JSP Basic

Java Server Page!

📌HTML vs JSP (차이점!)📌

1. HTML 과 다르게 , **톰캣 서버가 번역한 결과를 HTML 태그로 변환하여 웹 브라우저에 표시**함

JSP가 HTML과 다르다는 것은 아래의 간단한 예를 통해서 확인해볼 수 있다!

jsp 페이지의 동작 과정

- <% %> 는 스크립트릿이라고 하는데, 앞서 살펴봤던 컨텍스트패스와 jsp 파일에 대한 요청으로 구성된 요청 url를 서버에서 받으면 해당 부분에 대해서 서버가 번역/해석하여 html로 응답해준다. 그 응답결과는 화면에 나타나게 된다!

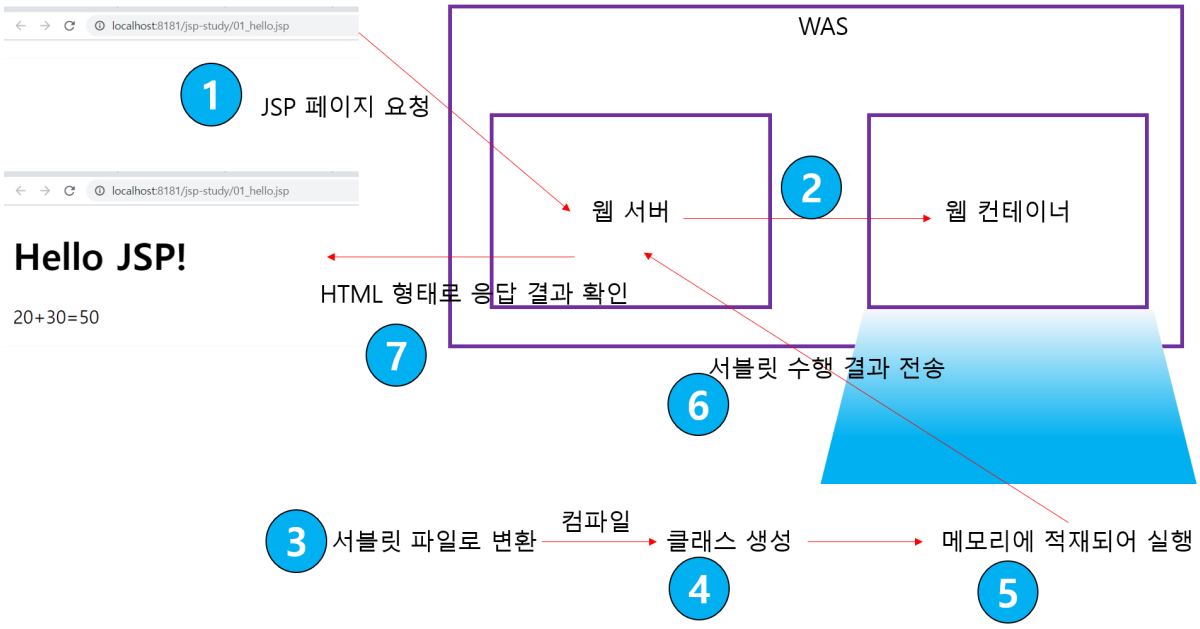
2. JSP 는 자바나 톰캣 서버가 설치되어 있어야 실행됨

🍑 JSP는 서블릿과 비교했을 때, out.println()을 통해 html 문서를 출력하는 과정을 생략하고 태그를 바로 작성할 수 있다는 장점이 존재(하지만, 단점이라면(JSP) JSP는 스트립태그 등 내부에 자바코드를 작성할 수 있다는 점!)

