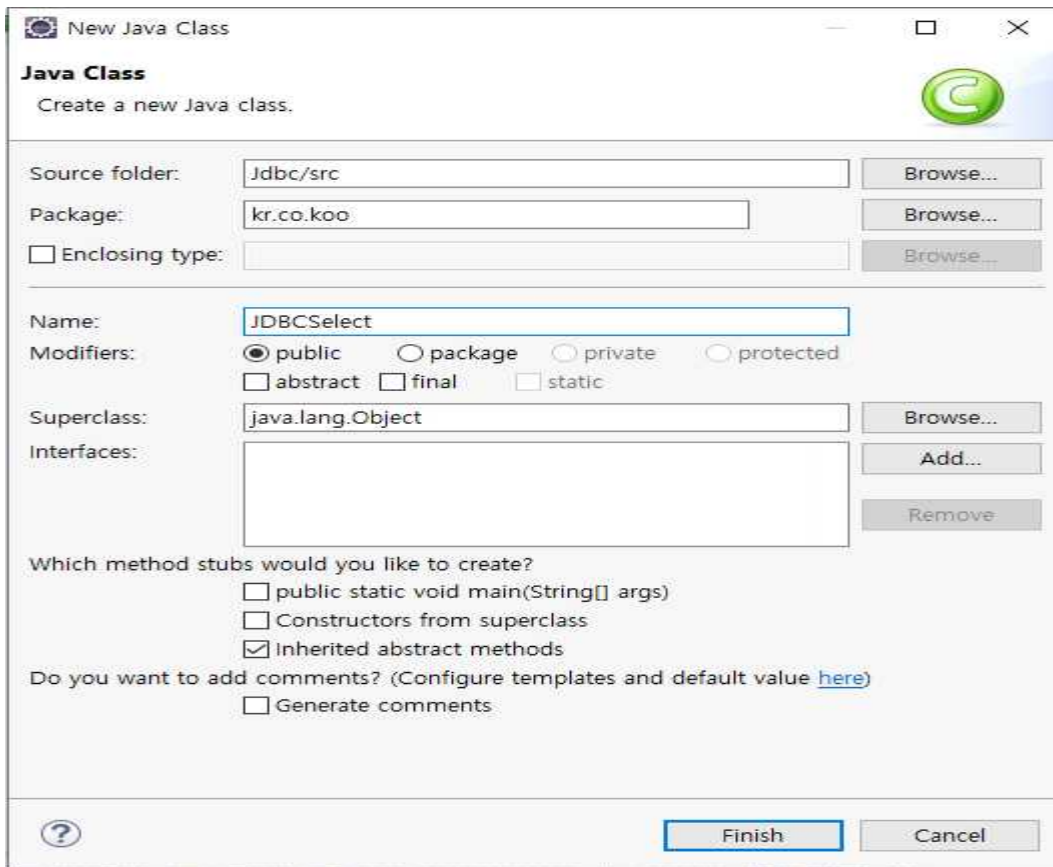
\* class 파일: JdbcSelect.java

- 패키지명: kr.co.koo

- 파일명: JdbcInsert.java



> 프로젝트명: Jdbc



> Java Resources> src 하위에 class파일 생성

### 3. oracle driver 다운로드

<https://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html>

> ojdbc6.jar 다운

> WebContent\WEB-INF\lib 하위에 붙여넣기

Menu

ORACLE

Search

Sign In

Country/Region

Contact

Oracle Technology Network / Database / Database Downloads

Database Downloads  
Database In-Memory  
Multitenant  
More Key Features  
Application Development  
Big Data Appliance  
Cloud Database Services  
Private Database Cloud  
Data Warehousing & Big Data  
Database Appliance  
Exadata Database Machine  
High Availability  
Manageability  
Migrations  
Security  
Unstructured Data  
Upgrades  
Windows  
Database A - Z  
Multilingual Engine

### Oracle Database 11.2.0.4 JDBC Driver & UCP Downloads

You must accept the [OTN License Agreement](#) to download this software.

☐ Accept License Agreement | ☐ Decline License Agreement

### Oracle Database 11g Release 2 (11.2.0.4) JDBC Drivers & UCP Downloads

#### Zipped JDBC Driver and Companion JARs

[ojdbc-full.tar.gz](#) (6,761,477 bytes) - (SHA1 Checksum: 1ce3d1055b94ee1c6148d74a440c937d0a2d30e)

The TAR archive contains the latest 11.2.0.4 JDBC Thin driver ([ojdbc6.jar](#) and [ojdbc5.jar](#)), Universal Connection Pool ([ucp.jar](#)), other companion jars, and README that has more information about the contents of the tar file.

OR

#### Unzipped JDBC Driver and Companion JARs

The JARs included in the [ojdbc-full.tar.gz](#) are also available as individual downloads in this section.

[ojdbc6.jar](#) (2,739,670 bytes) - (SHA1 Checksum: a483a046eee2f404d864a6ff5b09dc0e1be3fe6c)  
Certified with JDK8, JDK7, and JDK6;

[ojdbc5.jar](#) (2,091,137 bytes) - (SHA1 Checksum: 5543067223760fc2277fe3f603d8c4630927679c)  
For use with JDK1.5;

Project Explorer

Jdbc

Deployment Descriptor: Jdbc

JAX-WS Web Services

Java Resources

src

kr.co.koo

JDBCSelect.java

Libraries

JavaScript Resources

build

WebContent

META-INF

WEB-INF

lib

ojdbc6.jar

JspBasic

Servers

- 124 -

# JdbcInsert.java

=> [ctrl + f11]> java application

```
package kr.co.koo;
```

```
import java.sql.*;
import java.util.Scanner;
```

```
public class JDBCInsert {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);

        System.out.println("회원정보 입력을 시작합니다.");
        System.out.println("이름: ");
        String name = scan.next();

        System.out.println("아이디: ");
        String id = scan.next();

        System.out.println("비밀번호: ");
        String pw = scan.next();

        System.out.println("이메일: ");
        String email = scan.next();

        // DB 연동 순서
        // 1. DB 사용자계정명과 암호 DB URL등 초기 데이터 변수를 설정.
        String uid = "jspuser";
        String upw = "1234";
        String url = "jdbc:oracle:thin:@localhost:1521:xe";

        /*
         * java.sql 패키지의 내부의 클래스들을 객체로 만들려면
         * 반드시 try ~ catch 블록 내부에서 객체를 생성해야 합니다.
         * 그리고 해당 객체들이 가진 메서드들을 호출할 때도 try 블록 내부에서
         * 호출해야 합니다.
         * 그 이유는 java.sql 패키지의 생성자와 메서드가 전부
         * throws 선언을 해놨기 때문입니다.
         */
        Connection conn = null;
        Statement stmt = null;
        // 2. JDBC 드라이버 호출
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            // 3. DB작업을 위한 클래스들의 객체 생성
            /*
             * a - Connection 객체:
             * DB와의 연결을 위한 객체, Connection객체를 생성하려면
             * 직접 new 연산자를 통해 생성할 수 없고 DriverManager클래스가
             * 제공하는 정적 메서드인 getConnection()을 호출하여 객체를 생성합니다.
             */
            conn = DriverManager.getConnection(url, uid, upw);

            /*
             * b - Statement 객체:
             * SQL문 실행을 위한 Statement객체 생성.
             * Statement객체는 Connection객체가 제공하는
             * createStatement() 메서드를 호출하여 생성합니다.
             */
            stmt = conn.createStatement();

            /*
             * c - Statement객체를 통한 SQL문 실행.
             * 1. select명령일 경우에 executeQuery()
             * 2. insert, update, delete명령일 경우
             * executeUpdate()를 사용합니다.
             */

            String id = "cdcd";
            String pw = "1234";
```

```

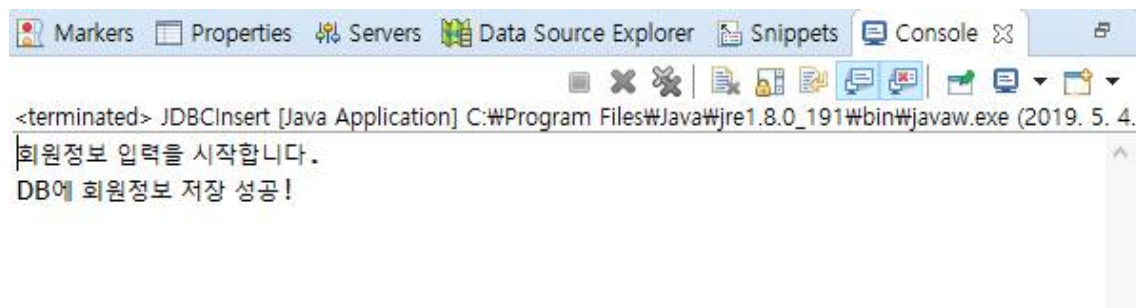
String name = "사오정";
String email = "abab@naver.com";

String sql = "insert into member values('"+id+"', '"+pw+"', '"+name+"', '"+email+"')";
/*
 * executeUpdate()는 실행에 성공했을 시 1, 실패했을 시 0을 리턴.
 */
int resultNum = stmt.executeUpdate(sql);
if(resultNum == 1) {
    System.out.println("DB에 회원정보 저장 성공!");
}else{
    System.out.println("DB에 회원정보 저장 실패!");
}

} catch (Exception e) {
    e.printStackTrace();
} finally {
    try {
        conn.close();
        stmt.close();
        scan.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

}
}

