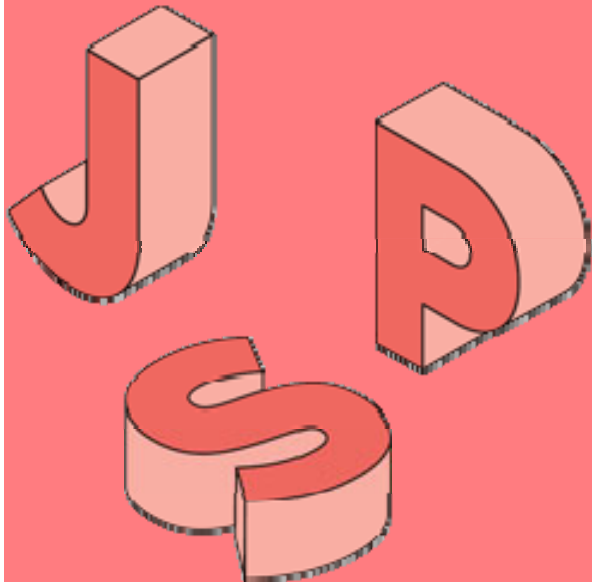


Chapter 02

자바 웹 개발을 위한 환경 구축



목차

1. 자바 웹 개발환경의 개요
2. 클라우드 개발환경과 배포 프로세스
3. 자바 웹 개발환경 구축
4. 프로젝트 생성 및 톰캣 연동

학습목표

- 자바 웹 개발을 위한 개발환경을 이해하고 설치한다.
- 클라우드 개발환경과 배포 프로세스의 개념을 익힌다.
- 이클립스를 설치하고 기본 사용법을 익힌다.







Section 01

자바 웹 개발환경의 개요

자바란?

■ 자바의 특징

- 간결하면서도 강력한 객체지향 프로그래밍 언어
- 플랫폼에 독립적이어서 여러 운영체제나 하드웨어에서도 동일하게 실행할 수 있음
- 많은 오픈소스 라이브러리를 통해 생산성이 향상되고 유지보수 비용이 절감됨
- GUI 기반의 응용 프로그램 개발에는 적합하지 않음
- 하드웨어를 정밀하게 제어해야 하는 프로그램 개발에는 비교적 적합하지 않음
- 최신 모던 프로그래밍 언어에 비해 간결함이 떨어지고 코드가 복잡함
- 지난 5년 동안 프로그래밍 언어 인기도에서 자바가 계속 1위였으며,
2022년 7월 기준으로 C언어에 이어 3위를 차지함

Jul 2022	Jul 2021	Change	Programming Language		Ratings	Change
1	3	▲		Python	13.44%	+2.48%
2	1	▼		C	13.13%	+1.50%
3	2	▼		Java	11.59%	+0.40%
4	4			C++	10.00%	+1.98%
5	5			C#	5.65%	+0.82%
6	6			Visual Basic	4.97%	+0.47%

자바란?

■ 자바 가상머신(JVM)

- 자바의 가장 큰 특징은 가상머신(Virtual Machine)이라는 개념임
- 가상머신은 말 그대로 물리적인 기계 장치가 아닌 가상의 기계를 의미함
 - 즉 소프트웨어로 구현된 하드웨어를 말함
- JVM으로 인해 특정 하드웨어나 운영체제의 영향을 받지 않고 동일한 프로그램의 개발이 가능해짐
- JVM은 개발과 실행이 가능한 기술임
- 자바 소스는 컴파일 후 바이트코드가 생성되며, JVM은 바이트코드를 해석하여 운영체제에서 실행할 수 있도록 번역하는 역할을 담당함

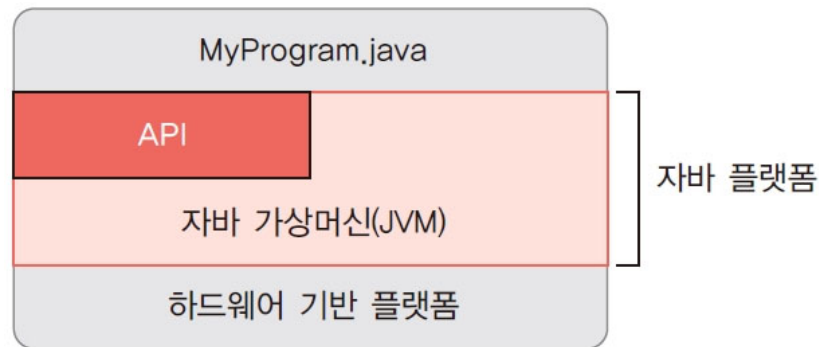


그림 2-2 JVM과 자바 플랫폼

자바란?

■ 자바 플랫폼

- API와 JVM으로 구성되며 특정 자바 프로그램이 실행되는 환경을 의미함
- 자바 플랫폼에는 Java SE, Java EE, Java ME, Java Card, Java TV 등이 있음
- 실행환경에 따라 API 구성 등이 달라짐
 - 일반적인 목적 : Java SE를 설치
 - 대규모 개발 목적 : JavaEE를 추가 설치

여기서 잠깐! Java EE 시장의 축소

스프링과 같은 오픈소스 프레임워크의 성장으로 인해 Java EE 시장이 점차 축소되고, 오라클은 Java EE를 이클립스 재단으로 이관했다. 따라서 자바 웹 개발을 위해 Java EE 플랫폼까지는 필요하지 않으며 아파치 톰캣과 같은 서블릿 컨테이너만 설치해도 충분하다.

여기서 잠깐! 자바의 버전

자바의 최신 버전은 2020년 말 기준으로 Java SE 14다. 매년 버전업되어 최신 버전이 나오지만 개발의 안정성과 호환을 위해 LTS(Long Term Support) 버전을 사용하는 것이 좋다. 현재 LTS 버전은 11이며, 최대 2026년까지 기술 및 보안 업데이트 등이 보장되어 있기 때문에 대부분의 프로젝트나 학습에서는 11을 권장한다.

자바 웹 개발환경

- 자바 웹 개발을 위한 기본적인 도구
 - 자바 개발도구(JDK), 통합개발환경(IDE), 서블릿 컨테이너, 데이터베이스
- 구축 개발환경의 버전

표 2-1 JSP 개발환경

항목	프로그램명	이 책의 버전
자바 개발도구	JDK	AdoptOpenJDK 11(LTS)
통합개발환경	이클립스	Eclipse IDE for Java Enterprise Developer
서블릿 컨테이너	아파치 톰캣	Apache Tomcat 9
데이터베이스	H2	H2 Database v1.4

자바 웹 개발환경

■ 자바 개발도구(JDK)

- 자바 개발도구는 자바 실행을 위한 JRE(Java Runtime Environment)와 컴파일러 등을 포함하며, 보통 JDK(Java Development Kit)로 불림
- 통합개발환경(IDE)을 비롯해 아파치 톰캣과 같이 자바로 만들어진 서버 실행을 위해 서도 JDK는 필요함
- JDK 패키지는 상용 제품인 Oracle JDK와 공개 버전인 Open JDK로 나뉘어 배포됨
 - 몇몇 구성요소와 설치 방법 등의 차이를 제외하면 기본적인 기능과 성능은 동일함
 - 특별히 오라클에 종속되거나 반드시 라이선스를 지불해야만 하는 환경이 아니라면 Open JDK의 사용이 권장됨

자바 웹 개발환경

- 통합개발환경(Integrated Development Environment, IDE)
 - 이클립스(Eclipse)
 - 대표적인 자바 통합개발도구로 IBM에서 개발해 오픈소스로 기증한 개발도구
 - 자바 이외의 개발도구로도 사용할 수 있음
 - 무료로 사용할 수 있다는 장점과 다양한 플러그인 등으로 오랜 기간 대표적인 자바 개발도구로 자리 잡음
 - 점차 상용 개발도구인 IntelliJ에 자리를 내어주고 있음
 - 이클립스는 개발 목적에 따라 여러 버전이 있으며, 웹 개발을 위해서는 Eclipse IDE for Java Enterprise Developer를 설치해야 함

자바 웹 개발환경

■ 통합개발환경(Integrated Development Environment, IDE)

■ IntelliJ IDEA

- 젯브레인스(Jetbrains)사에서 개발한 자바 개발도구로, 코틀린 역시 이 회사에서 개발함
- IntelliJ는 저사양 컴퓨터에서 사용하기 어렵다는 단점이 있음
- 하지만 많은 기능과 편리함으로 인해 가장 대표적인 자바 개발도구가 됨
- 다만 본격적인 개발을 위해서는 상용버전을 사용해야 하고, 특히 자바 웹 개발은 상용 버전인 Ultimate 버전을 사용해야 함

■ VS Code(Visual Studio Code)

- 프론트엔드 개발이 본격화되면서 가장 빠르게 성장하고 있는 개발도구
- 마이크로소프트에서 제공하는 오픈소스 개발도구로 가벼운 기본 구조로 저사양의 컴퓨터에서도 비교적 쉽게 사용할 수 있음
- 자바 기반의 개발이 가능하지만 관련된 익스텐션을 직접 설치해야 하고 개별적인 환경 설정을 해야 하기 때문에 불편함이 있음

자바 웹 개발환경

- 서블릿 컨테이너(Web Application Server, WAS)
 - 이름과 같이 웹 애플리케이션을 구동하는 서버를 의미함
 - 서버 컴퓨터가 WAS로 동작하려면 서블릿 컨테이너가 필요함
 - 실제 서비스 시스템을 구축할 때는 정적 콘텐츠 서비스를 위한 웹 서버와 WAS를 병행해 운영하며 설정을 통해 상호 연동되는 구조를 가짐
 - 아파치 톰캣, Jetty, Undertow 등이 사용됨

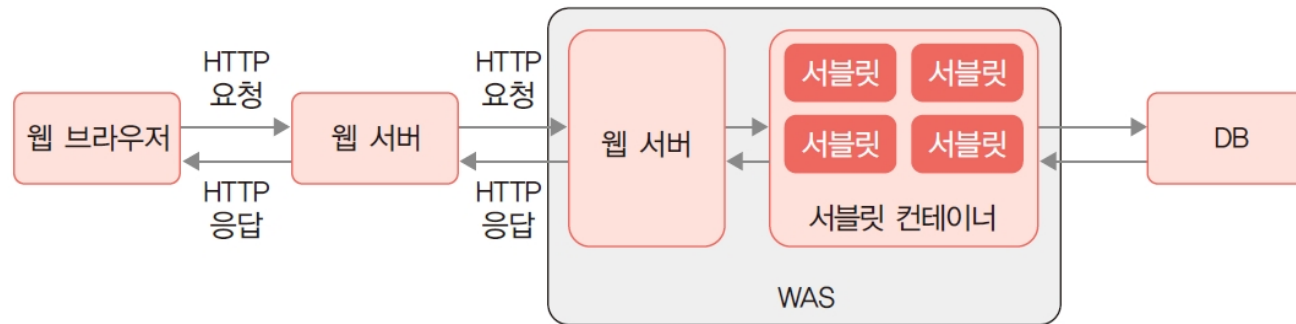


그림 2-3 서블릿 컨테이너 구조


Section 02

클라우드 개발환경과 배포 프로세스

클라우드 개발환경

■ 클라우드 개발환경

- 최근 클라우드 인프라와 협업을 위한 기본적인 배경지식과 경험이 요구되고 있음
- 개발환경이 점점 복잡해짐과 더불어 개발 결과물을 실제 서버로 배포하는 과정과 절차도 점점 복잡해지고 있음

여기서  **잠깐!** 나는 클라우드 개발환경에 얼마나 익숙한가?

