

1. **Maven** 개요
2. **Maven** 디렉터리 구조
3. 빌드 Lifecycle
4. **Maven** 플러그인
5. 의존성 관리 메커니즘
6. **Maven** 저장소
7. 프로젝트 객체 모델(**Project Object Model**)
8. **Maven** 이클립스 통합: m2eclipse
9. 참고자료

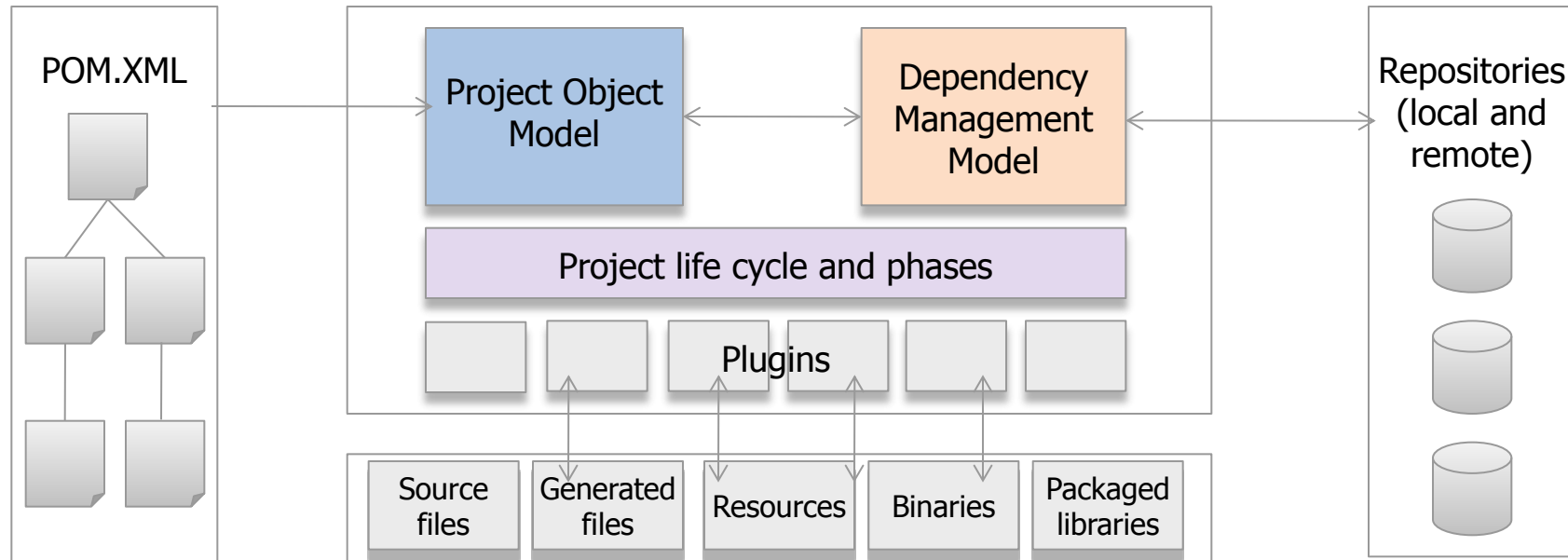
불필요한 설정을 최소화 한다는 개념 아래 **Ant**와 같은 빌드 기능을 제공할 뿐 아니라 구조화 된 빌드 기능을 통해 **learning curve** 및 재사용 성을 향상 시킴

□Maven 특징

Maven 장점	Maven 단점
<ul style="list-style-type: none">• 뛰어난 의존성 관리 의존성 자동 업데이트 저장소를 통한 라이브러리 일괄 관리•모든 프로젝트에 걸쳐 쉽게 적용 가능한 일관적인 사용법•라이브러리 및 메타 데이터 저장을 위한 지속적으로 확장되고 있는 저장소•쉽게 작성 가능한 플러그 인을 통한 확장성•동시에 다수의 프로젝트 핸들링 할 수 있는 쉬운 설정 기반의 메커니즘•간단한 설정을 통한 배포 관리•Java, C++ 등 다수의 프로그래밍 언어 지원	<ul style="list-style-type: none">• repository 관리의 불편함<ul style="list-style-type: none">– Maven 프로젝트의 급속한 발전으로 central repository가 제공하는 라이브러리들이 급속히 증가하고 있으나 아직 3rd 파티 라이브러리 등 미제공 라이브러리들이 있음• pom.xml 파일 관리<ul style="list-style-type: none">–메이븐 프로젝트 관리에 대한 모든 내용이 pom.xml 파일에 담기게 되므로 길고 장황하게 될 수 있음• 프로젝트에 특화된 복잡한 빌드 기능 제약<ul style="list-style-type: none">– 메이븐 프로젝트 특성상 소프트웨어 빌드에 통용되는 라이프 사이클을 제공하고 있어 세부 항목 또는 특화된 빌드 환경에 대한 지원이 미약함

의존성 관리, 라이브러리 관리, 빌드 생명 주기 및 단계 객체 모델을 갖는 프로젝트 관리 도구로 표준화된 빌드 기능 뿐만 아니라 리포팅 및 **documentation** 생성 기능 등을 제공