```



## # JdbcDelete.Java

```
package kr.co.koo;

import java.sql.*;
import java.util.Scanner;

public class JdbcDelete {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);

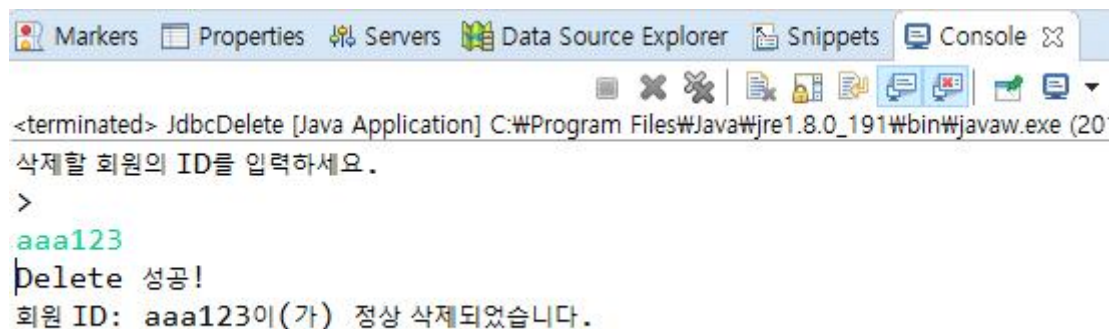
        System.out.println("삭제할 회원의 ID를 입력하세요.");
        System.out.println("> ");
        String id = scan.next();

        Connection conn = null;
        Statement stmt = null;

        String url = "jdbc:oracle:thin:@localhost:1521:xe";
        String uid = "jspuser";
        String upw = "1234";
        String sql = "delete from member where id = 'id'";
        String sql = "delete from member where id = '"+id+"'";

        try {
            // 1. 드라이버 로드
            Class.forName("oracle.jdbc.driver.OracleDriver");// [ctrl]+1=> try ~ catch
            // 2. 데이터베이스 Connection 객체 생성
            conn = DriverManager.getConnection(url, uid, upw);
            // 3. 쿼리문 실행을 위한 Statement 객체 실행
            stmt = conn.createStatement();
            // 4. 쿼리문을 실행
            int resultNum = stmt.executeUpdate(sql);

            if(resultNum == 1) {
                System.out.println("Delete 성공!");
                System.out.println("회원 ID: "+id+"이(가) 정상 삭제되었습니다.");
            } else {
                System.out.println("Delete 실패!");
            }
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } finally {
            try {
                conn.close(); // ctrl + 1
                stmt.close();
                scan.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```



```
<terminated> JdbcDelete [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (20
삭제할 회원의 ID를 입력하세요.
>
aaa123
Delete 성공!
회원 ID: aaa123이(가) 정상 삭제되었습니다.
```

## # JdbcSelect.java

=> [ctrl + f11]> java application

```
package kr.co.koo;
```

```
import java.sql.*;
```

```
public class JDBCSelect {
```

```
    public static void main(String[] args) {
```

```
        Connection conn = null;
```

```
        Statement stmt = null;
```

```
        ResultSet rs = null; // select 쿼리 실행시에만 필요한 객체.
```

```
        // 1. JDBC 드라이버 로드.
```

```
        String url = "jdbc:oracle:thin:@localhost:1521:xe"; // sqldeveloper > 사용자계정> 속성
```

```
        String uid = "jspuser";
```

```
        String upw = "1234";
```

```
        String sql = "SELECT * FROM MEMBER";
```

```
        try {
```

```
            Class.forName("oracle.jdbc.driver.OracleDriver");// [ctrl + 1] => try ~ catch
```

```
            conn = DriverManager.getConnection(url, uid, upw);
```

```
            stmt = conn.createStatement();
```

```
            // Select 쿼리문을 실행하려면 Statement객체와 executeQuery()를 호출
```

```
            // executeQuery()는 실행 결과를 ResultSet객체에 담아서 리턴함.
```

```
            rs = stmt.executeQuery(sql);
```

```
            /*
```

```
             * select쿼리문을 실행한 결과데이터가 1개라도 존재한다면, (데이터가 없으면 false)
```

```
             * ResultSet이 제공하는 next()메서드는 true를 리턴하고 데이터를 읽어옵니다.
```

```
             * 2회차 반복시부터는 다음 row 데이터를 읽어옵니다.
```

```
             */
```

```
            while(rs.next()) {
```

```
                /*
```

```
                 * select의 실행결과를 하나씩 읽어오려면
```

```
                 * getString(), getInt() 등의 메서드를 사용합니다.
```

```
                 * 해당 메서드의 매개값으로 읽어올 테이블의 컬럼명을 적어줍니다.
```

```
                */
```

```
                String id = rs.getString("id");
```

```
                String pw = rs.getString("pw");
```

```
                String name = rs.getString("name");
```

```
                String email = rs.getString("email");
```

```
                System.out.println("아이디: " + id); // sysso + [ctrl + space]
```

```
                System.out.println("비밀번호: " + pw);
```

```
                System.out.println("이름: " + name);
```

```
                System.out.println("이메일: " + email);
```

```
                System.out.println("=====");
```

```
            }
```

```
        } catch (Exception e) {
```

```
            e.printStackTrace();
```

```
        } finally {
```

```
            try {
```

```
                conn.close();
```

```
                stmt.close();
```

```
                rs.close();
```

```
            } catch (Exception e2) {
```

```
                e2.printStackTrace();
```

```
            }
```

```
        }
```

```
    }
```

```
}
```



```
<terminated> JDBCSelect [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (2019. 5. 14. 오후 1:09)
아이디: abc123
비밀번호: 1234
이름: 홍길동
이메일: null
=====
아이디: def456
비밀번호: 4433
이름: 이순신
이메일: null
=====
아이디: kkk456
비밀번호: 3456
이름: 김건모
이메일: null
=====
아이디: null/000
비밀번호: 6753
```



## # JdbcSelect2.java

```
package kr.co.koo;

import java.sql.*;
import java.util.*;

public class JdbcSelect2 {
    public static void main(String[] args) {
        /*
         * 회원의 ID를 입력받아 해당 ID의 회원정보를 모두 출력하는
         * JDBC 프로그래밍 코드를 작성하세요.
         */
        String uid = "jspuser";
        String upw = "1234";
        String url = "jdbc:oracle:thin:@localhost:1521:xe";

        Scanner scan = new Scanner(System.in);
        System.out.println("검색할 회원의 ID를 입력하세요.");
        System.out.print("> ");

        String sld = scan.next();
        String sql = "select * from member where id like '%sld%'";
        String sql = "select * from member where id like '%" + sld + "%'";

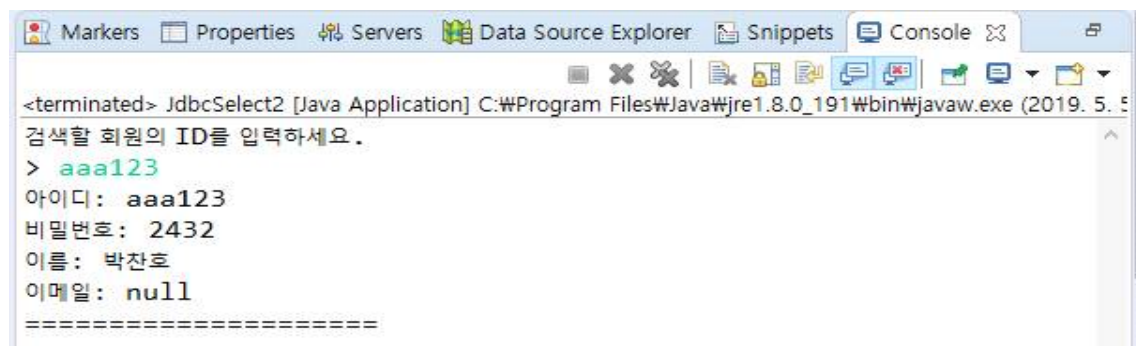
        Connection conn = null;
        Statement stmt = null;
        ResultSet rs = null;

        try {
            Class.forName("oracle.jdbc.driver.OracleDriver"); // [ctrl]+1=> try ~ catch
            conn = DriverManager.getConnection(url, uid, upw);
            stmt = conn.createStatement();

            rs = stmt.executeQuery(sql);

            while(rs.next()) {
                String id = rs.getString("id");
                String pw = rs.getString("pw");
                String name = rs.getString("name");
                String email = rs.getString("email");

                System.out.println("아이디: " + id);
                System.out.println("비밀번호: " + pw);
                System.out.println("이름: " + name);
                System.out.println("이메일: " + email);
                System.out.println("=====");
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                conn.close(); // [ctrl]+1
                stmt.close();
                rs.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```



## 9. JDBC - MVC 패턴 실습

### 1) 개념

#### \* Statement 객체를 대신하는 PreparedStatement 객체

- Statement 객체와 PreparedStatement 객체는 쿼리문을 실행하는 동일한 기능을 제공합니다.
- 그런데 PreparedStatement 객체를 사용하는 이유는 이 객체가 값 변환을 자동으로 해주는 기능을 제공하고, 간결한 코드를 만들 수 있기 때문입니다.
- Statement 객체는 지정할 값이 많아질 경우 따옴표가 복잡하게 얽히기 때문에 코드 작성에서 오류가 발생할 수도 있고, 코드 수정시에도 어려움이 발생합니다.
- 그러나 PreparedStatement 객체는 값을 지정할 때 값 부분을 물음표(?)로 처리하기 때문에 간단히 값을 지정할 수 있습니다. 이 때 첫번째 물음표의 인덱스는 1이며, 이후 물음표의 인덱스는 나오는 순서대로 인덱스 값이 1씩 증가합니다.