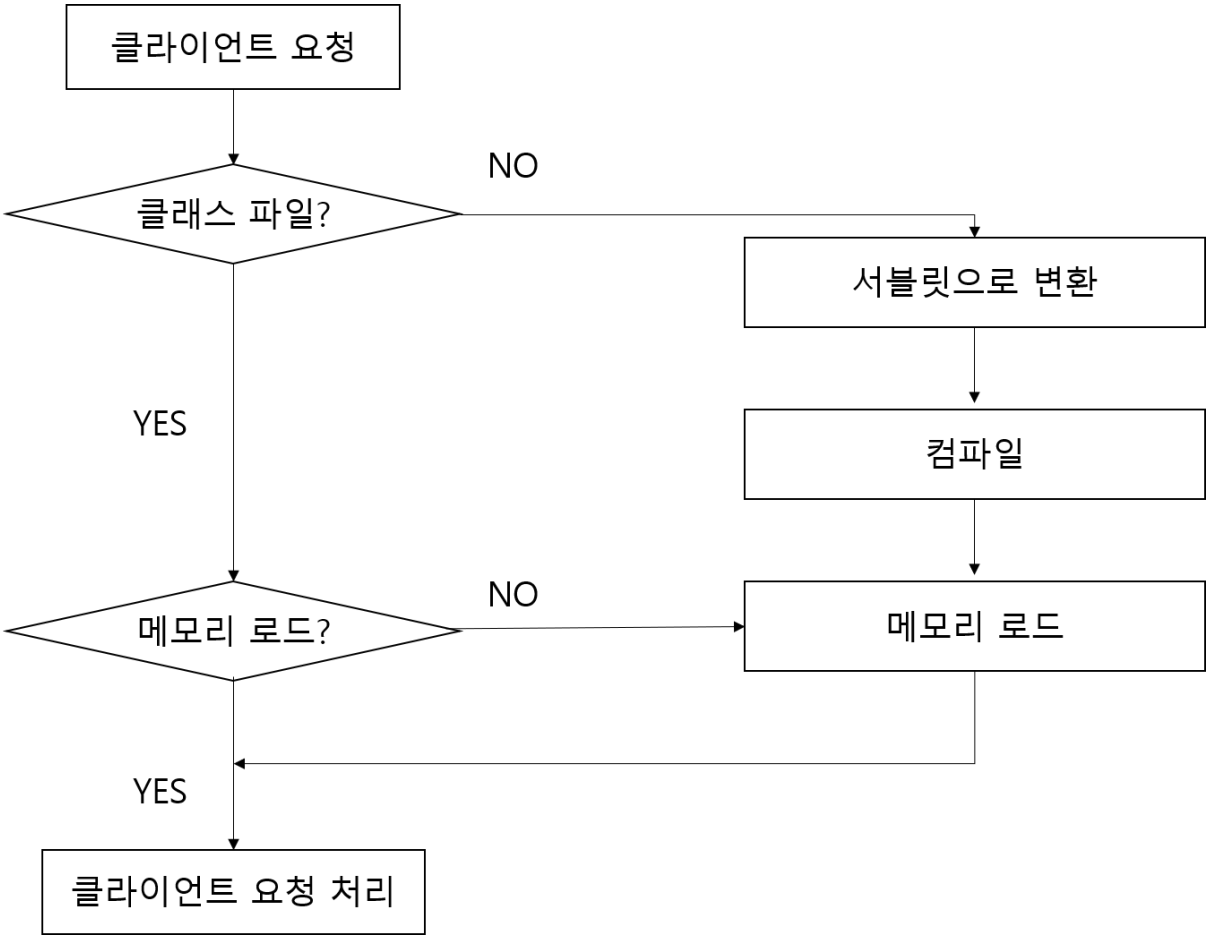
⚠ JSP에는 서버에서 보낸 데이터에 따라 값이 바뀔 수 있는 내용들을 출력할 필요가 있을 때 사용!

⚠ HTML에는 변화가 없는 단순 상수값 출력 시 사용

📌JSP 페이지가 서블릿 컨테이너에서 실행되는 과정📌



서블릿 컨테이너가 JSP 페이지를 실행하는 순서



서블릿 컨테이너가 JSP 페이지를 실행하는 순서에 대한 순서도

먼저 두 가지 그림 중 첫 번째 그림은 서블릿 컨테이너가 jsp 페이지를 실행하는 데까지의 과정을 나타내는 그림이고, 마지막 그림은 이를 순서도로 나타낸 것이다!