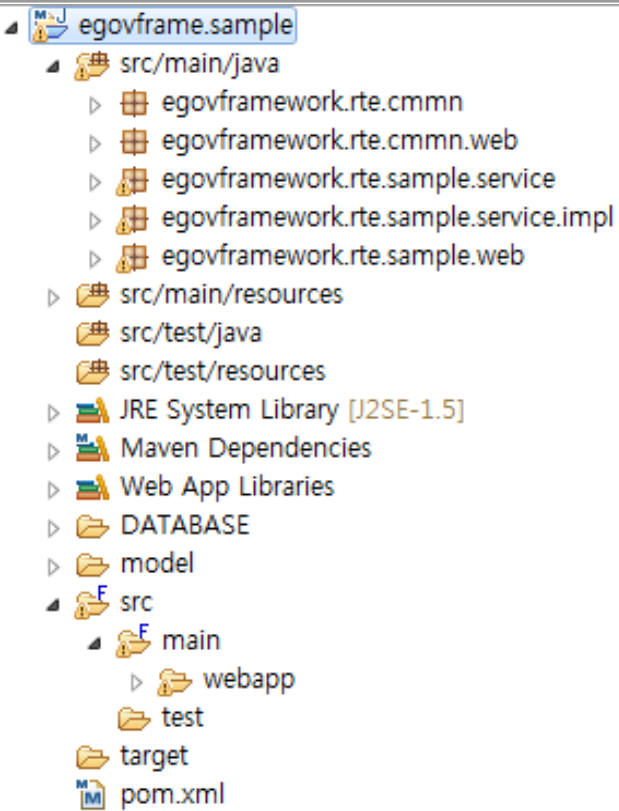
□Maven 아키텍처



- **프로젝트 객체 모델(POM)**: 메이븐 엔진 내장 + POM.XML 파일에서 선언적으로 제공
- 의존성 관리 모델: 로컬 및 리모트 저장소를 이용하여 관리
- 빌드 생명주기와 각 단계: 잘 정의된 단계들과 빌드 사이클에 따라 플러그인들을 조율

Best practices을 기반으로 정규화된 디렉터리 구조를 제공하고 있으며 모든 소스 파일들은 **/src** 디렉터리 밑에 빌드 된 **output**은 **/target** 디렉터리 밑에 위치함

Maven 표준 디렉터리 구조

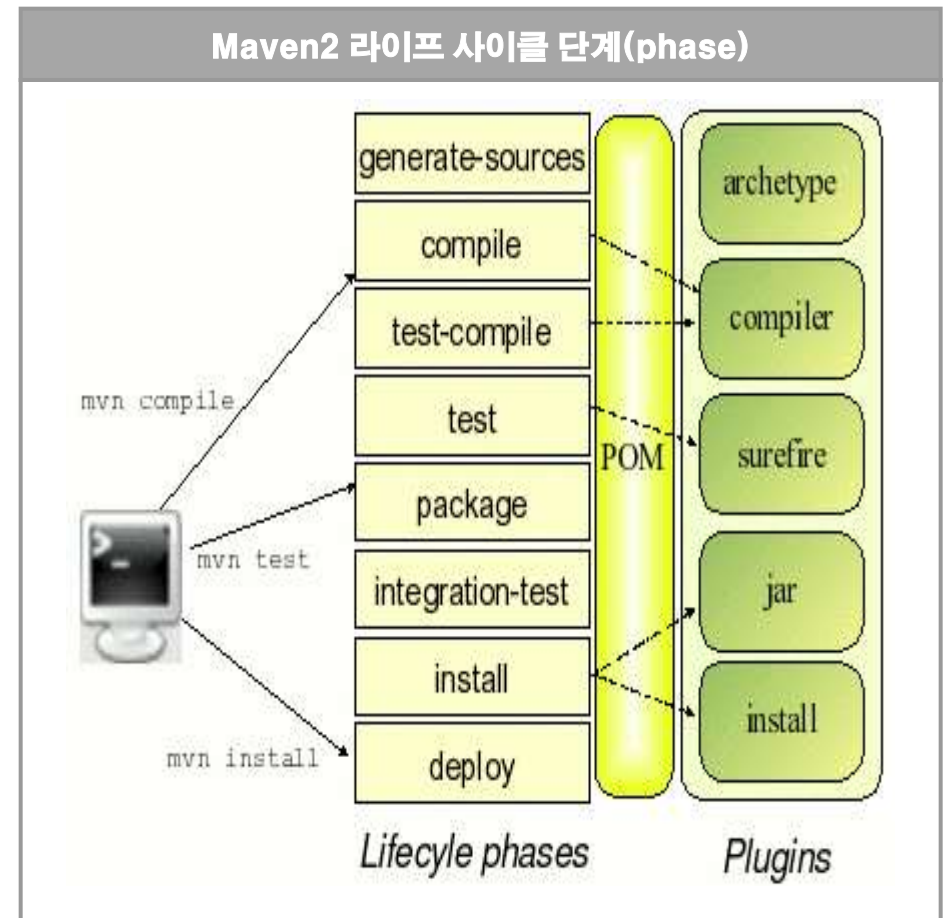


디렉터리/파일	설명
/pom.xml	프로젝트 객체 모델. 해당 프로젝트에 대한 전반적인 정보를 갖는다.
/src/main/java	Java 소스 파일 위치
/src/main/resources	배포할 리소스, XML, properties, ...
/src/main/webapp	웹 어플리케이션 관련 파일 위치(WEB-INF, css 등)
/src/test/java	테스트 케이스 java 소스
/src/test/resources	테스트 케이스 리소스
/target	빌드 된 output이 위치하는 디렉터리