#### \* MVC 패턴에서의 Model

##### 1. DAO 클래스(Data Access Object)

- 데이터베이스에 접속해서 데이터의 추가, 삭제, 수정 등의 작업을 하는 클래스입니다.
- 일반적으로 JSP 혹은 Servlet에서 위의 로직을 함께 기술할 수도 있지만, 유지보수 및 코드의 모듈화를 위해 별도의 DAO 클래스를 만들어 사용합니다.
- 보통 한 개의 테이블마다 한 개의 DAO 클래스를 작성합니다.
- DAO 클래스는 테이블로부터 데이터를 읽어와 자바 객체로 변환하거나 자바 객체의 값을 테이블에 저장합니다.
- 따라서 DAO를 구현하면 테이블의 컬럼과 매핑되는 값을 갖는 자바빈 클래스를 항상 작성해야 합니다. 이 자바빈 클래스를 VO 클래스라 부릅니다.
- DAO 클래스는 front-end 개발과 back-end 개발을 나누어 할 수 있게 도와줍니다.

##### 2. VO 클래스(Value Object) / DTO 클래스(Data Transfer Object)

- DAO 클래스를 이용하여 데이터베이스에서 데이터를 관리할 때 데이터를 일반적인 변수에 할당하여 작업할 수도 있지만, 별도의 VO 클래스를 작성하여 데이터베이스와 관련된 변수들의 모음역할을 합니다.
- VO 클래스는 자바빈 클래스로 생성합니다.

## 2) Sample

Sample 01> MVC 아님! (위치: WebContent\user)

### [작업 순서]

#### 1) sql

```
# 회원 관리 JDBC
# 회원 테이블 생성
create table users (
    name varchar(20) not null,
    id varchar(20) not null primary key,
    pw varchar(20) not null,
    phone1 varchar(20),
    phone2 varchar(20),
    phone3 varchar(20),
    gender varchar(20)
);
select * from users;
```

#### 2) java

[Jdbc>WebContent>user]

```
# join_form.jsp      ! Ctrl + F11
# join_ok.jsp
# join_result.jsp
```

```
# login_form.jsp     ! Ctrl + F11
# login_ok.jsp
# login_fail_pw.jsp
# login_fail_id.jsp
# login_welcome.jsp
# logout.jsp
```

```
# update_form.jsp    ! Ctrl + F11
# update_ok.jsp
```

```
# delete_ok.jsp
```

=> 연결되는 프로그램임!

전체 돌릴때는 join\_form.jsp에서만 실행하면 됨!

```
insert into users values('사오정', 'abc1234', '1234', '010', '1212', '3434', '여')
```

```
select pw, name from users where id ='abc1234'
```

- ① => update\_form.jsp
- ② => delete\_ok.jsp
- ③ => logout.jsp

**\* 실습 1> login\_ok.jsp에서 시작**

```
// 1. 폼데이터 처리(id, pw 파라미터)
// 2. SQL문 작성(select로 가입당시의 패스워드를 조회)
// 3. DB연동에 필요한 변수들 선언.(url, uid, upw, conn, stmt, rs)
// 4. try안에서 JDBC로 DB 연결
// 5. ResultSet을 통해 결과값 소비를 통해 해당 로그인 결과에 맞는 페이지 안내.
// 6. 로그인 성공시 세션 처리.
```

**\* 실습 2> update\_ok.jsp**

```
/*
1. 폼태그에서 넘어온 데이터들(파라미터)를 각각 변수에 저장.
2. 해당 데이터를 sql문을 통해 DB에 전송.
3. 전송이 완료됐을 시 executeUpdate()가
   1을 리턴하면 세션에 변경된 이름을 저장한 후 update_result.jsp로 리다이렉트,
   0을 리턴하면 login_form.jsp로 리다이렉트.
*/
```

**\* 실습 3> delete\_ok.jsp**

=> 회원탈퇴

### Sample 02> Prepared Statement 사용법

# pstmt\_basic.jsp ! WebContent/prepared\_statement/

### Sample 03> MVC로 변경! (위치: WebContent\user2)

=> jsp코드 안에 들어있는 DB 처리 관련 내용들을 DAO, DTO 클래스로 분리

[기존에 있던 소스]

# join\_form.jsp ★

# join\_ok.jsp ! DB 로직 (o) => 소스 변경 (o) (PreparedStatement 사용, UserVO, UserDao 사용)

# join\_result.jsp

# login\_form.jsp ★

# login\_ok.jsp ! DB 로직 (o) => 소스 변경 (o) (PreparedStatement 사용)

# login\_fail\_pw.jsp

# login\_fail\_id.jsp

# login\_welcome.jsp

# logout.jsp

# update\_form.jsp ★ ! DB 로직 (o)

# update\_ok.jsp ! DB 로직 (o) => 소스 변경 (o) (PreparedStatement 사용)

# delete\_ok.jsp ! DB 로직 (o) => 소스 변경 (o) (PreparedStatement 사용, UserVO, UserDao 사용)

[추가된 소스]

# UserVO.java ! DB 데이터 관리(src\kr\co\koo\)

# UserDao.java ! DB 로직 처리(src\kr\co\koo\)

# show\_all\_users.jsp ! 모든 회원정보 조회

## 10. Transaction/ Connection pool

### 1) 개념

#### \* JDBC에서 트랜잭션 처리

- 두 개 이상의 쿼리를 모두 성공적으로 실행해야 데이터가 정상적으로 처리되는 경우 DBMS는 트랜잭션(transaction)을 이용해서 두 개 이상의 쿼리를 마치 한 개의 쿼리처럼 처리할 수 있게 해줍니다.
- 트랜잭션은 시작과 종료를 갖고 있으며, 트랜잭션이 시작되면 이후로 실행되는 쿼리 결과는 DBMS에 곧바로 반영되지 않고 임시로 보관됩니다.
- 이후 트랜잭션을 커밋(commit)하면 임시 보관한 쿼리 결과를 실제 데이터베이스에 반영합니다.
- 트랜잭션은 커밋하기 전에 예러가 발생하면 임시로 보관한 모든 쿼리 결과를 실제 데이터에 반영하지 않고 롤백(rollback)합니다.

#### \* 연결 풀(Connection Pool)

=> 관련 링크: <https://youtu.be/OqBXwAC6MSM>

- 연결 풀은 데이터베이스 메모리 내에 있는 데이터베이스 커넥션들로 구성된 하나의 캐시입니다.
- 데이터베이스 연결 풀은 데이터에 대한 요청이 발생하면 재사용되는 것으로, 데이터베이스의 수행 능력을 향상시키기 위해 사용됩니다.
- 연결 풀에서 하나의 연결이 생성되어 풀에 배치되면 새로운 연결이 만들어지지 않도록 재사용하지만, 만약 모든 연결이 사용 중에 있으면 새로운 연결이 만들어져 풀에 추가됩니다.
- 연결 풀을 통해 사용자는 데이터베이스 연결을 위해 기다리는 시간을 축소시켜줍니다.
- 커넥션 풀 설정은 Eclipse-> Servers폴더에 -> context.xml을 수정합니다.

// mysql의 경우...

```
<Resource
    auth="Container"
    driverClassName = "com.mysql.jdbc.Driver"
    url = "jdbc:mysql://localhost:3306/practice"
    username = "jsp"
    password = "jsp"
    name = "jdbc/mysql"
    type = "javax.sql.DataSource"
    maxActive = "300"
    maxWait = "1000"
/>
```

## 2) Sample

### Sample 1) 트랜잭션 처리

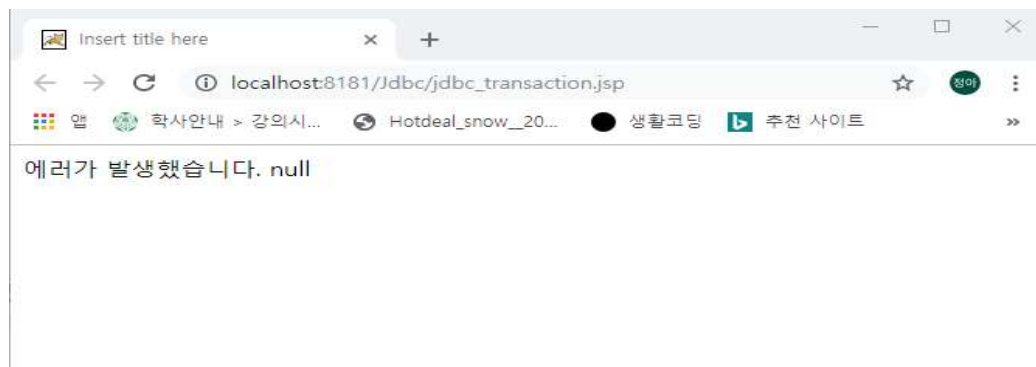
#### 1. sql 생성

```
create table item (  
    item_id int not null primary key,  
    name varchar(30)  
);
```

```
create table item_detail (  
    item_id int not null primary key,  
    detail varchar(100)  
);
```

#### 2. jdbc\_transaction.jsp 작성 ! Ctrl + F11

- 위치: WebContent\jdbc\_transaction.jsp



&& => DB에도 반영 안됨!



## Sample 2) Connection Pool

=> 미리 만들어 놓고 뿌려주자!