JSP 페이지는 그림을 참조하여 생각, 접근해보면 아래와 같은 단계로 실행된다

1. 브라우저에서 jsp 페이지에 대해서 웹 서버에게 요청["클라이언트 요청"]
2. 웹 서버가 서블릿 컨테이너에게 jsp 페이지에 대한 요청을 전달
3. 요청에 대해서 서블릿 컨테이너가 서블릿 파일을 생성["서블릿 파일로 변환"]
   ▶ **JSP 페이지가 최초로 요청되었을 때 단 한 번만 수행되는 과정!** 페이지가 재요청되면, 메모리에 존재하는 서블릿 파일로 서비스됨! ▶ **서블릿과 실행 시간이 많이 차이나지 않음!**
4. 생성된 서블릿 파일을 **컴파일**하여 **서블릿 클래스 파일** 생성(.class)[클래스 파일인지 확인]

5. 서블릿파일이 메모리에 적재되어 수행됨["메모리 로드, 수행"]
6. 서블릿 수행 결과가 웹 서버에 전송됨
7. 사용자가 서블릿의 출력결과인 HTML 형태의 문서를 응답결과로 받아볼 수 있게 됨 ["클라이언트 요청 처리"]

참고사항-jsp 파일이 서블릿 파일로 변환되면서 나타나는 서블릿 파일의 특징

💡 1.service() : get 방식과 post 방식 구분없이 모든 요청에 대한 처리 수행 (_jspService())

💡 2.서비스 메서드(1) 뒤에는 서블릿에서의 PrintWriter 객체와 같은 내장 객체를 JSP 에서 사용할 수 있도록 선언하고 할당하는 부분

*내장 객체 : 이미 객체가 생성되어 있어서 바로 사용하면 되는 상태의 객체

📌JSP 기본 태그📌

동적 페이지를 구현하기 위한 JSP 태그 종류

Aa 종류	사용용도	형식
스크립트릿 (scriptlet)	자바 코드를 기술 -**_jspService()** 메서드 내부에 들어감 - static 변수나 전역변수가 아닌 인스턴스 변수가 들어가야 함!	<% %>
선언 (declaration)	**변수와 메서드** 선언 -JSP 페이지가 초기화될 때 초기화되어서, 페이지 내의 어떠한 스크립트릿이나 표현식에서도 접근, 사용 가능	<%! %>
표현식 (expression)	**계산식이나 함수를 호출한 결과**를 **문자열 형태**로 출력	<%= %>
주석 (comment)	JSP 페이지에 설명을 넣음 -스트립태그 내에서의 주석: // /**/ - HTML 코드 내에서 별도 태그로 jsp 주석 사용: <%— —%>	<%-- --%>
지시자 (Directive)	JSP 페이지의 속성을 지정함	<%@ %>

1. 선언 <%! %> - 관례적으로 HTML 문서 맨 위에 위치시킴!

```
<%!
  String str ="안녕하세요!";//변수
   public int abs(int n){ //메서드
    if(n < 0 ){
      return (-1)*n;
   }
   return n;
}
%>
```

- 선언부분에서 선언한 변수는 jsp 문서 전체에서 사용 가능
- ; 을 꼭 붙이기!
- 변수, 메서드를 넣을 수 있음!

⚠ 변수, 메서드 정의 ▶ 선언문

⚠ 변수값 출력, 메서드 호출 ▶ 스크립트릿 태그

선언문과 스크립트릿 태그만 이용해서 간단한 jsp 문서를 만들어보자!

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
 <%!
  //선언문
  //한줄주석
  /*여러
  줄
  주
  석
  */
  String str= "Hello JSP!";
  int num1 = 20;
  int num2 = 30;
  public int sum(int a, int b){
    return (a+b);
  }
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>JSP</title>
</head>
```

```
<body>

  <%
    //스크립트릿
    out.println(str+"<br>");
    out.println(num1+"+"+num2+"="+(sum(num1,num2)));
  %>
</body>
</html>
```

그러면 위의 서블릿 컨테이너 동작 과정 그림 중 일부였던

Hello JSP!

20 + 30 = 50

이 출력될 수 있다!

⚠ 스크립트릿에 위치한 변수와 선언문에 있는 변수의 차이점 ⚠

- 스크립트릿에 위치한 변수 : 지역변수!
- 선언문에 있는 변수 : 전역변수!

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
 <%!
  //선언문
  //한줄주석
  /*여러
  줄
  주
  석
  */
  String str= "Hello JSP!";
  int num1 = 20;
  int num2 = 30;
  int global = 0;
  public int sum(int a, int b){
    return (a+b);
  }
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
```

