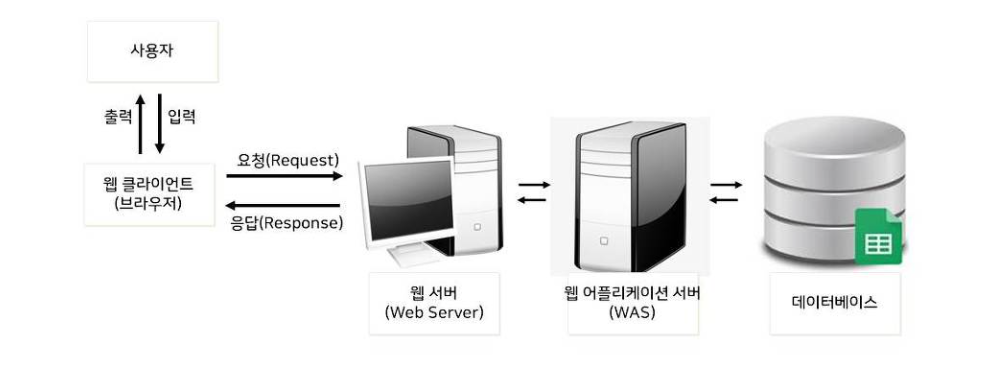
**Web Application Server(WAS) 동작 원리**



1. 회원페이지에서 회원정보를 입력 후, <확인> 클릭 (WEB BROWSER -> WEB SERVER)

2. 입력된 회원 정보를 전송 (WEB SERVER -> WAS(TOMCAT)

3. 입력된 회원정보를 읽어와서 데이터베이스에 저장(WAS -> DATABASE)

4. 회원가입 성공 실패 여부를 결과 값으로 획득(DATABASE -> WAS)

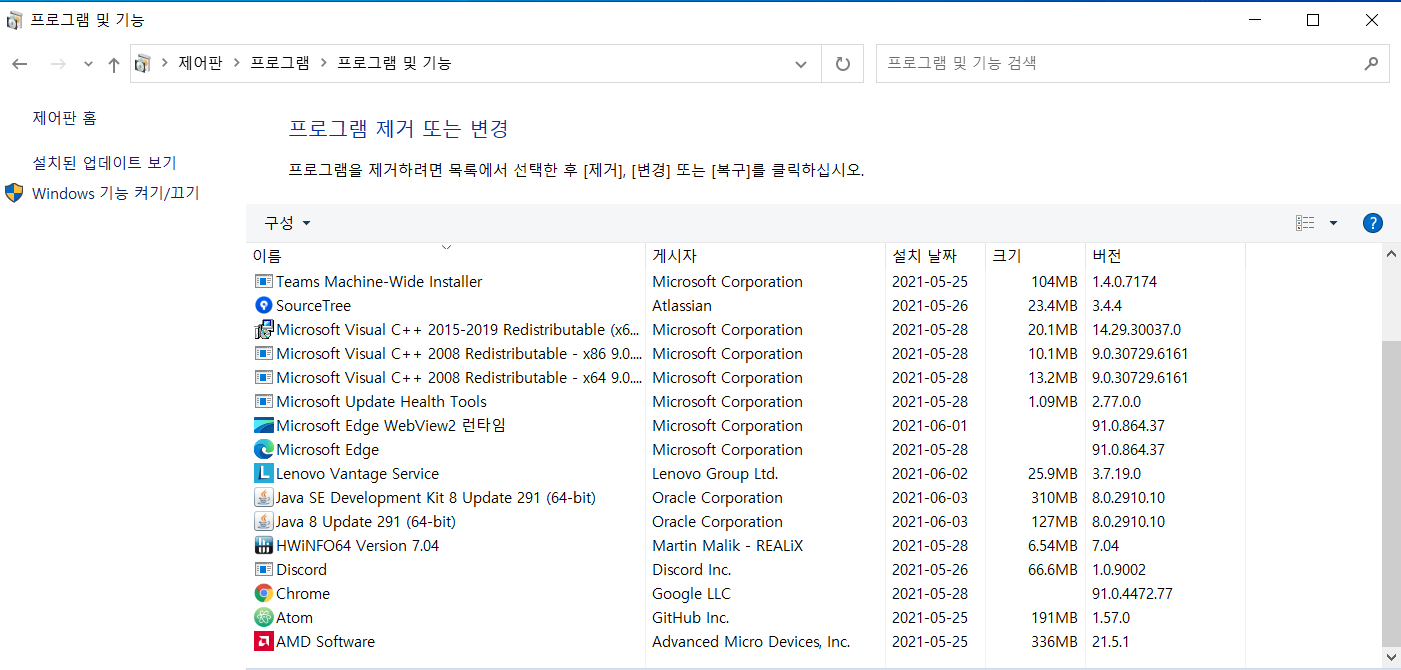
5. 회원가입 처리 결과 전송 (WAS -> WEB SERVER)

6. 입력된 정보를 다음 페이지에서 확인하기 위해서 출력 (WEB SERVER -> WEB BROWSER)

필수 목록

* JDK : JAVA DEVELOPER KIT 1.8
* TOMCAT : WAS (+ Web Server) 8.0 -< Eclipse 에서 설치
* Eclipse : IDE

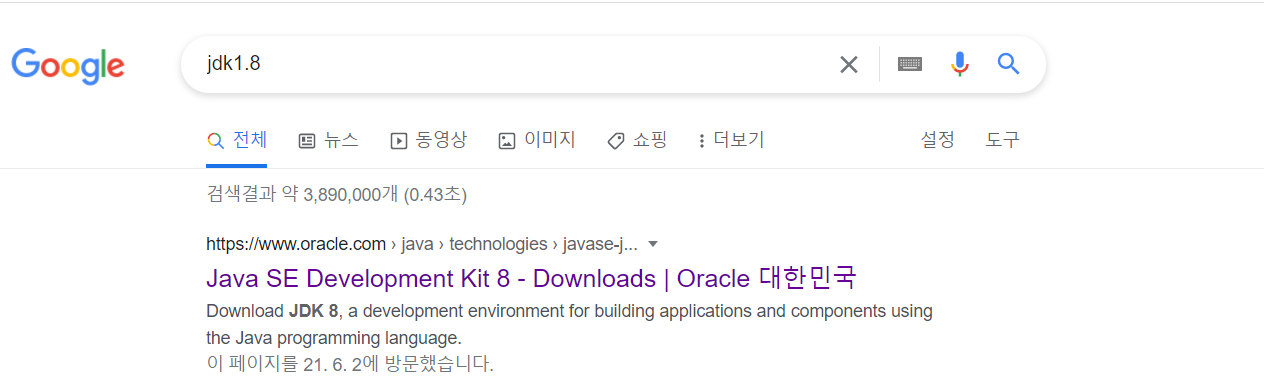
기존의 **JDK** 삭제 방법



**제거** 선택

윈도우 검색창에 **CMD** 클릭

C:\Users\사용자명(컴퓨터마다 다름)>**java -version** 입력하여 자바 삭제 확인 (생략 가능)



