0330 JSP 보강

Maven

- 빌드관리 툴

  ⇒ 빌드?

- 교재 110페이지
- 개발
- 하나의 어플리케이션이 완성되는 과정

1. 소스코드 컴파일
2. 컴파일 돼서 만들어진 걸 대상으로 패키징
3. 파일조작

   ⇒ 배타과정 ⇒ 테스트

   (실제로 설계에 맞게 잘 구현되었는지)
4. 배포

⇒ 이 과정을 빌드라고 함

빌드를 구성할 수 있는 공정 단계하나하나를 페이즈(phase)?

메이븐 : 빌드 공정 관리 툴

⇒ 메이븐은 각각의 페이즈를 관리?

신입 개발자 입장에서 메이븐이 나 대신 뭘 해주는지

1. 템플릿 프로젝트(프로젝트의 틀) 만들어준다

   (개발할 수 있는 환경을 자기가 아예 템플릿 구조로 만들어줌)
2. 의존성을 대신 관리해준다

   (메이븐의 dependency 관리)

⇒ 이 두 과정에서 maven이 사용하는 중요한 개념 ⇒ 저장소

⇒ 우리가 10개의 라이브러리 사용함

⇒ 만약 마트 없으면 따로따로 돌아다니면서 받아야 함

⇒ 근데 만약 큰 창고가 있다면? 거기서만 찾으면 됨

⇒ 이게 maven이 사용하는 central repository?

⇒ 중앙저장소에서 사기만 한다고 되는 게 아님

⇒ 그걸 내 로컬 저장소에 저장해야 함

⇒ 내 로컬 저장소가 필요

```
<localRepository>D:/B_Util/6.maven/.m2/repository</localRepository>
```

⇒ 이게 우리집 냉장고 위치

⇒ 이제부터는 중앙저장소에 필요하 거 찾은 다음 여기 넣기만 하면 됨

⇒ 엄청 큰 창고 : 찾을 때 오래걸림

⇒ 넣을 때 분류해서 넣어놓으면 더 빨리 찾을 수 있음

⇒ 실제로 그걸 가져오는 게 아니라 어디에 뭐가 있다는 메타 정보만 가져와서 목차를 가져옴

⇒ 그럼 검색이 가능해짐

⇒ 이렇게 트리구조로 되어 있음

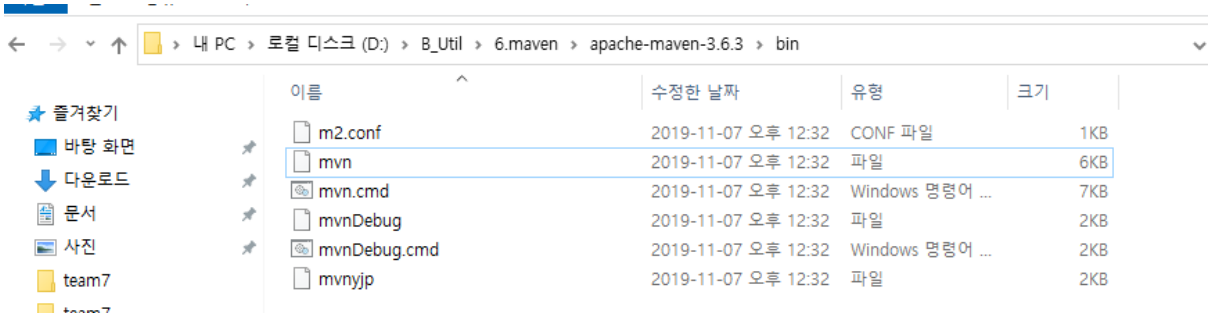
(indexing이 되어 있다)

메이븐 설치과정, 기본개념 ⇒ 교재

Welcome to Apache Maven

Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

⇒ Maven홈페이지에 maven이 뭐하는 앤지 써있음



⇒ 실행명령어로는 딱 mvn하나 쓰인다

→ 이걸 어디서든 쓰려면 path 설정해야 함

```
C:₩Users₩PC-14>set m2_home
M2_HOME=D:₩B_Util₩6.maven₩apache-maven-3.6.3

C:₩Users₩PC-14>path
PATH=D:₩B_Util₩4.Oracle₩appClient;D:₩B_Util₩4.Oracle₩app₩oracle₩product₩11.2.0₩server₩bin;C:₩WINDOWS₩system32;C:₩WINDOWS;
C:₩WINDOWS₩System32₩wbem;C:₩WINDOWS₩System32₩WindowsPowerShell₩v1.0₩;C:₩WINDOWS₩System32₩OpenSSH₩;D:₩B_Util₩2.Java₩JDK1.8
₩bin;C:₩Program Files (x86)₩Brackets₩command;C:₩Program Files₩Git₩cmd;C:₩Program Files₩TortoiseSVN₩bin;D:₩B_Util₩6.maven₩
apache-maven-3.6.3₩bin;C:₩Program Files₩MySQL₩MySQL Shell 8.0₩bin₩;D:₩anaconda3;D:₩anaconda3₩Library₩mingw-w64₩bin;D:₩ana
conda3₩Library₩usr₩bin;D:₩anaconda3₩Library₩bin;D:₩anaconda3₩Scripts;C:₩Users₩PC-14₩AppData₩Local₩Microsoft₩WindowsApps;C
:₩Program Files₩Bandizip₩;C:₩Users₩PC-14₩AppData₩Local₩GitHubDesktop₩bin;C:₩Program Files₩MongoDB₩Server₩3.4₩bin;
```

