

JSP

(Java Server Page)



JSP (Java Server Page)

JSP란

동적인 웹 페이지를 java언어를 이용하여 html, xml기반으로 작성할 수 있는 기술을 말한다. ** mvc2패턴에서는 view로 활용
servlet은 수정시 재컴파일(서버 리부팅) 해야 하지만 jsp는 동적으로 컴파일하기 때문에 서버를 리붓할 필요없이 유연성하게 작업이 가능하다

JSP 특징



- I. JSP 파일이 변경되지 않는다면, 'jsp' 파일에 대한 컴파일은 다시 일어나지 않는다.
- II. JSP 파일이 변경될 때 마다, 컨테이너는 translation, compile, load, initialization 과정을 수행한다.
** 구 버전의 JSP 파일을 overwrite 할 경우 제대로 반영이 되지 않는 경우가 발생할 수 있다.
- III. JSP 파일의 배포 환경(위치)은 HTML과 동일하다
= WEB_ROOT 폴더 하단.





JSP (Java Server Page)

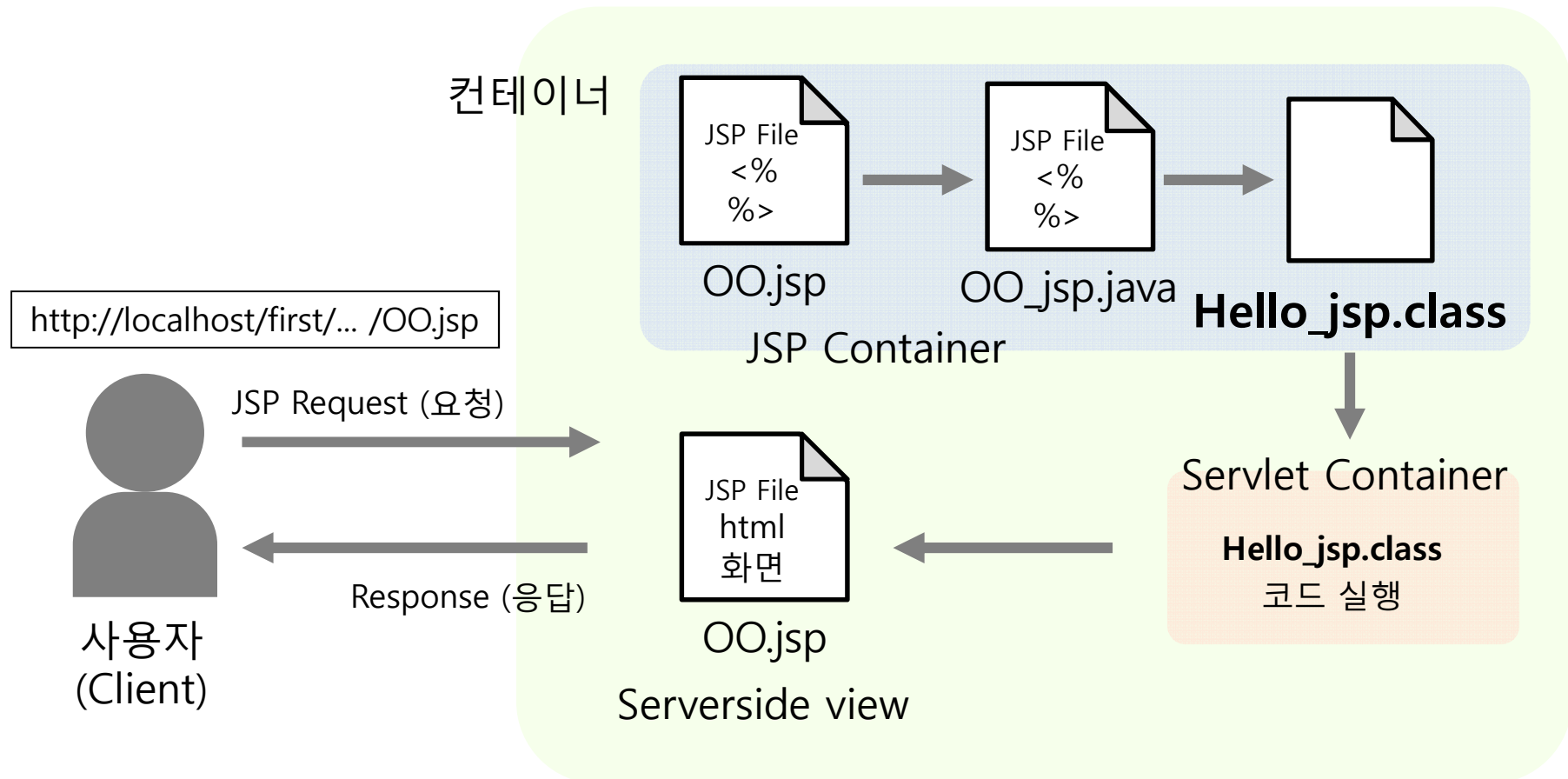
Servlet과 JSP 비교

구분	Servlet	JSP
형태	Java 코드에 HTML 코드를 삽입 	HTML 코드에 Java 코드를 삽입 
예시	<code>out.println("<HTML>");</code>	<code><% for (int i=0; i<10; i++) { %></code>
특징	Business 로직 처리에 적합	화면 로직 처리에 적합



JSP (Java Server Page)