다음의 질문 리스트를 가볍게 체크하며 클라우드 개발환경에 관해 어느 정도 배경지식이 있는지 점검해보자.

- 깃Git에 대해 전반적으로 이해하며 이를 개발 소스 관리에 활용할 수 있는가?
- 깃허브GitHub, 깃랩GitLab 등의 소스 공유 및 관리 도구를 사용할 수 있는가?
- 데브옵스DevOps 개념을 이해하고 적응할 수 있는가?
- 아마존 웹 서비스AWS, 구글 클라우드 플랫폼GCP 등의 클라우드 인프라를 활용할 수 있는가?
- 도커Docker 등의 컨테이너를 기반으로 한 개발 경험이 있는가?

배포 프로세스

■ 배포(Deployment)

- 배치라고도 하며, 개발된 결과물을 실제 사용자에게 전달하는 작업을 의미함
- 웹의 경우
 - 운영 서버로 소스코드를 복사하고 WAS에 등록하는 과정을 의미함
- 모바일 앱의 경우
 - 앱 스토어에 앱을 업로드하는 절차 등을 의미함
- 그 외의 경우
 - FTP 서버, 홈페이지 또는 깃허브 등을 통해 버전을 릴리즈(Release)하는 작업 등

배포 프로세스

■ 웹 애플리케이션 배포

- 이클립스에서 JSP 또는 서블릿을 실행하면 현재 프로젝트 구조를 WAR 형태로 패키징한 후 아파치 톰캣에 전달해 실행하는 구조임
- 반대로 완성된 프로젝트를 운영 서버에 설치하는 경우 이클립스에서 프로젝트를 .war 파일로 패키징한 다음 서버에 옮기는 과정이 필요함
 - 운영 서버가 리눅스일 경우 다음의 과정을 겪음
 - ① 개발자의 윈도우 컴퓨터에서 운영 서버에 원격으로 접속하여 .war 파일을 전송
 - ② 톰캣이 설치된 디렉터리의 [webapps] 폴더로 .war 파일 이동
 - ③ 톰캣에서 자동으로 압축을 풀어 사용함
- 이와 같은 방법은 비효율적이며 불편한 점이 많음
 - 해결책 : 데브옵스라 불리는 개발 · 운영 통합 인프라스트럭처

배포 프로세스

■ 데브옵스(DevOps)

- 개발(Development)과 운영(Operation)의 합성어
- 규모가 큰 소프트웨어의 신속한 개발과 지속적 유지보수, 배포 등의 운영을 병행하기 위한 노력을 통칭함
- 수시로 서비스를 배포하는 형태, 웹 기반의 서비스 형태가 필수요소로 자리잡음
- 이러한 새로운 전략을 위해 개발 팀과 운영 팀이 병합되어 개발, 테스트, 배포, 운영에 이르는 애플리케이션 생명주기를 개발하게 됨

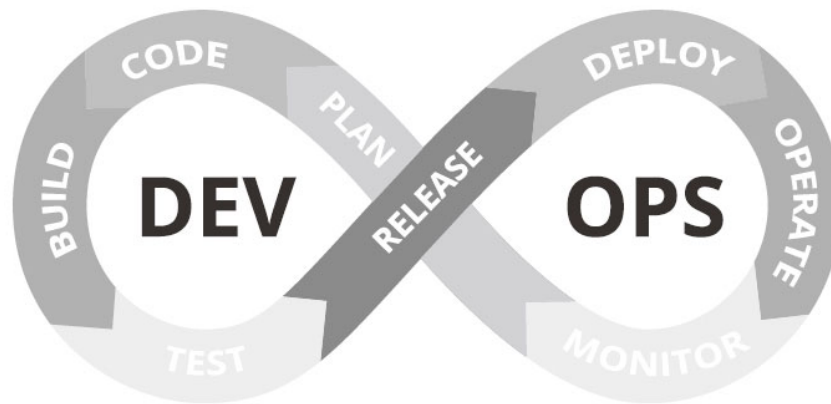


그림 2-5 데브옵스의 개념과 프로세스

배포 프로세스

■ 데브옵스의 장점

- **빠른 속도** : 계획부터 배포까지 작업 속도를 빠르고 효율적으로 제공하기 때문에 시장 변화에 빠르게 대처하고 비즈니스 성과를 창출할 수 있음
- **빠른 배포** : 새로운 기능의 릴리즈와 오류 수정 속도가 빨라져 고객의 요구에 빠르게 대응할 수 있음
- **안정성** : 지속적 통합, 지속적 전달, 모니터링, 로깅을 통해 안정적인 서비스 품질을 고객에게 제공할 수 있음
- **확장 가능성** : 복잡하거나 변화하는 시스템을 효율적으로 관리할 수 있음
- **협업 강화** : 개발 팀과 운영 팀이 긴밀하게 협력할 수 있기 때문에 책임을 공유할 수 있기 때문에 비효율을 줄이고 시간을 절약할 수 있음

배포 프로세스

■ 데브옵스의 구성요소

- **SCM(Source Code Management)** : 팀 단위의 소스코드 버전 관리, 깃, SVN 등의 도구를 사용함
- **CI(Continuous Integration)** : 빌드와 테스트의 통합, Jenkins, Travis CI 등의 도구를 사용함
- **CD(Continuous Deploy)** : 지속적인 배포, 원하는 시점에 바로 배포가 가능한 설정이 필요함
- **CM(Configuration Management)** : 서비스 설정의 통합 관리, 운영 서버 OS, 라이브러리 버전, 컴파일 등을 포함함

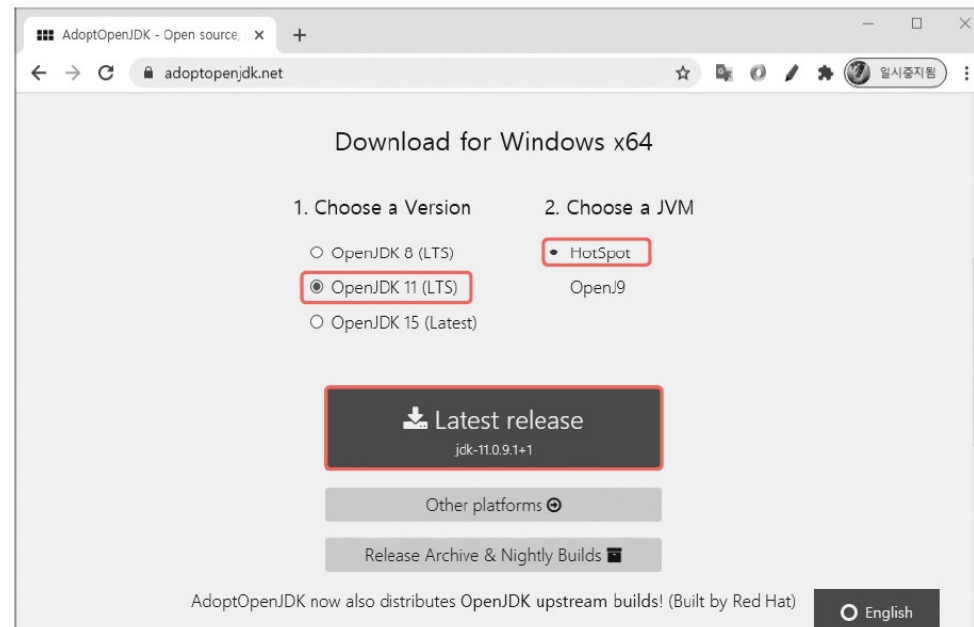
Section 03

자바 웹 개발환경 구축

JDK 설치하기

■ AdoptOpenJDK 설치하기

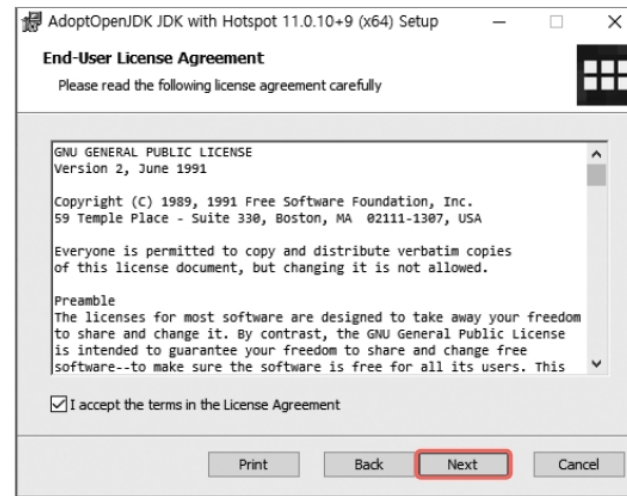
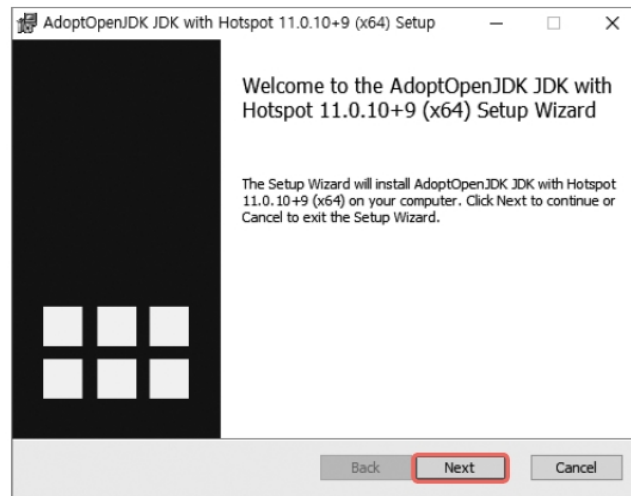
- ① 웹 브라우저에서 <https://adoptopenjdk.net>에 접속하여 'OpenJDK 11 (LTS)' 버전을 선택하고 JVM은 'HotSpot'을 선택한 후, <Latest release>를 클릭하여 다운로드



