KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

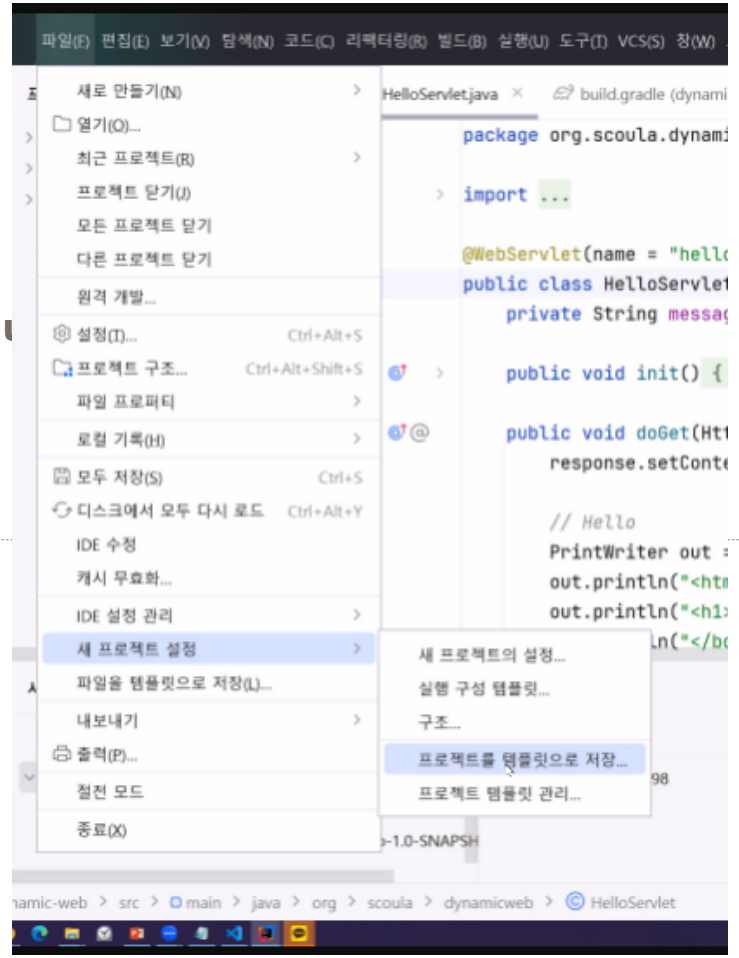
JSP의 이해

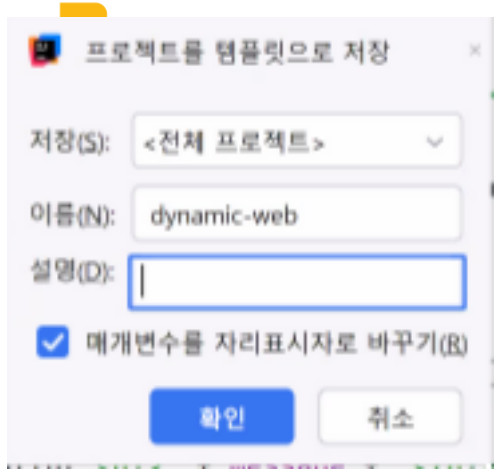
[KB] IT's Your Life

실전 프로젝트는
초기부터 구성구조가
복잡하다

템플릿으로 만들어 활용하자!

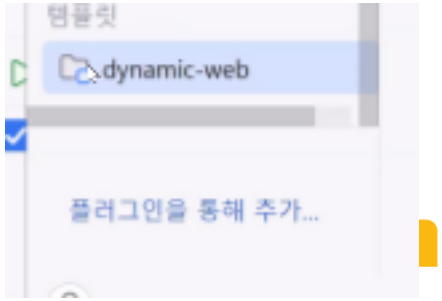
번외 프로젝트 전체를 템플릿으로 만드는 방법





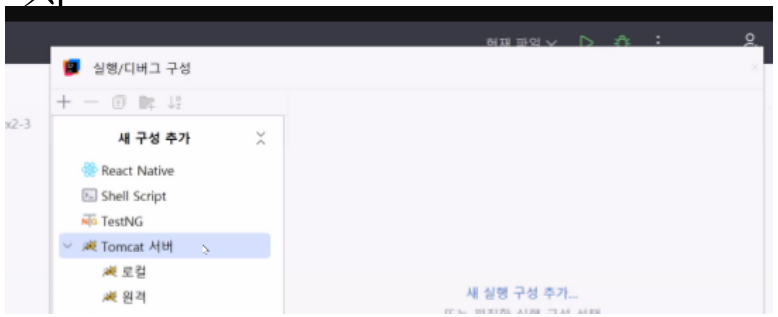
현재프로젝트가
intellij의
템플릿으로 등록됨

새프로젝트 생성시
템플릿 메뉴에 있음



페이지 8에서 하는 과정을 또하면 되고

톰캣이 없어져 있으니 수동으로
구성해주자



1 JSP의 기본요소

- **HTML과 태그로 구성되어 있으며, 태그 안에 자바 코드를 삽입하여 구현**

html사이사이에 로직이나 변수를 넣겠다.
=> 태그안에 자바코드를 삽입하여 구현!

- **JSP에서 사용 가능한 기본 태그 요소**
  - HTML 요소
  - JSP 스크립팅 요소
  - JSP 표준 액션 태그 요소
  - EL 요소
  - 커스텀 태그 라이브러리 요소

JSP 스크립팅 요소

- JSP 페이지가 서블릿으로 변환시 JSP 엔진에 의해 처리
- <% %> 형식을 가짐 html이 아닌 jsp의 무언가이다

jsp가지고 웹앱을 만드는것. jsp가지고 비즈니스로직도 만드는 실습
=> model1
=>실전에 쓰지 않음

스크립팅 요소	예제
Comment tag	<%-- 주석 -- %>
Directive tag	<%@ 지시자 %>
Declaration tag	<%! 자바 선언문 %>
Scriptlet tag	<% 자바 코드 %>
Expression tag	<%= 자바 표현식 %>

젤 많이 씀
나머지는
잘안쓰게돼..

