

Maven 기반 프로젝트 구성

노트북: 0074_jsp 개요

만든 날짜: 2023-01-22 오전 12:48

수정한 날짜: 2023-01-25 오후 2:25

작성자: nomad.lab.kr@gmail.com

빌드 도구에 관해서 알아보고 Maven 기반으로 프로젝트를 전환.

1. 빌드 도구

보통 프로그램 언어를 배우는 과정에서는 단순한 예제 소스로 구성된 프로그램을 만들기 때문에

코드 이외의 부분에 대해서는 고려할 사항이 거의 없음.

그러나 실제 프로젝트에서는 직접 작성해야 할 수십 개의 클래스와 여러 외부 라이브러리 (예 apache jstl 라이브러리 등)로 구성되는 경우가 많음.

또한 프로그램의 실제 구현 코드 외에 테스트를 위한 코드도 존재하며

프로젝트 규모가 커질수록 이클립스의 자동 컴파일 옵션은 개발자에게 오히려 비효율적인 요소가 될 수 있음.

이처럼 실제 프로젝트 환경은 학습 환경과 다르며 복잡한 프로젝트의 소스를 체계적으로 컴파일하고

관련 라이브러리의 버전이나 종속성 관리를 쉽게 도와줄 방법이 필요.

이를 위해 가장 오래된 자바 빌드 도구로는 Ant가 있으면 2004년에는 아파치 프로젝트로 Maven이 새롭게 나오게 됨.

이후 오랜 기간 Maven은 절대 다수가 사용하는 자바 빌드 도구가 되었으며 특히 스프링 프레임워크 개발에서 기본 빌드 도구로 활용.

시간이 지남에 따라 좀 더 유연하면서도 복잡한 처리를 쉽게 하기 위한 요구 사항의 증대로

2012년 Gradle이 나오게 되었고, 안드로이드 앱 개발의 기본 빌드 도구가 되었음.

현재 Maven과 Gradle이 가장 대표적인 빌드 도구.

2. 나에게 맞는 빌드 도구 선택하기

어떤 빌드 도구를 선택하는 것이 좋을까?

결론부터 말하면 지금 당장은 Maven과 Gradle 중 무엇을 선택해도 전혀 상관이 없음.

또한 빌드 도구를 잘 활용하기 위해서는 많은 경험이 필요하고 프로젝트 규모가 어느 정도 되어야 제대로 된 활용이 가능하기 때문에 초급 개발자 수준에서는 빌드 도구 선택에 대해 지나치게 고민할 필요가 없음.

스프링 프레임워크나 안드로이드의 경우 개발도구에서 기본적으로 빌드 관련 설정과 실행을 연계해주기 때문에 빌드 도구가 잘 이해되지 않는다고 어려워할 필요가 없음.

빌드 도구의 특징

1) 설정 방식

* Maven은 빌드 설정을 'pom.xml'파일에 작성하는데, XML 구조이기 때문에 프로젝트가 커질수록 스크립트의 내용이 길어지고 가독성이 떨어지는 문제가 있음.

* Gradle은 Groovy라고 하는 JVM 기반 언어를 통해 프로그램 구조로 설정.

따라서 훨씬 적은 양의 스크립트로 짧고 간결하게 작성할 수 있음.

2) 다중 프로젝트

* Maven은 다중 프로젝트에서 특정 설정을 다른 모듈에서 사용하려면 상속을 받아야 함.

* Gradle은 설정 주입 방식을 사용하여 다중 프로젝트에 적합.

3) 개발환경

* 안드로이드 프로젝트는 기본적으로 Gradle을 사용.

* 스프링 프레임워크 기반 프로젝트는 Maven, Gradle 중에 선택할 수 있음.

* 이클립스는 Maven에 친화적이고 IntelliJ는 Gradle에 친화적.

프로젝트 규모나 설정 조건에 따라 다르기는 하지만 Gradle이 Maven보다 10 ~ 100배 성능 향상이 있다고 알려져 있음.

이에 따라 Maven도 지속적으로 사용되고 있지만 최신 프로젝트에서는 Gradle 사용 비중이 더 높음.

3. 리포지터리

빌드 도구를 사용하는 주요 목적 두 가지는 컴파일/실행 설정과 라이브러리 설정.

이 중에서도 라이브러리 설정을 꼭 알아두어야 함.