JDK 설치하기

■ AdoptOpenJDK 설치하기

② 다운로드한 파일을 실행하고, <Next>를 클릭하여 설치 진행하기

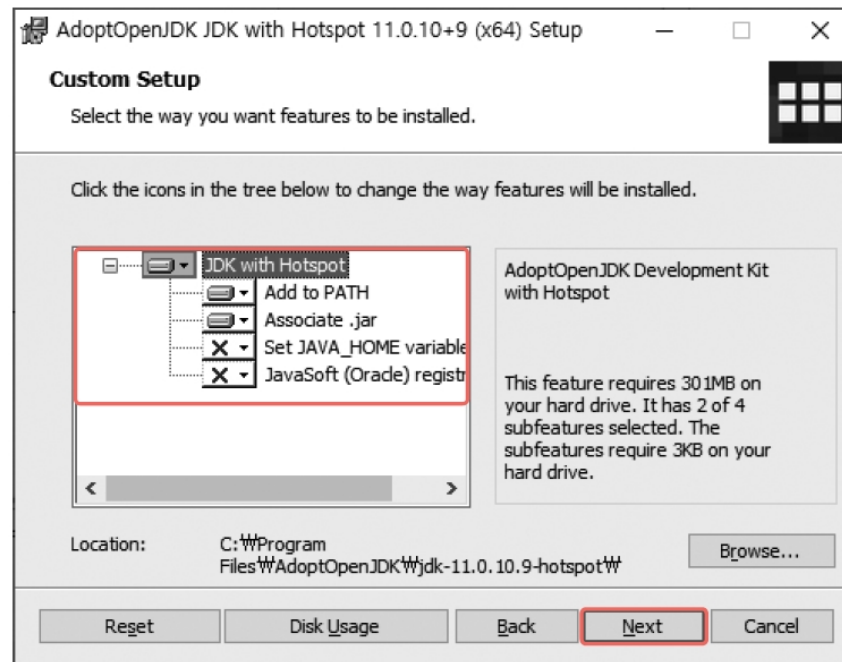


JDK 설치하기

■ AdoptOpenJDK 설치하기

③ [Custom Setup] 창에서 설치 옵션을 기본값으로 두고 <Next>를 클릭

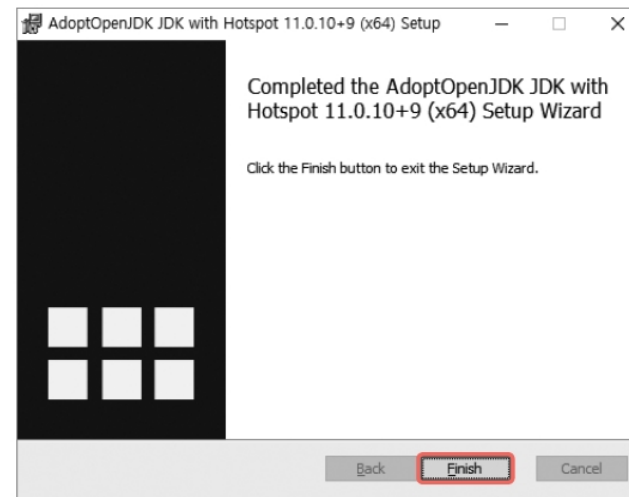
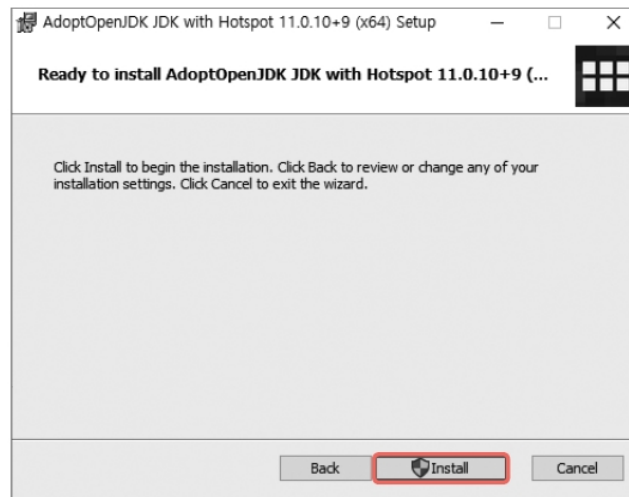
- 'Add to PATH' : 별도의 작업 없이 윈도우 PATH에 JDK 설치 경로를 추가하는 옵션
- 'Associate.jar' : jar 파일 실행을 연동하는 옵션
- 'Set JAVA_HOME variable'에는 체크하지 않기



JDK 설치하기

■ AdoptOpenJDK 설치하기

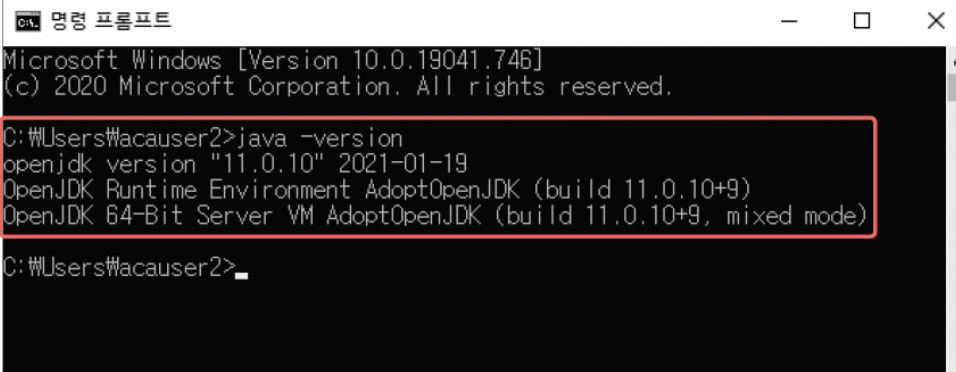
④ <Install>을 클릭하여 설치를 진행하고, 설치가 끝나면 <Finish>를 클릭



JDK 설치하기

■ AdoptOpenJDK 설치하기

- ⑤ 설치가 완료되면 윈도우의 시작 버튼에서 'cmd'를 입력하여 명령 프롬프트 창을 띄우고 'java -version'을 입력
- 설치된 OpenJDK의 버전이 나오면 정상적으로 설치된 것임



```
CA 명령 프롬프트
Microsoft Windows [Version 10.0.19041.746]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\macauser2>java -version
openjdk version "11.0.10" 2021-01-19
OpenJDK Runtime Environment AdoptOpenJDK (build 11.0.10+9)
OpenJDK 64-Bit Server VM AdoptOpenJDK (build 11.0.10+9, mixed mode)

C:\Users\macauser2>
```

여기서 잠깐! JDK 설치 유의사항

대부분의 경우 문제없이 설치되지만, 만일 사용자의 컴퓨터에 이미 설치된 JDK가 있다면 다음 사항에 유의한다.

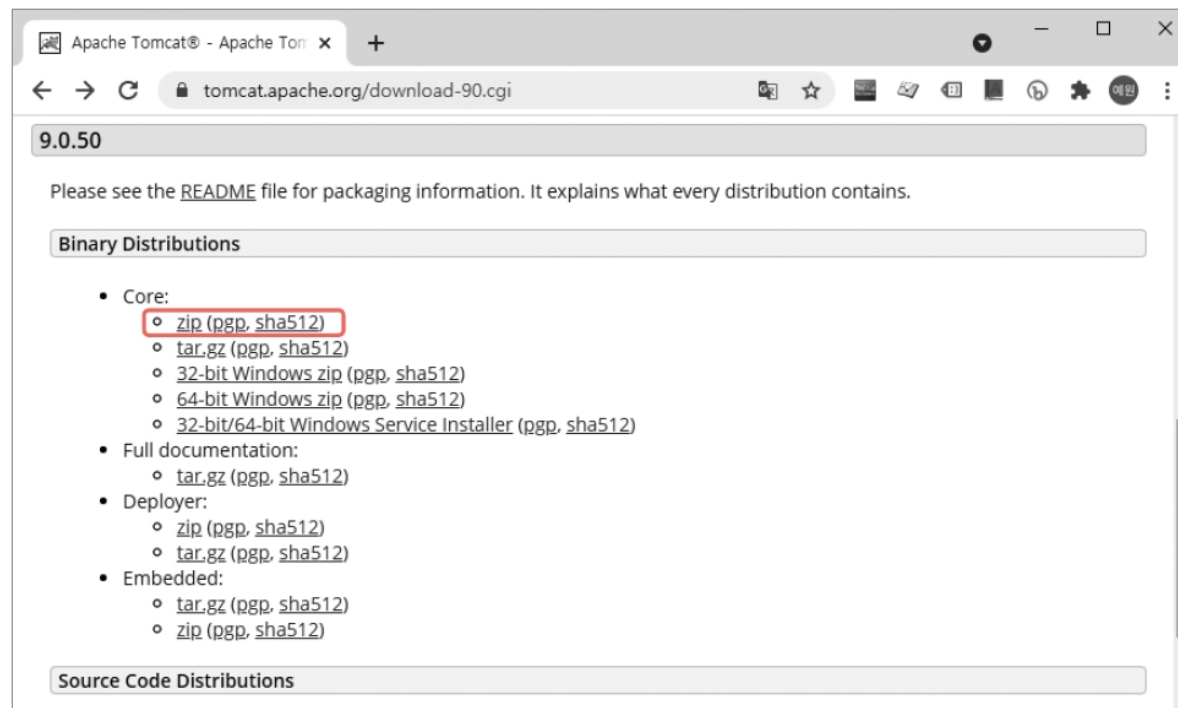
- 기존 설치 버전이 JDK 11이라면 설치된 버전을 그대로 사용해도 된다.
- 설치된 JDK가 Oracle JDK 또는 JDK 11 이외의 버전이라면 가능한 삭제하길 권장한다.
- OpenJDK 설치 전 시스템 환경변수의 PATH에 JDK 관련 경로 및 JAVA_HOME 환경변수 관련 설정이 되어 있다면 삭제 후 설치를 권장한다.

TIP 리눅스 또는 맥을 사용하는 경우에도 동일하며, 추가적인 설치 옵션은 <https://adoptopenjdk.net/installation.html#macos-pkg>를 참고한다.

