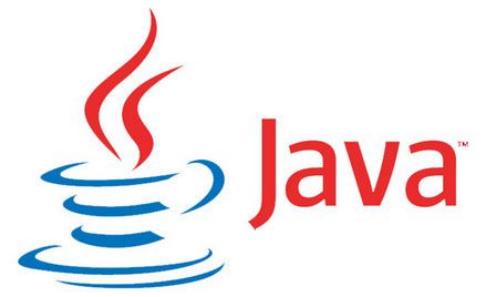


JSP 기초

- 우송 IT 교육원



환경설정

자바 소개

- 안드로이드 및 데스크톱 애플리케이션이나 웹사이트를 개발하는 핵심 언어
- 1995년 썬마이크로시스템즈(Sun Microsystems)에서 처음 발표
- 2010년 오라클에서 썬 인수, 자바 개발 도구(JDK) 배포해 기술 지원



자바 개발 도구(JDK) 설치

- JDK에는 Open JDK와 Oracle JDK 두 가지 있음

구분	Open JDK	Oracle JDK
라이선스	GNU GPL version 2	Oracle Technology Network License
사용료	무료	개발 및 학습용: 무료, 상업용: 유료
개발 소스 공개 의무	없음	없음

- Open JDK 다운로드: <https://jdk.java.net> 및 <https://adoptium.net>
- Oracle JDK 다운로드: <https://www.oracle.com/java/technologies/downloads/archive/>
- JDK LTS 버전 다운로드: <https://adoptium.net>

- 하위호환성이 좋다.
- OracleJDK(유료) 와 OpenJDK(무료)가 있다.
- OracleJDK에는 LTS버전이 있다.
- 전자정부표준프레임워크에 적용되는 버전은 자바8(3.7이상), 자바11(4.0)이다.
 - **자바8(LTS)** - 전자정부표준프레임워크 3.7 이상에서 사용
 - 자바9, 자바10
 - **자바11(LTS)** - 전자정부표준프레임워크 4.0 에서 최초 도입
 - 자바12, 자바13, 자바14, 자바15, 자바16
 - 자바17(LTS)
 - 자바18, 자바19

- 윈도우버전 jdk 11 다운로드 :
- 검색 keyword :
 - jdk 11.0.18 download --
<https://www.oracle.com/java/technologies/downloads/archive/>
 - JDK 8u202 download --- (1.8.202 이후 라이센스 변경으로 유료화)

압축 푼 후 디렉토리를 다음과 같이 만들어 둘 것

- jdk11, 1.8 경로

C:\Java**jdk-11.0.18** \bin
 \conf
 \include
 \jmods
 \legal
 \lib

C:\Java**jdk1.8.0_202** \bin
 \include
 \jre
 \lib
C:\Java**jre1.8.0_202** \bin
 \lib

자바환경변수 설정하기 (11.0.18 기준으로 사용)

1. JAVA_HOME

C:\Java\jdk-11.0.18

2. path

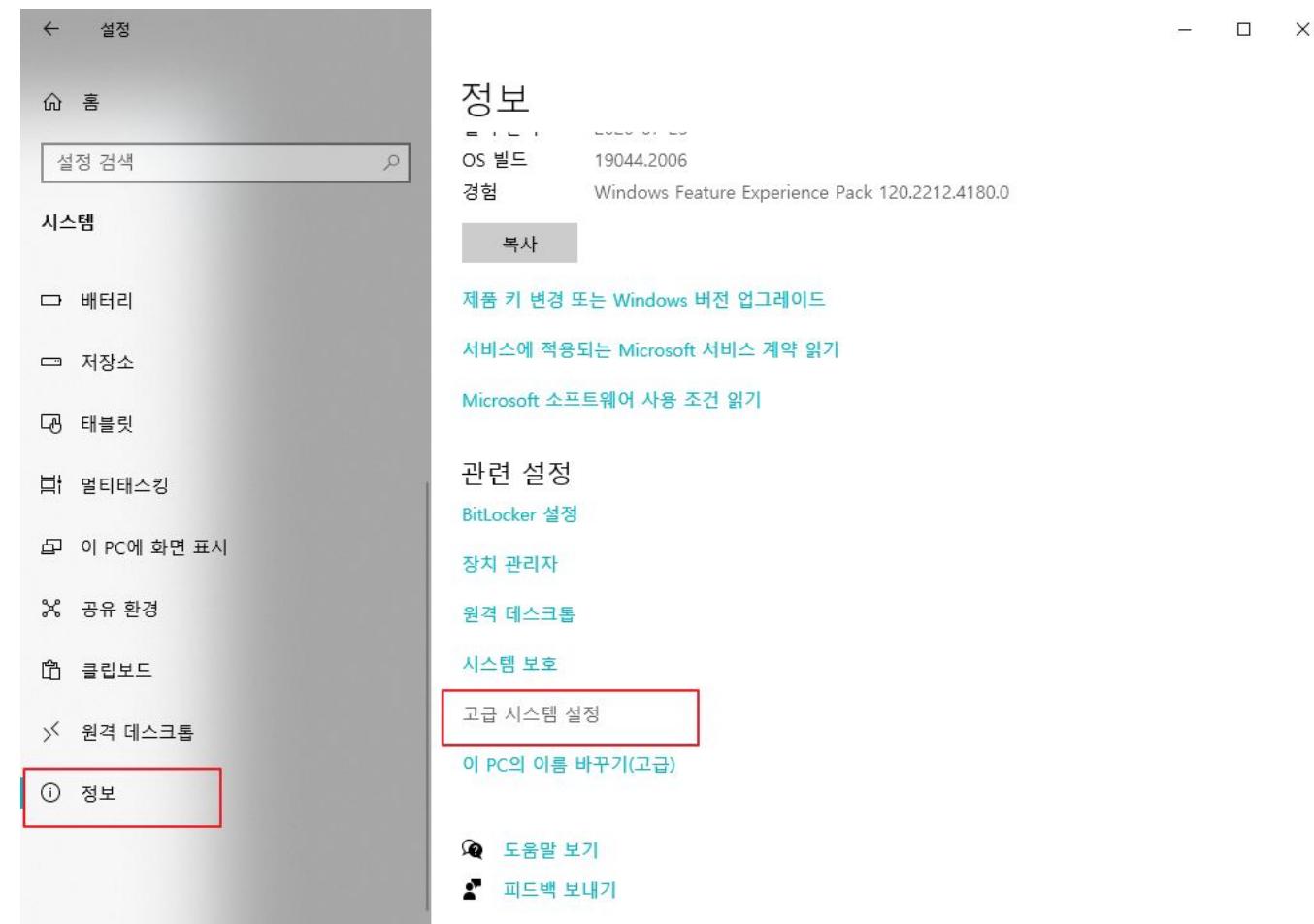
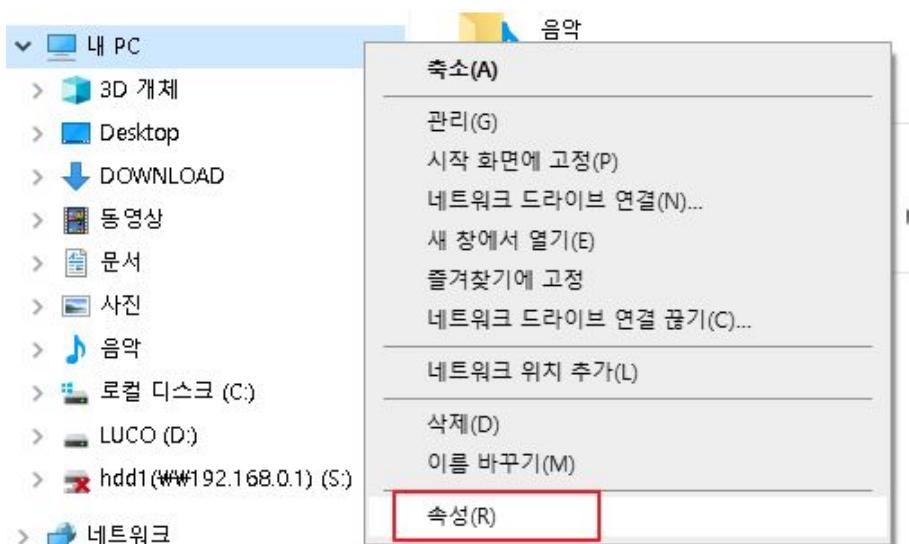
%JAVA_HOME%\bin

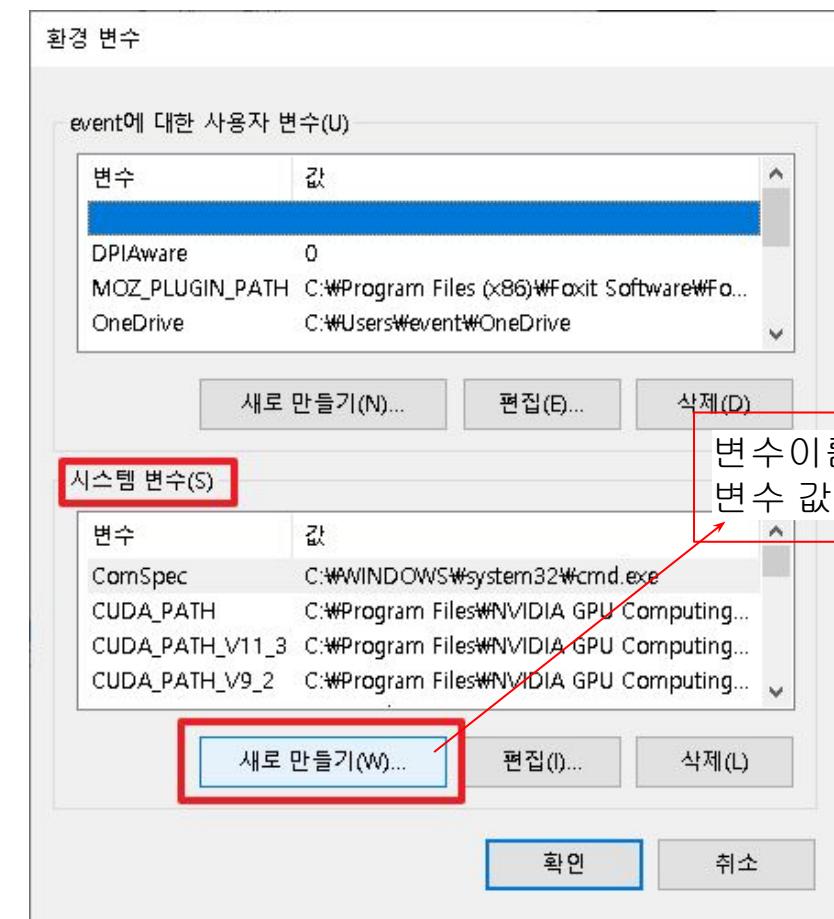
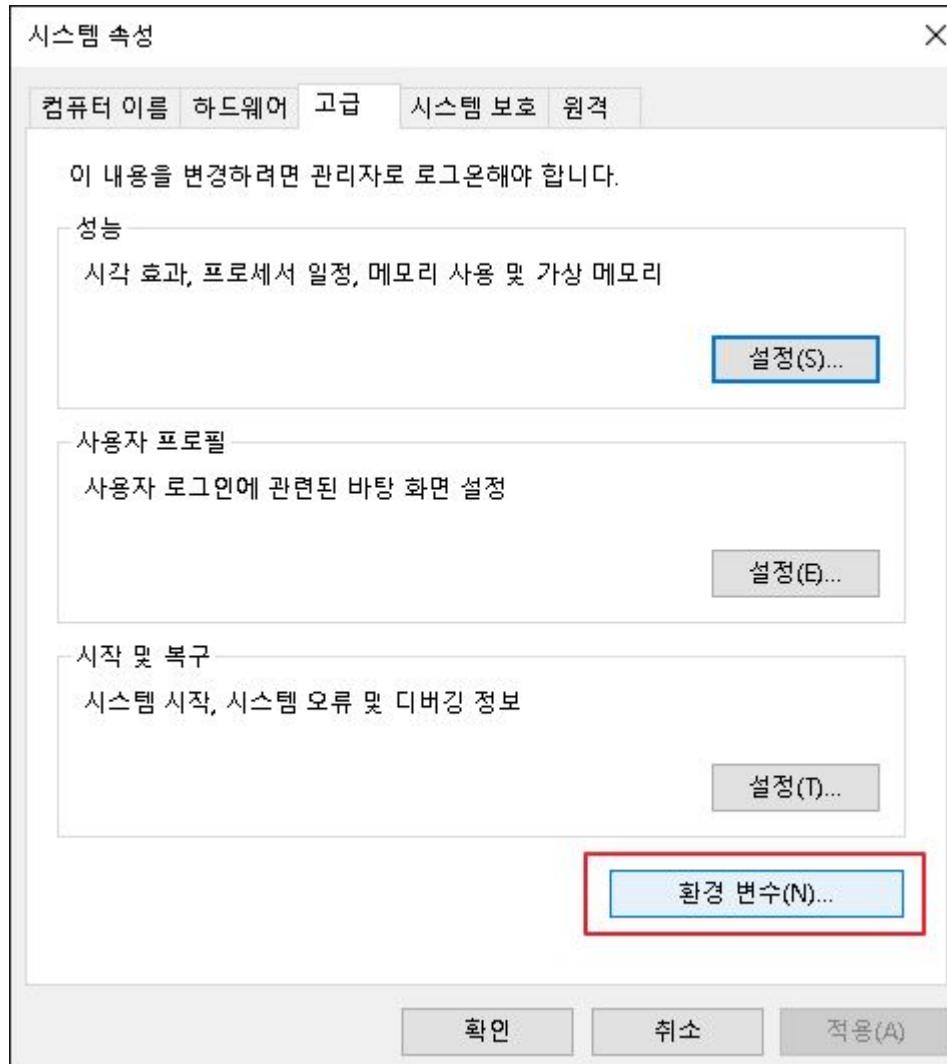
3. classpath (필요없음)

%JAVA_HOME%\lib

1. JAVA_HOME C:\Java\jdk-11.0.18
2. path %JAVA_HOME%\bin
3. classpath %JAVA_HOME%\lib

내PC -> 속성 -> 정보 -> 고급시스템설정

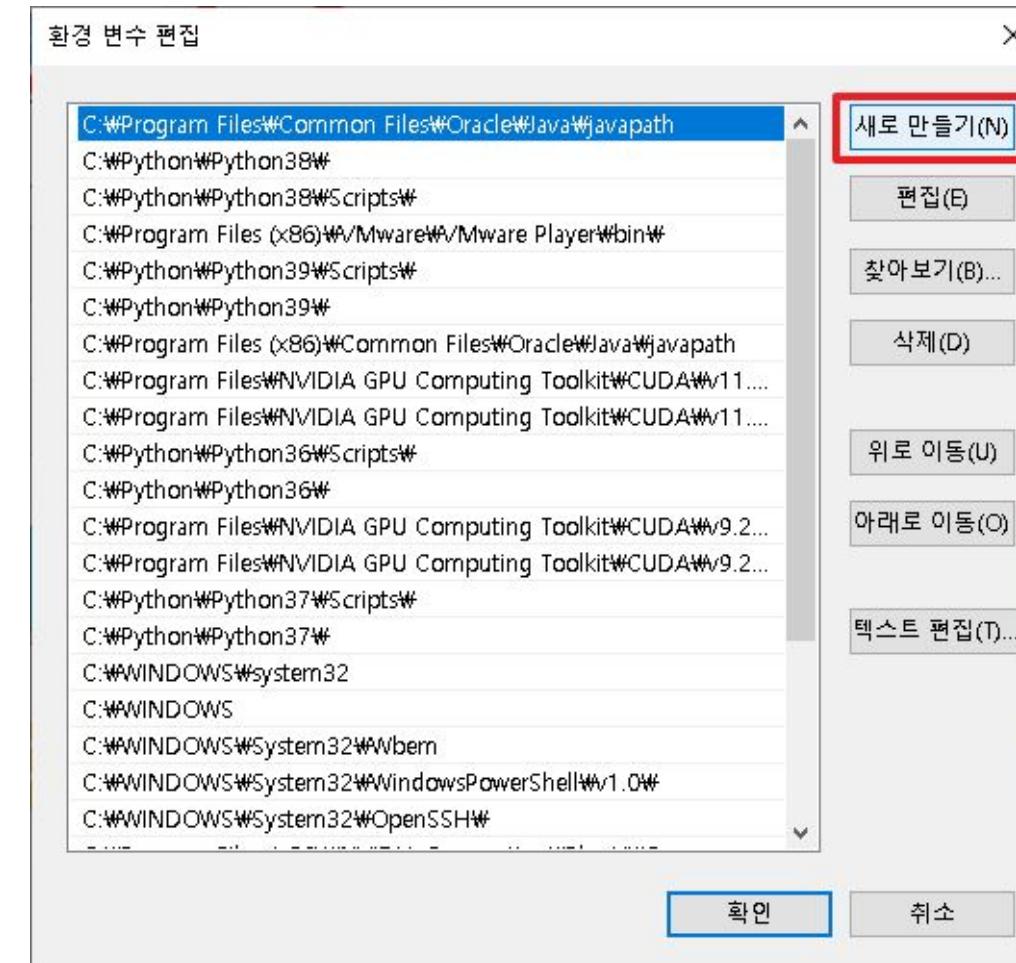
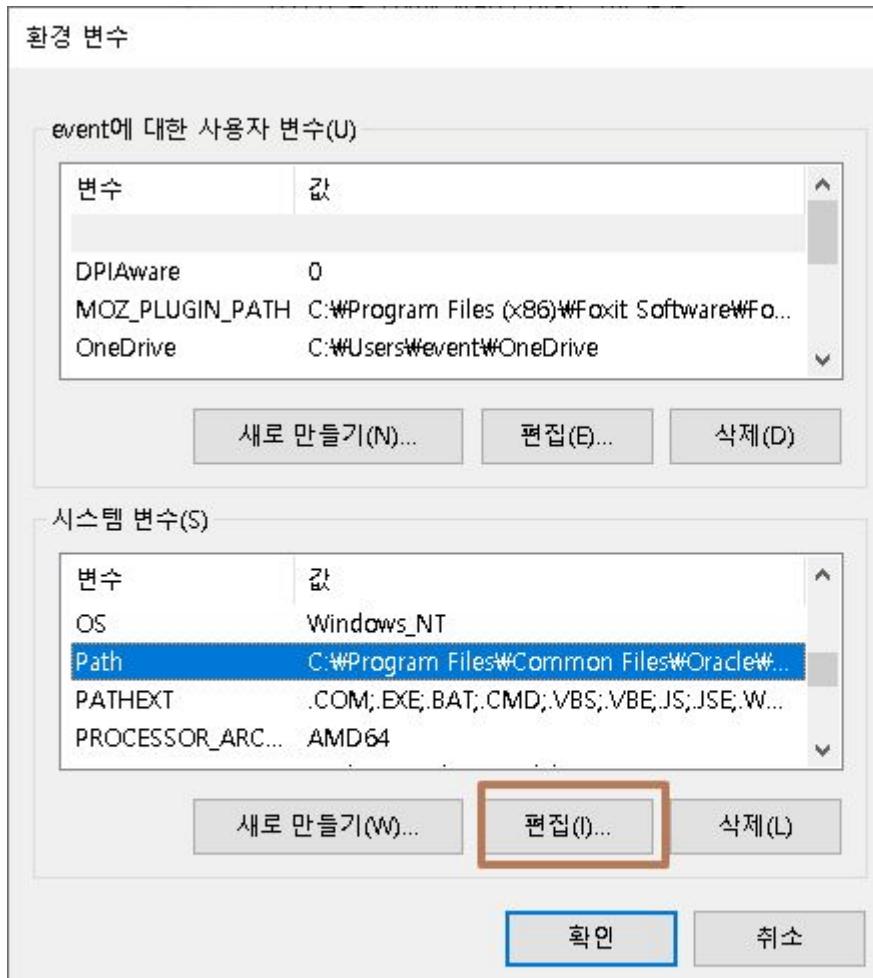




JAVA_HOME 이 없는 경우
-> 새로 만들기

자바 시작하기

환경변수 설정

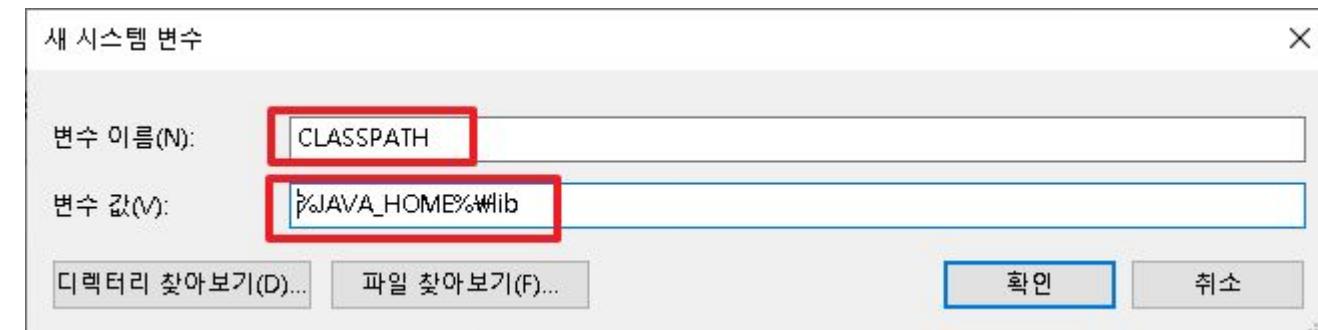


%JAVA_HOME%\bin



CLASSPATH

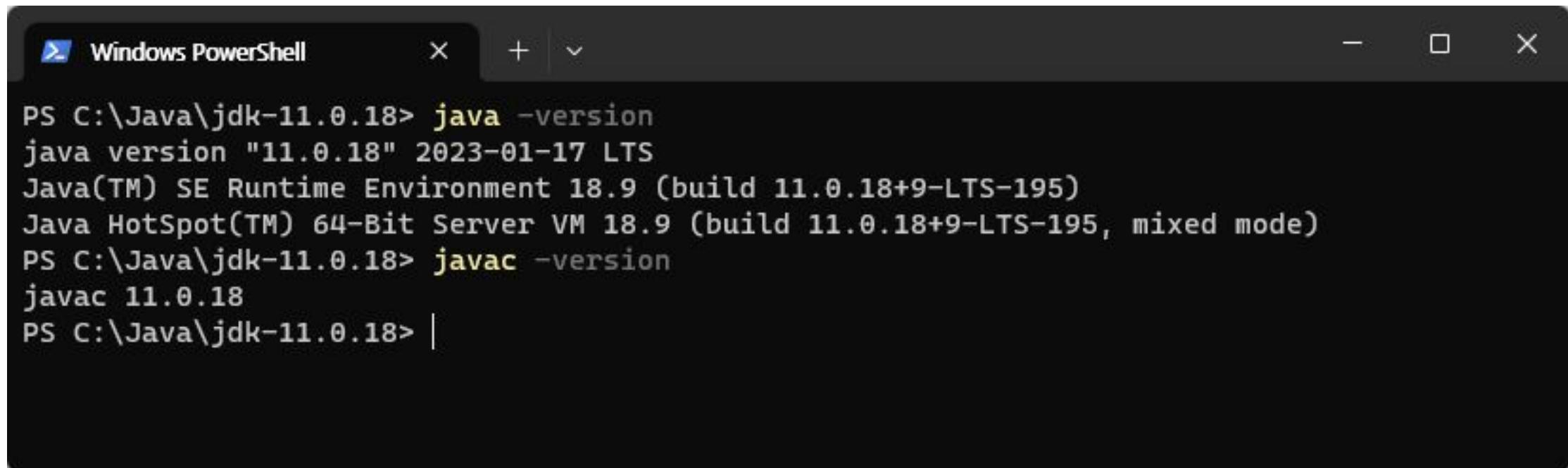
%JAVA_HOME%\lib



확인을 눌러 설치를 완료

명령프롬프트에서 java -version 입력하고 엔터 -> 제대로 설치되었는지 확인

명령프롬프트에서 javac -version 입력하고 엔터 -> 제대로 설치되었는지 확인



A screenshot of a Windows PowerShell window titled "Windows PowerShell". The window shows the command "java -version" being run in the directory "C:\Java\jdk-11.0.18", which returns the Java version information. Below it, the command "javac -version" is run, returning the Java compiler version. The window has standard operating system controls at the top.

```
PS C:\Java\jdk-11.0.18> java -version
java version "11.0.18" 2023-01-17 LTS
Java(TM) SE Runtime Environment 18.9 (build 11.0.18+9-LTS-195)
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.18+9-LTS-195, mixed mode)
PS C:\Java\jdk-11.0.18> javac -version
javac 11.0.18
PS C:\Java\jdk-11.0.18> |
```

STS3

“sts download”

- ❑ Spring Tool Suite 3.9.17
- ❑ Eclipse 4.19

The screenshot shows a news article from the Spring website. The title is "Spring Tools 4.11.0 released". The article is dated June 21, 2021, and has 7 comments. It announces the release of Spring Tools 4.11.0 for Eclipse, Visual Studio Code, and Theia. The article highlights major changes to the Eclipse-based distribution, including support for Java 16 and early-access builds for Apple Silicon (ARM M1). It also mentions that the distribution now ships with an embedded JDK16 runtime, so no specific IDE configuration is required. A sidebar on the right lists various Spring projects, with "Spring Tools 4" highlighted by a red box.

Spring Tools 4.11.0 released

RELEASES | MARTIN LIPPERT | JUNE 21, 2021 | 7 COMMENTS

Dear Spring Community,

I am happy to announce the 4.11.0 release of the Spring Tools 4 for Eclipse, Visual Studio Code, and Theia.

major changes to the Spring Tools 4 for Eclipse distribution

- updated to Eclipse 2021-06 release (including support for Java 16) ([new](#) and noteworthy)
- early-access builds for Apple Silicon platform (ARM M1) available

reminder

- the Eclipse-based distribution of the Spring Tools 4 requires a JDK11 (or newer) to run on
- the Eclipse-based distribution ships with an embedded JDK16 runtime, no need to install or configure a specific IDE on anymore

additional highlights

Overview
Spring Boot
Spring Framework
Spring Cloud
Spring Cloud Data Flow
Spring Data
Spring Integration
Spring Batch
Spring Security
View all projects

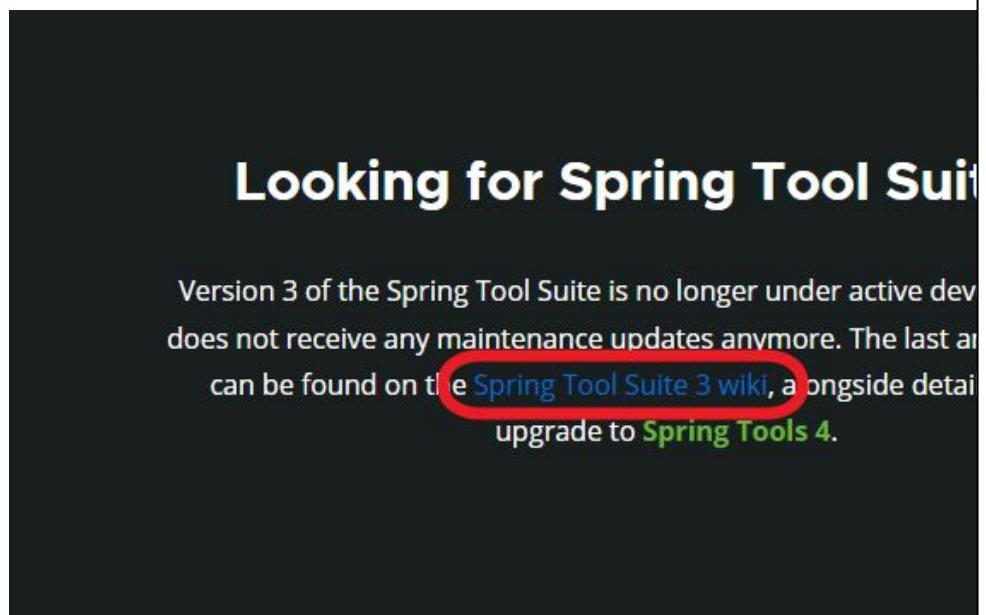
DEVELOPMENT TOOLS

Spring Tools 4

Spring Initializr

“sts download”

- ❑ Spring Tool Suite **3.9.17**
- ❑ Eclipse **4.19**



Previous STS3 Versions

Spring Tool Suite 3.9.17 (New and Noteworthy)

full distribution on Eclipse 4.20

- 3.9.17**: https://download.springsource.com/release/STS/3.9.17.RELEASE/dist/e4.20/spring-tool-suite-3.9.17.RELEASE-e4.20.0-win32-x86_64.zip
- https://download.springsource.com/release/STS/3.9.17.RELEASE/dist/e4.20/spring-tool-suite-3.9.17.RELEASE-e4.20.0-macosx-cocoa-x86_64.dmg
 - https://download.springsource.com/release/STS/3.9.17.RELEASE/dist/e4.20/spring-tool-suite-3.9.17.RELEASE-e4.20.0-linux-gtk-x86_64.tar.gz

full distribution on Eclipse 4.19

- https://download.springsource.com/release/STS/3.9.17.RELEASE/dist/e4.19/spring-tool-suite-3.9.17.RELEASE-e4.19.0-win32-x86_64.zip
- https://download.springsource.com/release/STS/3.9.17.RELEASE/dist/e4.19/spring-tool-suite-3.9.17.RELEASE-e4.19.0-macosx-cocoa-x86_64.dmg
- https://download.springsource.com/release/STS/3.9.17.RELEASE/dist/e4.19/spring-tool-suite-3.9.17.RELEASE-e4.19.0-linux-gtk-x86_64.tar.gz

full distribution on Eclipse 4.18

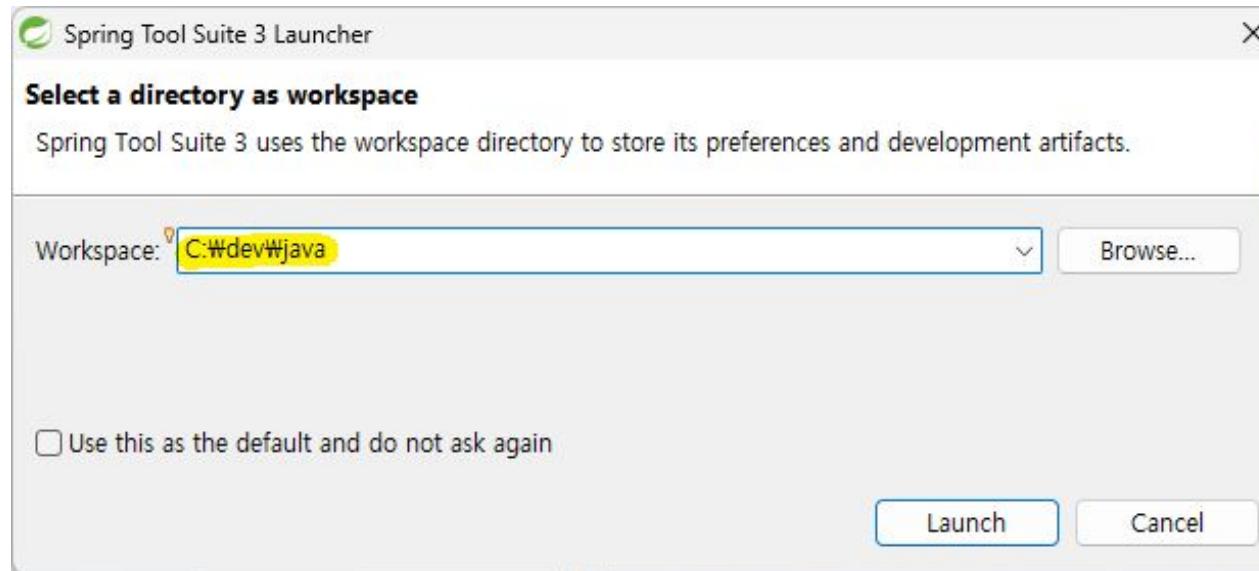
- https://download.springsource.com/release/STS/3.9.17.RELEASE/dist/e4.18/spring-tool-suite-3.9.17.RELEASE-e4.18.0-win32-x86_64.zip
- https://download.springsource.com/release/STS/3.9.17.RELEASE/dist/e4.18/spring-tool-suite-3.9.17.RELEASE-e4.18.0-macosx-cocoa-x86_64.dmg
- https://download.springsource.com/release/STS/3.9.17.RELEASE/dist/e4.18/spring-tool-suite-3.9.17.RELEASE-e4.18.0-linux-gtk-x86_64.tar.gz

■ 설치

C:\Java\sts-3.9.17.RELEASE\STS.exe 로 실행

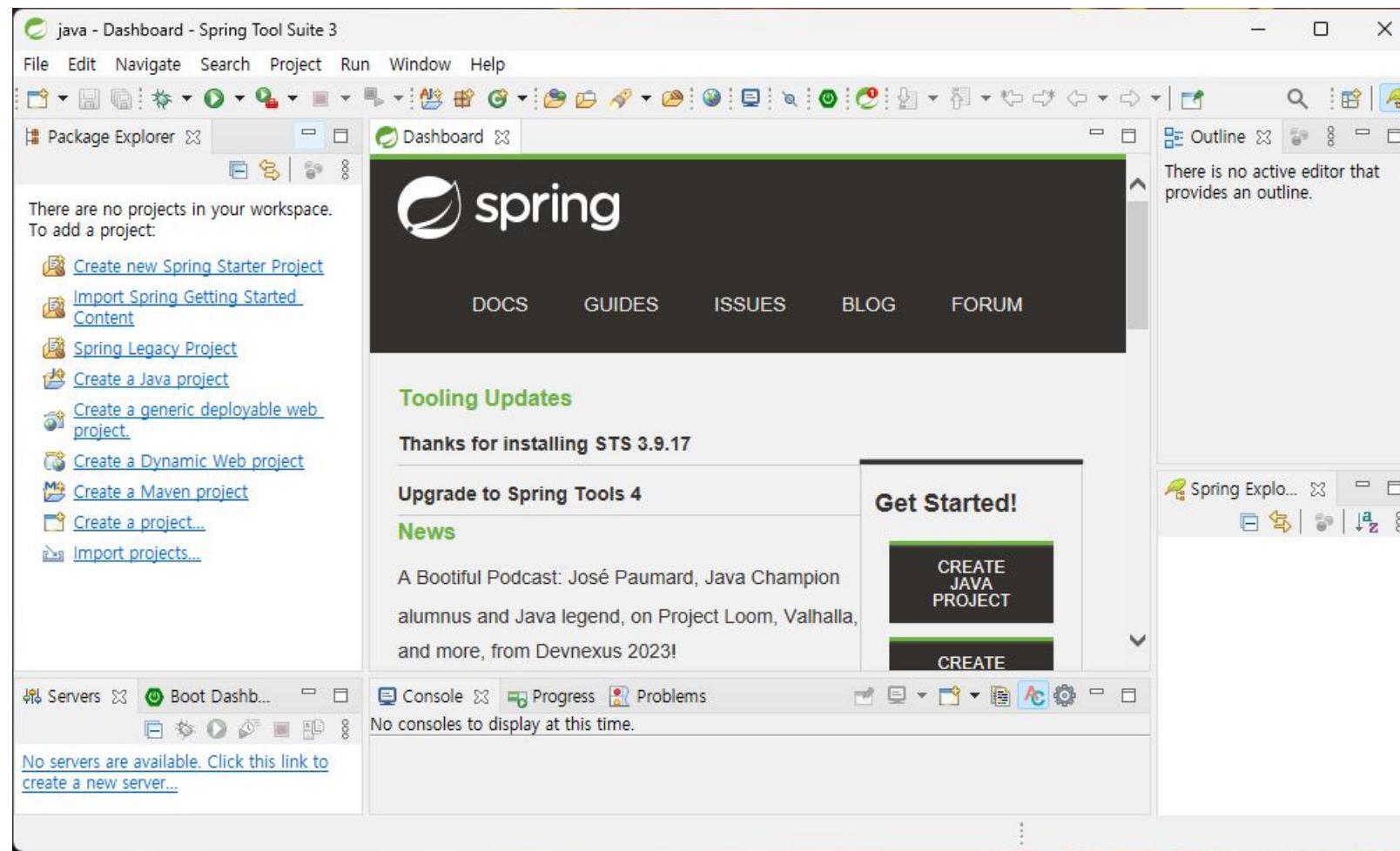
이클립스 최초 실행

- 앞으로 작성할 프로젝트의 소스를 저장할 폴더 위치(workspace)를 지정



- workspace 를 **C:\dev\java** 로 지정 (option)
- workspace 를 **C:\dev\jsp** 로 지정 (option)
- workspace 를 **C:\dev\spring** 로 지정 (option)