JSTL 사용을 위해 관련 라이브러리를 직접 다운로드하고 'WEB-INF' 폴더에 복사해서 사용,

만약 필요한 라이브러리가 30개 정도 된다면 개발자가 일일이 해당 라이브러리 홈페이지를 찾아

다운로드라고 복사하는 과정을 수행하는 것이 상당히 번거로움.

필요한 라이브러리가 또 다른 라이브러리를 참조하고 있다면 해당 라이브러리도 필요한데

이들 관계는 또 어떻게 할 것이며, 버전에 따른 업데이트나 특정 버전 설치 등 너무나도 복잡한 문제가 발생하게 됨.

빌드 도구를 사용하면 꼭 필요한 핵심 라이브러리만 설정 파일에 등록해두면 해당 라이브러리에서

필요로 하는 다른 라이브러리는 자동으로 설치되기 때문에 신경 쓸 필요가 없다는 이점이 있음.

리포지터리는 이러한 라이브러리를 통합 보관하는 일종의 저장소이며, 설정 파일을 내용을 참고해 해당 라이브러리를

글로벌 저장소에서 로컬 저장소로 다운로드한 다음 프로젝트에 복사하는 과정을 거쳐 사용할 수 있음.

해당 라이브러리가 로컬 저장소에 있다면 인터넷에서 다운로드 하지 않고 바로 사용할 수 있음.

필요에 따라서는 개발 회사가 자신들에게 필요한 라이브러리만 저장하거나

자체 라이브러리 저장을 위한 저장소를 두고 사용.

4. 이클립스 Maven 설정

이클립스에서 Maven 프로젝트를 설정하는 방법.