⇒ path에 이렇게 설정되어 있음

⇒ 어디에서든 mvn명령어 쓸 수 있다

```
C:₩Users₩PC-14>mvn -version
Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)
Maven home: D:₩B_Util₩6.maven₩apache-maven-3.6.3₩bin₩..
Java version: 1.8.0_121, vendor: Oracle Corporation, runtime: D:₩B_Util₩2.Java₩JDK1.8₩jre
Default locale: ko_KR, platform encoding: MS949
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

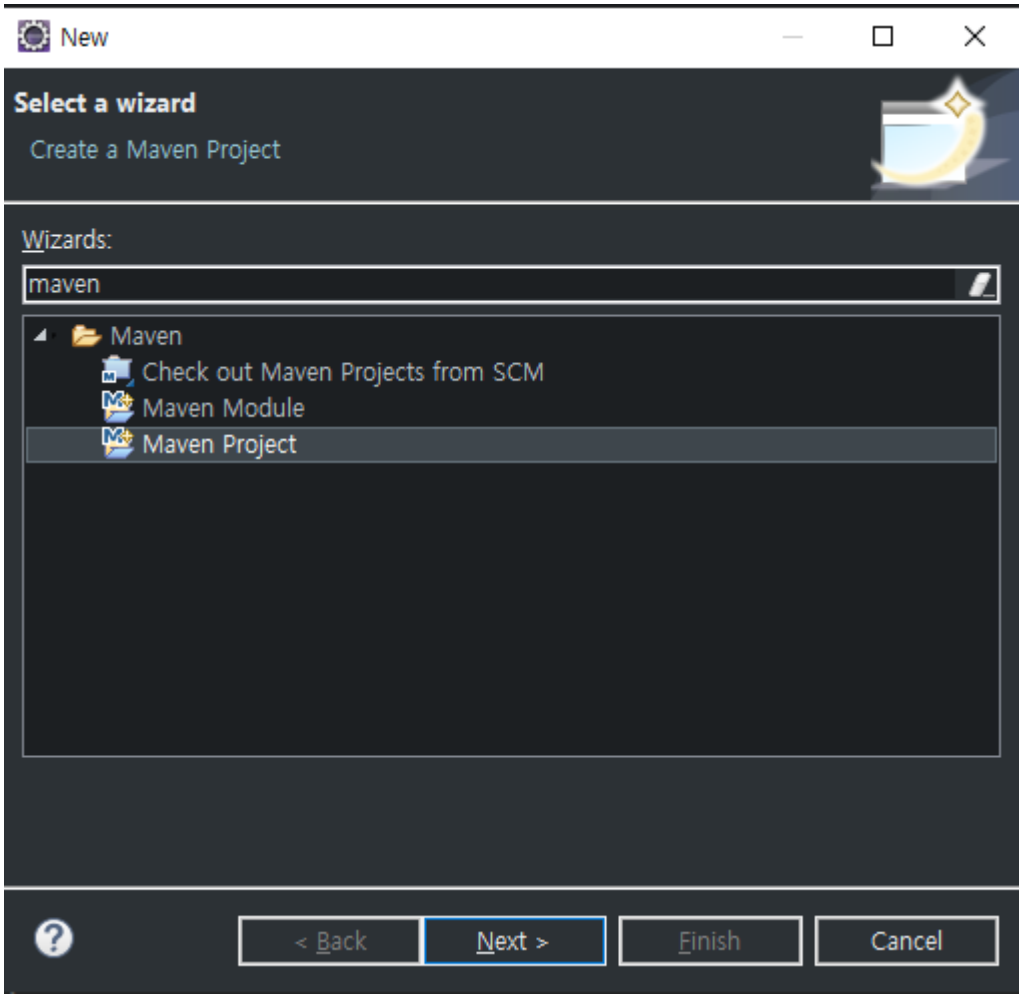
⇒ 이렇게 보면 maven 버전 정보 확인 가능

(maven 쓸 수 있다)

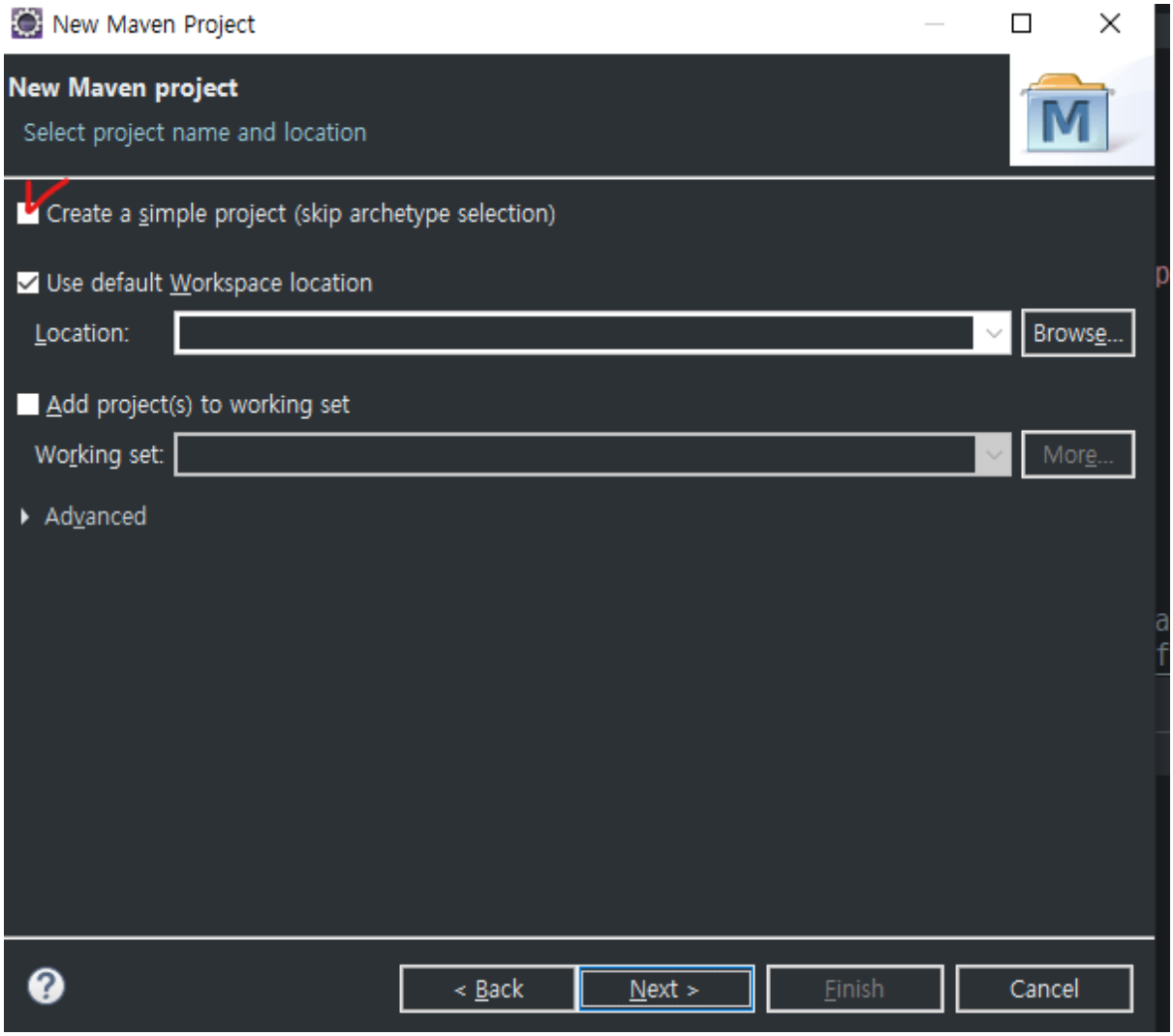
⇒ 근데 CUI에서 할 수 없으니까 이클립스에 얹어줌

(저번에 이클립스에서 설정했던 것들)

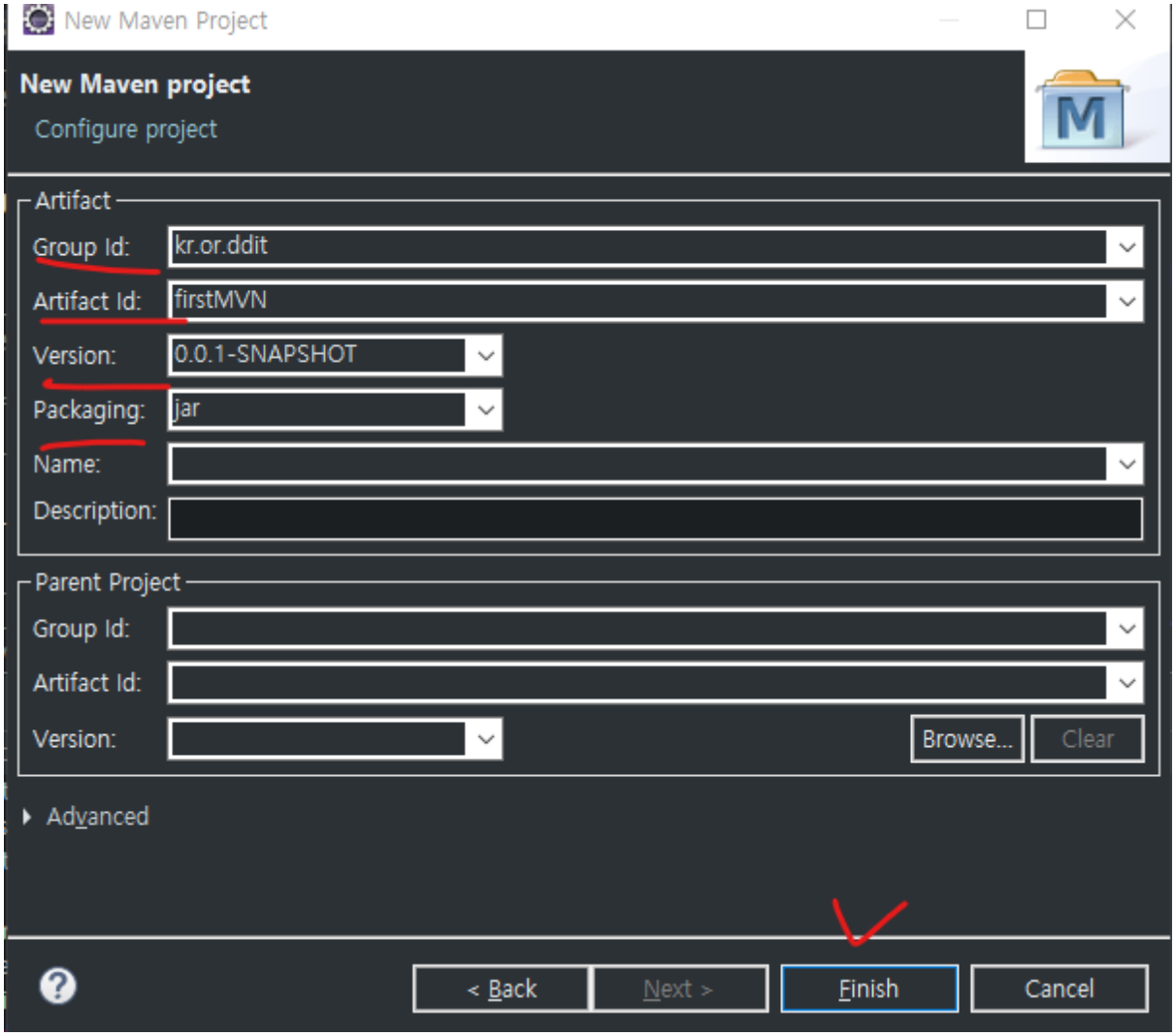
Maven Project만들기



⇒ 그냥 프로젝트


New Maven Project
New Maven project
Select project name and location
Create a simple project (skip archetype selection)
Use default Workspace location
Location:
Browse...
Add project(s) to working set
Working set:
More...
Advanced
< Back
Next >
Finish
Cancel

⇒ 우리는 이거 skip



→ Group Id:분류값으로 사용될 수 있게 하는 것

(보통 사이트의 도메인 이름 거꾸로)

대분류: kr / 중분류 : or / 소분류 : ddit

→ Artifact Id : 제품명

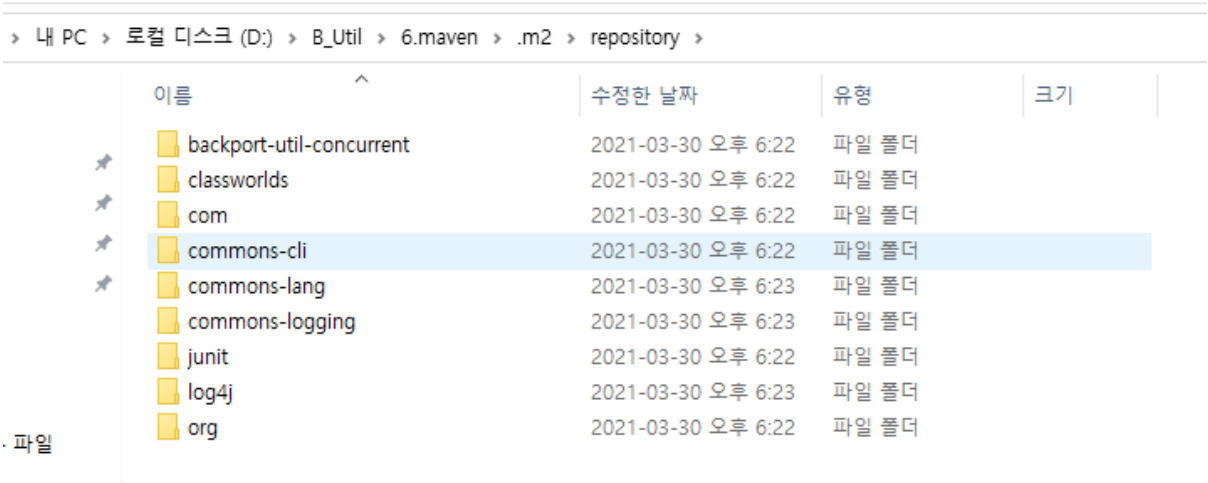
→ Version : 우리 어플리케이션 계속 유지보수 해야 하니까 그거

→ Packaging: 압축방식(jar vs war)

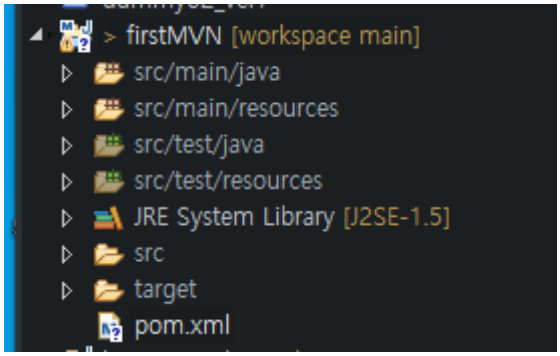
→ 우리는 standalone ⇒ jar

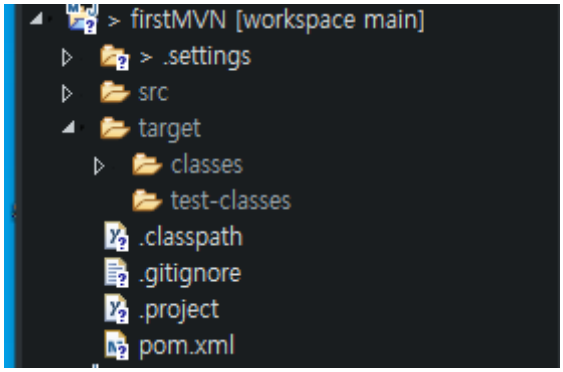
⇒ M되어 있는 애 ⇒ Maven이라는 빌드 관리 툴이 관리하는 프로젝트

%m2_home%



⇒ 필요한 것들은 maven이 알아서 가져옴





⇒ 이렇게 다 돌아감

⇒ 개발 영역을 main(주 개발영역)과 test(테스트 영역)으로 나눔

→ 이게 템플릿해주는 것

⇒ packaging의 대상이 되는 것? main

(maven은 아예 배포용과 test용 분리함)

⇒ java 소스 개발해야하면? src/main/java에

⇒ ibatis이용해야 함 ⇒ src/main/resources

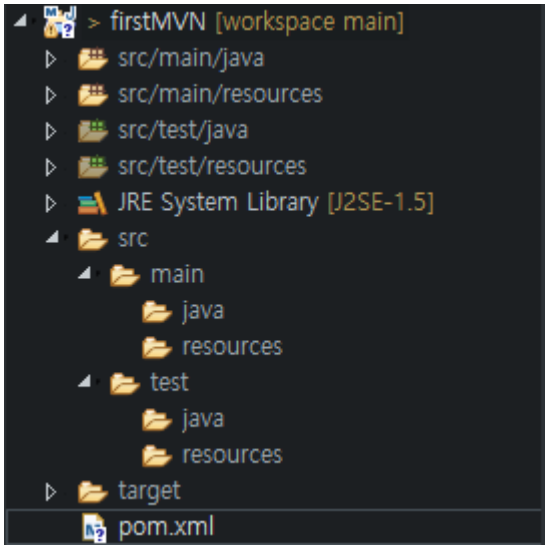
⇒ 테스트해야함 ⇒ src/test/resources

⇒ 사용 목적이 다 다른 클래스 패스가 생김

⇒ 클래스패스가 2개로 분리됨(main / test)

⇒ 어디에?

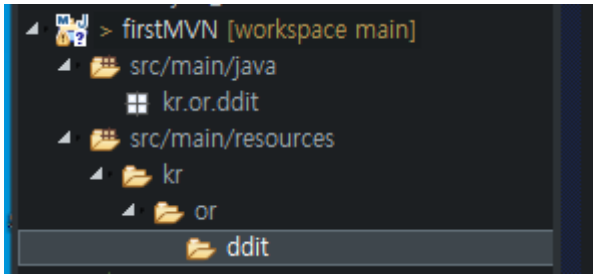
⇒ target은 지금까지 봐왔던 build와 같은 역할



⇒ 사실 위에 있는 것들은 maven이 가지고 있는 plugin이 보기 좋게 만드는 것

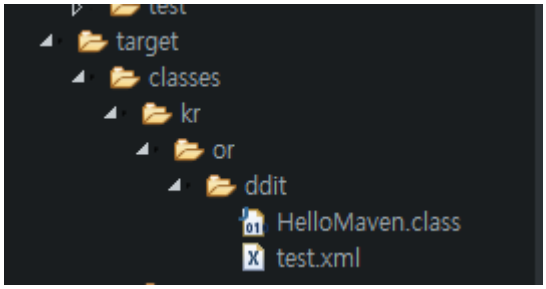
⇒ 개발할 때는 위에 있는 애들로 해야 함(package모양인 것?)

(밑에 있는 건 navigator ⇒ 그냥 경로 쉽게 보여주는 것)

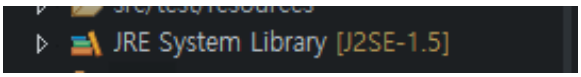


⇒ java 소스 ⇒ 위

⇒ 소스 외의 것들 ⇒ 밑에 폴더



⇒ src/main/java와 src/main/resouces는 같은 위치에 있는 걸 따로 분리해서 보이는 것 뿐(템플릿프로젝트)



⇒ 보면 java 컴파일러 버전이 1.5로 되어 있음

(우리는 1.8버전 사용)

⇒ 우리가 썼던 try~어쩌구 그런거 못 쓸 수도 있음

⇒ 컴파일 phase가 문제가 됨

(위에서 각각의 phase단계 있었는데 그 중 컴파일 phase)

⇒ 각 페이즈마다 그걸 관리하는 plugin있음

⇒ compiler설정 변경 ⇒ 어디서? pom.xml

⇒ DOM : Document Object Model

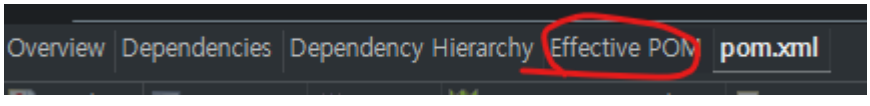
⇒ POM : Project Object Model

: 페이즈들에 대한 모든 설정이 이 안에 들어감

(유일한 maven의 설정파일)

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="ht
  <modelVersion>4.0.0</modelVersion>
  <groupId>kr.or.ddit</groupId>
  <artifactId>firstMVN</artifactId>
  <version>0.0.1-SNAPSHOT</version>
</project>
```

⇒ groupId, artifactId, version으로 설정



⇒ Effective POM가보니까 더 길게 되어 있음

⇒ 기본값들이 xml형태로 만들어져 있는 것

(필요하면 이 기본값들은 언제든지 변경 가능)