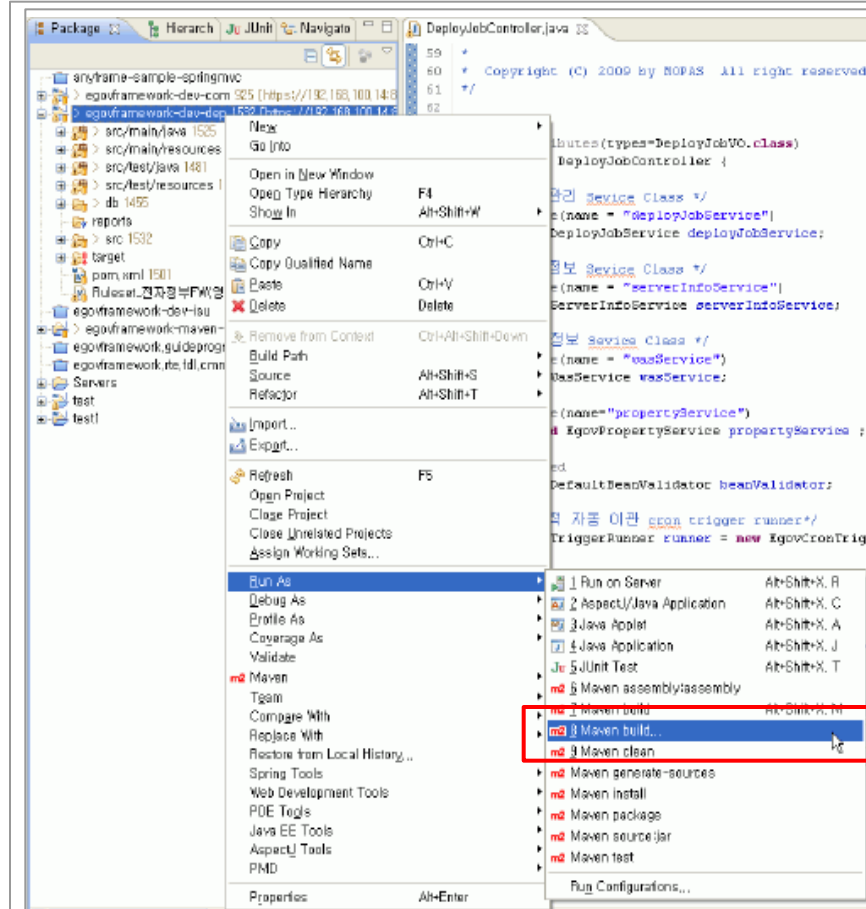
메이븐 빌드는 소프트웨어 프로젝트의 핵심적인 빌드 라이프 사이클 개념을 따르고 있으며 빌드 부터 artifact의 배포까지의 라이프 사이클을 정의하고 있음

빌드 생명 주기 설명	
생명주기 단계	설명
validate	현재 설정과 POM의 내용이 유효한지 확인
generate-sources	코드 생성기가 이 다음의 단계들에서 컴파일 되고 처리할 소스 코드를 생성하기 시작하는 순간
compile	소스 코드를 컴파일 한다. 컴파일 된 클래스들은 타겟 디렉터리 트리 구조에 저장된다.
test	컴파일 된 단위 테스트를 실행하고 그 결과를 표시한다.
package	실행 가능한 바이너리 파일들을 WAR나 JAR같은 배포용 압축 파일로 묶는다.
install	압축 파일을 로컬 메이븐 저장소에 추가한다.
deploy	압축 파일을 원격 메이븐 저장소에 추가한다.