JSP 실행 방식

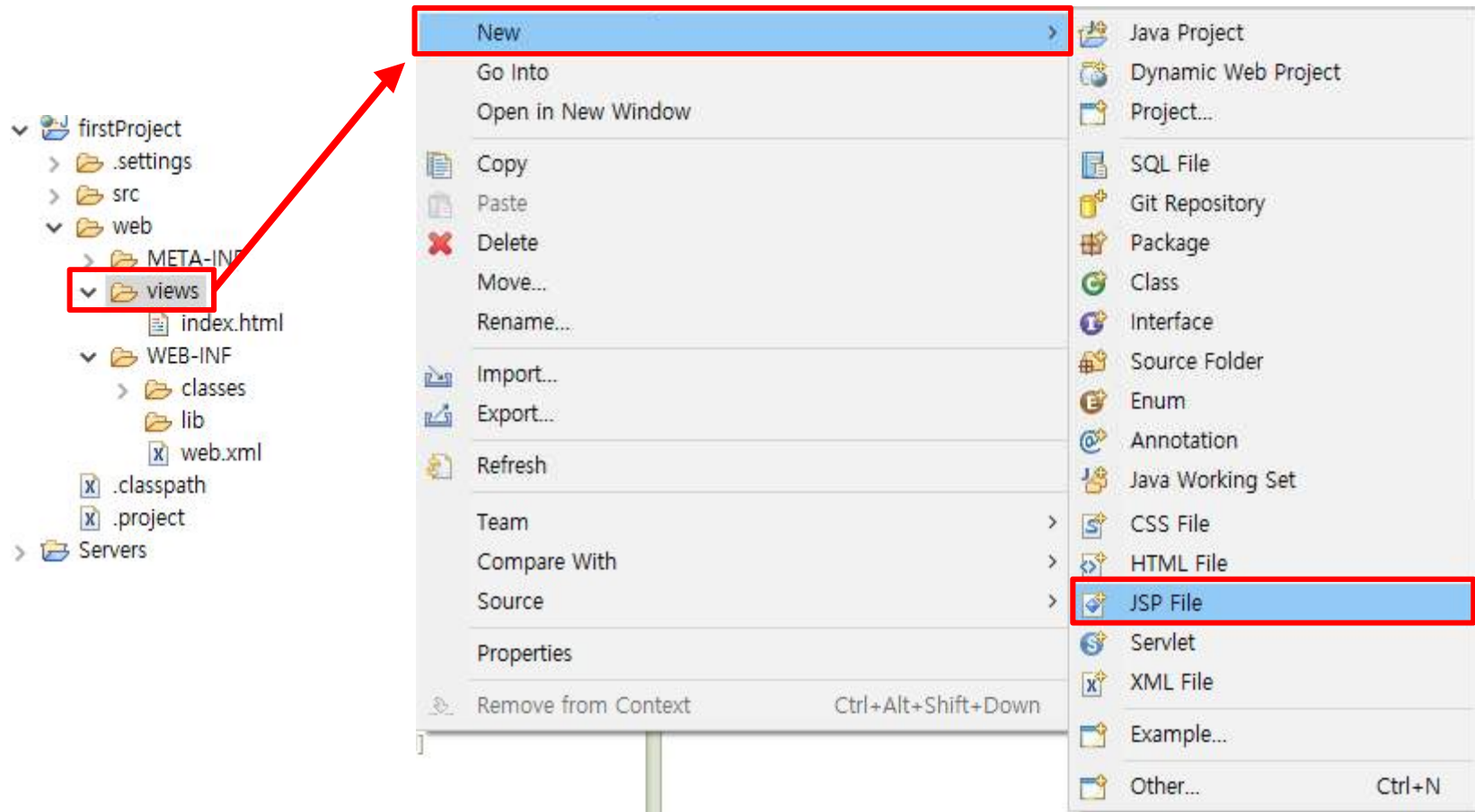


※ jsp변환 위치 : `₩workspace₩metadata₩plugins₩org.eclipse.wst.server.core₩tmp0₩work₩Catalina₩localhost₩프로젝트명₩org₩apache₩jsp₩`