⇒ 컴파일러도

```
<plugin>
  <artifactId>maven-compiler-plugin</artifactId>
  <version>3.1</version>
  <executions>
    <execution>
      <id>default-compile</id>
      <phase>compile</phase>
      <goals>
        <goal>compile</goal>
      </goals>
    </execution>
    <execution>
      <id>default-testCompile</id>
      <phase>test-compile</phase>
      <goals>
        <goal>testCompile</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

⇒ 이거 1.8로 바꿔야 함

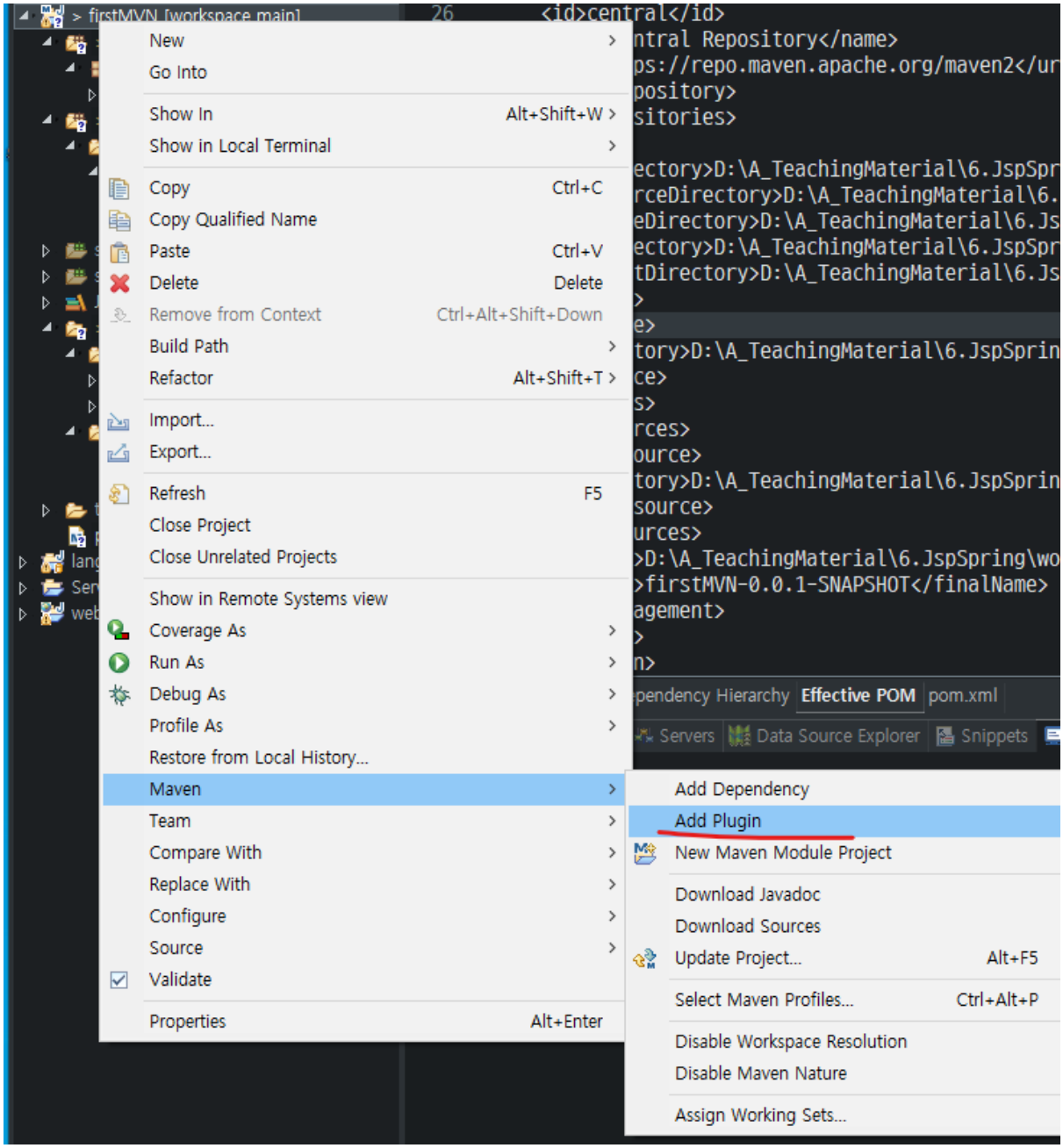
⇒ 나만의 플러그인으로 덮어 씌우면 됨

⇒ 그럼 나만의 플러그인 갖고 있어야 함

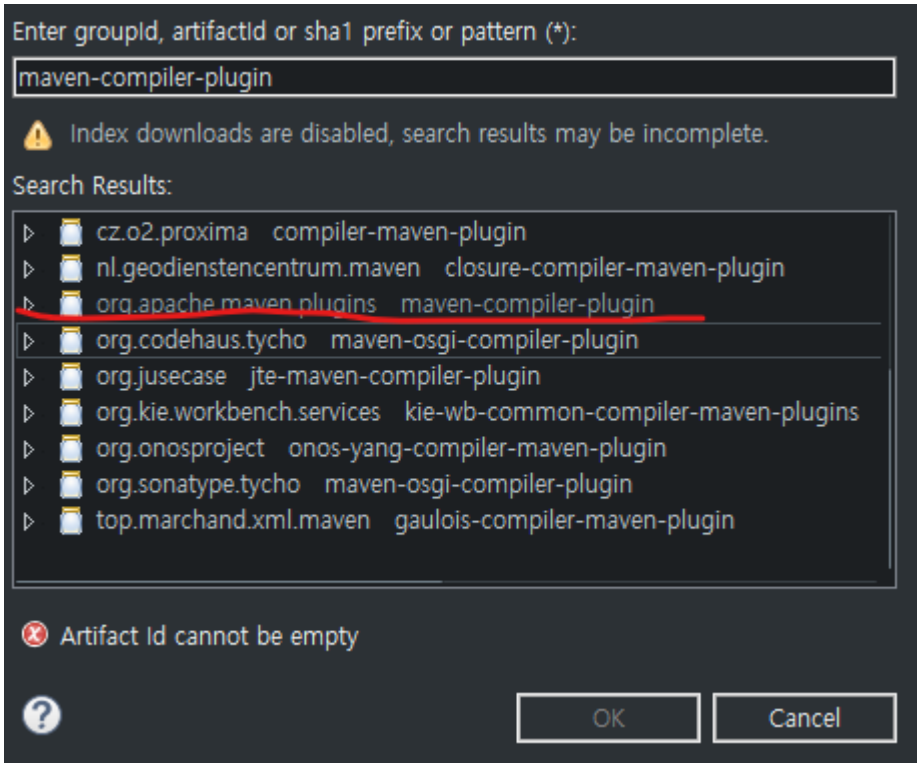
⇒ 메이븐의 모든 것은 중앙저장소에 있다

⇒ 거기에 찾아서 넣어주면 됨

⇒ 이거 구조보니까 <build> → <plugins> → <plugin>

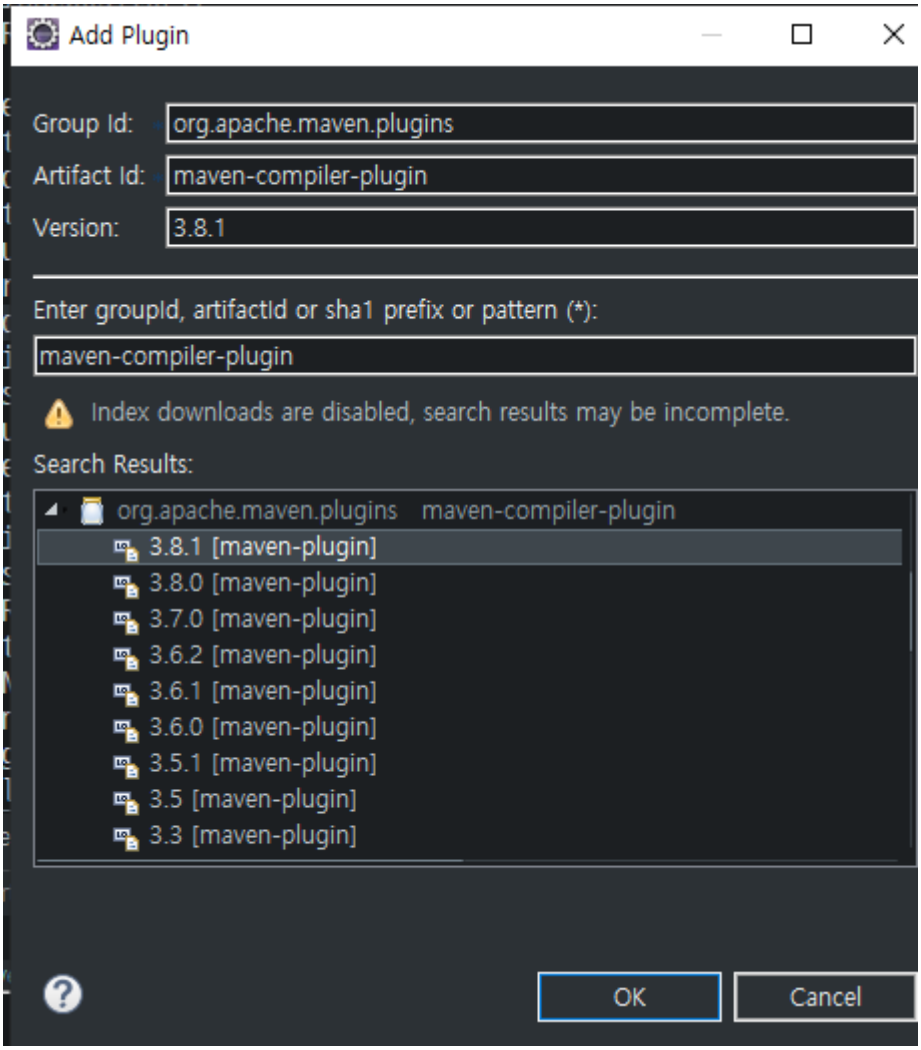


⇒ 저기 들어가면 밑에 창이 뜸



⇒ 우리의 기본값(약간 글씨 어두운 것)

⇒ 이거 확장해보면 우리가 쓰는 3.8.1 있음



⇒ 3.8.1 선택하고 변경

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:sc
  <modelVersion>4.0.0</modelVersion>
  <groupId>kr.or.ddit</groupId>
  <artifactId>firstMVN</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.8.1</version>
        </plugin>
    </plugins>
  </build>
</project>
```

⇒ maven 반드시 검색

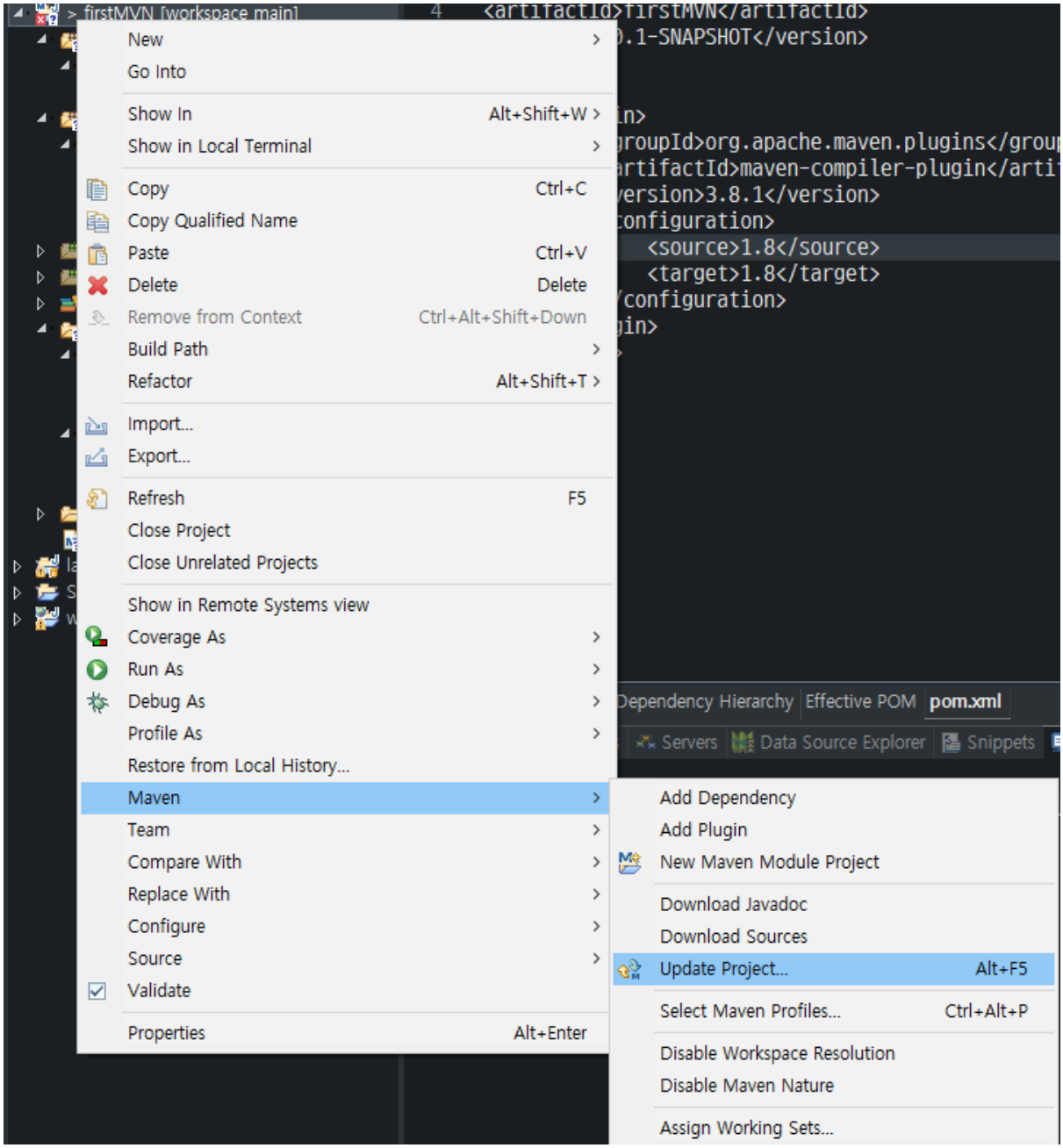
⇒ 검색할 때 꼭 필요한 3가지가 저 세 개

⇒ 저 세개 알면 다 가져올 수 있다

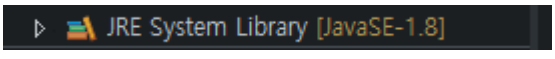
⇒ 그리고 그 밑에 configuration 설정해줘야 함(1.8)