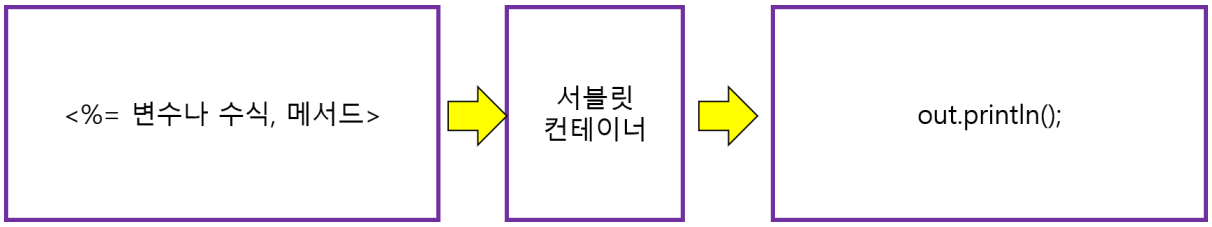
```
<title>JSP</title>
</head>
<body>

  <%
    //스크립트릿
    int local = 0;
    out.println(str+"<br>");
    out.println(num1+"+"+num2+"="+(sum(num1,num2))+"<br>");
    out.println("global: "+(++global)+"<br>");
    out.println("local: "+(++local));
  %>
</body>
</html>
```

Hello JSP!
20+30=50
global: 6
local: 1

위의 예시처럼 선언문과 스크립트릿 부분 각각에 변수를 위치시키고, 전위증가를 해주면, 선언문에 있던 변수인 global은 전역변수이기 때문에 변수가 공유되어 계속 1씩 증가한다. 하지만, 스크립트릿 태그 부분에 위치한 local 변수는 지역변수이기 때문에, 실행될때마다 초기화되어 계속 1만 출력된다

2. 표현식 <%= >



- 변수의 값이나 계산식, 함수를 호출한 결과를 문자열 형태로 출력

- out.println()과 같은 기능!
- ; (세미콜론)은 작성하지 않아야 오류나지 않음!