JSP (Java Server Page)

JSP 만들기

웹 프로젝트 내 web/views 를 선택 후, 우클릭하여 JSP 파일 생성

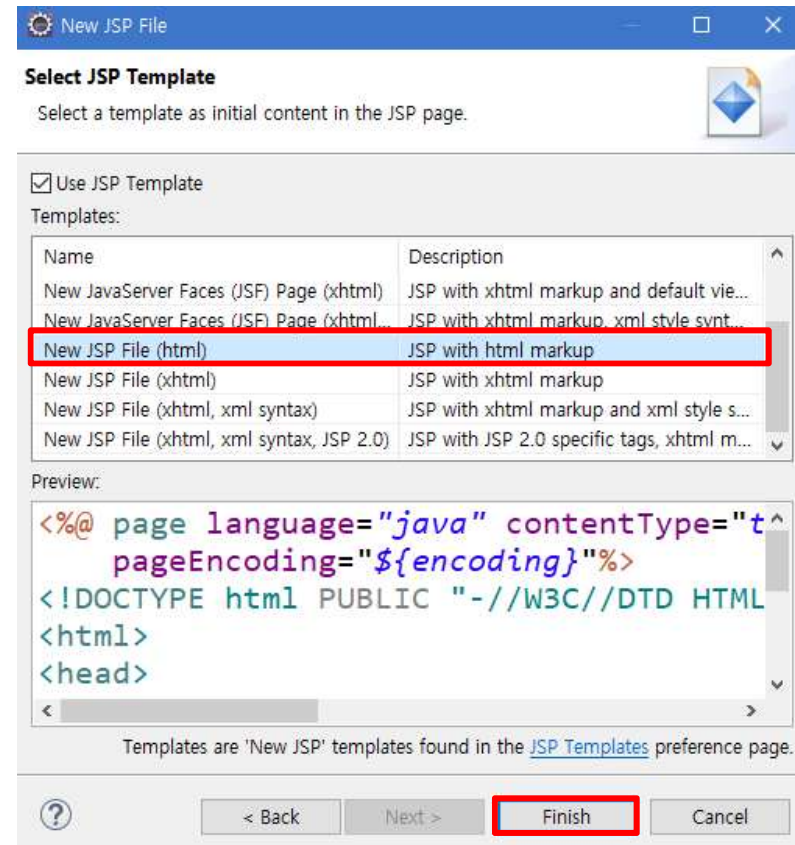
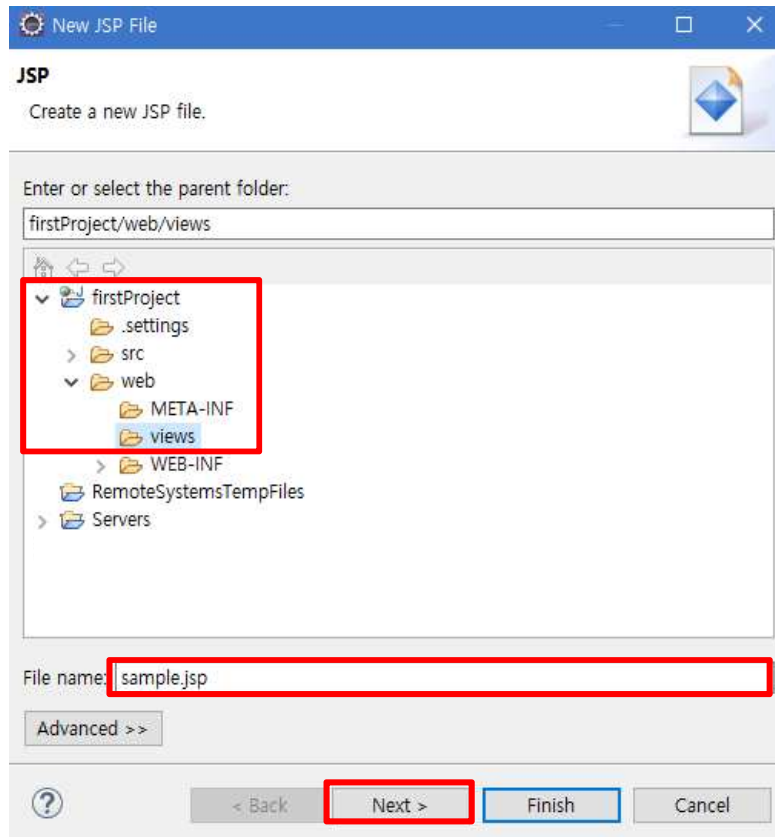