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/20
  <modelVersion>4.0.0</modelVersion>
  <groupId>kr.or.ddit</groupId>
  <artifactId>firstMVN</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.8.1</version>
            <configuration>
                <source>1.8</source>
                <target>1.8</target>
            </configuration>
        </plugin>
    </plugins>
  </build>
</project>
```

⇒ 변경 안됨

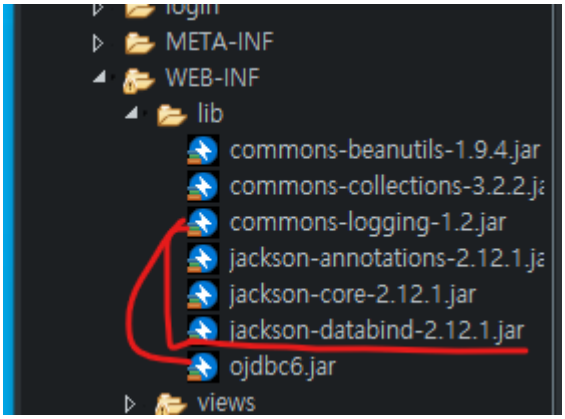


→ 바뀜



→ 변경됨

2. dependency관리



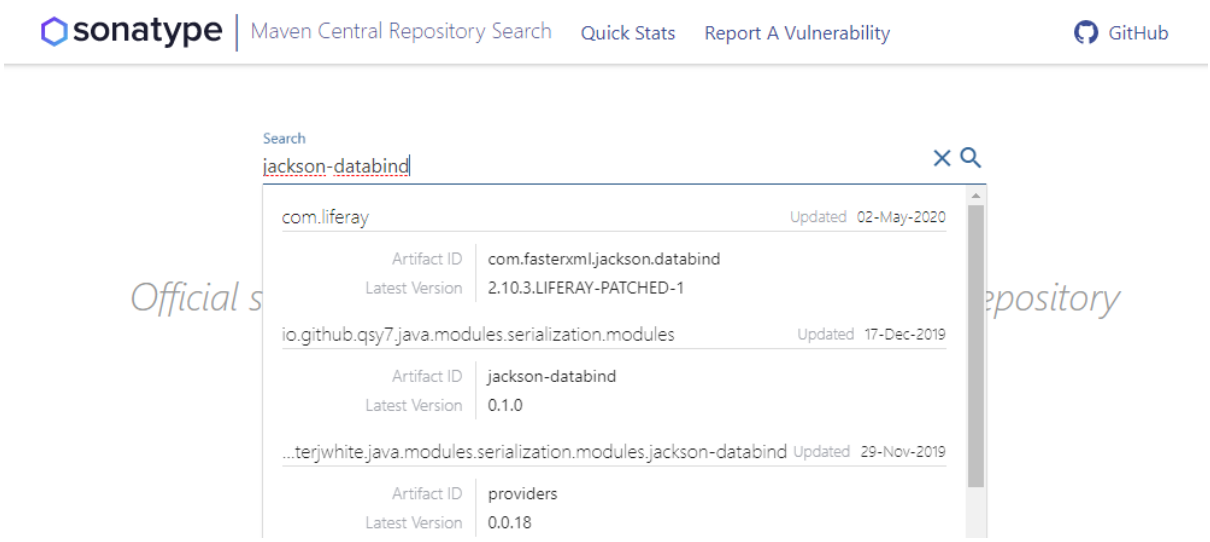
⇒ 우리가 이거 하나 쓰려고 다른 거 다 찾아서 따로 다운받음

⇒ 이걸 maven이 대신 해줌

⇒ 이게 dependency관리

⇒ 만약 maven인덱싱 안되어 있으면?

https://search.maven.org/

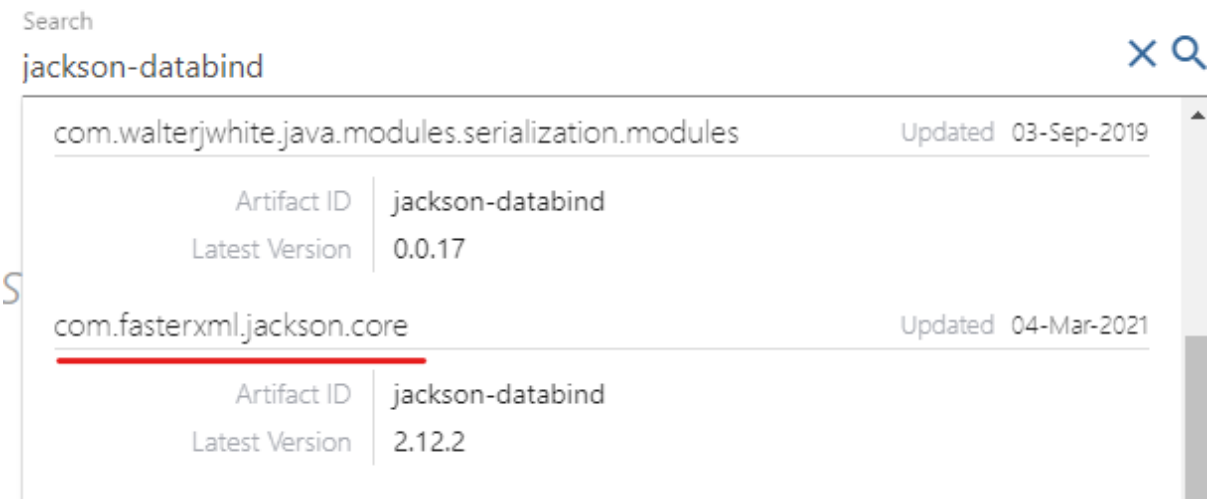


⇒ 여기엔 원본도 있고 복사본도 있고,,,

⇒ 잘 찾아야 함(원본)

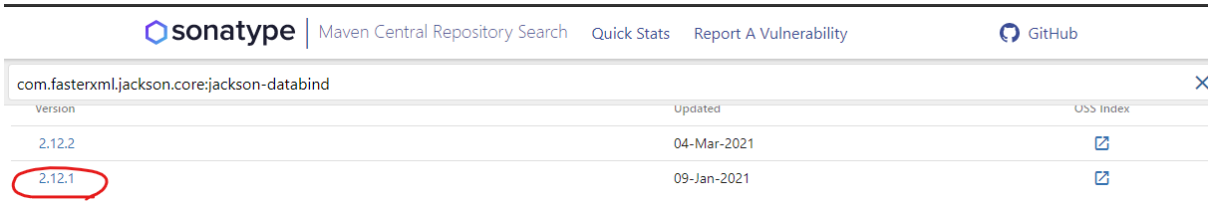
⇒ groupId 보기 ⇒ artifact id 보기 ⇒ version 잘 알고 가야 함

(이거 일치하는지 보기)

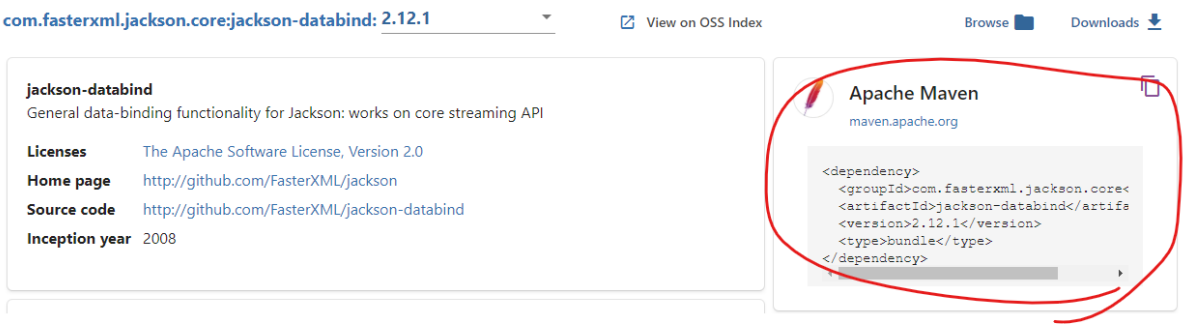


⇒ 이게 원본

⇒ api쓸 때 구조 잘 알아야 한다



⇒ 우리가 썼던 버전 들어가면



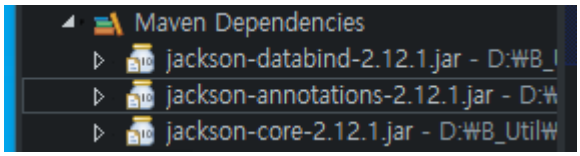
⇒ 저거 복사하기 → pom.xml에 추가하기

⇒ <type>bundle</type>삭제

(나중에 jar파일로 들어가게 할 거니까?)