바로 출력하라 라는 뜻.

vue의 {{ }}랑 비슷

최종 값이 하나 나와야함!

주석 태그

JSP 페이지 주석

```
<%-- JSP 태그 주석 --%>
```

자바 코드 주석

```
<%
 /*
      여러 줄 주석
 */
 // 한 줄 주석
%>
```

- **JSP 지시어(Directive tag)**
  - 해당 페이지를 어떻게 처리할 것인지에 대한 설정 정보를 지시하는 용도

- **page 지시어** 많이쓰는 지시어
  - <%@ page 속성명="속성값" 속성명2="속성값2" %>

  - 하나의 page 지시어를 사용하여 여러 개의 속성을 동시에 지정 가능
  - 여러 개의 page 지시어를 사용하여 개별적으로 속성 지정 가능
  - import 속성을 제외한 나머지 속성들은 중복 지정이 불가능

페이지 지시어 속성들

속성명	기본값	설 명	예
contentType	text/html	응답 시의 MIME 타입	contentType="text/html"
import	없음	import할 패키지 지정	import="java.util.ArrayList"
session	true	HttpSession 사용 여부	session="true"
errorPage	없음	에러 발생 시 처리할 에러 페이지 지정	errorPage="error.jsp"
isErrorPage	false	현재 페이지를 에러 페이지로 지정	isErrorPage="false"
language	java	사용할 언어 설정	language="java"
extends	없음	상속받은 클래스명 지정	extends="패키지포함 클래스명"
buffer	8kb	출력 버퍼 크기 지정	buffer="8kb"
autoFlush	true	출력 버퍼가 모두 찼을 경우에 처리 방법 지정	autoFlush="true"
isThreadSafe	true	스레드 동시 실행 여부	isThreadSafe="true"
info	없음	현재 페이지 정보 출력	info="장바구니 페이지"
pageEncoding	ISO-8859-1	현재 페이지의 문자 인코딩 지정	pageEncoding="EUC-KR"
isELIgnored	false	EL 표기법 사용 여부 지정	isELIgnored="false"

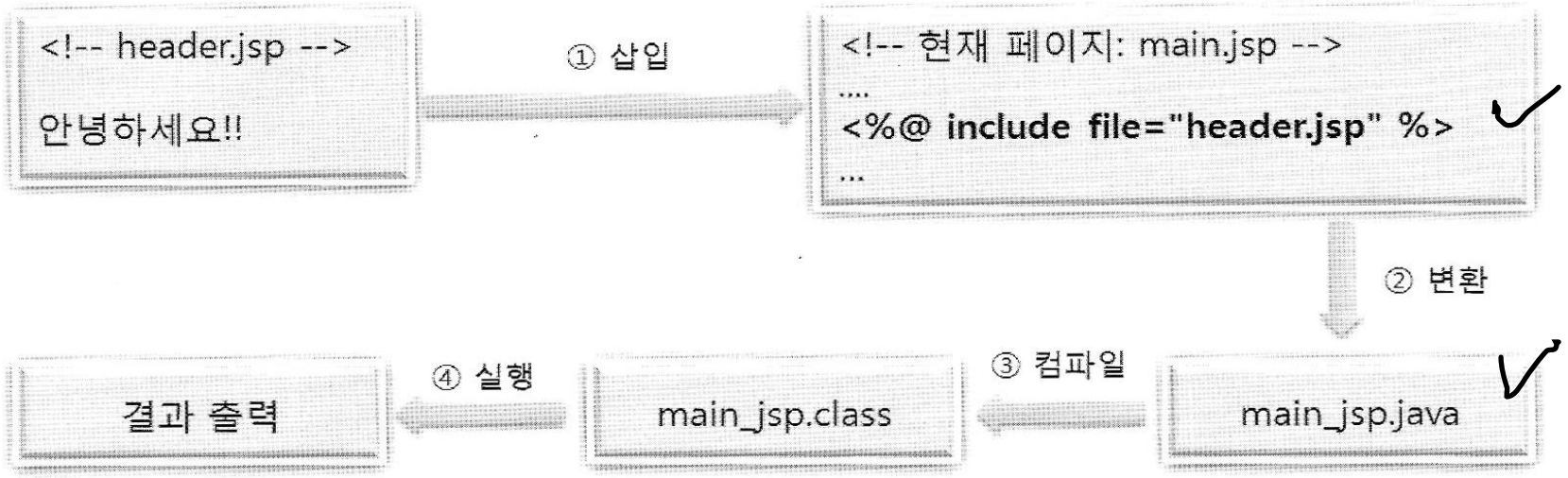
include 지시어

- 현재 페이지에서 다른 html 및 jsp 페이지를 삽입할 수 있는 방법 제공
- 각 페이지 마다 공통적인 내용을 가지는 경우 매번 작성하지 않고 삽입
- footer, header, side 영역의 공통 내용 처리
- 정적인 방법으로 삽입 컴파일타임때 삽입됨

  <%@ include file="포함할 페이지"%>

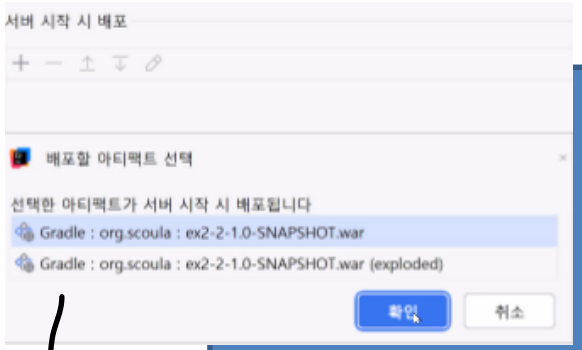
공통요소를 매번 추가하면
오버헤드=>
인클루드 지시어 활용.