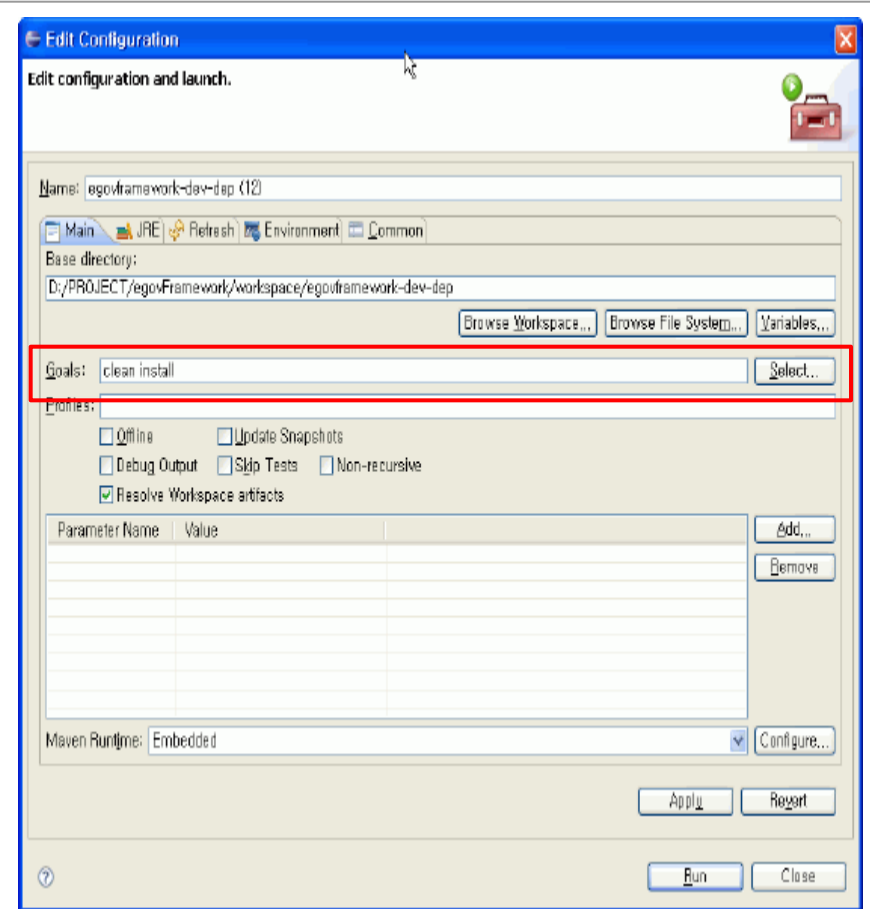
\$mvn compile



이클립스에서 빌드 라이프 사이클에서 정의한 각 단계들 실행하기

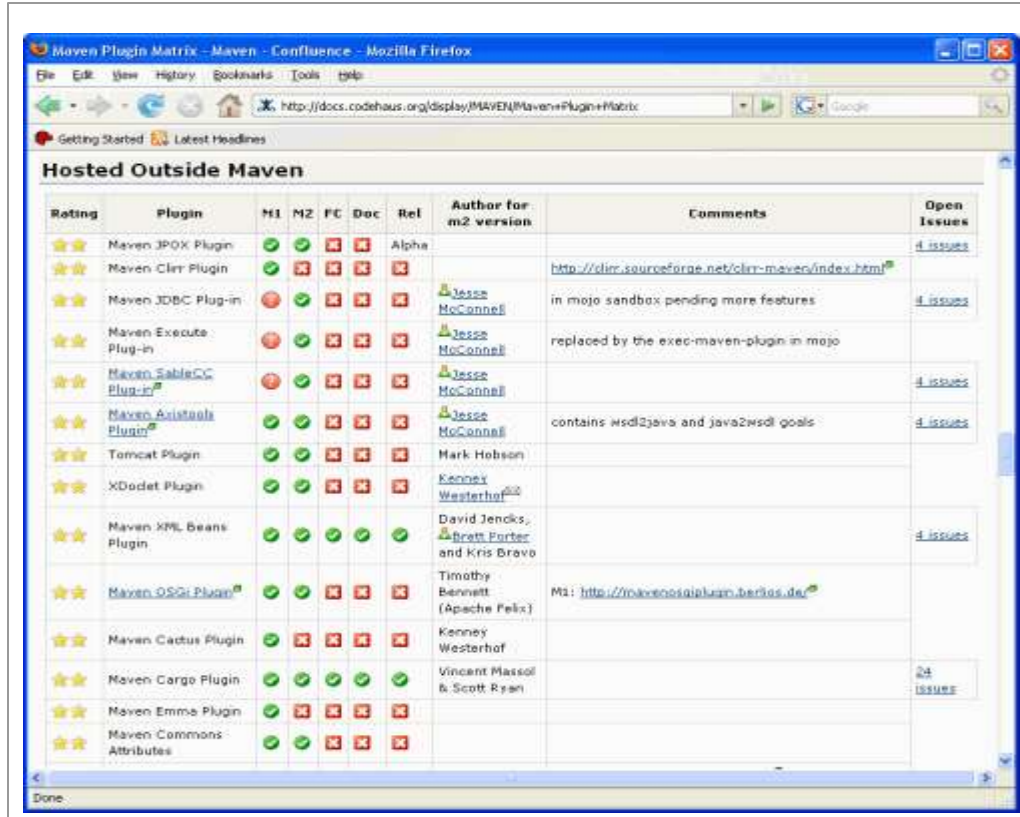


메이븐 프로젝트 -> Run As -> Maven build..



Goals: clean install

메이븐은 대부분의 빌드 작업 시에 플러그인을 활용하며 메이븐 엔진이 생명주기 단계들과 플러그인을 바인딩 시켜 빌드 작업을 컨트롤 함



Rating	Plugin	M1	M2	FC	Doc	Rel	Author for m2 version	Comments	Open Issues
☆☆	Maven JPOX Plugin	✓	✓	✗	✗	Alpha			4 issues
☆☆	Maven Clien Plugin	✓	✗	✗	✗			http://clien.sourceforge.net/clien-maven/index.html	
☆☆	Maven JDBC Plug-in	✓	✓	✗	✗		Jesse McConnell	in mojo sandbox pending more features	4 issues
☆☆	Maven Execute Plug-in	✓	✓	✗	✗		Jesse McConnell	replaced by the exec-maven-plugin in mojo	
☆☆	Maven SableCC Plug-in	✓	✓	✗	✗		Jesse McConnell		4 issues
☆☆	Maven Axis2tools Plugin	✓	✓	✗	✗		Jesse McConnell	contains wsdl2java and java2wsdl goals	4 issues
☆☆	Tomcat Plugin	✓	✓	✗	✗		Mark Hobson		
☆☆	XDodet Plugin	✓	✓	✗	✗		Kenney Westerhof		
☆☆	Maven XML Beans Plugin	✓	✓	✓	✓		David Jencks, Brett Porter and Kris Bravo		4 issues
☆☆	Maven OSGi Plugin	✓	✓	✗	✗		Timothy Bennett (Apache Felix)	M1: http://mavenosgiplugin.berlios.de	
☆☆	Maven Cactus Plugin	✓	✗	✗	✗		Kenney Westerhof		
☆☆	Maven Cargo Plugin	✓	✓	✓	✓		Vincent Massol & Scott Ryan		24 issues
☆☆	Maven Emma Plugin	✓	✗	✗	✗				
☆☆	Maven Commons Attributes	✓	✓	✗	✗				