```
    </build>
    <dependencies>
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-databind</artifactId>
            <version>2.12.1</version>
        </dependency>
    </dependencies>
```

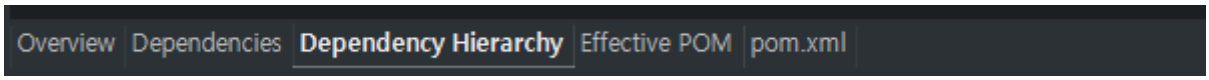
⇒ 이거 하나 넣었더니



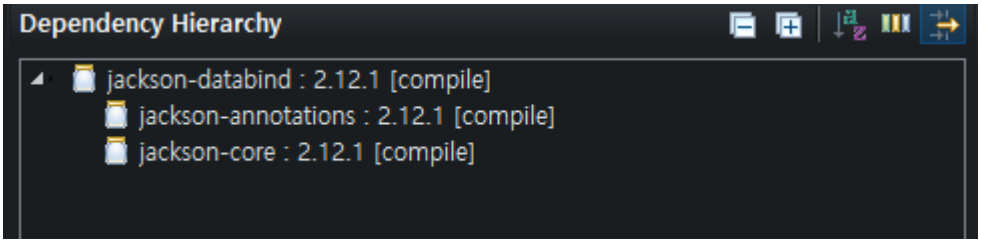
⇒ 이렇게 들어옴

⇒ 그럼 이거 계속 넣음 ⇒ 계속 쌓임

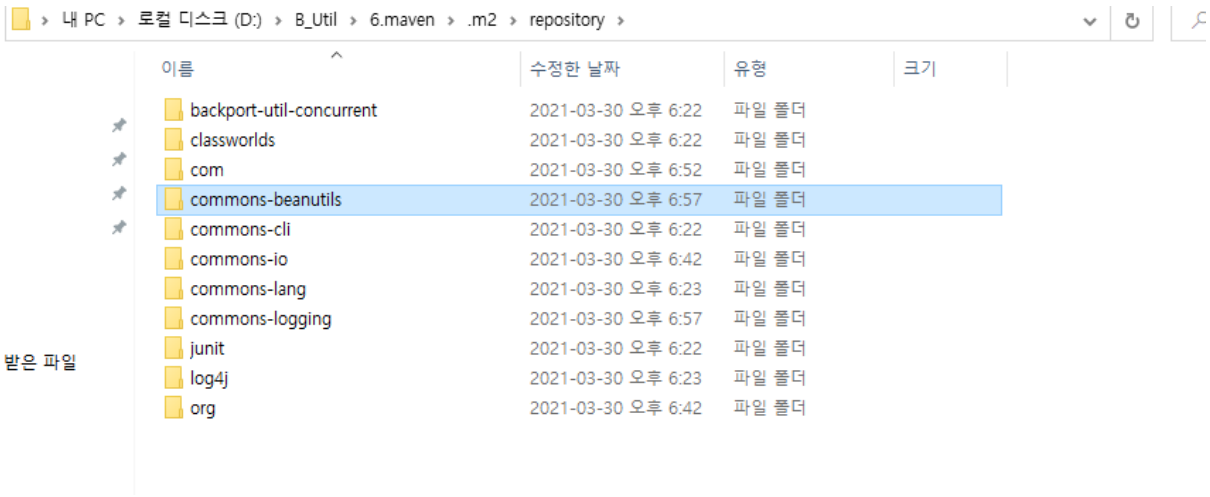
⇒ 어떤 dependency인지 알고 싶음



⇒ 여기서 계층 구조를 알 수 있다

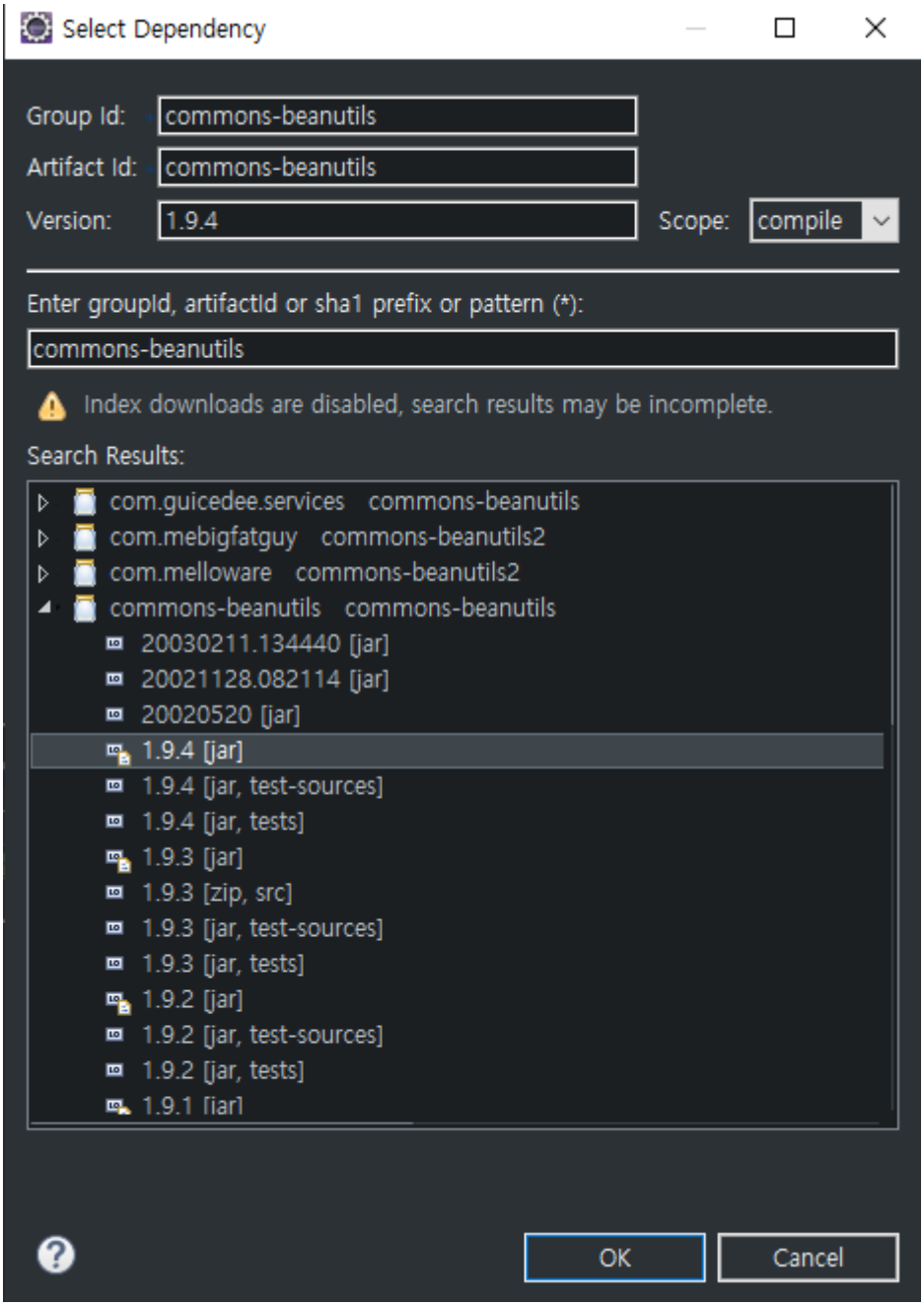


파일에서 확인해보기 ⇒ 내가 설치한 것들이 저장되어 있는 걸 볼 수 있음



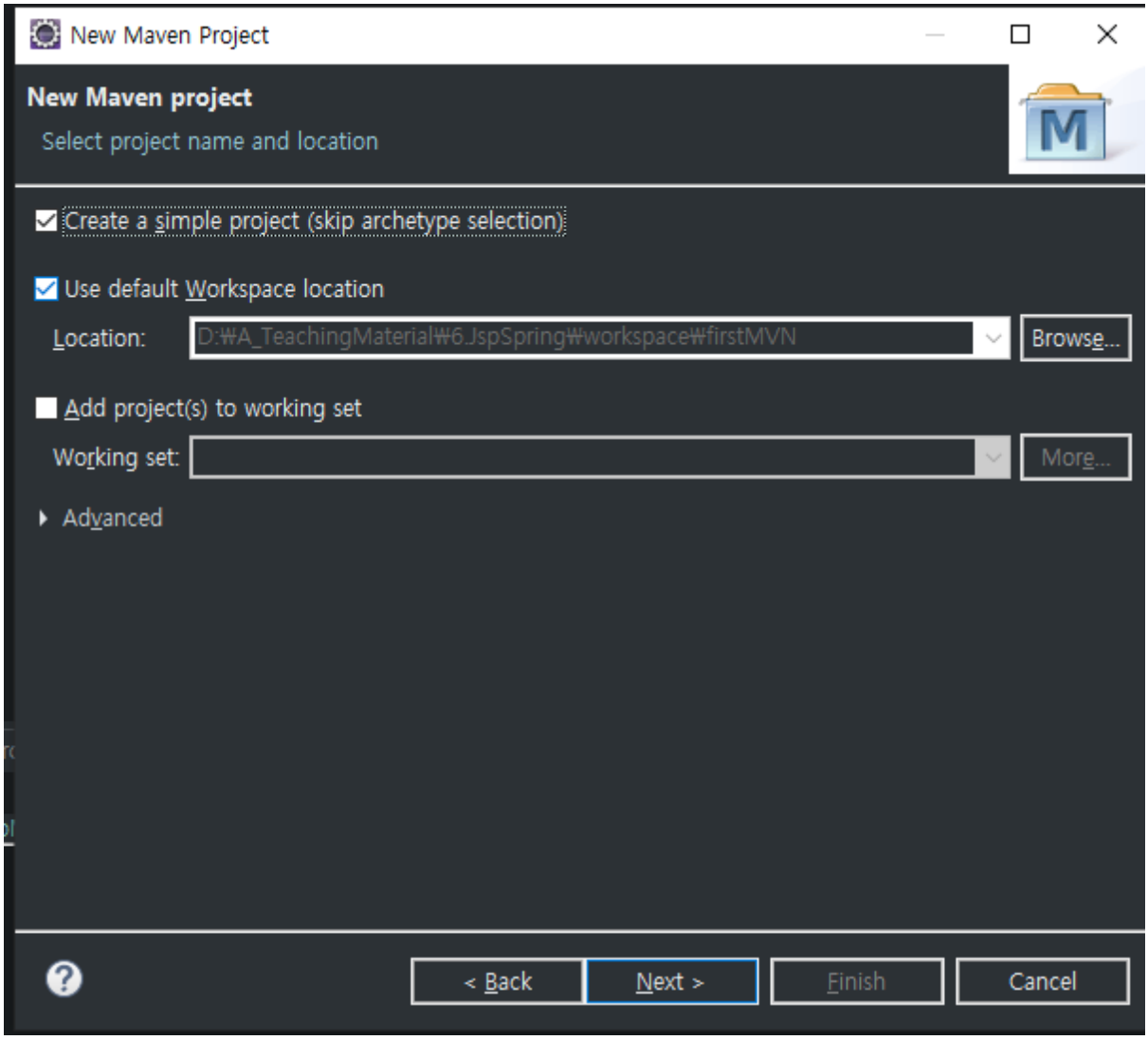
⇒ group ID → folder 이름

대 → 중 → 소 분류 : 폴더의 계층 구조



((이것도 똑같은 방법으로 가져오기 ⇒ commons-beanutils))

web용(똑같이 maven 만듦)

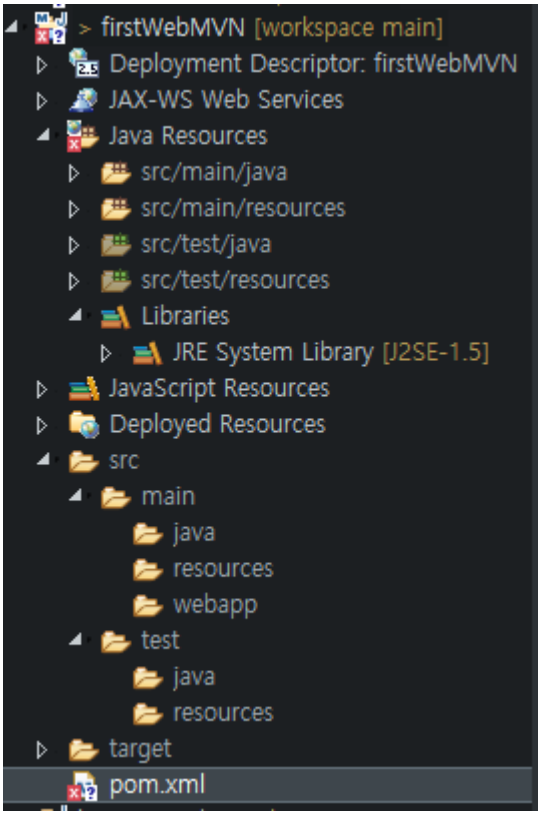


⇒ archetype selection : 다양하게 존재하는 템플릿 구조들

(다른 템플릿도 쓸 수 있다)

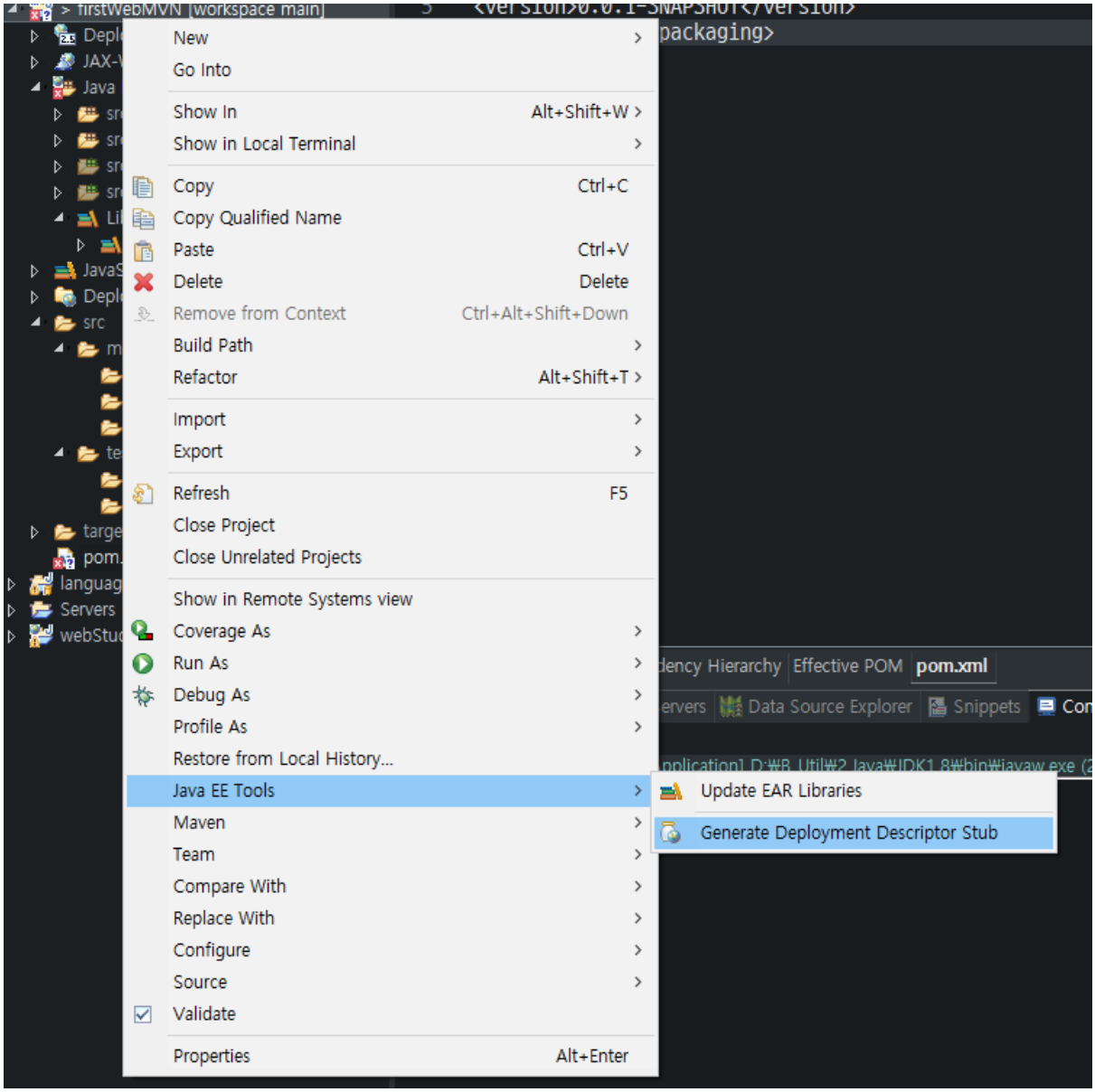
⇒ 잘못된 archetype많아서 쌤은 잘 안쓰심

(전세계 사람들이 써서 복사본도 많고 그만큼 불안정해서)

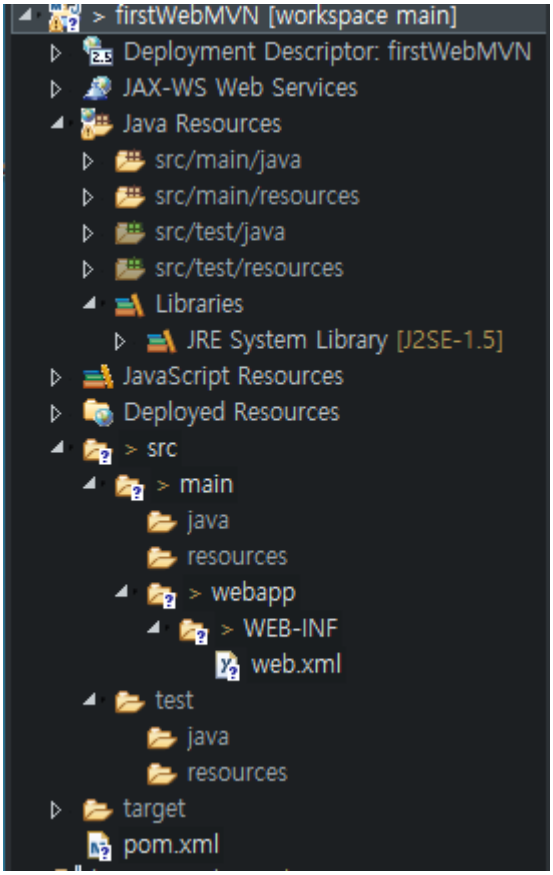


⇒ 웹용은 webapps있어야 함 ⇒ 어디있지?

⇒ src확장했더니 여기에 있음 → webapp



⇒ 이거 눌러서 web.xml생성 가능



⇒ 생김

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-ins
  <modelVersion>4.0.0</modelVersion>
  <groupId>kr.or.ddit</groupId>
  <artifactId>firstWebMVN</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>war</packaging>
  <properties>
    <java-version>1.8</java-version>
  </properties>
  <build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.8.1</version>
            <configuration>
                <source>${java-version}</source>
                <target>${java-version}</target>
            </configuration>
        </plugin>
    </plugins>
  </build>
</project>
```

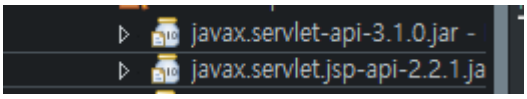
placeholder ⇒ ${}

⇒ 이렇게 하면 위에서 설정한 java-version 쓸 수 있다

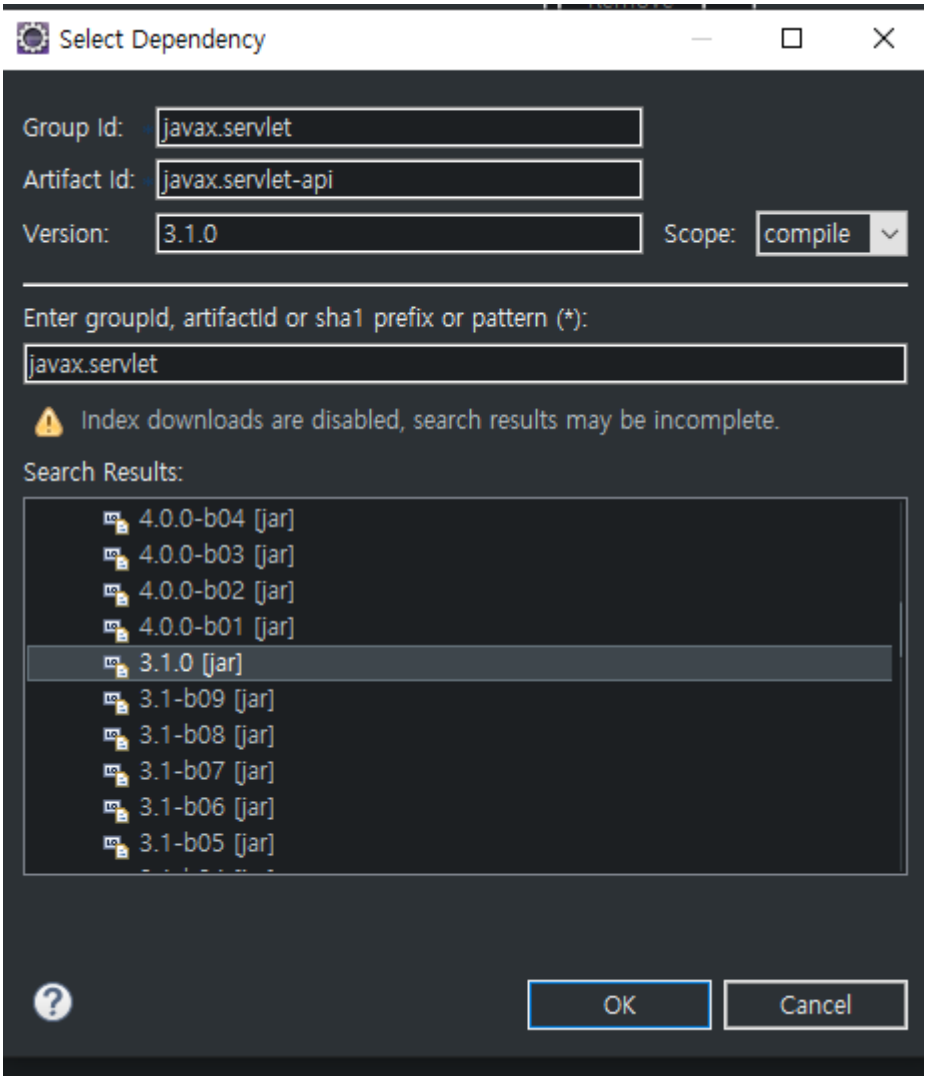
(그냥 쓰면 값으로 인식하니까)