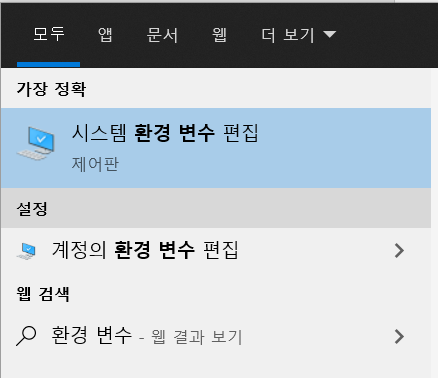
구글 검색창에 **jdk1.8** 검색 후 다운로드

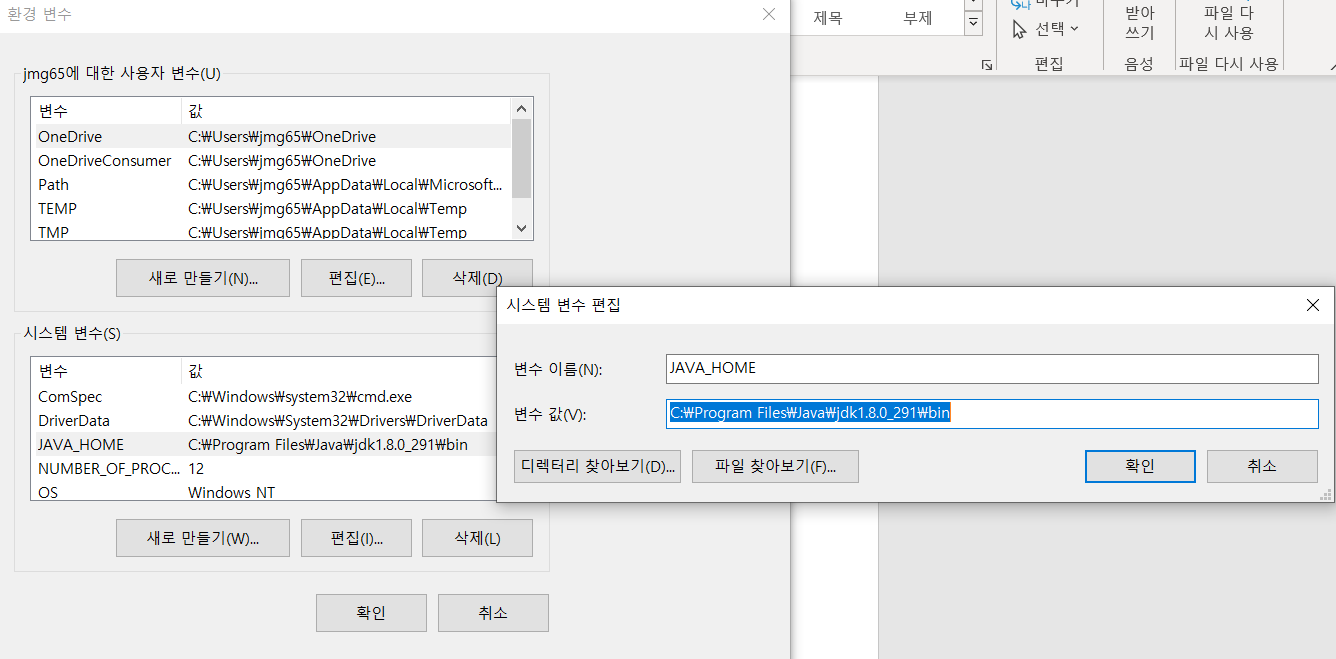
자바 다운로드는 기존 다운로드 방법과 같다. (설명 생략)

설치 후 이클립스 실행 시 실행 오류가 발생할 경우 환경변수 설정 필요

(저의 경우 JVM ~~~~ 오류 발생)

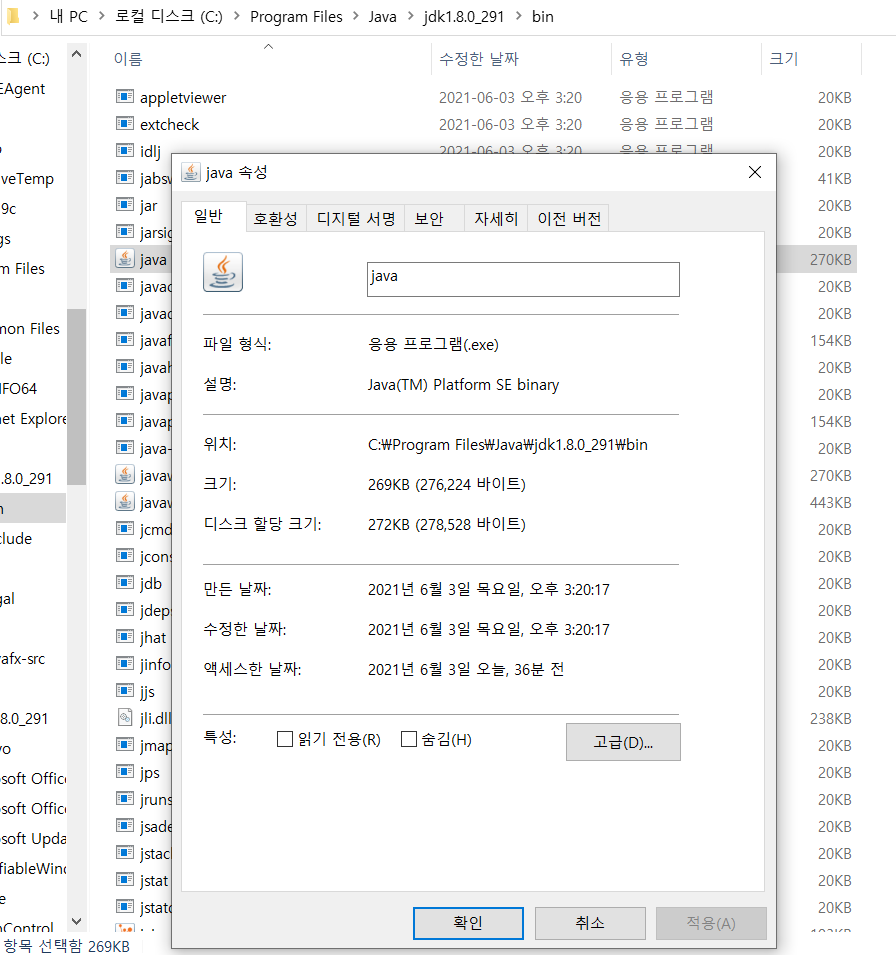
**환경변수** 설정





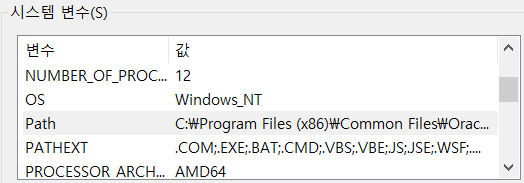
**환경변수** 탭에 진입 -> **시스템 변수** -> **JAVA\_HOME** 편집 (존재하지 않다면 새로 만들기)