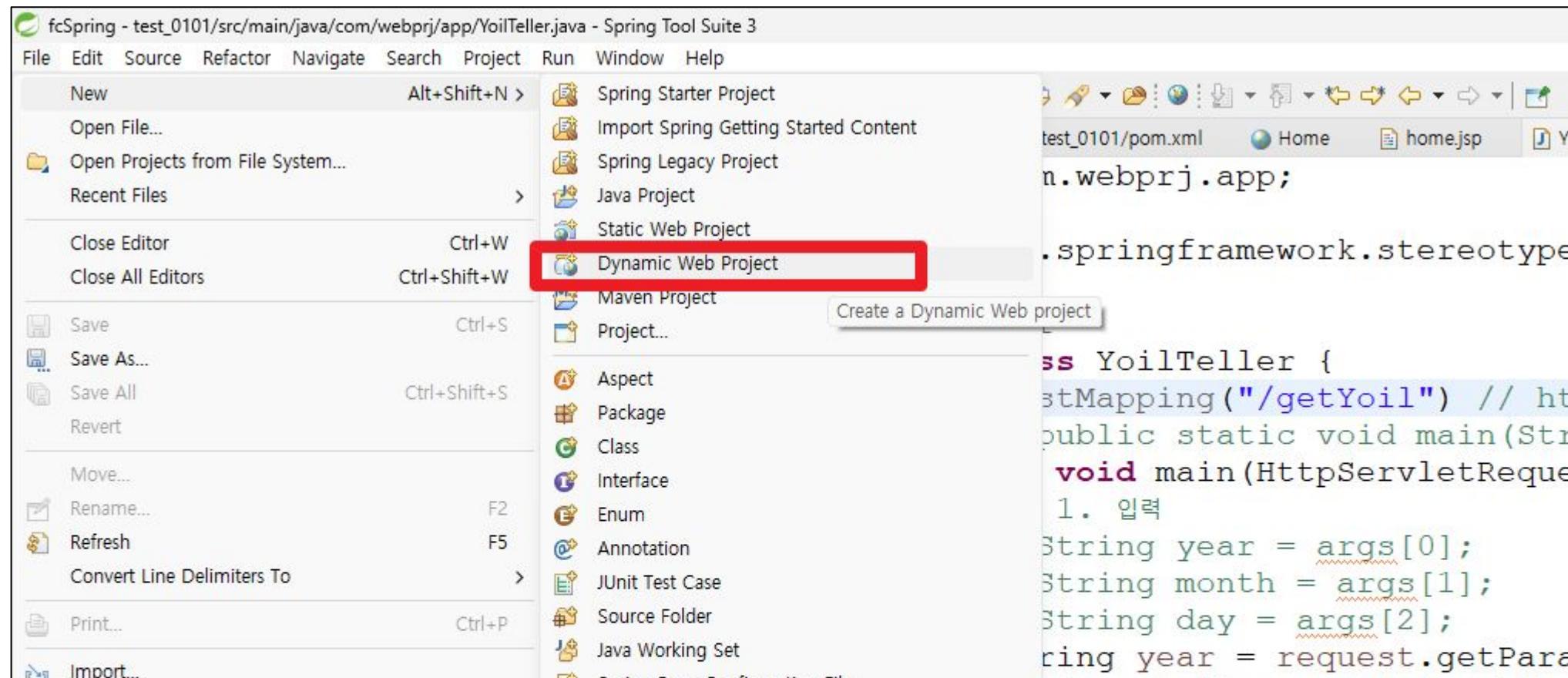
Welcome 창 닫기



준비

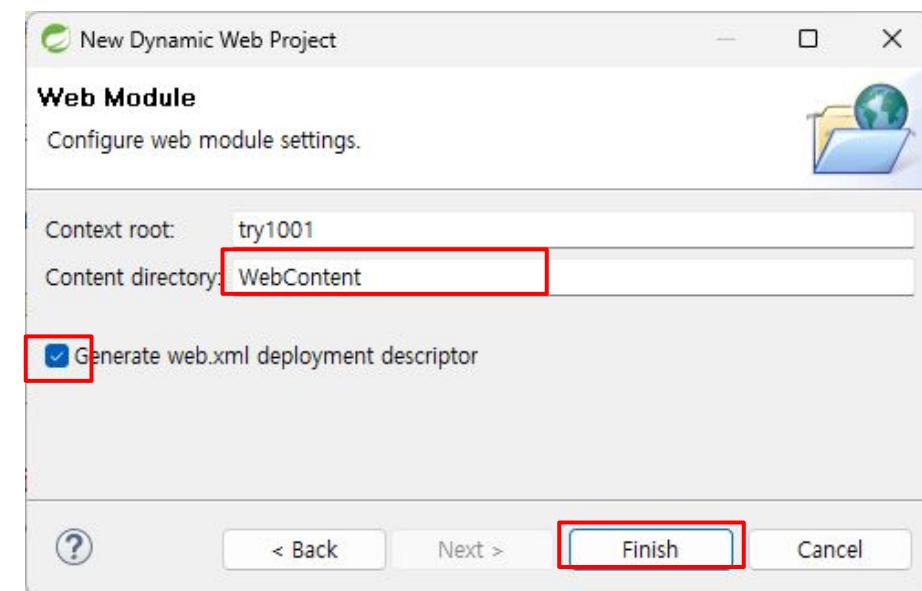
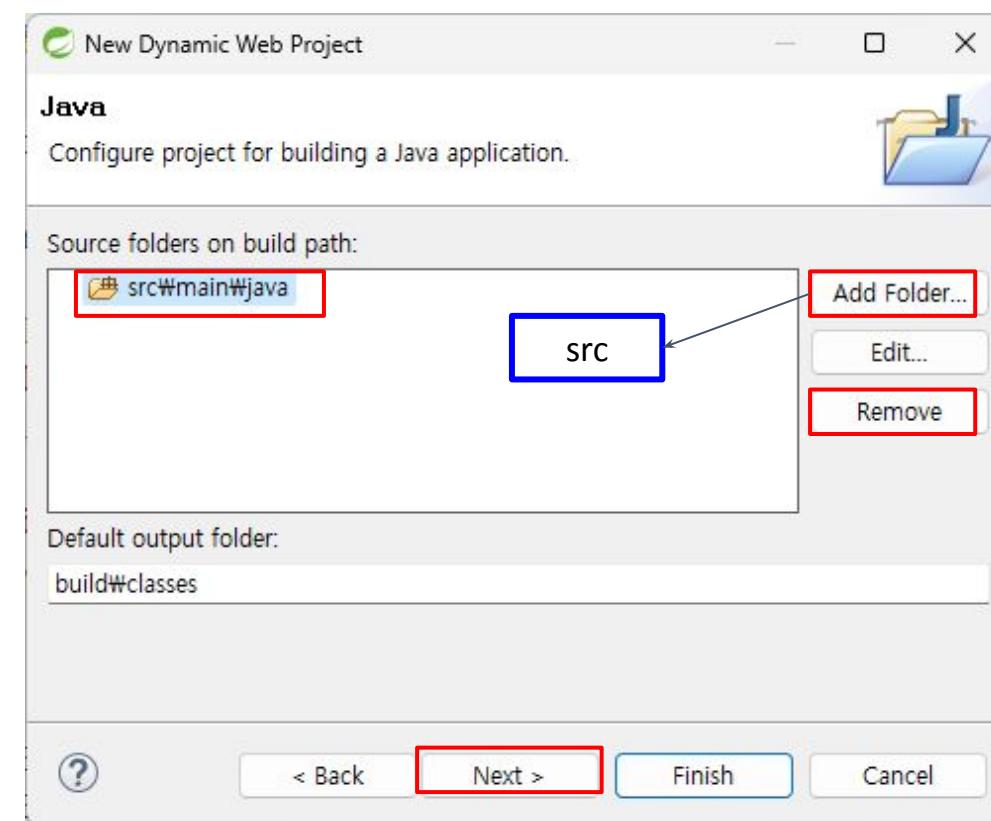
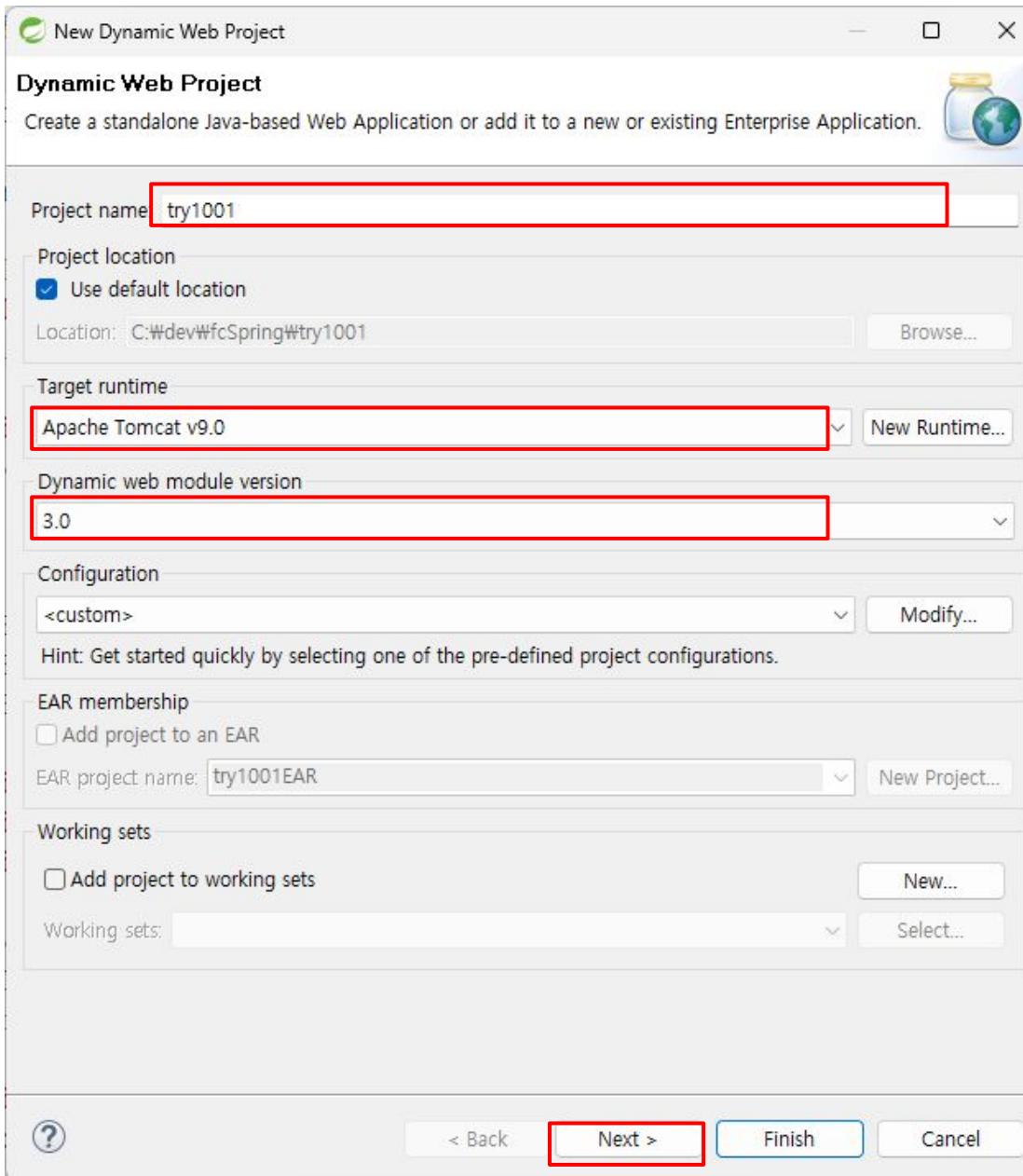
- Oracle Express 11g XE
 - <https://www.oracle.com/database/technologies/xe-prior-release-downloads.html>
- sqldeveloper
 - <http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html>
- MySQL 5.7 & Workbench
 - <https://www.mysql.com/downloads/>
 - <https://downloads.mysql.com/archives/installer/>

File > New > Dynamic Web Project 실행



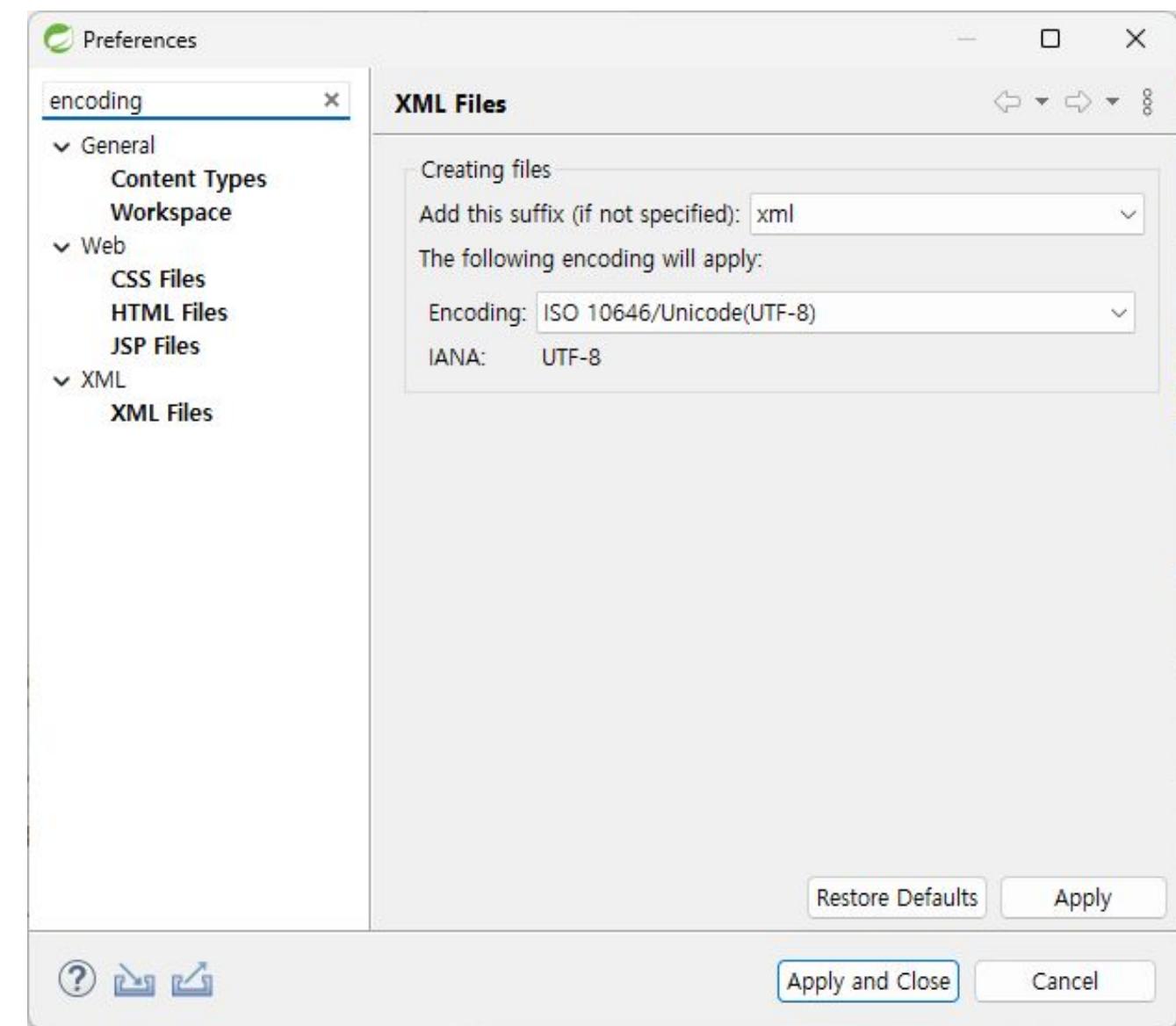
File > New > Dynamic Web Project 실행

- 프로젝트 이름
- Tomcat 9.0
- Dynamic Web Module 3.0
- source folder : **src**
- content directory : **WebContent**
- generate **web.xml**



이클립스 프로젝트 생성

- [Window]-[Preferences]에서 [General]-[Workspace]-<Text file encoding>을 ‘UTF-8’ 선택
- [Window]-[Preferences]에서 “encoding” 검색 설정



File > New > Dynamic Web Project 실행

- 프로젝트 이름 : hello_jsp
- Tomcat 9.0
- Dynamic Web Module 3.0
- source folder : **src**
- content directory : **WebContent**
- generate **web.xml**

File > New > Spring Legacy Project 실행

- Spring MVC Project

- 프로젝트 이름
- Tomcat 9.0
- Dynamic Web Module 3.0
- source folder : **src**
- content directory : **WebContent**
- generate **web.xml**



Spring Legacy Project

① Click 'Next' to load the template contents.

Project name: sp1001

Use default location

Location: C:\dev\fcSpring\sp1001



Select Spring version: Default

Browse...

Templates:

- ▼ Simple Projects
 - Simple Java
 - Simple Spring Maven
 - Simple Spring Web Maven
- > Batch
- > GemFire
- > Integration
- > Persistence
- Simple Spring Utility Project
- Spring MVC Project

↻ requires downloading

[Configure templates...](#) Refresh

Description:

A new Spring MVC web application development project

URL:<https://dist.springsource.com/release/STS/help/org.springframework.templates.mvc-3.2.2.zip>

Working sets

Add project to working sets

New...

Working sets:

Select...



< Back

Next >

Finish

Cancel



New Spring Legacy Project

Project Settings - Spring MVC Project

Define project specific settings. Required settings are denoted by "*".



Please specify the top-level package e.g. com.mycompany.myapp*

com.webprg.app



< Back

Next >

Finish

Cancel

1강. 웹프로그래밍

- 웹프로그래밍이란?
- JAVA웹
- 웹프로그램의 동작
- 필요한 학습

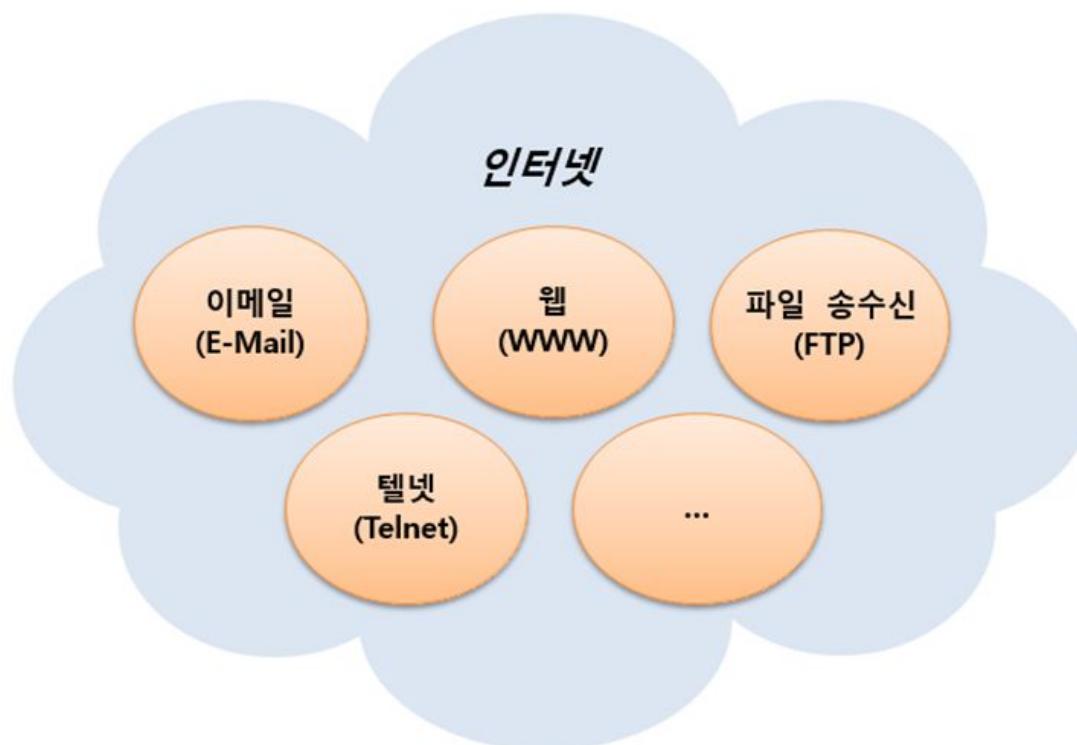
인터넷과 웹의 개요

인터넷

컴퓨터가 서로 연결되어 TCP/IP라는 통신 프로토콜을 이용하여 정보를 주고받는 전 세계의 컴퓨터 네트워크

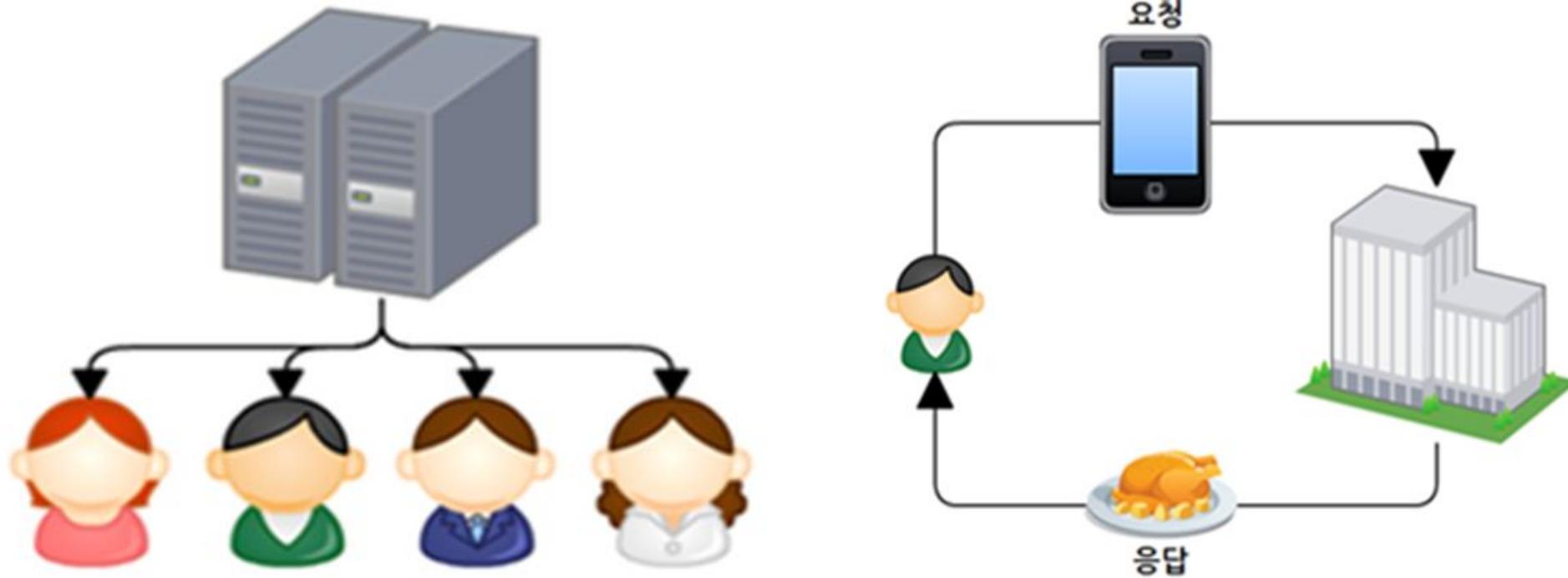
웹

인터넷에 연결된 컴퓨터들을 통해 사람들이 정보를 공유할 수 있는 정보 공간
월드 와이드 웹(world wide web)의 줄임말



웹의 동작 원리

웹은 기본적으로 클라이언트/서버 방식으로 동작



가장 널리 쓰이는 웹 서버

아파치(Apache)

톰캣(Tomcat)

IIS(Internet Information Server)

❖ 정적 웹 페이지와 동적 웹 페이지

▪ 정적 웹 페이지

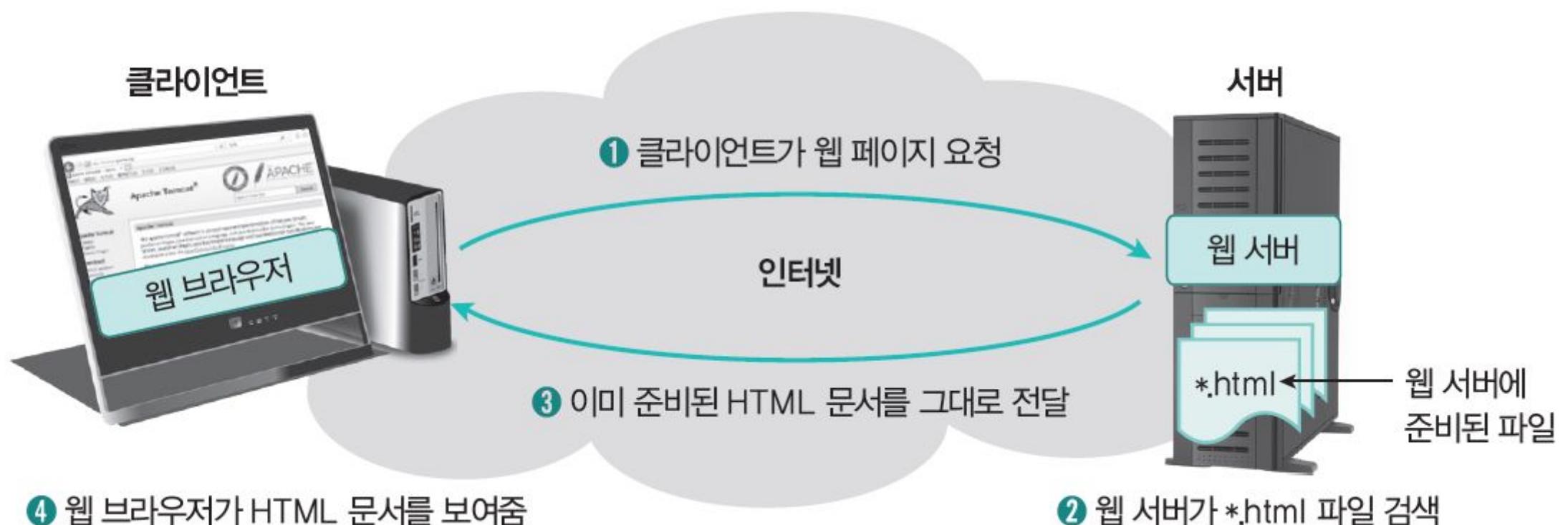
- 컴퓨터에 저장된 텍스트 파일을 그대로 보는 것
- HTML(HyperText Markup Language)

▪ 동적 웹 페이지

- 저장된 내용을 다른 변수로 가공 처리하여 보는 것
- PHP(Personal Home Page), ASP(Active Server Page), JSP

❖ 정적 웹 페이지의 동작 방식

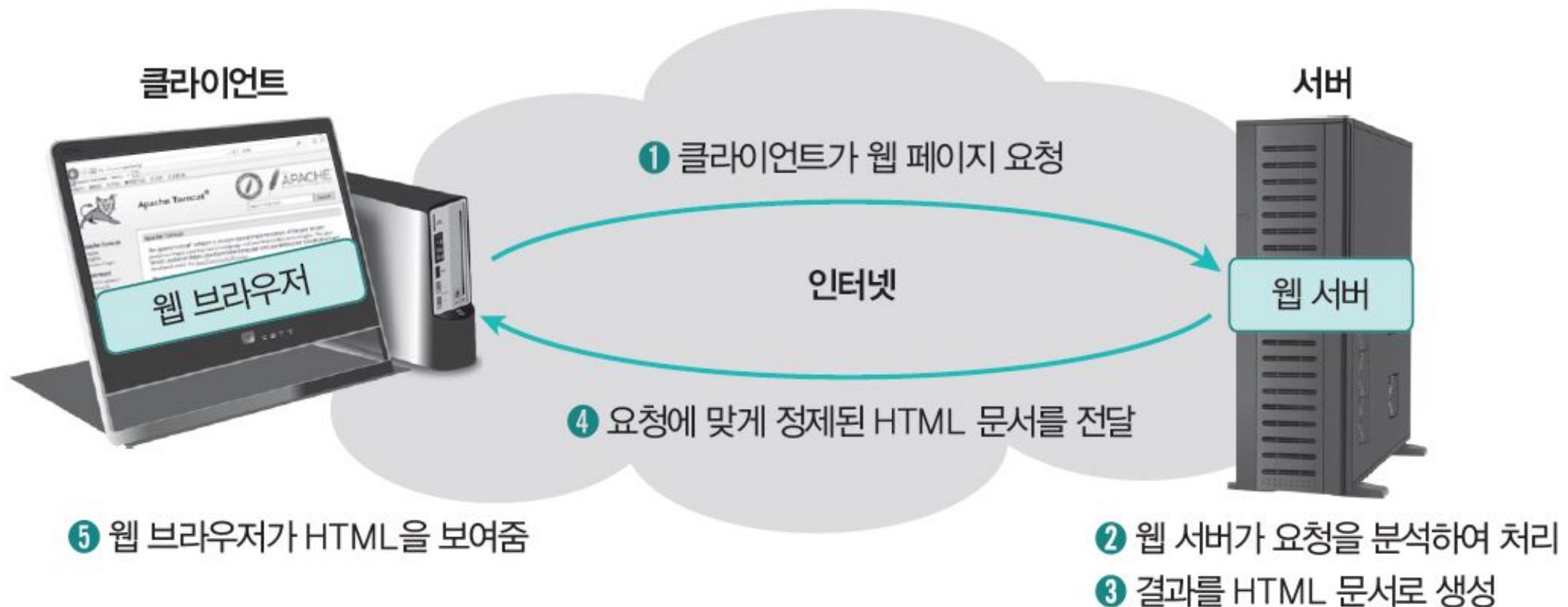
그림 1-4 정적 웹 페이지의 동작 방식



1. 웹과 JSP 프로그래밍 이해하기

❖ 동적 웹 페이지의 동작 방식

그림 1-5 동적 웹 페이지의 동작 방식



❖ 웹 프로그래밍과 JSP

▪ 웹 프로그래밍 언어

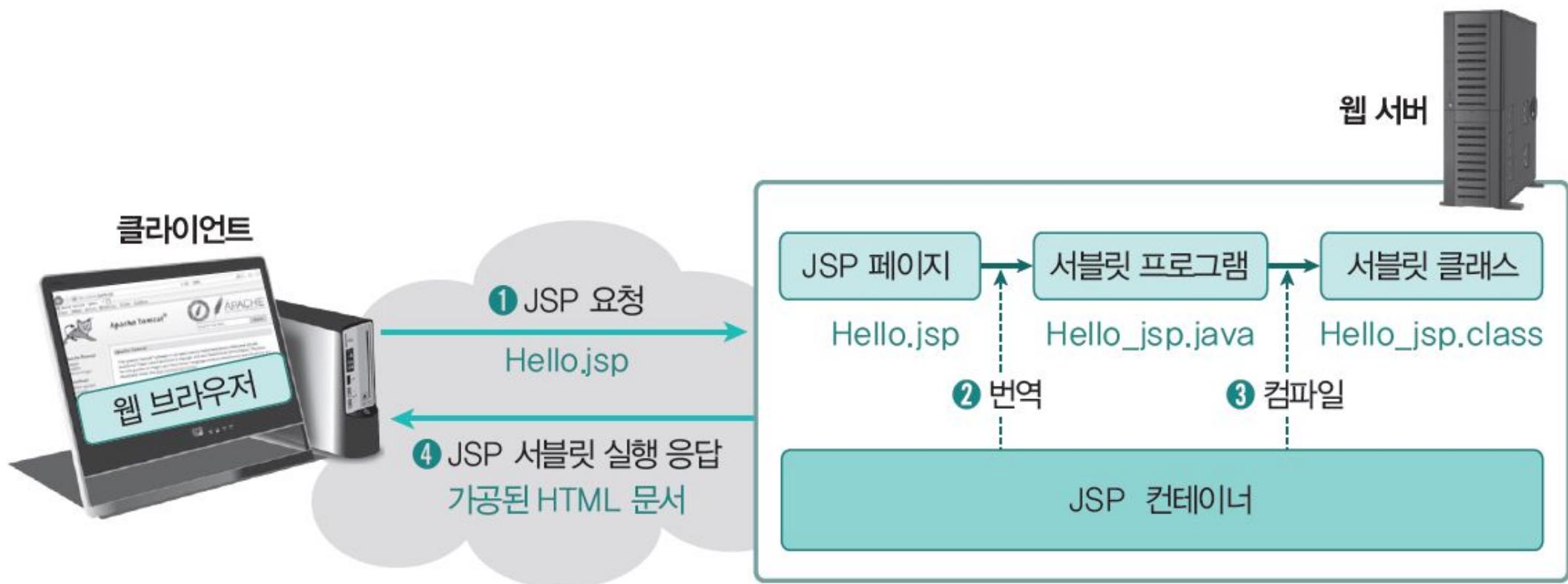
- 클라이언트 측 실행 언어와 서버 측 실행 언어로 구분
- 자바를 기반으로 하는 JSP는 서버 측 웹 프로그래밍 언어 중 하나

▪ JSP의 특징

- JSP는 서블릿 기술의 확장
- JSP는 유지 관리가 용이
- JSP는 빠른 개발이 가능
- JSP로 개발하면 코드 길이를 줄일 수 있음

❖ JSP 페이지의 처리 과정

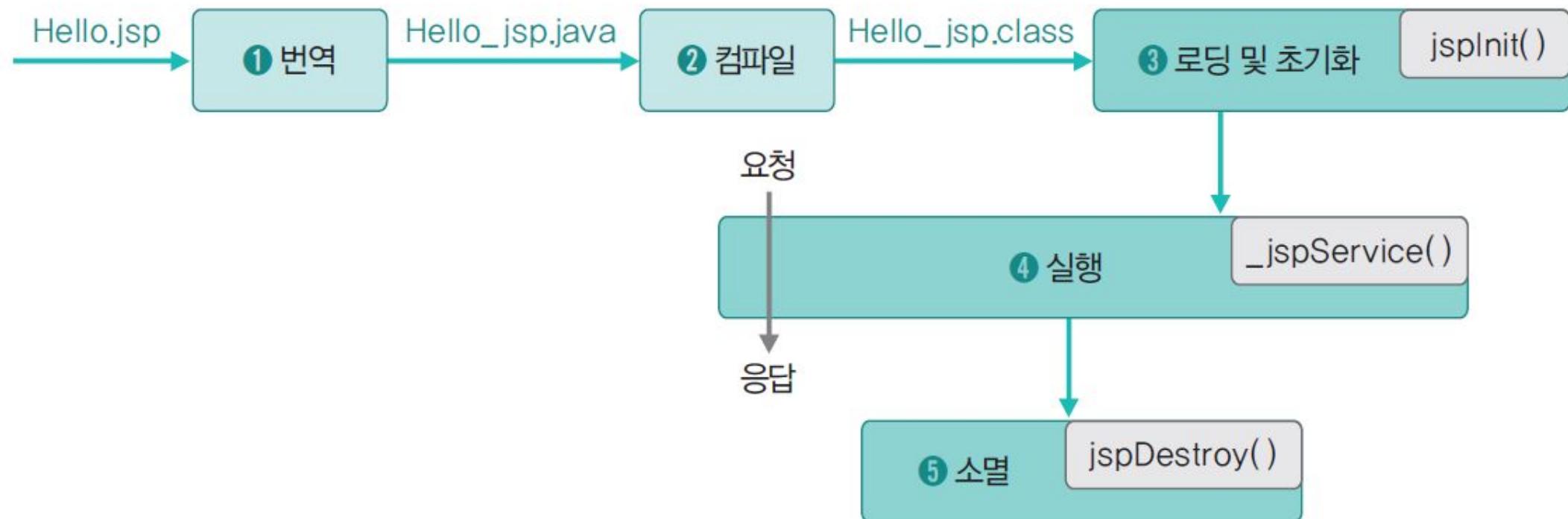
그림 1-6 JSP 페이지의 처리 과정



1. 웹과 JSP 프로그래밍 이해하기

❖ JSP 생명주기

그림 1-7 JSP 생명 주기



2. JSP 개발 환경 구축하기

❖ JSP 개발 환경 도구

- 자바 개발환경 : JDK
- 웹서버 : 톰캣
- 통합 개발 환경 : 이클립스

표 1-1 JSP 개발 환경 도구

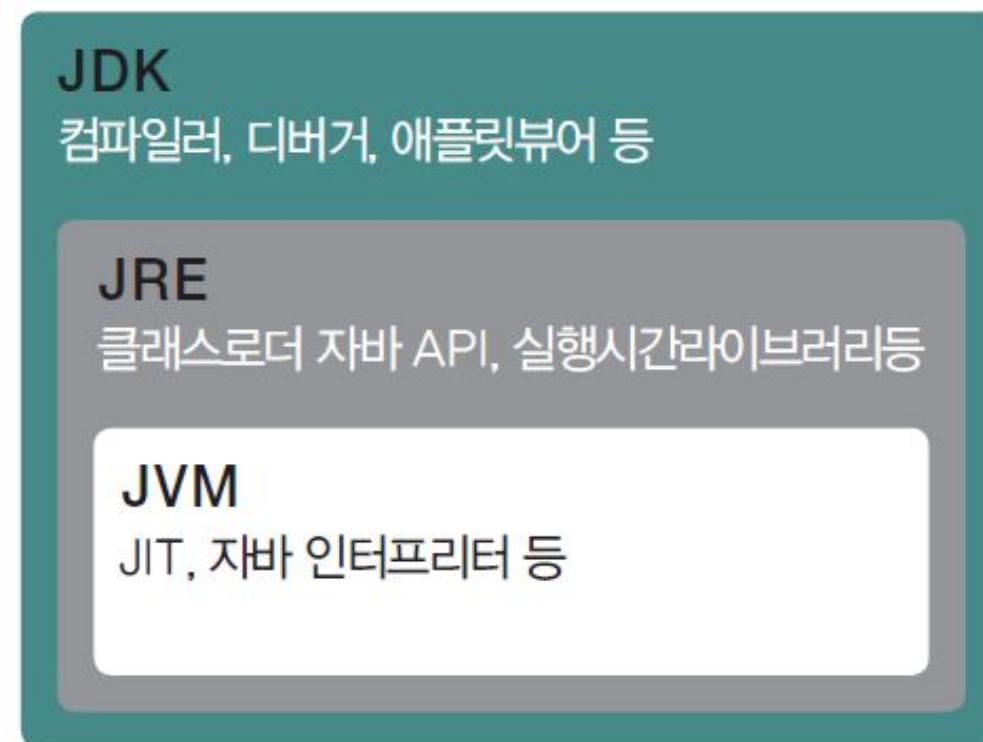
요소	프로그램명	설명
자바 개발 환경	JDK	JSP는 HTML 코드 내에 자바 코드를 작성하기 때문에 자바 개발 도구인 JDK가 반드시 설치되어 있어야 합니다.
웹 서버	톰캣	웹 프로그래밍 언어로 작성된 웹 페이지가 실행되어 웹 브라우저에 나타나도록 하기 위해 웹 컨테이너를 설치해야 합니다. JSP 웹 컨테이너로 자주 사용되는 것 중에서 가장 유명한 것은 톰캣입니다. 톰캣은 오픈소스 프로젝트로서 무료로 누구나 사용할 수 있습니다.
통합 개발 환경	이클립스	JSP 코드를 작성한 후 이를 컴파일하여 오류를 검사하고 실행 결과를 확인할 수 있는 통합 개발 환경(IDE)으로 개발자들에게 가장 인기 있는 이클립스(Eclipse)를 선택하여 설치합니다.