Maven Plugins

- 플러그인 플랫폼으로써의 메이븐 - 확장 용이
- 자주 사용하는 플러그인들을 모아 패키징 한 상태로 배포
- 대부분의 개발 작업 시에 추가 플러그인 불필요

단계	Plugin	설명
compile	maven-compiler-plugin	소스 코드 컴파일
test	maven-surefire-plugin	단위 테스트 실행
package	maven-jar-plugin	컴파일 된 바이너리 파일들로부터 jar 패키지 생성
emma	maven-emma-plugin	Code coverage report 생성

가용한 Maven 플러그인 매트릭스

□ pom.xml에서의 메이븐 플러그인 선언 예제

```
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>egovframework.dev</groupId>
  <artifactId>my-app</artifactId>
  <version>1.0</version>
  .....
  <pluginManagement>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <configuration>
          <source>${compileSource}</source>
          <target>${compileSource}</target>
          <encoding>${encoding}</encoding>
          <testFailureIgnore>true</testFailureIgnore>
        </configuration>
      </plugin>
      <!-- Egovframework JUnit Excel Reporting -->
      <plugin>
        <groupId>egovframework.dev</groupId>
        <artifactId>egovtest-maven-plugin</artifactId>
        <version>1.0.0-SNAPSHOT</version>
      </plugin>
      <!-- EMMA -->
      <plugin>
        <groupId>org.codehaus.mojo</groupId>
        <artifactId>emma-maven-plugin</artifactId>
        <version>1.0-alpha-1</version>
      </plugin>
      <!-- PMD manven plugin -->
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-pmd-plugin</artifactId>
        <version>2.4</version>
      </plugin>
    </plugins>
  </pluginManagement>
</project>
```


POM에서 선언적인 **dependency** 설정으로 빌드 및 배포 시에 필요한 라이브러리들을 관리하고 로컬 및 원격 저장소에서 선언된 라이브러리들을 다운로드 받아 사용함

의존성 설정

```
<project>
.....
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
  </dependency>
</dependencies>
</project>
```

의존성 scope

- **compile**: 기본값으로 모든 클래스 패스에서 사용 가능
- **provided**: 컴파일과 유사하나 패키지에는 포함되지 않는다. 컨테이너나 JDK에서 제공. 예) Servlet API for web apps
- **runtime**: 컴파일러가 아닌 런타임 시에만 사용됨
예) JDBC drivers
- **test**: 테스트 단계에서만 유용함 예) Junit
- **system**: provided와 유사하나 개발자가 직접 JAR 파일을 제공해야 함.
저장소에서 지정한 dependency를 찾지 않는다.