변수 값은 **JDK** 경로 붙여넣기



변수 값 경로 예시

**JAVA\_HOME** 설정 이후 시스템 변수의 **PATH**도 설정



**PATH**의 경로 역시 **JAVA\_HOME**과 같은 변수 값을 설정해주면 됩니다.

기존에 이클립스가 설치되어 있을 경우

이클립스 삭제



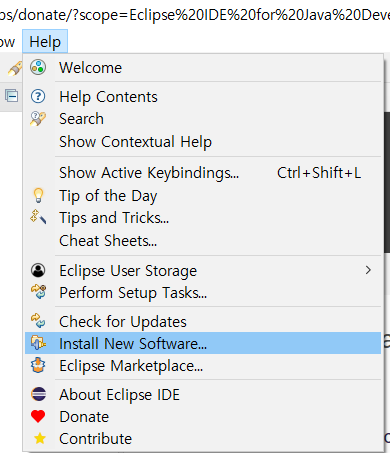
이클립스 폴더만 삭제해주어도 됩니다.

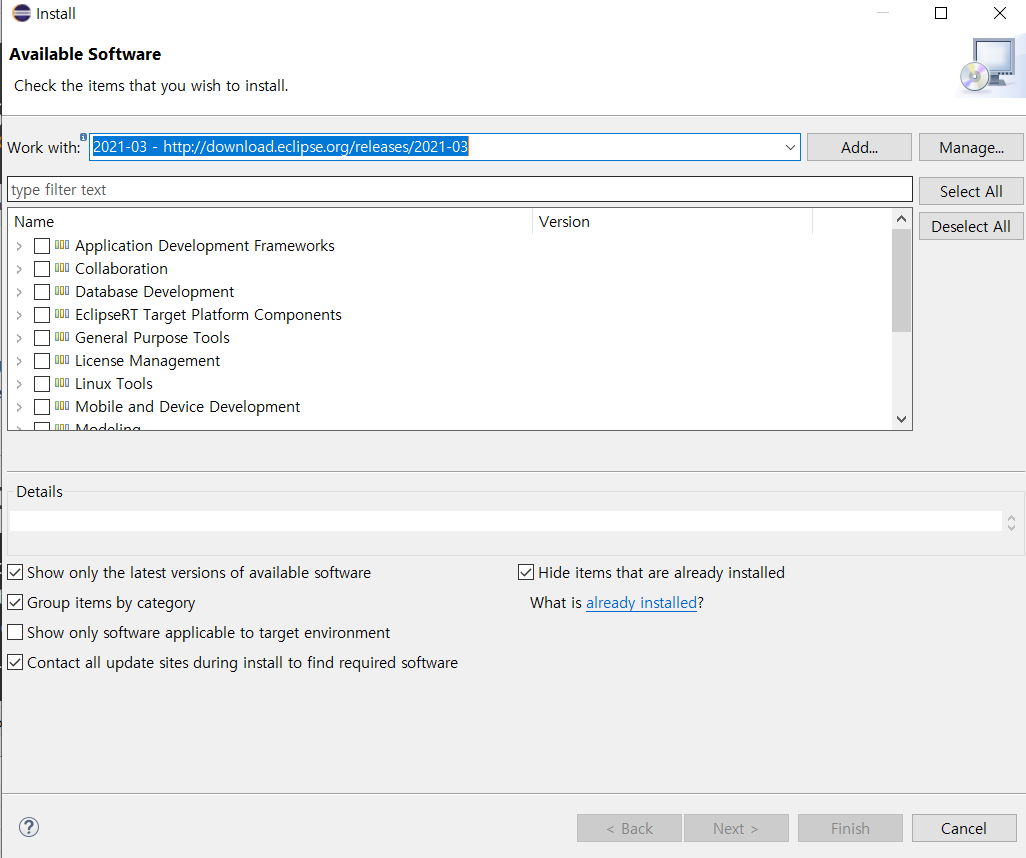
완전 삭제를 원할 경우 **.p2 , .ecilpse , eclipse , eclipse-workspace** 모두 지워주세요.

이후 다시 이클립스를 설치하면 실행이 가능할 것입니다.