2. JSP 개발 환경 구축하기

❖ 자바 설치하고 환경 설정하기

- 자바 개발 키트
 - JDK(Java Development Kit)
- 자바 실행 환경
 - JRE(Java Runtime Environment)

그림 1-8 JDK와 JRE의 관계



2. JSP 개발 환경 구축하기

❖ 웹 서버와 통합 개발 환경 설치하기

■ 웹 서버

- 톰캣
 - 아파치 소프트웨어재단(Apache Software Foundation)에서 개발한 웹 애플리케이션 서버
 - 자바 만들어진 웹 페이지를 구동하기 위한 엔진

■ 통합 개발 환경

- 이클립스
 - 자바 통합 개발 환경(IDE) 중 가장 많이 사용되는 개발 도구
 - 자바를 기반으로 애플리케이션을 개발하기 위해 이클립스를 사용

2. JSP 개발 환경 구축하기

예제 1-3 톰캣 설치하기

1. 아파치 사이트에 접속 및 다운로드하기

- Http://tomcat.apache.org

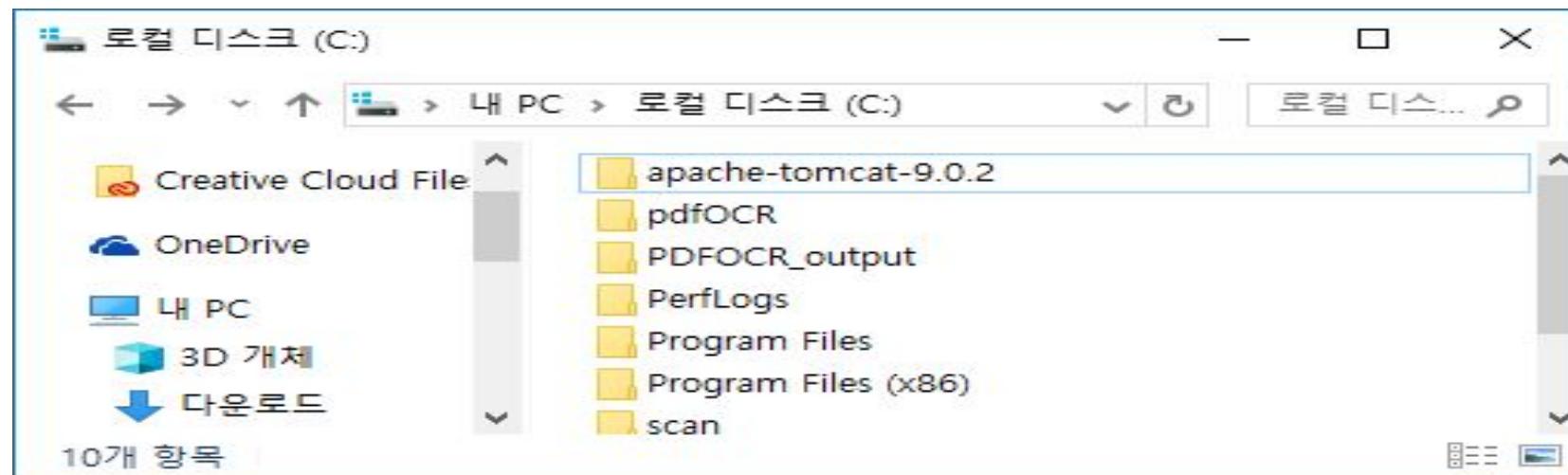
The screenshot shows the Apache Tomcat homepage at <http://tomcat.apache.org/>. The page features the Apache logo (a yellow cat) and the Apache Tomcat logo. A search bar is present. The main content area discusses the Apache Tomcat software, its open source nature, and its implementation of Java Servlet, JavaServer Pages, Java Expression Language, and Java WebSocket technologies. It also mentions the Java Community Process. The sidebar contains links for Apache Tomcat, Download (including version 9, 8, 7, Connectors, Native, Taglibs, Archives), and Documentation (Tomcat 9.0, 8.5, 8.0).

The screenshot shows the Apache Tomcat download page at <https://tomcat.apache.org/download-90.cgi#9.0.2>. The page title is "9.0.2 (beta)". It includes sections for "Problems?", "Get Involved", "Media", "Misc", and "Binary Distributions". The "Binary Distributions" section lists various download links for different operating systems and architectures, such as zip, tar.gz, and Windows Service Installer files.

2. JSP 개발 환경 구축하기

예제 1-3 톰캣 설치하기

2. 다운로드한 apache-tomcat-9.0.2-windows-x64.zip의 압축을 풀고
하위에 있는 apache-tomcat-9.0.2 폴더를 C 드라이브로 옮기기
(C:\ 경로에 바로 압축을 풀어도 됨)

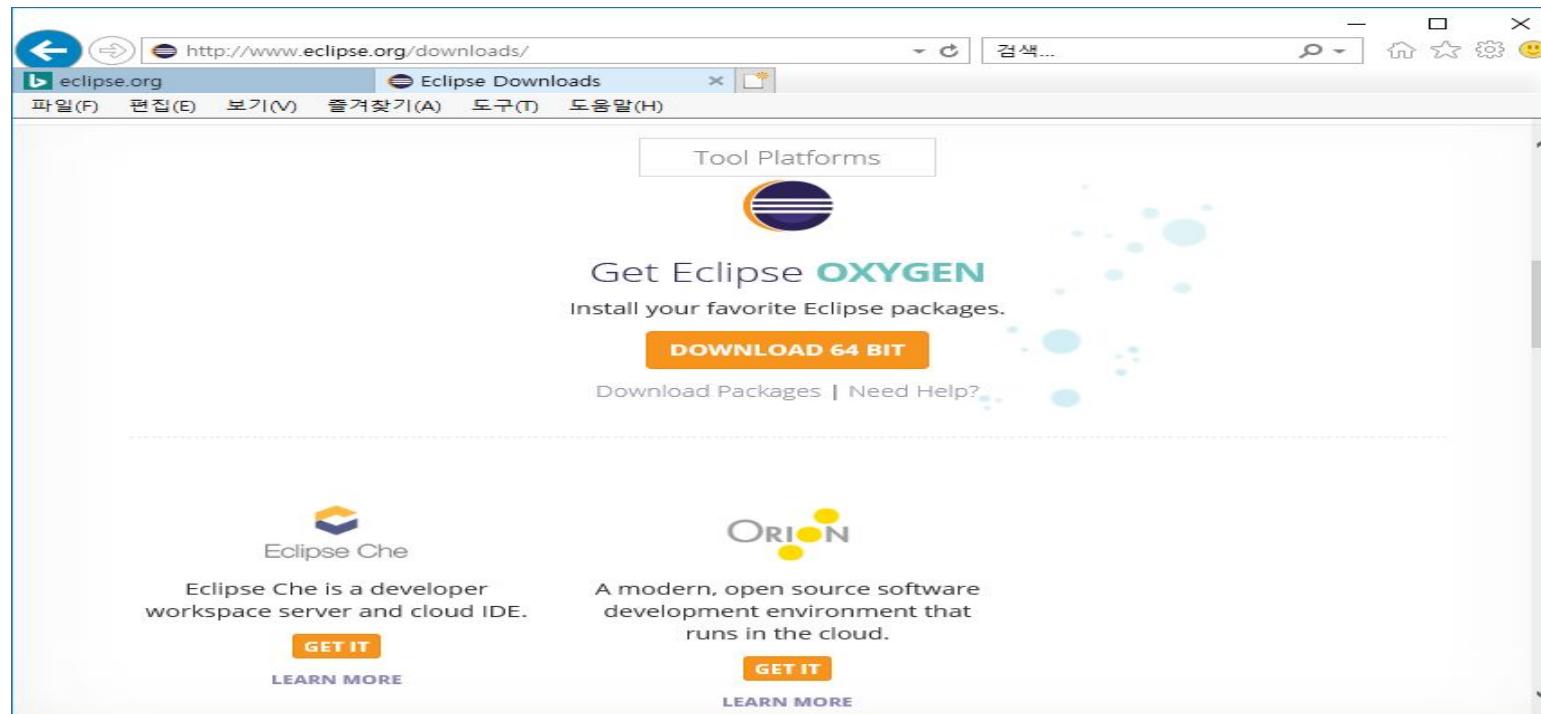


2. JSP 개발 환경 구축하기

예제 1-4 이클립스 다운로드하고 설치하기

1. 이클립스 사이트에 접속하기

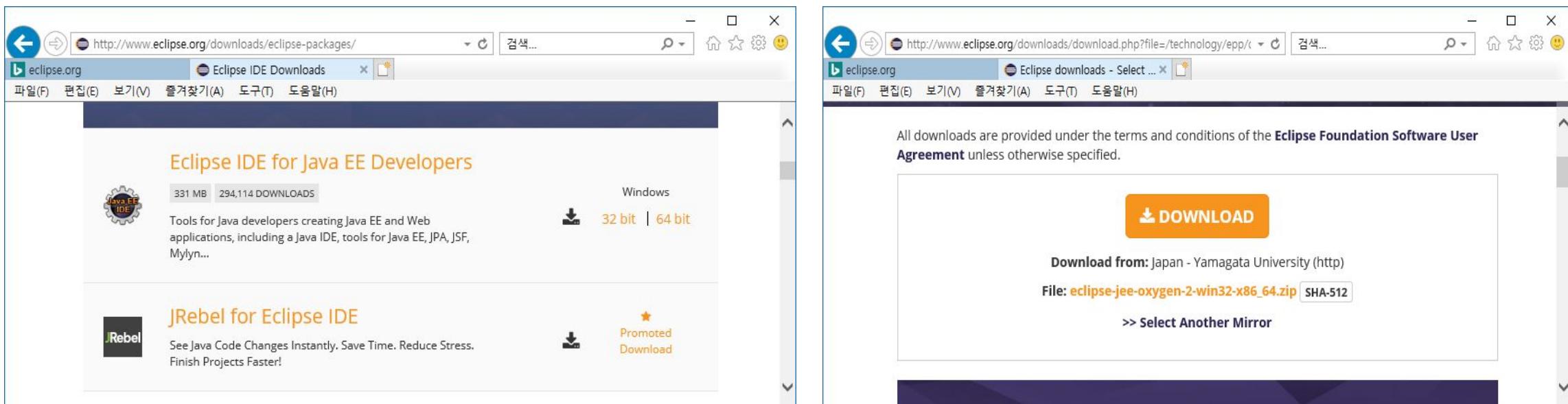
- <http://www.eclipse.org/downloads/>



2. JSP 개발 환경 구축하기

예제 1-4 이클립스 다운로드하고 설치하기

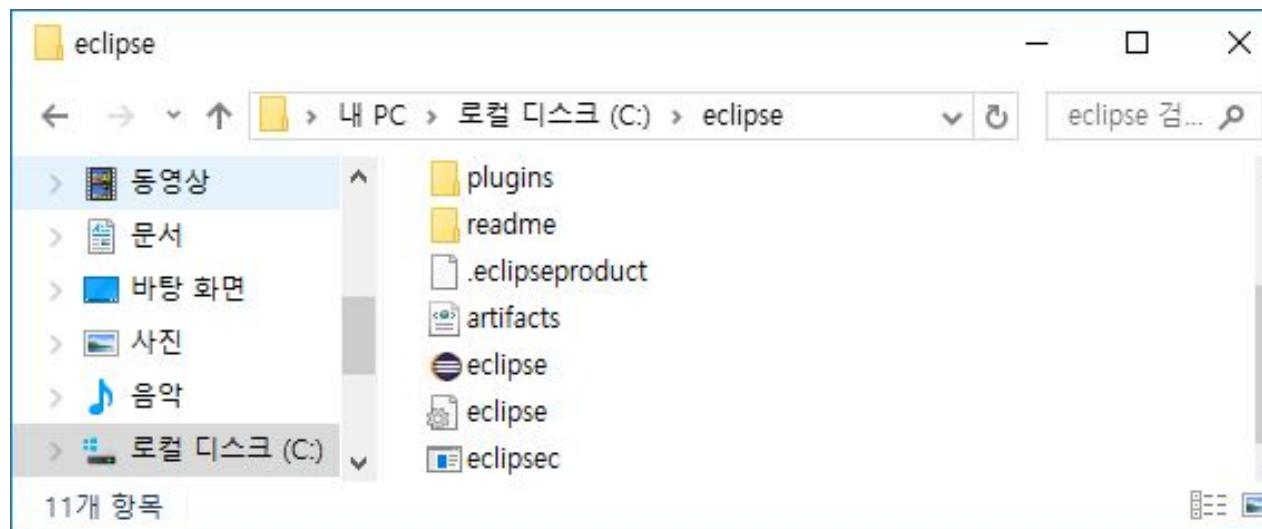
2. 이클립스 설치 파일 다운로드하기



2. JSP 개발 환경 구축하기

예제 1-4 이클립스 다운로드하고 설치하기

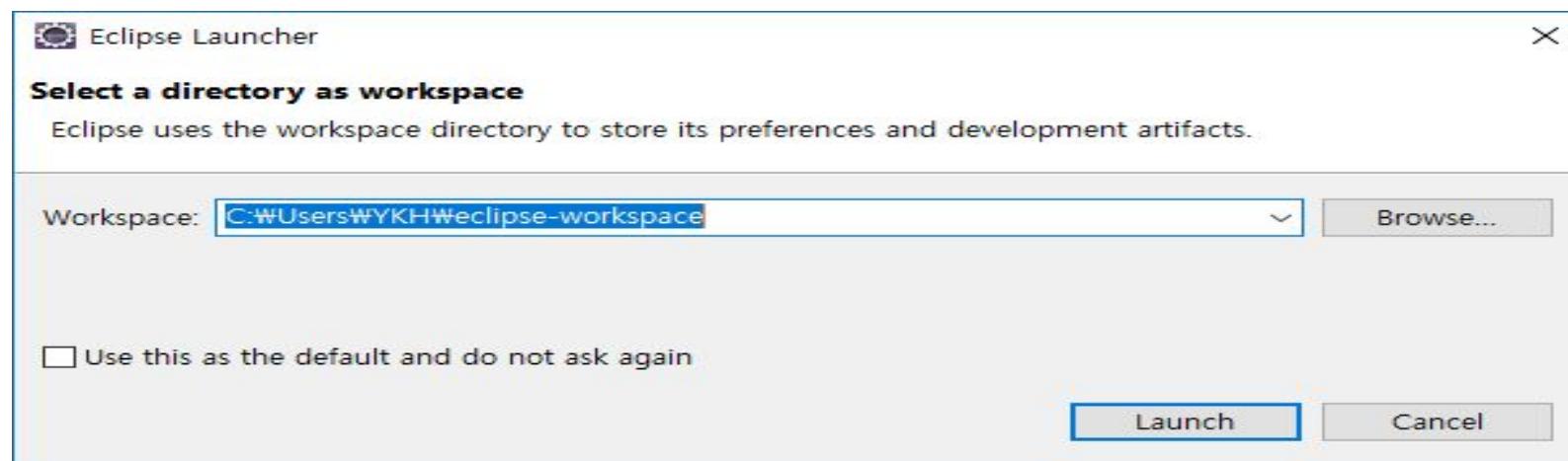
3. 설치 완료하고 실행하기



2. JSP 개발 환경 구축하기

예제 1-4 이클립스 다운로드하고 설치하기

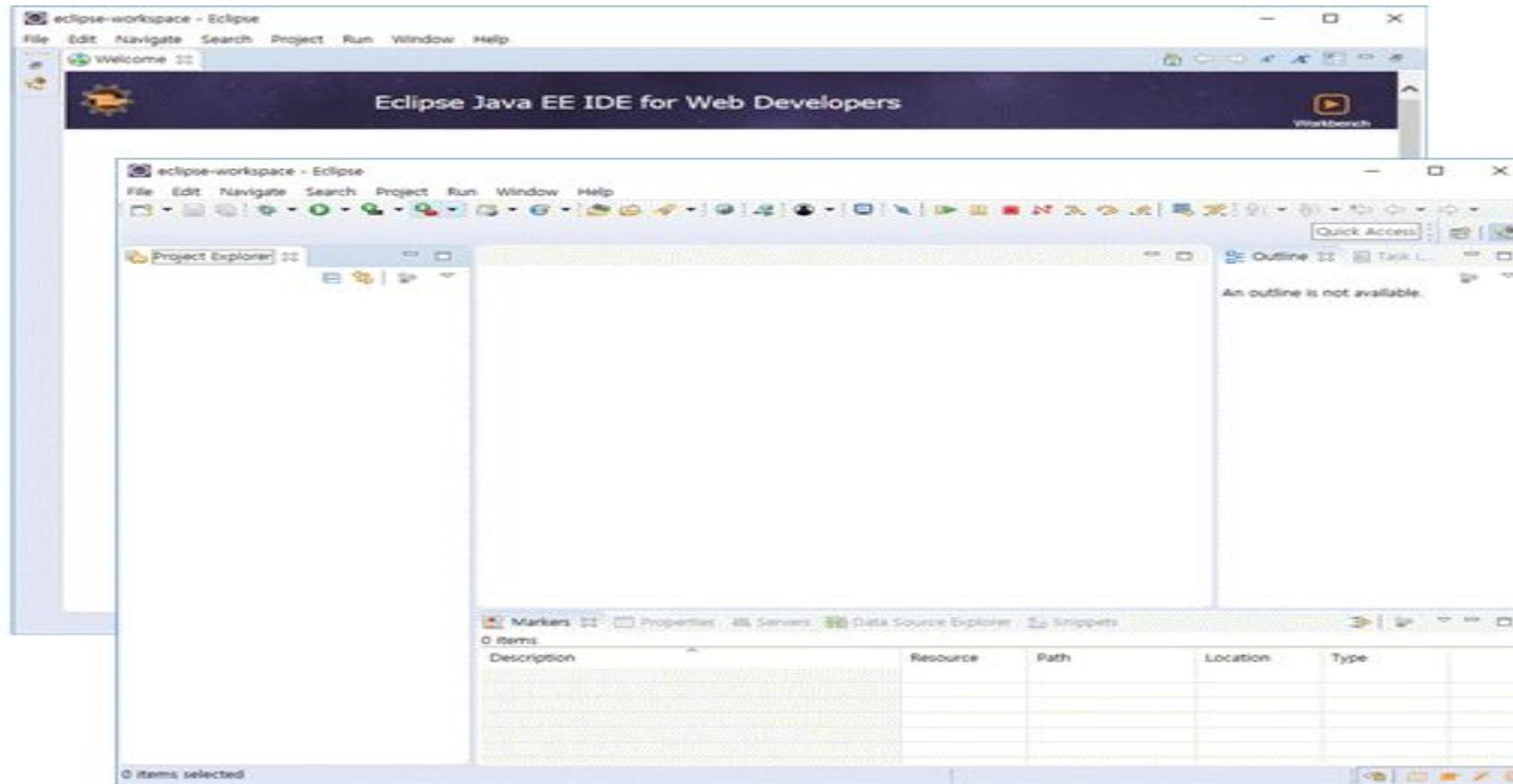
4. 이클립스 작업 공간 설정하기



2. JSP 개발 환경 구축하기

예제 1-4 이클립스 다운로드하고 설치하기

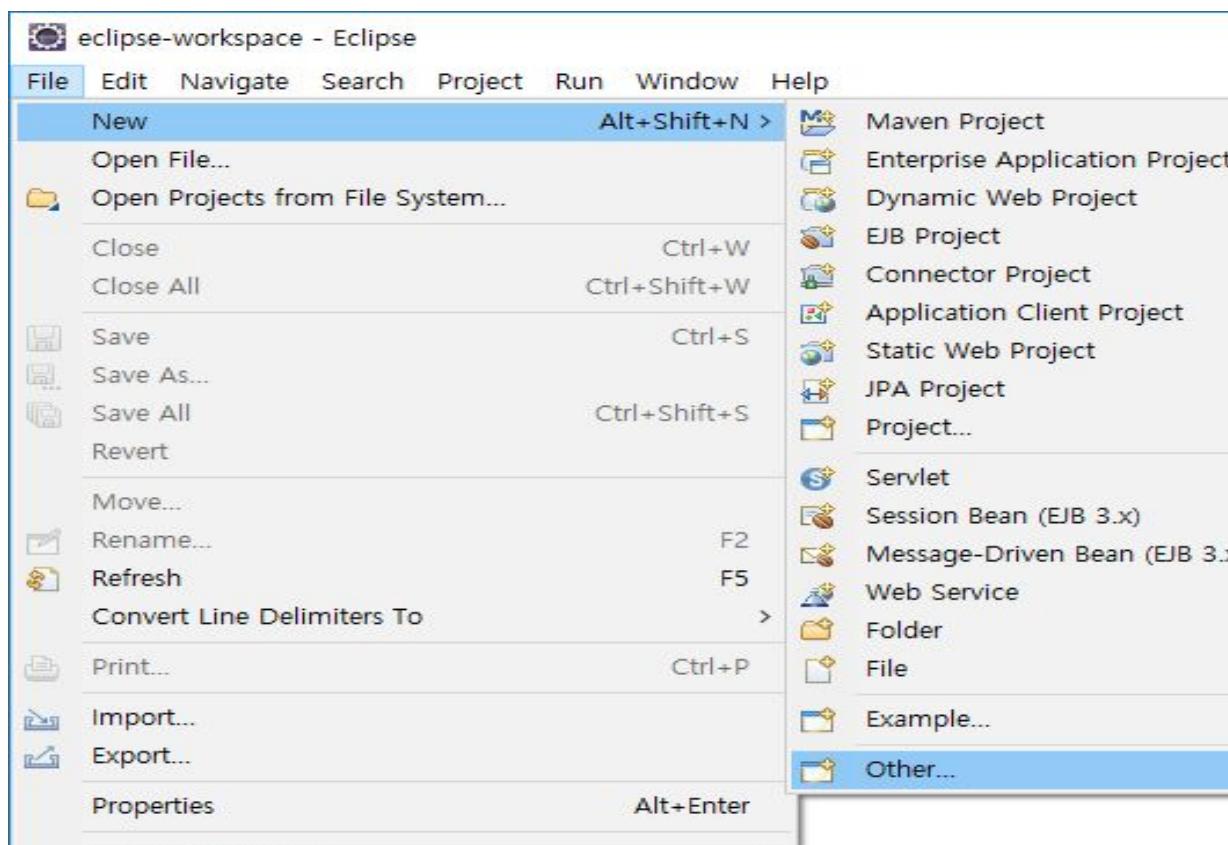
5. 이클립스 실행 화면



2. JSP 개발 환경 구축하기

예제 1-5 이클립스와 톰캣 서버 연동하기

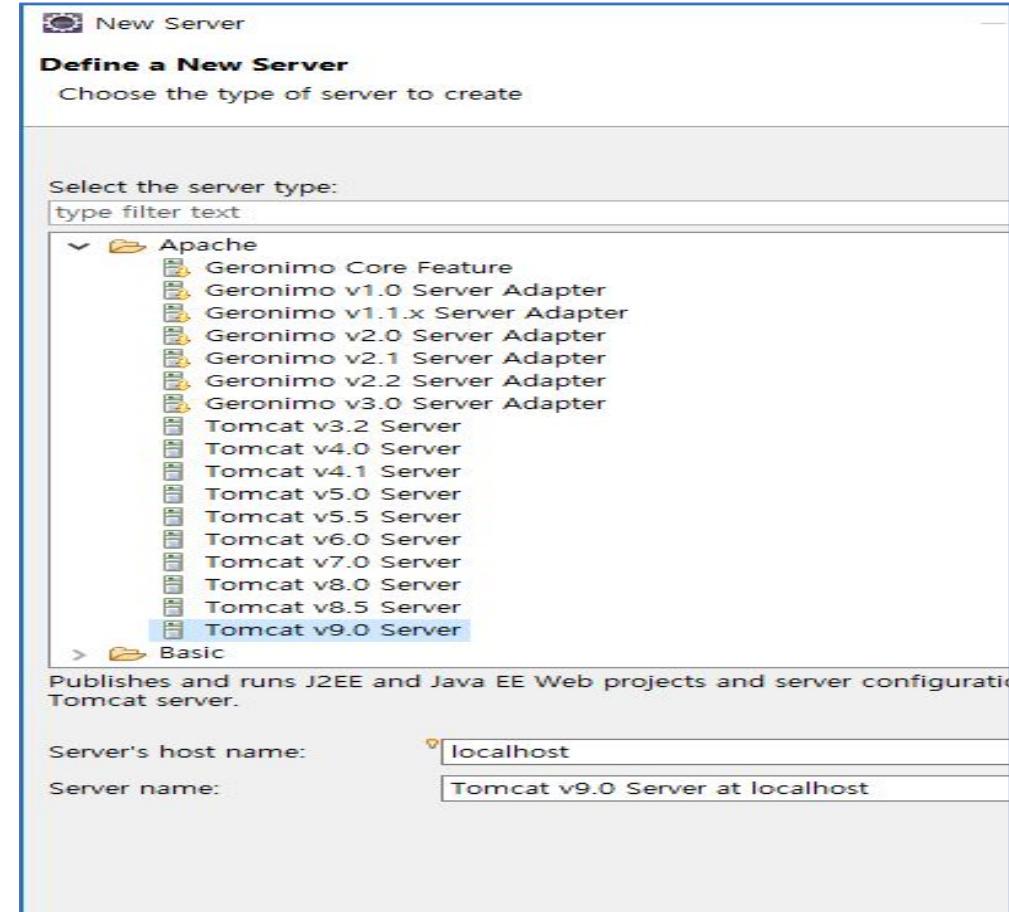
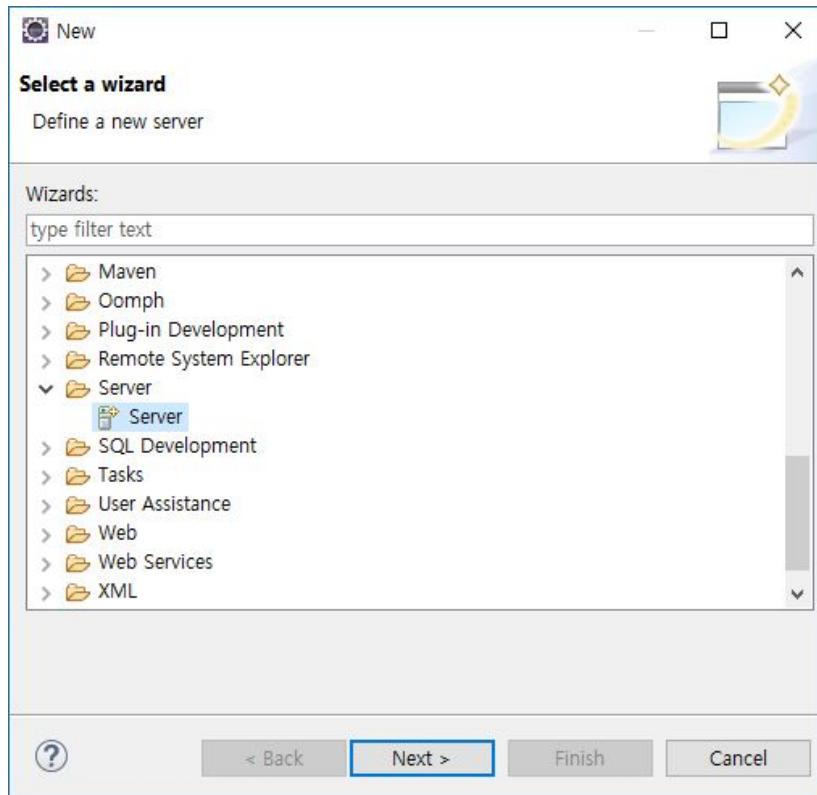
1. [Server] 프로젝트 생성하기