아파치 톰캣 설치하기

■ 아파치 톰캣 압축 버전(zip) 설치하기

- ① 웹 브라우저에서 <https://tomcat.apache.org/download-90.cgi>에 접속하여 [Binary Distributions] → [Core] → [zip]을 선택하여 다운로드

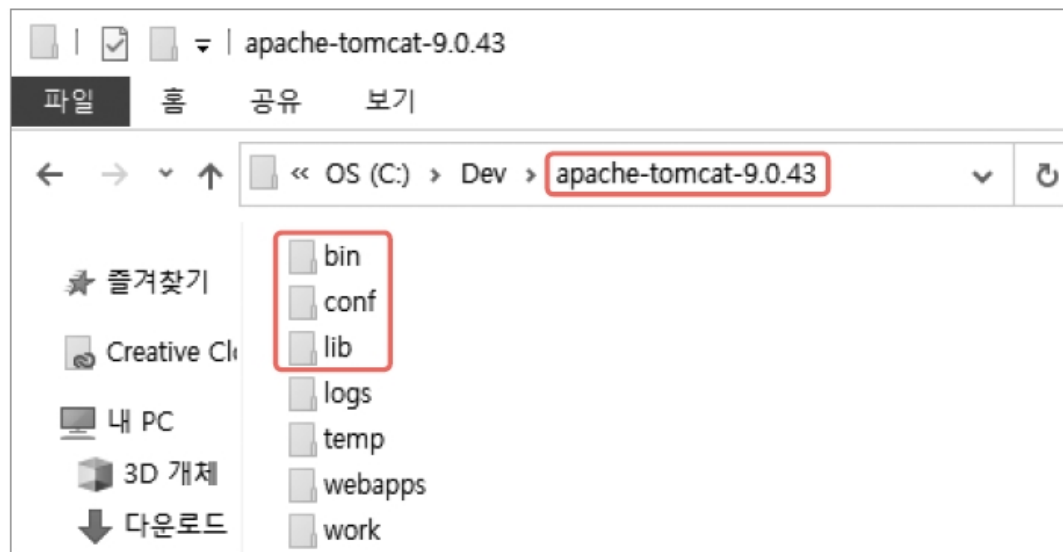


아파치 톰캣 설치하기

■ 아파치 톰캣 압축 버전(zip) 설치하기

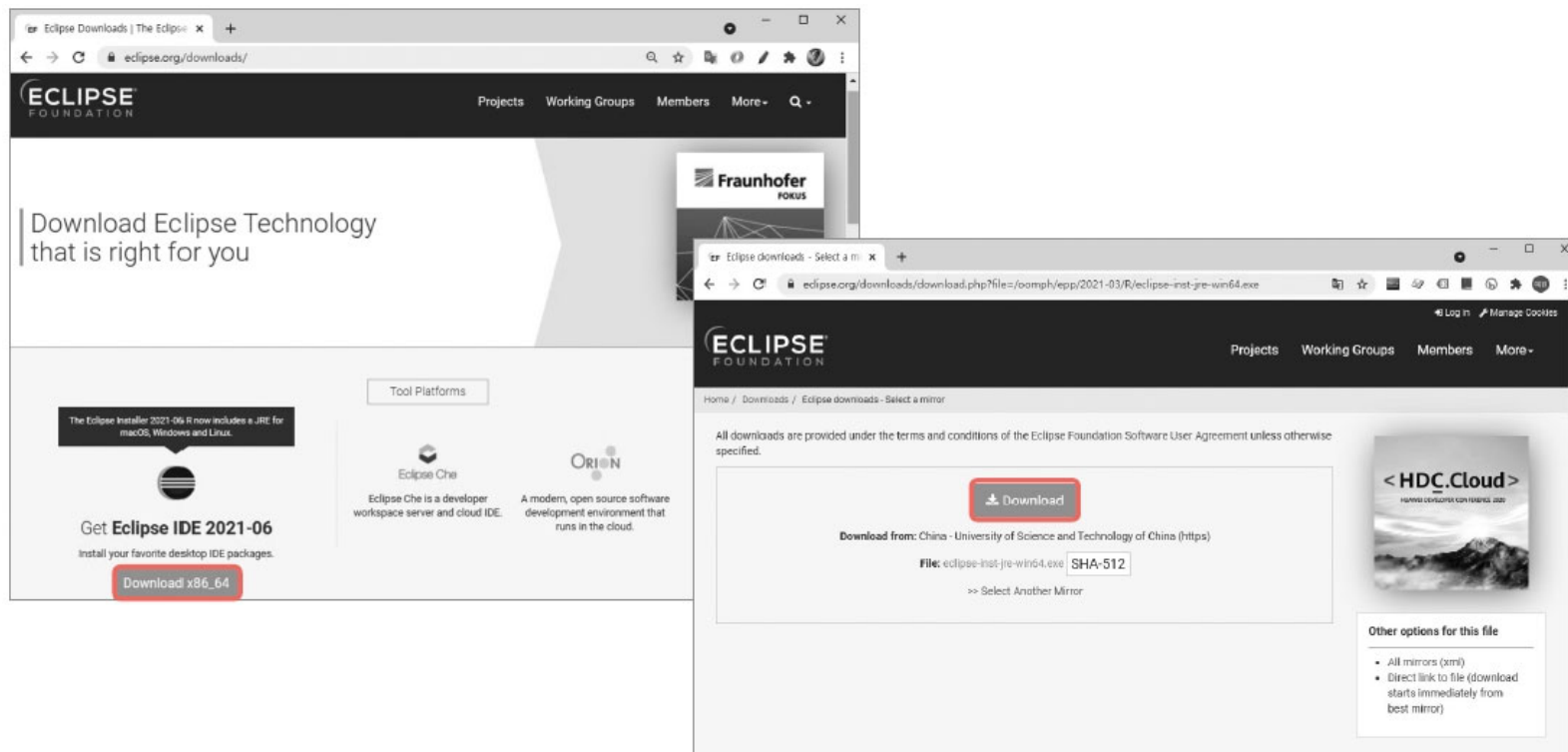
② 다운로드한 'apache-tomcat-9.0.zip' 파일의 압축을 풀고 C드라이브에 옮김

- 주의할 점) [apache-tomcat-9] 폴더 밑에 바로 [bin], [conf], [lib] 등의 폴더가 보이도록 복사해야 함



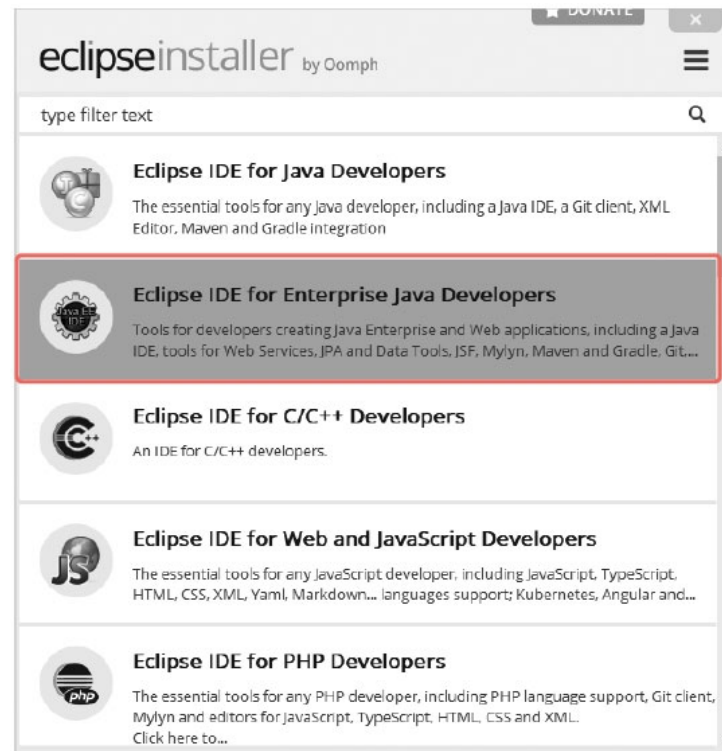
이클립스 설치하기

- 웹 개발용 이클립스(Eclipse IDE for Java Enterprise Developer) 설치
 - ① 웹 브라우저에서 <https://www.eclipse.org/downloads/>에 접속하여 <Download x86_64>를 클릭. 이어지는 화면에서 <Download>를 클릭하고 설치 진행



이클립스 설치하기

- 웹 개발용 이클립스(Eclipse IDE for Java Enterprise Developer) 설치
 - ① [eclipseinstaller] 창에서 'Eclipse IDE for Enterprise Java Developers'를 선택
 - 이후 나오는 라이선스 및 권한 설정은 모두 동의에 체크



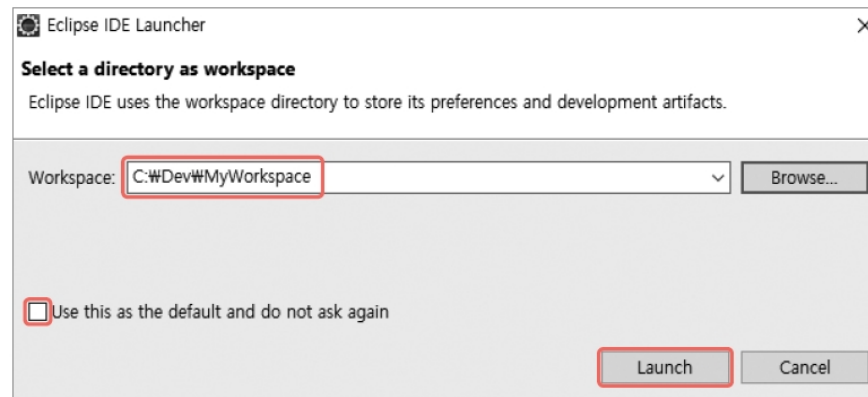
이클립스 설치하기

■ 워크스페이스 지정

- 이클립스에서 프로젝트 및 소스코드를 관리하기 위한 저장 공간을 의미함

- 하나의 개발 프로젝트에 여러 이클립스 프로젝트가 있을 수 있으며,
개발 단위에 따라 워크스페이스를 달리 둘 수도 있음

- ③ 이클립스를 실행하면 [Select a directory as workspace]라는 워크스페이스를 선택하는 창이 나옴. 여기서는 C드라이브의 [Dev]→[Workspace]를 기본 경로로 함
 - 주의할 점) 반드시 워크스페이스의 폴더 위치를 기억하고 확인해야 함