헤더,푸터, 사이드메뉴를 다 분리시켜
필요할때 해당위치에 인클루드 지시어 사용!



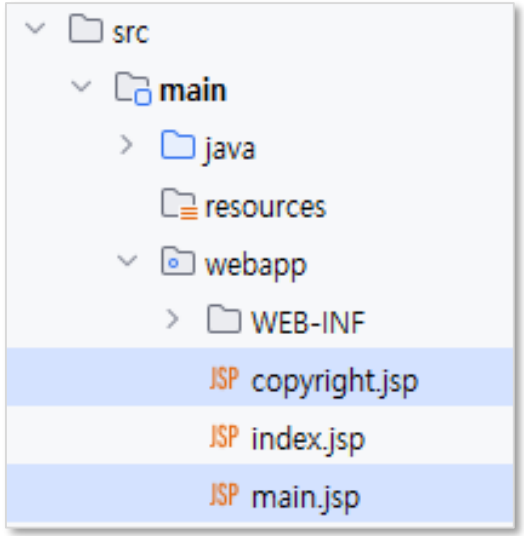
include directive 태그 실습

실습을 위해 전에 만든
dynamic_web 프로젝트 디렉을
전체 복붙하자



main.jsp

copyright.jsp 푸터jsp



복붙해서 프로젝트를
복사하면

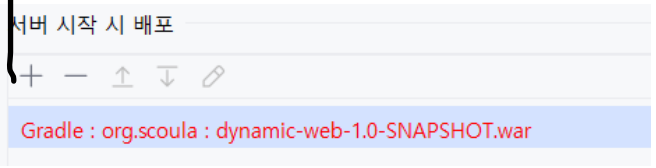
아티팩트아이디가 안 맞는다.

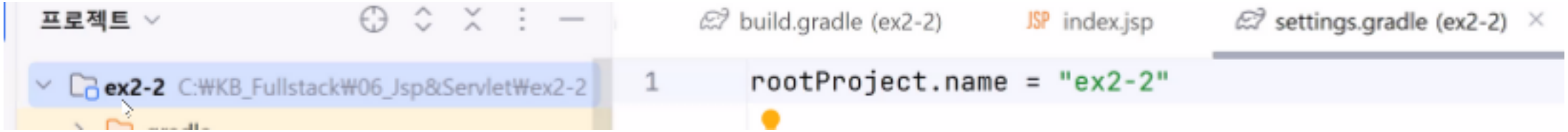
프로젝트 루트디렉토리와 안 맞
setting.gradle에 들어가서

rootProject.name이름을
변경하자

gradle 동기화를 하는거 잊지말

톰캣war도 맞춰야함 수동으로 워래것을 제거하고 추가하자





copyright.jsp

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

<div>
    Copyright (c) .....  All rights reserved.
</div>
```

main.jsp

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

  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>메인 화면 페이지</title>
</head>
<body>
  <h1>include 지시어 태그 실습</h1>
  내용 수정중입니다. 다음줄에 삽입이 됩니다. <br>

  <%@ include file="copyright.jsp" %>
</body>
</html>
```

상대경로로 지정

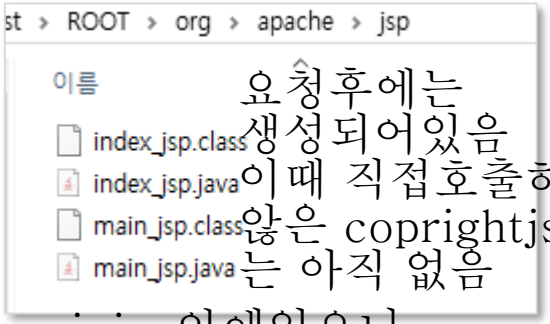
아직 —main.jsp나 copy—right.jsp를 요청보내지 않아서

jsp가 변환된 파일들이 들어있진 않다.

요청을 어떻게 보낼까
직접 아래처럼 타이ㅂ핑할 수 있고

인텔리제이에서 킬수있도록 브라우저 오픈 버튼을 제공한다

- localhost:8080/main.jsp



요청후에는 생성되어있음
이때 직접호출하지 않은 coprightjsp는 아직 없음

mainjsp안에있으니 번역하는 입장에서는 모름

taglib 지시어

- JSP에서 외부 라이브러리로 만든 태그를 지정할 때 사용
- JSTL(JSP Standard Tag Library)

```
<%@ taglib uri="TLD 파일 URI" prefix="네임스페이스명" %>
```

JSP 선언 태그(Declare tag)

- 자바의 멤버 변수와 메서드를 선언할 때 사용

```
<%!
  멤버 변수 선언
  메서드 선언;
%>

<%! int i; %>
<%! int i=0; %>
<%! public  void setInfo(int i) {
  if( i < 3) …
} %>
```

스크립틀릿 태그(Scriptlet tag)

```
<%
  자바 코드 문장;
  문장2;
%>
```

```
<% if (Calendar.getInstance( ).get(Calendar.AM_PM) == Calendar.AM) {%>
    Good Morning
<% } else { %>
    Good Afternoon
<% } %>
```

비즈니스로직 쓰기
드럽게 힘드네여;

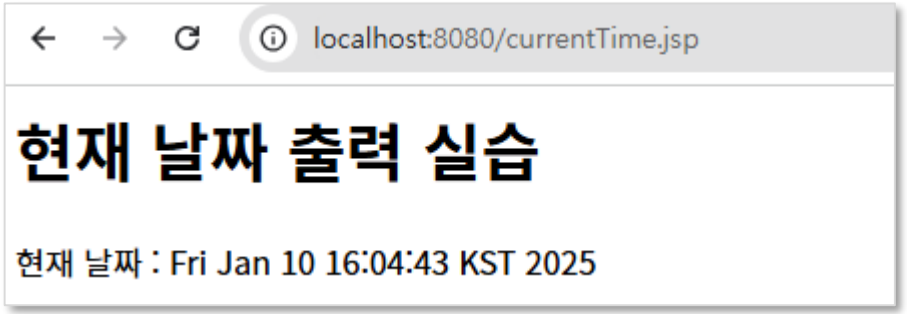
currentTime.jsp

```jsp
<%@ page import="java.util.Date" %>
<%@ page contentType="text/html; charset=UTF-8"%>
<!DOCTYPE>
<html lang="ko">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>스크립트릿 실습</title>
</head>
<body>
  <h1>현재 날짜 출력 실습</h1>
  <%
    Date d = new Date();
  %>
  현재 날짜 : <%= d %>
</body>
</html>
```