⇒ Servlet만드려고 했더니 code assistance 안됨

⇒ 왜지? Servlet이랑 JSP 추가 안 되어 있어서

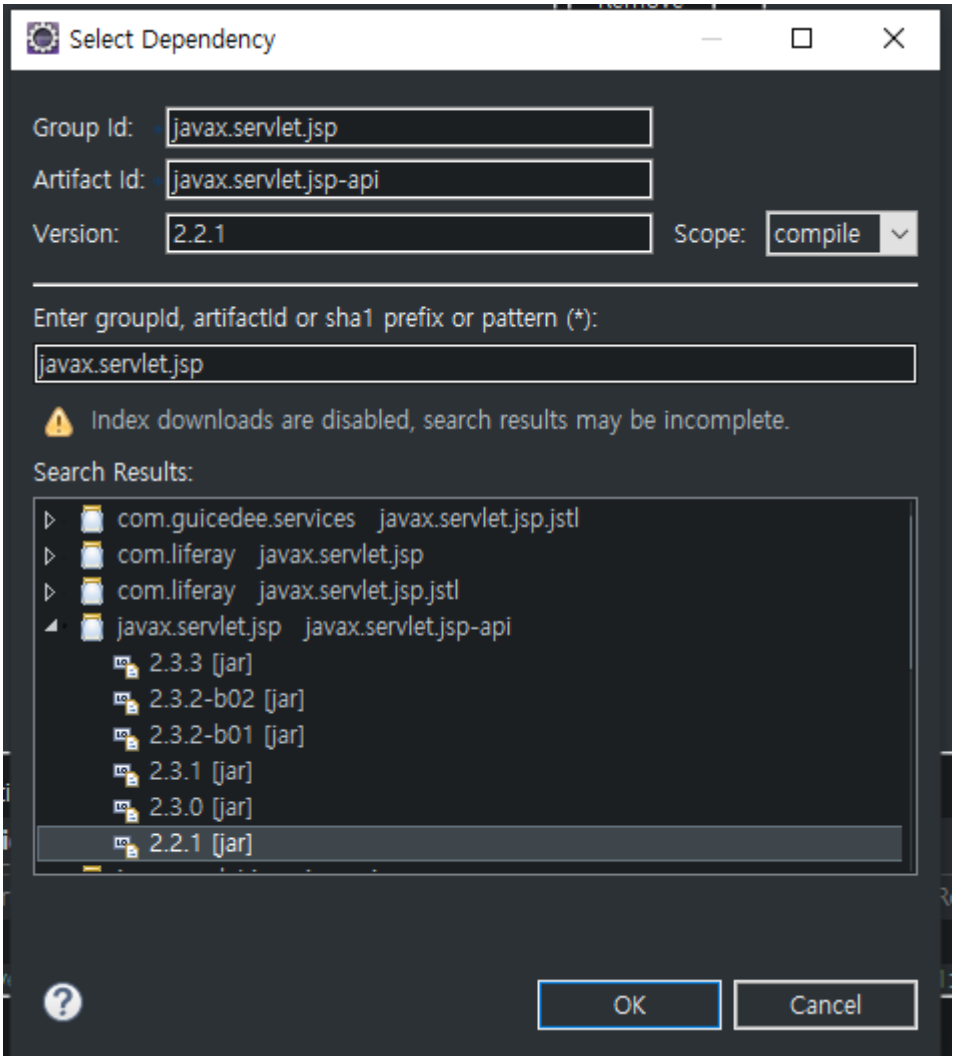


⇒ 이 둘이 없어서



⇒ 이렇게 servlet스펙 추가

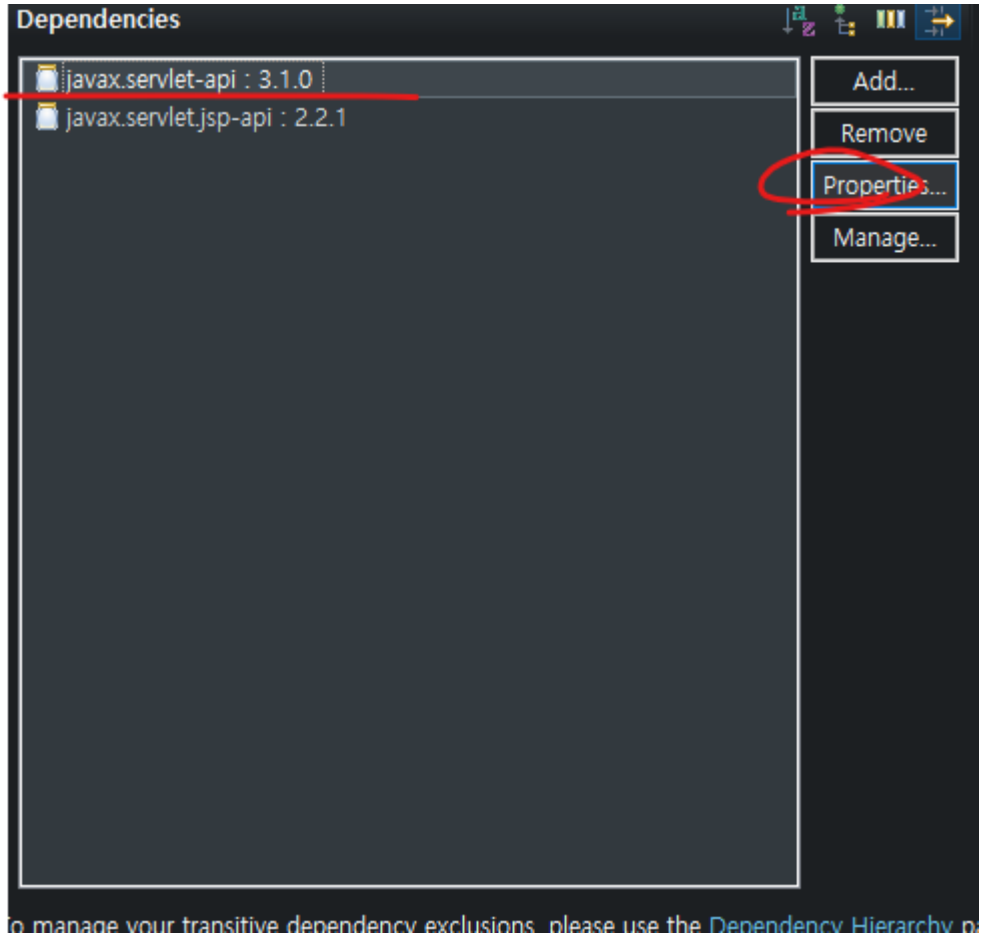
⇒ jsp도 추가해야 함



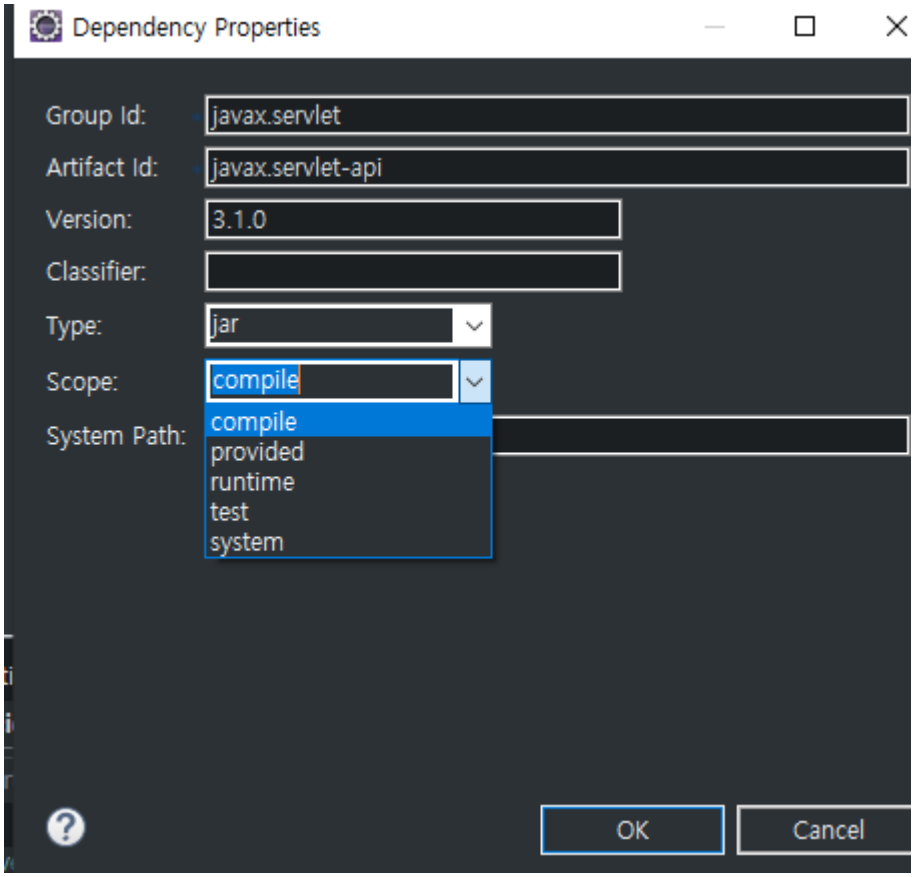
⇒ 이렇게 하고 한 가지 더 해야 함

(운영서버에서도 분명히 jar파일 있을거임)

⇒ 개발할 땐 이거 써야하지만 배포할 땐 이거 버리고 가야함

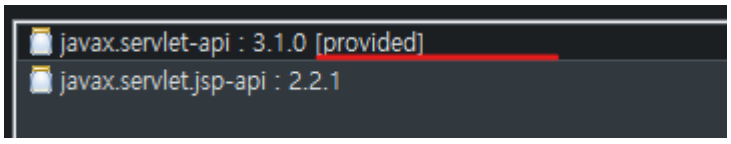

Dependencies
javax.servlet-api : 3.1.0
javax.servlet.jsp-api : 2.2.1
Add...
Remove
Properties...
Manage...

⇒ 저기서 설정



⇒ compile 가장 큰 개념(이거 쓰겠다)

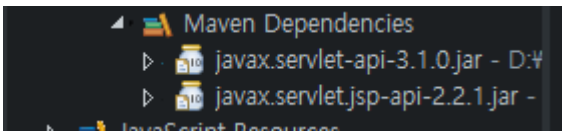
⇒ provided ⇒ 지금 이거 쓰는데 배포할 땐 제공되는 거 쓰겠다




```
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>javax.servlet-api</artifactId>
        <version>3.1.0</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>javax.servlet.jsp</groupId>
        <artifactId>javax.servlet.jsp-api</artifactId>
        <version>2.2.1</version>
        <scope>provided</scope>
    </dependency>
  </dependencies>
</project>
```