- ① context.xml ! Servers>
- ② UserDao.java ! Jdbc> Java Resources
- # login\_form.jsp ! C trl + F11

### ① context.xml

```
29 <Resource name="jdbc/oracle" auth="Container" type="javax.sql.DataSource"
30     dirverClassName="oracle.jdbc.driver.OracleDriver" url="jdbc:oracle:thin:@localhost:1521:xe"
31     username="jspuser" password="1234" maxActive="300" maxWait="1000" />
```

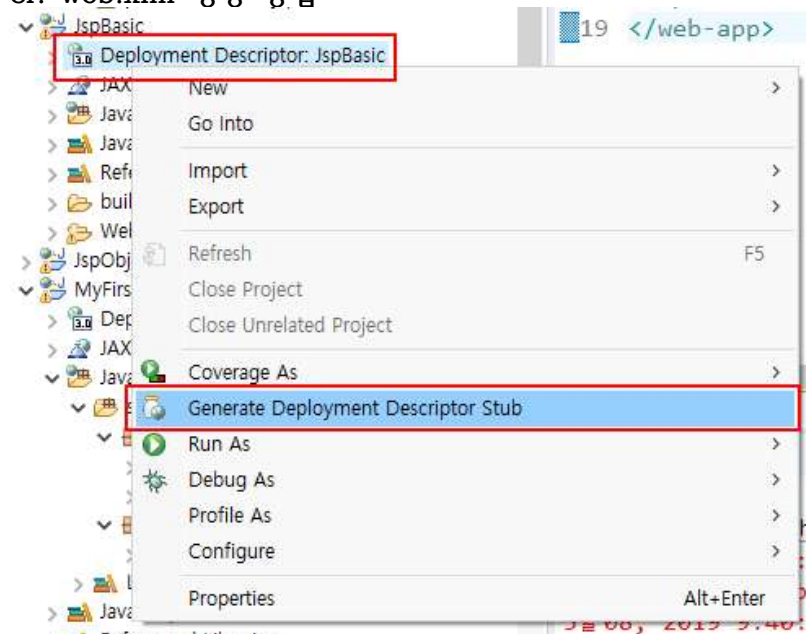
```
<Resource name="jdbc/oracle" auth="Container" type="javax.sql.DataSource"
    dirverClassName="oracle.jdbc.driver.OracleDriver" url="jdbc:oracle:thin:@localhost:1521:xe"
    username="jspuser" password="1234" maxActive="300" maxWait="1000" />
```

### ② web.xml

```
13 <resource-ref>
14     <description>Connection</description>
15     <res-ref-name>jdbc/oracle</res-ref-name>
16     <res-type>javax.sql.DataSource</res-type>
17     <res-auth>Container</res-auth>
18 </resource-ref>
19 </web-app>
```

```
<resource-ref>
    <description>Connection</description>
    <res-ref-name>jdbc/oracle</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
</resource-ref>
```

### cf. web.xml 생성 방법



### ③ UserDAO.java

```
package kr.co.koo;

import java.sql.*;
import java.util.ArrayList;

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.sql.DataSource;

// DAO클래스는 DB연동을 전담합니다.
public class UserDAO {
    /*
    String url = "jdbc:oracle:thin:@localhost:1521:xe";
    String uid = "jspuser";
    String upw = "1234";
    */
    private DataSource ds; // javax.sql

    // 싱글톤 패턴 클래스 생성.
    // 1. 자신의 클래스 내부에 스스로의 객체를 생성함.
    private static UserDAO dao = new UserDAO();

    //2. 외부에서 객체를 생성할 수 없도록 생성자에 private제한을 붙임.
    private UserDAO() {
        try {
            Context ct = new InitialContext(); // javax.naming
            ds = (DataSource) ct.lookup("java:/comp/env/jdbc/oracle");
            System.out.println("[DataSource] :: " + ds.toString());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    /*
    try {
        Class.forName("oracle.jdbc.driver.OracleDriver");
    } catch (Exception e) {
        e.printStackTrace();
    }
    */

    // 3. 외부에서 객체생성을 요구할 시 공개된 메서드를 통해 단 1개의 객체를 제공함.
    public static UserDAO getInstance() {
        if(dao != null) {
            dao = new UserDAO();
        }
        return dao;
    }
}

// UserDAO에서는 회원관리에 필요한 DB연동 로직들을 메서드로 작성합니다.
// 데이터베이스로부터 모든 회원의 정보를 가져오는 메서드
public ArrayList<UserVO> userSelectAll(){

    ArrayList<UserVO> userList = new ArrayList<>();

    String sql = "select * from users";

    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    try {
        conn = DriverManager.getConnection(url, uid, upw);
        conn = ds.getConnection();
        pstmt = conn.prepareStatement(sql);
        rs = pstmt.executeQuery();

        while(rs.next()) {
            String name = rs.getString("name");

```

```

        String id = rs.getString("id");
        String pw = rs.getString("pw");
        String phone1 = rs.getString("phone1");
        String phone2 = rs.getString("phone2");
        String phone3 = rs.getString("phone3");
        String gender = rs.getString("gender");

        UserVO vo = new UserVO(name, id, pw, phone1, phone2, phone3, gender);

        userList.add(vo);
    }
} catch (Exception e) {
    e.printStackTrace();
} finally {
    try {
        if(conn!= null) conn.close();
        if(pstmt!= null) pstmt.close();
        if(rs!= null) rs.close();

    } catch (Exception e2) {
        e2.printStackTrace();
    }
}
return userList;
}

```

// 회원데이터를 DB에 Insert하는 메서드 선언

```

public int join(UserVO vo) {
    String sql = "insert into users values (?,?,?,?,?,?)";

    int rn = 0;

    Connection conn = null;
    PreparedStatement pstmt = null;

    try {
        conn = DriverManager.getConnection(url, uid, upw);
        conn = ds.getConnection();
        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, vo.getName());
        pstmt.setString(2, vo.getId());
        pstmt.setString(3, vo.getPw());
        pstmt.setString(4, vo.getPhone1());
        pstmt.setString(5, vo.getPhone2());
        pstmt.setString(6, vo.getPhone3());
        pstmt.setString(7, vo.getGender());

        rn = pstmt.executeUpdate();

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            if(conn!=null) conn.close();
            if(pstmt!=null) pstmt.close();

        } catch (Exception e2) {
            e2.printStackTrace();
        }
    }
    return rn;
}

```

// 회원탈퇴를 처리하는 메서드 선언

```

public int deleteUser(String id) {
    String sql = "delete from users where id=?";

    int rn = 0;

    Connection conn = null;

```

```

PreparedStatement pstmt = null;

//
try {
    conn = DriverManager.getConnection(url, uid, upw);
    conn = ds.getConnection();
    pstmt = conn.prepareStatement(sql);
    pstmt.setString(1, id);

    rn = pstmt.executeUpdate();
} catch (Exception e) {
    e.printStackTrace();
} finally {
    try {
        } catch (Exception e2) {
            e2.printStackTrace();
        }
    }
return rn;
}
}

```

## 11. WebProject 1

### [작업순서]

#### 1) MyFirstWeb 생성 및 템플릿 내용 복.붙

##### ① MyFirstWeb

- context root: jsp\_pj

##### ② JDBC> WEB-INF> lib

ojdbc6.jar

Ctrl+C

MyFirstWeb> WebContent> WEB-INF> lib

Ctrl+V

##### ③ 웹템플릿 안의 모든 내용>

MyFirstWeb> WebContent 에 복.붙

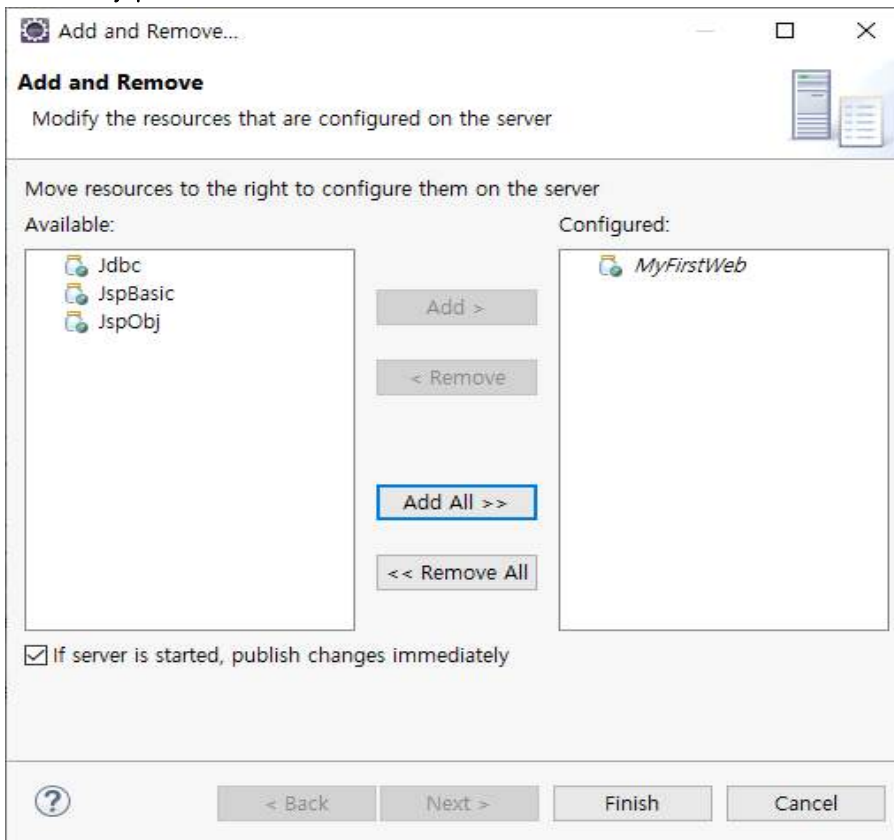
##### ④ configure build path

##### ⑤ 서버 중지 시키고>

Servers> Tomcat> 오.버> Add and Remove

<< Remove All

# index.jsp ! Ctrl + F11



## 2) 코드 작업

### ① 디자인 변경

- index.jsp : 웹 제목, 저작권, 이미지 등 내가 원하는데로 변경
- business-casual.css> body : 배경 이미지 바꾸기

### ② 경로 수정

# index.jsp

line 53. : lovelyz > jsp\_pj

line 60. : lovelyz > jsp\_pj

line 63. : member/member.jsp > /jsp\_pj/member/member.jsp

### ③ include 폴더 생성> header.jsp, footer.jsp 생성

# header.jsp

index.jsp의 (line.39 ~ line.79)

위, 아래에 header start, end 작성 후

<!-- header start -->

<div class="brand">My First Web</div> # <body> 다음

... ..