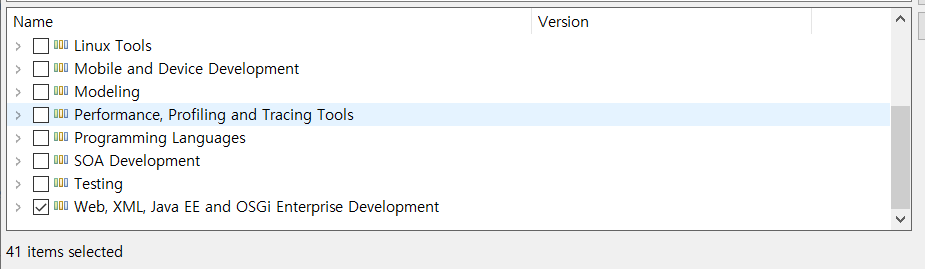
**TOMCAT 설치**

이클립스 메뉴의 HELP 선택



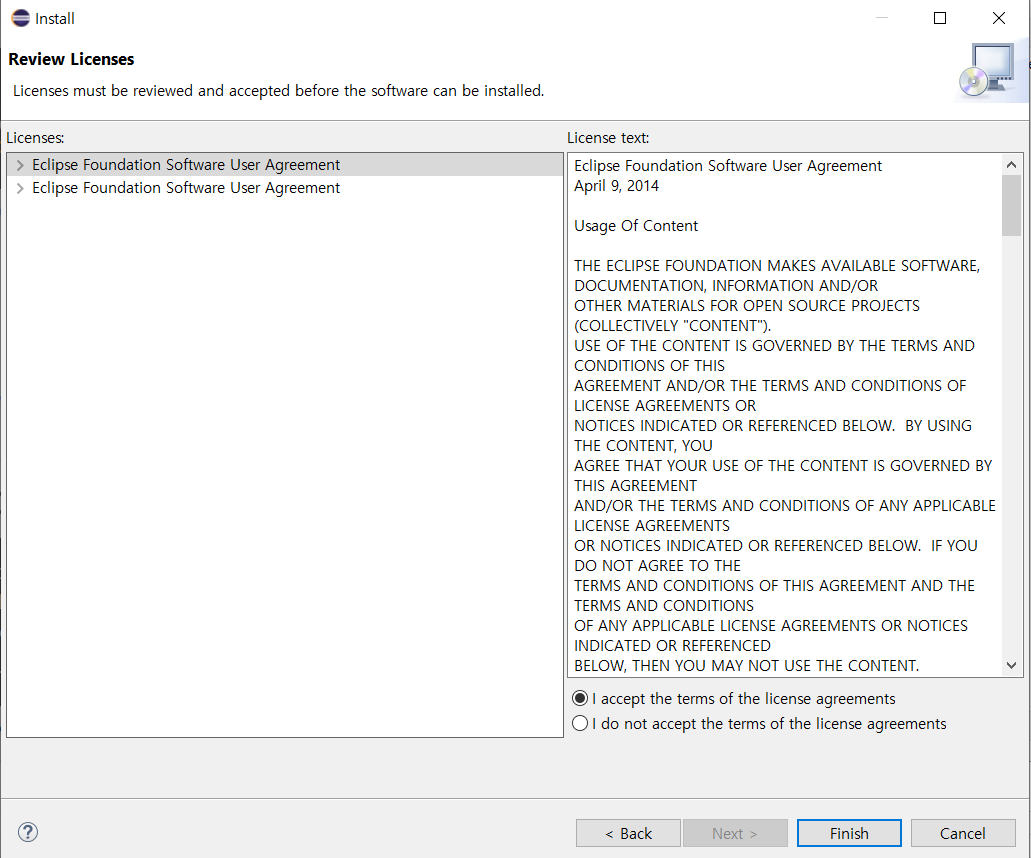


WORK WITH 부분에서 화살표 선택 후 본인의 **이클립스 버전**을 선택합니다.