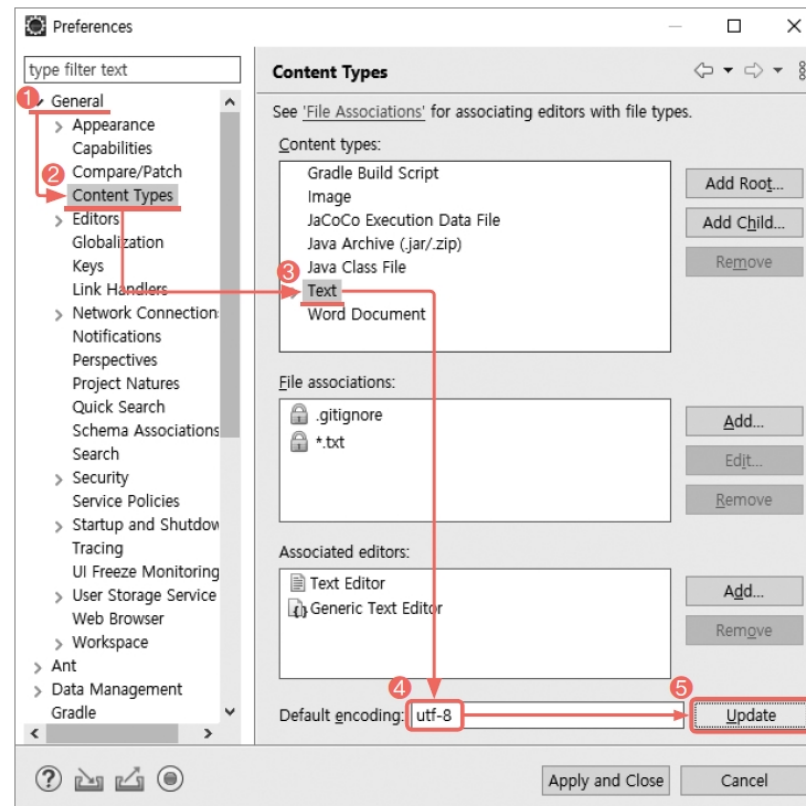
- ④ 실행이 완료되면 나오는 [Welcome] 화면에서 오른쪽 상단의 <Hide> 버튼을 눌러 메인 작업 화면으로 들어감

이클립스 환경 설정하기

■ 텍스트 인코딩 설정하기

- 한글은 여러 문제를 발생시키는 원인이 되기 때문에 캐릭터 인코딩으로 유니코드 인코드 방식 중 UTF-8을 기본으로 사용함

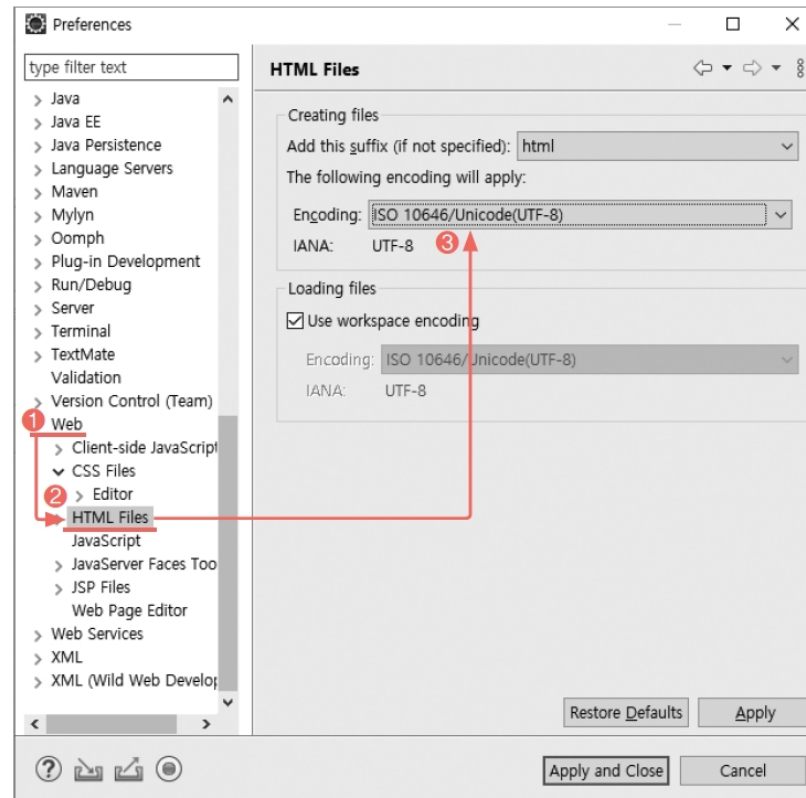
- ① [Window] → [Preferences]를 클릭하여 설정 화면을 열기. [General] → [Content Types] → [Text]를 선택하고, [Default encoding]에 'utf-8'을 입력하고 <Update> 버튼 클릭



이클립스 환경 설정하기

■ 텍스트 인코딩 설정하기

- ② HTML, CSS, JSP의 캐릭터 인코딩 설정도 변경하기. [Web] → [HTML Files]를 선택하고 다음 그림과 같이 인코딩 항목을 'ISO 10646/Unicode(UTF-8)'로 변경
 - 같은 방식으로 [CSS Files]와 [JSP Files]도 변경하기

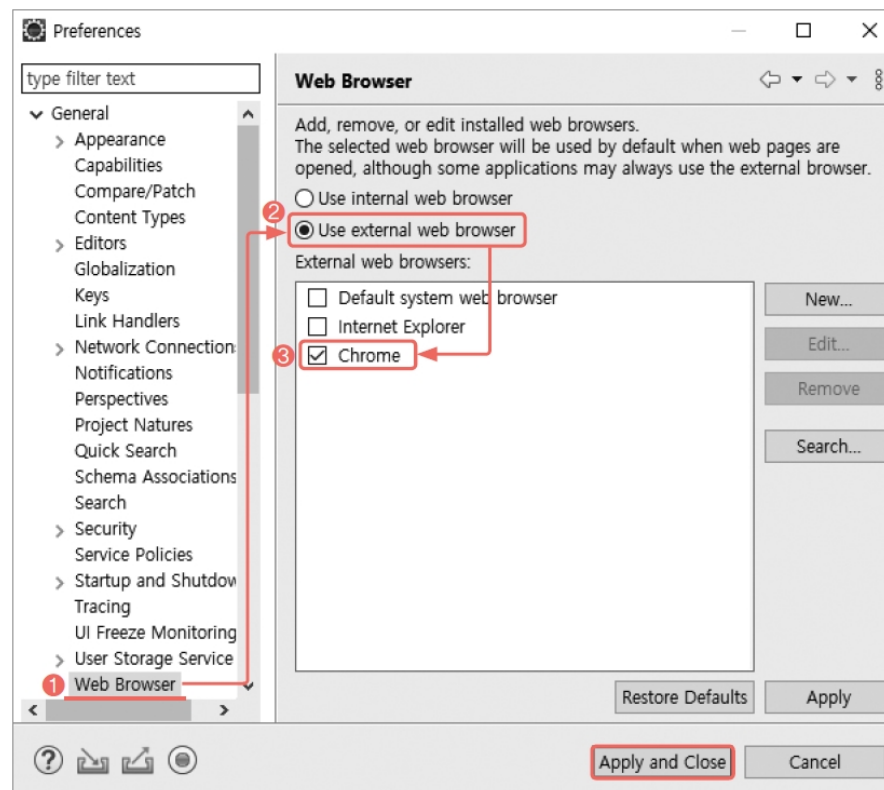


이클립스 환경 설정하기

■ 웹 브라우저 설정

- 이클립스에서 화면 결과를 확인할 때 이클립스 '내장 웹 브라우저' 또는 '외부 웹 브라우저'를 선택할 수 있음

- ③ 이클립스에서 구글 크롬을 기본 브라우저로 설정하기 위해 [General] → [Web Browser]에서 'Chrome'에 체크

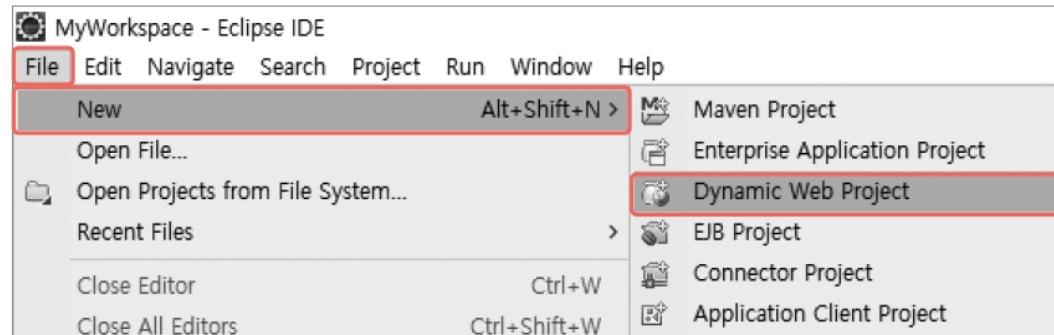


Section 04

프로젝트 생성 및 톰캣 연동

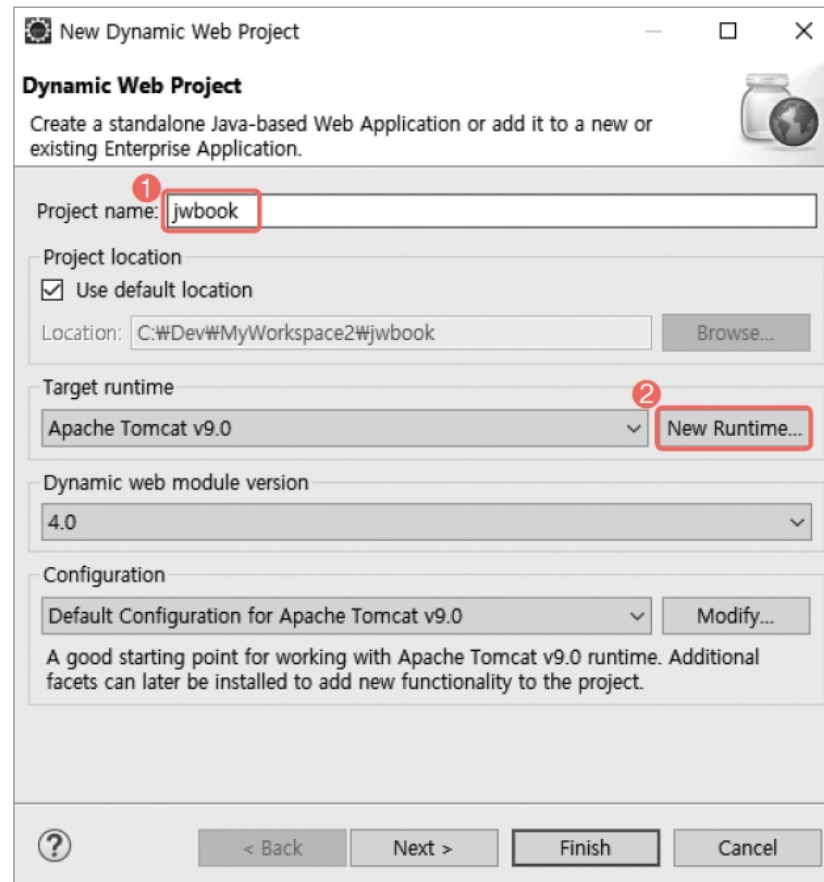
동적 웹 프로젝트 생성

- 정적 웹 프로젝트(Static Web Project)
 - HTML, CSS, 자바스크립트로만 구성된 웹을 개발할 때 생성함
 - 동적 웹 프로젝트(Dynamic Web Project)
 - 서블릿, JSP 등을 개발할 때 생성함
- ① [File] → [New] → [Dynamic Web Project]를 선택