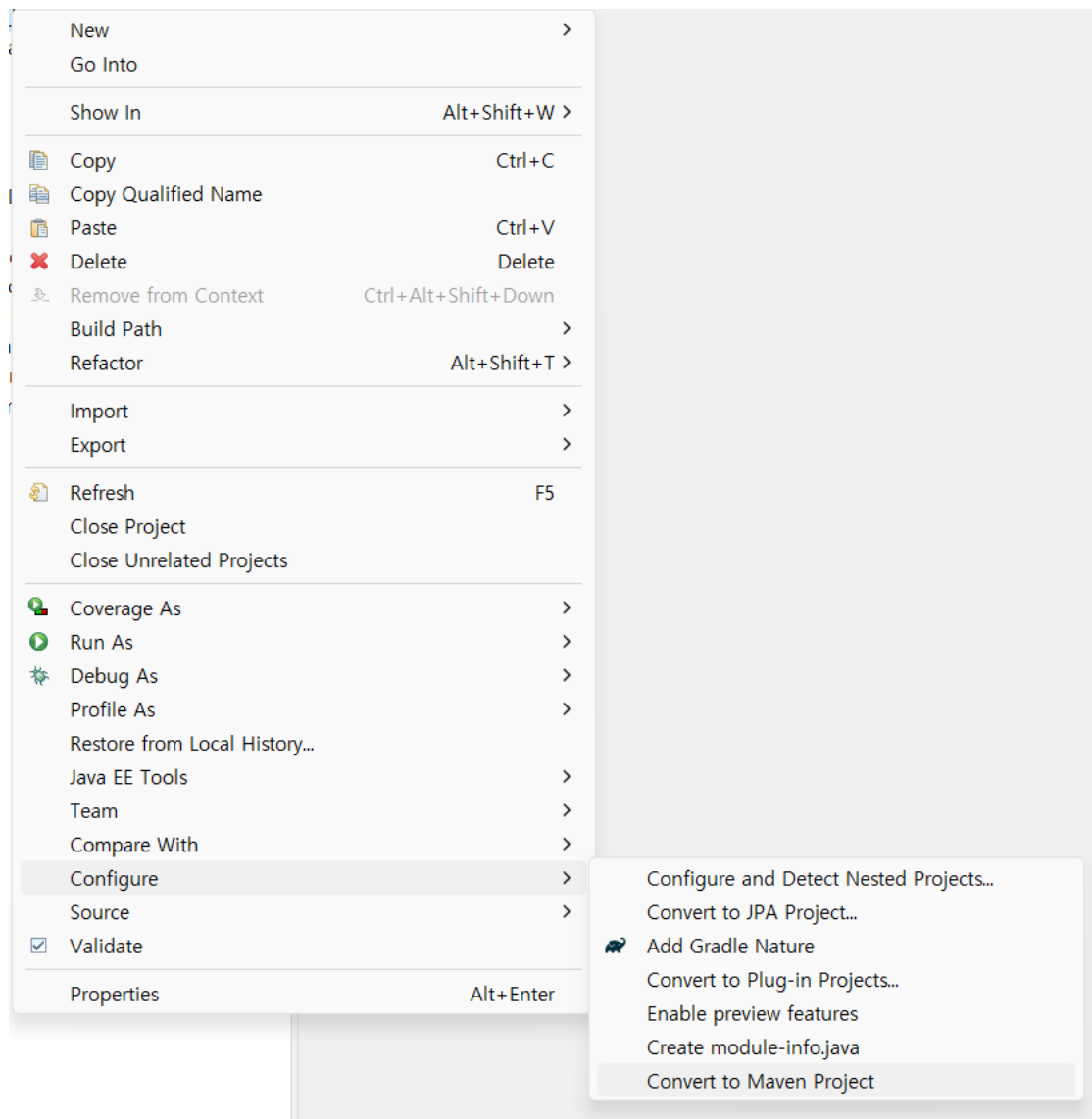
Maven과 Gradle 모두 기본적으로는 설정 파일을 먼저 만들고

명령줄 인터페이스 Command Line Interface, CLI를 통해 기본 프로젝트 구조를 생성한 다음,
개발도구에서 import하여 사용하는 방식.

동적 웹 프로젝트 Dynamic Web Project를 Maven 기반으로 변환해
사용.

간편하면서도 안정정인 방법.

1) 변환하려는 프로젝트를 선택하고 마우스 오른쪽 버튼을 눌러
Configure -> Convert to Maven Project를 선택.



2) Maven 프로젝트를 위해 몇 가지 필요한 사항을 등록하고, Finish
를 클릭.

Create new POM

Maven POM

This wizard creates a new POM (pom.xml) descriptor for Maven.

Project: /sample_address_book

Artifact

Group Id: sample_address_book

Artifact Id: sample_address_book

Version: 0.0.1-SNAPSHOT

Packaging: war

Name:

Description:














Finish Cancel

- * Group Id : 프로젝트의 고유 식별자로 기본 패키지 이름 규칙에 따라 작성.
- * Artifact Id : 생성되는 jar(war) 파일의 이름으로 소문자로만 작성.
- * Version : 숫자와 점으로 이루어진 버전 관리 번호.
- * Packaging : 빌드 산출물 형태로 jar(일반 앱 혹은 라이브러리) 혹은 war(웹) 형태.

프로젝트 이름으로 생성된 기본값을 사용하며, 웹 프로젝트로 빌드된 결과물은 war 패키지로 생성.

- 프로젝트의 패키지 명은 패키지명 규칙에 따라 역도메인 (예 : com.google.xxx, com.naver.blog.xxx) 방식을 따르는 것이 좋음

3) 자동으로 기존 프로젝트 구조에서 몇몇 폴더 구조와 'pom.xml' 파일이 생성된 것을 확인.

- >  Deployment Descriptor: sample_address boo
- >  JAX-WS Web Services
- >  JRE System Library [JavaSE-17]
- >  src/main/java
- >  Server Runtime [Apache Tomcat v9.0]
- >  Deployed Resources
- >  build
- ▼  src
 - ▼  main
 - >  java
 - >  webapp
- >  target
-  pom.xml

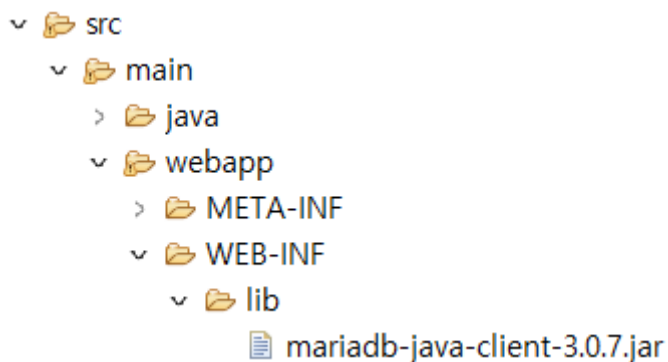
기본적으로 'pom.xml'파일이 오픈된 상태이며, 하단에 여러 탭이 있어 다양한 형태로 설정 파일 정보를 보여줌.

```
sample_address_book/pom.xml ×
1 <project xmlns="http://maven.apache.org/POM/4.0.
2   <modelVersion>4.0.0</modelVersion>
3   <groupId>sample_address_book</groupId>
4   <artifactId>sample_address_book</artifactId>
5   <version>0.0.1-SNAPSHOT</version>
6   <packaging>war</packaging>
7   <build>
8     <plugins>
9       <plugin>
10         <artifactId>maven-compiler-plugin</artif
11         <version>3.8.1</version>
12         <configuration>
13           <release>17</release>
14         </configuration>
15       </plugin>
16       <plugin>
17         <artifactId>maven-war-plugin</artifactId>
18         <version>3.2.3</version>
19       </plugin>
20     </plugins>
21   </build>
22 </project>
```