iew | Dependencies | Dependency Hierarchy | Effective POM | pom.xml

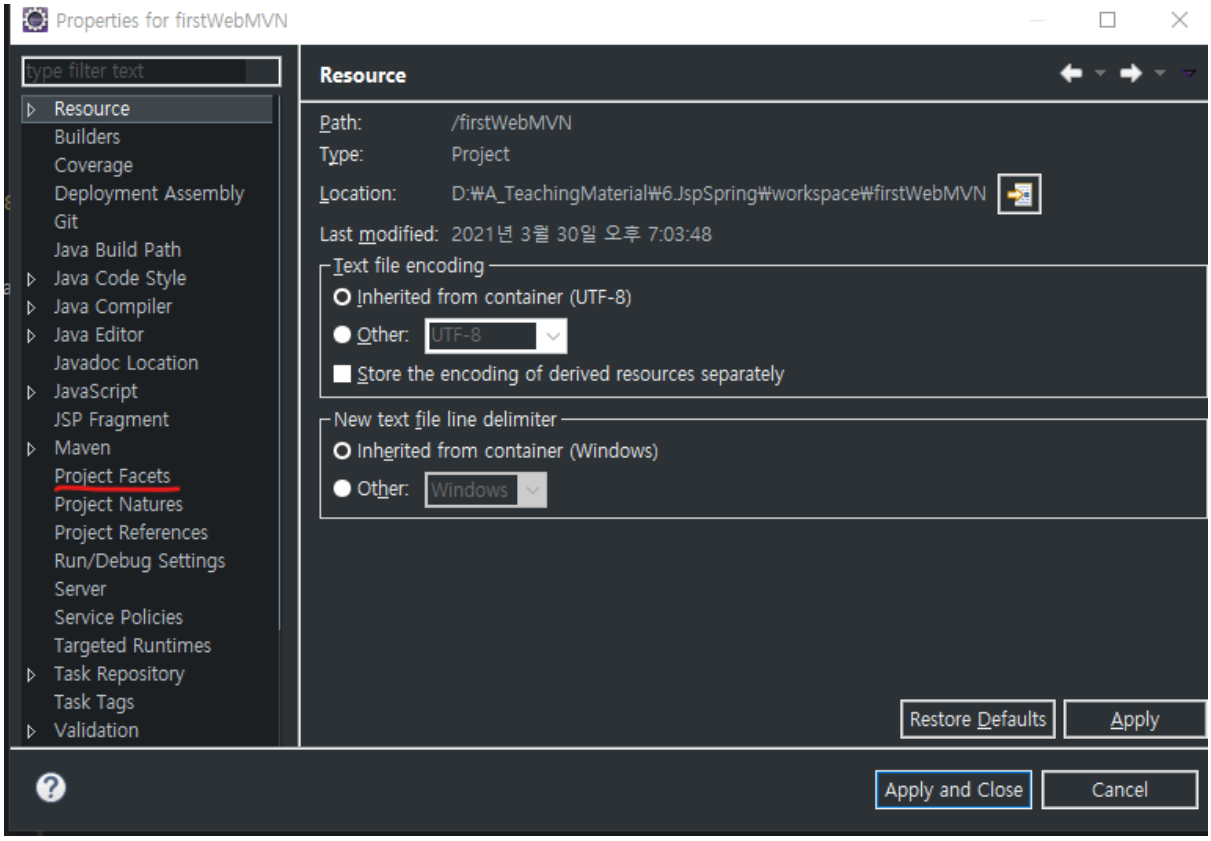
⇒ 만약 scope : test → test할 때만 쓰겠다



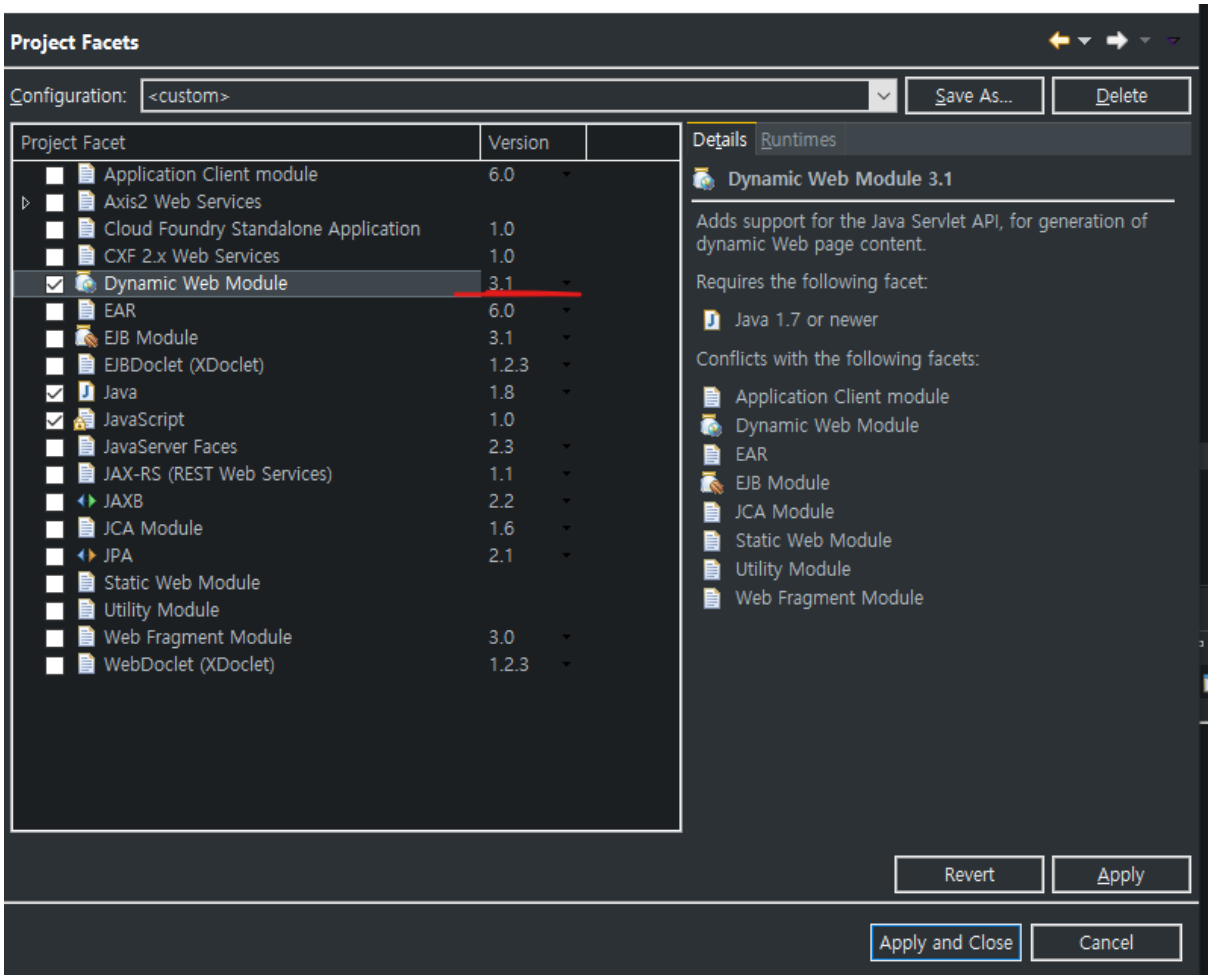
⇒ 이렇게 들어와 있음

⇒ 지구본 근데 아직 3.1 안됨

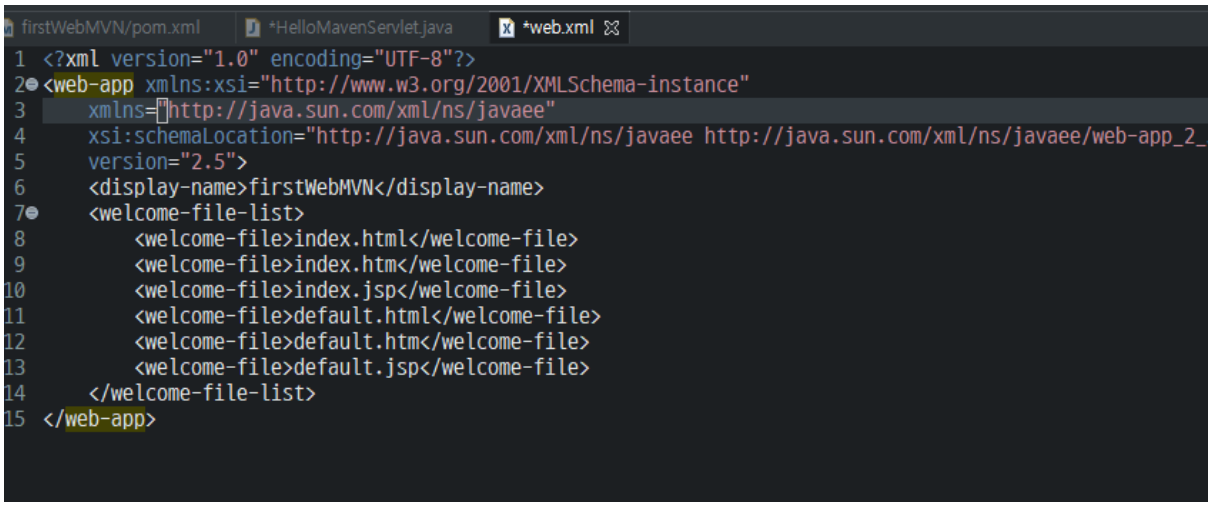
프로젝트에서 <alt> <enter>→ 속성 설정할 수 있는 창 뜸



⇒ Project Facets로 감



⇒ 이렇게 했는데도 안 바뀜



⇒ 아까 web.xml 만듦

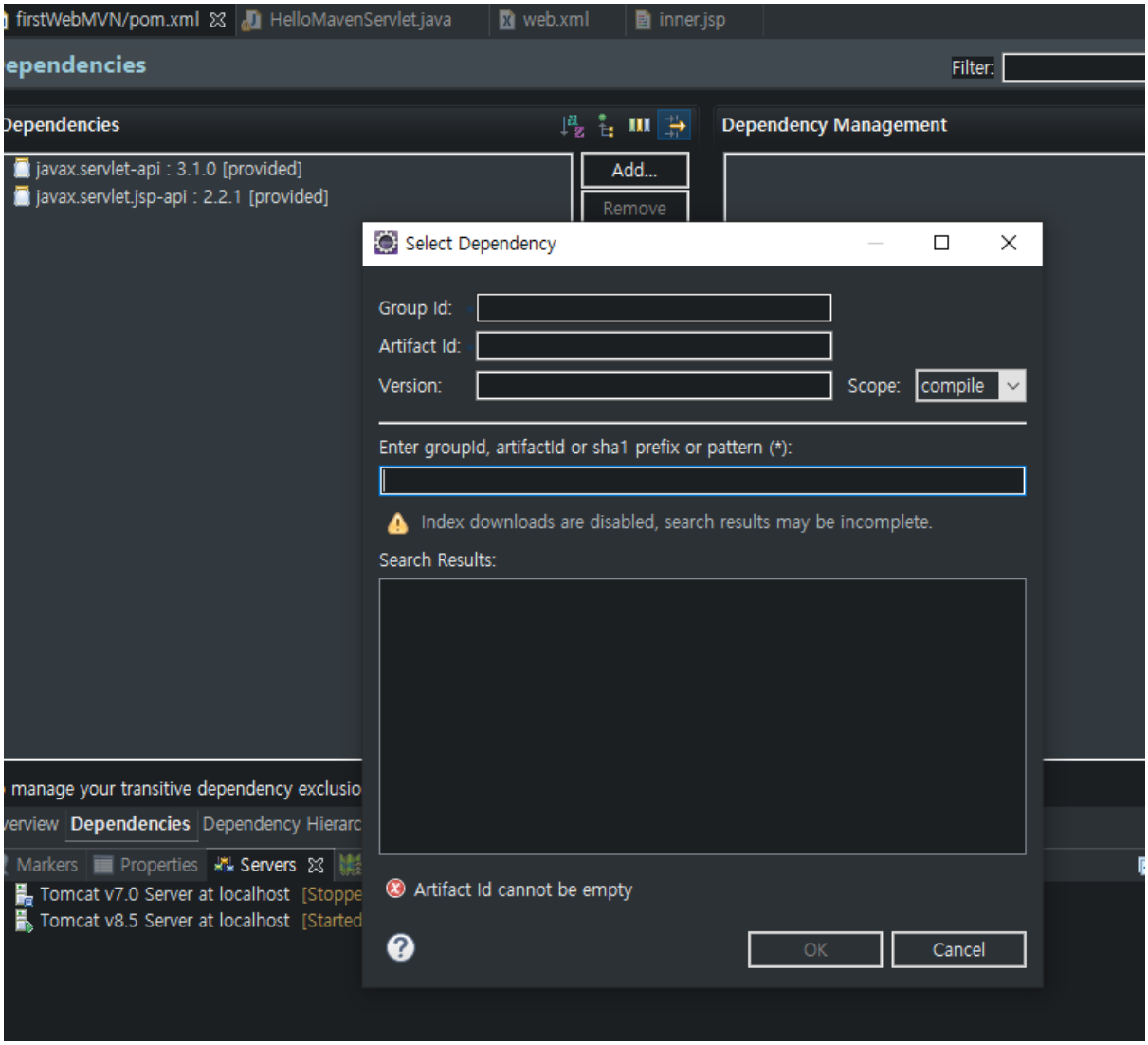
→ 근데 version이 2.5로 되어 있음

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/
    version="3.1">
    <display-name>firstWebMVN</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>
</web-app>
```

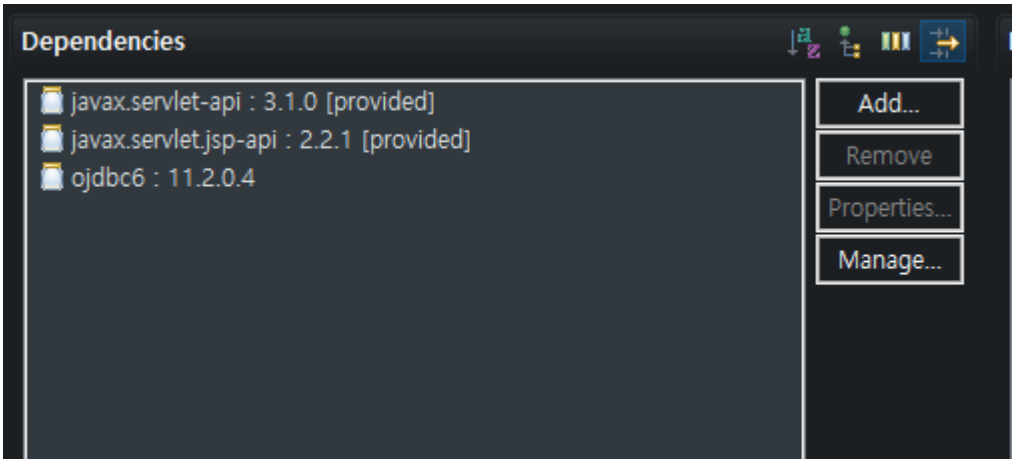
⇒ 그래서 web.xml삭제하고 다시 생성함

⇒ 그럼 정상적으로 생김

나 oracle 써야함



⇒ ojdbc 찾아봄



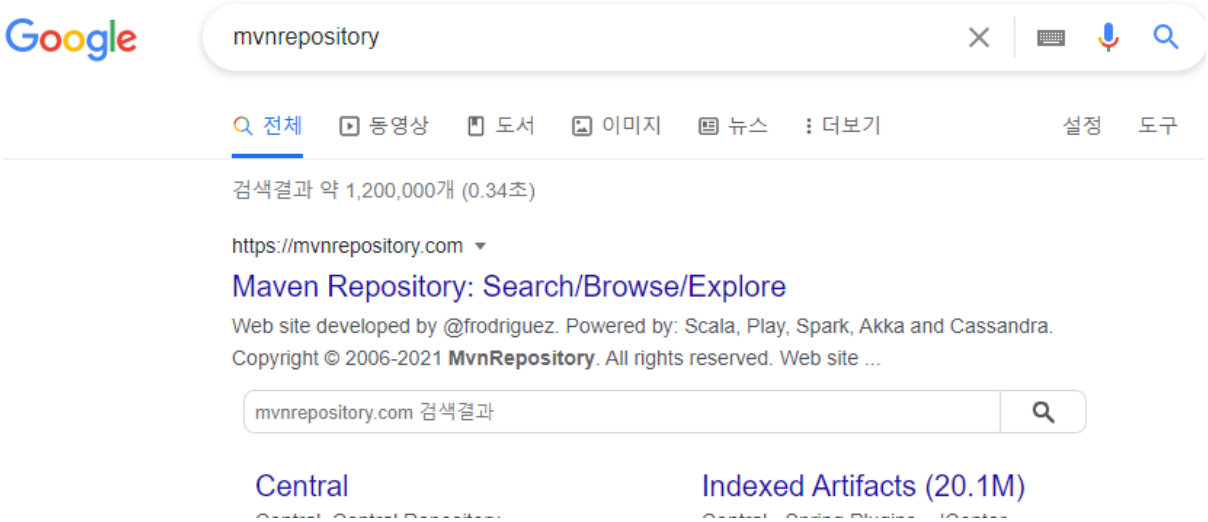
⇒ 넣음

⇒ 근데 원본 같지 않음

⇒ 근데 오라클 엄청 비쌈 ⇒ 원본 공개저장소에 안 넣어놓음

⇒ 그래서 저건 사람들이 복사본을 올려 놓은 것

⇒ 그럼 오라클은 다른 곳에서 찾아야 함



⇒ 통합검색 ⇒ https://mvnrepository.com/

(거의 모든 repository 들어 있음)

⇒ 아까는 중앙 저장소만 뒤진거임

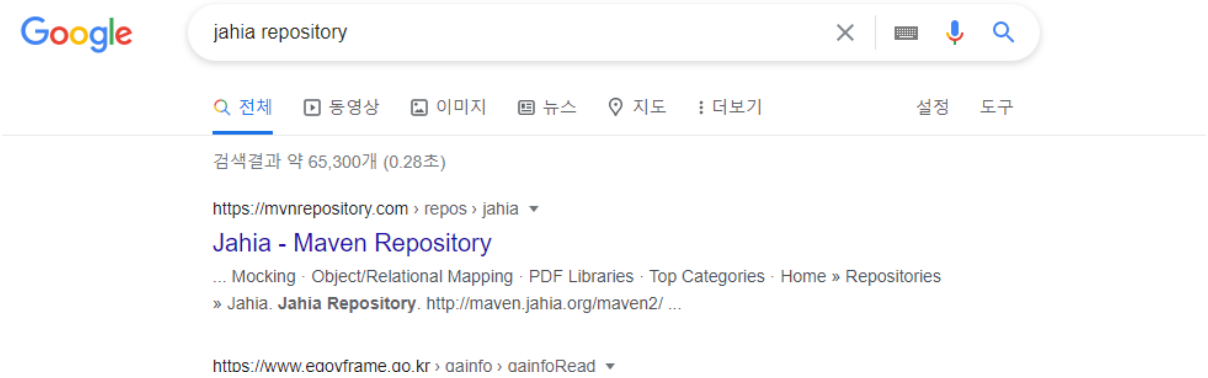
⇒ 여기서 더 많은 repository 뒤질 수 있음

⇒ 근데 여기는 오라클 원본 없음

(usage가 많은 게 좋은 건데 없음 ⇒ 불안정하다는 뜻)

다른 곳에서 찾아보자

⇒ 전세계에 존재하는 db jar파일 계속 업데이트해줌



⇒ 저기 들어가서 url 저거

URL	http://maven.jahia.org/maven2/
Jars	4,831 indexed jars

http://maven.jahia.org/maven2/

⇒ 여기 들어가면 검색 가능

Index of /groups/maven-jahia-org

Name	Last Modified	Size	Description
Parent Directory			
Jahia/	Mon Jun 03 04:28:15 UTC 2019		
atg/	Fri Mar 24 08:10:34 UTC 2017		
biz/	Thu Apr 13 13:28:50 UTC 2017		
com/	Mon Dec 03 16:24:59 UTC 2018		
commons-id/	Wed Mar 22 17:53:33 UTC 2017		
commons-xmlio/	Wed Mar 22 17:55:22 UTC 2017		
concurrent/	Wed Mar 22 17:24:03 UTC 2017		
de/	Tue May 09 16:31:38 UTC 2017		
external/	Wed Mar 22 18:01:01 UTC 2017		
imagej/	Wed Mar 22 17:54:57 UTC 2017		
io/	Sat Jan 13 00:01:44 UTC 2018		
javax/	Thu Jul 30 15:20:49 UTC 2020		
jgroups/	Wed Mar 22 17:52:36 UTC 2017		
microsoft/	Wed Mar 22 17:54:12 UTC 2017		
multithreadedtc/	Thu Jun 18 06:00:23 UTC 2020		
mysql/	Wed Mar 22 17:53:58 UTC 2017		
net/	Thu Mar 23 21:56:56 UTC 2017		
org/	Sat Jun 27 06:55:03 UTC 2020		
postgresql/	Wed Mar 22 17:52:59 UTC 2017		
slide/	Wed Mar 22 17:53:12 UTC 2017		
taglibs/	Thu Mar 23 12:49:17 UTC 2017		
xerces/	Wed Mar 22 17:57:14 UTC 2017		
archetype-catalog.xml	Tue Mar 30 10:32:12 UTC 2021	25	
archetype-catalog.xml.md5	Mon Jul 02 16:45:31 UTC 2012	32	
archetype-catalog.xml.sha1	Mon Jul 02 16:45:31 UTC 2012	40	