널일 경우 그냥 null문자열을 출력함

없는 변수를 출력하려 해도 잡아주지 않는다

- localhost:8080/currentTime.jsp



표현식 태그(Expression tag)

<%=변수%>
<%= 리턴 값이 있는 메서드 %>

```
<%=(new java.ul.Date()).toLocaleString() %>
```

내장 객체

따로 선언 없이 바로 사용할 수 있는 것들
변환 과정에서 자동으로 만들어지는 변수들이죠.

내장 객체(내장 변수)	설명
request	HttpServletRequest 객체 참조
response	HttpServletResponse 객체 참조
out	웹 브라우저 출력에 사용되는 JspWriter 객체 참조
session	HttpSession 객체 참조(session="true"인 경우에만 사용 가능)
application	ServletContext 객체 참조
page	자바 클래스의 this와 동일
config	ServletConfig 객체 참조
exception	발생되는 예외의 Throwable 객체에 대한 참조 (isErrorPage="true"인 경우에만 사용 가능)

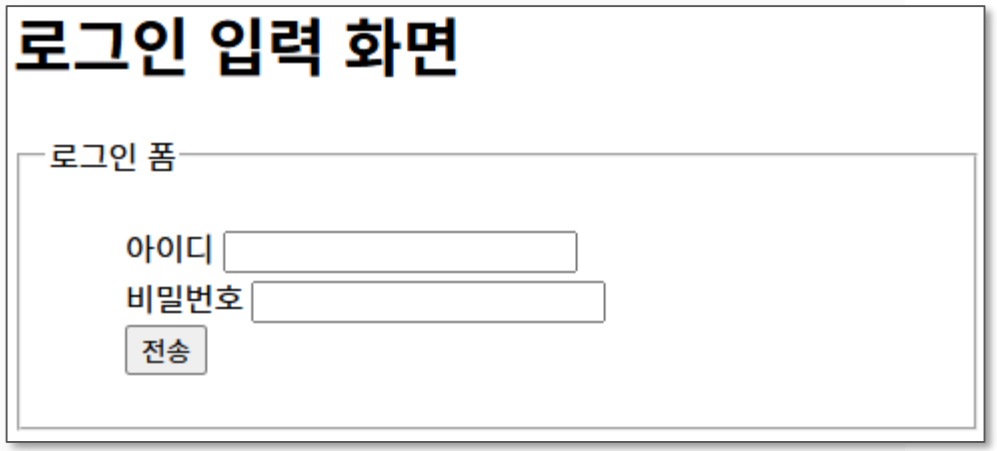
request 객체

- 서블릿의 HttServlett Request 객체를 참조하는 변수

request.jsp

```
<body>
<h1>로그인 입력 화면</h1>
<form action="loginInfo.jsp" method="get">
  <fieldset>
    <legend>로그인 폼</legend>
    <ul style="list-style: none" >
      <li>
       <label for="userid">아이디</label>
       <input type="text" name="userid" id="userid">
        </li>
        <li>
       <label for="password">비밀번호</label>
       <input type="password" name="password" id="password">
        </li>
      <li><input type="submit" value="전송"></li>
    </ul>
  </fieldset>
</form>
</body>
</html>
```

http://localhost:8080/request.jsp



loginInfo.jsp request 내장 객체 실습

```
<body>
<h1>로그인 입력 파라미터 출력</h1>
<%
  String userid = request.getParameter("userid");
  String Password = request.getParameter("password");
%>
아이디값: <%= userid %> <br>
비밀번호: <%= Password %> <br>
</body>
```

request.jsp

```
<form action="loginInfo.jsp" method="get">
  <fieldset>
    <legend>로그인 폼</legend>
    <ul  style="list-style: none">
      <li>
       <label for="userid">아이디</label>
       <input type="text" name="userid" >
       </li>
       <li>
       <label for="Password">비밀번호</label>
       <input type="password" name="password" >
       </li>
      <li><input type="submit" value="전송"></li>
    </ul>
  </fieldset>
</form>
```

/loginInfo.jsp?userid=hong&password=1234

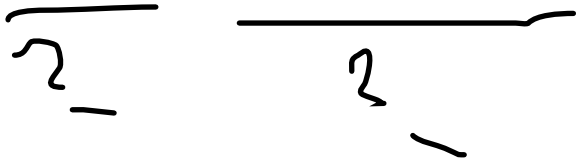
로그인 입력 파라미터 출력

아이디값: hong
비밀번호: 1234

그렇다면 전달되지 않는 데이터에 대해서
getParameter하면 어떻게 될까
명은 있는데 밸류가 없다면 비어있는 문자열
아예 명도 없으면 null이
반환된다

response 내장 객체

- HttpServletResponse 객체를 참조하는 변수
- 쿠키 저장, 요청 redirect 처리 등 수행



out 내장 객체

- 서블릿의 PrintWriter 객체와 동일한 기능의 JspWriter 객체를 참조
- 데이터 출력시 사용

- expression tag 이용

```
<%
  String name="홍길동";
%>
<%=name%>
```

- out 내장 객체 이용

```
<%
  String name="홍길동";
  out.print(name);
%>
```

서블릿스타일로 writer을 이용하여 바로 출력

out.jsp

```
<body>
<%
  String name="홍길동";

  out.print("이것은 out 내장 객체로 출력 : " + name +"<br>" );
%>
이것은 expression tag로 출력 : <%= name  %>
</body>
```

http://localhost:8080/out.jsp

이것은 out 내장 객체로 출력 : 홍길동
이것은 expression tag로 출력 : 홍길동

application 내장 객체

- ServletContext 객체를 참조하는 변수
- 로그 및 파일 접근, application scope에 저장되는 속성 설정 등을 구현

count.jsp

```
<body>
<h1>방문자수 설정하기 화면</h1>
<%! int count; %>
<%
  count++;
    application.setAttribute("countValue", count);
%>
현재 방문자수 : <%= count %>
</body>
```

앱 레벨 유지

키 밸류

stateless
요청에 대해 응답하고나면
서버에서는 해당 응답에 대한 상태를 잊는다.
심플하다
하지만 요청고 ㅏ요청사이 데이터가
유지되어야하는 상황이라면
(로그인 유지 상황)
적절하지 않다.