2. JSP 개발 환경 구축하기

예제 1-5 이클립스와 톰캣 서버 연동하기

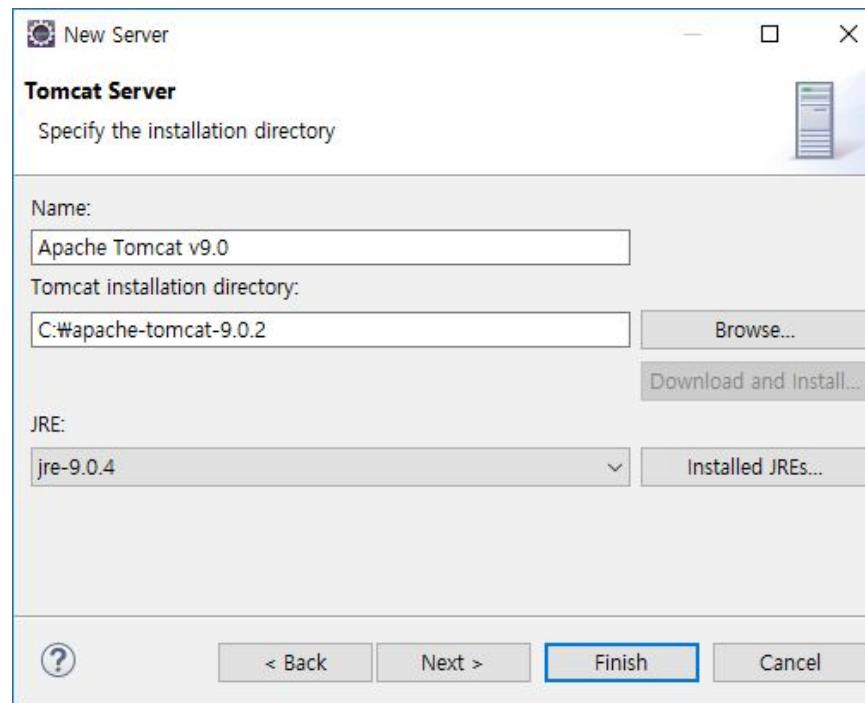
2. 웹 서버 유형 설정하기



2. JSP 개발 환경 구축하기

예제 1-5 이클립스와 톰캣 서버 연동하기

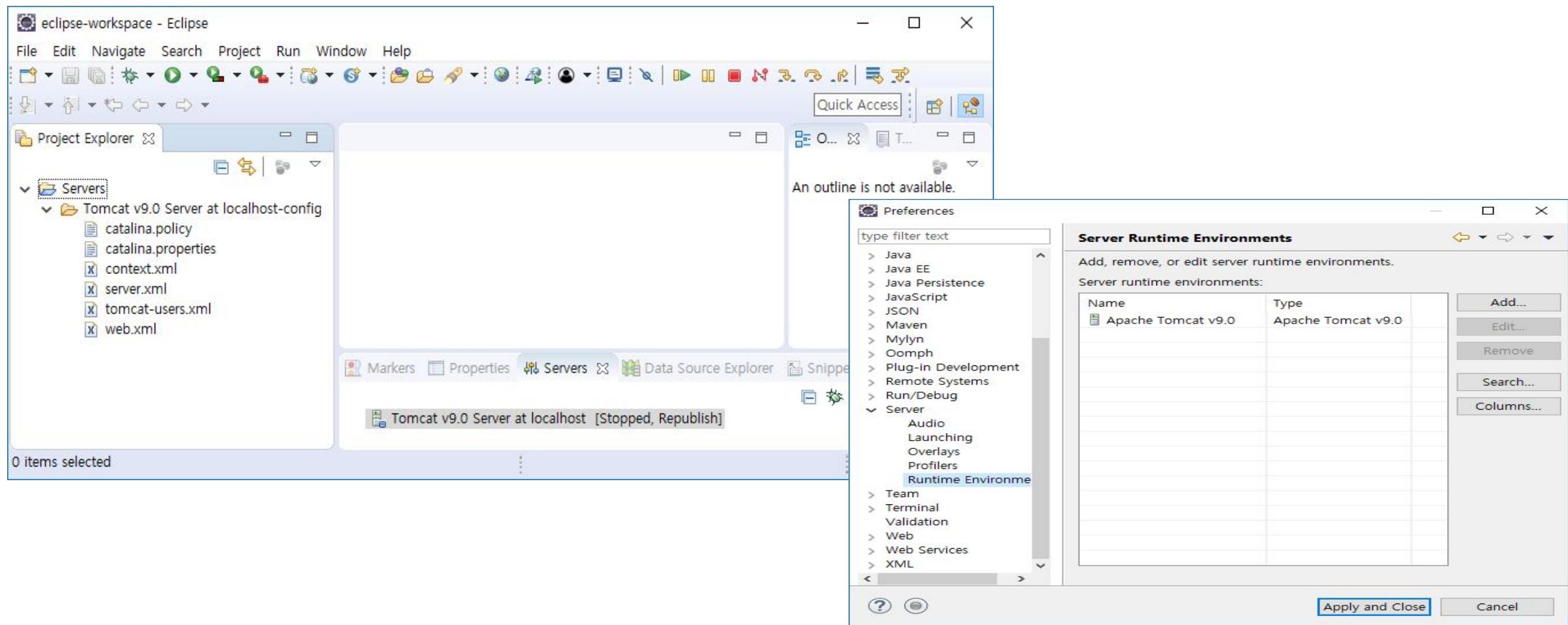
3. 웹 서버 위치와 JRE 설정하기



2. JSP 개발 환경 구축하기

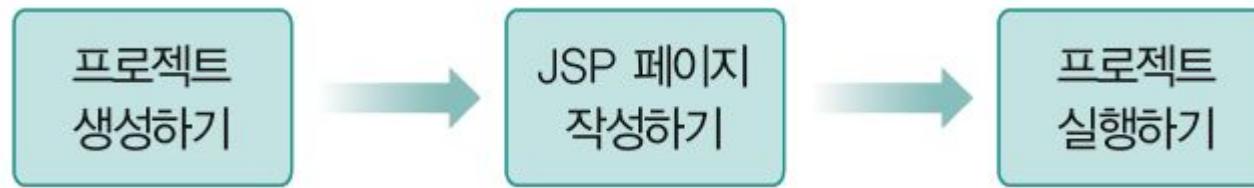
예제 1-5 이클립스와 톰캣 서버 연동하기

4. 연동 확인하기



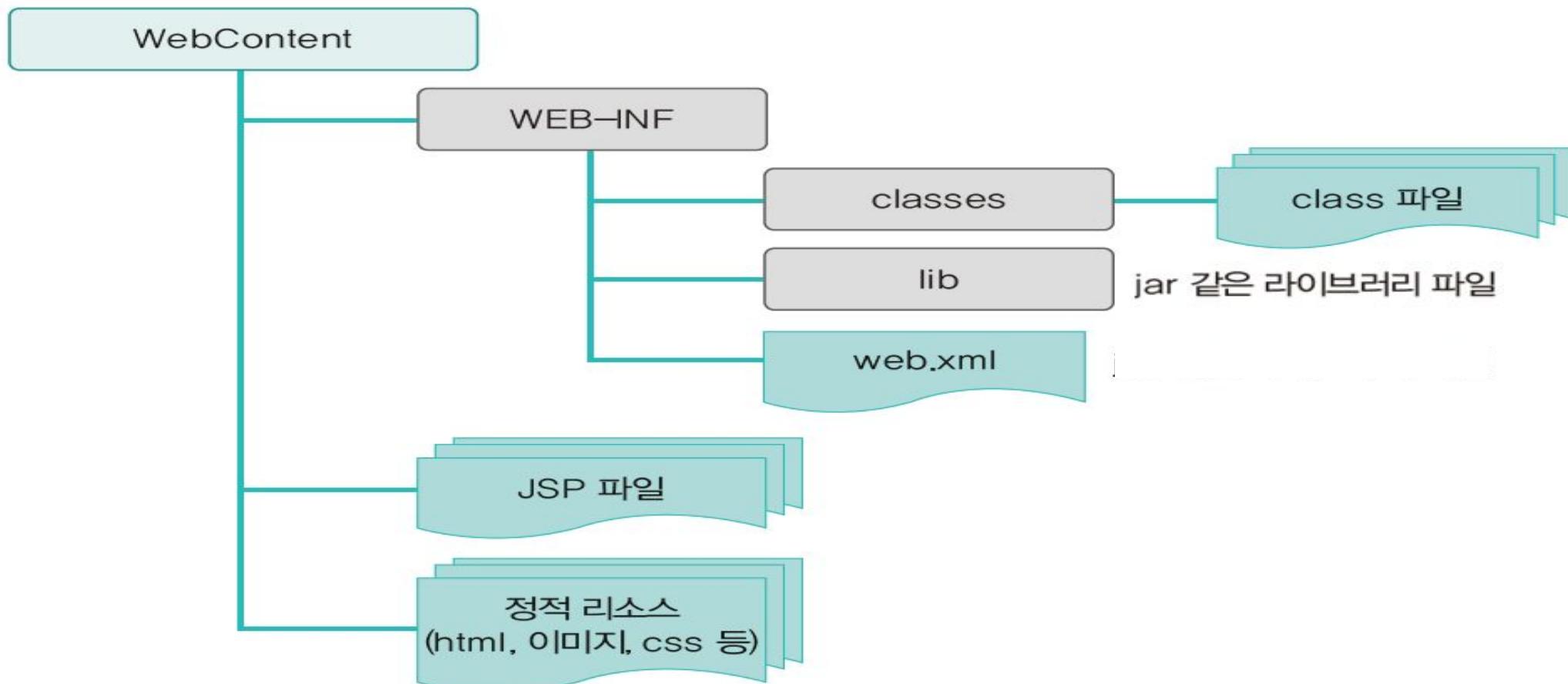
2. JSP 개발 환경 구축하기

- ❖ 프로젝트 만들고 실행하기



2. JSP 개발 환경 구축하기

❖ 동적 웹 프로젝트의 구조



2. JSP 개발 환경 구축하기

예제 1-6 프로젝트 만들고 실행하기

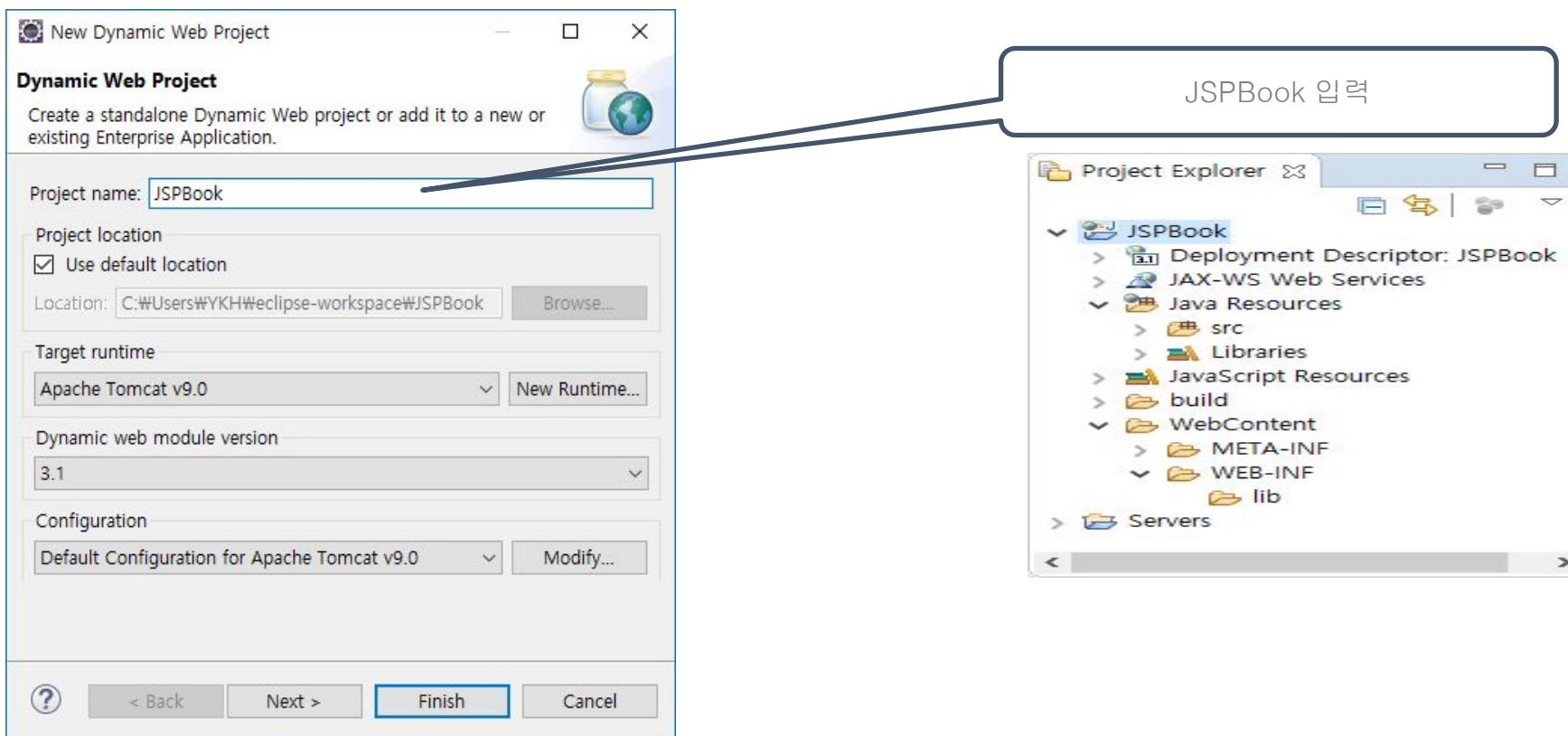
1. 동적 웹 프로젝트 생성하기



2. JSP 개발 환경 구축하기

예제 1-6 프로젝트 만들고 실행하기

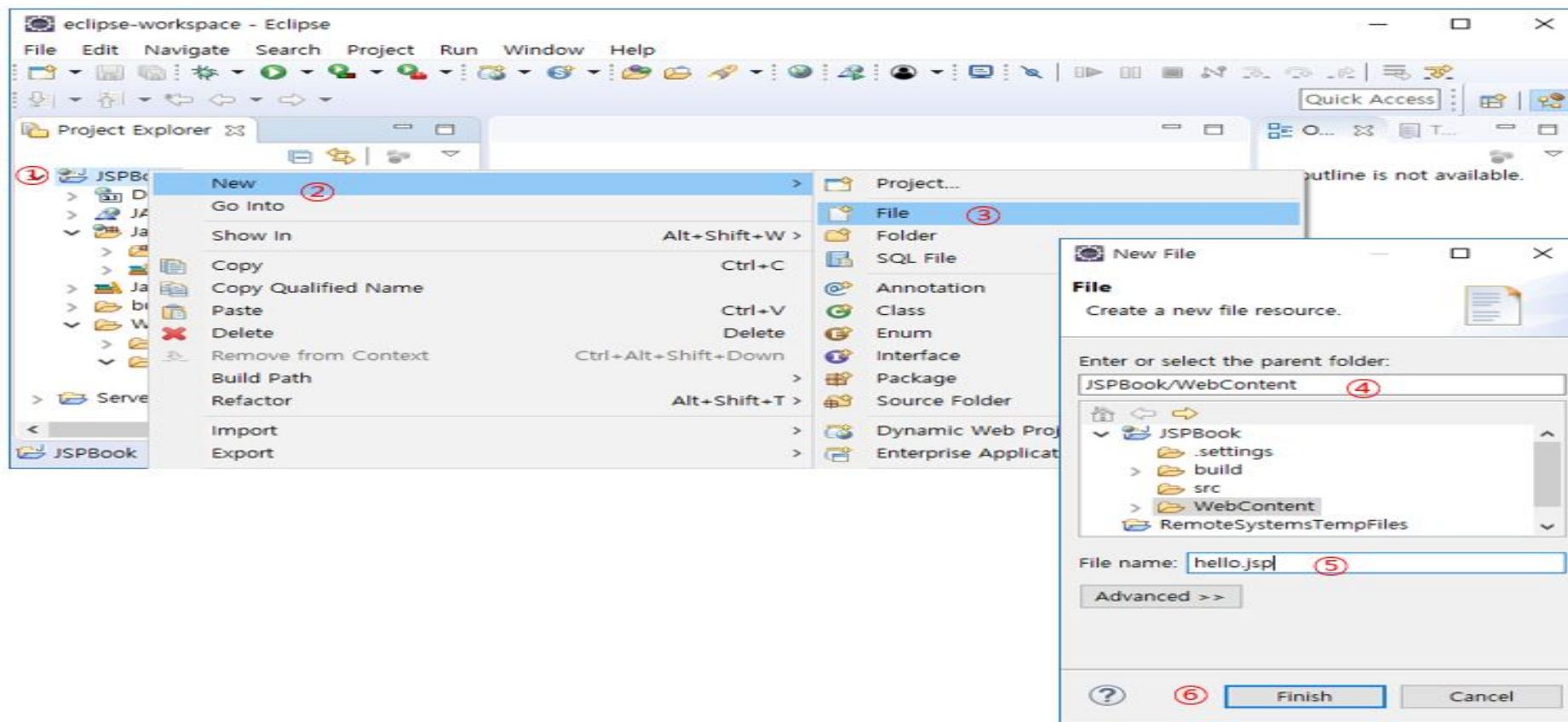
2. 프로젝트명 설정하기



2. JSP 개발 환경 구축하기

예제 1-6 프로젝트 만들고 실행하기

3. JSP 파일 생성하기



2. JSP 개발 환경 구축하기

예제 1-6 프로젝트 만들고 실행하기

4. JSP 페이지 코드 작성하기

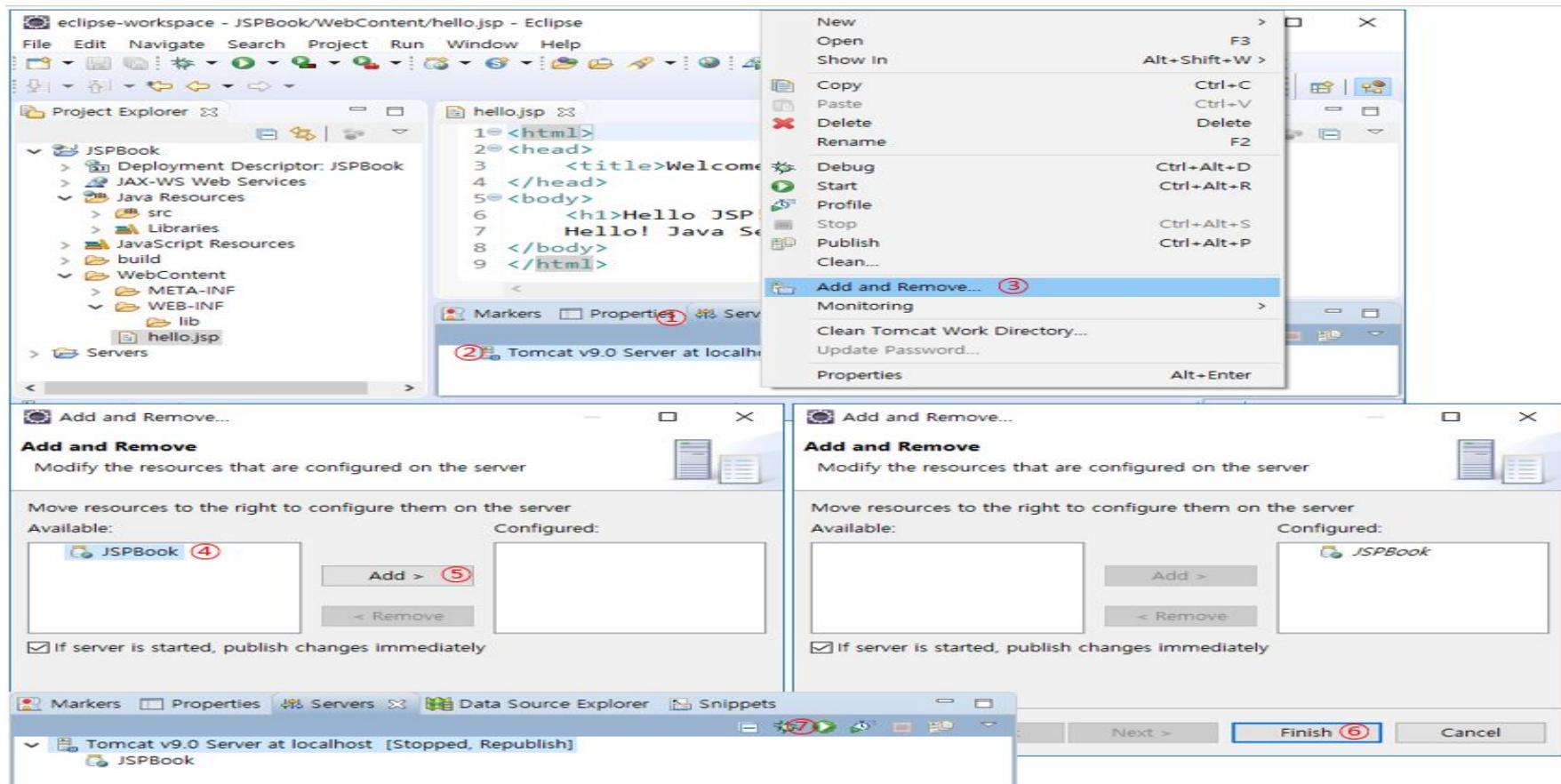


```
1<html>
2<head>
3    <title>Welcome</title>
4</head>
5<body>
6    <h1>Hello JSP!!</h1>
7    Hello! Java Server Pages.
8</body>
9</html>
```

2. JSP 개발 환경 구축하기

예제 1-6 프로젝트 만들고 실행하기

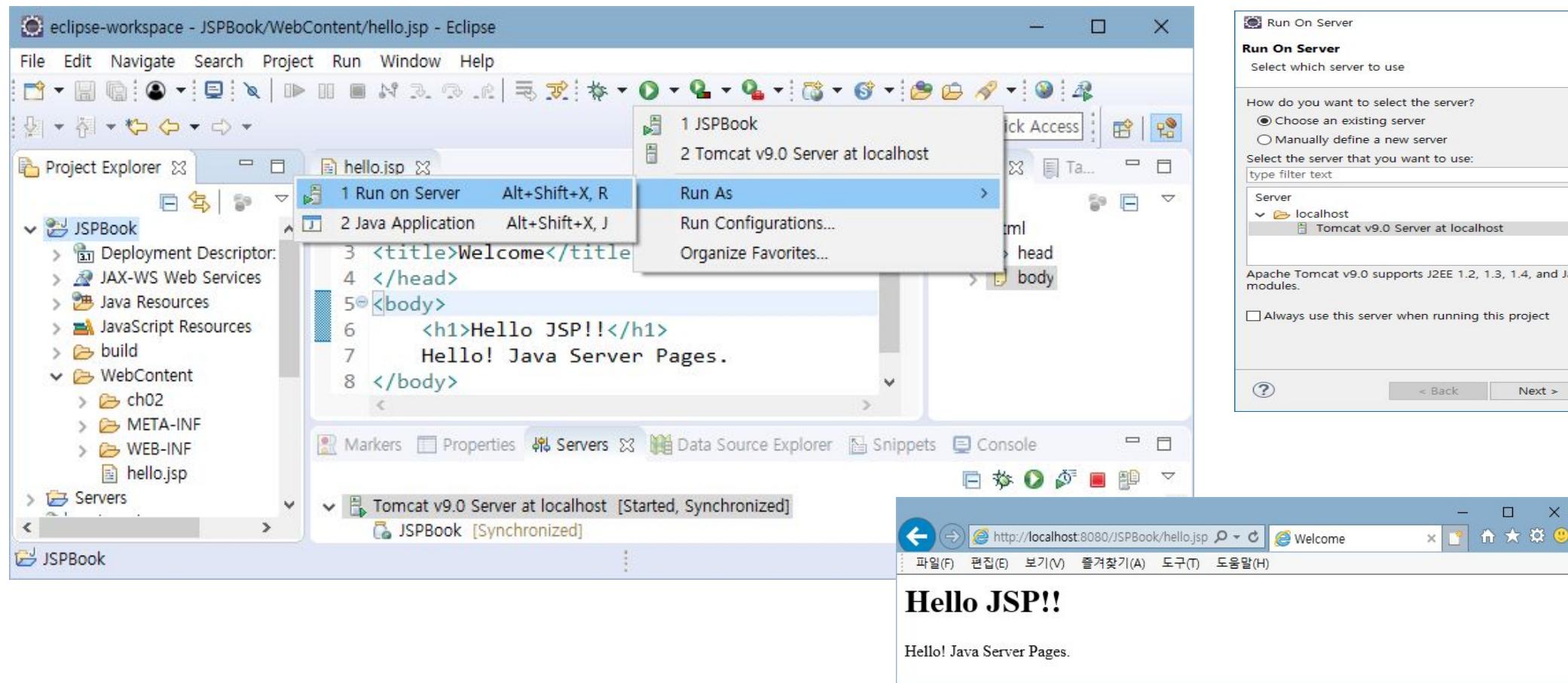
5. 톰캣 서버에 등록하기



2. JSP 개발 환경 구축하기

예제 1-6 프로젝트 만들고 실행하기

6. 실행 결과 확인하기



3. [웹 쇼핑몰] 프로젝트 생성하기

예제 1-7 웹 쇼핑몰 프로젝트 만들기



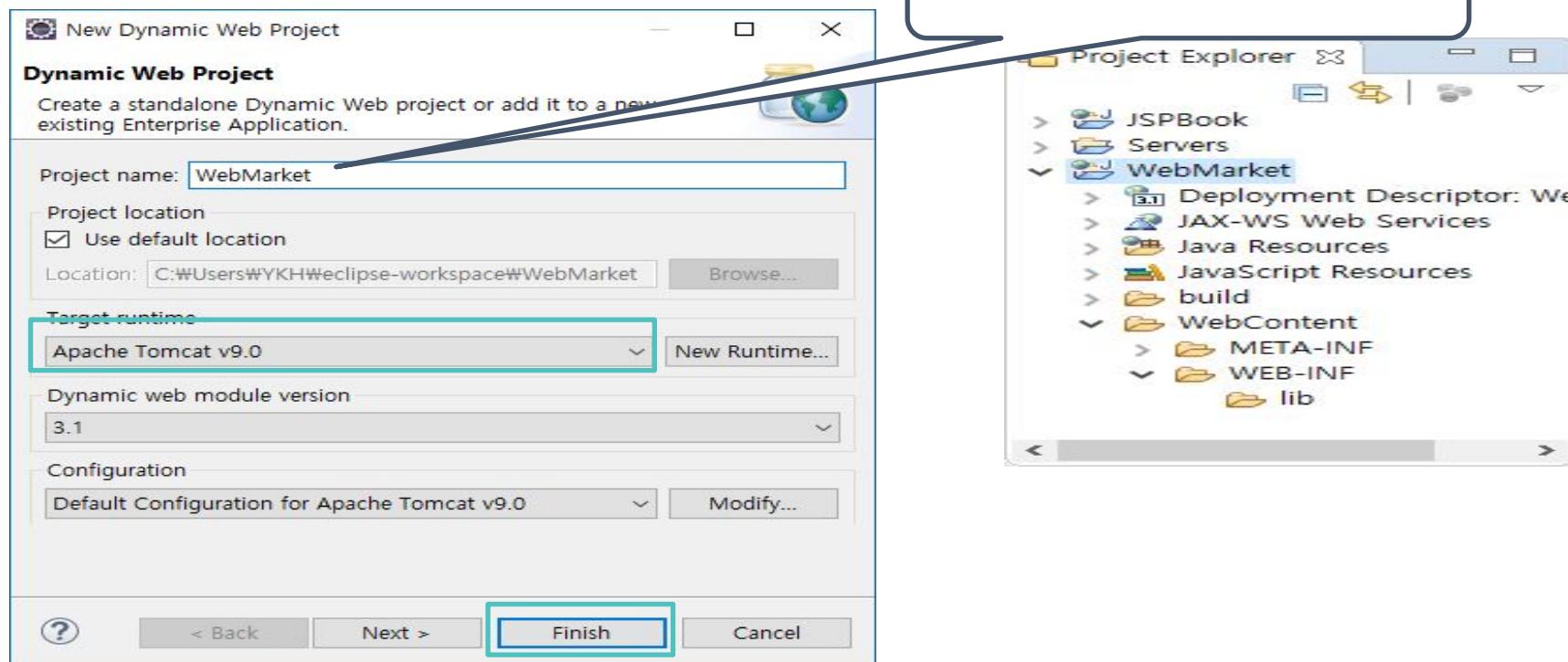
3. [웹 쇼핑몰] 프로젝트 생성하기

❖ [프로젝트 생성]

1. 동적 웹 프로젝트 생성하기

- 이클립스의 [File]-[New]-[Dynamic Web Project]를 선택

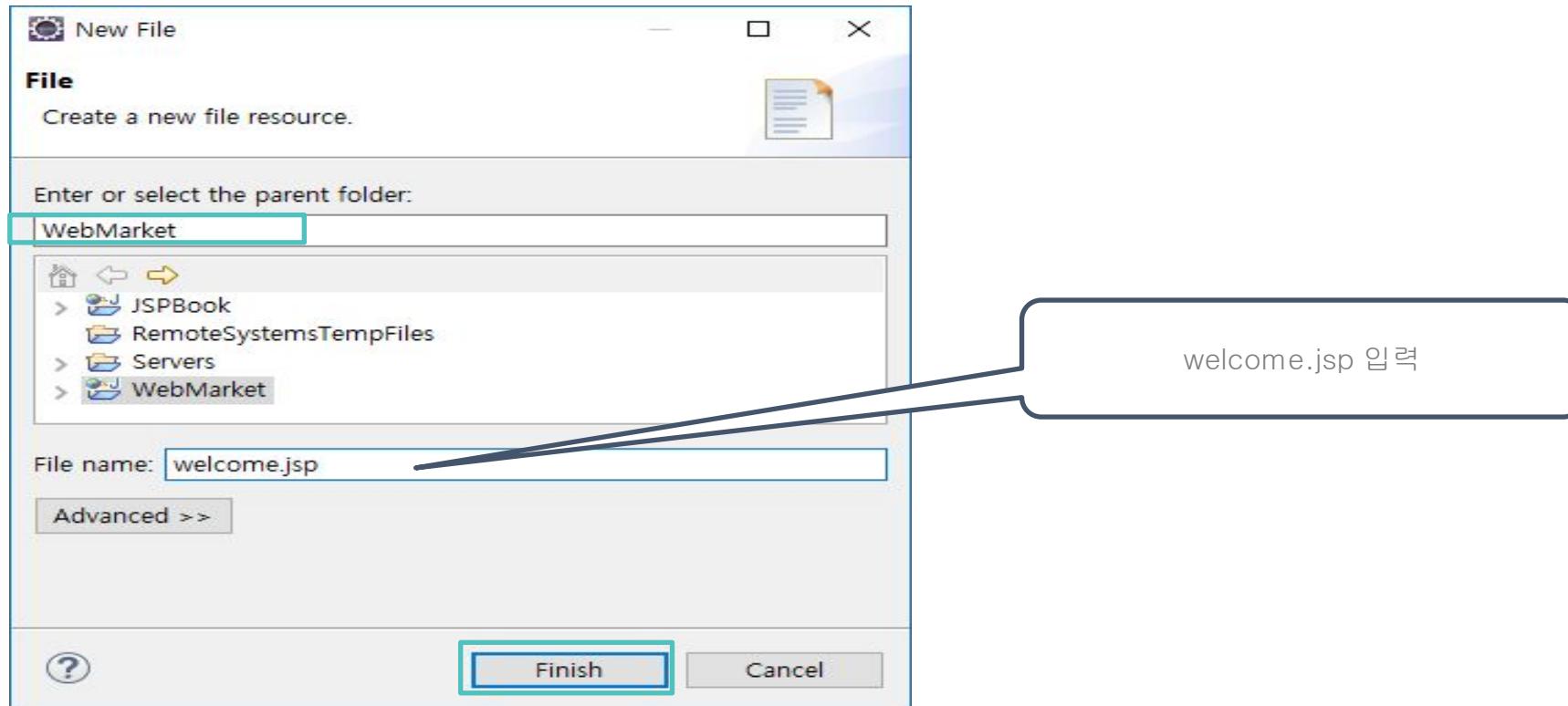
2. 프로젝트명 설정하기



3. [웹 쇼핑몰] 프로젝트 생성하기

❖ [JSP 페이지 작성하기]

3. JSP 파일 생성하기



3. [웹 쇼핑몰] 프로젝트 생성하기

❖ [JSP 페이지 작성하기]

4. JSP 페이지 코드 작성하기

WebMarket/WebContent/welcome.jsp

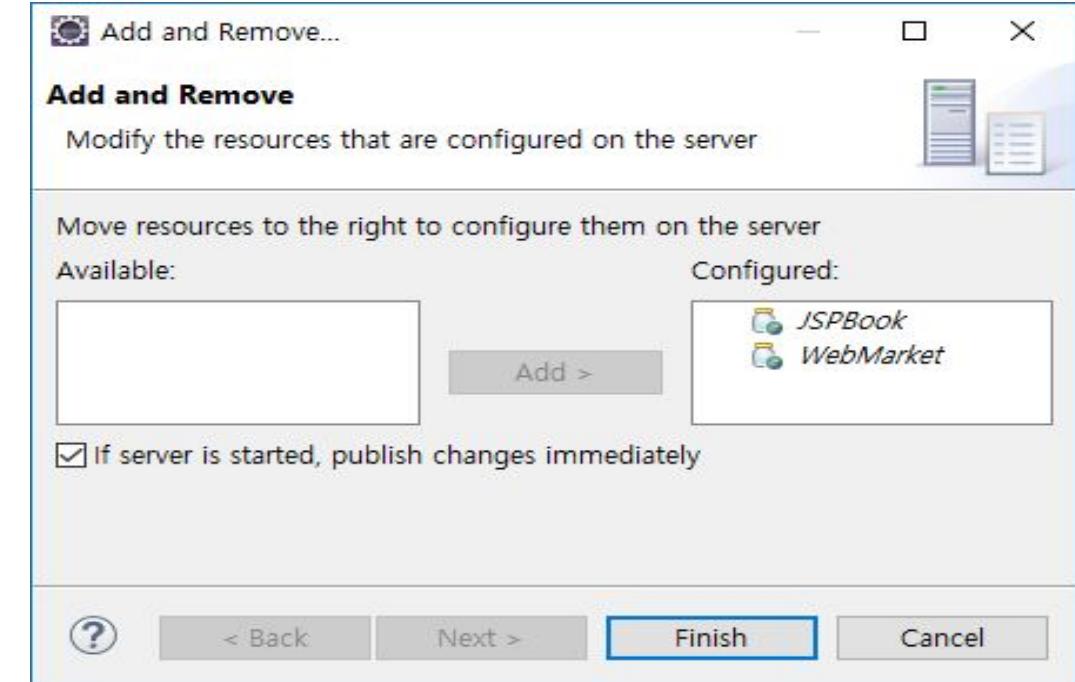
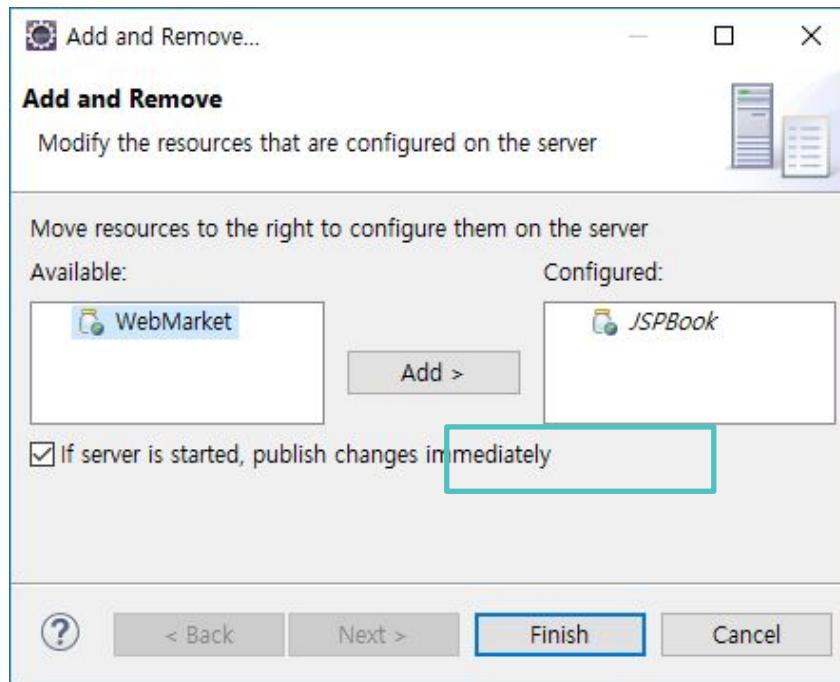
```
01 <html>
02 <head>
03 <title>Welcome</title>
04 </head>
05 <body>
06     <h1>Welcome to Web Shopping Mall</h1>
07     <h3>Welcome to Web Market!</h3>
08 </body>
09 </html>
```

3. [웹 쇼핑몰] 프로젝트 생성하기

❖ [프로젝트 실행하기]

5. 톰캣 서버에 등록하기

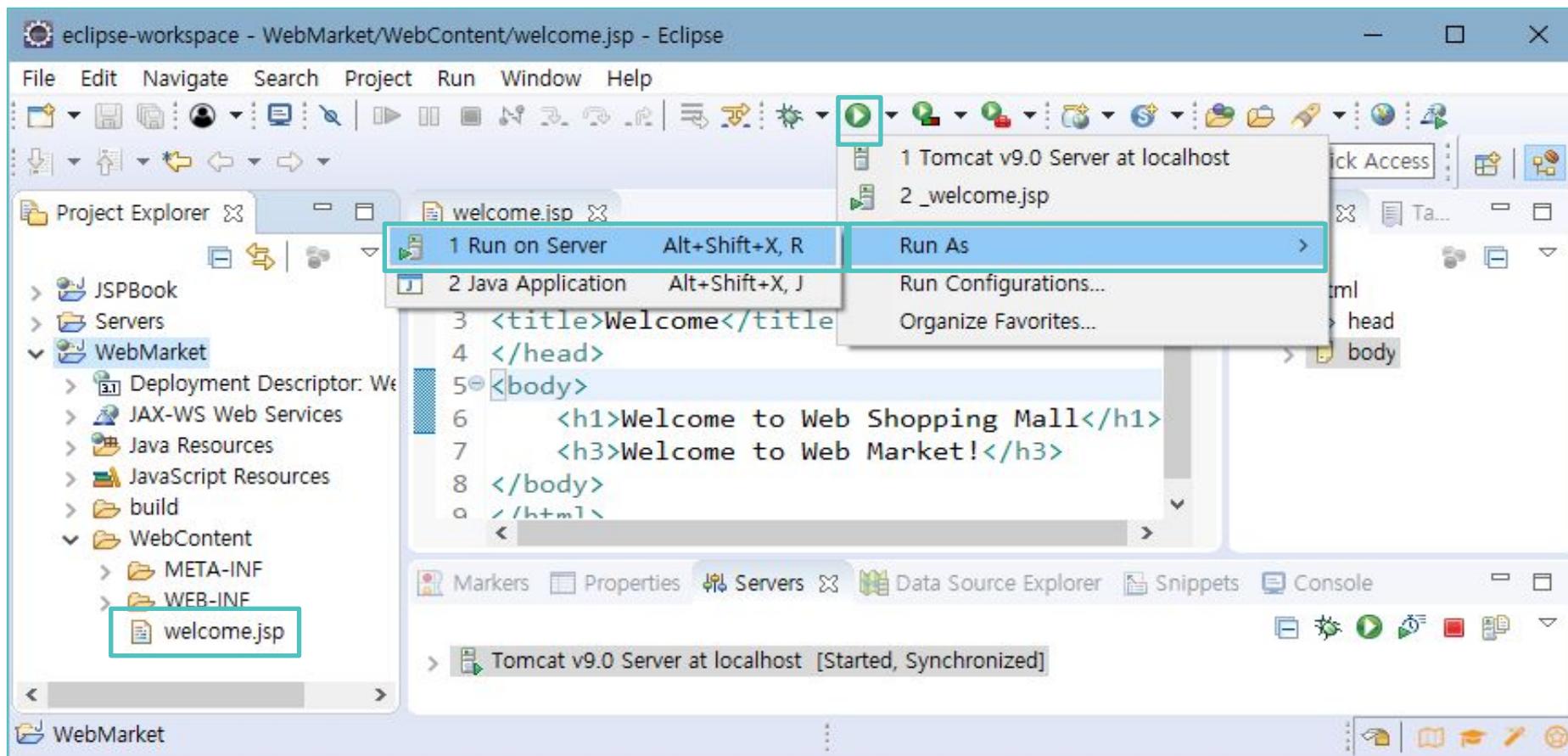
- 이클립스의 아래쪽 창 [Servers] 탭에 있는 Tomcat v9.0 Server at localhost에서 마우스 오른쪽 버튼을 누르고 [Add and Remove]를 선택



3. [웹 쇼핑몰] 프로젝트 생성하기

❖ [프로젝트 실행하기]

6. 실행 결과 확인하기



웹프로그래밍이란?

1. 웹프로그래밍 : 웹어플리케이션을 구현하는 행위
2. 웹어플리케이션 : 웹을 기반으로 작동되는 프로그램
3. 웹 : 1개 이상의 사이트가 연결되어 있는 인터넷 서비스의 한 형태
4. 인터넷 : 1개 이상의 네트워크가 연결되어 있는 형태

- 프로토콜(Protocol) : 네트워크상에서 약속한 통신규약 (Http, FTP, SMTP, POP, DHCP)
- IP : 네트워크상에서 컴퓨터를 식별할 수 있는 주소
- DNS : IP주소를 인간이 쉽게 외우도록 맵핑한 문자열
- Port : IP주소가 컴퓨터를 식별할 수 있게 해준다면, Port번호는 해당컴퓨터의 구동되고 있는 프로그램을 구분할 수 있는 번호

웹사이트

http://www.ludenscode.com:80/kr/index

프로토콜

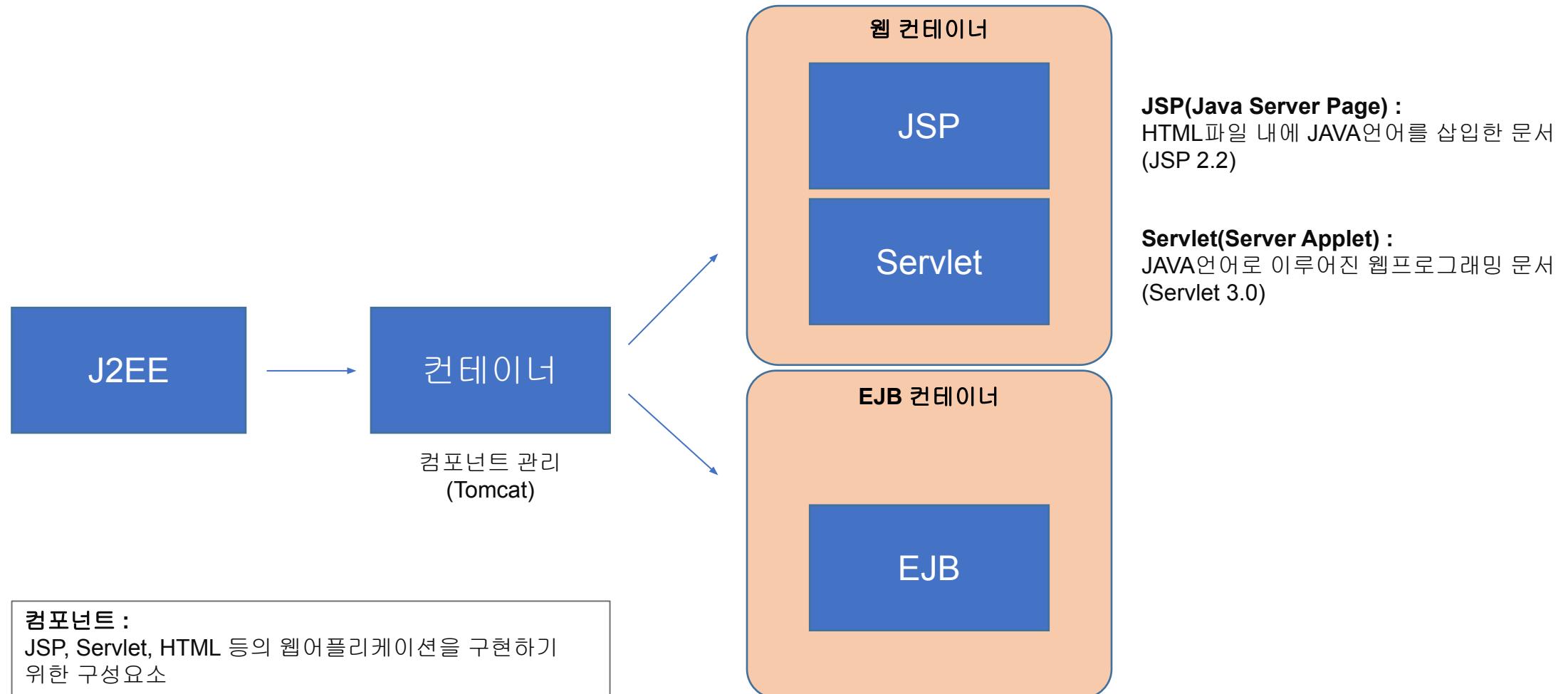
컴퓨터 주소(DNS를 통한 IP주소로 변경)

port

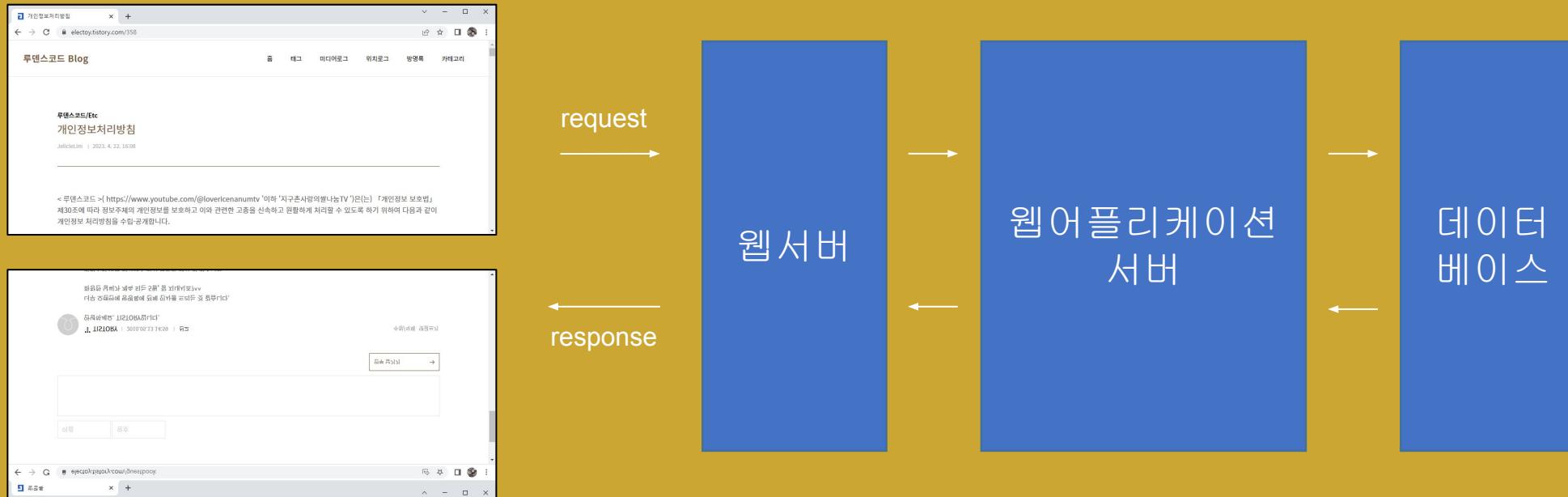
Information path

JAVA웹

JAVA플랫폼(J2SE, J2EE, J2ME)중에서 J2EE를 이용한 웹프로그래밍 입니다.



- 웹서버 : 클라이언트의 요청에 의해 정보를 제공해 주는 서버 (Apache, IIS).
별도의 구현이 필요한 로직이 있을 경우 웹어플리케이션 서버에 요청.
- 웹브라우저 : 웹서버에 정보를 요청하고, 웹서버로부터 정보를 받는 매개체. 이때 HTTP 프로토콜을 사용함.