JSP (Java Server Page)

JSP 만들기

생성될 경로를 확인하고 Next, New JSP File 양식 선택 후 Finish





JSP (Java Server Page)

JSP 만들기

생성된 JSP 파일을 확인하고 서버를 실행하여 접근 확인

The screenshot shows an IDE with a project explorer on the left and a code editor on the right. The project explorer shows a project named 'firstProject' with a 'views' folder containing 'sample.jsp'. The code editor shows the content of 'sample.jsp' with line numbers 1 through 12. The code is as follows:

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http
4 <html>
5 <head>
6 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
7 <title>Insert title here</title>
8 </head>
9 <body>
10 <h1>안녕, JSP야!</h1>
11 </body>
12 </html>
```

Below the code editor, a browser address bar is shown with the URL 'localhost:8800/first/views/sample.jsp' highlighted in a red box.

안녕, JSP야!



JSP Elements



JSP Elements

JSP Element 표기법

주석문(Comments tag)	<code><%-- 내용 --%></code>
지시자(Directive tag)	<code><%@ 내용 %></code>
선언문(Declaration tag)	<code><%! 내용 %></code>
스크립트릿(Scriptlet tag)	<code><% Java코드 %></code>
표현식(Expression tag)	<code><%= 출력내용 %></code>





JSP Elements

JSP Element 표기법 예제

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

<%!
    // JSP 선언 태그입니다.
    public static final String NAME = "JSP World";
%>

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>JSP Elements 실습</title>
</head>
<body>
<%-- JSP 주석 태그 입니다. --%>
<%
    // JSP 스크립트 태그입니다.
    out.println("Welcome :");
    String name = request.getParameter("name");
    if(name == null || name.isEmpty()){
        name = NAME;
    }
%>

<!-- JSP 값 표현 태그입니다. -->
<h3>Hello, <%= name %></h3>
</body>
</html>
```

JSP 지시자 태그

JSP 선언 태그

JSP 주석 태그

JSP 스크립트 태그

JSP 값 표현 태그





주석태그

● HTML 주석

내부에서 `out.write();` 로 변환되나 화면에는 보이지 않는다.

```
<!-- HTML 주석입니다 -->  
→ out.write(" <!-- HTML 주석입니다 --> \r\n");
```

● JSP 주석 태그

JSP 파일 내에만 존재하고, Servlet 코드에는 포함되지 않는다.

```
<%-- JSP 주석 태그 입니다. --%>
```

● Java 주석 태그

변환된 Servlet 코드에는 포함되지만 HTTP 응답으로 전송되지 않는다.

```
<%-- //Java 주석입니다. --%>  
→ //Java 주석입니다.
```





지시자 태그

- JSP page 전체에 영향을 미치는 정보를 기술할 때 쓰인다.

```
<%@ 지시자 [속성명="value"] ... %>
```

- 지시자 종류

page : jsp페이지에 대한 설정 정보를 컨테이너에게 알려주는 지시자

include : 페이지내부에 다른 jsp페이지를 불러오는 지시자

* 해더와 풋터 같이 반복적으로 들어올 페이지에 자주 사용

taglib : 커스텀 태그 사용(tld파일이용), 주로 JSTL사용시 사용

예제

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

```
<%@ include file="header.html"%>
```