</nav>

# <div class="container"> 전

<!-- header end -->

해당 내용 복사

header.jsp에 붙여넣기

# footer.jsp

index.jsp의 (line.152 ~ line.160 )

위, 아래에 footer start, end 작성 후

해당 내용 복사

footer.jsp에 붙여넣기

# index.jsp

header.jsp, footer.jsp에 해당 하는 내용 지우고

해당 내용 추가

<body> 태그 아래에

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

</body> 태그 위에

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

### ④ member.jsp도 index.jsp처럼 처리해줄 것

단 경로는,

<body> 태그 아래

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

</body> 태그 위에

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

현재 member.jsp는 index.jsp와는 다르게 member 폴더 안에 있으므로 해당 폴더에서 나와서 include폴더로 들어가야함!

### 3) sql문 작성

cf. MySQL 서비스가 로컬 컴퓨터에서 시작했다가 중지되었습니다. 오류 해결방법

<https://k-channel.tistory.com/213>

## 12. WebProject 2

### \* 실습 자료

// 회원가입 처리

```
user_join.jsp          # webContent>user>
header.jsp             # webContent>include>
user_join_ok.jsp       # webContent>user>... / 회원가입 - 1
```

// ID 중복 체크 / 유효성 처리(비밀번호 체크 등)

```
MemberDAO.java # public int insertMember(MemberVO member)
user_join_ok.jsp # webContent>user>... / 회원가입 - 2
member.js
```

// login페이지 꾸미기

```
user_login.jsp
user_login_ok.jsp
MemberDAO.java # public int userCheck(String id, String pw) / 로그인 유효성을 검증
```

// 로그인 후 처리

```
MemberDAO.java # public MemberVO getMemberInfo(String id) / 로그인한 회원의 정보를 DB로부터 가져옴
user_login_ok.jsp
} else {
    MemberVO member = dao.getMemberInfo(id);
    session.setAttribute("user_name", member.getName());
    세션 처리
```



### \*\*\* 작업순서

#### 1) 회원가입 처리 (//WebContent> user(folder)

##### ① user\_join.jsp

1> member.jsp 의 // index.jsp No!!!!!!(다 경로 상위로 올라가야함!)

<head>

...

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

<div class="container">

까지 복사!!!

2> </head> 위에

<style>

.div\_center {

margin-bottom: 20px;

padding: 30px 15px;

background: #fff;

background: rgba(255, 255, 255, 0.9);

}

</style>

내용 추가!

3> <div class="container"> ==> <div class="div\_center" align="center"> 변경

4> 바로 뒤에

</div>

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

추가!

cf. WEB-INF> web.xml

<welcome-file-list> : 사용자가 볼 홈 파일 지정, 우리 홈페이지 얼굴

cf. context root 없애기

Servers> server.xml

path="/seo" -> path="/" > synchronized 버튼 클릭

```
<Context docBase="MyFirstWeb" path="/" reloadable="true" source="org.eclipse.jst.jee.server:MyFirstWeb"/><Context
docBase="Jdbc" path="/Jdbc" reloadable="true" source="org.eclipse.jst.jee.server:Jdbc"/></Host>
```

## ② header.jsp

```
line.33: href="#" => href="/seo/user/user_join.jsp"  
<a href="/seo/user/user_join.jsp" style="color:red">JOIN</a>
```

## ③ user\_join\_ok.jsp

## ④ MemberDAO.java

```
public int insertMember(MemberVO member)
```

## 2) ID 중복 체크 / 유효성 처리(비밀번호 체크 등)

① MemberDAO.java

```
public int confirmId(String id)
```

② user\_join\_ok.jsp

③ member.js // WebContent> js>

## 3) login페이지 꾸미기

① user\_login.jsp // => header.jsp

② user\_login\_ok.jsp

③ MemberDAO.java

```
public int userCheck(String id, String pw) // 로그인 유효성을 검증
```

## 4) 로그인 후 처리

① MemberDAO.java

```
public MemberVO getMemberInfo(String id) // 로그인한 회원의 정보를 DB로부터 가져옴
```

② user\_login\_ok.jsp

```
} else {
```

```
MemberVO member = dao.getMemberInfo(id);
```

```
session.setAttribute("user_name", member.getName());
```

세션 처리

==> 홈피 LOGIN, JOIN => MYPAGE, LOGOUT으로 바뀌게 처리!!!

## 13. WebProject 3

### [작업순서]

#### 1) login 후처리

① header.jsp

line.6 ~ 10

line.37 ~ 51

#### 2) logout 처리

① user\_logout.jsp

#### 3) mypage

① user\_mypage.jsp

user\_login.jsp 의 line.5(<head>) ~ line.45(<body>, <div> 시작)까지 복.붙  
</div>

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

#### 4) 비밀번호 변경 ==> 실습

① change\_pw.jsp

의 <head> ~ </body>이전에 까지

user\_mypage.jsp의 line.13(<head>) ~ line.53(<body>, <div> 시작)까지 복.붙  
</div>

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

② change\_pw\_ok.jsp

<%--

1. 폼데이터를 처리합니다.(예전비번, 바꿀비번)
2. dao객체를 생성하여 조건문으로 조건식에 예전비번과 아이디정보를 바탕으로 해당 비번이 일치하는지부터 확인.
3. 일치한다면 비밀번호를 바꾸는 로직을 실행. alert으로 "비밀번호가 정상적으로 변경되었습니다."출력
4. 예전비번이 일치하지 않는다면 "현재 비밀번호가 다릅니다." 출력 후 다시 전페이지로 리다이렉팅함.
5. 비번 변경 실패시 "비밀번호 변경에 실패했습니다"를 출력하세요.
6. sql : update members set user\_pw=? where user\_id=?

--%>

③ MemberDAO.java

=> public int changePassword(String pw, String id)

#### 5) 회원정보 변경

- ① user\_update.jsp
- ② user\_update\_ok.jsp
- ③ MemberDAO.java => updateUser(MemberVO member)

## 14. WebProject 4 / EL / JSTL

### 1) 개념

#### \* EL(Expression Language)

- EL은 일종의 스크립트 언어로 자료 타입, 수치 연산자, 논리 연산자, 비교 연산자 등을 제공하며 표현식을 대체할 수 있습니다.

- EL의 사용법

ex) 표현식 `<%= value %>` // EL `${value}`

- EL 내부에 사용하는 연산자

1. 산술: +, -, \*, /, %

2. 관계: ==, !=, <, <=, >, >=

3. 조건: `a ? b : c` (a조건식이 참이면 b를 실행, 거짓이면 c를 실행)

4. 논리: &&, ||

#### \* 액션 태그와 EL

- 액션태그 `<jsp:getProperty name="member" property="name"/>`

- EL -> `${member.name}`

- EL의 내장객체

1. pageScope: JSP의 page 객체를 참조하는 객체

2. requestScope: JSP의 request 객체를 참조하는 객체

3. sessionScope: JSP의 session 객체를 참조하는 객체

4. applicationScope: JSP의 application 객체를 참조하는 객체

5. param: 요청 파라미터를 참조하는 객체

6. paramValues: 요청 파라미터(배열)를 참조하는 객체

7. initParam: 서블릿컨텍스트 초기화 파라미터를 참조하는 객체

8. cookie: 쿠키 객체를 참조하는 객체.

#### \* JSTL (JSP Standard Tag Library)

- JSP의 경우 HTML태그와 같이 사용되어 전체적인 코드의 가독성이 떨어집니다.

- 그래서 이런 단점을 보완하고 만들어진 것이 JSTL입니다.

- JSTL을 사용하면 자바의 제어문을 HTML 태그화 시킬 수 있습니다.

- JSTL의 경우 우리가 사용하는 Tomcat 기본 컨테이너에 포함되어 있지 않으므로, 별도의 라이브러리를 설치하고 사용합니다.

- 설치방법:

<http://jakarta.apache.org> 접속 ... ↓ ↓ ↓

Insert title here x The Jakarta Site - The Apache J x +

← → ↻ ⓘ 주의 요약 jakarta.apache.org

**Support**

- [License](#)
- [Mailing Lists](#)
- [Jakarta Wiki](#)

**Ex-Jakarta**

- [Ant](#)
- [Avalon](#)
- [BCEL](#)
- [BSF](#)
- [Commons](#)
- [DB](#)
- [Excalibur](#)
- [Gump](#)
- [HiveMind](#)
- [HttpComponents](#)
- [James](#)
- [JCS](#)
- [JMeter](#)
- [Logging](#)
- [Lucene](#)
- [Maven](#)
- [POI](#)
- [Portals](#)
- [Struts](#)
- [Taglibs](#)
- [Tapestry](#)
- [Tomcat](#)
- [Turbine](#)
- [Velocity](#)
- [Watchdog](#)