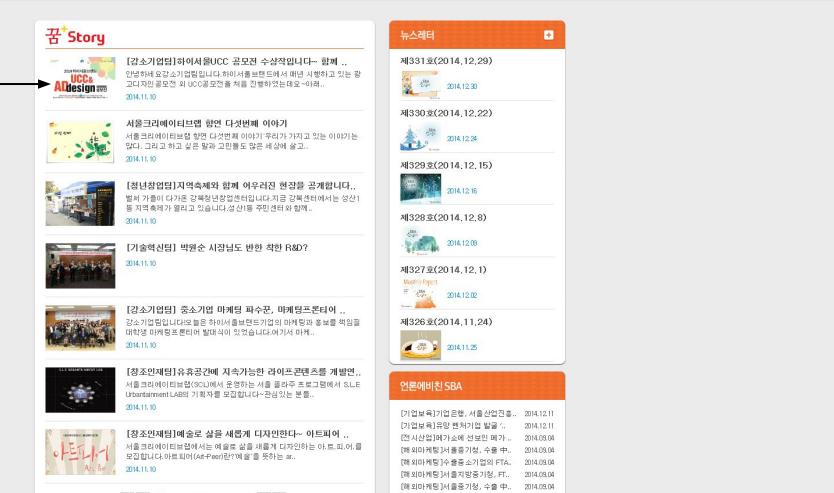
필요한 학습

1. JAVA : JAVA웹어플리케이션을 구현하기 위한 선행 학습 필요
2. HTML : 웹어플리케이션을 구현하기 위한 기본 언어
3. JavaScript : 클라이언트 기능을 구현하기 위한 언어
4. Jquery : JavaScript의 대표적인 라이브러리로써, 클라이언트 사이드 스크립트 언어를 단순화 할 수 있다.
5. CSS : 웹어플리케이션의 레이아웃 및 스타일을 지정하는 언어

Javascript / CSS



DataBase 자료



HTML문서

request
↔
response

웹서버
웹어플리케이션서버
데이터베이스

2강. 개발 환경 설정

- JDK 설치
- Path 설정
- 이클립스 다운로드
- 톰캣 설치
- 톰캣 환경 설정

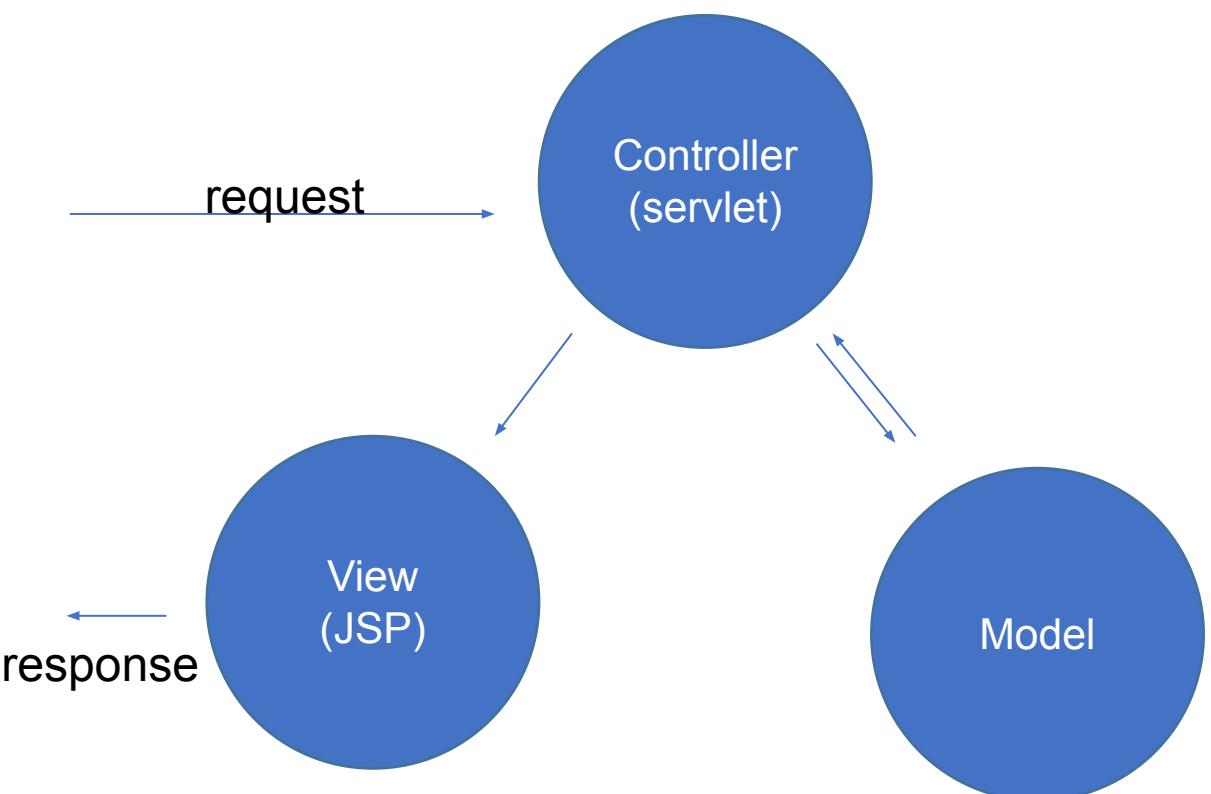
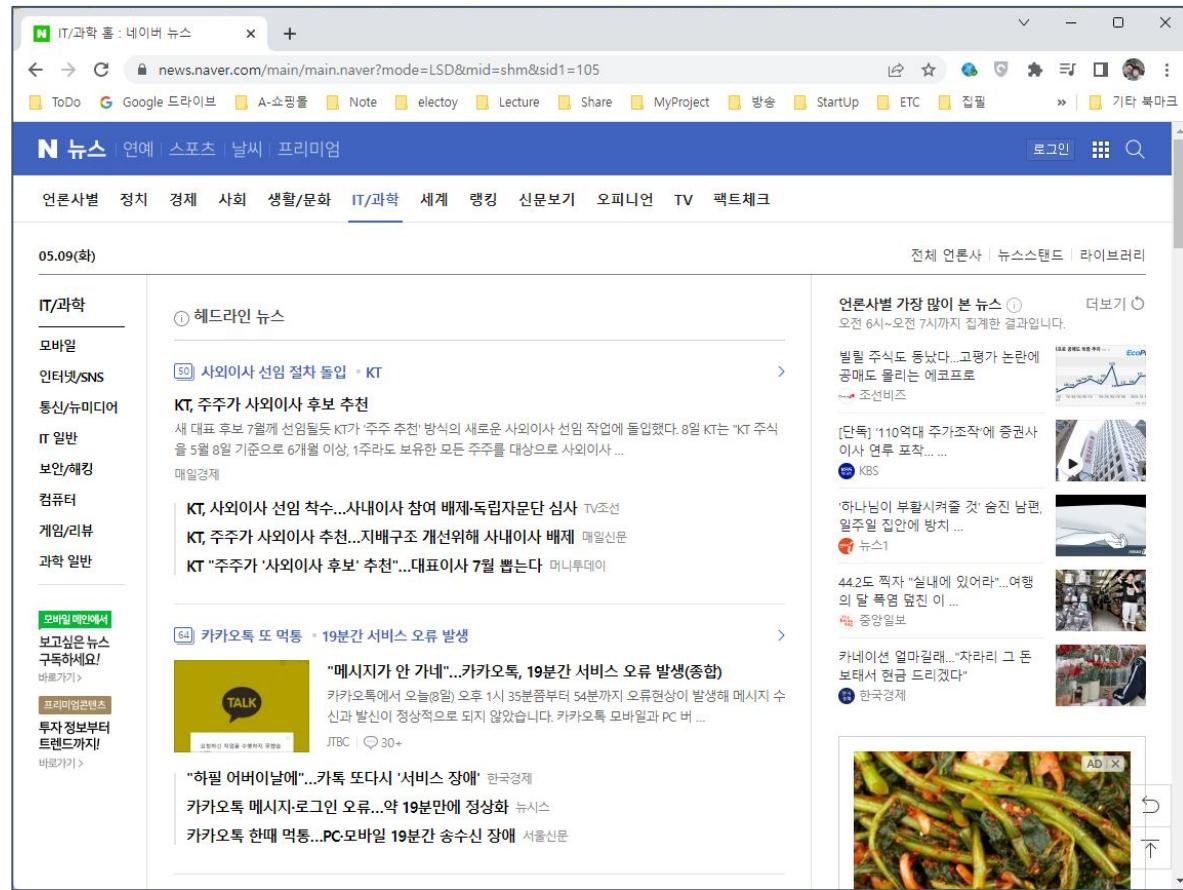
3강. JSP 맛보기

- JSP 문서 작성 하기
- JSP 아키텍처

3-1. JSP문서 작성 하기

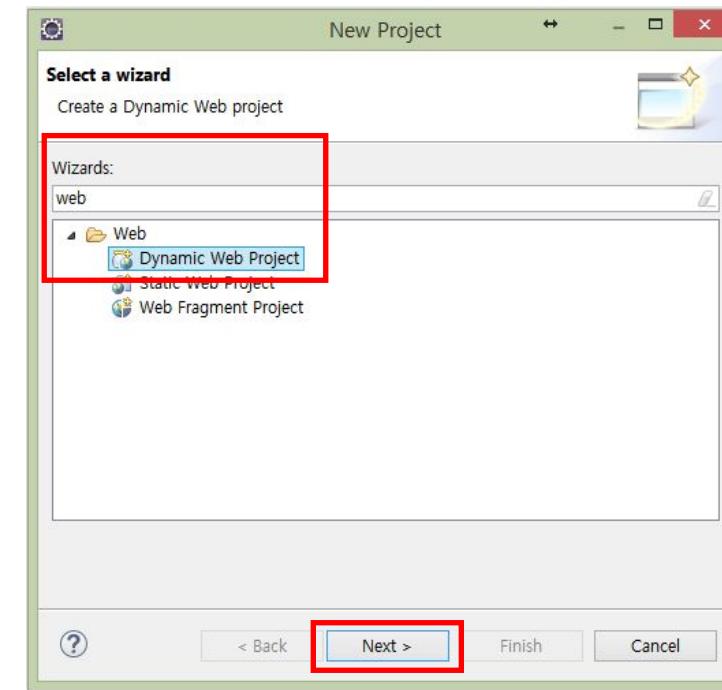
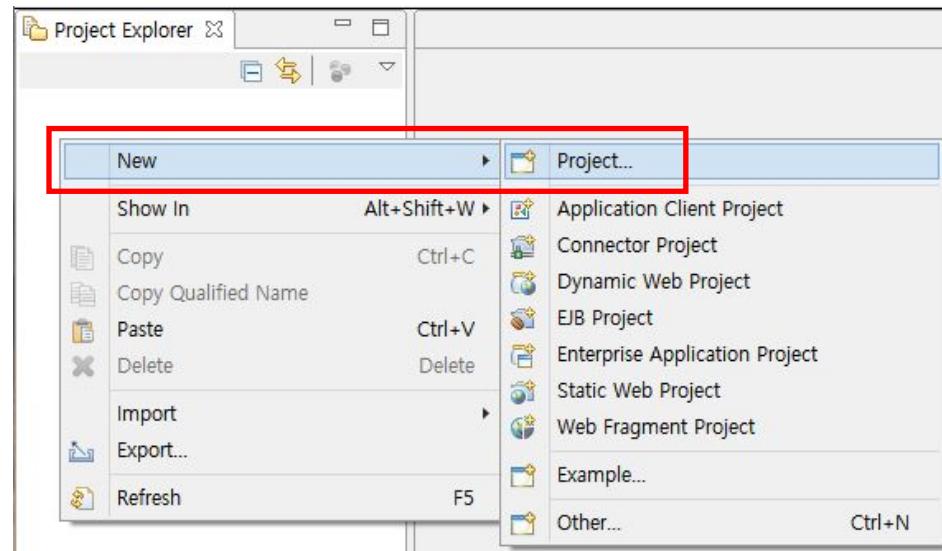
JSP특징

- 동적 웹애플리케이션 컴포넌트.
- .jsp 확장자.
- 클라이언트의 요청에 동적으로 작동하고, 응답은 html을 이용.
- jsp는 서블릿으로 변환되어 실행
- MVC패턴에서 View로 이용됨.



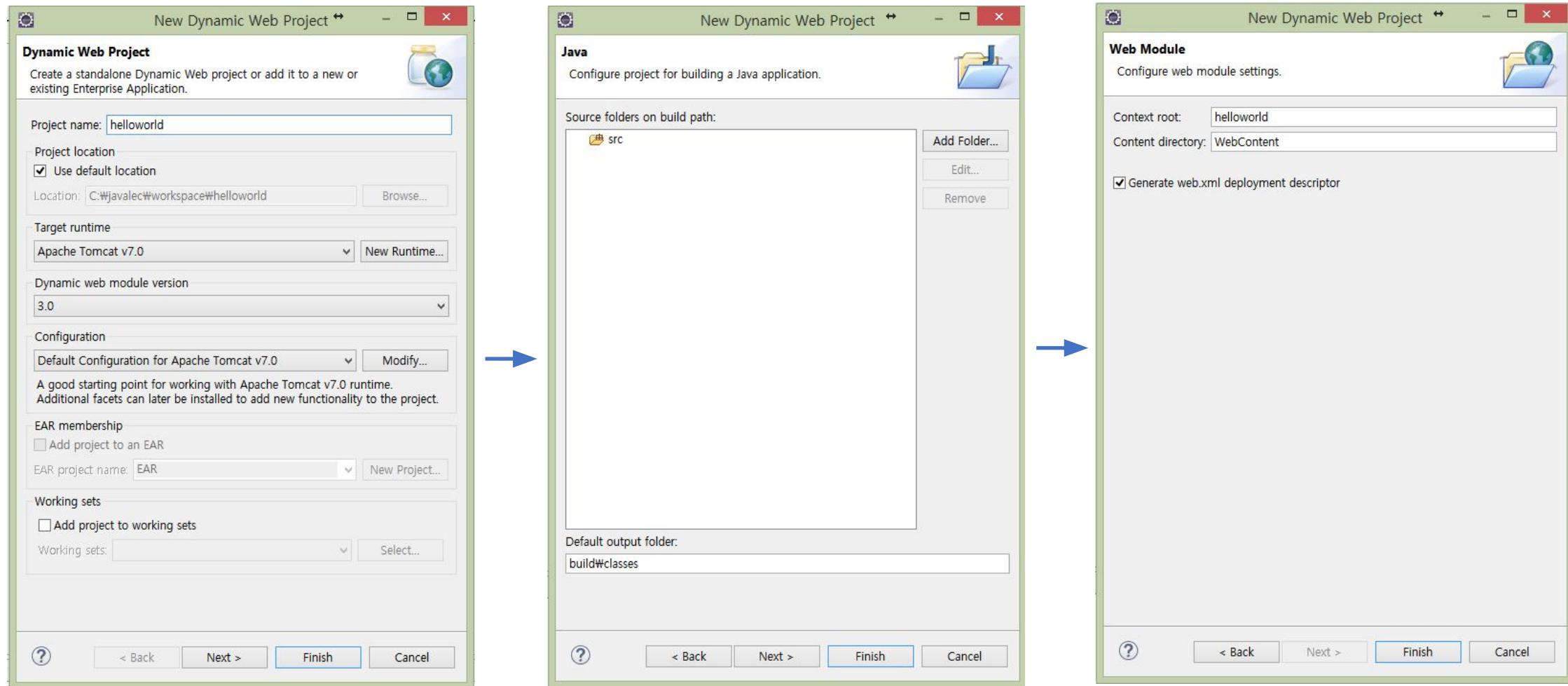
3-1. JSP문서 작성 하기

1. 프로젝트 생성



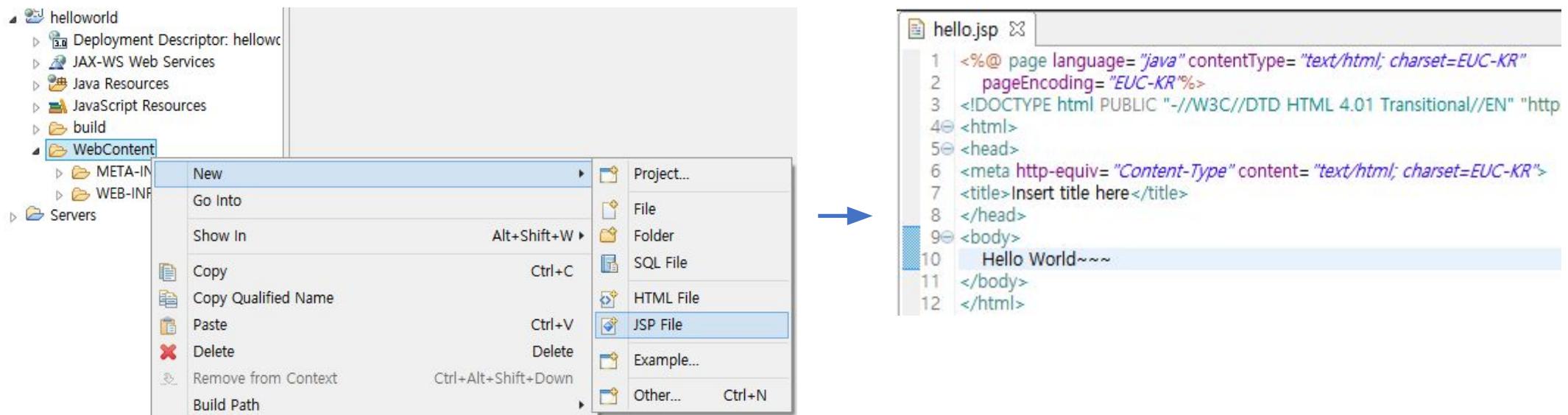
3-1. JSP문서 작성 하기

1. 프로젝트 생성



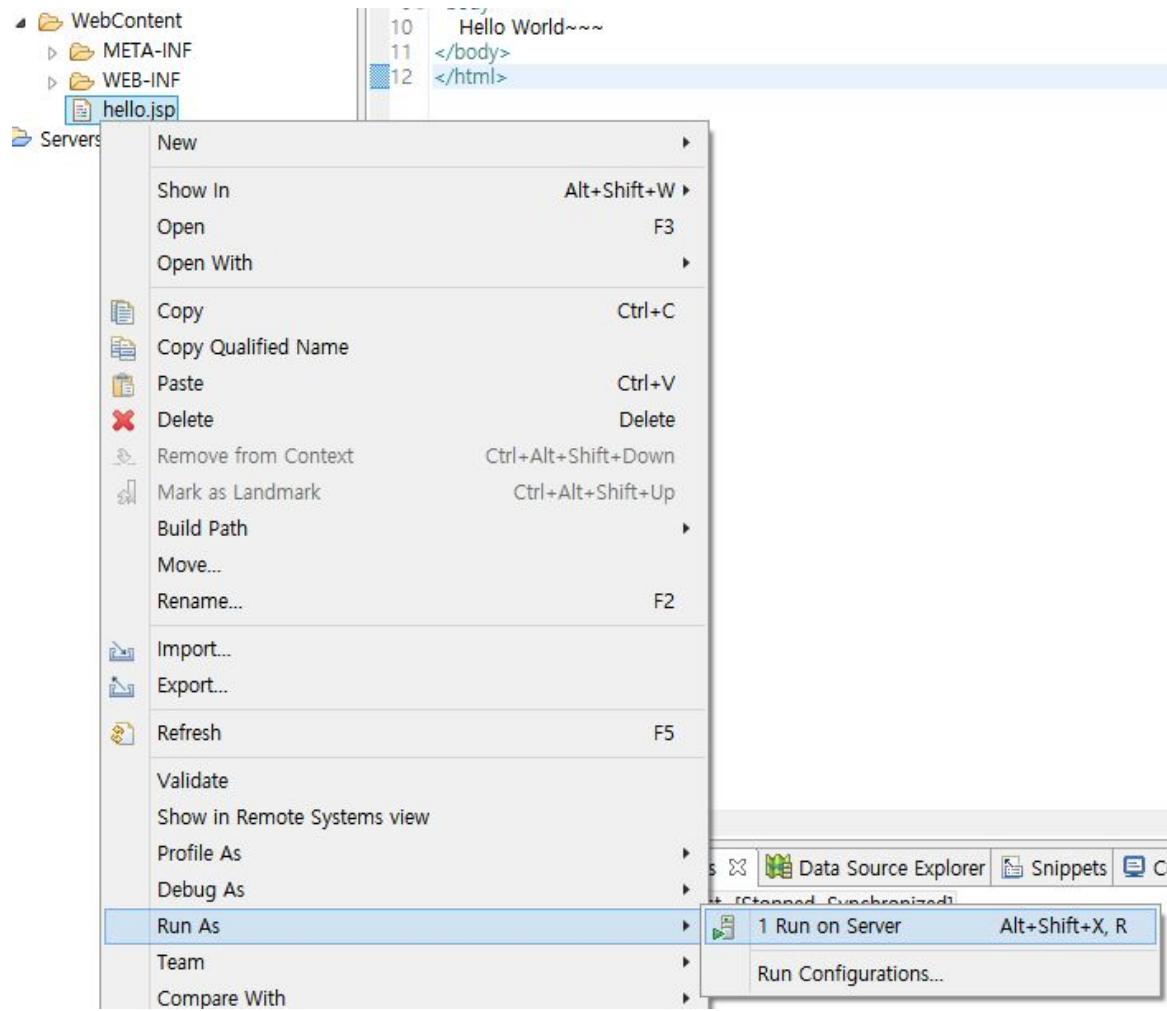
3-1. JSP문서 작성 하기

2. jsp파일 생성



3-1. JSP문서 작성 하기

3. jsp파일 실행



3-2. JSP 아키텍처

1) 아키텍처



4강. Servlet 맛보기

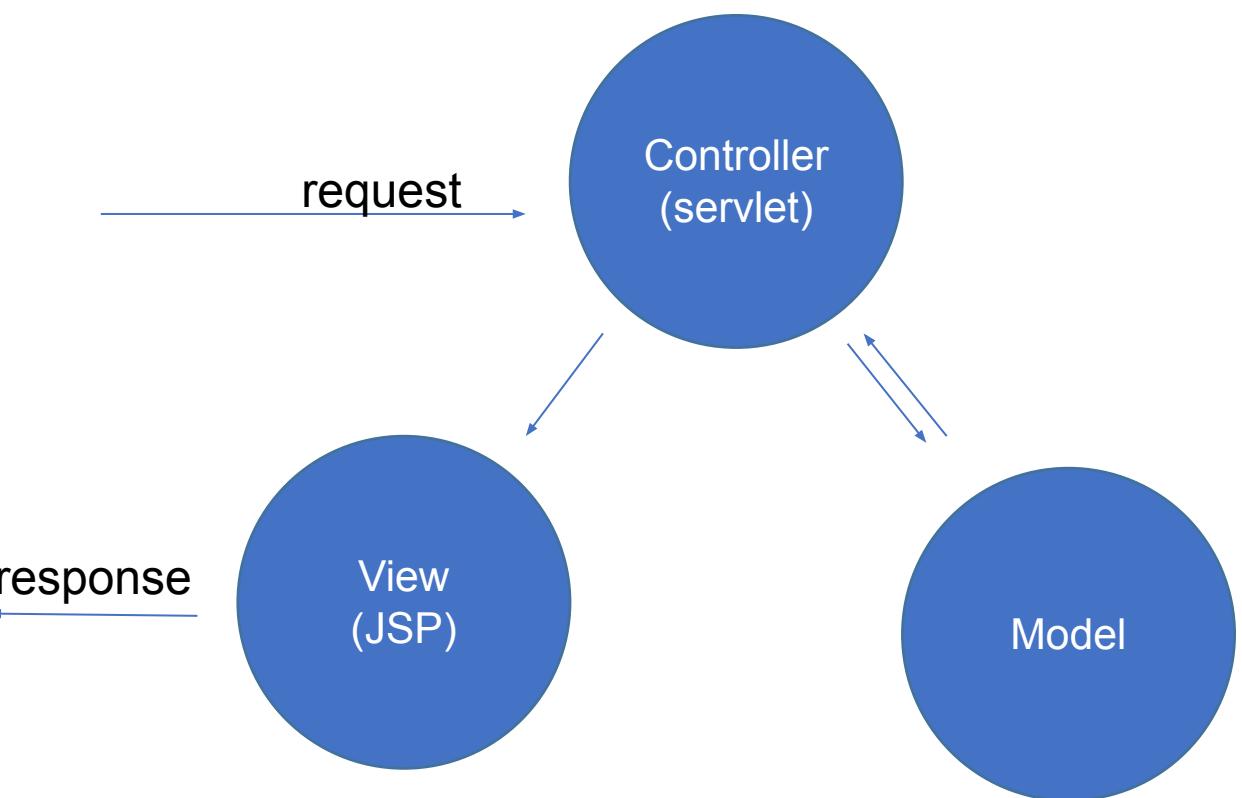
- Servlet 문서 작성 하기
- web.xml에 서블릿 맵핑
- 어노테이션을 이용한 서블릿 맵핑

4-1. Servlet 문서 작성 하기

Servlet특징

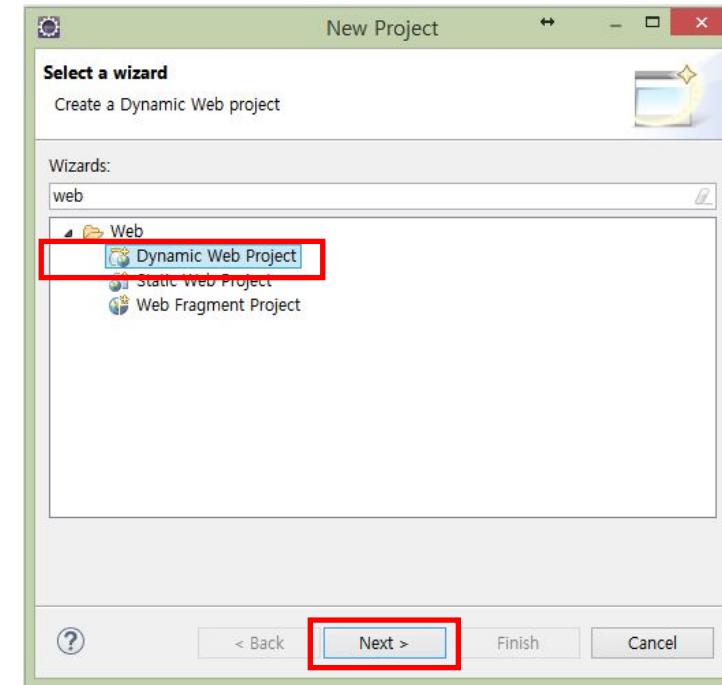
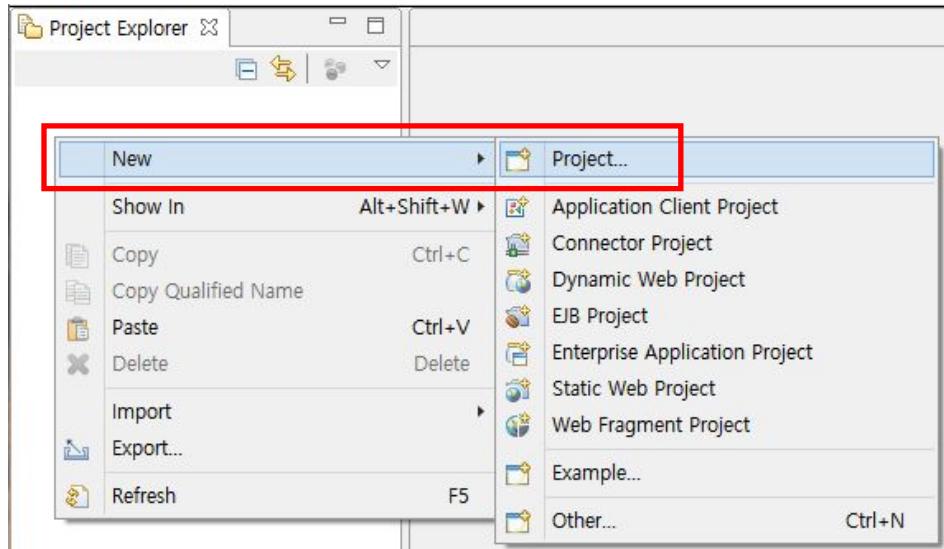
- 동적 웹애플리케이션 컴포넌트.
- .java 확장자.
- 클라이언트의 요청에 동적으로 작동하고, 응답은 html을 이용.
- java thread이용하여 동작.
- MVC패턴에서 Controller로 이용됨.

The screenshot shows the Naver News homepage. The URL in the address bar is news.naver.com/main/main.naver?mode=LSD&mid=shm&sid1=105. The page is filtered for the 'IT/과학' category. The main content area displays several news articles. One article from KT discusses their support for the presidential election candidate. Another article from KakaoTalk discusses its service. The sidebar on the left shows navigation links for various news categories like IT/과학, 모바일, 인터넷/SNS, and more.



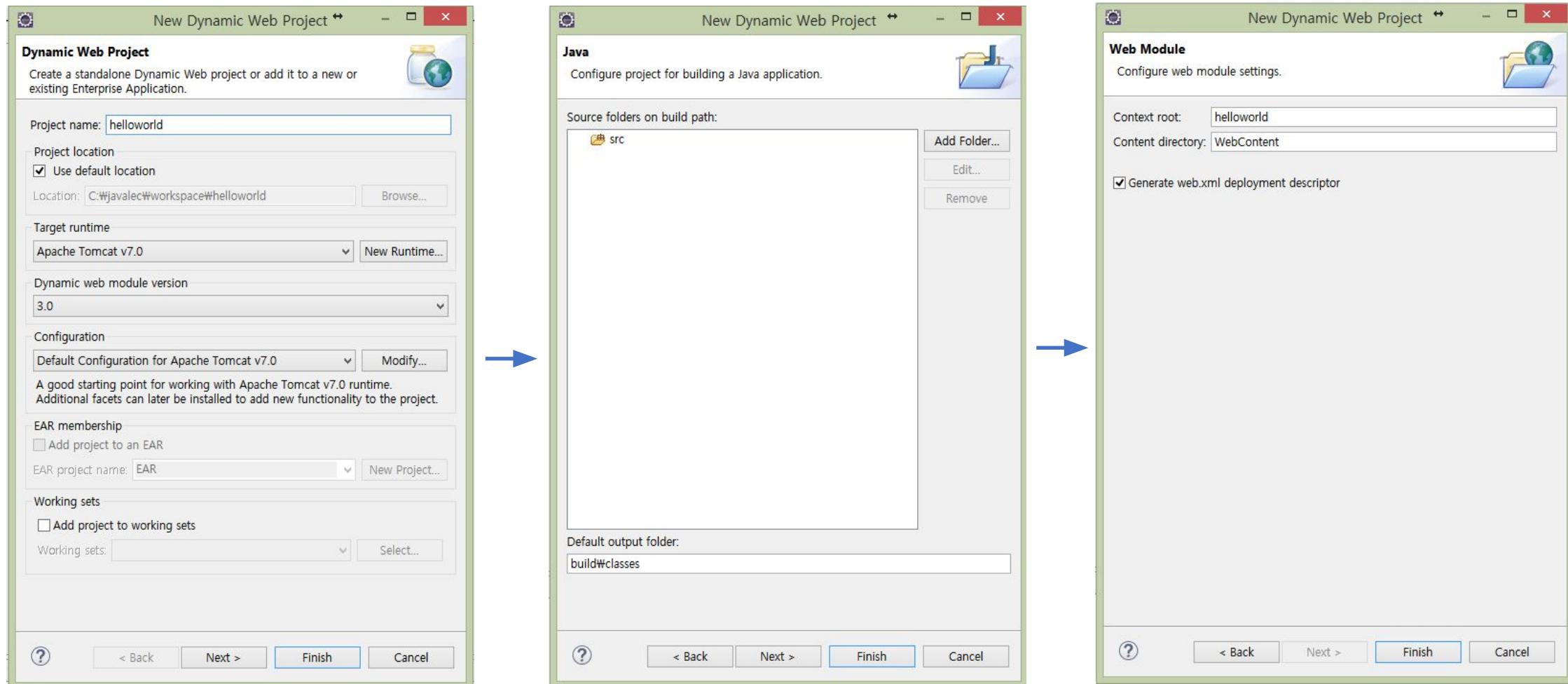
4-2. Servlet 문서 작성 하기

1. 프로젝트 생성



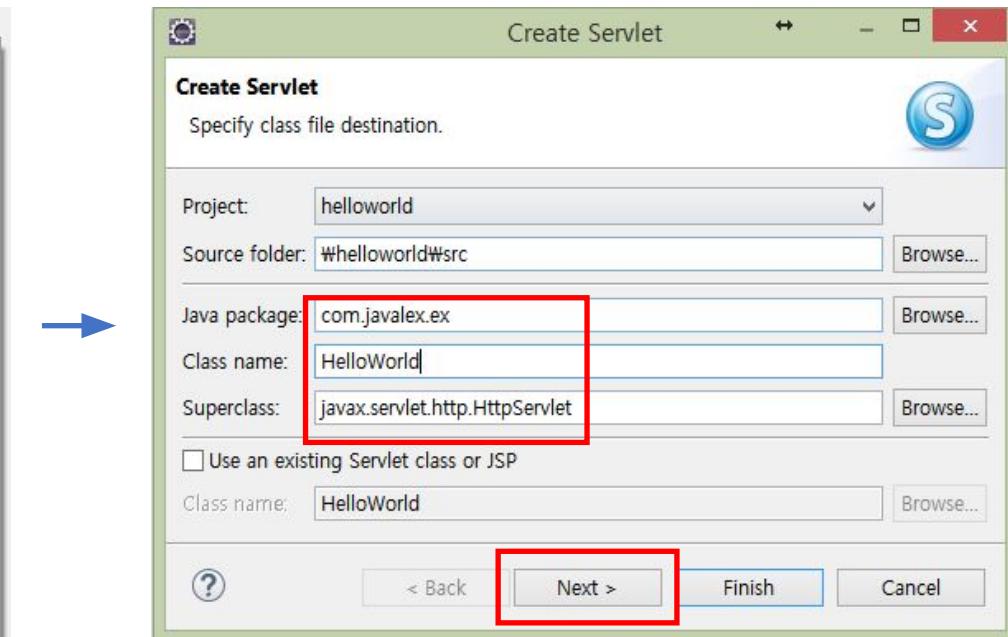
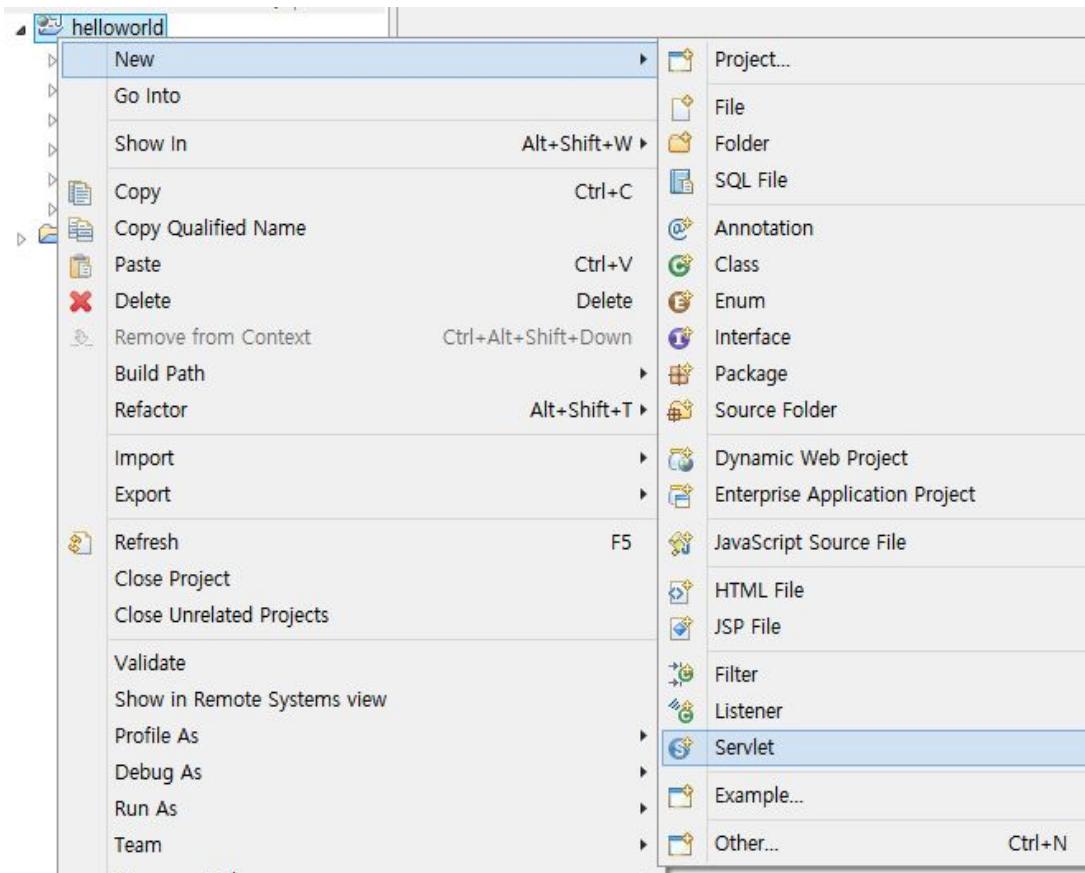
4-2. Servlet 문서 작성 하기