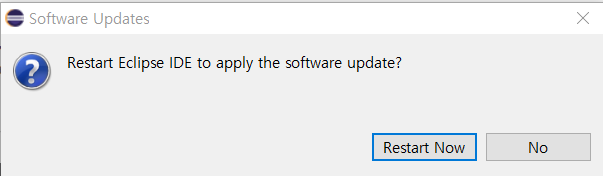
스크롤을 내려 **WEB, XML~** 선택하여 NEXT 선택

목록이 뜨면 NEXT 선택



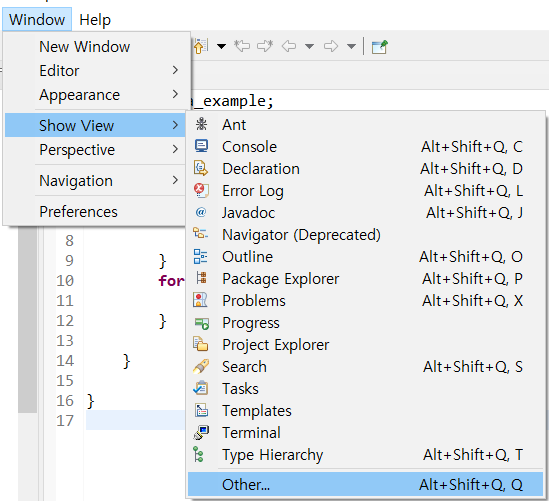
라이선스 동의 후 FINISH

설치완료 시

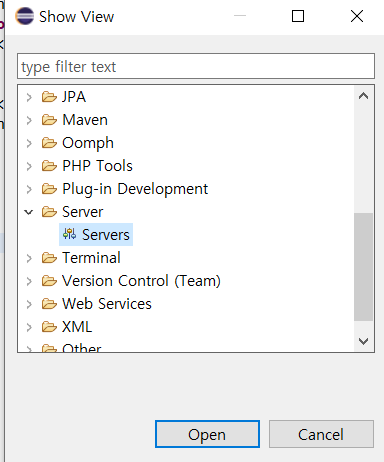


애플리케이션 재실행 클릭

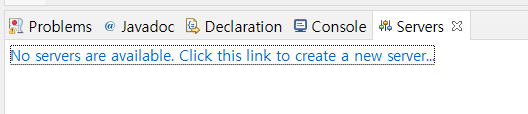
재실행 완료 후



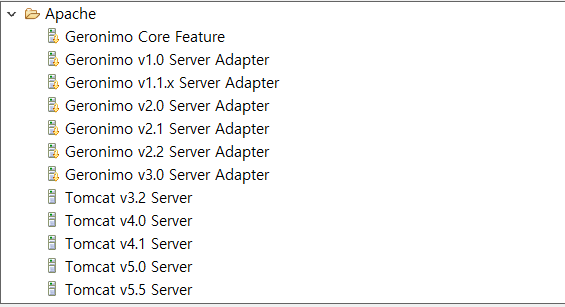
메뉴에서 Window 선택 후 Show View -> Other 클릭