유지도 범위마다 다르다.

로그인한시점에서 로그아웃시점가지만
유지할래 => 세션레이어

한페이지 내부에서만 유지되어라 페이지레벨
(기본)

톰캣이 어플을 기동할때부터 종료할때까지 유지
(어플리케이션 레벨)

리퀘스트레벨도 있다. 아마 리퀘스트 요청되면
유지 끝나는?

count_view.jsp

```
<body>
<h1>방문자수 조회하기 화면</h1>
<%
    int count = (Integer)application.getAttribute("countValue");
%>
현재까지 총 방문자수: <%= count %>
</body>
```

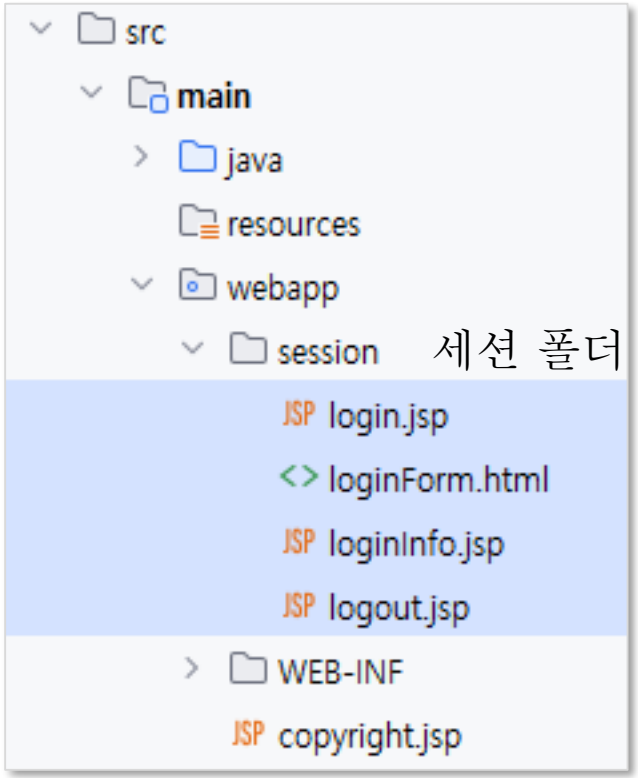
http://localhost:8080/count.jsp

http://localhost:8080/count_view.jsp

session 내장 객체

세션레이어!

- HttpSession 객체를 참조하는 변수
- page 지시어에서 session="false"를 지정하면 사용 불가



세션 폴더 만들기 성격별로 나누는게 좋다

경로는 /session.login.jsp

session/loginForm.html

```
<body>
<h1>session 내장 객체 실습</h1>
<form action="login.jsp" method="get">
    <fieldset>
        <legend>로그인 폼</legend>
        <ul style="list-style: none">
            <li>
                <label for="userid">아이디</label>
                <input type="text" name="userid" id="userid">
            </li>
            <li>
                <label for="password">비밀번호</label>
                <input type="password" name="password" id="password">
            </li>
            <li><input type="submit" value="전송"></li>
        </ul>
    </fieldset>
</form>
</body>
```

session/login.jsp

```
<body>
<h1>로그인 정보 세션 저장</h1>
<%
  String id = request.getParameter("userid");
  if (id == null) {
    response.sendRedirect("loginForm.html");
  } else {
    String pw = request.getParameter("password");
    session.setAttribute("userid", id);
    session.setAttribute("password", pw);
    out.print("안녕하세요 " + id + "<br>");
    out.print("<a href='loginInfo.jsp'>정보보기</a>");
  }
%>
</body>
```

쿼리스트링@@?userid="...."password="....."

다시 폼으로 돌려보내

세션 스코프에다가 저장. 세션 만큼 유지되는

loginInfo.jsp

```
<body>
<h1>로그인 정보 보기</h1>
<%
  String id = (String) session.getAttribute("userid");
  if (id == null) {
    response.sendRedirect("loginForm.html");

  } else {
    String pw = (String) session.getAttribute("password");
    out.print("사용자 아이디값: " + id + "<br>");
    out.print("사용자 비밀번호값: " + pw + "<br>");
    out.print("<a href='logout.jsp'>로그 아웃</a>");

  }
%>
</body>
```

반드시 캐스팅이 필요하다

톰캣이 재기동되면

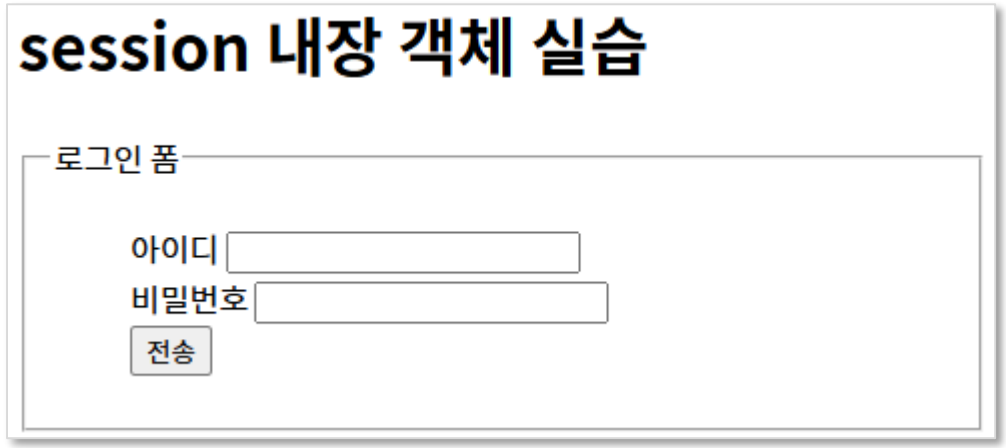
모든 스코프(범위)의
유지 정보는
다 날라갈 수 있으니

파일을 다 만들어지고나서
그때 되서야 온전한 테스트
가 가능하다.