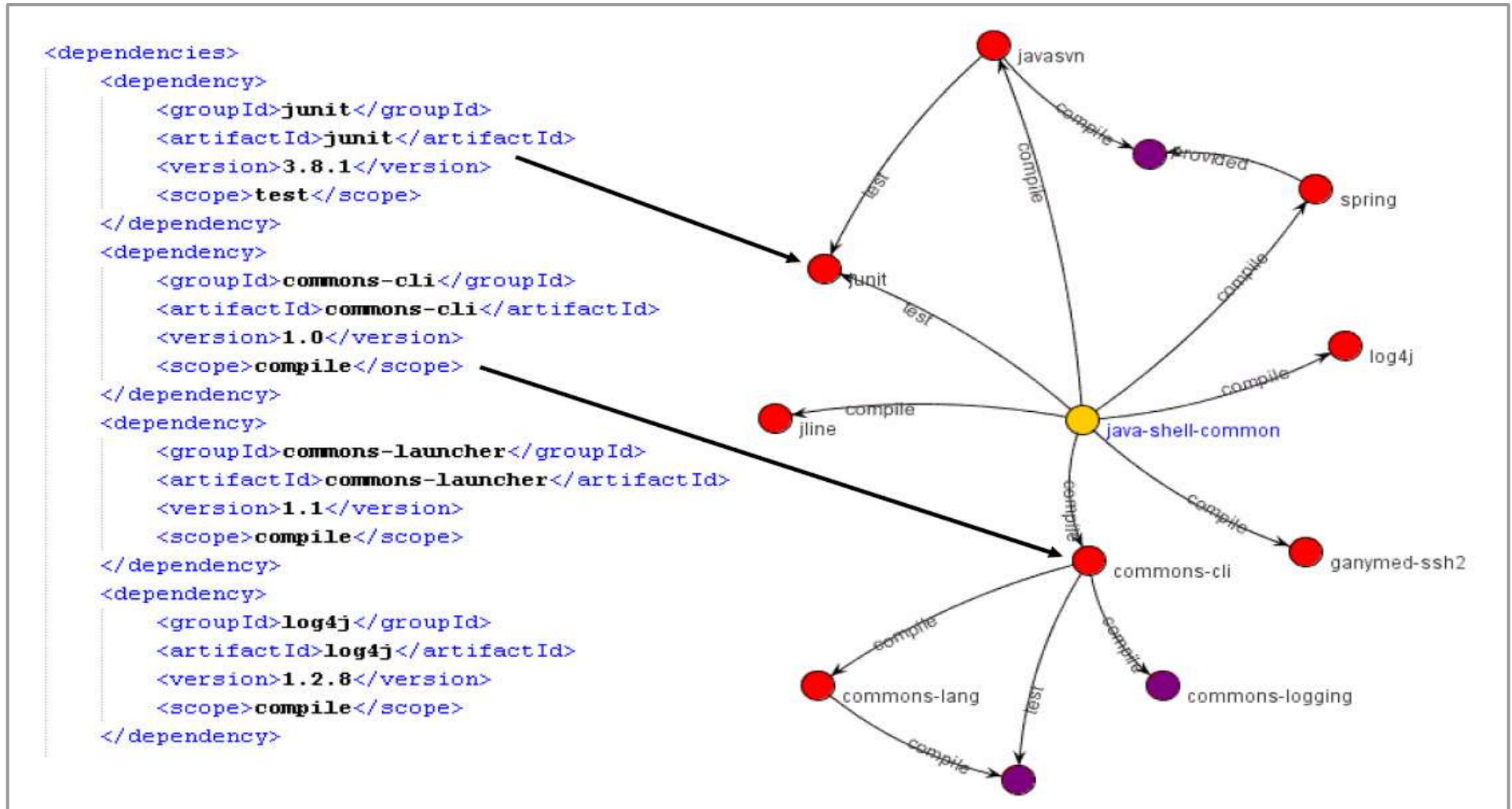
의존성 분석 순서

1. 로컬 저장소에서 의존성 확인
2. 원격 저장소 리스트에서 의존성 확인
3. 1과 2가 실패하면 의존성 에러 보고

의존성 선언 항목

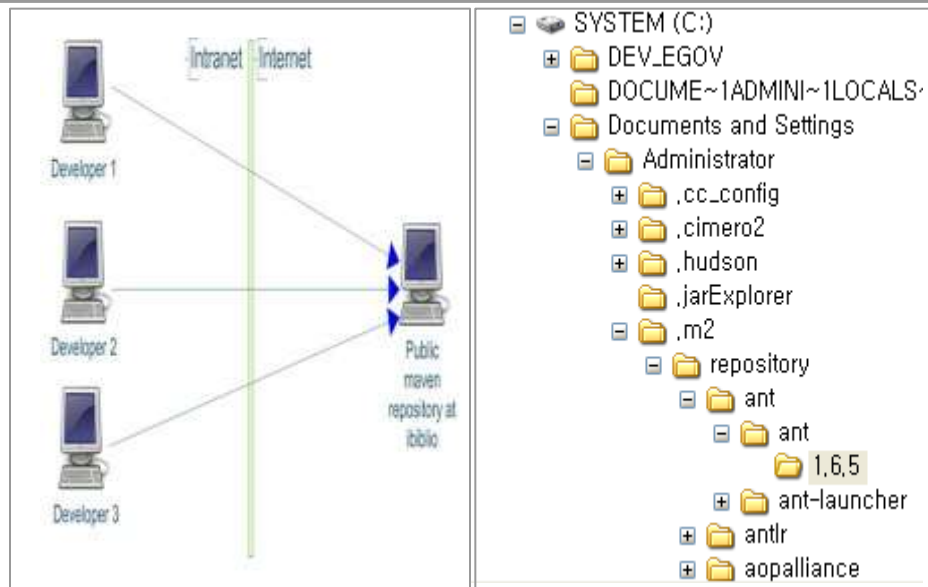
1. **<groupId>**: 부분적인 프로젝트나 조직에서의 라이브러리 집합을 식별하기 위해 제공
2. **<artifactId>**: 프로젝트의 실제 이름으로 groupId와 합쳐져 프로젝트 식별에 쓰임
3. **<version>**: 선언한 의존성 artifact의 버전으로 프로젝트 내에서 사용하는 artifact의 일관성을 추구할 수 있음

□ pom.xml 파일에서의 의존성 설정 및 의존성 분석



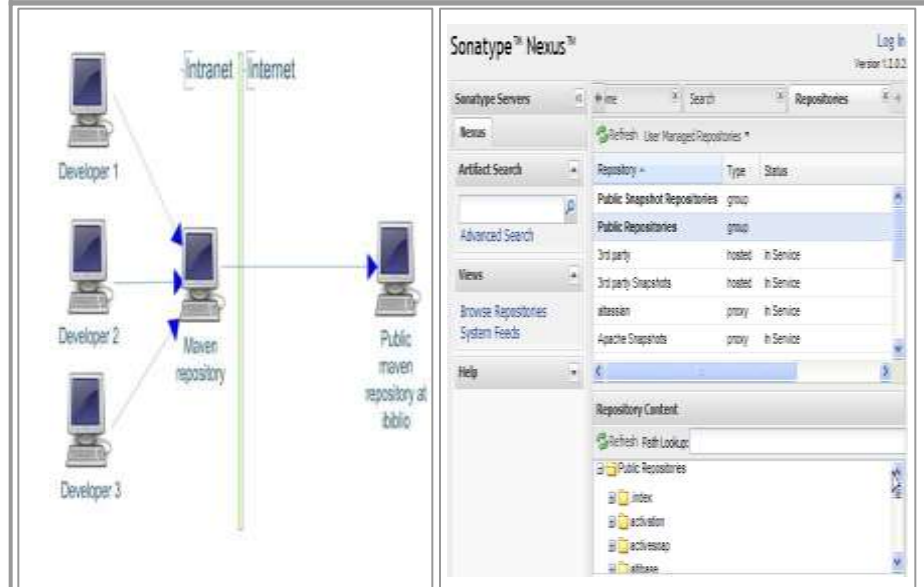
artifact들의 저장소로 로컬 및 리모트 repository로 구성되며 프로젝트는 pom.xml에서 선언한 dependency들을 저장소로부터 불러와서 사용함