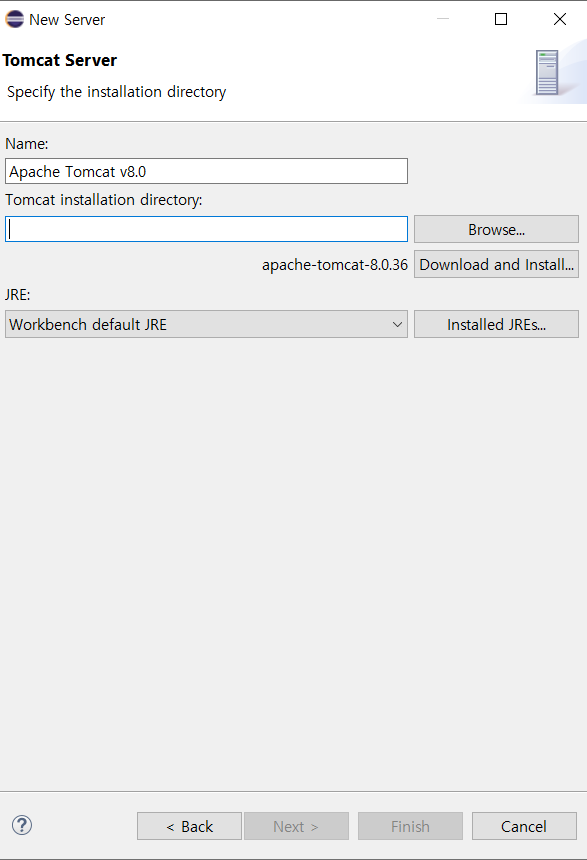
Server 폴더에서 Server 클릭 후 Open



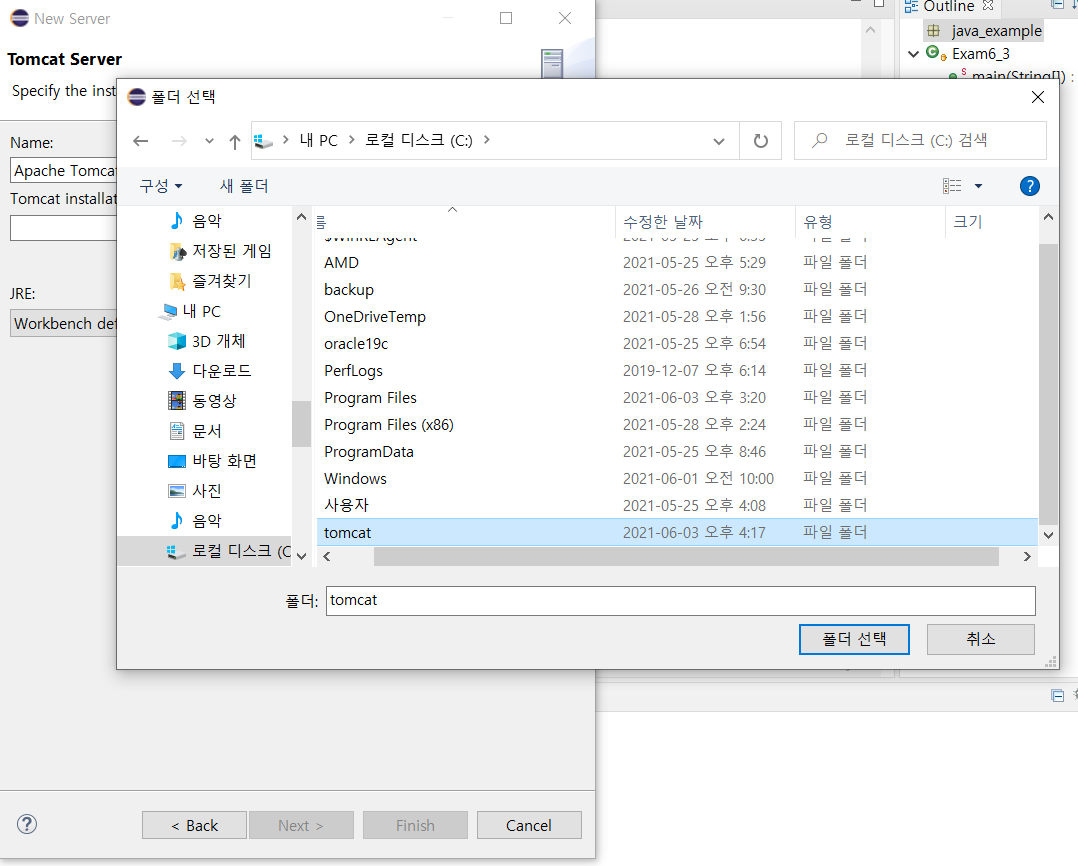
콘솔 창 옆 No servers are available ~ 선택



Apache 폴더 선택 후 Tomcat v8.0 Server 선택

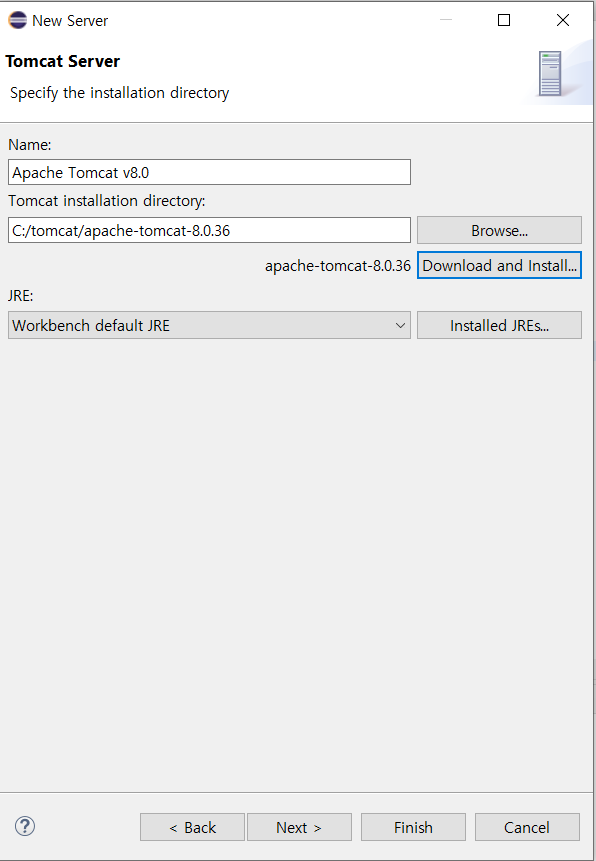


**Browser** 탭 아래에 **Download and install** 선택



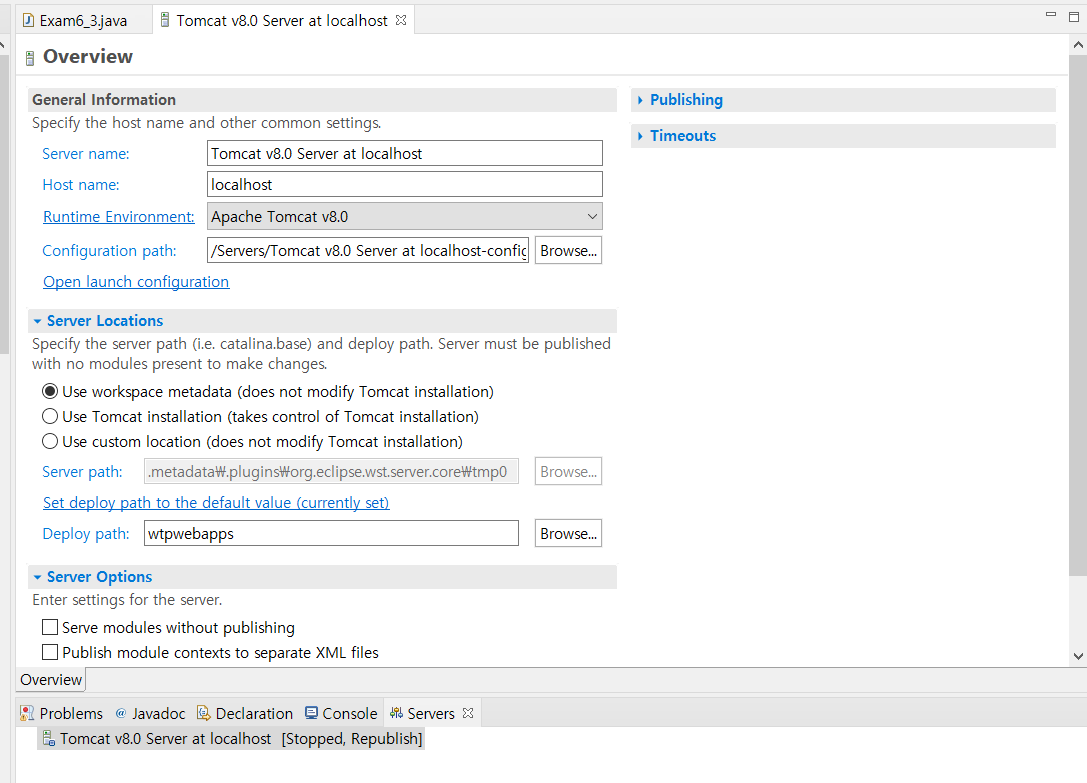
라이센스 동의 시 경로 설정 창이 나옵니다.