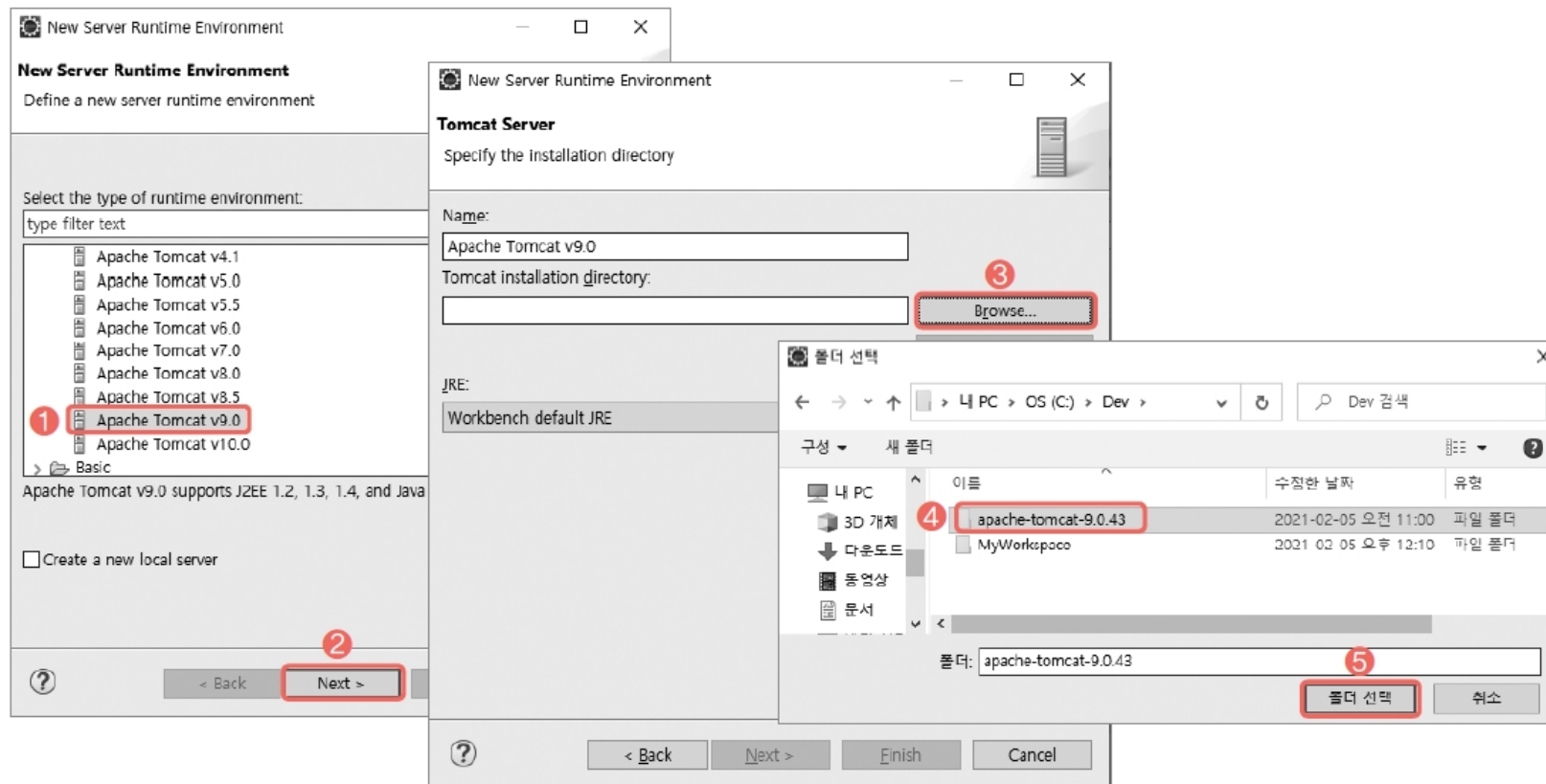
동적 웹 프로젝트 생성

- ② [Dynamic Web Project] 창에서 프로젝트 이름에 'jwbook'을 입력하고 <Target runtime> 설정을 위해 <New Runtime...> 버튼을 클릭
- <Target runtime> : 서블릿과 JSP 실행을 위한 런타임을 의미함
 - 서블릿 컨테이너를 지정하는 항목으로 톰캣 또는 다른 서블릿 컨테이너를 선택할 수 있음



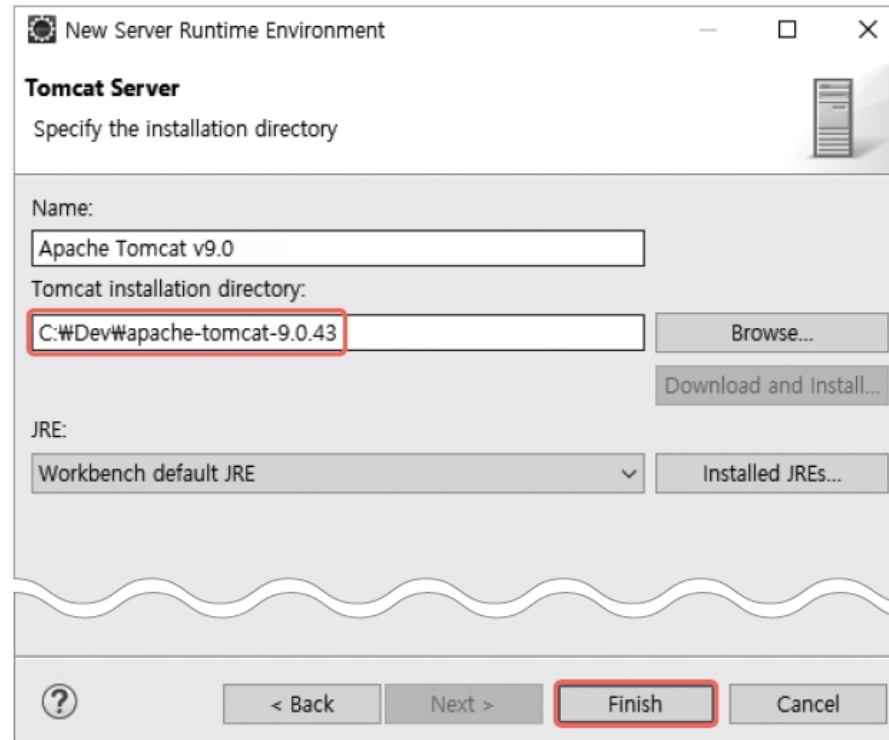
동적 웹 프로젝트 생성

- ③ 톰캣의 버전과 경로를 등록하기 위해 [New Server Runtime Environment] 창에서 'Apache Tomcat v9.0'을 선택하고 <Next>를 클릭. [Tomcat Server] 화면에서 <Browse...> 버튼을 클릭하여 톰캣이 위치한 폴더를 찾아 선택