Overview Dependencies Dependency Hierarchy Effective POM pom.xml

4) 프로젝트에 필요한 라이브러리를 등록.

WEB-INF에 복사해 두었던 mariadb 관련 라이브러리를 삭제.



5) 필요한 라이브러리는 리포지터리 사이트에서 검색한 버전에 맞게 선택한 화면에서 코드를 복사해서 사용.

메이븐 리포지터리 <https://mvnrepository.com/> 에 접속하여 'mariadb'를 검색하고 3.0.7 버전의 코드를 복사하여 사용.

MVN REPOSITORY

mariadb

Repository

Central

110

Sonatype

15

Spring Plugins

11

OpenHAB

8

Spring Lib M

7

ONAP


3

XWiki Releases

3

Found 139 results

Sort: **relevance** | popular | newest




1. MariaDB Java Client

org.mariadb.jdbc » mariadb-java-client

JDBC driver for MariaDB and MySQL

Last Release on Jan 12, 2023



MariaDB Java Client

JDBC driver for MariaDB and MySQL

License

LGPL 2.1

Categories

JDBC Drivers

Tags

database sql jdbc driver mariadb client mysql

Ranking

#615 in MvnRepository (See Top Artifacts)
#7 in JDBC Drivers

Used By

683 artifacts

Central (99) Clojars (1) Spring Plugins (1) Redhat GA (6) Redhat EA (1) ICM (1)

	Version	Vulnerabilities	Repository	Usa
3.1.x	3.1.1		Central	7
	3.1.0		Central	14
	3.0.10		Central	0
	3.0.9		Central	19
	3.0.8		Central	38
	3.0.7		Central	29
3.0.x	3.0.6		Central	34
	3.0.5		Central	28
	3.0.4		Central	32
	3.0.3		Central	20

Maven Gradle Gradle (Short) Gradle (Kotlin) SBT Ivy Grape Leiningen Buildr

```
<!-- https://mvnrepository.com/artifact/org.mariadb.jdbc/mariadb-java-client -->
<dependency>
  <groupId>org.mariadb.jdbc</groupId>
  <artifactId>mariadb-java-client</artifactId>
  <version>3.0.7</version>
</dependency>
```