메이븐 저장소 구조



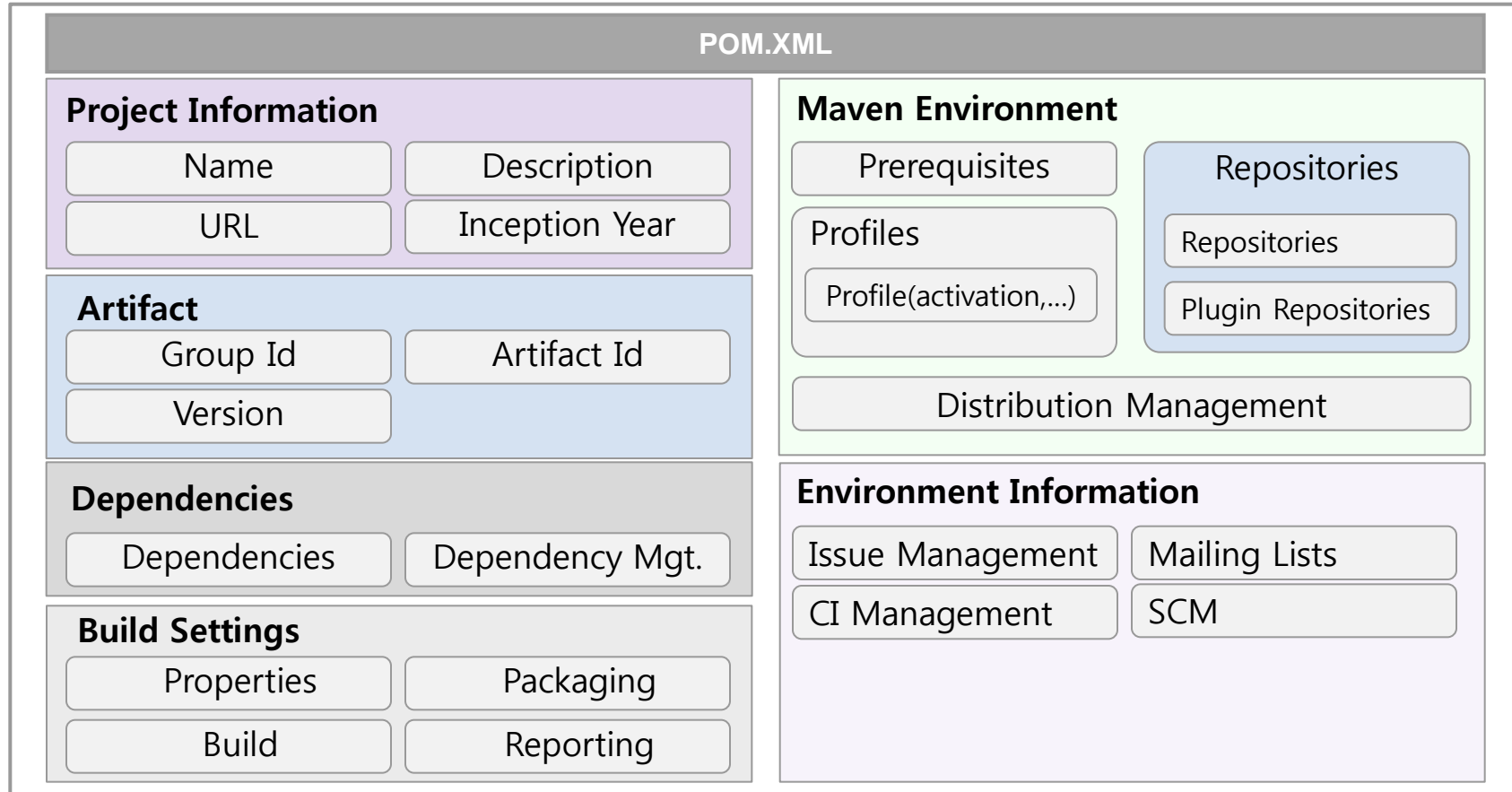
- 로컬 repository는 artifact들을 로컬 파일 시스템에 cache하는 기능으로 리모트 repository에서 다운로드 받아 artifact들을 저장하고 관리한다.
- 리모트 repository는 주 로 HTTP 서버로 3rd 파티에서 제공하는 artifact들을 제외 한 거의 모든 artifact들을 제공한다.

Nexus 연동



- Nexus는 메이븐 repository 관리툴로 리모트 repository의 단점을 보완하고 maven 프로젝트의 사용 편의성을 높이기 위해 사용된다. 각 개발자들은 리모트 repository가 아닌 Nexus에서 dependency를 다운 받아 사용함으로써 프로젝트 내의 artifact 버전 등의 일관성을 유지하고 3rd 파티 artifact 등 라이브리리를 효과적으로 공유할 수 있다.