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





JSP Elements

● page 속성

속 성	값	설 명
info	"text"	페이지 설명하는 문자열 작성자 정보, 버전, 저작권 정보 등
language	"java"	사용할 프로그램 언어 지정
contentType		생성할 문서의 타입, 문자셋 지정(MIME)
import		페이지에서 다른 패키지 클래스 사용시 지정
errorpage		에러발생시 처리 페이지 지정
isErrorPage		해당페이지를 에러페이지로 지정
trimDirectiveWhitespaces	"true"/ "false"	html에서 jsp코드의 빈 곳을 없애주는 지능
pageEncoding	"문자코드"	출력버퍼가 다 채워졌을때 설정 true(비움) / false(에러)
session	"true"/ "false"	세션관리처리여부 true(default) / false()





JSP Elements

● page 속성

속 성	기본값	설 명
buffer	"none"/ "숫자kb"	출력 버퍼 설정
autoFlush	"true"/ "false"	출력버퍼가 다 채워졌을때 설정 true(비움) / false(에러)
isThreadSafe	"true"/ "false"	SingleThreadModel의 추가상속여부 결정 하 나의 jsp페이지가 동신에 여러 브라우저의 요청을 처리할 수 있는지 지정 true(default) / false(순차대로 처리)
extends		서블릿 클래스가 상속할 부모클래스 지정 * 잘 쓰지 않음 / 컨테이너가 알아서 설정
isELIgnored	"true"/ "false"	EL표현식의 사용여부를 결정 true(표현) / false(무시)





page 지시자 태그 사용법

- 여러 개의 page 구문을 사용할 수 있지만, import 속성을 제외하고는 한 페이지에 한 번씩만 선언할 수 있다.
- page 지시어는 JSP 파일의 어느 위치에 와도 상관 없으나, 가장 첫 부분에 사용하는 것이 좋다.

```
<%@ page import="java.io.*"%>  
<%@ page contentType="text/html" %>
```





import

- 변환될 서블릿 클래스에 필요한 자바 클래스의 import 문을 정의한다.
- java.lang, javax.servlet, javax.servlet.http, javax.servlet.jsp 는 기본적으로 import 되어 있다.
- 여러 package import시 ';' 기호를 이용하여 구분한다.

```
<%@ page import="java.io.*"%>  
<%@ page contentType="text/html" %>
```

contentType

- MIME 타입과 문자 인코딩을 설정한다.

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





isErrorPage

- 현재 페이지가 JSP 오류 처리용 페이지인지를 정의한다.
- 값은 true 또는 false(default)이다.
- true인 경우, exception 내장 객체를 사용할 수 있다.

```
<%@ page isErrorPage="true" %>
```

errorPage

- 해당 JSP 페이지가 발생시키는 모든 runtime exception을 처리할 다른 JSP페이지를 지정한다.
- 값은 상대적인 URL이다.

```
<%@ page errorPage="/error/errorForm.jsp" %>
```





JSP Exception 처리-보조

- JSP 페이지에서 발생하는 Exception을 처리하기 위해서는 별도의 예외 처리 페이지를 지정한다.
- 하나의 JSP 페이지에 대한 예외 처리 페이지는 하나만 지정할 수 있기 때문에 예외마다 다른 예외 처리는 불가능하다.

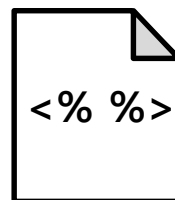
예외가 발생할 페이지

```
<%@page errorPage="/error/exceptionPage.jsp"%>
```

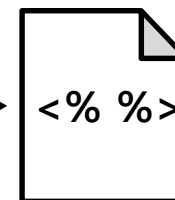
예외를 처리할 페이지

```
<%@page isErrorPage="true"%>
```

/throws_error.jsp



/error/exceptionPage.jsp



Web Container는 발생한 오류를 catch하여
예외 처리 페이지로 전달한다.





include 지시자 태그

- include 지시자 태그를 사용하면 다른 페이지(JSP, HTML)를 포함할 수 있다.

- 문법

```
<%@ include file="페이지 경로" %>
```

- 사용 예제

```
<%@ include file="footer.html" %>
```





선언태그

- Servlet 클래스의 멤버변수/메소드에 해당하는 코드를 작성할 때 사용된다.

멤버 변수 선언

```
<%! public static final String DEFAULT_NAME="홍길동"; %>  
<%! int counter = 0; %>
```

멤버 메소드 선언

```
<%!  
    public String getName(HttpServletRequest request) {  
        return request.getParameter("name");  
    }  
%>
```





Scriptlet tag

- `_jspService` 메소드의 로컬 변수와 코드를 작성할 때 사용된다.