<!--

<https://mvnrepository.com/artifact/org.mariadb.jdbc/mariadb-java-client> -->

<dependency>

<groupId>org.mariadb.jdbc</groupId>

<artifactId>mariadb-java-client</artifactId>

<version>3.0.7</version>

</dependency>

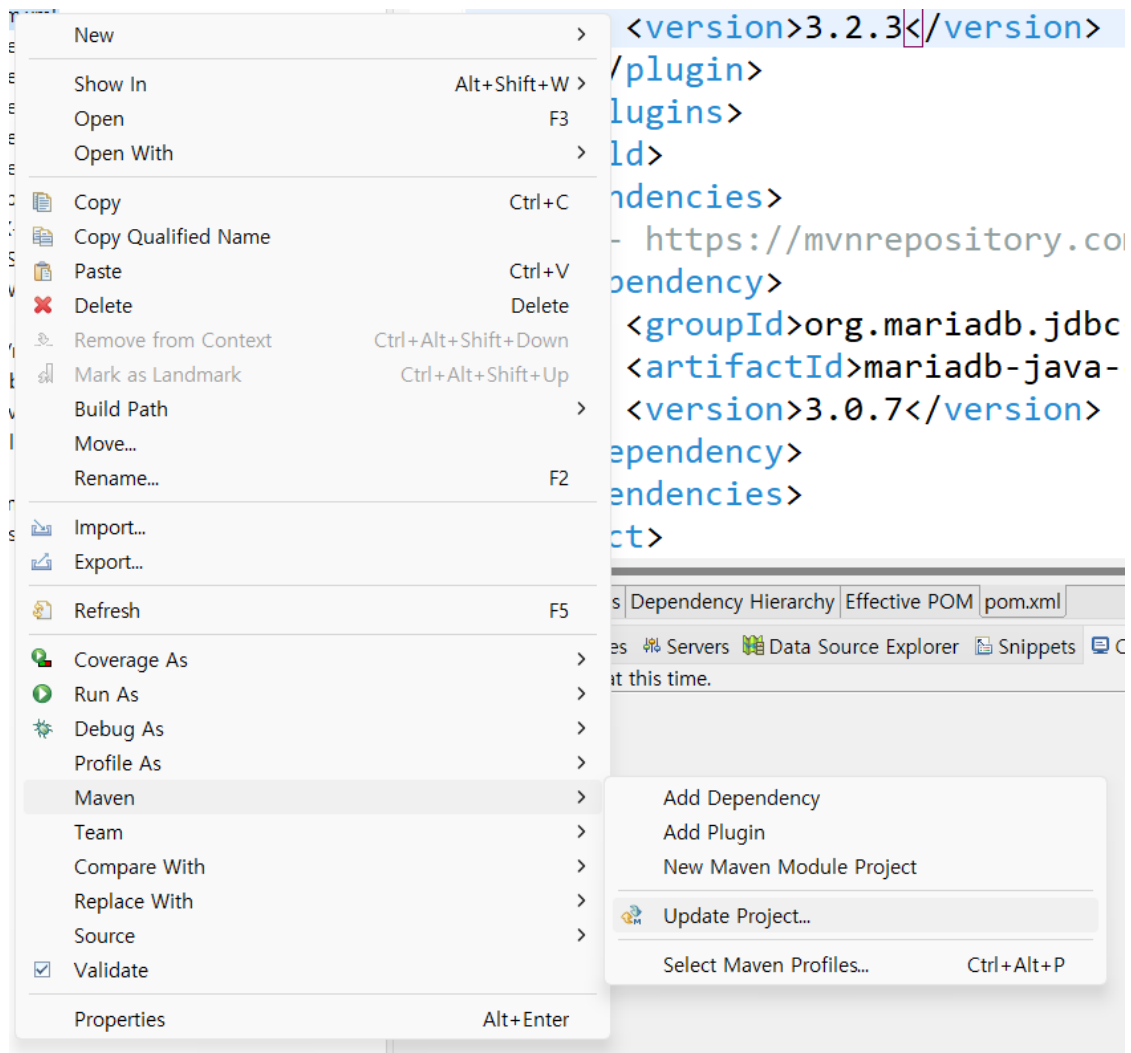
6) 복사한 코드를 pom.xml의 아래 부분에 <dependencies>...
</dependencies>를 추가하고
이 사이에 붙여넣고 mariadb 라이브러리 의존성에 추가.
다른 라이브러리를 추가할 경우 <dependencies>...
</dependencies> 사이에 넣어주어야 함.
라이브러리에 따라 참조하는 다른 라이브러리가 있다면 함께 다운로드.