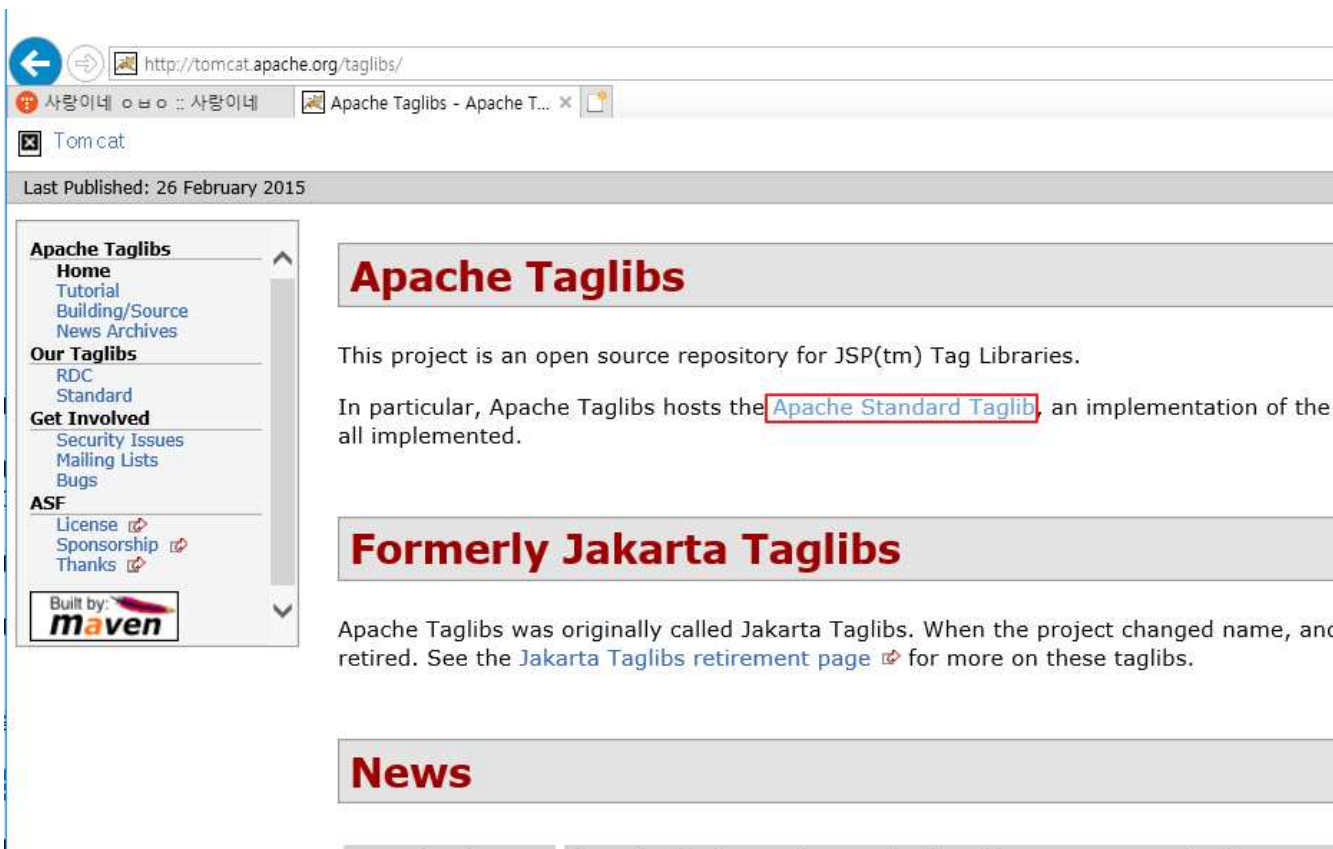
**Welcome to The Apache Jakarta™ Project**

Founded in 1999, the Jakarta Project housed a diverse set of popular open source Java solutions. In 2005, as a part of creating a flatter Apache Software Foundation, Jakarta subprojects began to become full top-level Apache projects. This process has continued to this day, all subprojects have now left the Jakarta project to become top level projects, join other TLPs (Commons), or in some cases been retired.

**News**




Latest Jakarta News

- [21 December 2011 - Jakarta Retired](#)
- [26 October 2011 - JMeter becomes a top level project](#)
- [03 October 2011 - Apache JMeter 2.5.1 Released](#)
- [11 September 2011 - BSF moves to Apache Commons](#)
- [17 August 2011 - Apache JMeter 2.5 Released](#)
- [05 August 2011 - Cactus moves to Apache Attic](#)
- [25 June 2011 - JCS moves to Apache Commons](#)
- [25 June 2011 - BCEL moves to Apache Commons](#)
- [17 April 2011 - Regexp is retired](#)
- [01 September 2010 - ORO is retired](#)
- [01 September 2010 - ECS is retired](#)
- [14 July 2010 - Apache JMeter 2.4 Released](#)
- [24 June 2010 - Jakarta BSF 3.1 Released](#)
- [09 October 2009 - Jakarta BSF 3.0 Released](#)
- [04 October 2009 - Taglibs is retired](#)
- [21 June 2009 - Apache JMeter 2.3.4 Released](#)
- [24 May 2009 - Apache JMeter 2.3.3 Released](#)
- [05 April 2009 - Jakarta BSF 3.0-beta3 Released](#)
- [26 January 2009 - Apache Cactus 1.8.1 Released](#)
- [14 June 2008 - Apache JMeter 2.3.2 Released](#)



## JSP(tm) Standard Tag Library implementations

Version	JSTL version	Requirements	Getting the Taglib
Standard 1.2.3	JSTL 1.2	Servlet 2.5, JavaServer Pages 2.1	<a href="#">download</a>  (javadoc)
Standard 1.1	JSTL 1.1	Servlet 2.4, JavaServer Pages 2.0	<a href="#">download</a>
Standard 1.0	JSTL 1.0	Servlet 2.3, JavaServer Pages 1.2	<a href="#">download</a>

Name	Last modified	Size	Description
 <a href="#">Parent Directory</a>		-	
 <a href="#">binaries/</a>	2005-10-05 20:39	-	
 <a href="#">source/</a>	2005-10-05 20:38	-	

← → ↻ ⓘ 주의 요함   archive.apache.org/dist/jakarta/taglibs/standard/binaries/		
	<a href="#">jakarta-taglibs-standard-1.0.3.zip</a>	2003-02-19 22:23 5.0M
	<a href="#">jakarta-taglibs-standard-1.0.3.zip.asc</a>	2003-02-19 22:23 232
	<a href="#">jakarta-taglibs-standard-1.0.tar.gz</a>	2002-06-21 22:49 4.9M
	<a href="#">jakarta-taglibs-standard-1.0.tar.gz.asc</a>	2002-06-21 22:49 232
	<a href="#">jakarta-taglibs-standard-1.0.zip</a>	2002-06-21 22:58 5.0M
	<a href="#">jakarta-taglibs-standard-1.0.zip.asc</a>	2002-06-21 22:58 232
	<a href="#">jakarta-taglibs-standard-1.1.0-B1.tar.gz</a>	2003-09-24 00:09 2.7M
	<a href="#">jakarta-taglibs-standard-1.1.0-B1.tar.gz.asc</a>	2003-09-24 00:09 304
	<a href="#">jakarta-taglibs-standard-1.1.0-B1.zip</a>	2003-09-24 00:09 2.7M
	<a href="#">jakarta-taglibs-standard-1.1.0-B1.zip.asc</a>	2003-09-24 00:09 304
	<a href="#">jakarta-taglibs-standard-1.1.0.tar.gz</a>	2004-01-28 20:11 2.7M
	<a href="#">jakarta-taglibs-standard-1.1.0.tar.gz.asc</a>	2004-01-28 20:11 304
	<a href="#">jakarta-taglibs-standard-1.1.0.zip</a>	2004-01-28 20:11 2.8M
	<a href="#">jakarta-taglibs-standard-1.1.0.zip.asc</a>	2004-01-28 20:11 304
	<a href="#">jakarta-taglibs-standard-1.1.1.tar.gz</a>	2004-07-19 21:53 872K
	<a href="#">jakarta-taglibs-standard-1.1.1.tar.gz.asc</a>	2004-07-19 21:53 186
	<a href="#">jakarta-taglibs-standard-1.1.1.zip</a>	2004-07-19 21:53 931K
	<a href="#">jakarta-taglibs-standard-1.1.1.zip.asc</a>	2004-07-19 21:53 186
	<a href="#">jakarta-taglibs-standard-1.1.2.tar.gz</a>	2004-10-25 20:57 873K
	<a href="#">jakarta-taglibs-standard-1.1.2.tar.gz.asc</a>	2004-10-25 20:57 186
	<a href="#">jakarta-taglibs-standard-1.1.2.zip</a>	2004-10-25 20:57 933K
	<a href="#">jakarta-taglibs-standard-1.1.2.zip.asc</a>	2004-10-25 20:57 186
	<a href="#">jakarta-taglibs-standard-oldxml-compatible.tar.gz</a>	2002-06-21 22:59 1.1M
	<a href="#">jakarta-taglibs-standard-oldxml-compatible.tar.gz.asc</a>	2002-06-21 22:59 232
	<a href="#">jakarta-taglibs-standard-oldxml-compatible.zip</a>	2002-06-21 23:01 1.1M
	<a href="#">jakarta-taglibs-standard-oldxml-compatible.zip.asc</a>	2002-06-21 23:01 232

- 복.불 위치:

D:\9월\_1230\_JSP\_서정아\zUtil\el\jakarta-taglibs-standard-1.1.2\lib >...

를

WebContent\WEB-INF\lib > ...

에 복.불



## \* JSTL Core Library

- Core 라이브러리는 기본적인 라이브러리로 출력, 제어문, 반복문 같은 기능이 포함되어 있습니다.
- 코어 라이브러리를 사용하기 위해서 반드시 JSP 파일 내의 지시자 태그로

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

- 사용법:

### ① 출력 태그

<c:out value="출력값">

### ② 변수 선언 태그

- 변수 태그는 JSTL이 지원하는 태그에서 사용할 수 있는 변수를 설정하기 위해 사용합니다. EL 변수를 생성합니다.
- <c:set var="변수명" value="설정값" scope="범위">
- a. var: 값을 저장할 EL 변수의 이름을 지정함.
- b. value: 변수의 값을 지정함. 표현식, EL, 텍스트를 사용하여 지정가능.
- c. scope: 변수를 저장할 영역을 지정함. page, request, session, application 중 하나를 지정해야 하며 지정하지 않을 경우 page를 기본 값으로 사용.