C드라이브에 **tomcat** 폴더 생성



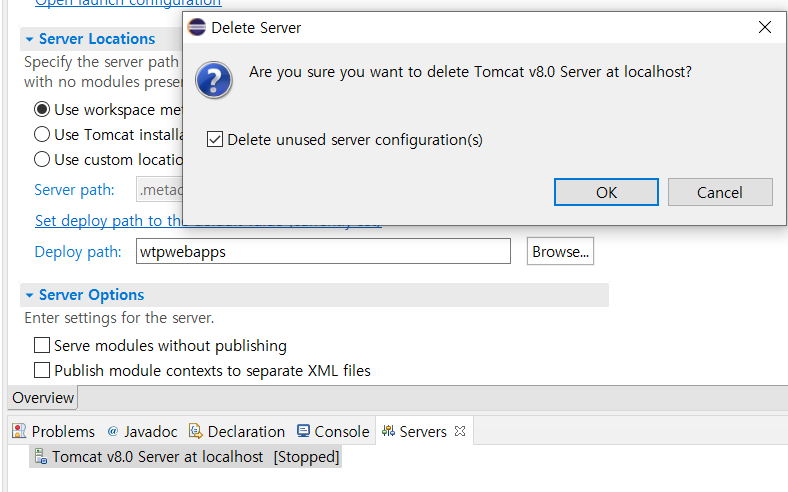
경로 설정이 완료된 화면

**Finish** 선택

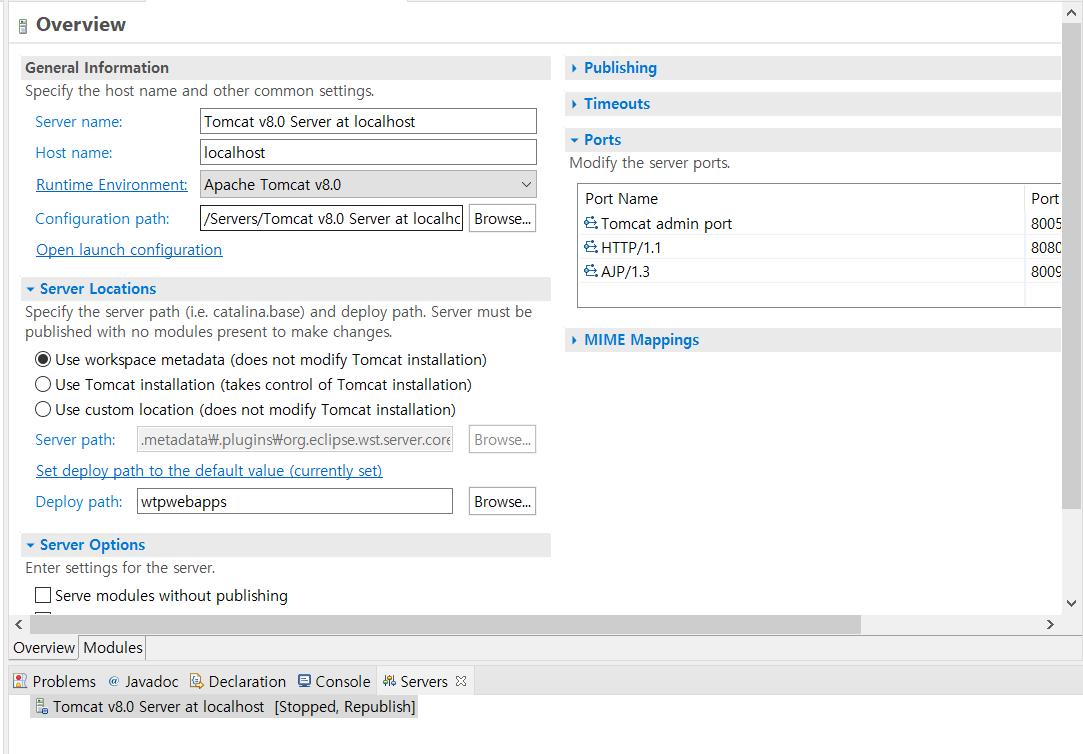


**Servers** 클릭 시 **Overview** 창 활성화

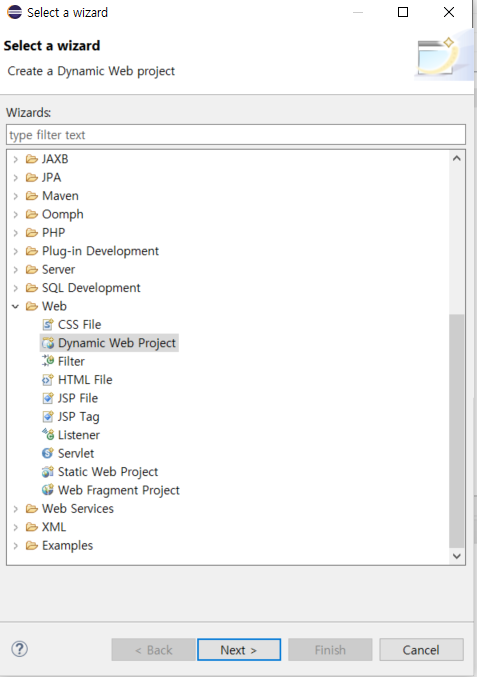
**Port** 탭이 뜨지 않을 경우



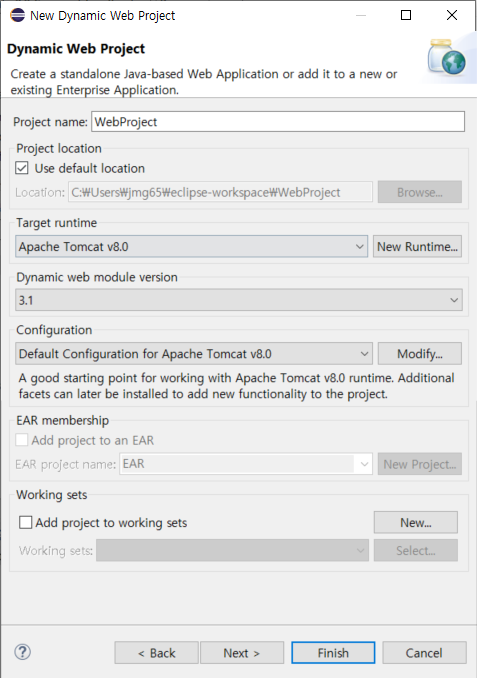
**Tomcat** 서버 삭제 후 다시 생성



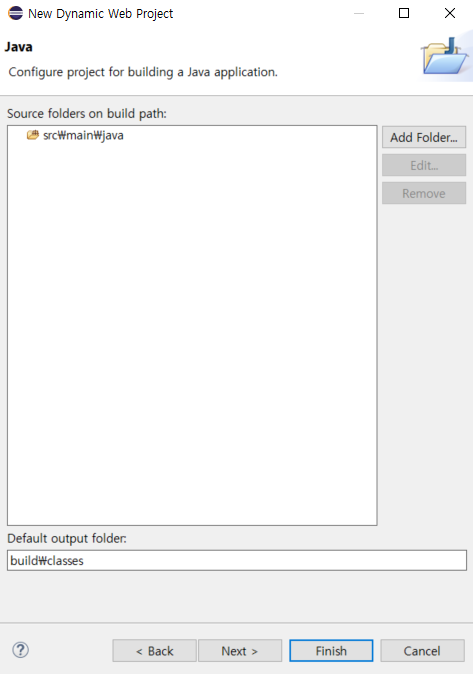
Port 탭이 생성된 것을 볼 수 있다.



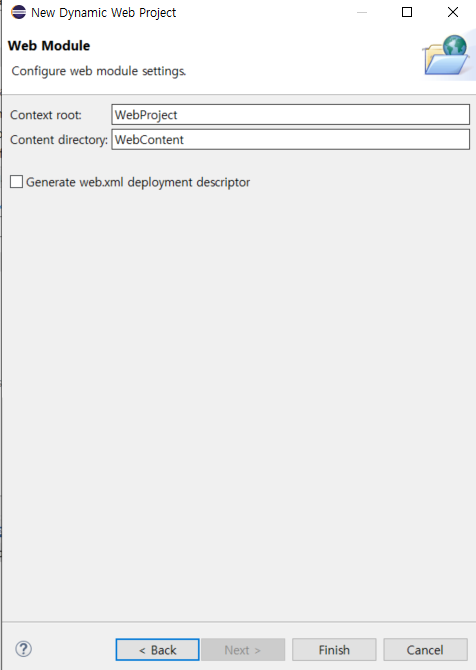
최상단 메뉴의 File -> New - > Others -> Web 폴더 내 Dynamic Web Project 선택



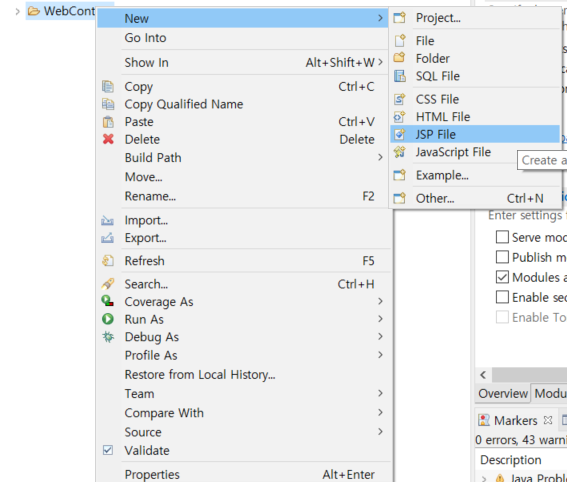
Project 생성 화면에서 Project name : WebProject 입력 후 next



Next 클릭



Context directory에 WebContent 입력 후 Finish



WebContent 우클릭 -> New -> JSP File 선택 -> index.jsp 생성

**Servlet (Server + Applet)**

자바를 이용하여 웹에서 실행되는 프로그램을 작성하는 기술

자바 클래스 형태의 웹 애플리케이션

Javax.servlet.http package HttpServlet class

**자바 코드 내부에 HTML 코드가 들어가는 구조**

**JSP (Java Server Page)**

자바로 서버 페이지를 작성하기 위한 언어

HTML + JSP tag(<% %> , 스크립트릿)

**HTML 문서 내부에 자바 코드가 들어가는 구조**

ServerPage – 웹 서버에서 실행되는 페이지로, 요청에 필요한 페이지를 위한 로직이나 데이터베이스와의 연동을 위해 필요한 것들을 포함.

Java Server Page

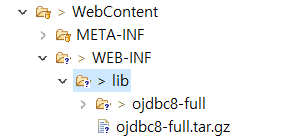
* 언어는 자바!
* 서버에서 실행되어야함

**ORACLE 데이터 베이스 연동**

* 자바 프로그램에서 데이터베이스 처리를 위해 제공 하는 도구 : JDBC
* JDBC : 자바로 만든 애플리케이션이 데이터베이스에 일관된 방식으로 접근할 수 있도록 API(LIBRARY)를 제공하는 클래스의 집합
* 연동절차 : JDBC Driver load –> Connect Database –> execute query –> Disconnect Database
* 인터페이스 : DriverManager –> Connection Interface -> statement Interface -> ResultSet Interface
* import java.sql.\*

1. JDBC Driver 로드 (프로그램 시작 시) – Database 업체에서 제공

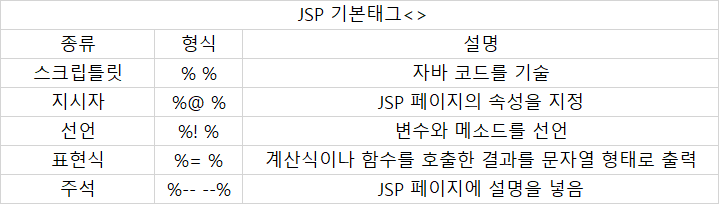
* JDK1.8 기준 ojdbc8.jar
* <https://www.oracle.com/kr/database/technologies/appdev/jdbc-downloads.html>