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
 <%!
  //선언문
  //한줄주석
  /*여러
  줄
  주
  석
  */
  String str= "Hello JSP!";
  int num1 = 20;
  int num2 = 30;
  int global = 0;
  public int sum(int a, int b){
    return (a+b);
  }
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>JSP</title>
</head>
<body>
  <%
    //스크립트릿
    int local = 0;
  %>
  <%--jsp 페이지 내 주석 --%>
  <%--표현식 --%>
  <%=str +"<br>"%>
  <%=num1 %>+<%=num2 %>+<%=sum(num1,num2) +"<br>"%>
  <%=num1 +"+"+num2+"="+sum(num1,num2) +"<br>"%>
  <%=(++global)+"<br>" %>
  <%=(++local) +"<br>"%>
</body>
</html>
```

위에서 테스트한 과정을 <%= >를 이용해서 사용해보자!

결과는 위와 동일한데, 20+30=50 부분만 하나의 <%=%>에 작성하는 방식과 나누어 작성하는 방식으로 작성해보았기 때문에 두 번 나타나는 결과를 볼 수 있다!

3. 지시자(Directive)

- JSP 파일의 최상단에 위치
- JSP 페이지에 대한 전체 속성 지정

<%@+page/include/taglib(◀지시자) 지시자_속성="값"%> 로 사용

- 가장 흔한 예시는 JSP 파일을 생성할 때, 가장 먼저 맞이하는 부분인 아래와 같은 부분이다!

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

지시자 종류

Aa 종류	사용용도
page	JSP 페이지 전반적인 환경 설정
include	현재 페이지에 다른 파일의 내용을 삽입할 때 사용(html의 iframe태그와 의미는 유사)
taglib	태그 라이브러리에서 태그를 꺼내와서 사용할 수 있는 기능 제공

(1) page 지시자

<%@page 속성="값"%>

- JSP **페이지에 여러 가지 정보**를 나타내기 위해서 사용

page 지시자의 속성

Aa 속성	값	특징

Aa 속성	값	특징
language	기본값- java (생략 가능)	-JSP에서 사용할 언어 지정 -JSP에서는 스크립트 언어로 java 이외에는 사용하지 못함 -사용하는 목적: 명시적으로 정의하기 위함 -기본적으로 생략해주어도, language="java"가 서버측에서 인식될 수는 있음
extends	상속을 할 부모클래스	-특정 클래스로부터 상속받기 위함 -서블릿 컨테이너에서 기본적으로 처리하기 때문에 개발자가 특별히 지정하는 경우는 없음
import	사용할 패키지의 풀네임	-JSP에서 사용할 패키지를 사용하기 위함
session	true,false	-세션[=웹 서버의 서비스를 받는 사용자를 구분하는 단위]을 사용할 지 그 여부를 인지
buffer	-버퍼를 사용하지 않을 경우: none -버퍼를 사용할 경우: xKB(예: 8KB-기본값)	-출력 스트림을 출력하기 전에 버퍼에 데이터를 속성에 기재한 용량만큼씩 나누어 출력 -버퍼 용량을 작게하여 여러번 전송하는 것이 대기시간을 줄일 수 있음
autoFlush	-true,false -기본값: true -autoflush="false": buffer를 사용해야 사용 가능 -버퍼를 사용하되, 버퍼가 다 차면 에러를 발생시킬 지, 버퍼를 비우고 새로운 데이터를 받을지에 대한 속성	-flush: 강제로 버퍼를 비우기(== 데이터 송신) -버퍼가 꽉 차면 자동으로 버퍼를 비우도록 함 -만약, 데이터가 다 차기 전에 버퍼를 비우고 싶다면, autoflush="false"와 out내장객체.flush()로 비워줄 수 있음
isThreadSafe	-true: 비동기화를 막음으로써 순차적으로! 동기화!됨! -false: 비동기화되어 여러 사용자가 접근!(따라서 일반적으로 사용하지 않음)	-멀티스레드: 하나의 프로그램을 여러개의 스레드로 처리하는 것 ▶ 자원 공유를 통해 오버헤드 감소 -JSP: 멀티스레드 처리 가능 -동기화: 순차적! From Top to Bottom -비동기화: 비순차적! 모듈같은 느낌! -멀티스레드의 장점이자 단점인 자원공유로 인하여 하나의 자원에 여러 사용자가 동시 접근할 경우 문제가 발생할 수 있음! (예: 은행 입출금) -위의 자원공유로 인한 비동기화를 막고자할 경우에는 isThreadSafe 속성을 이용!
info	현재 JSP페이지에 대한 간략한 설명	-프로그래머가 페이지 관리를 손쉽게 하기 위해 설정 -설정하지 않더라도 페이지 처리에는 영향x -현재 페이지에 대한 정보를 알려줌
errorPage	에러가 발생했을 때 연결될 페이지(예: error.jsp)	-에러가 발생했을 때 보여줄 에러 페이지 설정

Aa 속성	값	특징
isErrorPage	-true: 현재 페이지가 에러 페이지가 되고, exception 객체를 이용하여 예외의 원인을 알 수 있게 됨 -false: 현재 페이지가 에러페이지가 아님(기본값)	-현재 페이지가 에러 페이지인지 아닌지 설정
contentType	-MIME 타입(예: text/html, text/javascript) -MIME 타입의 기본값: text/html -charset을 동시에 ;로 구분하여 적어줄 수 있음 -charset 기본값 : ISO-8859-1	-JSP 페이지의 MIME Type(Multipurpose Internet Mail Extensions) 결정
pageEncoding	-기본값 : ISO-8859-1	-페이지에 대한 인코딩 방식 지정

page 지시자의 속성은 위와 같이 사용할 수 있다!

예시만 확인하고 가자!

<%@page language="java" extends="javax.servlet.jsp.HttpJspBase" import="java.util.Calendar" info="메인페이지" errorPage="error.jsp" isErrorPage="false" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

간단히 하나의 예로써 page 지시자의 속성을 다져보자

해석해보자면,

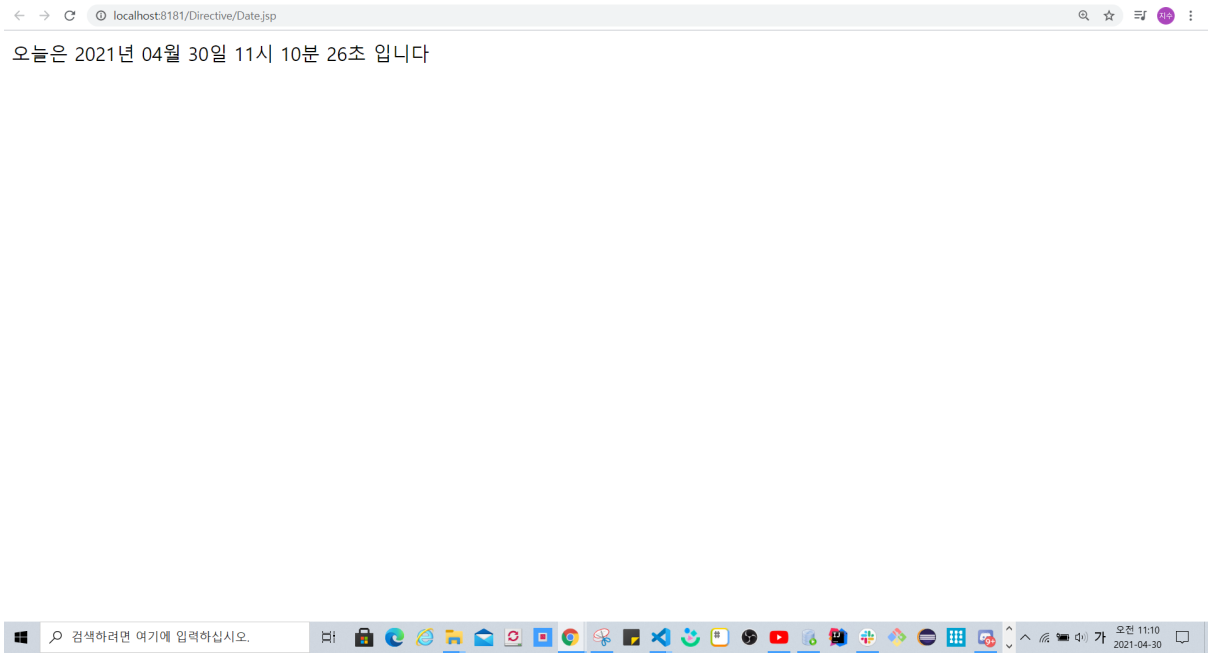
- JSP 문서의 언어는 자바로,
- HttpJspBase 클래스를 상속받고

- java.util 패키지의 Calendar 클래스를 이용하며
- 프로그래머에게 알려줄 수 있는 현재 jsp 페이지에 대한 설명은 "메인페이지"이다
- 그리고 지금 페이지는 에러페이지가 아닌데, 에러가 발생하면 컨텍스트패스 내에 존재하는 error.jsp 파일을 띄워줄 것이다
- 그리고 현재 문서는 html 문서로 작성될 것이고, 인코딩은 UTF-8 방식을 따를 것이다
- 또한 페이지에 대한 문자 인코딩은 UTF-8 방식을 기준으로 할 것이다

라는 것을 의미한다

가. import 속성 연습