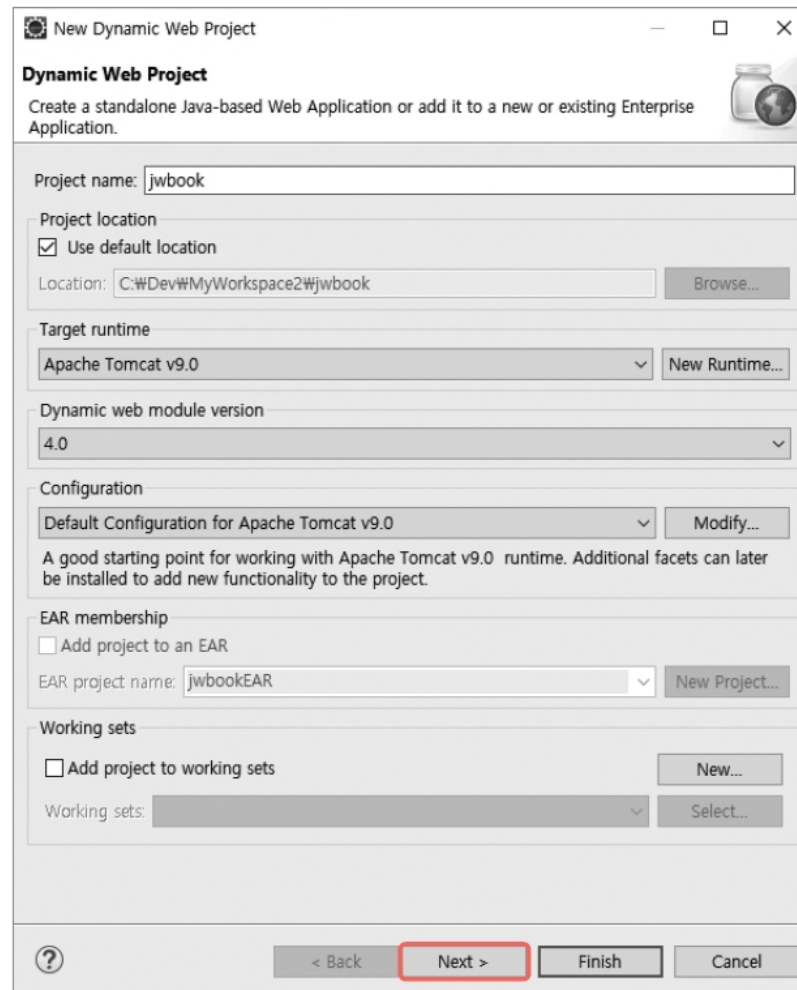
동적 웹 프로젝트 생성

- ④ 톰캣의 경로 설정을 완료했다면 <Finish> 버튼을 클릭



동적 웹 프로젝트 생성

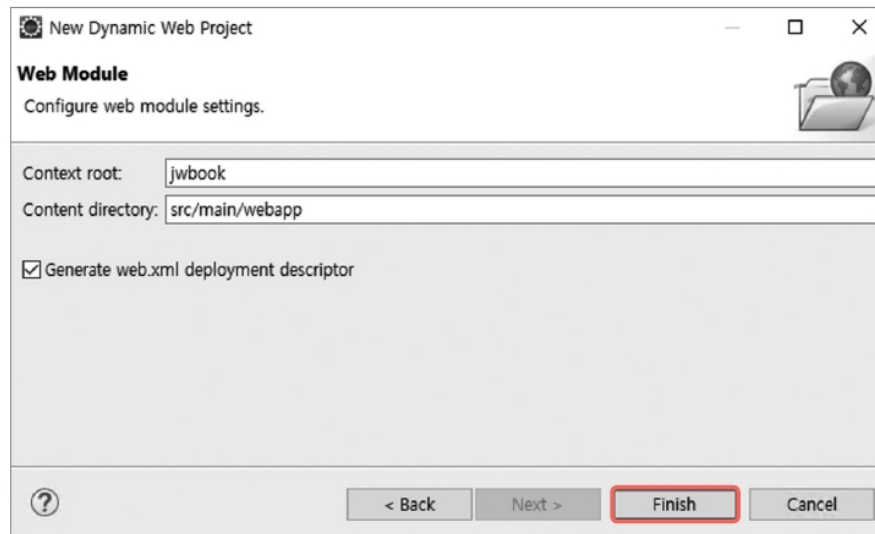
- ⑤ 설정이 완료된 [Dynamic Web Project] 창에서 <Next> 버튼을 클릭. 다음에 나오는 [Java] 창은 소스 폴더를 지정하는 화면으로 기본적으로 [src] 폴더를 사용하므로 <Next>를 클릭



동적 웹 프로젝트 생성

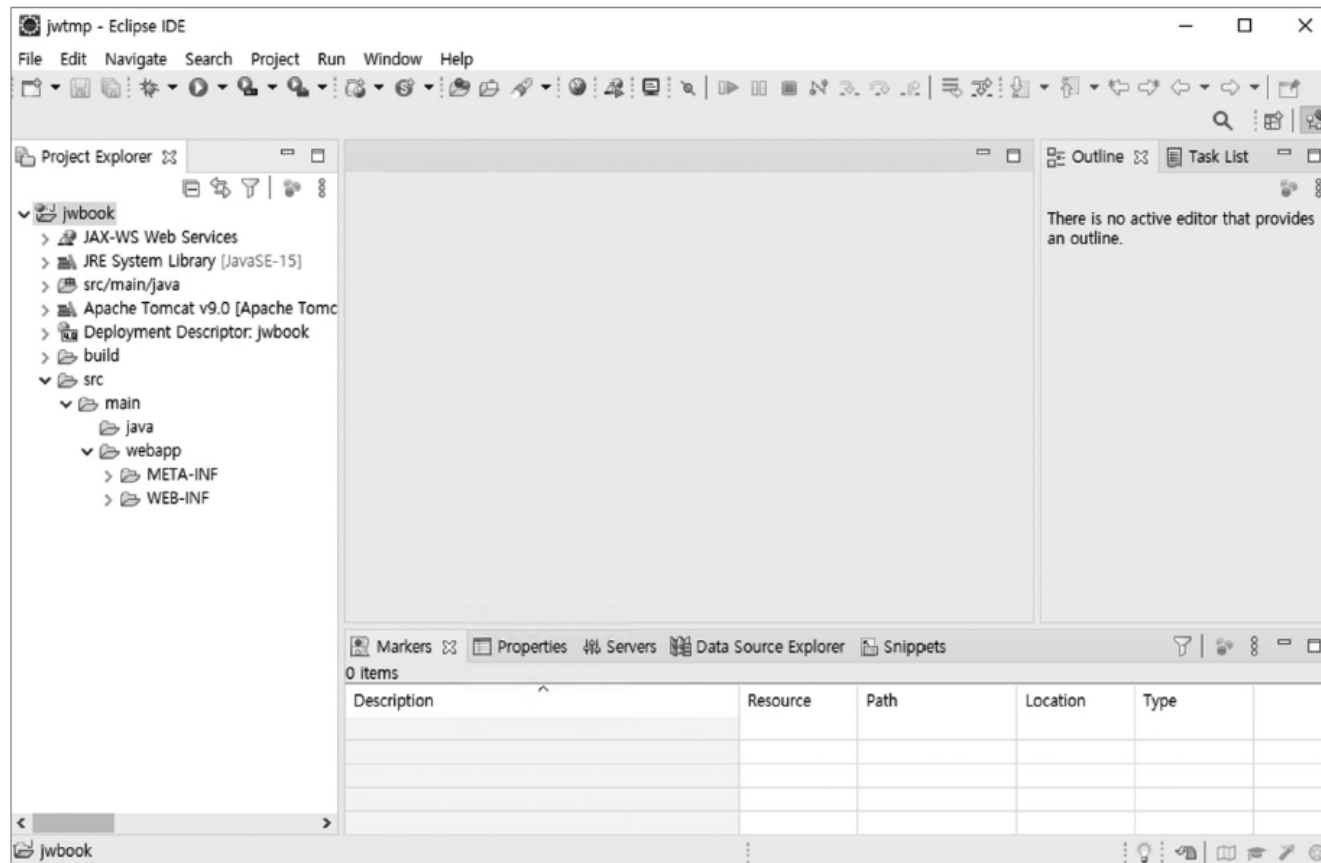
⑥ [Web Module] 창에서는 기본값을 사용하고 'Generate web.xml deployment descriptor'를 체크하고 <Finish>를 클릭하여 프로젝트를 생성

- **Context root** : 톰캣을 통해 실행할 때 사용되는 진입점임. 프로젝트 이름이 기본으로 사용됨
- **Content directory** : 프로젝트에서 웹 콘텐츠(html, css, jsp, 이미지)가 위치하는 경로를 지정
- **web.xml** : 웹 애플리케이션과 관련된 정보를 서블릿 컨테이너에 제공하기 위한 설정 파일로 옵션 사항



동적 웹 프로젝트 생성

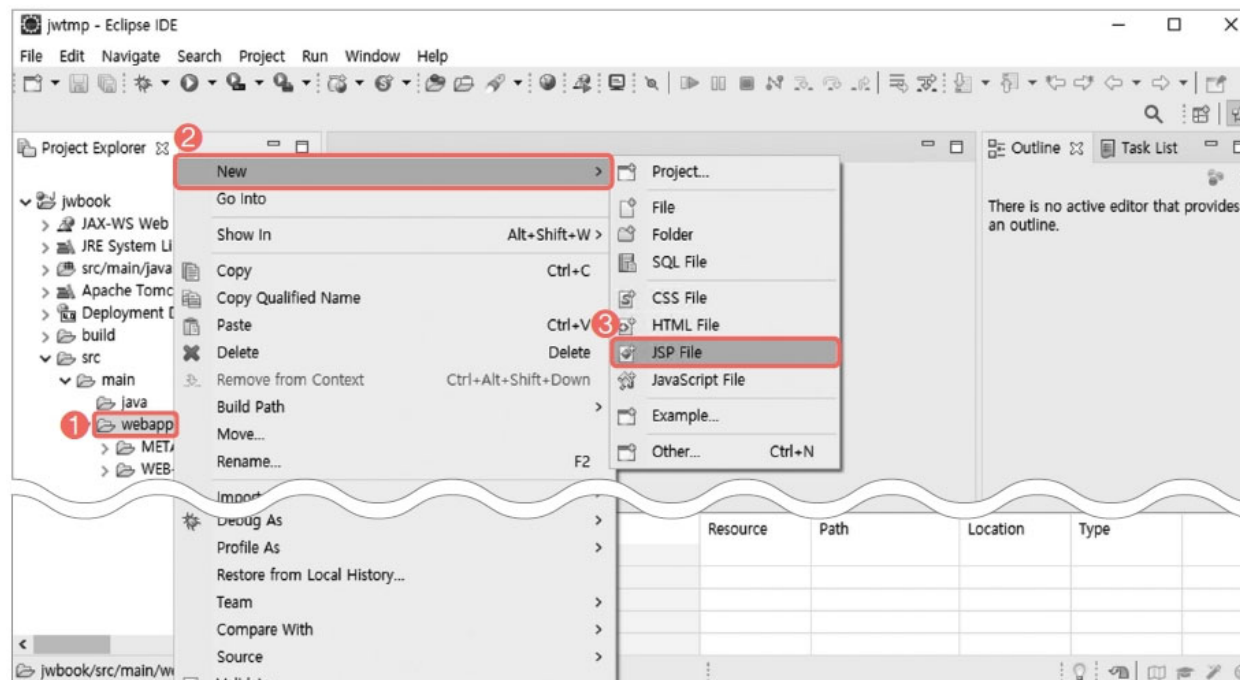
⑦ 생성된 프로젝트 화면은 다음과 같음



아파치 톰캣 연동과 JSP 실행

■ JSP 파일 만들기

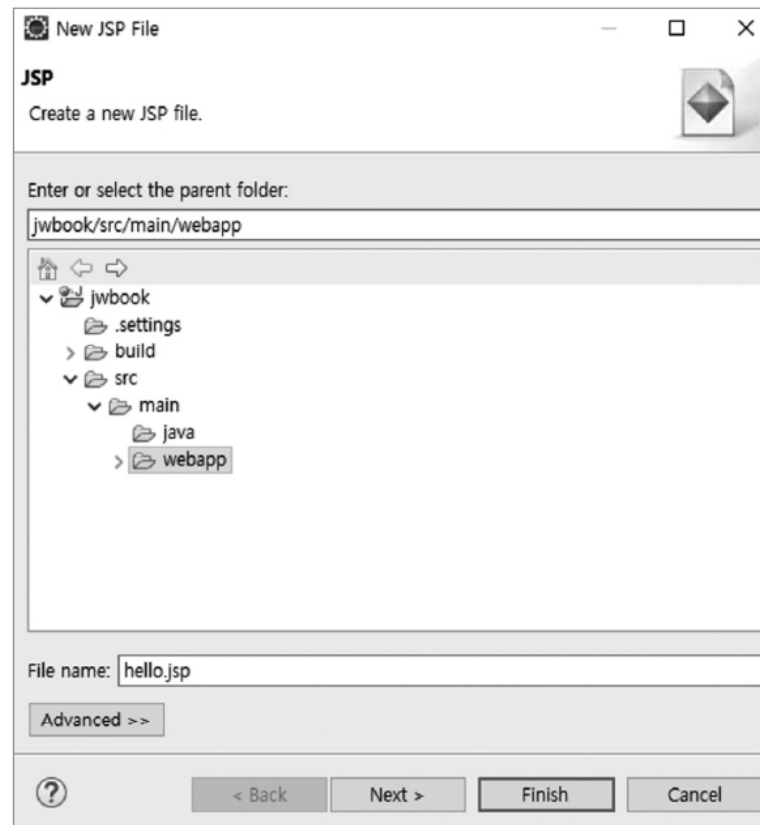
- ① [Project Explorer]의 프로젝트 폴더 구조에서 'jwbook' 하단의 [src] → [main] → [webapp] 폴더를 선택한 다음 마우스 오른쪽 버튼을 눌러 [New]→[JSP File] 선택