logout.jsp

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

http://localhost:8080/session/loginForm.html



http://localhost:8080/session/logout.jsp

http://localhost:8080/session/login.jsp?userid=hong&password=1234

http://localhost:8080/session/loginInfo.jsp

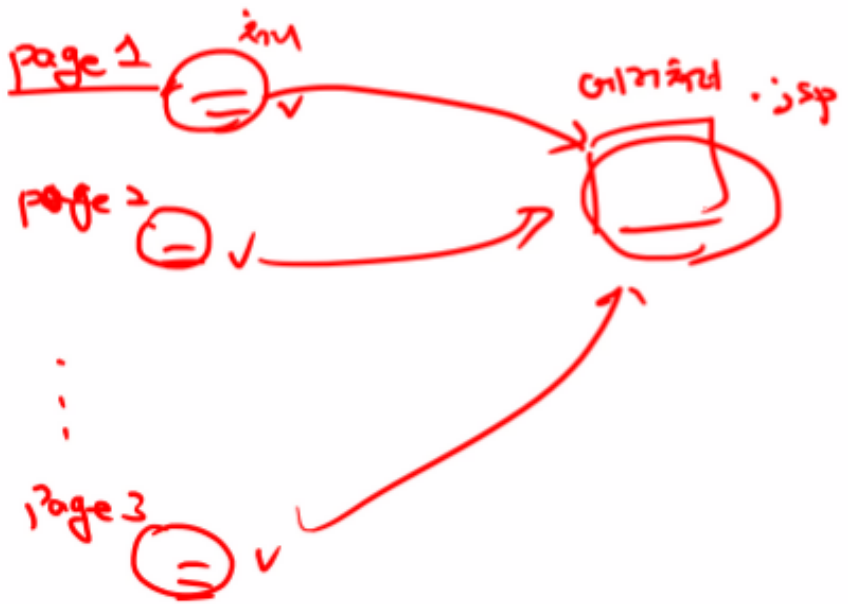
exception 객체

- JSP 페이지 내에서 발생된 예외 클래스인 java.lang.Throwable 객체를 참조하는 변수
- 예외 발생시 예외를 처리하는 JSP 페이지를 따로 구현할 목적으로 사용
- 반드시 page 지시어에서 isErrorPage="true"인 경우에만 사용 가능
  - 기본값은 false

개발자는 각 페이지 단위로 예외를 바라보고 각 페이지별로
해당 예외를 처리하는게 좋다

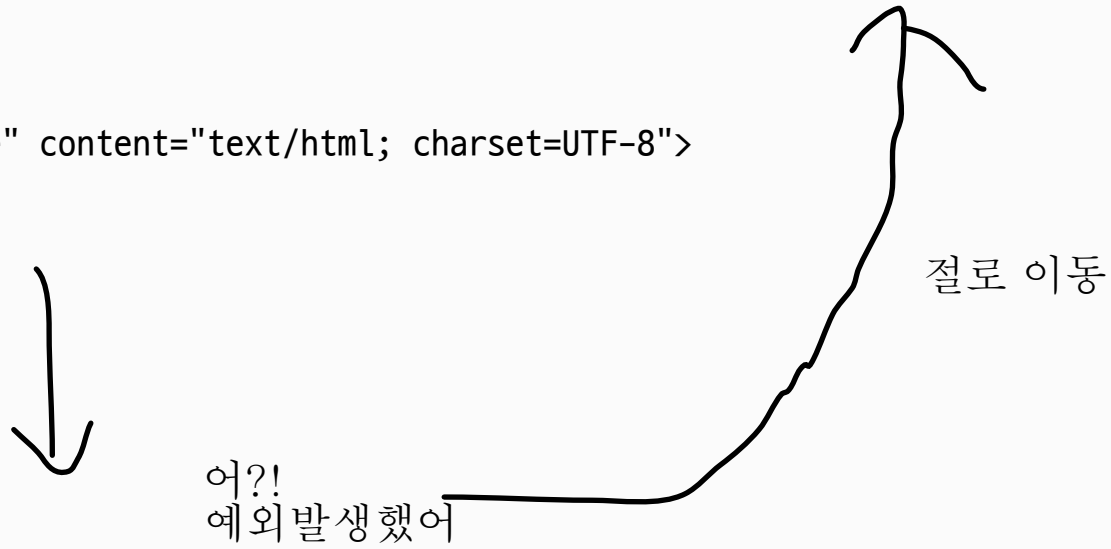
하지만 사용자 입장에서는 바로 처리할 필요는 없다!
사용자 입장에선는 해당 페이지가 잠시 오류가 생겼다. 라고 표시만 해도 됨.
같은 내용을 출력해야하니 하나의 페이지로 하는게 효율적 관리도 좋ㄷ음

하나의 페이지로 이동하여 출력하고 예외처리도 하나의 페이지에서 하는게 좋음.
예외처리를 한군데에 모으는 것.