로컬 변수 선언

```
<% int i = 0; %>
```

자바 코드 내용 기술

```
<% if ( i > 10 ) { %>
```

i가 10보다 큼니다.

```
<% } else { %>
```

i가 10보다 작습니다.

```
<% } %>
```





Scriptlet 변수 VS Declaration 변수

스크립트릿과 선언문 모두 변수를 선언할 수 있는데 이것은 차이는 지역변수와 전역변수의 차이이다. jsp에서 만들어지는 클래스에서 내부에서 사용할 수 있는지 클래스내부 메소드안에서만 사용가능한지 할지 이다.

즉 멤버변수(선언문), 지역변수(스크립트릿)의 차이가 된다.

** 스크립트릿으로 선언된 변수는 변환된 프로젝트명_jsp.java파일의 객체 _jspService메소드 내부에 생성





JSP Elements

표현태그

- Servlet 코드에서 `out.print()` 의 역할을 수행한다.

예제

현재 시간은 `<%= new java.util.Date() %>` 입니다.

※ 표현 태그에서는 `';` 을 붙이지 않는다

`<%= new java.util.Date(); %>`

Servlet
변환

`out.print(new java.util.Date(););` → **Syntax Error !!**



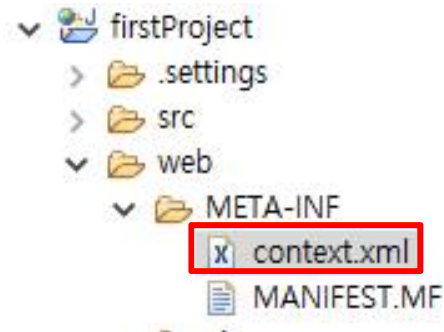


JSP Elements

JSP도 Servlet이다!

JSP도 컴파일 시 Servlet으로 변환되어 서비스 된다.

해당 과정을 직접 보기 위해 먼저 web/META-INF/ 경로에 context.xml 파일을 생성하여 다음과 같이 기록한다.



```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Licensed to the Apache Software Foundation (ASF) under one or more contrib
    ownership. The ASF licenses this file to You under the Apache License, Ver
    License at http://www.apache.org/licenses/LICENSE-2.0 Unless required by a
    WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the L
    --><!-- The contents of this file will be loaded for each web application -->
<Context workDir="D:\workspace\firstProject\web\WEB-INF\jspwork">

    <!-- Default set of monitored resources -->
    <WatchedResource>WEB-INF/web.xml</WatchedResource>

    <!-- Uncomment this to disable session persistence across Tomcat restarts -->
    <!--
    <Manager pathname="" />
    -->

    <!-- Uncomment this to enable Comet connection tacking (provides events
    on session expiration as well as webapp lifecycle) -->
    <!--
    <Valve className="org.apache.catalina.valves.CometConnectionManagerValve" />
    -->
    <Resource auth="Container" driverClassName="oracle.jdbc.OracleDriver" maxActiv
    url="jdbc:oracle:thin:@127.0.0.1:1521:xe" username="student" />
</Context>
```





JSP Elements

JSP도 Servlet이다!

context.xml은 웹 애플리케이션 서버에서 사용할 자원을 설정하는 파일이다. 이 중 Context workDir 속성은 컴파일 된 JSP class 파일이 위치할 경로를 가리킨다. 해당 경로는 자신의 프로젝트 내 WEB-INF 경로에 맞게 작성하자

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Licensed to the Apache Software Foundation (ASF) under one or more contrib
    ownership. The ASF licenses this file to You under the Apache License, Ver
    License at http://www.apache.org/licenses/LICENSE-2.0 Unless required by a
    WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the L
--><!-- The contents of this file will be loaded for each web application -->
<Context workDir="D:\workspace\firstProject\web\WEB-INF\jspwork">

    <!-- Default set of monitored resources -->
    <WatchedResource>WEB-INF/web.xml</WatchedResource>

    <!-- Uncomment this to disable session persistence across Tomcat restarts -->
    <!--
    <Manager pathname="" />
    -->