1. 프로젝트 생성



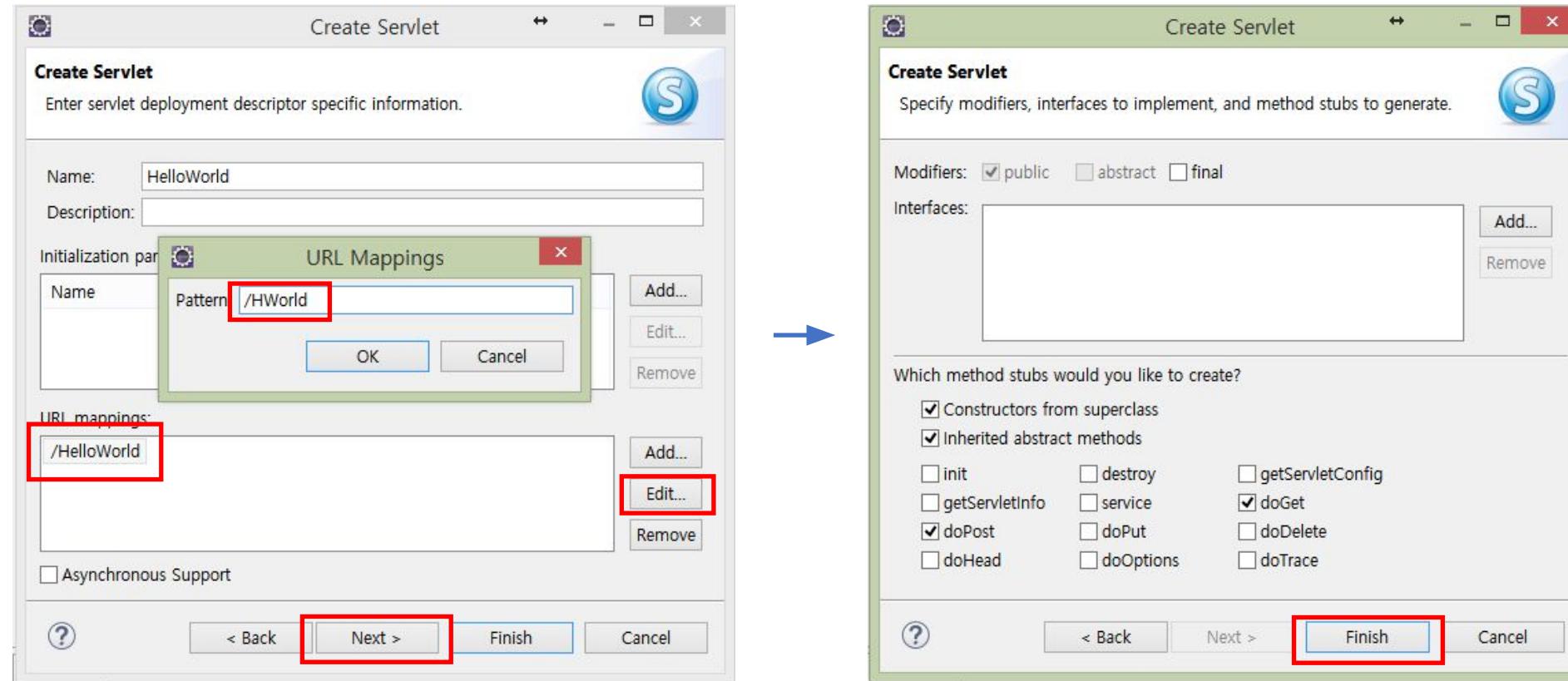
4-2. Servlet 문서 작성 하기

2. servlet파일 생성



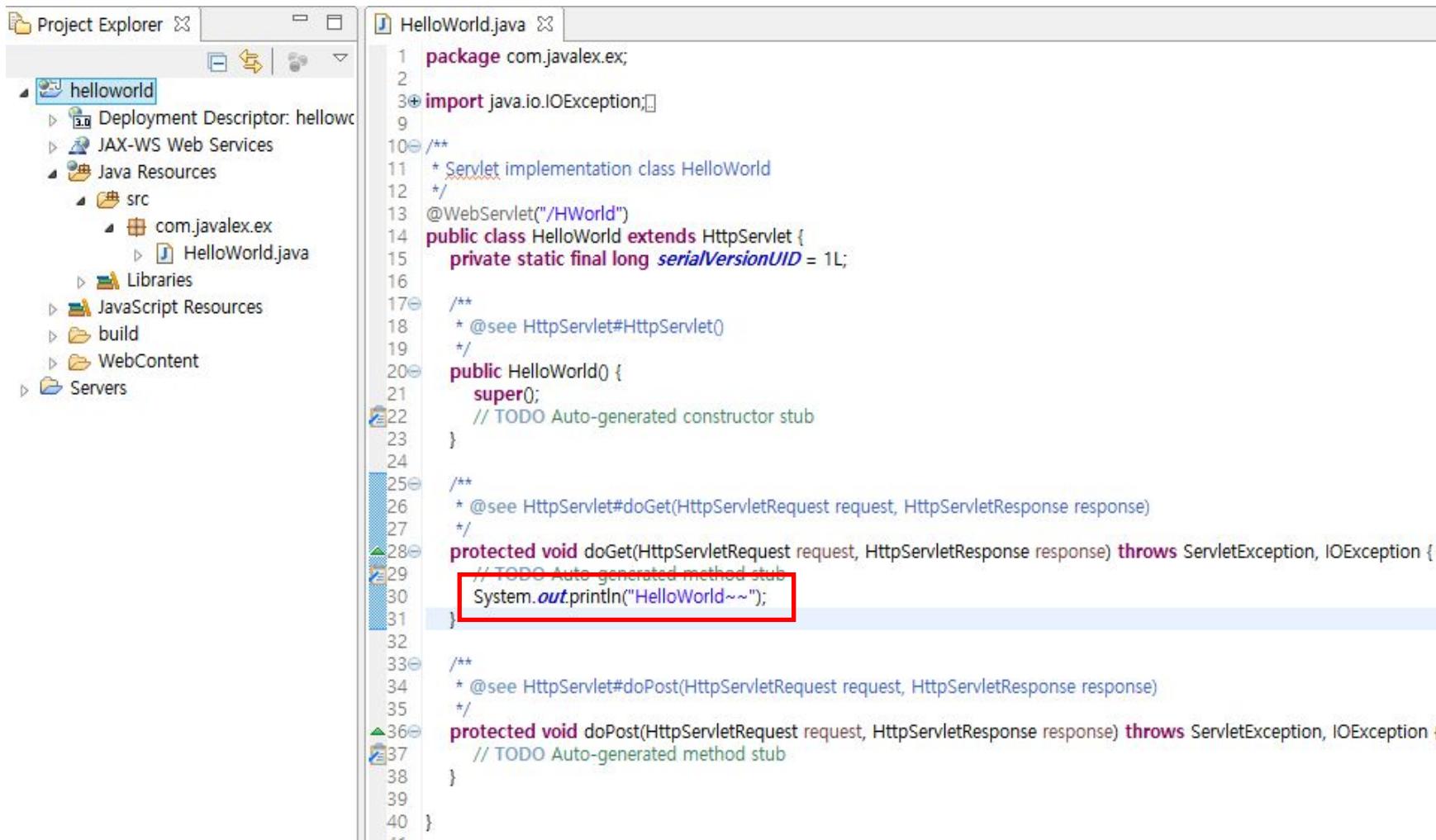
4-2. Servlet 문서 작성 하기

2. servlet파일 생성



4-2. Servlet 문서 작성 하기

2. servlet파일 생성



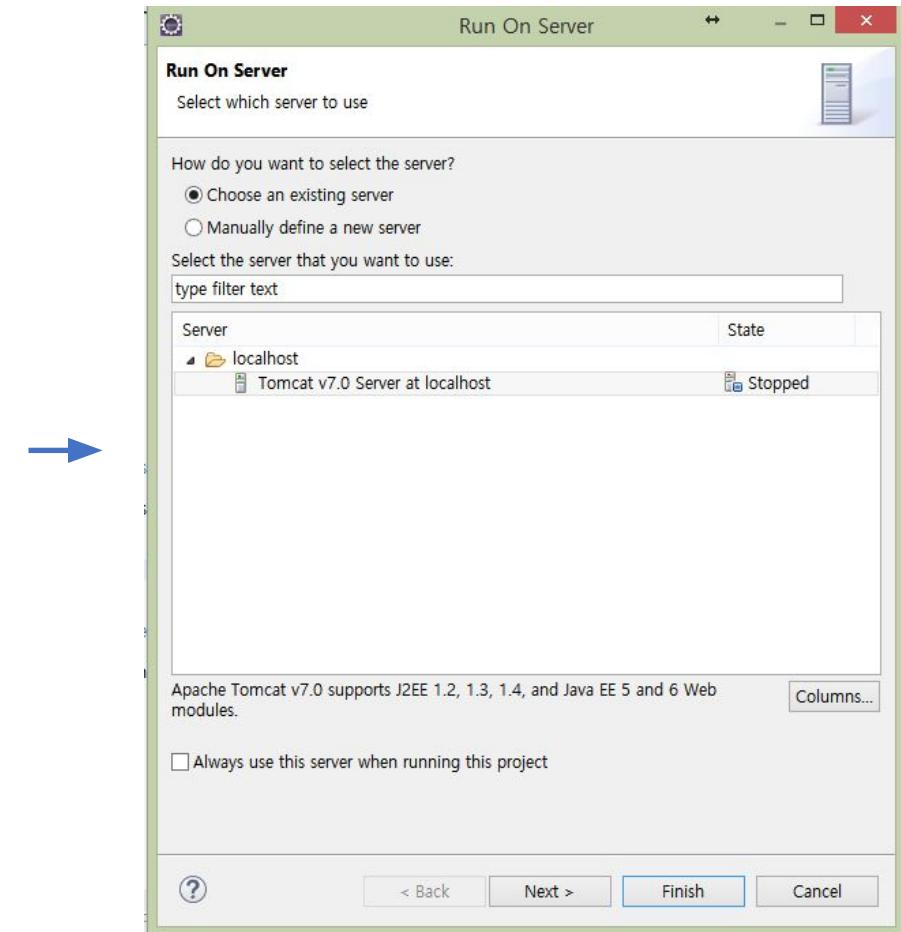
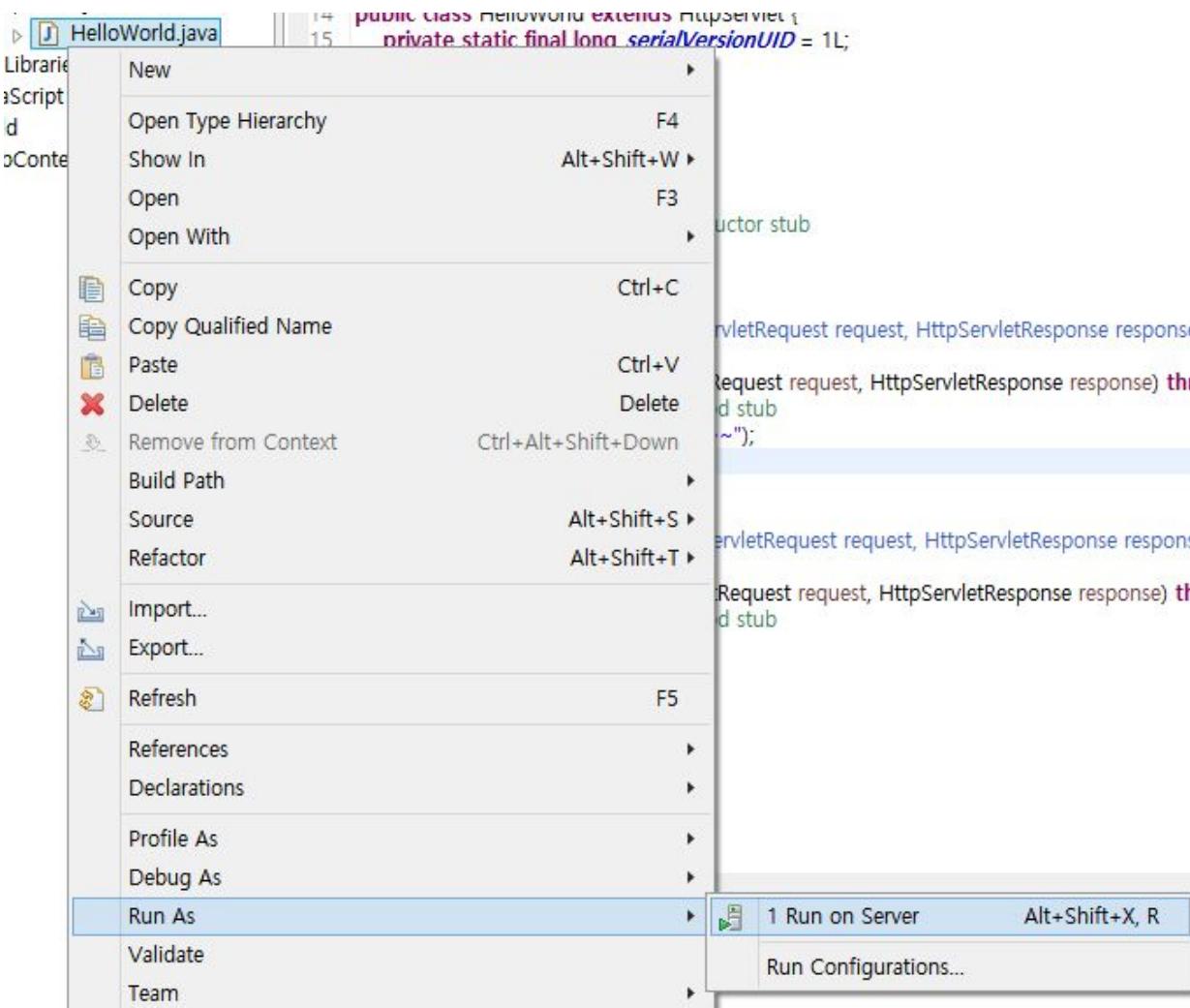
The screenshot shows the Eclipse IDE interface. On the left, the Project Explorer view displays a project named "helloworld" containing a Deployment Descriptor, JAX-WS Web Services, Java Resources (with a src folder containing com.javalex.ex and HelloWorld.java), Libraries, JavaScript Resources, build, WebContent, and Servers. The HelloWorld.java file is open in the main editor window. The code is a Java servlet implementation:

```
1 package com.javalex.ex;
2
3 import java.io.IOException;
4
5 /**
6  * Servlet implementation class HelloWorld
7  */
8 @WebServlet("/HWorld")
9 public class HelloWorld extends HttpServlet {
10     private static final long serialVersionUID = 1L;
11
12     /**
13      * @see HttpServlet#HttpServlet()
14      */
15     public HelloWorld() {
16         super();
17         // TODO Auto-generated constructor stub
18     }
19
20     /**
21      * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
22      */
23     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
24         // TODO Auto-generated method stub
25         System.out.println("HelloWorld~~");
26     }
27
28     /**
29      * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
30      */
31     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
32         // TODO Auto-generated method stub
33     }
34
35 }
36
37
38
39 }
40 }
```

A red box highlights the line of code: `System.out.println("HelloWorld~~");`.

4-2. Servlet 문서 작성 하기

3. Servlet 실행



4-2. Servlet 문서 작성 하기

3. Servlet 실행

A screenshot of the Eclipse IDE's Console view. The title bar includes tabs for "Markers", "Properties", "Servers", "Data Source Explorer", "Snippets", and "Console". The console output is as follows:

```
정보: Deploying web application directory C:\javalec\apache-tomcat-7.0.57\apache-tomcat-7.0
12월 30, 2014 4:13:55 오후 org.apache.catalina.startup.HostConfig deployDirectory
정보: Deployment of web application directory C:\javalec\apache-tomcat-7.0.57\apache-tomcat-7.0
12월 30, 2014 4:13:55 오후 org.apache.catalina.startup.HostConfig deployDirectory
정보: Deploying web application directory C:\javalec\apache-tomcat-7.0.57\apache-tomcat-7.0
12월 30, 2014 4:13:55 오후 org.apache.catalina.startup.HostConfig deployDirectory
정보: Deployment of web application directory C:\javalec\apache-tomcat-7.0.57\apache-tomcat-7.0
12월 30, 2014 4:13:55 오후 org.apache.catalina.startup.HostConfig deployDirectory
정보: Starting ProtocolHandler ["http-bio-8181"]
12월 30, 2014 4:13:55 오후 org.apache.coyote.AbstractProtocol start
정보: Starting ProtocolHandler ["ajp-bio-8009"]
12월 30, 2014 4:13:55 오후 org.apache.catalina.startup.Catalina start
정보: Server startup in 1529 ms
HelloWorld~~
```

The last line of the log, "HelloWorld~~", is highlighted in red.

4-2. web.xml에 서블릿 맵핑

너무 길고, 보안에 노출되어 있는 경로를 간단하게 맵핑

기존 경로 : http://localhost:8181/helloworld/servlet/com.javalec.ex.HelloWorld

URL맵핑 경로 : http://localhost:8181/helloworld/HWorld



4-2. web.xml에 서블릿 맵핑

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-in
<display-name>helloworld</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>
<servlet>
<servlet-name>helloworld</servlet-name>
<servlet-class>com.javalec.ex.HelloWorld</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>helloworld</servlet-name>
<url-pattern>/hw</url-pattern>
</servlet-mapping>
</web-app>
```

<servlet-name>

: 임의의 이름을 만들어 줍니다.

<servlet-class>

: 매핑 할 클래스 파일명을 패키지명을 포함하여 정확하게 입력 합니다.

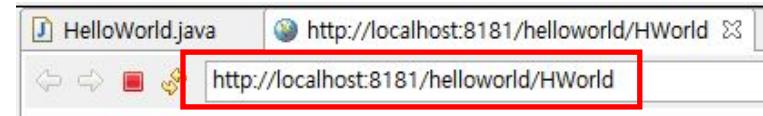
<url-pattern>

: servlet-class의 클래스를 매핑 할 임의의 이름을 입력 합니다. 주위 할 점은 '/'로 시작해야 합니다.



4-3. 어노테이션을 이용한 서블릿 맵핑

```
1 package com.javalec.ex;
2
3 import java.io.IOException;
4
5 /**
6  * Servlet implementation class HelloWorld
7 */
8
9 @WebServlet("/HWorld")
10 public class HelloWorld extends HttpServlet {
11     private static final long serialVersionUID = 1L;
12
13     /**
14      * @see HttpServlet#HttpServlet()
15      */
16
17     public HelloWorld() {
18         super();
19         // TODO Auto-generated constructor stub
20     }
21
22     /**
23      * @see HttpServlet#doGet(HttpServletRequest request)
24      */
25     protected void doGet(HttpServletRequest request) {
26         // TODO Auto-generated method stub
27         System.out.println("HelloWorld~~");
28     }
29 }
```



@WebServlet("HWorld")

: 맵핑명(HWorld)을 java소스에 직접 입력 합니다.

5강. Servlet 본격적으로 살펴보기-I

- 프로젝트 만들기
- doGet()
- doPost()
- 컨텍스트 패스(Context Path)

5-1. 프로젝트 만들기

Servlet은 JAVA언어를 사용하여 웹프로그램을 제작하는 것

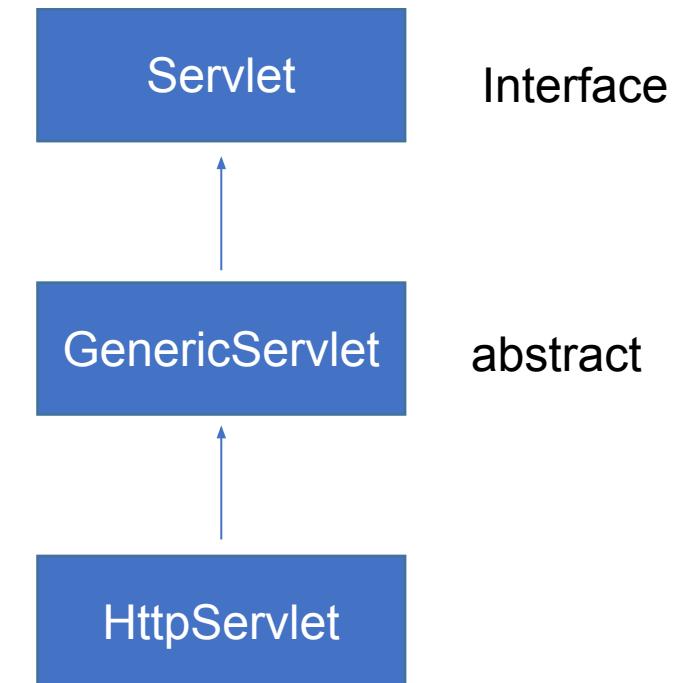
간단한 Servlet 프로젝트를 만들어 보면서 전체적인 구조(흐름) 이해

- Servlet클래스는 HttpServlet 클래스를 상속 받음.

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

}
/** *
 * @see HttpServlet#HttpServlet()
 */
```

HttpServlet
클래스를 상속



5-1. 프로젝트 만들기

- 요청처리객체 및 응답처리객체를 톰캣에서 받음.

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    System.out.println("HelloWorld~~~");

    response.setContentType("text/html");
    PrintWriter writer = response.getWriter(); ← 웹브라우저에 출력하기 위한 스트림

    writer.println("<html>");
    writer.println("<head>");
    writer.println("</head>");
    writer.println("<body>");
    writer.println("<h1>HelloWorld~~~</h1>");
    writer.println("</body>");
    writer.println("</html>");

    writer.close();
}
```

5-1. 프로젝트 만들기

- GET & POST 방식

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    System.out.println("HelloWorld~~~");

    response.setContentType("text/html");
    PrintWriter writer = response.getWriter();
    |
    writer.println("<html>");
    writer.println("<head>");
    writer.println("</head>");
    writer.println("<body>");
    writer.println("<h1>HelloWorld~~~</h1>");
    writer.println("</body>");
    writer.println("</html>");

    writer.close();
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    // TODO Auto-generated method stub
}
```

doGet() 호출

Form태그 method 속성값 = get

GET 방식 :

URL값으로 정보가 전송되어 보안에 약함.

POST 방식 :

header를 이용해 정보가 전송되어 보안에 강함.

Form태그 method 속성값 = post

doPost() 호출

5-2. doGet()

- html내 form태그의 method속성이 get일 경우 호출 됩니다.
- 웹브라우저의 주소창을 이용하여 servlet을 요청한 경우에도 호출 됩니다.

doGet메소드는 매개변수로 HttpServletRequest와 HttpServletResponse를 받습니다



5-2. doGet()

HttpServletResponse 객체의 setContentType() 메소드 호출하여 응답방식 결정

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    System.out.println("HelloWorld~~");

    response.setContentType("text/html; charset=euc-kr");
    PrintWriter writer = response.getWriter();

    writer.println("<html>");
    writer.println("<head>");
```

HttpServletResponse 객체의 getWriter() 메소드를 이용하여 출력 스트림을 얻습니다.

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    System.out.println("HelloWorld~~");

    response.setContentType("text/html; charset=euc-kr");
    PrintWriter writer = response.getWriter();

    writer.println("<html>");
    writer.println("<head>");
```

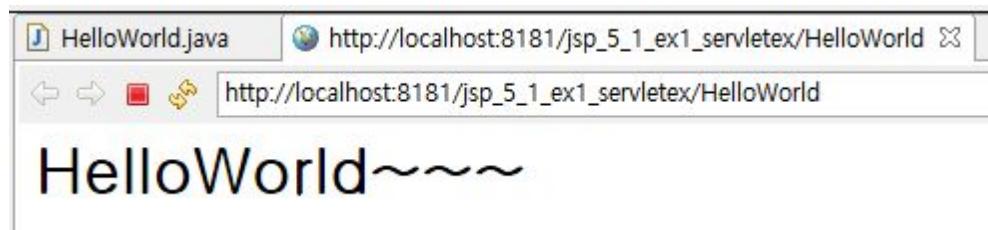
5-2. doGet()

출력스트림의 `println()` 메소드를 이용하여 출력하면, 웹브라우저에 출력 됩니다.

```
response.setContentType("text/html; charset=euc-kr");
PrintWriter writer = response.getWriter();

writer.println("<html>");
writer.println("<head>");
writer.println("</head>");
writer.println("<body>");
writer.println("<h1>HelloWorld~~~</h1>");
writer.println("</body>");
writer.println("</html>");

writer.close();
```



마지막에 출력객체 닫습니다.

```
writer.close();
```

5-3. doPost()

- html내 form태그의 method속성이 post일 경우 호출

```
post.html PostMethod.java http://localhost:8181/jsp_5_1_ex1_servletex/PostMethod
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="EUC-KR">
5 <title>Insert title here</title>
6 </head>
7 <body>
8
9 <form action="PostMethod" method="post">
10 <input type="submit" value="post">
11 </form>
12
13 </body>
14 </html>
```

HTML

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
// TODO auto-generated method stub
System.out.println("doPost");

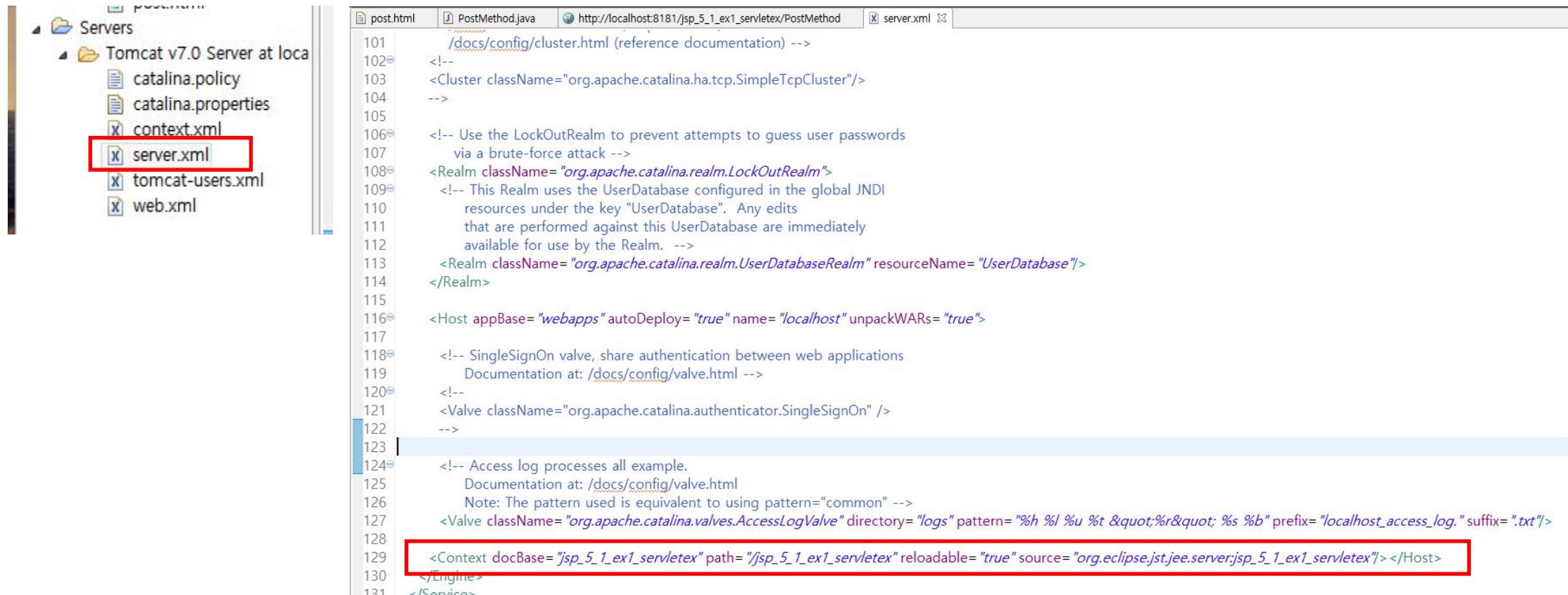
response.setContentType("text/html; charset=euc-kr");
PrintWriter writer = response.getWriter();
writer.println("<html>");
writer.println("<head>");
writer.println("</head>");
writer.println("<body>");
writer.println("<h1>POST 방식입니다. 따라서 doPost 메소드 호출 되었습니다.</h1>");
writer.println("</body>");
writer.println("</html>");

}
```

Servlet

5-4. 컨텍스트 패스(Context Path)

WAS(Web Application Server)에서 웹 어플리케이션을 구분하기 위한 path입니다.
이 클립스에서 프로젝트를 생성하면, 자동으로 server.xml에 추가 됩니다.



The screenshot shows the Eclipse IDE interface with the 'Servers' view on the left and the 'server.xml' configuration file on the right. The 'server.xml' file contains XML code for the Tomcat server. A specific section of the code, which defines a context path, is highlighted with a red box. This section is located between lines 129 and 130 of the XML code.

```
101     /docs/config/cluster.html (reference documentation) -->
102<!--
103<Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/>
104-->
105
106<!-- Use the LockOutRealm to prevent attempts to guess user passwords
107    via a brute-force attack -->
108<Realm className="org.apache.catalina.realm.LockOutRealm">
109    <!-- This Realm uses the UserDatabase configured in the global JNDI
110        resources under the key "UserDatabase". Any edits
111        that are performed against this UserDatabase are immediately
112        available for use by the Realm. -->
113    <Realm className="org.apache.catalina.realm.UserDatabaseRealm" resourceName="UserDatabase"/>
114</Realm>
115
116<Host appBase="webapps" autoDeploy="true" name="localhost" unpackWARs="true">
117
118    <!-- SingleSignOn valve, share authentication between web applications
119        Documentation at: /docs/config/valve.html -->
120    <!--
121    <Valve className="org.apache.catalina.authenticator.SingleSignOn" />
122    -->
123
124    <!-- Access log processes all example.
125        Documentation at: /docs/config/valve.html
126        Note: The pattern used is equivalent to using pattern="common" -->
127    <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs" pattern="%h %l %u %t \"%r\"%s %b" prefix="localhost_access_log." suffix=".txt">
128
129<Context docBase="jsp_5_1_ex1_servletex" path="/jsp_5_1_ex1_servletex" reloadable="true" source="org.eclipse.jst.jee.server:jsp_5_1_ex1_servletex"></Host>
130</Engine>
131</Service>
```

6강. Servlet 본격적으로 살펴보기-II

- Servlet 작동 순서
- Servlet 라이프사이클(생명주기)
- Servlet 선처리, 후처리

6-1. Servlet 작동 순서

클라이언트에서 **Servlet** 요청이 들어 오면 서버에서는 **Servlet** 컨테이너를 만들고, 요청이 있을 때마다 스레드가 생성 됩니다.

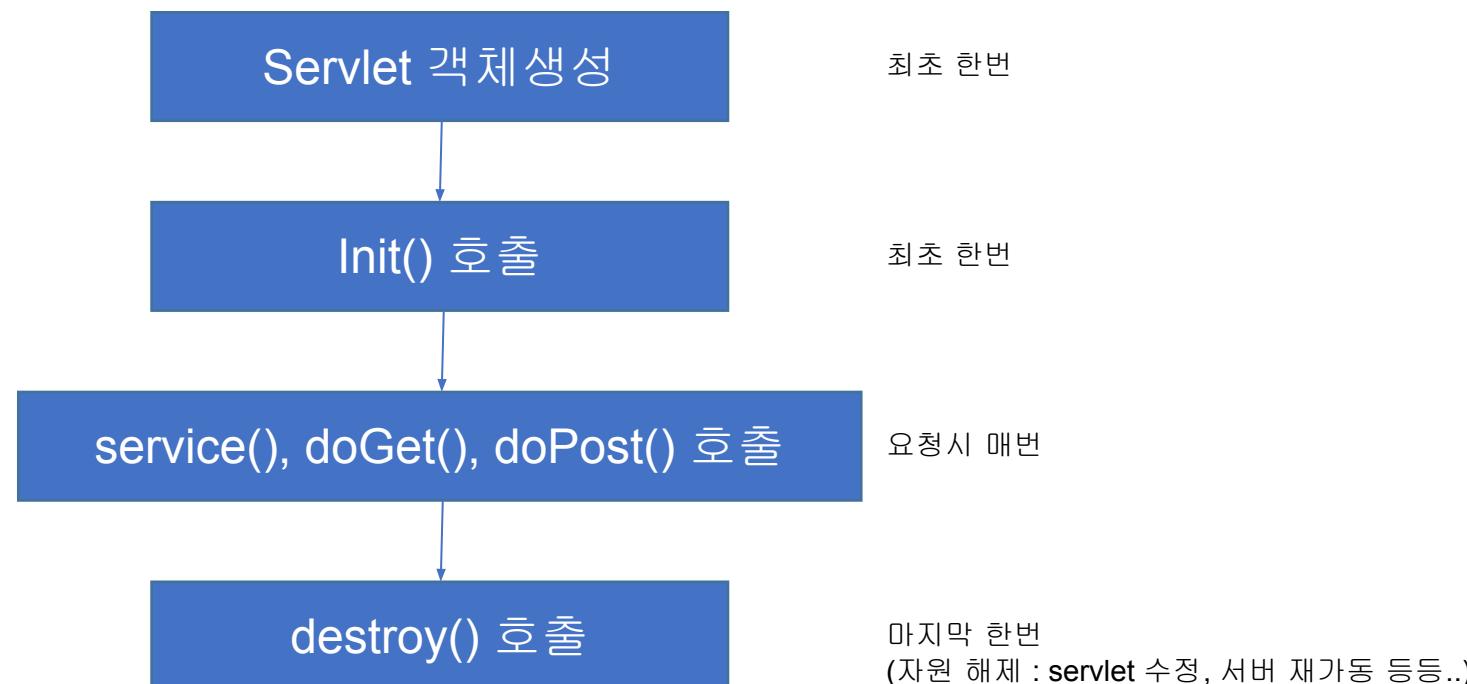


6-2. Servlet 라이프사이클(생명주기)

Servlet의 사용도가 높은 이유는 빠른 응답 속도 때문입니다.

Servlet은 최초 요청 시 객체가 만들어져 메모리에 로딩되고, 이후 요청 시에는 기존의 객체를 재활용하게 됩니다. 따라서 동작 속도가 빠릅니다.

Servlet의 라이프사이클을 살펴 봅니다.



6-3. Servlet 선처리, 후처리

Servlet의 라이프 사이클중 init()과 destroy()메소드와 관련하여 선처리(init()전)와 후처리(destroy()후) 작업이 가능 합니다.



7강. Servlet 본격적으로 살펴보기-III

- HTML Form 태그
- Servlet Parameter
- 한글처리

7-1. HTML form태그

Html의 form태그는 서버쪽으로 정보를 전달할 때 사용하는 태그

Html의 모든 태그를 학습할 필요는 없지만 웹프로그래머로서 Html언어를 초,중급 정도는 할 수 있어야

input

태그의 종류를 지정

속성(type, name, value)

- type : 태그 종류 지정(ex. text, password, submit, checkbox, radio, reset)
- name : input태그 이름
- value : name에 해당하는 값(ex. name = value)

type = text

일반적인 데이터를 입력하기 위해 사용합니다.

```
<input type="text" name="name" size="10">
```

type = password

로그인, 회원가입 페이지 등에서 비밀번호 입력하기 위해
사용합니다.

```
<input type="password" name="name" size="10">
```

7-1. HTML form태그

type = submit

form내의 데이터를 전송할 때 사용합니다.

```
<input type="submit" value="전송">
```

type = reset

Form내의 데이터를 초기화 할 때 사용합니다.

```
<input type="reset" value="초기화">
```

type = checkbox

데이터값을 여러 개 전송해야 할 때 사용합니다.

```
<input type="checkbox" name="hobby" value="read">독서  
<input type="checkbox" name="hobby" value="cook">요리  
<input type="checkbox" name="hobby" value="run">조깅  
<input type="checkbox" name="hobby" value="swim">수영  
<input type="checkbox" name="hobby" value="sleep">취침
```

7-1. HTML form태그

type = radio

checkbox와 달리 여러 개의 데이터 값 중 한 개의 값만을 전송할 때 사용합니다.

<input type="radio" name="major" value="kor">국어

<input type="radio" name="major" value="eng" checked="checked">영어

<input type="radio" name="major" value="mat" >수학

<input type="radio" name="major" value="des" >디자인

select

리스트형태의 데이터를 사용합니다.

```
<select name="protocol">
    <option value="http">http</option>
    <option value="ftp" selected="selected">ftp</option>
    <option value="smtp">smtp</option>
    <option value="pop">pop</option>
</select>
```

7-1. HTML form태그

form 태그

Input 태그들의 값을 서버로 전송하기 위한 정보를 담고 있습니다.

```
<form action="FormEx" method="post">
```



요청하는 컴포넌트 이름
(ex. join.jsp, info.html, HWorld)



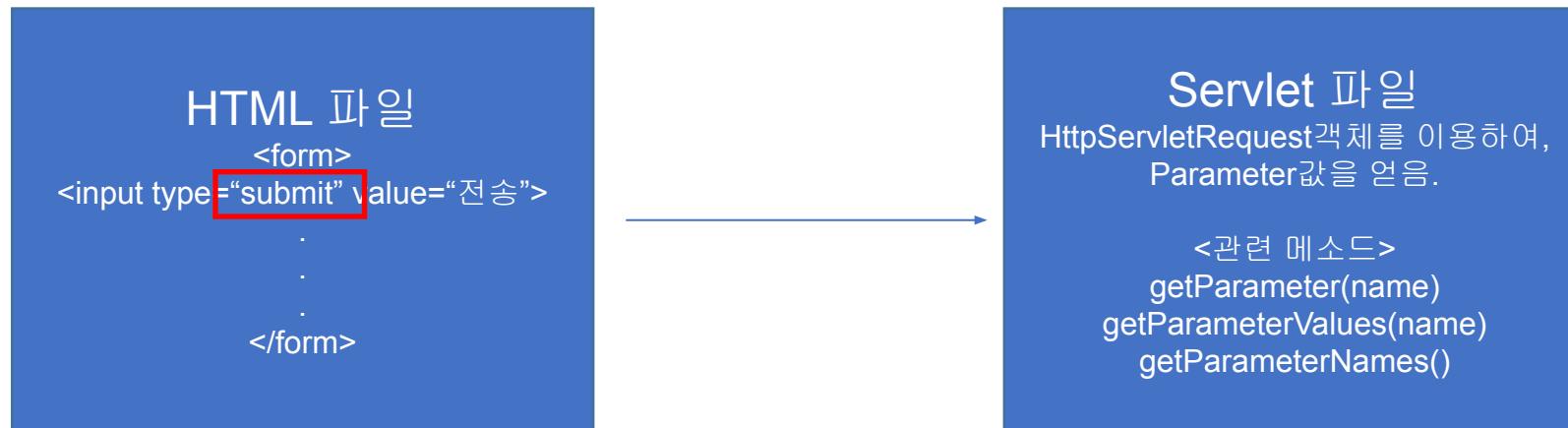
요청을 처리하는 방식
(ex. get, post)

Get : http://IP주소:port번호/컨텍스트/path/MemberJoin?id=“abcdefg”&name=“홍길동”

Post : http://IP주소:port번호/컨텍스트/path/MemberJoin

7-2. Servlet Parameter

Form태그의 submit 버튼을 클릭하여 데이터를 서버로 전송하면, 해당파일(Servlet)에서는 HttpServletRequest객체를 이용하여 Parameter값을 얻을 수 있다.



```
<form action="FormEx" method="post">  
    이름 : <input type="text" name="name" size="10"><br/>  
    아이디 : <input type="text" name="id" size="10"><br/>  
    비밀번호 : <input type="password" name="pw" size="10"><br/>
```

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) {  
    // TODO Auto-generated method stub  
    System.out.println("doPost");  
  
    String id = request.getParameter("id");  
    String pw = request.getParameter("pw");
```

아이디 : abcdefg
비밀번호 : 1234

```
    response.setContentType("text/html; charset=EUC-KR");  
    PrintWriter writer = response.getWriter();  
  
    writer.println("<html><head></head><body>");  
    writer.println("아이디 : " + id + "<br />");  
    writer.println("비밀번호 : " + pw + "<br />");
```

7-3. 한글처리

Tomcat 서버의 기본 문자 처리 방식은 IOS-8859-1 방식

개발자가 별도의 한글 인코딩을 하지 않으면 한글이 깨져 보이는 현상 발생

Get방식과 Post방식에 따라서 한글처리 방식에 차이가 있음

Get방식 요청

<server.xml 수정>

```
<Connector URIEncoding="EUC-KR" port="8181" cor
```

Post방식 요청

<request.setCharacterEncoding() 메소드 이용>

```
protected void doPost(HttpServletRequest request  
// TODO Auto-generated method stub  
System.out.println("doPost");
```

```
request.setCharacterEncoding("EUC-KR");
```

8강. Servlet 본격적으로 살펴보기-IV

- 서블릿 초기화 파라미터 : `ServletConfig`
- 데이터 공유 : `ServletContext`
- 웹어플리케이션 감시 : `ServletContextListener`

8-1. 서블릿 초기화 파라미터 : ServletConfig

특정 **Servlet**이 생성될 때 초기에 필요한 데이터들이 있다.