divide.jsp

```
<%@ page contentType="text/html; charset=UTF-8" language="java" errorPage="error.jsp"%>
<!DOCTYPE>
<html lang="ko">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>exception 실습</title>
</head>
<body>
<%
     // 0으로 나누어 예외 발생
     int n = 4/0;
%>
</body>
</html>
```



error.jsp

에러페이지라는 것을 알려줘야함

```
<%@ page contentType="text/html; charset=UTF-8" language="java"  isErrorPage="true"%>
<!DOCTYPE>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>exception 실습</title>
</head>
<body>
  <h1>divide.jsp 발생된 예외를 처리하는 페이지</h1>
  <%
     out.print("발생된 예외는 : " + exception.getMessage());
  %>
</body>
</html>
```

이 값이 트루면
내장 객체가 하나 추가됨. exception객체가 추가됨

이때 사용가능한 exception객체는

앞 페이지(오류가 발생한)에서
전달한 것이다. !!!!

http://localhost:8080/divide.jsp

divide.jsp 발생된 예외를 처리하는 페이지

발생된 예외는 : / by zero

표준 액션 태그

어떤 액션이 일어날까. 내부적으로는 메소드가 호출됨.

- 스크립트릿 태그 사용을 대체하기 위한 목적
- 기능이 제한적임 → JSTL로 보완

표준 액션 태그	설명
<jsp:useBean />	자바빈 컴포넌트를 사용하기 위한 액션 태그
<jsp:setProperty />	자바빈 인스턴스에 데이터를 저장하기 위한 액션 태그
<jsp:getProperty />	자바빈 인스턴스에서 데이터를 얻기 위한 액션 태그
<jsp:include />	**제공된 JSP 파일을 삽입하기 위한 액션 태그**
<jsp:forward />	제공된 JSP 파일로 forward하기 위한 액션 태그
<jsp:param />	include 및 forward할 때 요청 파라미터를 추가하기 위한 액션 태그

인클루되는 행위가 일어남

접두어:태그명

jsp에서 만든
태그명이다
라는 뜻

<jsp:include/> 액션태그

<%@ include 머시기 %> 디렉티브랑 거의 유사함.

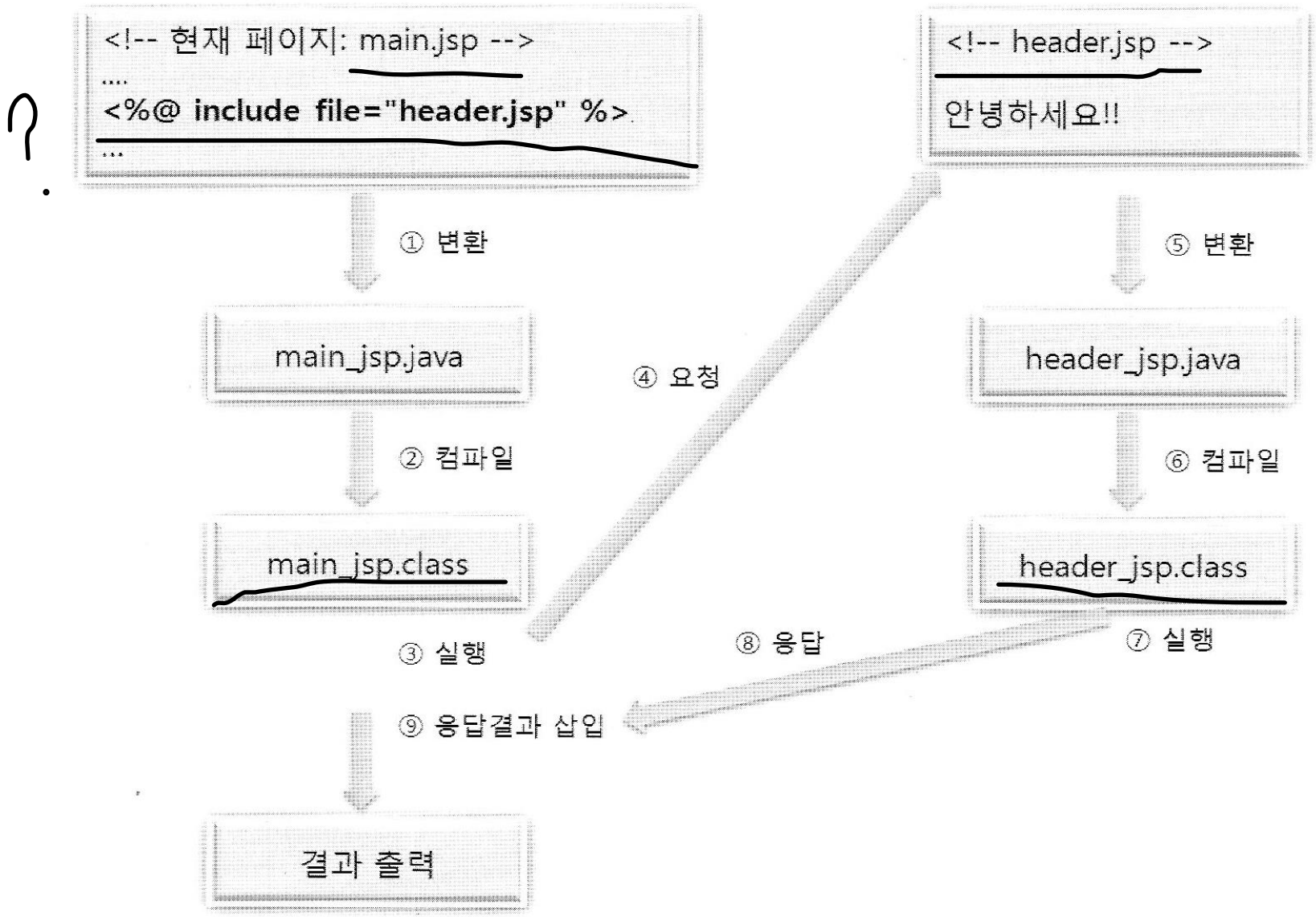
차이점은 위에껀 정적이고

액션태그 include는 동적이다.