### ③ 변수 제거 태그

- <c:remove var="변수명" scope="범위">

- remove 태그를 사용할 때 주의점은 삭제할 변수의 scope을 지정하지 않으면 동일한 이름으로 저장된 모든 영역의 변수를 삭제합니다.

## ★④ 조건문(if) 태그

- if 태그는 자바 언어의 if 블록과 비슷한 기능을 제공합니다. 중첩된 if~else 문과 같은 효과를 낼 수는 없지만 단순한 if 블록을 쉽게 대체할 수 있기 때문에 많이 사용합니다.

- <c:if test="조건식" var="조건 처리 변수명">

## ⑤ 조건문(choose) 태그

- choose 태그는 자바의 switch 구문과 if~else 구문을 혼합한 형태로 다수의 조건문을 하나의 블록에서 수행할 때 사용합니다.

- <c:choose>

    <c:when test="조건식"> 처리 내용 </c:when>

    <c:when test="조건식"> 처리 내용 </c:when>

    <c:otherwise> 처리 내용 </c:otherwise>

</c:choose>

## ★⑥ 반복문 태그

- <c:forEach items="객체명" begin="시작값" end="끝 값" step="증감식" var="변수명">

- forEach 태그는 배열, 컬렉션 또는 맵에 저장되어 있는 값들을 순차적으로 처리할 때 사용합니다.

- 배열의 경우는 객체의 배열뿐만 아니라 기본 데이터 타입의 배열도 알맞게 처리를 하며 기본 데이터 타입은 래퍼 클래스를 통해 처리합니다.

## 2) Sample

### Sample 1) 회원탈퇴

```
[MyFirstWeb]
- delete_check.jsp
- user_delete.jsp
- user_delete_ok.jsp
- MemberDAO.java
==> public int deleteUser(String id){
// index.jsp ★
```

### Sample 2) EL(Expression Language)

```
[MyFirstWeb> WebContent> EL> ...]
// 기본
# el_basic.jsp          ! 🖱 Ctrl + F11
# el_obj.jsp            ! 🖱 Ctrl + F11
# el_obj_ok.jsp

// MVC 패턴 2 이해하기
# Thermometer.java (kr.co.koo.el)
# el_thermometer.jsp  ! 🖱 Ctrl + F11
```

#### cf. HashMap

: 파이썬의 dictionary

List : index로 관리 (index, value) ex> class[0], class[1], class[2]

- classes.add(i, value)
- classes.get(i)
- List<String>

Map : 데이터의 이름, 데이터에 별명 붙임 (key, value) ex> ("멍멍이":"홍길동"), ("야옹이":"이순신"), ("메뚜기":"강감찬")

- classes.put(key, value) ex> classes.put("메뚜기", "강감찬");
- classes.get(key)
- Map<String, String>

#### cf. EL 사용 이유

자바의 객체(class)를 EL에서 바로 사용 가능

### Sample 3) JSTL

```
[MyFirstWeb4\WebContent\JSTL]
```

```
# jstl_if.jsp          ! 🖱 Ctrl + F11
# jstl_choose.jsp      ! 🖱 Ctrl + F11

# jstl_choose2.jsp     ! 🖱 Ctrl + F11
# jstl_choose3.jsp     ! 🖱 Ctrl + F11
# jstl_foreach.jsp     ! 🖱 Ctrl + F11
```

## 15. JSTL - MVC2

### 1) 개념

#### \* MVC Model 2 Architecture

- 모델2 구조는 웹 브라우저의 요청을 하나의 서블릿이 받으며 서블릿은 그 요청을 알맞게 처리한 후, 그 결과를 보여줄 JSP 페이지로 포워딩합니다.

- 이 구조의 특징은 웹 브라우저의 모든 요청을 단일 진입점, 즉 하나의 서블릿에서 처리한다는 점입니다. 하나의 서블릿이 모든 요청을 받기 때문에 서블릿은 웹 브라우저의 요청을 구분하는 방법이 필요합니다.

==> join.jsp + join\_ok.jsp ==> controller

- MVC 모델2 구조의 핵심은 모델(Model)은 비즈니스와 관련된 로직만 처리하면 되며 사용자에게 보일 화면이나 요청의 흐름 제어에 대해서는 전혀 처리하지 않으며, 뷰(View)는 사용자에게 알맞은 화면을 보여주는 역할만 수행할 뿐, 비즈니스 로직이나 요청 흐름 제어 등을 처리하지 않습니다. 또한 컨트롤러(Controller)는 사용자의 요청에 대해서 알맞은 모델을 사용하고 사용자에게 보여줄 뷰를 선택합니다.

- MVC 모델2 구조를 사용함으로써 코드의 유지보수가 쉬워지며 어플리케이션을 쉽게 확장할 수 있습니다.

#### \* 1) MVC의 컨트롤러: 서블릿

==> 요청 분석. 방법? url

- 모델 2 구조에서 서블릿은 MVC 패턴의 컨트롤러 역할을 합니다. 서블릿은 웹 브라우저의 요청과 웹 어플리케이션의 전체적인 흐름을 제어합니다.

- 컨트롤러의 흐름 제어 처리 로직

1. 웹 브라우저가 전송한 HTTP 요청을 받아 요청방식에 맞게 doGet(), doPost()를 호출함.
2. 웹 브라우저가 어떤 기능을 요청했는지 분석함.
3. 모델을 사용하여 요청한 기능을 수행.
4. 모델로부터 전달받은 결과물을 알맞게 가공한 후, request나 session의 setAttribute() 메서드를 이용하여 결과값을 속성에 저장. 이렇게 저장한 결과값은 뷰인 JSP에서 사용함.
5. 웹 브라우저에 결과를 전송할 JSP 페이지를 선택한 후, 해당 JSP로 포워딩(혹은 리다이렉트)함.

#### \* 2) MVC의 뷰: JSP

- 모델 2 구조에서 ★ JSP는 뷰 역할을 담당합니다. 뷰 역할을 하는 JSP는 컨트롤러에서 request 객체나 session 객체에 저장된 데이터를 사용하여 웹 브라우저에 알맞은 화면을 출력합니다.

- 뷰 역할을 하는 JSP는 웹 브라우저가 요청한 결과를 보여주는 ① 프레젠테이션의 역할을 할 뿐만 아니라 ② 웹 브라우저의 요청을 컨트롤러에 전달해주는 매개체가 되기도합니다.

- 즉, 뷰 역할을 하는 JSP는 웹 브라우저가 지속적으로 컨트롤러에 요청을 보낼 수 있는 링크나 폼을 제공해서 웹 브라우저가 업무 흐름에 따라 컨트롤러에게 알맞은 요청을 보낼 수 있도록 합니다.

### \* 3) MVC의 모델

- 컨트롤러는 서블릿을 통해 구현하고 뷰는 JSP를 통해서 구현하는데 모델은 명확하게 어떤 것을 통해서 구현한다는 규칙은 없습니다. 단지 비즈니스 로직을 처리해주면 모델이 될 수 있습니다.

- 컨트롤러의 서블릿이 웹 브라우저의 요청을 분석하여 알맞은 모델을 호출하면서부터 모델의 기능이 시작됩니다. 모델은 컨트롤러가 요청한 작업을 처리한 후 알맞은 결과를 컨트롤러에게 전달하는데, 이때 처리한 결과값을 저장하는 객체로 보통 자바빈을 사용합니다.

### \* URL-Pattern

1. 디렉토리 패턴: 디렉토리 형태로 서버의 해당 컴포넌트를 찾아서 실행하는 구조입니다.

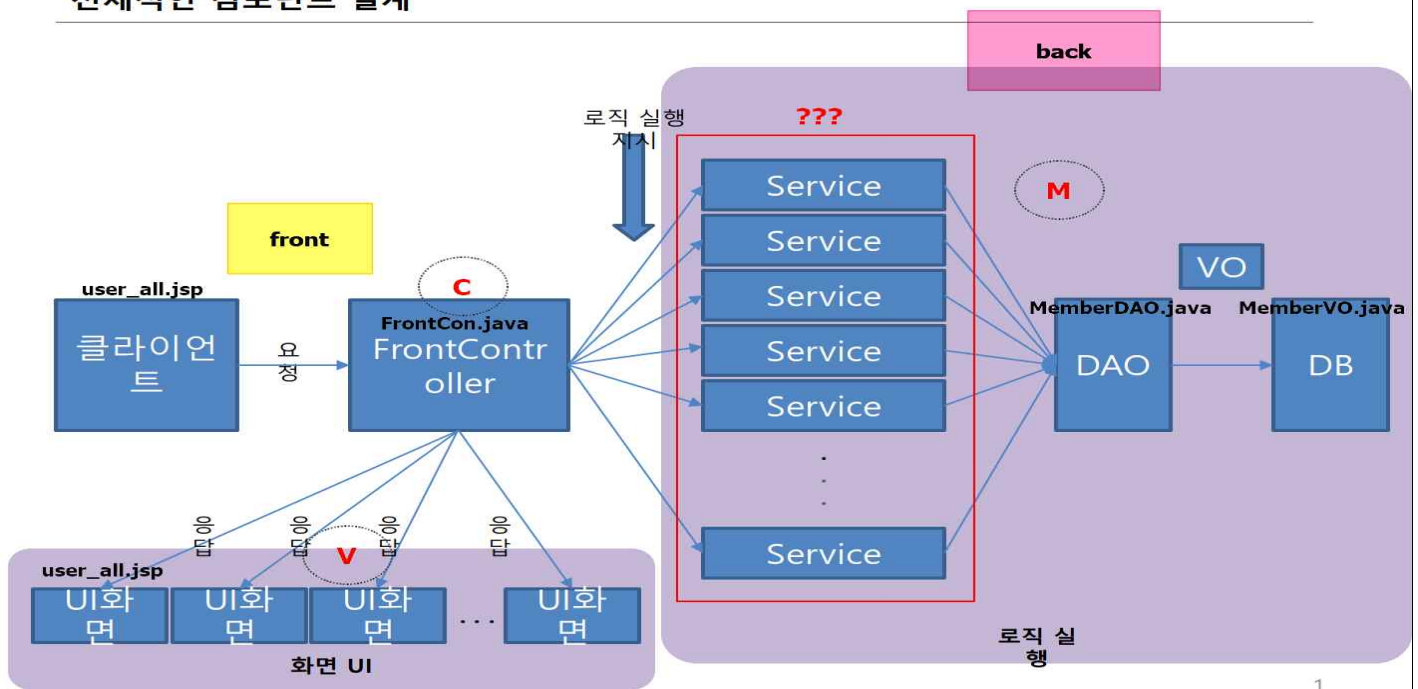
ex) `http://localhost:8181/cr/Hello` --> `/Hello` 서블릿  
`http://localhost:8181/cr/World` --> `/World` 서블릿