예를 들어 특정 경로 및 아이디 정보 등, 이러한 데이터들을 초기화 파라미터

web.xml에 기술하고 **Servlet**파일에서는 **ServletConfig** 클래스를 이용해서 접근(사용)

또한 초기화 파라미터를 **web.xml**이 아닌 **Servlet**파일에 직접 기술하는 방법도 살펴 봅니다.

web.xml파일에 초기화 파라미터(Initialization Parameter) 기술

Servlet 클래스 제작

```
<servlet>
  <servlet-name>ServletInitParam</servlet-name>
  <servlet-class>com.javalec.ex.ServletInitParam</servlet-class>
```

```
<init-param>
  <param-name>id</param-name>
  <param-value>abcdef</param-value>
</init-param>
<init-param>
  <param-name>pw</param-name>
  <param-value>1234</param-value>
</init-param>
<init-param>
  <param-name>path</param-name>
  <param-value>C:\WWWjavalec\WWWworkspace</param-value>
</init-param>
```

```
</servlet>
<servlet-mapping>
  <servlet-name>ServletInitParam</servlet-name>
  <url-pattern>/InitParam</url-pattern>
</servlet-mapping>
```

```
String id = getInitParameter("id");
String pw = getInitParameter("pw");
String path = getInitParameter("path");
```

web.xml파일에 초기화 파라미터 기술

ServletConfig 메소드 이용해서
데이터 불러오기

8-1. 서블릿 초기화 파라미터 : ServletConfig

Servlet 파일에 초기화 파라미터(Initialization Parameter) 기술

Servlet 클래스 제작

@WebInitParam에 초기화 파라미터 기술

ServletConfig 메소드 이용해서
데이터 불러오기

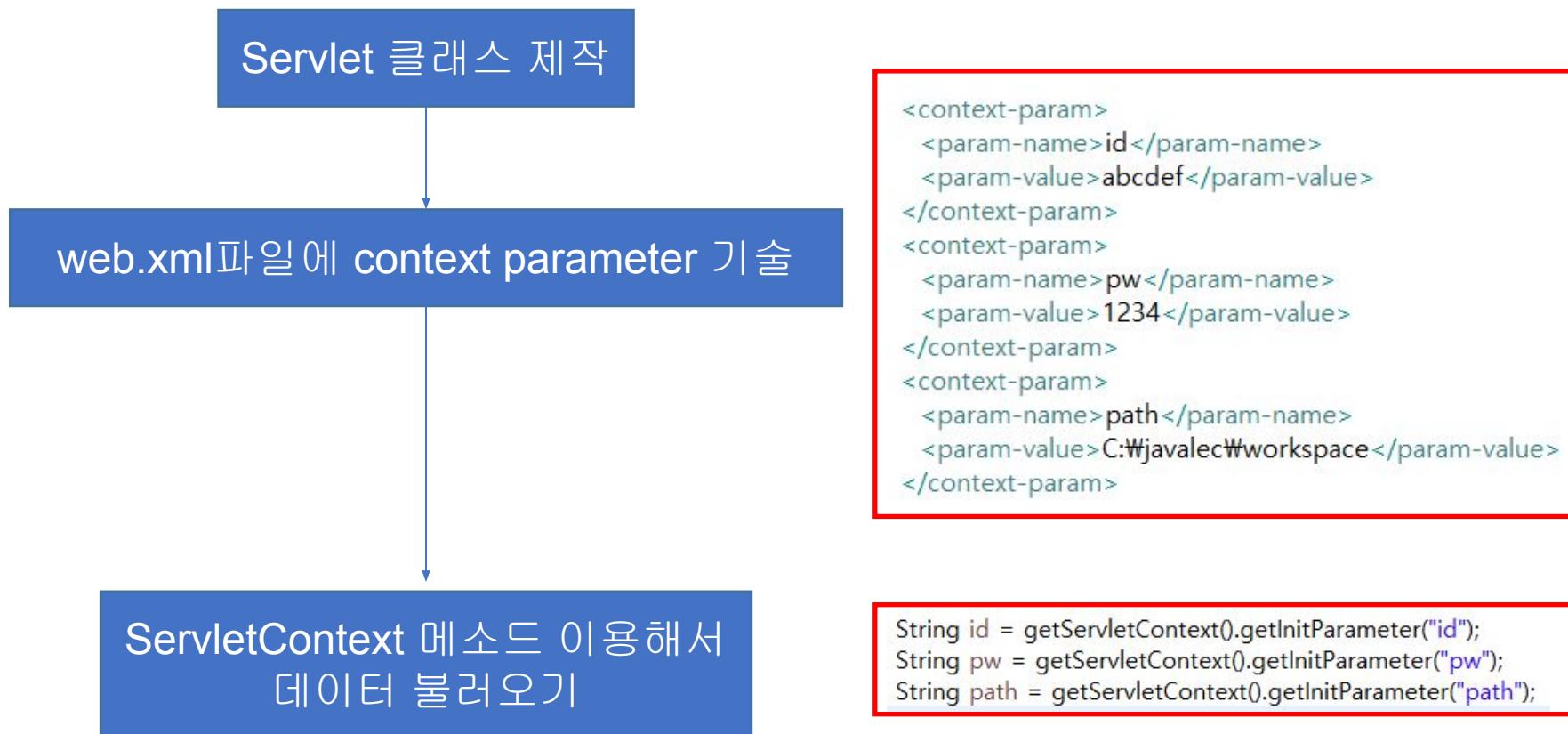
```
/\n@.WebServlet(urlPatterns={"/ServletInitParam"}, initParams={@WebInitParam(name="id", value="abcdef"), @WebInitParam(name="pw", value="1234567890"), @WebInitParam(name="path", value="C:/temp")})\npublic class ServletInitParam extends HttpServlet {\n    ...
```

```
String id = getInitParameter("id");\nString pw = getInitParameter("pw");\nString path = getInitParameter("path");
```

8-2. 데이터 공유 : ServletContext

여러 Servlet에서 특정 데이터를 공유해야 할 경우 context parameter를 이용해서 web.xml에 데이터를 기술하고, Servlet에서 공유하면서 사용

web.xml파일에 context parameter 기술



8-3. 웹어플리케이션 감시 : ServletContextListener

웹어플리케이션의 생명주기(LifeCycle)를 감시하는 리스너(Listener) : ServletContextListener

리스너의 해당 메소드가 웹 어플리케이션의 시작과 종료 시 호출 (**contextInitialized(), contextDestroyed()**)

web.xml파일에 리스너 클래스 기술

리스너 클래스 제작

```
public class ContextListenerEx implements ServletContextListener{  
  
    public ContextListenerEx() {  
        // TODO Auto-generated constructor stub  
    }  
  
    @Override  
    public void contextDestroyed(ServletContextEvent arg0) {  
        // TODO Auto-generated method stub  
        System.out.println("contextDestroyed");  
    }  
  
    @Override  
    public void contextInitialized(ServletContextEvent arg0) {  
        // TODO Auto-generated method stub  
        System.out.println("contextInitialized");  
    }  
}
```

web.xml파일에 리스너 클래스 기술

```
<listener>  
    <listener-class>com.javalec.ex.ContextListenerEx</listener-class>  
</listener>
```

8-3. 웹어플리케이션 감시 : ServletContextListener

리스너 클래스에 기술(@WebListener)



9강. JSP 본격적으로 살펴보기-I

- JSP 태그의 개념 이해
- JSP 동작 원리
- JSP 내부 객체

9-1. JSP 태그의 개념 이해

Servlet은 JAVA언어를 이용하여 문서를 작성하고, 출력객체를 이용하여 HTML코드를 삽입

JSP는 Servlet과 반대로 HTML코드에 JAVA언어를 삽입하여 동적 문서를 만듦

HTML코드안에 JAVA코드를 삽입하기 위해서는 태그를 이용

JSP태그 종류

지시자 : <%@ %> : 페이지 속성

주석 : <%-- --%>

선언 : <%! %> : 변수, 메소드 선언

표현식 : <%= %> : 결과값 출력

스크립트릿 : <% %> : JAVA 코드

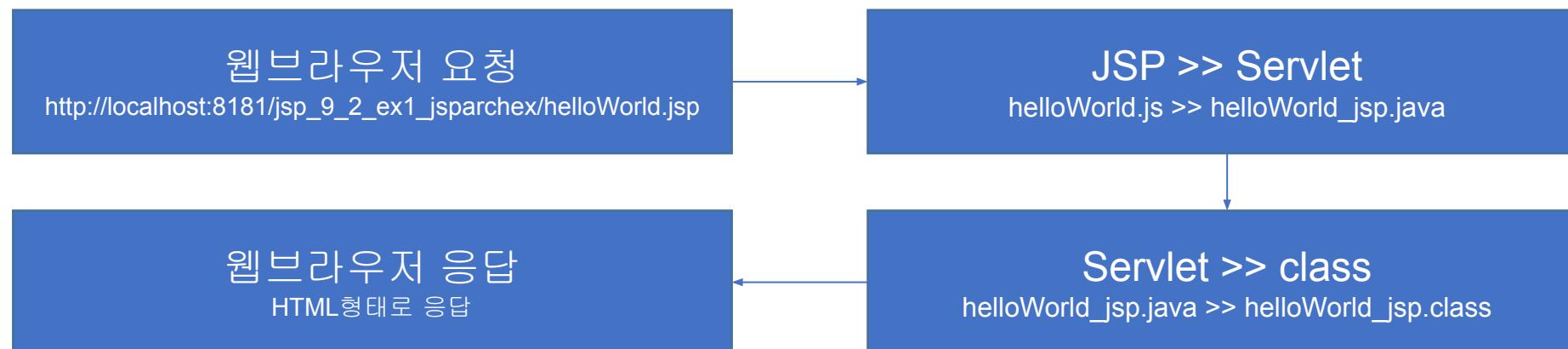
액션태그 : <jsp:action> </jsp:action> : 자바빈 연결

9-2. JSP 동작 원리

JSP가 요청되어 응답하기까지의 과정을 이해하면, 개발에 많은 도움이 됩니다.

클라이언트가 웹브라우저로 `helloWorld.jsp`를 요청하게 되면 JSP컨테이너가 JSP파일(.java)로 변환합니다.

그리고 Servlet파일(.java)은 컴파일 된 후 클래스 파일(.class)로 변환되고, 요청한 클라이언트한테 html파일 형태로 응답 됩니다.



□ 이름	수정
META-INF	201
WEB-INF	201
helloWorld.jsp	201

□ 이름	수정한 날짜	유형	크기
helloWorld_jsp.class	2015-01-02 오후 3:05	CLASS 파일	5KB
helloWorld_jsp.java	2015-01-02 오후 3:05	JAVA 파일	4KB

9-3. JSP 내부 객체

개발자가 객체를 생성하지 않고 바로 사용할 수 있는 객체가 내부객체입니다.

JSP에서 제공되는 내부객체는 JSP컨테이너에 의해 **Servlet**으로 변화될 때 자동으로 객체가 생성됩니다.

내부 객체 종류

입출력 객체 : **request, response, out**

서블릿 객체 : **page, config**

세션 객체 : **session**

예외 객체 : **exception**

10강. JSP 본격적으로 살펴보기-II

- 스크립트릿(스크립트태그), 선언, 표현식
- 지시자
- 주석

10-1. 스크립트릿, 선언, 표현식

JSP문서안에 JAVA언어를 넣기 위한 방식들 입니다. 실제 개발에서 많이 쓰이므로 잘 익혀 둡니다.

<% ... %> 사용

JSP 페이지가 서블릿 프로그램에서 서블릿 클래스로 변환할 때

JSP 컨테이너가 자바 코드가 삽입되어 있는 스크립트 태그를 처리하고 나머지는 HTML 코드나 일반 텍스트로 간주

스크립트 태그	형식	설명
선언문(declaration)	<%! ... %>	자바 변수나 메소드를 정의하는 데 사용합니다.
스크립틀릿(scriptlet)	<% ... %>	자바 로직 코드를 작성하는 데 사용합니다.
표현문(expression)	<%= ... %>	변수, 계산식, 메소드 호출 결과를 문자열 형태로 출력하는 데 사용합니다.

```
<html>
<head>
<title>Scripting Tag</title>
</head>
<body>
    <h2>Scripting Tag</h2>
```

```
    <%! int count = 3;
```

```
        String makeItLower(String data) {
            return data.toLowerCase();
        } %>
```

```
<%
    for (int i = 1; i <= count; i++) {
        out.println("Java Server Pages " + i + ".<br>");
    }
%>
```

```
<%=makeItLower("Hello World")%>
</body>
</html>
```

scripting.jsp

Scripting Tag

Java Server Pages 1.
Java Server Pages 2.
Java Server Pages 3.
hello world

- ① 선언문 태그를 사용하여 자바 변수와 메소드 정의

- ② 스크립틀릿 태그로 자바 로직 코드 작성

- ③ 표현문 태그로 선언문의 메소드를 호출하여 문자열 형태로 출력

10-1. 스크립트릿, 선언, 표현식

스크립트릿(scriptlet) : <% java 코드 기술 %>

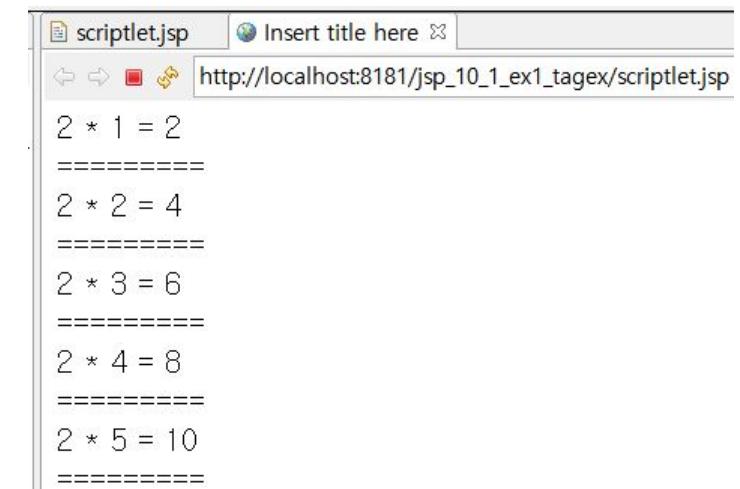
JSP페이지에서 JAVA언어를 사용하기 위한 요소 중 가장 많이 사용되는 요소 입니다.

우리가 알고 있는 거의 모든 JAVA코드를 사용할 수 있습니다.

```
<body>
<%
int i = 0;
while(true){
    i++;
    out.println("2 * " + i + " = " + (2 * i) + "<br />");
%>
=====
<br />
<%
    if(i >= 9) break;
}
%>

</body>
```

```
<body>
=====
2 * 1 = 2<br />
=====
2 * 2 = 4<br />
=====
2 * 3 = 6<br />
=====
2 * 4 = 8<br />
=====
2 * 5 = 10<br />
```



10-1. 스크립트릿, 선언, 표현식

선언(declaration) : <%! java 코드 기술 %>

JSP페이지 내에서 사용되는 변수 또는 메소드를 선언할 때 사용 합니다.

여기서 선언된 변수 및 메소드는 전역의 의미로 사용됩니다.

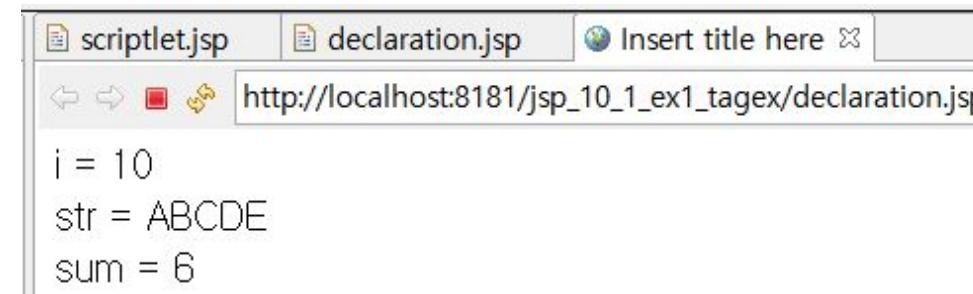
```
<body>
```

```
<%!
    int i = 10;
    String str = "ABCDE";
%>

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

<%
    out.println("i = " + i + "<br />");
    out.println("str = " + str + "<br />");
    out.println("sum = " + sum(1,5) + "<br />");
%>

</body>
```



선언문 태그

- 변수나 메소드를 선언
 - 변수 - 전역변수로 사용
 - 메소드 - 전역 메소드로 사용

```
<%! 자바 코드; %>
```

각 행이 세미콜론으로 끝나야 함

전역 변수

```
<html>
<head>
<title>Scripting Tag</title>
</head>
<%-- 선언문 태그 [메소드] --%>
<%! int count = 0; %>
<body>
    Page Count is:
    <%-- 스크립틀릿 태그 --%>
    <%
        out.println(++count);
    %>
</body>
</html>
```

Page Count is: 1

① 전역변수 count를 0으로 초기화

② 전역변수 count를 1 증가시킴

선언문 태그의 기능과 사용법

```
public class hello_jsp extends HttpServlet {  
    int count=0; 1  
    public void _jspService(HttpServletRequest request, HttpServletResponse  
    response) throws IOException, ServletException {  
        PrintWriter out = response.getWriter();  
        response.setContentType("text/html");  
        out.write("<html><body>");  
  
        out.write("Page count is:");  
        out.print(++count); 2  
        out.write("</body></html>");  
    }  
}
```

전역 메소드

```
<html>
<head>
<title>Scripting Tag</title>
</head>
<body>
    Page Count is:
    <%-- 스크립틀릿 태그 --%>
    <%
        out.print(myMethod(0));
    %>
    <%-- 선언문 태그 [메소드] --%>
    <%! public int myMethod(int count) { %-- 전역 메소드 myMethod() 설정
        return ++count;
    } %>
</body>
</html>
```

전역 메소드 myMethod() 호출

전역 메소드 myMethod() 설정

10-1. 스크립트릿, 선언, 표현식

표현식(expression) : <%= java 코드 기술 %>

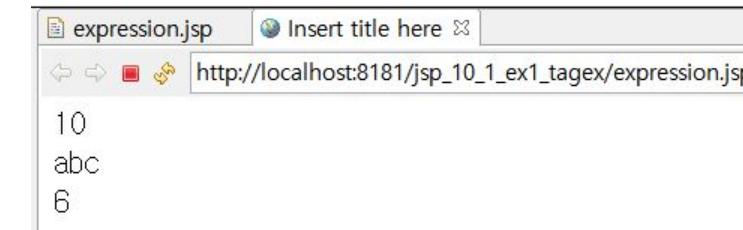
JSP페이지 내에서 사용되는 변수의 값 또는 메소드 호출 결과값을 출력하기 위해 사용 됩니다.
결과값은 String 타입이며, ';'를 사용 할 수 없습니다.

```
<body>
<%!
    int i = 10;
    String str = "abc";

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

<%= i %><br />
<%= str %><br />
<%= sum(1, 5) %>

</body>
```



스크립틀릿 태그

자바 코드로 이루어진 로직 부분을 표현

out 객체를 사용하지 않고도 쉽게 HTML 응답을 만들어냄

<% 자바 코드; %>

각 행이 세미콜론으로 끝나야 함

```
<html>
<head>

<title>Scripting Tag</title>
</head>
<%
    int count = 0;
%>
<body>
    Page Count is
    <%
        out.println(++count);
%>
</body>
</html>
```

Page Count is 1

지역변수 count를 0으로 초기화

지역변수 count를 1 증가시킴

선언문 태그	스크립틀릿 태그
변수뿐만 아니라 메소드를 선언할 수 있습니다.	스크립틀릿 태그는 메소드 없이 변수만을 선언할 수 있습니다.
서블릿 프로그램으로 변환될 때 <code>_jspService()</code> 메소드 외부에 배치됩니다.	서블릿 프로그램으로 변환될 때 <code>_jspService()</code> 메소드 내부에 배치됩니다.

예제 2-5 스크립틀릿 태그에 0부터 10까지의 짝수 출력하기

```
01 <html>
02 <head>
03 <title>Scripting tag</title>
04 </head>
05 <body>
06     <%
07         for (int i = 0; i <= 10; i++) {
08             if (i % 2 == 0)
09                 out.println(i + "<br>");
10         }
11     %>
12 </body>
13 </html>
```



표현문 태그

- 웹 브라우저에 출력할 부분을 표현
- 표현문 태그에 숫자, 문자, 불린(Boolean) 등의 기본 데이터 타입과 자바 객체 타입도 사용 가능

<%= 자바 코드 %>

각 행을 세미콜론으로 종료할 수 없음

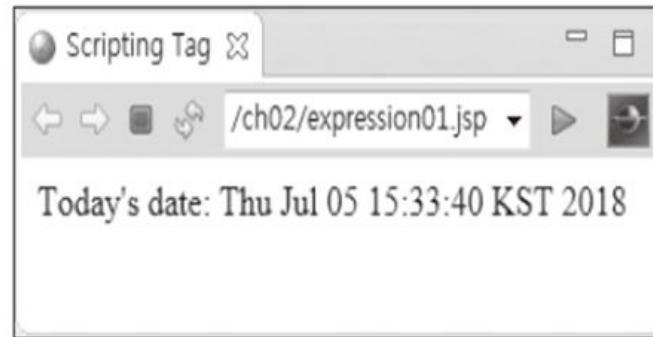
```
<html>
<head>
<title>Scripting Tag</title>
</head>
<%
    int count = 0;
%
<body>
    Page Count is
    <%= ++count %>
</body>
</html>
```

Page Count is 1

지역변수 count를 1 증가시킴

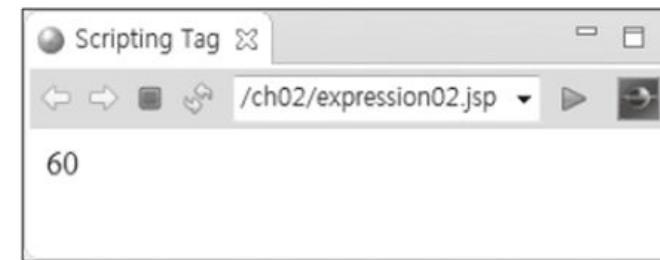
예제 2-6 표현문 태그로 현재 날짜 출력하기

```
01 <html>
02 <head>
03 <title>Scripting Tag</title>
04 </head>
05 <body>
06     <p>
07         Today's date:
08         <%= new java.util.Date() %></p>
09 </body>
10 </html>
```



예제 2-7 표현문 태그로 연산 결과 출력하기

```
01 <html>
02 <head>
03 <title>Scripting Tag</title>
04 </head>
05 <body>
06     <%
07         int a = 10;
08         int b = 20;
09         int c = 30;
10     %>
11     <%= a + b + c %>
12 </body>
13 </html>
```



```
<html>
<head>
<title>A Comment Test</title></head>
<body>
    <h2>A Test of Comments</h2>
    <%-- This comment will not be visible in the page source --%>
</body>
</html>
```


10-2. 지시자(디렉티브 태그)

JSP페이지의 전체적인 속성을 지정할 때 사용 합니다.

page, include, taglib 가 있으며, <%@ 속성 %> 형태로 사용 됩니다.

JSP 페이지를 어떻게 처리할 것인지를 설정하는 태그

JSP 페이지가 서블릿 프로그램에서 서블릿 클래스로 변환할 때

JSP 페이지와 관련된 정보를 JSP컨테이너에 지시하는 메시지

디렉티브 태그	형식	설명
page	<%@ page ... %>	JSP 페이지에 대한 정보를 설정합니다.
include	<%@ include ... %>	JSP 페이지의 특정 영역에 다른 문서를 포함합니다.
taglib	<%@ taglib ... %>	JSP 페이지에서 사용할 태그 라이브러리를 설정합니다.

page 지시자

- 페이지의 속성을 지정할 때 사용 합니다. 주로 사용되는 언어 지정 및 import문을 많이 사용 합니다.
- 현재 JSP 페이지에 대한 정보를 설정하는 태그
- JSP 페이지의 어디에서든 선언할 수 있지만 일반적으로 JSP 페이지의 최상단에 선언하는 것을 권장

```
page.jsp
1 <%@page import="java.util.Arrays"%>
2 <%@ page language="java" contentType="text/html; charset=EUC-KR"
3   pageEncoding="EUC-KR"%>
```

<%@ page 속성1="값1" [속성2="값2" …] %>

<%@ 사이에 공백이 없어야 함

page 디렉티브 태그의 속성

속성	설명	기본 값
language	현재 JSP 페이지가 사용할 프로그래밍 언어를 설정합니다.	java
contentType	현재 JSP 페이지가 생성할 문서의 콘텐츠 유형을 설정합니다.	text/html
pageEncoding	현재 JSP 페이지의 문자 인코딩을 설정합니다.	ISO-8859-1
import	현재 JSP 페이지가 사용할 자바 클래스를 설정합니다.	
session	현재 JSP 페이지의 세션 사용 여부를 설정합니다.	true
buffer	현재 JSP 페이지의 출력 버퍼 크기를 설정합니다.	8KB
autoFlush	출력 버퍼의 동작 제어를 설정합니다.	true
isThreadSafe	현재 JSP 페이지의 멀티스레드 허용 여부를 설정합니다.	true
info	현재 JSP 페이지에 대한 설명을 설정합니다.	
errorPage	현재 JSP 페이지에 오류가 발생했을 때 보여줄 오류 페이지를 설정합니다.	
isErrorPage	현재 JSP 페이지가 오류 페이지인지 여부를 설정합니다.	false
isELIgnored	현재 JSP 페이지의 표현 언어(EL) 지원 여부를 설정합니다.	false
isScriptingEnabled	현재 JSP 페이지의 스크립트 태그 사용 여부를 설정합니다.	

10-2. 지시자

include 지시자

현재 페이지내에 다른 페이지를 삽입할 때 사용

file속성을 이용

(jsp_10_2_ex1_directiveex)

```
<h1> include.jsp 페이지 입니다. </h1><br />
<%@ include file="include01.jsp" %>
<h1> 다시 include.jsp 페이지 입니다. </h1><br />
```

include.jsp 페이지 입니다.

include01.jsp 페이지 입니다.

다시 include.jsp 페이지 입니다.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>Insert title here</title>
</head>
<body>

<h1> include.jsp 페이지 입니다. </h1><br />

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>Insert title here</title>
</head>
<body>

<h1> include01.jsp 페이지 입니다. </h1>

</body>
</html>

<h1> 다시 include.jsp 페이지 입니다. </h1><br />

</body>
</html>
```

10-2. 지시자

taglib 지시자

사용자가 만든 tag들을 태그라이브러리라고 함

이러한 태그라이브러리를 사용하기 위해 taglib지시자를 사용

uri 및 prefix 속성이 있으며, uri는 태그라이브러리의 위치 값을 가지며, prefix는 태그를 가리키는 이름 값을 가짐

taglib 지시자에 대한 학습은 추후에 살펴볼 JSTL학습 때 진행

10-3. 주석

실제 프로그램에는 영향이 없고, 프로그램 설명들의 목적으로 사용되는 태그
HTML 및 JSP 주석이 별도로 존재

(jsp_10_3_ex1_comments)

HTML 주석

<!-- comments -->로 기술 하며, 테스트 용도 및 프로그램 설명 용도로 사용

```
<!-- <h1>여기는 주석입니다.</h1> -->  
<h1>여기는 주석이 아닙니다.</h1>
```

```
<!-- <h1>여기는 주석입니다.</h1> -->  
<h1>여기는 주석이 아닙니다.</h1>
```

JSP 주석

<%-- comments -->로 기술 하며, HTML주석과 마찬가지로 테스트 용도 및
프로그램 설명 용도로 사용 합니다.

JAVA언어의 주석도 사용 됩니다. (//, /* */)

```
<%-- 여기는 주석입니다. --%>
```

여기는 주석이 아닙니다.

여기는 주석이 아닙니다.

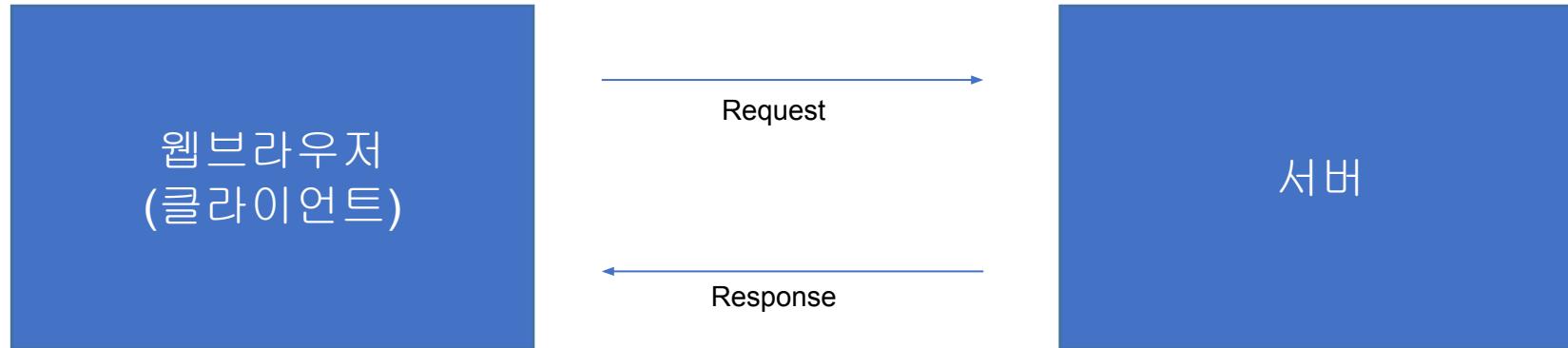
웹브라우저 소스에 차이가 있습니다.

11강. JSP 본격적으로 살펴보기-III

- request 객체의 이해
- response 객체의 이해

11-1. request 객체의 이해

웹브라우저를 통해 서버에 어떤 정보를 요청하는 것을 **request**라고 합니다. 그리고 이러한 요청 정보는 **request** 객체가 관리 합니다.



Request 객체 관련 메소드(jsp_11_1_ex1_requestobj)

`getContextPath()` : 웹어플리케이션의 컨텍스트 패스를 얻습니다.

`getMethod()` : `get`방식과 `post`방식을 구분할 수 있습니다.

`getSession()` : 세션 객체를 얻습니다.

`getProtocol()` : 해당 프로토콜을 얻습니다.

`getRequestURL()` : 요청 URL을 얻습니다.

`getRequestURI()` : 요청 URI를 얻습니다.

`getQueryString()` : 쿼리스트링을 얻습니다.

```
<%  
    out.println("서버 :" + request.getServerName() + "<br />");  
    out.println("포트 번호 :" + request.getServerPort() + "<br />");  
    out.println("요청 방식 :" + request.getMethod() + "<br />");  
    out.println("프로토콜 :" + request.getProtocol() + "<br />");  
    out.println("URL :" + request.getRequestURL() + "<br />");  
    out.println("URI :" + request.getRequestURI() + "<br />");  
%>
```

11-1. request 객체의 이해

Parameter 메소드(jsp_11_1_ex1_requestobj)

앞에서 살펴본 요청관련 메소드 보다 실제 많이 쓰이는 메소드는 **parameter**와 관련된 메소드들
Jsp페이지를 제작하는 목적이 데이터 값을 전송하기 위해서 이므로, **parameter** 관련 메소드는 중요

getParameter(String name) : name에 해당하는 파라미터 값을 구함.

getParameterNames() : 모든 파라미터 이름을 구함.

getParameterValues(String name) : name에 해당하는 파라미터값들을 구함.

The screenshot shows a Java IDE with two code editors and a browser window. The left editor contains 'form.html' with a form for inputting user information. The right editor contains 'requestparam.jsp' which reads this information from the request. The browser window shows the output of the JSP page, displaying the submitted data.

form.html:

```
7<body>
8
9<form action="requestparam.jsp" method="post">
10    이름 : <input type="text" name="name" size="10">
11    아이디 : <input type="text" name="id" size="10">
12    비밀번호 : <input type="password" name="pw" size="10">
13    취미 : <input type="checkbox" name="hobby" value="cooking">
14    <input type="checkbox" name="hobby" value="running">
15    <input type="checkbox" name="hobby" value="swimming">
16    <input type="checkbox" name="hobby" value="sleeping">
17    <input type="checkbox" name="hobby" value="reading">
18    전공 : <input type="radio" name="major" value="korean">
19    <input type="radio" name="major" value="eng" checked="checked">
20    <input type="radio" name="major" value="mat">
21    <input type="radio" name="major" value="des">
22    <select name="protocol">
23        <option value="http">http</option>
24        <option value="ftp" selected="selected">ftp</option>
25        <option value="smtp">smtp</option>
26        <option value="pop">pop</option>
27    </select><br/>
28    <input type="submit" value="전송">
29    <input type="reset" value="초기화">
30</form>
```

requestparam.jsp:

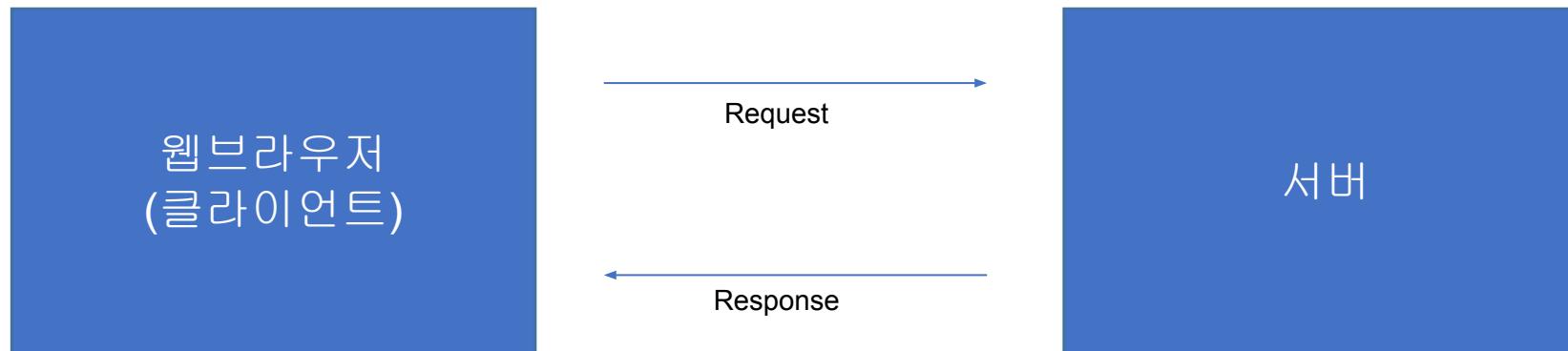
```
9</head>
10<body>
11<%!
12    String name, id, pw, major, protocol;
13    String[] hobbys;
14%>
15<%
16    request.setCharacterEncoding("EUC-KR");
17
18    name = request.getParameter("name");
19    id = request.getParameter("id");
20    pw = request.getParameter("pw");
21    major = request.getParameter("major");
22    protocol = request.getParameter("protocol");
23
24    hobbys = request.getParameterValues("hobby");
25%>
26
27
28    이름 : <%= name %><br />
29    아이디 : <%= id %><br />
30    비밀번호 : <%= pw %><br />
31    취미 : <%= Arrays.toString(hobbys) %><br />
32    전공 : <%= major %><br />
```

Browser Output:

이름 : 흥길동
아이디 : abcdefg
비밀번호 : 12324
취미 : [cook, swim]
전공 : des
프로토콜 : http

11-2. response 객체의 이해

웹브라우저의 요청에 응답하는것을 **response**라고 하며, 이러한 응답(response)의 정보를 가지고 있는 객체를 **response 객체** 라고 합니다.



Request 객체 관련 메소드(jsp_11_2_ex1_responseobj)

`getCharacterEncoding()` : 응답할때 문자의 인코딩 형태를 구합니다.

`addCookie(Cookie)` : 쿠키를 지정 합니다.

`sendRedirect(URL)` : 지정한 URL로 이동합니다.

```
<%  
String str = request.getParameter("age");  
age = Integer.parseInt(str);  
  
if( age >= 20){  
    response.sendRedirect("pass.jsp?age=" + age);  
} else {  
    response.sendRedirect("ng.jsp?age=" + age);  
}  
%>
```



12강. 액션태그

- 액션태그란?
- forward, include, param 태그 살펴보기

12-1. 액션태그란?

JSP페이지 내에서 어떤 동작을 하도록 지시하는 태그 : 예를 들어 페이지 이동, 페이지 include 등
우선 forward, include, param 태그를 살펴봄, 추후 bean 학습시 추가 학습

12-2. forward, include, param 태그 살펴보기

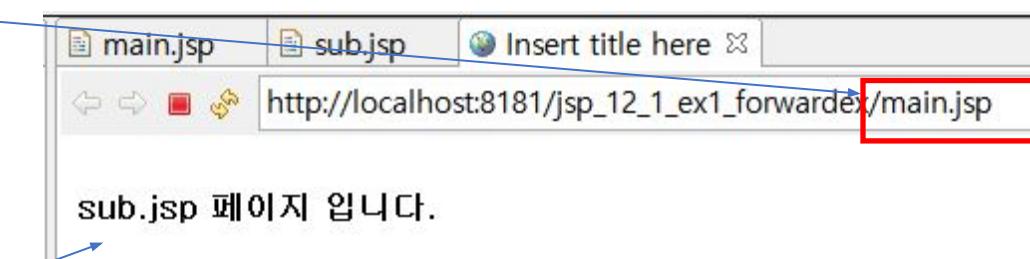
forward

현재의 페이지에서 다른 특정페이지로 전환할 때 사용
사용방법이 간단하여 예제를 통해 쉽게 이해
(jsp_12_2_ex1_actionex)

<h1>main.jsp 페이지 입니다.</h1>

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

<h5>sub.jsp 페이지 입니다.</h5>



12-2. forward, include, param 태그 살펴보기

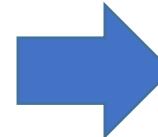
include

현재 페이지에 다른 페이지를 삽입할 때 사용 합니다.
(jsp_12_2_ex1_actionex)

```
<h1> include01.jsp 페이지 입니다. </h1>
<jsp:include page= "include02.jsp" flush= "true" />
<h1> 다시 include01.jsp 페이지 입니다. </h1>
```



```
<h1> include02.jsp 페이지 입니다. </h1>
```



include01.jsp 페이지 입니다.
include02.jsp 페이지 입니다.
다시 include01.jsp 페이지 입니다.

12-2. forward, include, param 태그 살펴보기

param

forward 및 include 태그에 데이터 전달을 목적으로 사용되는 태그입니다. 이름과 값으로 이루어져 있습니다.

(jsp_12_2_ex1_actionex)

```
<jsp:forward page="forward_param.jsp">
    <jsp:param name="id" value="abcdef" />
    <jsp:param name="pw" value="1234" />
</jsp:forward>
```

```
<%!
    String id, pw;
%>
```

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

forward_param.jsp 입니다.

아이디 : abcdef
비밀번호 : 1234

```
<h1>forward_param.jsp 입니다.</h1>
아이디 : <%= id %><br />
비밀번호 : <%= pw %>
```

forward 액션 태그

현재 JSP 페이지에서 다른 페이지로 이동하는 태그

JSP 컨테이너는 현재 JSP 페이지에서 **forward** 액션 태그를 만나면

- 그 전까지 출력 버퍼에 저장되어 있던 내용을 모두 삭제하고
- forward 액션 태그에 설정된 페이지로 프로그램의 제어가 이동

```
<jsp:forward page="파일명" />
```

반드시 끝나는 태그가 있어야 함

또는

```
<jsp:forward page="파일명">    </jsp:forward>
```

page 속성 값

- 현재 JSP 페이지에서 이동할 페이지의 외부 파일명
- 외부 파일은 현재 JSP 페이지와 같은 디렉터리에 있으면 파일명만 설정하고, 그렇지 않으면 전체 URL(또는 상대 경로)을 설정해야 함

[forward 액션 태그 사용 예]

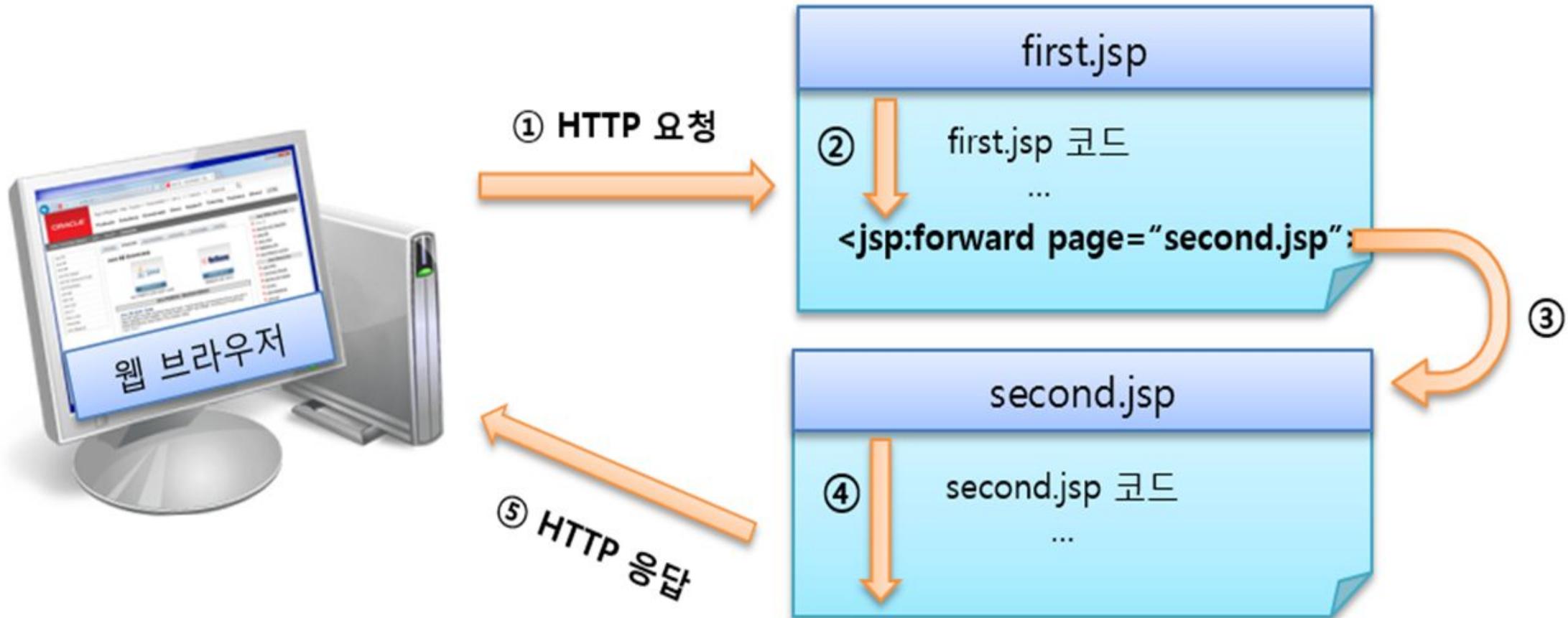
```
//first.jsp
<%@ page contentType="text/html; charset=utf-8"%>
<html>
<head>
<title>Action Tag</title>
</head>
<body>
    <h3>이 파일은 first.jsp입니다.</h3>
    <jsp:forward page="second.jsp" />
    <p>====first.jsp의 페이지=====</p>
</body>
</html>
```