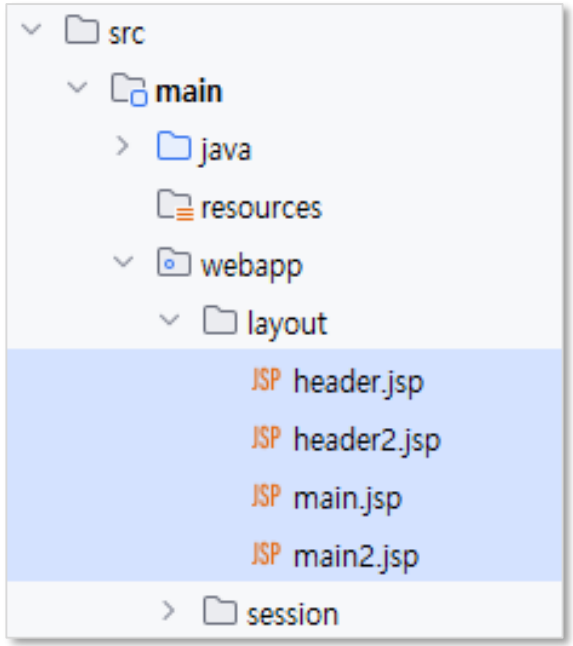
눈으로 보는 결과는 같은데 과정이 틀리다.

```
<jsp:include page="삽입될 페이지" flush="true"/>
```

- 동적으로 삽입이 이루어짐.
- 내용물이 소스 차원에 포함되는 것이 아니고 지정한 페이지의 실행 결과가 포함됨(메서드 호출과 유사)


<!-- 현재 페이지: main.jsp -->
....
<%@ include file="header.jsp" %>
...
<!-- header.jsp -->
안녕하세요!!
① 변환
⑤ 변환
main_jsp.java
header_jsp.java
② 컴파일
⑥ 컴파일
main_jsp.class
header_jsp.class
④ 요청
③ 실행
⑧ 응답
⑦ 실행
⑨ 응답결과 삽입
결과 출력

<jsp:include/> 액션태그 실습



layout/header.jsp

```
<%@page import="java.util.Calendar"%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%
    Calendar cal = Calendar.getInstance();
    int hour = cal.get(Calendar.HOUR_OF_DAY);
    int minute = cal.get(Calendar.MINUTE);
    int second = cal.get(Calendar.SECOND);
%>
현재 시간은  <%= hour %> 시 <%= minute %> 분 <%= second %> 초  입니다.
```

layout/main.jsp

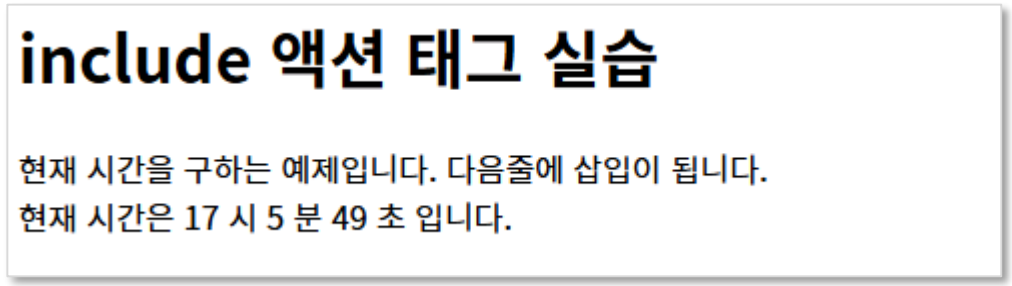
```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<!DOCTYPE>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>include 액션 실습</title>
</head>
<body>
  <h1>include 액션 태그 실습</h1>
  현재 시간을 구하는 예제입니다. 다음줄에 삽입이 됩니다.<br>

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

</body>
</html>
```

디렉티브 include와 또 다른점
또다른 ㅂ여 정보를 같이 넘겨줄수있음

http://localhost:8080/layout/main.jsp



포함될 페이지에 임의의 파라미터 값을 지정하여 요청 가능

```
<jsp:include page="삽입될 페이지" flush="true">
    <jsp:param name="파라미터 명" value="값" />
    <jsp:param name="파라미터 명" value="값" />
</jsp:include>
```

추가정보를 이렇게 넘길수있음

layout/header2.jsp

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

<%

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

    Calendar cal = Calendar.getInstance();
    int hour = cal.get(Calendar.HOUR_OF_DAY);
    int minute = cal.get(Calendar.MINUTE);
    int second = cal.get(Calendar.SECOND);
%>
안녕하세요 당신의  닉네임은  <%= nickName %>입니다.<br>
현재 시간은  <%= hour %> 시 <%= minute %> 분 <%= second %> 초  입니다.
```

가급적으로 이렇게 자바코드를 쓰지않고

태그를 사용해야한다.

model2에서는 그렇게 해야한다.

아까 앞서 말한
model1방식 jsp로만 개발
은 개오바임

그래서 model2방식이용

요청은 서블릿이 비즈니스로직 돌려서
결과를 얻어.
이를 아까 본 scop객체에 저장.
setAttribute해서.

그리고 나서
jsp로 이동(forwarding)
출력만을 함.
getAtrribute해서.
출력할 거 만들고

응답하면 됨!!!
model2를 하면 돼!!

jsp가 많이 줄어들었기에
역량이 필요가 덜하다

layout/main2.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<!DOCTYPE>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>include 액션 실습</title>
</head>
<body>
  <h1>include  태그 실습</h1>
  현재 시간을 구하는 예제입니다. 다음줄에 삽입이 됩니다. <br>

  <jsp:include page="header2.jsp" flush="true" >
    <jsp:param  name="nickName"  value="Mr Hong" />
  </jsp:include>
</body>
</html>
```

http://localhost:8080/layout/main2.jsp

include 태그 실습

현재 시간을 구하는 예제입니다. 다음줄에 삽입이 됩니다.
안녕하세요 당신의 닉네임은 Mr Hong입니다.
현재 시간은 17 시 6 분 28 초 입니다.