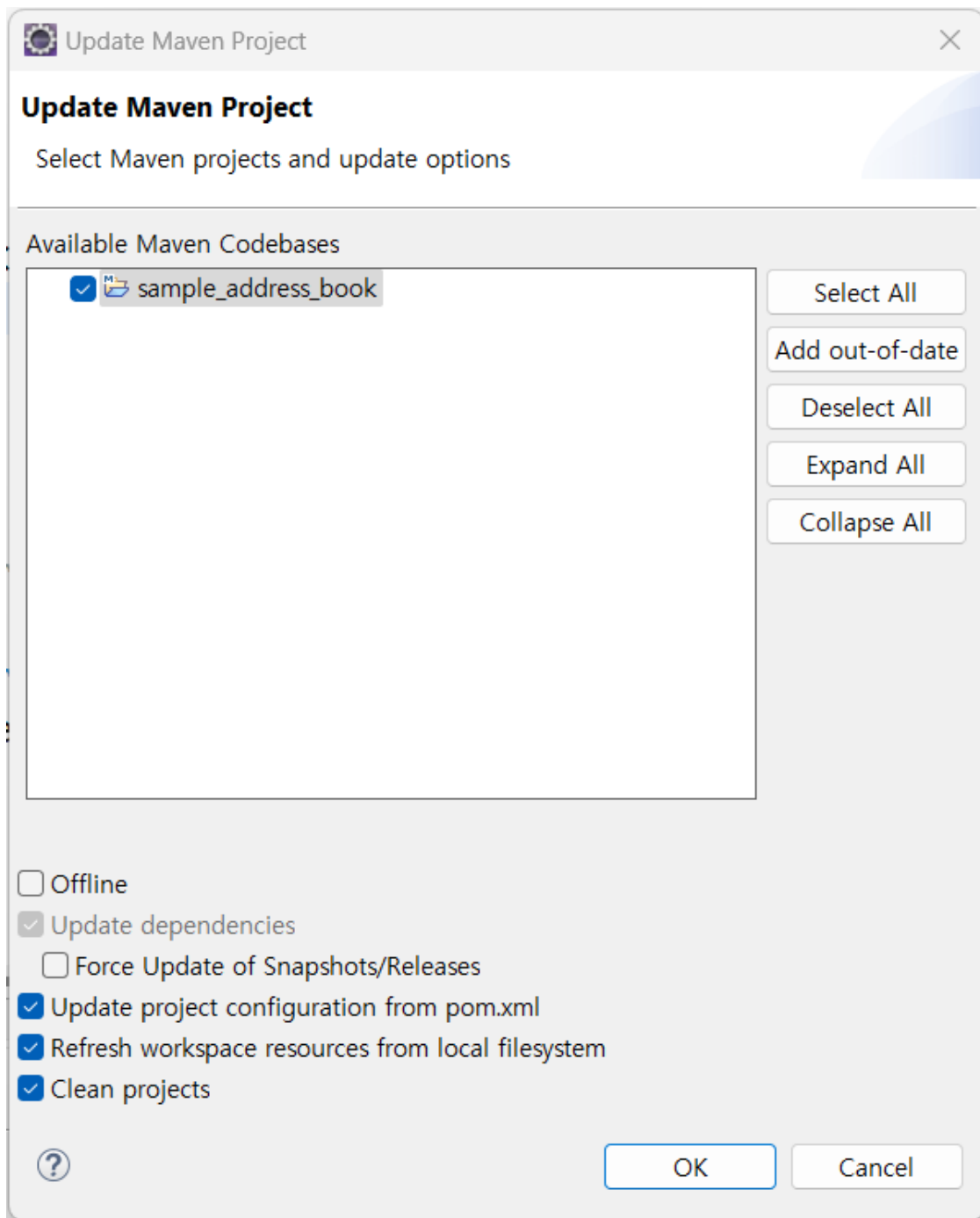
```
sample_address_book/pom.xml ×
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xs:
2   <modelVersion>4.0.0</modelVersion>
3   <groupId>sample_address_book</groupId>
4   <artifactId>sample_address_book</artifactId>
5   <version>0.0.1-SNAPSHOT</version>
6   <packaging>war</packaging>
7   <build>
8     <plugins>
9       <plugin>
10        <artifactId>maven-compiler-plugin</artifactId>
11        <version>3.8.1</version>
12        <configuration>
13          <release>17</release>
14        </configuration>
15      </plugin>
16      <plugin>
17        <artifactId>maven-war-plugin</artifactId>
18        <version>3.2.3</version>
19      </plugin>
20    </plugins>
21  </build>
22 </project>

23 <dependencies>
24   <!-- https://mvnrepository.com/artifact/org.mariadb.jdbc/mariadb-java-client -->
25   <dependency>
26     <groupId>org.mariadb.jdbc</groupId>
27     <artifactId>mariadb-java-client</artifactId>
28     <version>3.0.7</version>
29   </dependency>
30 </dependencies>
31 </project>
```

7) 파일을 저장한 다음, pom.xml 파일을 프로젝트 탐색기에서 선택
한 다음 마우스 오른쪽 버튼을 눌러,

Maven -> Update Project...를 선택하고 나오는 화면에서 OK 버튼을 클릭.





8) 이클립스는 해당 라이브러리를 로컬 리포지터리에 다운로드하고 프로젝트에 참조할 수 있는 상태로 만듦.
프로젝트 탐색기에서 Maven Dependencies 에서 해당 라이브러리가 추가된것 확인.