```
//second.jsp
<%@ page contentType="text/html; charset=utf-8"%>
<html>
<head>
<title>Action Tag</title>
</head>
<body>
    <h3>이 파일은 second.jsp입니다.</h3>
    Today is : <%=new java.util.Date()%>
</body>
</html>
```

이 파일은 second.jsp입니다.

Today is : Wed Sep 05 15:35:07 KST 2018

forward 액션 태그의 페이지 흐름 처리 과정



예제 4-1 forward 액션 태그로 현재 날짜와 시각을 출력하는 페이지로 이동하기

JSPBook/WebContent/ch04/forward.jsp

```
01 <%@ page contentType="text/html; charset=utf-8"%>
02 <html>
03 <head>
04 <title>Action Tag</title>
05 </head>
06 <body>
07     <h2>forward 액션 태그</h2>
08     <jsp:forward page="forward_date.jsp" />
09     <p>—————
10 </body>
11 </html>
```

```
01 <%@ page contentType="text/html; charset=utf-8"%>
02 <html>
03 <head>
04 <title>Action Tag</title>
05 </head>
06 <body>
07     <p>오늘의 날짜 및 시간
08     <p><%=new java.util.Date().toLocaleString()%>
09 </body>
10 </html>
```

❖ include 액션 태그

- include �렉티브 태그처럼 현재 JSP 페이지의 특정 영역에 외부 파일의 내용을 포함하는 태그
- 현재 JSP 페이지에 포함할 수 있는 외부 파일은 HTML, JSP, 서블릿 페이지 등

```
<jsp:include page="파일명" flush="false"/>
```

- page 속성 값
 - 현재 JSP 페이지 내에 포함할 내용을 가진 외부 파일명
 - 외부 파일은 현재 JSP 페이지와 같은 디렉터리에 있으면 파일명만 설정하고, 그렇지 않으면 전체 URL(또는 상대 경로)을 설정해야 함
- flush 속성 값
 - 설정한 외부 파일로 제어가 이동할 때 현재 JSP 페이지가 지금까지 출력 버퍼에 저장한 결과를 처리, 기본 값은 false
 - true로 설정하면 외부 파일로 제어가 이동할 때 현재 JSP 페이지가 지금까지 출력 버퍼에 저장된 내용을 웹 브라우저에 출력하고 출력 버퍼를 비움

[include 액션 태그 사용 예]

```
//first.jsp
<%@ page contentType="text/html; charset=utf-8"%>
<html>
<head>
<title>Action Tag</title>
</head>
<body>
    <h3>이 파일은 first.jsp입니다.</h3>
    <jsp:include page="second.jsp" flush="false" />
    <p>Java Server Page</p>
</body>
</html>
```

이 파일은 first.jsp입니다.

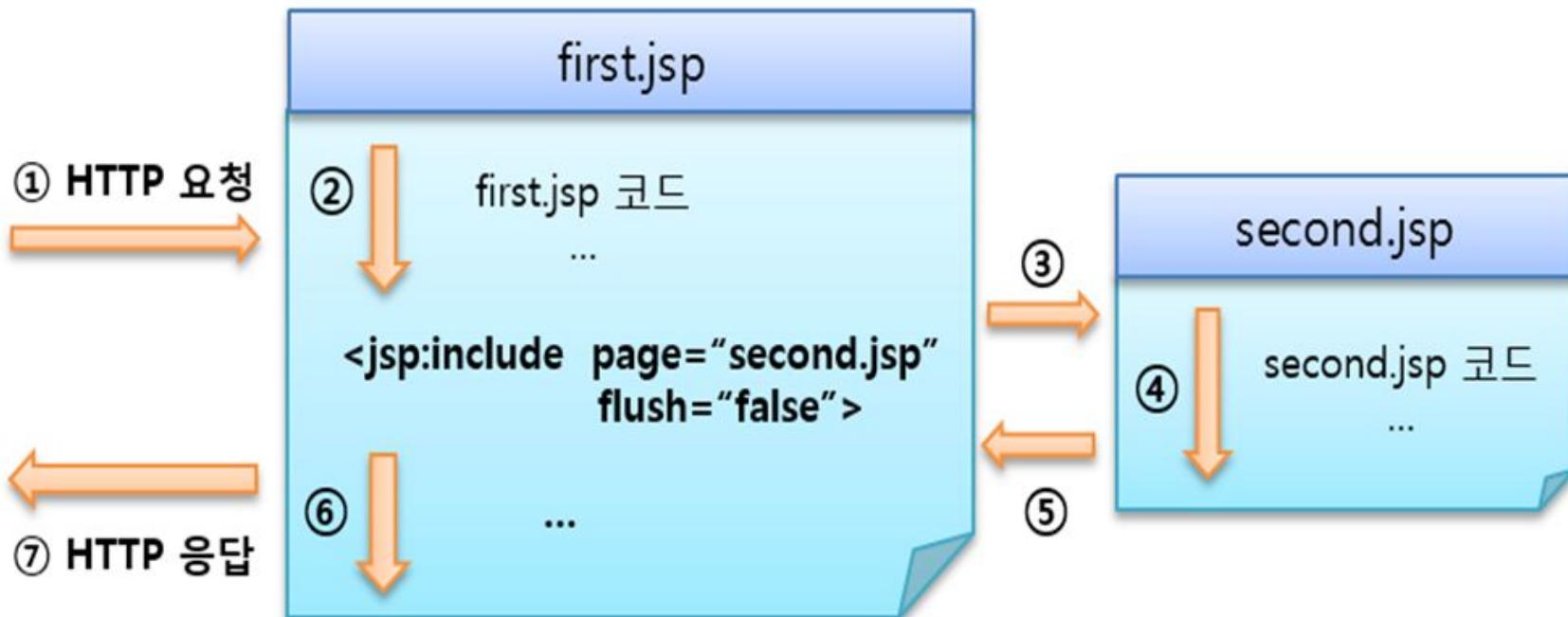
이 파일은 second.jsp입니다.

Today is : Wed Sep 05 15:38:45 KST 2018

Java Server Page

```
//second.jsp
<%@ page contentType="text/html; charset=utf-8"%>
<html>
<head>
<title>Action Tag</title>
</head>
<body>
    <h3>이 파일은 second.jsp입니다.</h3>
    Today is : <%=new java.util.Date()%>
</body>
</html>
```

❖ include 액션 태그의 처리 과정



구분	include 액션 태그	include 디렉티브 태그
처리 시간	요청 시 자원을 포함합니다.	번역 시 자원을 포함합니다.
기능	별도의 파일로 요청 처리 흐름을 이동합니다.	현재 페이지에 삽입합니다.
데이터 전달 방법	request 기본 내장 객체나 param 액션 태그를 이용하여 파라미터를 전달합니다.	페이지 내의 변수를 선언한 후 변수에 값을 저장합니다.
용도	화면 레이아웃의 일부분을 모듈화할 때 주로 사용합니다.	다수의 JSP 웹 페이지에서 공통으로 사용되는 코드나 저작권과 같은 문장을 포함하는 경우에 사용합니다.
기타	동적 페이지에 사용합니다.	정적 페이지에 사용합니다.

예제 4-2 include 액션 태그에 현재 날짜와 시각을 출력하는 페이지 포함하기

JSPBook/WebContent/ch04/include.jsp

```
01 <%@ page contentType="text/html; charset=utf-8"%>
02 <html>
03 <head>
04 <title>Action Tag</title>
05 </head>
06 <body>
07     <h2>include 액션 태그</h2>
08     <jsp:include page="include_date.jsp" flush="true" />
09     <p>-----
10 </body>
11 </html>
```



```
01 <%@ page contentType="text/html; charset=utf-8"%>
02 <html>
03 <head>
04 <title>Action Tag</title>
05 </head>
06 <body>
07     <p>오늘의 날짜 및 시각</p>
08     <p><%=new java.util.Date().toLocaleString()%></p>
09 </body>
10 </html>
```

❖ param 액션 태그

- 현재 JSP 페이지에서 다른 페이지에 정보를 전달하는 태그
- 이 태그는 단독으로 사용되지 못하며 <jsp:forward>나 <jsp:include> 태그의 내부에 사용
- 다른 페이지에 여러 개의 정보를 전송해야 할 때는 다중의 param 액션 태그 사용

```
<jsp:forward page="파일명" >  
  <jsp:param name="매개변수명1" value="매개변수값1" />  
  [<jsp:param name="매개변수명2" value="매개변수값2" /> ...]  
</jsp:forward>
```

[jsp:param 액션 태그 사용 예]

```
//first.jsp
<%@ page contentType="text/html; charset=utf-8"%>
<html>
<head>
<title>Action Tag</title>
</head>
<body>
    <h3>이 파일은 first.jsp입니다.</h3>
    <jsp:include page="second.jsp">
        <jsp:param name="date" value="<%=new java.util.Date()%>" />
    </jsp:include>
    <p>Java Server Page</p>
</body>
</html>
```

```
//second.jsp
<%@ page contentType="text/html; charset=utf-8"%>
<html>
<head>
<title>Action Tag</title>
</head>
<body>
    Today is : <%=request.getParameter("date")%>
</body>
</html>
```

이 파일은 first.jsp입니다.
Today is : Wed Sep 05 15:43:02 KST 2018
Java Server Page

예제 4-3 forward 액션 태그와 param 액션 태그에 아이디와 이름 전달하기

JSPBook/WebContent/ch04/param01.jsp

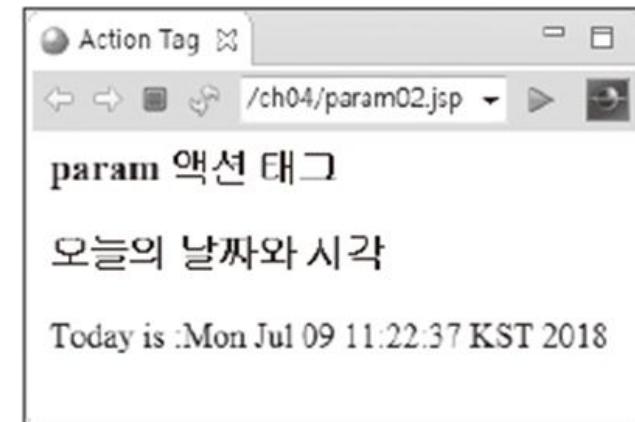
```
01 <%@ page contentType="text/html; charset=utf-8"%>
02 <html>
03 <head>
04 <title>Action Tag</title>
05 </head>
06 <body>
07     <h3>param 액션 태그</h3>
08     <jsp:forward page="param01_data.jsp">
09         <jsp:param name="id" value="admin" />
10         <jsp:param name="name" value='<%=java.net.URLEncoder.encode("관리자")%>' />
11     </jsp:forward>
12     <p>Java Server Page
13 </body>
14 </html>
```



```
01 <%@ page contentType="text/html; charset=utf-8"%>
02 <html>
03 <head>
04 <title>Action Tag</title>
05 </head>
06 <body>
07     <p> 아이디 : <%=request.getParameter("id")%>
08     <%
09         String name = request.getParameter("name");
10     %>
11     <p> 이 름 : <%=java.net.URLDecoder.decode(name)%>
12 </body>
13 </html>
```

예제 4-4 include 액션 태그와 param 액션 태그에 제목과 현재 날짜 전달하기

```
01 <%@ page contentType="text/html; charset=utf-8"%>
02 <html>
03 <head>
04 <title>Action Tag</title>
05 </head>
06 <body>
07     <h3>param 액션 태그</h3>
08     <jsp:include page="param02_data.jsp">
09         <jsp:param name="title" value='<%=java.
           net.URLEncoder.encode("오늘의 날짜와 시각")%>' />
10         <jsp:param name="date" value="<%=java.util.Calendar.getInstance().
           getTime()%>" />
11     </jsp:include>
12 </body>
13 </html>
```



```
01 <%@ page contentType="text/html; charset=utf-8"%>
02 <html>
03 <head>
04 <title>Action Tag</title>
05 </head>
06 <body>
07     <%
08         String title = request.getParameter("title");
09     %>
10     <h3>%=java.net.URLDecoder.decode(title)%</h3>
11     Today is :<%=request.getParameter("date")%>
12 </body>
13 </html>
```


13강. 쿠키

- 쿠키란?
- 쿠키 문법

13-1. 쿠키란?

웹브라우저에서 서버로 어떤 데이터를 요청 하면, 서버측에서는 알맞은 로직을 수행한 후 데이터를 웹브라우저에 응답

서버는 웹브라우저와의 관계를 종료

웹브라우저에 응답 후 연결상태를 끊는 것은 http프로토콜의 특징

연결이 끊긴 후에도 정보를 지속적으로 유지하기 위한 수단으로 쿠키를 사용

쿠키는 서버에서 생성하여, 서버가 아닌 클라이언트측에 특정 정보를 저장

클라이언트가 서버에 요청 할 때 쿠키를 포함

쿠키는 **4kb**로 용량이 제한적이며, 300개까지 데이터 정보를 가질 수 있습니다.

13-2. 쿠키 문법

쿠키는 서버에서 생성되고, 클라이언트측에 전송되어 저장

쿠키 생성 방법 및 관련 메소드



13-2. 쿠키 문법

쿠키 관련 메소드

- `setMaxAge()` : 쿠키 유효기간을 설정 합니다.
- `setpath()` : 쿠키사용의 유효 디렉토리를 설정 합니다.
- `setValue()` : 쿠키의 값을 설정 합니다.
- `setVersion()` : 쿠키 버전을 설정 합니다.
- `getMaxAge()` : 쿠키 유효기간 정보를 얻습니다.
- `getName()` : 쿠키 이름을 얻습니다.
- `getPath()` : 쿠키사용의 유효 디렉토리 정보를 얻습니다.
- `getValue()` : 쿠키의 값을 얻습니다.
- `getVersion()` : 쿠키 버전을 얻습니다.

14강. 세션

- 세션이란?
- 세션 문법

14-1. 세션이란?

앞에서 웹브라우저와의 관계를 유지하는 수단으로 쿠키

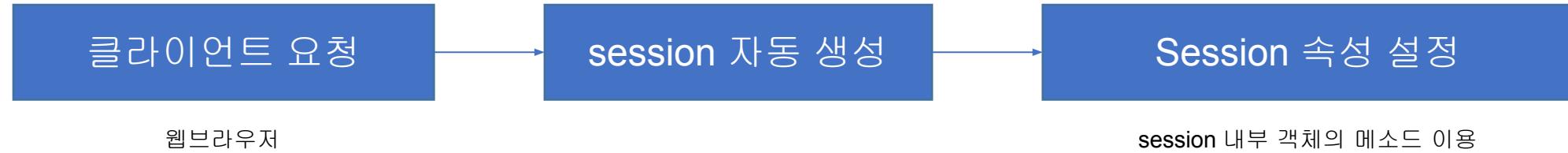
세션도 쿠키와 마찬가지로 서버와의 관계를 유지하기 위한 수단

쿠키와 달리 세션은 클라이언트에 저장되는 것이 아니라, 서버에 존재

세션은 서버에서 접근이 가능하여 보안이 좋고, 저장할 수 있는 데이터에 한계가 없음

14-2. 세션 문법

세션은 클라이언트의 요청이 발생하면 자동생성 됩니다. 그리고 **session**이라는 내부 객체를 지원하여 세션의 속성을 설정 할 수 있습니다.



14-2. 세션 문법

세션 관련 메소드

`setAttribute()` : 세션에 데이터를 저장 합니다.

`getAttribute()` : 세션에서 데이터를 얻습니다.

`getAttributeNames()` : 세션에 저장되어 있는 모든 데이터의 이름(유니크한 키값)을 얻습니다.

`getId()` : 자동 생성된 세션의 유니크한 아이디를 얻습니다.

`isNew()` : 세션이 최초 생성되었는지, 이전에 생성된 세션인지를 구분 합니다.

`getMaxInactiveInterval()` : 세션의 유효시간을 얻습니다. 가장 최근 요청시점을 기준으로 카운트

(...\\apache-tomcat...\\apache-tomcat...\\conf\\web.xml 참조)

`removeAttribute()` : 세션에서 특정 데이터 제거 합니다.

`Invalidate()` : 세션의 모든 데이터를 삭제 합니다.

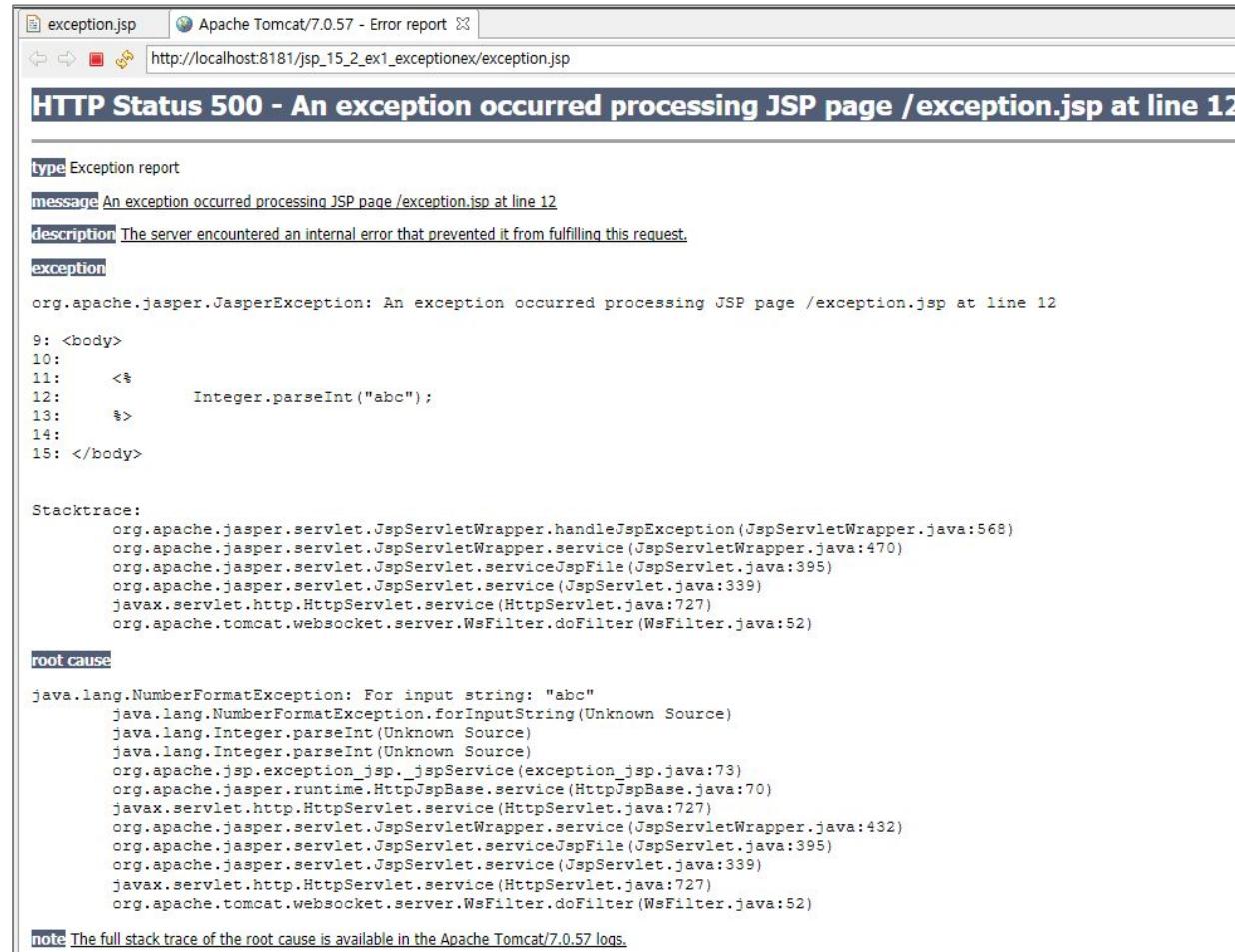
15강. 예외 페이지

- 예외 페이지의 필요성
- page 지시자를 이용한 예외 처리
- web.xml파일을 이용한 예외 처리

15-1. 예외 페이지의 필요성

JAVA언어 예외처리 처럼 JSP, Servlet에서도 예외가 발생

예외적인 상황이 발생했을 경우 웹컨테이너(톰캣)에서 제공되는 기본적인 예외 페이지가 보여 진다면, 사용자로 하여금 원가 불쾌한 느낌이 들면서, 다시는 해당 사이트에 접속하려 들지 않을 것 입니다. 따라서 약간은 다소 딱딱한 에러 페이지를 보다 친근한 느낌이 느껴지는 페이지로 유도



The screenshot shows a browser window with the title "exception.jsp | Apache Tomcat/7.0.57 - Error report". The URL in the address bar is "http://localhost:8181/jsp_15_2_ex1_exception/exception.jsp". The main content is a 500 Internal Server Error page titled "HTTP Status 500 - An exception occurred processing JSP page /exception.jsp at line 12". The page contains the following information:

- type**: Exception report
- message**: An exception occurred processing JSP page /exception.jsp at line 12
- description**: The server encountered an internal error that prevented it from fulfilling this request.
- exception**:

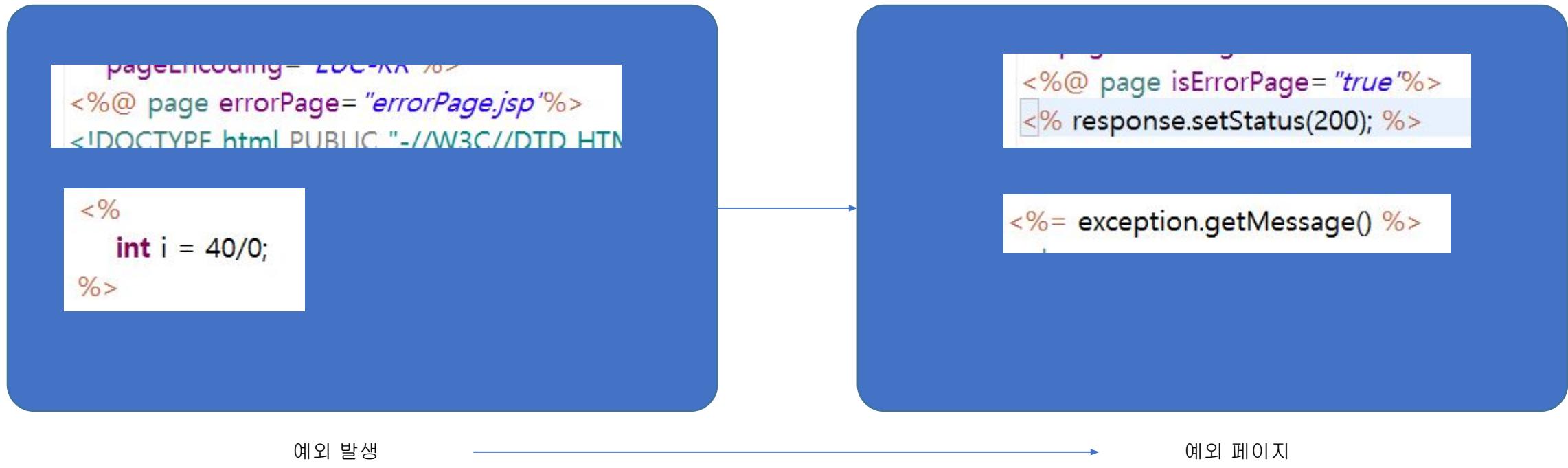
```
org.apache.jasper.JasperException: An exception occurred processing JSP page /exception.jsp at line 12

9: <body>
10:
11:     <%
12:         Integer.parseInt("abc");
13:     %>
14:
15: </body>
```
- Stacktrace**:

```
org.apache.jasper.servlet.JspServletWrapper.handleJspException(JspServletWrapper.java:568)
org.apache.jasper.servlet.JspServletWrapper.service(JspServletWrapper.java:470)
org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet.java:395)
org.apache.jasper.servlet.JspServlet.service(JspServlet.java:339)
javax.servlet.http.HttpServlet.service(HttpServlet.java:727)
org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:52)
```
- root cause**:

```
java.lang.NumberFormatException: For input string: "abc"
java.lang.NumberFormatException.forInputString(Unknown Source)
java.lang.Integer.parseInt(Unknown Source)
java.lang.Integer.parseInt(Unknown Source)
org.apache.jsp._jspService(exception_jsp.java:73)
org.apache.jasper.runtime.HttpJspBase.service(HttpJspBase.java:70)
javax.servlet.http.HttpServlet.service(HttpServlet.java:727)
org.apache.jasper.servlet.JspServletWrapper.service(JspServletWrapper.java:432)
org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet.java:395)
org.apache.jasper.servlet.JspServlet.service(JspServlet.java:339)
javax.servlet.http.HttpServlet.service(HttpServlet.java:727)
org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:52)
```
- note**: The full stack trace of the root cause is available in the Apache Tomcat/7.0.57 logs.

15-2. page지시자를 이용한 예외 처리



15-3. web.xml파일을 이용한 예외 처리

(jsp_15_3_ex1_exceptionex)

```
<error-page>
    <error-code>404</error-code>
    <location>/error404.jsp</location>
</error-page>
<error-page>
    <error-code>500</error-code>
    <location>/error500.jsp</location>
</error-page>
```

404에러 발생시 error404.jsp 페이지로 이동

500에러 발생시 error500.jsp 페이지로 이동

16강. 자바 빈

- 빈 이란?
- 빈 만들기
- 빈 관련 액션 태그(useBean, getProperty, setProperty)

16-1. 빈 이란?

반복적인 작업을 효율적으로 하기 위해 빈을 사용 합니다.

빈이란? JAVA언어의 데이터(속성)와 기능(메소드)으로 이루어진 클래스 입니다.

jsp페이지를 만들고, 액션태그를 이용하여 빈을 사용 합니다. 그리고 빈의 내부 데이터를 처리 합니다.

16-2. 빈 만들기

JAVA언어를 학습하면서 데이터 객체를 많이 만들어본 경험이 있을 것
데이터 객체에는 데이터가 있어 그에 해당하는 getter와 setter가 있음
빈을 만든다는 것은 데이터 객체를 만들기 위한 클래스를 만드는 것
(jsp_16_2_ex1_beanex)

```
package com.javalex.ex;  
  
public class Student {  
  
    private String name;  
    private int age;  
    private int grade;  
    private int studentNum;  
  
    public Student() {  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
  
    public int getAge() {  
        return age;  
    }  
  
    public void setAge(int age) {  
        this.age = age;  
    }  
}
```

16-3. 빈 관련 액션 태그(useBean, setProperty, getProperty)

액션태그 중에서 Bean관련한 태그가 있다. 주로 데이터를 업데이트하고, 얻어오는 역할을 한다.

useBean

특정 Bean을 사용한다고 명시 할 때 사용

```
<jsp:useBean id="student" class="com.jspstd.ex.Lect" scope="page"/>
```



Scope

page : 생성된 페이지 내에서만 사용 가능 합니다.

request : 요청된 페이지 내에서만 사용 가능 합니다.

session : 웹브라우저의 생명주기와 동일하게 사용 가능 합니다.

application : 웹 어플리케이션 생명주기와 동일하게 사용 가능 합니다.

16-3. 빈 관련 액션 태그(useBean, setProperty, getProperty)

setProperty

데이터 값을 설정 할 때 사용 합니다.

```
<jsp:setProperty name="student" property="name" value="홍길동"/>
```

빈 이름

속성 이름

속성(데이터) 값

getProperty

데이터 값을 가져올 때 사용 합니다.

```
<jsp:getProperty name="student" property="name" />
```

빈 이름

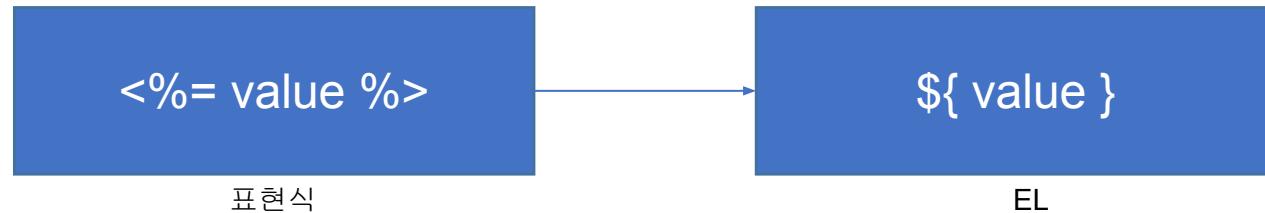
속성 이름

23강. EL(Expression Language)

- EL(Expression Language)?
- 액션태그로 사용되는 EL
- 내장객체

23-1. EL(Expression Language)?

EL(Expression Language)란, 표현식 또는 액션 태그를 대신해서 값을 표현하는 언어입니다.
(jsp_23_1_ex1_elx)



EL 연산자 (jsp_23_1_ex1_elx)

산술 : +, -, *, / %
관계형 : ==, != <, >, <=, >=
조건 : a? b : c
논리 : &&, ||

23-2. 액션태그로 사용되는 EL (jsp_23_2_ex1_elex)

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



```
 ${member.name }
```

23-3. 내장 객체

(jsp_23_3_ex1_elex)

pageScope : page 객체를 참조하는 객체

requestScope : request 객체를 참조하는 객체

sessionScope : session 객체를 참조하는 객체

applicationScope : application 객체를 참조하는 객체

param : 요청 파라미터를 참조하는 객체

paramValues : 요청 파라미터(배열)를 참조하는 객체

initParam : 초기화 파라미터를 참조하는 객체

cookie : cookie 객체를 참조하는 객체

24강. JSTL(JSP standard Tag Library)

- JSTL 개요 및 설치
- JSTL 라이브러리

24-1. JSTL 개요 및 설치

JSP의 경우 HTML 태그와 같이 사용되어 전체적인 코드의 가독성이 떨어집니다.

그래서 이러한 단점을 보완하고자 만들어진 태그 라이브러리가 **JSTL**입니다.

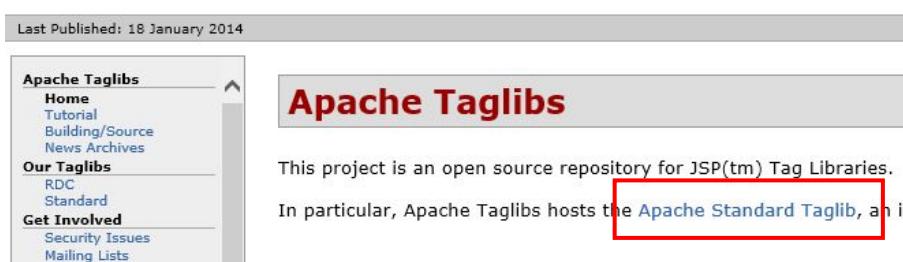
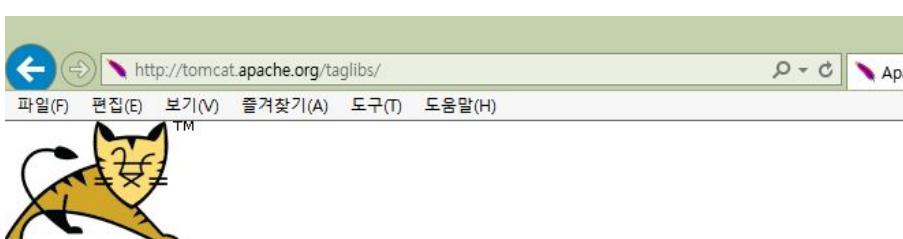
JSTL의 경우 우리가 사용하는 Tomcat 컨테이너에 포함되어 있지 않으므로, 별도의 설치를 하고 사용합니다.

JSTL 설치

<http://jakarta.apache.org/> 접속 한 후, 좌측의 Taglibs 클릭



Apache Standard Taglib 클릭



24-1. JSTL 개요 및 설치

JSTL 설치

Standard 1.1 download 클릭

Standard Taglib

JSP(tm) Standard Tag Library implementations

Apache hosts the Apache Standard Taglib, an implementation of the JSP Standard Tag Library (JSTL) specification. Various versions are available.

Version	JSTL version	Requirements	Getting the Taglib
Standard 1.2.1	JSTL 1.2	Servlet 2.5, JavaServer Pages 2.1	download (javadoc)
Standard 1.1	JSTL 1.1	Servlet 2.4, JavaServer Pages 2.0	download
Standard 1.0	JSTL 1.0	Servlet 2.3, JavaServer Pages 1.2	download

binaries 클릭

archive.apache.org

This site contains the historical archive of

For current releases, please visit the [mirro](#)

Name	Last modified
Parent Directory	
 binaries/	2005-1
 source/	2005-1

Apache/2.4.10 (Unix) mod_wsgi/3.5 Python

24-1. JSTL 개요 및 설치

JSTL 설치

jakarta-taglibs-standard-1.1.2.zip 클릭

jakarta-taglibs-standard-1.1.0.zip.asc	2004-01-28 20:11	304
jakarta-taglibs-standard-1.1.1.tar.gz	2004-07-19 21:53	872K
jakarta-taglibs-standard-1.1.1.tar.gz.asc	2004-07-19 21:53	186
jakarta-taglibs-standard-1.1.1.zip	2004-07-19 21:53	931K
jakarta-taglibs-standard-1.1.1.zip.asc	2004-07-19 21:53	186
jakarta-taglibs-standard-1.1.2.tar.gz	2004-10-25 20:57	873K
jakarta-taglibs-standard-1.1.2.tar.gz.asc	2004-10-25 20:57	186
jakarta-taglibs-standard-1.1.2.zip	2004-10-25 20:57	933K
jakarta-taglibs-standard-1.1.2.zip.asc	2004-10-25 20:57	186
jakarta-taglibs-standard-oldxml-compat.tar.gz	2002-06-21 22:59	1.1M
jakarta-taglibs-standard-oldxml-compat.tar.gz.asc	2002-06-21 22:59	232
jakarta-taglibs-standard-oldxml-compat.zip	2002-06-21 23:01	1.1M
jakarta-taglibs-standard-oldxml-compat.zip.asc	2002-06-21 23:01	232

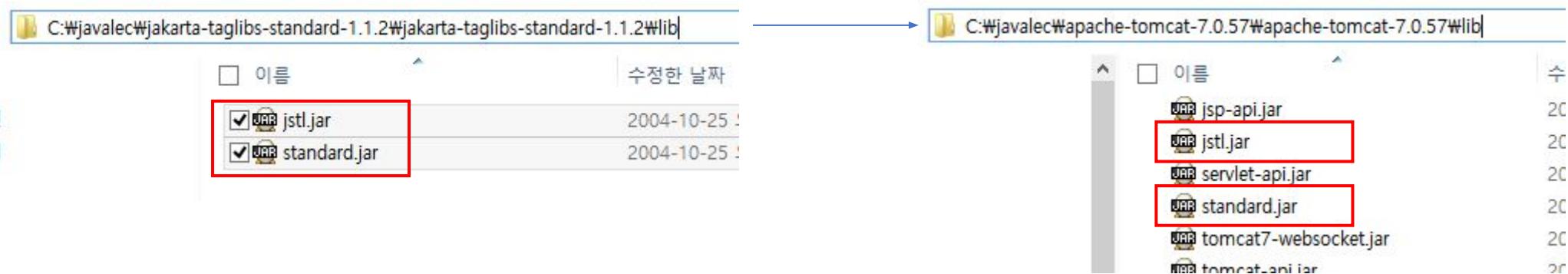
압축해제

apache-tomcat-7.0.57.zip
cos-26Dec2008.zip
eclipse-jee-luna-SR1-win32-x86_64.zip
jakarta-taglibs-standard-1.1.2.zip
jdk-7u71-docs-all.zip
OracleXE112_Win64.zip
sqldeveloper-4.1.0.17.29-no-jre.zip
workspace.zip
apache-tomcat-7.0.57.zip
cos-26Dec2008.zip
eclipse-jee-luna-SR1-win32-x86_64.zip
jakarta-taglibs-standard-1.1.2.zip
jdk-7u71-docs-all.zip

24-1. JSTL 개요 및 설치

JSTL 설치

라이브러리 파일 복사



24-2. JSTL 라이브러리

JSTL에서는 다섯 가지 라이브러리 제공 (Core, XML Processing, I18N formatting, SQL, Functions)

lib	URI	Prefix	ex
Core	http://java.sun.com/jsp/jstl/core	c	<c:tag
XML Processing	http://java.sun.com/jsp/jstl/xml	x	<x:tag
I18N formatting	http://java.sun.com/jsp/jstl/fmt	fmt	<fmt:tag
SQL	http://java.sun.com/jsp/jstl/sql	sql	<sql:tag
Functions	http://java.sun.com/jsp/jstl/functions	fn	fn:function()

Core

Core 라이브러리는 기본적인 라이브러리로 출력, 제어문, 반복문 같은 기능이 포함되어 있습니다.
(jsp_24_2_ex1_elex)

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

24-2. JSTL 라이브러리

출력 태그 : <c:out>

```
<c:out value="출력 값" default="기본값" escapeXml="true or false">
```

변수 설정 태그 : <c:set>

```
<c:set var="변수명" value="설정 값" target="객체" property="값" scope="범위">
```

변수를 제거하는 태그 : <c:remove>

```
<c:remove var="변수명" scope="범위">
```

예외 처리 태그 : <c:catch>

```
<c:catch var="변수명">
```

24-2. JSTL 라이브러리

제어문(if) 태그 : <c:if>

```
<c:if test="조건" var="조건 처리 변수명" scope="범위">
```

제어문(swich) 태그 : <c:choose>

```
<c:choose>
<c:when test="조건"> 처리 내용 </c:when>
<c:otherwise> 처리 내용 </c:otherwise>
</c:choose>
```

반복문(for) 태그 : <c:forEach>

```
<c:forEach items="액체명" begin="시작 인덱스" end="끝 인덱스" step="증감식"
var="변수명" varStatus="상태변수">
```

24-2. JSTL 라이브러리

페이지 이동 태그 : <c:redirect>

```
<c:redirect url="url">
```

파라미터 전달 태그 : <c:param>

```
<c:param name="파라미터명" value="값">
```