    <!-- Uncomment this to enable Comet connection tacking (provides events
    on session expiration as well as webapp lifecycle) -->
    <!--
    <Valve className="org.apache.catalina.valves.CometConnectionManagerValve" />
    -->
    <Resource auth="Container" driverClassName="oracle.jdbc.OracleDriver" maxActiv
    url="jdbc:oracle:thin:@127.0.0.1:1521:xe" username="student" />
</Context>
```

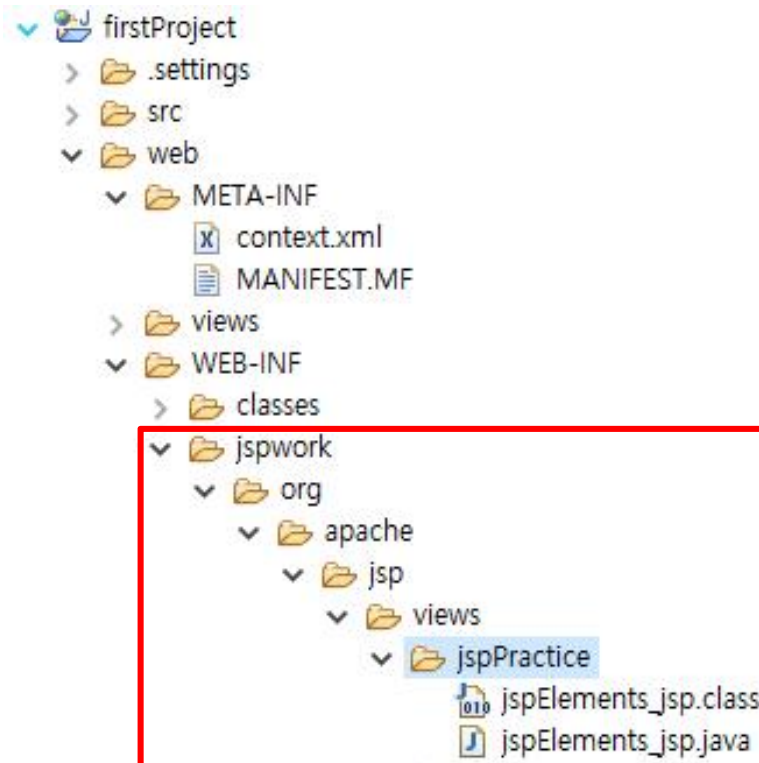




JSP Elements

JSP도 Servlet이다!

설정이 끝났다면 서버를 재실행하고 Project Explorer를 갱신하여 생성된 소스코드를 확인해보자.





JSP Elements

JSP Elements - Java 코드 변환 내용

jspElements.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" %>
<%!
    // JSP 선언 태그입니다.
    public static final String NAME = "JSP World";
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>JSP Elements 실습</title>
</head>
<body>
<%-- JSP 주석 태그입니다. --%>
<%
    // JSP 스크립트 태그입니다.
    out.println("Welcome :");
    String name = request.getParameter("name");
    if(name == null || name.isEmpty()){
        name = NAME;
    }
%>
<!-- JSP 값 표현 태그입니다. -->
<h3>Hello, <%= name %></h3>
</body>
</html>
```

jspElements_jsp.java

```
response.setContentType("text/html; charset=UTF-8");
pageContext = _jspxFactory.getPageContext(this, request, response,
    null, true, 8192, true);
_jspx_page_context = pageContext;
```

```
public static final String NAME = "JSP World";
```

```
out.println("Welcome :");
String name = request.getParameter("name");
if(name == null || name.isEmpty()){
    name = NAME;
}
```

```
out.write("<h3>Hello, ");
out.print( name );
out.write("</h3>\r\n");
```

※ JSP 주석 태그는 컴파일 시 포함되지 않는다.



JSP 내장 객체



JSP 내장 객체란?

JSP에서 기본적으로 제공하는 객체들로 request, response, out 등 Scriptlet tag와 Expression tag에서 사용할 수 있도록 암시적으로 선언된 객체를 뜻한다.

jspElements.jsp

```
<%  
    // JSP 스크립트릿 태그입니다.  
    out.println("Welcome :");  
    String name = request.getParameter("name");  
    if(name == null || name.isEmpty()){  
        name = NAME;  
    }  
%>
```





JSP 내장 객체의 종류

내장 객체 명	설명
request	HttpServletRequest 객체 참조 변수
response	HttpServletResponse 객체 참조 변수
out	JspWriter 객체 참조 변수
session	HttpSession 객체 참조 변수
application	ServletContext 객체 참조 변수
page	현재 JSP 페이지에 대한 참조 변수
exception	발생 하는 Throwable 객체에 대한 참조 변수