2. 확장자 패턴: 확장자 형태로 서버의 해당 컴포넌트를 찾아서 실행하는 구조입니다.

ex) `http://localhost:8181/cr/Hello.do` --> `*.do` 서블릿  
`http://localhost:8181/cr/World.do` --> `*.do` 서블릿

### \* MVC 2

#### 전체적인 컴포넌트 설계



## 2) Sample

### [작업순서]

#### Sample 1) Test

##### 1) MVC 2 Basic

[WebContentWcontrollerW]

- controller\_basic.jsp ★

[kr.co.koo.controller]

- FrontCon.java

##### 2) MVC 2 웹 적용

- user\_all.jsp ★

- MemberDAO.java

=> public ArrayList<MemberVO> membersAll()

- user\_all\_result.jsp

## 16. MVC 2-2

### [작업순서]

#### 0) 게시판용 db table 작성

# MyFirstWeb 게시판 테이블

```
create table board (
    board_no int primary key auto_increment,
    board_name varchar(20) not null,
    board_title varchar(100) not null,
    board_content varchar(300),
    board_date datetime default current_timestamp,
    board_hit int default 0
);
select * from board;
```

#### 1) 게시판 꾸미기

- BoardVO.java //kr.co.koo.board.model
- BoardDAO.java
- board\_list.jsp
- board\_write\_view.jsp
- BoardController.java // kr.co.koo.board.controller
- **IBoardService.java**
  - BWriteService.java // kr.co.koo.board.service

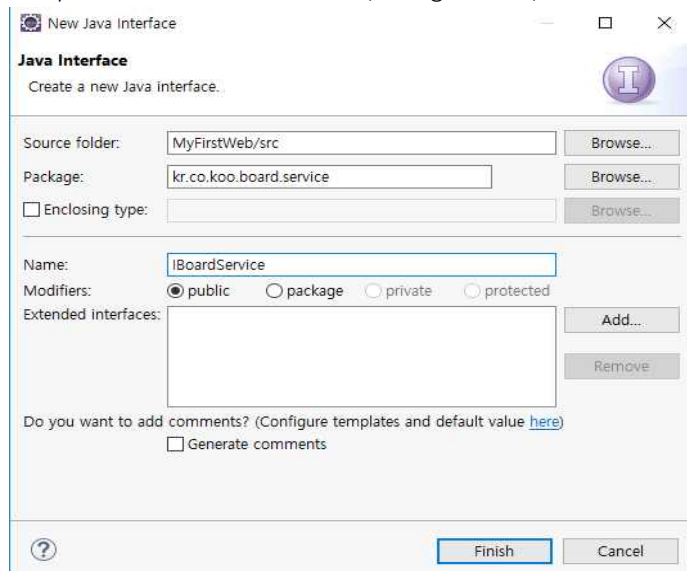
// 요청 처리

```
String bName = request.getParameter("bName");
String bTitle = request.getParameter("bTitle");
String bContent = request.getParameter("bContent");
BoardDAO
...
- BListService.java
```

#### 2) 글 목록 생성

- board\_list.jsp // 제목 클릭하면...
- board\_content.jsp
- BoardController.java
- BListService.java
- BoardVO.java

=> public BoardVO content(String bNum)



## 17. MVC 2-3

### \*\*\* 작업순서

1) html코드가 엔터와 공백 인식하지 못하는 문제점 해결

# BoardDAO.java

```
bContent = bContent.replace("WrWn", "<br>");  
bContent = bContent.replace("Wu0020", "&nbsp;");
```

2) 조회수 처리

# BoardDAO.java

```
private void upHit(String bNum)
```

3) 게시물 수정

① board\_content.jsp

=> line. 74 a href 링크 연결 (수정)

```
<a href="BUpdateView.board2?bNum=${article.bNum}">[수정]</a>&nbsp;
```

② BoardController.java

```
}else if(com.equals("/board/BUpdateView.board2")) {
```

...

추가

③ BUpdateViewService.java // 수정 게시물 select

```
kr.co.koo.board.service
```

④ BoardDAO.java

```
public BoardVO updateView(String bNum)
```

⑤ board\_update\_view.jsp

```
<input type="hidden" name="bNum" value="${article.bNum}">
```

4) 게시물 수정 2

```
<%--
```

1. form태그의 action url = "BUpdate.board"
  2. Service클래스 이름은 BUpdateService
  3. DAO의 수정 메서드이름은 updateArticle
  4. 수정 완료 후 안내할 UI는 게시물 목록입니다.
- ```
--%>
```

① board\_update\_view.jsp

=> *BUpdate.board2*

③ BoardController.java

② BUpdateService.java

④ BoardDAO.java

```
public int updateArticle(int bNum, String bName, String bTitle, String bContent)
```

⑤ BoardController.java

```
ui = "/board/BList.board2";
```



## 5) 로그인 연결, 작성자 필드 수정

### ① board\_list.jsp

- line.4 ~ 11

```
<%
    if(session.getAttribute("user_id") == null){
%>
<script>
    alert("로그인이 필요한 서비스입니다.")
    document.location.href="/jsp_pj"
</script>
<% } %>
```

### ② board\_write\_view.jsp

```
<input type="hidden" name="bName" value="${user_name}">
${user_name}(${user_id})
```

### ③ board\_update\_view.jsp

바로 위와 동일!!!

## 6) modal 추가 // 회원가입

- header.jsp에 다음 내용 추가

===== modal [start]=====

```
<!-- Modal -->
<div class="modal fade" id="myModal" role="dialog">
    <div class="modal-dialog">
        <!-- Modal content-->
        <div class="modal-content">
            <div class="modal-header">
                <h4 class="modal-title">My First Web 회원 가입</h4>
                <button type="button" class="close" data-dismiss="modal">×</button>
            </div>
            <div class="modal-body">
                <form action="members/join_ok.jsp" method="post" class="form-horizontal">
                    <div class="form-group">
                        <label for="inputId3" class="col-sm-3 control-label">* ID</label>
                        <div class="col-sm-6">
                            <input type="text" class="form-control" id="inputId3" name="id">
                        </div>
                    </div>
                    <div class="form-group">
                        <label for="inputPassword3" class="col-sm-3 control-label">*
                            비밀번호</label>
                        <div class="col-sm-6">
                            <input type="password" class="form-control" id="inputPassword3"
                                name="pw" placeholder="8자 이상으로 입력하세요.">
                        </div>
                    </div>
                    <div class="form-group">
                        <label for="inputPassword3" class="col-sm-3 control-label">*
                            비밀번호 확인</label>
                        <div class="col-sm-6">
                            <input type="password" class="form-control"
                                id="inputPassword3-confirm" placeholder="위와 동일한 비밀번호를 입력하세요.">
                        </div>
                    </div>
                </form>
            </div>
        </div>
    </div>
</div>
```

```

</div>
<div class="form-group">
  <label for="inputName3" class="col-sm-3 control-label">★
    이름</label>
  <div class="col-sm-6">
    <input type="text" class="form-control" id="inputName3"
      name="name">
  </div>
</div>
<div class="form-group">
  <label for="inputaddr3" class="col-sm-3 control-label">★
    주소</label>
  <div class="col-sm-6">
    <input type="text" class="form-control" id="inputaddr3"
      name="address">
  </div>
</div>

<div class="form-group">
  <label for="inputEmail3" class="col-sm-3 control-label">★
    이메일</label>
  <div class="col-sm-6">
    <input type="text" class="form-control" id="inputEmail3"
      name="email" placeholder="example@myworld.com">
  </div>
</div>

<div class="modal-footer">
  <input type="submit" class="btn btn-default" value="완료"
    onclick="return confirm('회원 가입을 완료하시겠습니까?')">
  <button type="button" class="btn btn-default"
    data-dismiss="modal">닫기</button>
</div>
</form>
</div>
</div>
</div>

```

===== modal [end]=====

## 7) delete 로직 추가

<%--

1. board\_content.jsp의 삭제 요청 url = "BDelete.board2"
2. Service클래스 이름은 BDeleteService
3. DAO의 삭제 메서드이름은 deleteArticle
4. 삭제 완료 후 안내할 UI는 게시물 목록입니다.

--%>

① board\_content.jsp

=> <a href="BDelete.board2?bNum=\${article.bNum}">[삭제]</a>&nbsp;

② BoardController.java

③ BDeleteService.java

④ BoardDAO.java

public void deleteArticle(int bNum) {

⑤ BoardController.java

ui = "/board/BList.board2";