* 압축 파일을 압축 해제하여 WebProject\WebContent\WEB-INF\lib 에 위치시킨다.
* Ojdbc8-full 폴더 내 ojdbc8.jar : 자바와 데이터베이스를 연결할 때 사용할 API를 포함
* 
* 이용하여 드라이버 로드

1. DriverManger Class 를 이용하여 데이터베이스에 접속

JSP TAG

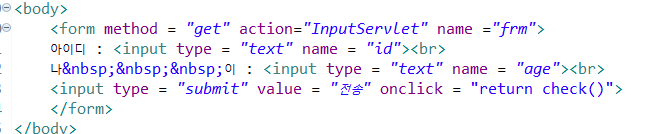


**<Queary String>**

서버에서 client 가 보낸 데이터를 획득하기 위해 사용하는 기술

**get 방식**

**form tag 의 method = “get”**

****

URL 주소 뒤에 입력 데이터를 함께 제공

**형식** : “ resource?이름=값 “

**예시**

Ex ) 네이버에 abc 123 주민기 검색

(& 기준으로 URL을 자른 것임)

**URL :** <https://search.naver.com/search.naver>

**QS :** ?where=nexearch

&sm=top\_hty

&fbm=0

&ie=utf8

&query=abc+123+%EC%A3%BC%EB%AF%BC%EA%B8%B0

abc 123 의 공백의 경우 +로 구분, 한글의 경우 16진수로 변환하여 표시, 쿼리 스트링은 & 기호로 구분됨

이 문장은 총 다섯개의 쿼리 스트링으로 이루어져 있다.

<내장 객체 영역>

**객체의 유효기간**

Page - 하나의 jsp 페이지를 처리할 때 사용되는 영역

Request - 하나의 요청을 처리할 때 사용되는 영역

Session - 하나의 브라우저와 관련된 영역

Application - 하나의 웹 애플리케이션과 관련된 영역

**Page 영역**

pageContext 내장 객체 할당 : 해당 페이지 내에서 정보를 사용

**request 영역**

URL 입력 또는 페이지 링크를 클릭했을 때 브라우저가 웹 서버에 전송하는 요청

request 내장 객체 정보 저장

브라우저가 결과를 받으면 request 내장 객체 소멸

하나의 요청을 처리하는데 사용되는 모든 JSP 페이지를 포함.

request.getParameter(String name)

setAttribute(String name, Object value) 파라미터가 아닌 원하는 정보도 저장해두면 getAttribute() 사용하여 가져올 수 있다.

**session 영역**

웹 브라우저를 닫기 전까지 페이지를 이동하더라도 사용자 정보를 잃지 않고 서버에 보관 가능 객체

로그인 인증 처리

하나의 브라우저와 관련된 영역

application 영역

하나의 웹 어플리케이션과 관련된 전체 영역 포함

page + request + session + ...

application 내장 객체 공유

<Action Tag>

<jsp:forward> : 다른 사이트로 이동할 때 사용

<jsp:include> : 자원을 현재페이지의 내용에 포함

<jsp:param> : 파라미터를 추가할 때 사용

<jsp:useBean> : 빈(Bean)을 생성하고 사용하기 위한 환경을 정의

<jsp:setProperty> : 빈에서 속성 값을 할당

<jsp:getProperty> : 빈에서 속성 값을 얻어올 때 사용

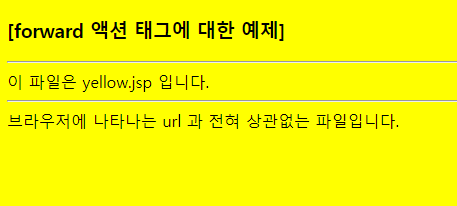
<jsp:...속성=값></jspp:...>

<jsp:forward page=”yellow.jsp”>

<jsp:param name = “url” value=”red.jsp”/>

<jsp:param name = “news” value=”Happy New Year!”/>

</jsp:forward>



jsp:param 단독 사용 불가 -> include or forward 안에서 사용 가능하다.

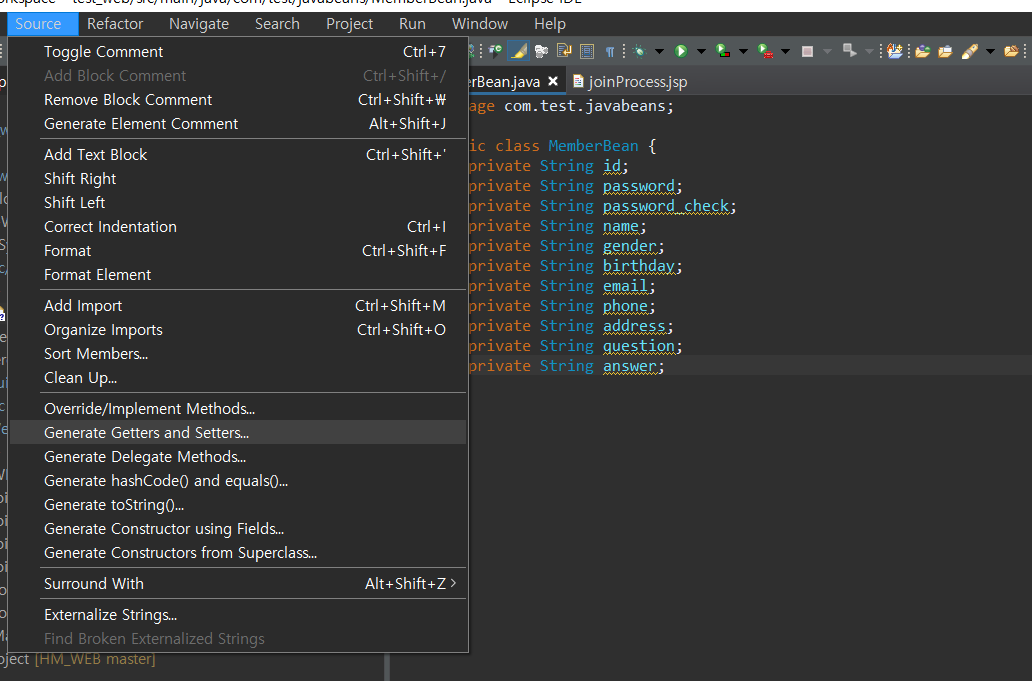
**javaBean**

javabean <- loginForm에서 효율적으로 사용 가능

회원정보를 하나의 묶음으로 관리하기 위해서 사용

은닉화

getter, setter method <- javaBean 데이터를 저장하거나 조회



getter setter 생성

<%@ page import=”com.test.javabeans.MemberBean”%>

<%

MemberBean member = new MemberBean();

%>

<jsp:useBean class = “com.test.javabeans.MemberBean” id=”member” scope=”request”/>