아파치 톰캣 연동과 JSP 실행

■ JSP 파일 만들기

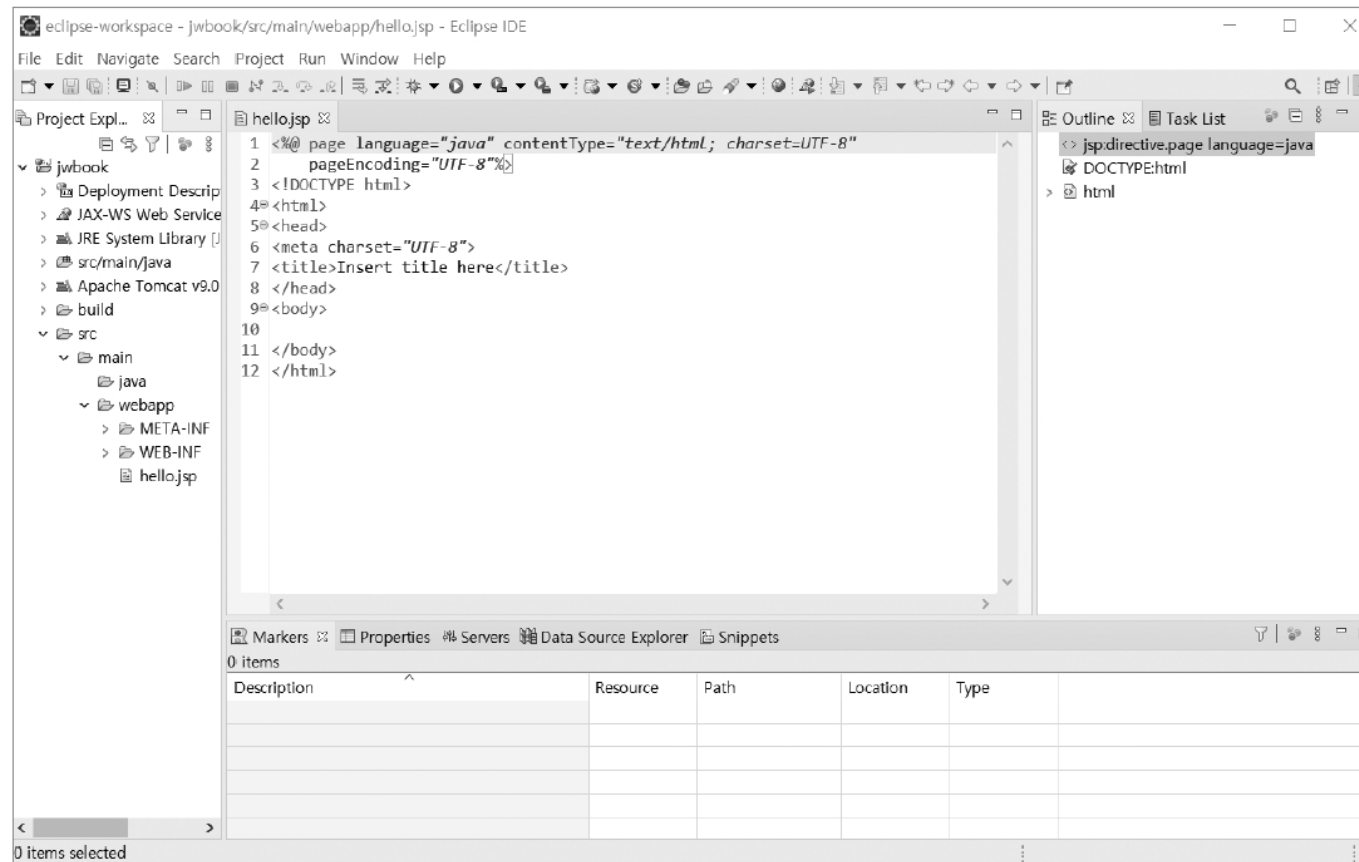
- ② 파일 이름에 'hello.jsp'를 입력하고 <Finish>를 클릭



아파치 톰캣 연동과 JSP 실행

■ JSP 파일 만들기

③ 생성된 'hello.jsp' 파일의 화면은 다음과 같음



아파치 톰캣 연동과 JSP 실행

■ JSP 파일 만들기

- ③ 생성된 'hello.jsp' 파일의 화면은 다음과 같음
- 'hello.jsp' 화면에 기본 생성된 코드를 완성하기

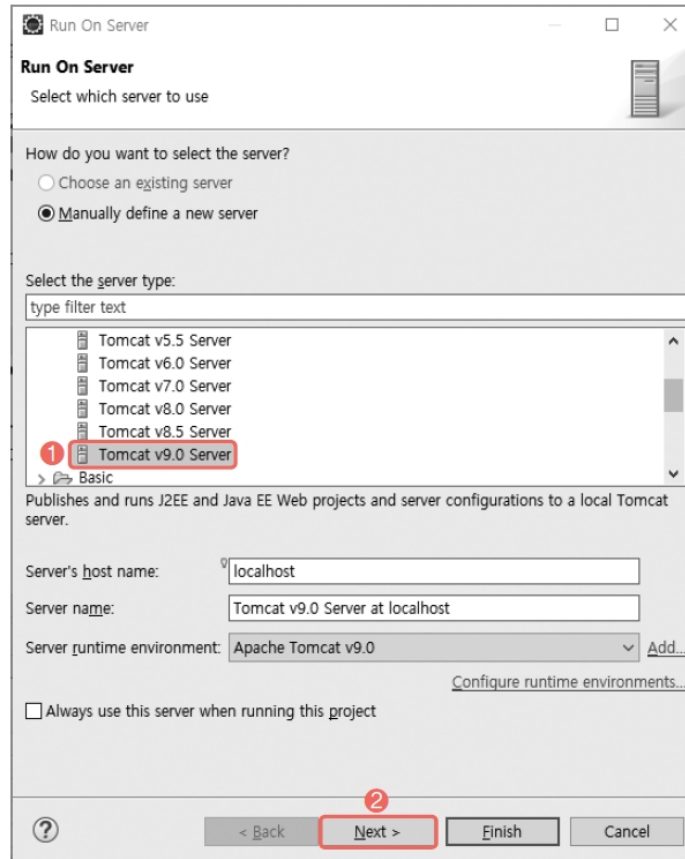
[예제 2-1]

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Hello World</title>
</head>
<body>
    <h2>Hello World</h2>
    <hr>
    현재 날짜와 시간은
    <%=java.time.LocalDateTime.now()%>
    입니다.
</body>
</html>
```

아파치 톰캣 연동과 JSP 실행

■ JSP 실행

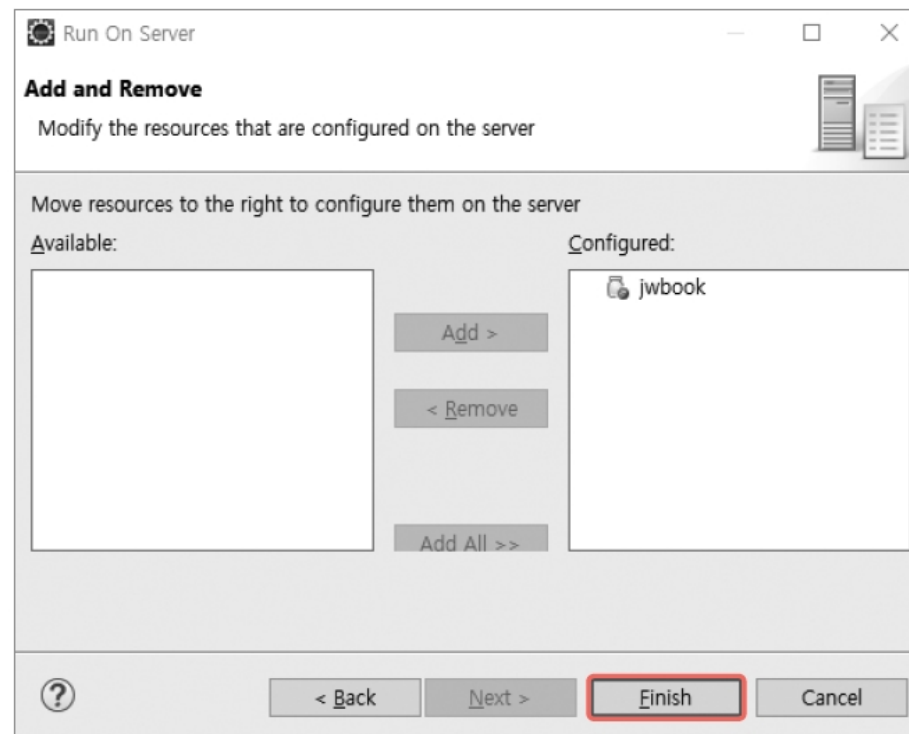
- ④ [Project Explorer]에서 'hello.jsp' 파일을 선택하고 마우스 오른쪽 버튼을 눌러 [Run As] → [Run On Server]를 선택. 실행을 위한 서버를 선택하는 화면에서 프로젝트를 생성할 때 설정한 'Tomcat v9.0' 서버와 'Server runtime environment'가 보임.
<Next>를 클릭



아파치 톰캣 연동과 JSP 실행

■ JSP 실행

- ⑤ <Configured>에 'jwbook' 프로젝트가 보임. <Finish>를 클릭하여 프로젝트를 실행



아파치 톰캣 연동과 JSP 실행

■ JSP 실행

- ⑥ 톰캣 서버가 실행되고 크롬 브라우저를 통해 다음과 같이 실행 결과를 볼 수 있음