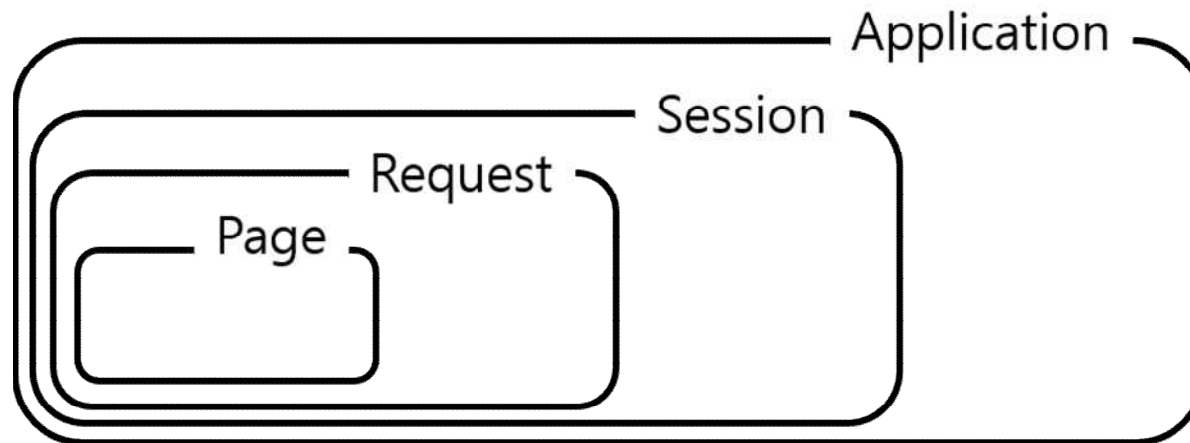




JSP 내장 객체

JSP 내장 객체의 영역

영역	설명
page	하나의 JSP페이지를 처리할 때 사용되는 영역
request	하나의 요청을 처리할 때 사용되는 영역
session	하나의 브라우저와 관련된 영역
application	하나의 웹 어플리케이션과 관련된 영역





Request 주요 메소드

메소드 명	설명
getParameter(name)	name 파라미터의 값을 리턴한다
getParameterValues(name)	name 파라미터의 값을 배열 형태로 리턴한다. (checkbox 등에 쓰임)
getParameterNames()	요청에 포함된 파라미터 이름들을 리턴한다.
getMethod()	현재 요청 방식을 리턴한다. (GET, POST)
getSession()	현재 세션 객체를 리턴한다.
setCharacterEncoding()	클라이언트에서 서버로 전달된 값을 지정한 문자셋으로 변경한다.





Response 주요 메소드

메소드 명	설명
sendRedirect(url)	응답 결과를 요청으로 하여 지정된 url에 재전송한다.
setStatus(int status_code)	응답으로 전송될 상태 코드를 설정한다. 성공일 경우 기본값은 '200', OK이다.
sendError(int status_code)	에러가 발생할 경우 응답 헤더에 상태 코드를 설정한다.
setContentType(String)	서버에서 클라이언트로 전달될 값의 데이터 타입을 설정한다.





HTTP Request 전송의 GET 방식과 POST 방식

HTTP 프로토콜을 통해 데이터를 전송할 때 보통 두 가지의 Request Method를 사용하는데 바로 GET 방식과 POST 방식이다.

GET 방식은 요청한 정보와 함께 전달되는 파라미터 값이 URL 내부에 쿼리 스트링(query string)으로 저장되어 보내진다.

POST 방식은 서버로 파라미터 값이 전달될 때, HTTP 메시지 바디 안에 query string이 저장되어 보내진다.





Query String이란?

사용자가 서버로 데이터를 전달할 때 전송된 데이터들을 URL의 뒷부분에 '?'로 구분 지어 전송하는 것을 말한다.

key 는 input 태그의 name 값을, value는 input 태그의 value 값을 뜻한다.

'?' 는 Query String의 시작을 의미하며, '&' 는 각 데이터 간의 구분자를 뜻한다.

[표현 식]

`http://localhost:8800/first/test.jsp?key=value`

[표현 예제]

`http://localhost:8800/first/test.jsp?id=sample&pwd=sample`