```
<%@ page import="java.text.SimpleDateFormat" %>
<%@ page import="java.util.Calendar" %>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" %>
<%
    Calendar cal = Calendar.getInstance();
    SimpleDateFormat s1 = new SimpleDateFormat("YYYY년 MM월 dd일 hh시 mm분 ss초");
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>page 지시자-import, extends</title>
</head>
<body>
  <%="오늘은 "+s1.format(cal.getTime())+" 입니다<br>" %>
</body>
</html>
```



나. errorPage, isErrorPage 속성을 이용한 에러 페이지 생성, 연결

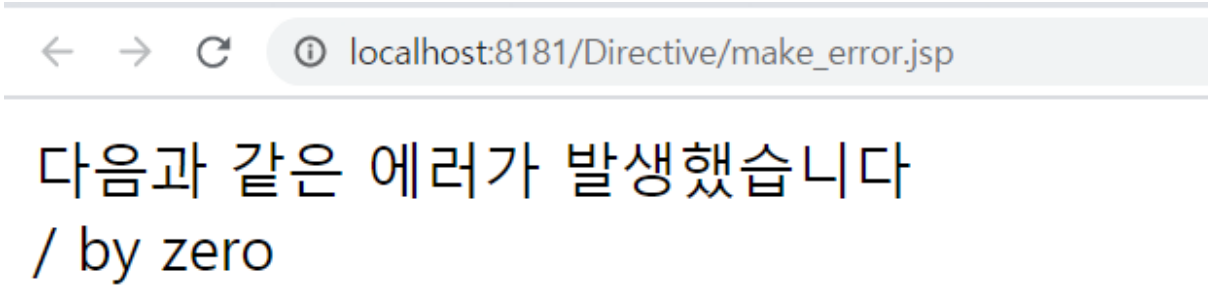
메인페이지 - make_error.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%--error 페이지 연결 --%>
<%@ page isErrorPage="false" errorPage="error.jsp" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>에러 페이지 만들기</title>
</head>
<body>
<%=2 %>/<%=0 %>=<%=2/0 %>
</body>
</html>
```