⇒ group id는 url 반대로 쓰기

⇒ oracle com으로 끝남

Index of /groups/maven-jahia-org/com

Name	Last Modified	Size	Description
Parent Directory			
caucho/	Thu Apr 13 13:28:59 UTC 2017		
extjs/	Thu Mar 23 20:23:29 UTC 2017		
google/	Fri Jun 26 12:37:57 UTC 2020		
googlecode/	Fri May 05 16:44:13 UTC 2017		
graphql-java-kickstart/	Mon Dec 03 16:24:59 UTC 2018		
graphql-java/	Sat Dec 23 11:59:20 UTC 2017		
hazelcast/	Wed Mar 22 15:09:39 UTC 2017		
ibm/	Fri Oct 06 21:37:56 UTC 2017		
microsoft/	Thu Mar 23 07:50:36 UTC 2017		
mogobiz/	Thu Feb 16 10:30:45 UTC 2017		
oracle/	Wed Mar 22 17:46:22 UTC 2017		
phloc/	Thu Apr 13 10:16:06 UTC 2017		
sun/	Wed May 03 11:01:38 UTC 2017		
vdurmont/	Thu Apr 13 10:16:19 UTC 2017		
yahoo/	Fri Mar 31 08:39:42 UTC 2017		

Index of /groups/maven-jahia-org/com/oracle

Name	Last Modified	Size	Description
Parent Directory			
ojdbc5/	Mon Mar 29 12:02:25 UTC 2021		
ojdbc6/	Tue Mar 30 09:09:15 UTC 2021		
ojdbc7/	Tue Mar 30 08:28:50 UTC 2021		
orai18n/	Tue Mar 30 10:12:05 UTC 2021		

Index of /groups/maven-jahia-org/com/oracle/ojdbc6/12.1.0.2

Name	Last Modified	Size	Description
Parent Directory			
ojdbc6-12.1.0.2.jar	Wed Sep 17 09:02:19 UTC 2014	3692096	
ojdbc6-12.1.0.2.jar.md5	Wed Sep 17 09:02:19 UTC 2014	32	
ojdbc6-12.1.0.2.jar.sha1	Wed Sep 17 09:02:19 UTC 2014	40	
ojdbc6-12.1.0.2.pom	Wed Sep 17 09:02:19 UTC 2014	455	
ojdbc6-12.1.0.2.pom.md5	Wed Sep 17 09:02:19 UTC 2014	32	
ojdbc6-12.1.0.2.pom.sha1	Wed Sep 17 09:02:19 UTC 2014	40	

⇒ 여기까지 들어오기

⇒ 그럼 이거 다운받아서 써야 함

⇒ 이거 써야하는데 central repository가 아님

(경우에 따라선 하나가 아니라 repository 여러 개 써야 함)

⇒ respository 추가하기 (pom.xml에)

```
<repositories>
  <repository>
    <id>jahia</id>
    <url>http://maven.jahia.org/maven2/</url>
  </repository>
</repositories>
```

⇒ 이렇게 추가해주면 됨

⇒ 밑에 있는 dependencies에 저 repository에 있는 것도 쓸 수 있다

```
<dependency>
  <groupId>com.oracle</groupId>
  <artifactId>ojdbc6</artifactId>
  <version>12.1.0.2</version>
</dependency>
```

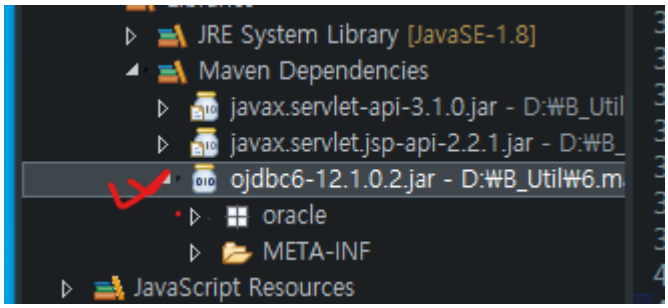
⇒ 이렇게 추가

Index of /groups/maven-jahia-org/com/oracle/ojdbc6/12.1.0.2

⇒ 뒤에 보면 12.1.0.2 버전 정보

⇒ 그럼 그 바로 위에 있는 게 artifactId ⇒ ojdbc6

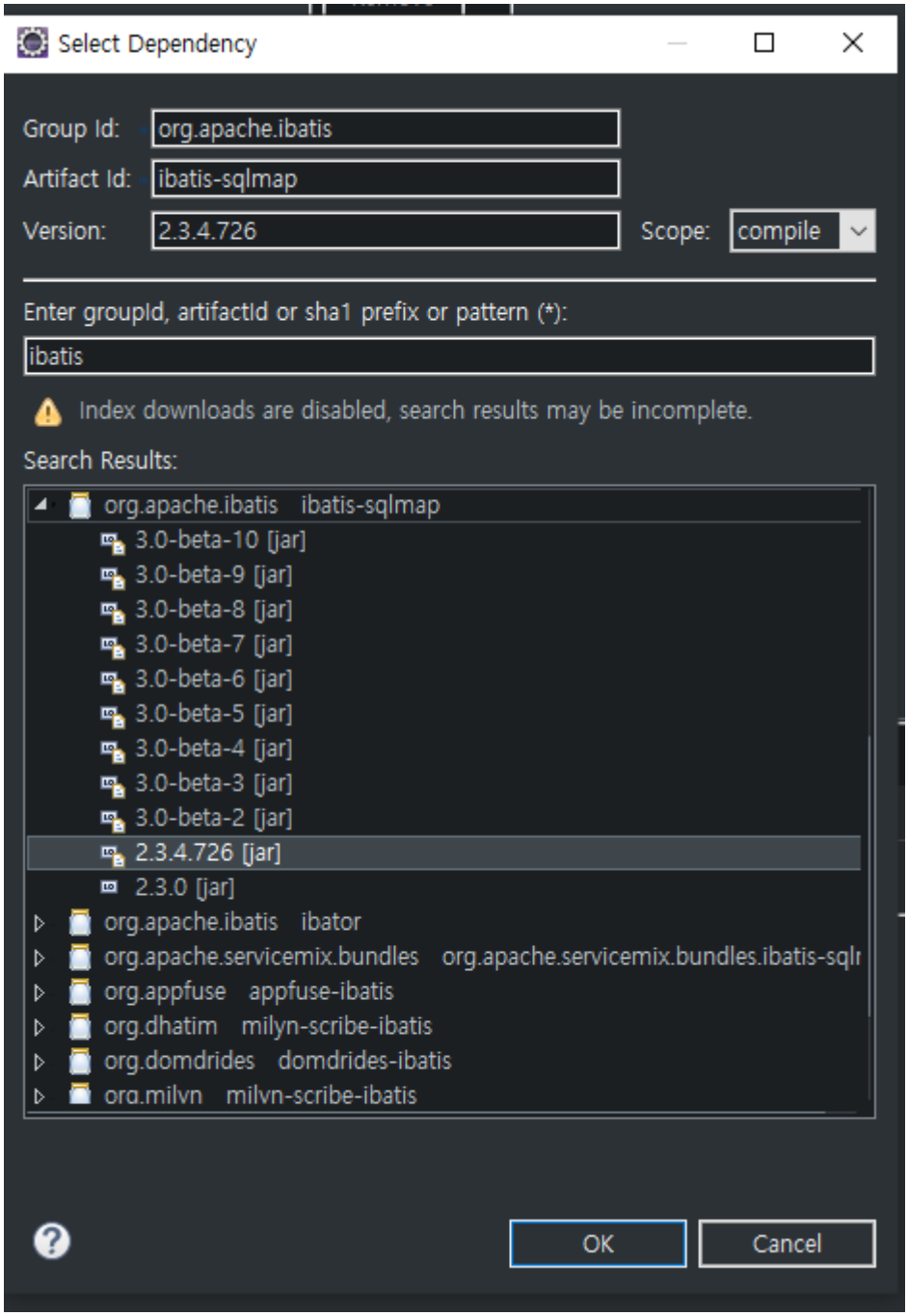
⇒ 그 위는 다 groupId(com.oracle)



⇒ 설치됨

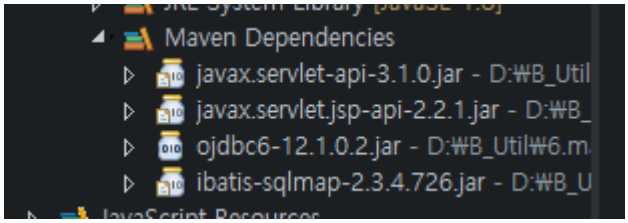
---

ibatis설치



⇒ ibatis는 원래 apache에서 만들어지고 판권이 google로 넘어가면서 mybatis로 이름 바뀜

⇒ beta, alpha, seta 붙은 건 쓰지 말자



⇒ 이렇게 ibatis가 추가 되어 있다

숙제

1. 집에 똑같은 개발환경 만들기