에러페이지 error.jsp

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

```
<%@ page isErrorPage = "true" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>ERROR Page</title>
</head>
<body>
  <%="다음과 같은 에러가 발생했습니다<br>" %>
  <%=exception.getMessage() %>
</body>
</html>
```



- 메인 페이지는 에러페이지가 아니지만, 에러를 expression을 통해서 발생시켜주고, 그랬을 때 에러 페이지를 연결해준다
- 그러면 에러페이지에 대한 처리 결과로 NaN 에러 메시지를 띄운다!

(2) include 지시자

- 현재 jsp 파일 내부에 다른 HTML 문서 및 JSP 페이지 내용 삽입
- 속성은 단 하나! file 속성만 존재!(값은 URL!)

(예시)


```
<%include file="footer.jsp"%>
```

- 아래와 같이 body 영역 최하단에 넣어준다!

*메인페이지와 서브페이지에 모두 들어갈 footer.jsp 파일을 만들어주고, 각 페이지에 include 지시자를 이용해서 넣어주자!

-footer.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Footer</title>
</head>
<body>
  <div id="copyright">
    All rights reserved ©jsJeong
  </div>
</body>
</html>
```

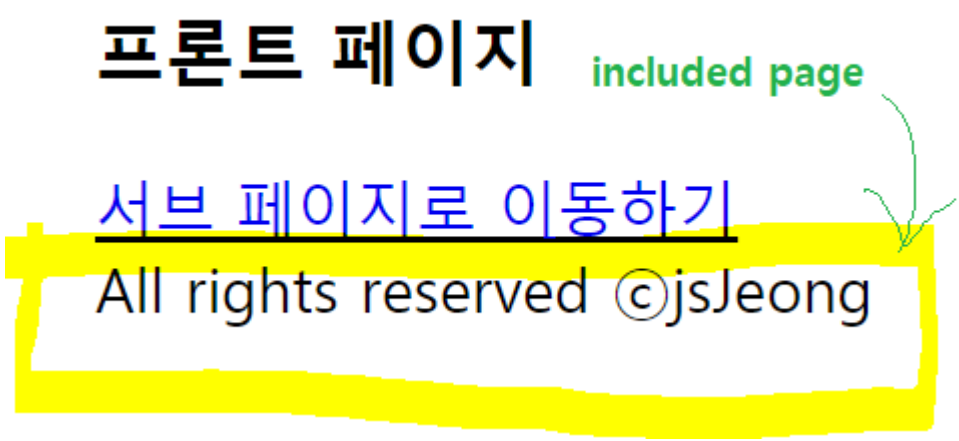
-메인페이지

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Front Page</title>
</head>
<body>
  <h3>프론트 페이지</h3>
  <a href="08_sub.jsp">서브 페이지로 이동하기</a>
  <%@ include file="footer.jsp" %>
</body>
</html>
```

-서브페이지

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

```
<html>
<head>
<meta charset="UTF-8">
<title>Sub Page</title>
</head>
<body>
  <h3>Sub Page</h3>
  <a href="08_main.jsp">프론트 페이지로 돌아가기</a>
  <%@ include file="footer.jsp" %>
</body>
</html>
```



그러면 우선, 메인페이지를 실행시키면 아래와 같이, footer 내용이 들어가 있는것을 확인해 볼 수 있다!

(3) taglib 지시자

- 확장된 기능을 사용하기 위해 jsp 문서 내에서 내가 지정한 태그명으로 태그를 사용할 수 있도록 함!

(사용방법)

💡 <%@ taglib uri="uri" prefix="value"%>

- uri: 확장된 기능을 지원할 uri
- prefix: jsp 태그 내에서 명명할 고유한 태그의 이름

[예시]-

https://www.guru99.com/jsp-actions.html#4