프로젝트의 구조와 내용을 설명하고 있으며 **pom.xml** 파일에 프로젝트 관리 및 빌드에 필요한 환경 설정, 의존성 관리 등의 정보들을 기술함



- 프로젝트의 세부 메타 데이터 정보를 포함
 - 버전 및 설정 관리, 빌드 환경, 라이브러리 저장소 및 의존성

프로젝트의 pom.xml 파일에 빌드 정보들을 기술 함

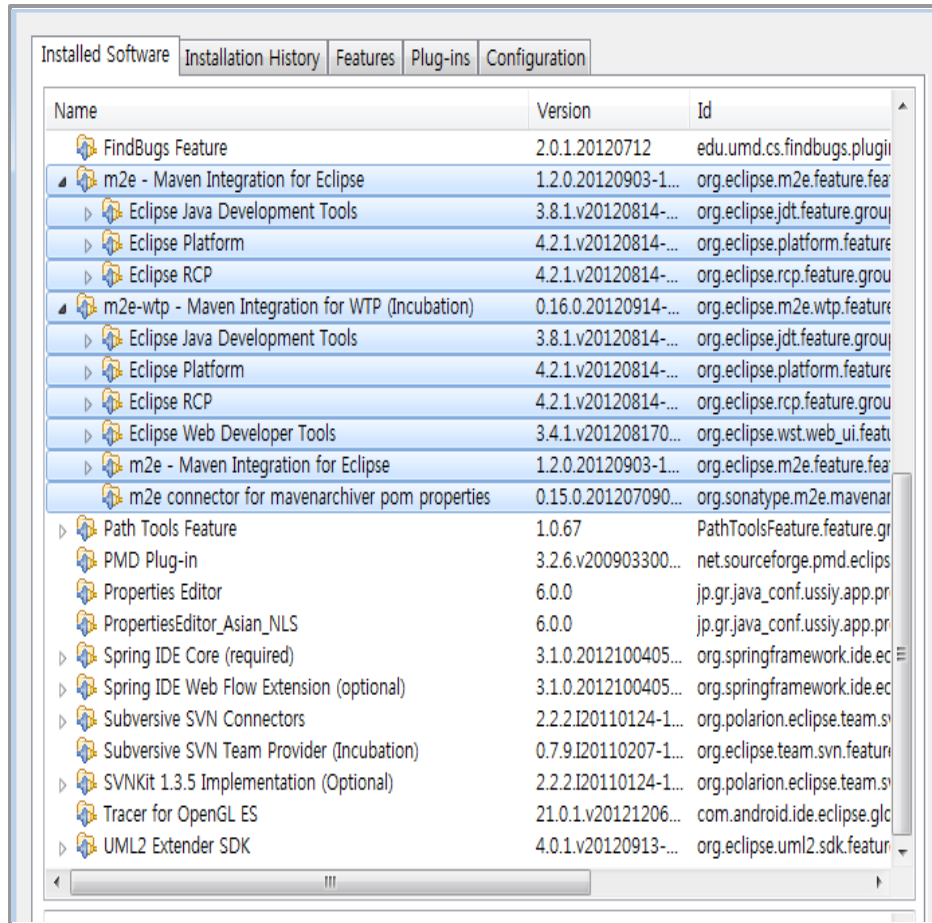
```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>egovframework.dev.com</groupId>
  <artifactId>egovframework-dev-com</artifactId>
  <version>1.0</version>
  <packaging>war</packaging>
  <name>egovframework-dev-com Maven Webapp</name>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.4</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <configuration>
        <source>1.5</source>
        <target>1.5</target>
      </configuration>
    </plugin>
  </plugins>
</project>
```

Artifact 정보

의존성 관리

플러그인 선언

m2eclipse는 Eclipse IDE에서 Maven을 사용하기 위한 플러그인으로 Maven 프로젝트 생성 뿐 아니라 Maven 빌드와 WTP(Web Tools Project)의 통합 등 다양한 기능을 제공함



m2eclipse 특징

- 메이븐 프로젝트 생성 및 불러오기
- 의존성 관리 및 의존성 통합
- dependency 자동 다운로드 및 업데이트
- 리모트 repository 탐색 기능 제공
- POM 파일 관리 화면 제공 및 dependency list에 대한 자동 업데이트
- 다양한 SCM 저장소로부터 메이븐 프로젝트 check out
- 이클립스에서 메이븐 멀티 모듈 프로젝트 생성 기능 제공
- Web Tools Project (WTP)와의 연동
- aspectj Development Tools (AJDT)과의 연동
- Subversion 플러그인과의 연동

❑ 배포된 eGovFramework Archetype을 이용한 메이븐 프로젝트 생성 예

Maven 프로젝트 생성

eGovFramework에서는 메이븐 프로젝트 생성을 위해 두 가지 방식을 제공하고 있음

1. 구현 도구에서 제공하는 **Perspective**를 이용한 Maven 프로젝트 생성
2. Maven **archetype**을 이용한 프로젝트 생성
: 'Archetype is a Maven project templating toolkit.' - 프로젝트에 특화된 pom.xml 및 resource들을 포함한 Maven 프로젝트 생성

Maven Archetype 종류

1. 표준 Archetype
 - maven-archetype-j2ee-simple
 - maven-archetype-quickstart
 - maven-archetype-portlet
2. eGovFramework Archetype
 - egovframework-maven-webapp

* Maven 제공 Archetype 종류:

<http://repo1.maven.org/maven2/org/apache/maven/archetypes/>

Maven 프로젝트의 archetype 선택

New Maven project

Select project name and location

☐ Create a simple project (skip archetype selection)

☒ Use default Workspace location

Location: D:\PROJECT\egovFramework\workspace

Browse...

New Maven project

Select an Archetype

Catalog: Local C:\java\m2\archetype-catalog.xml

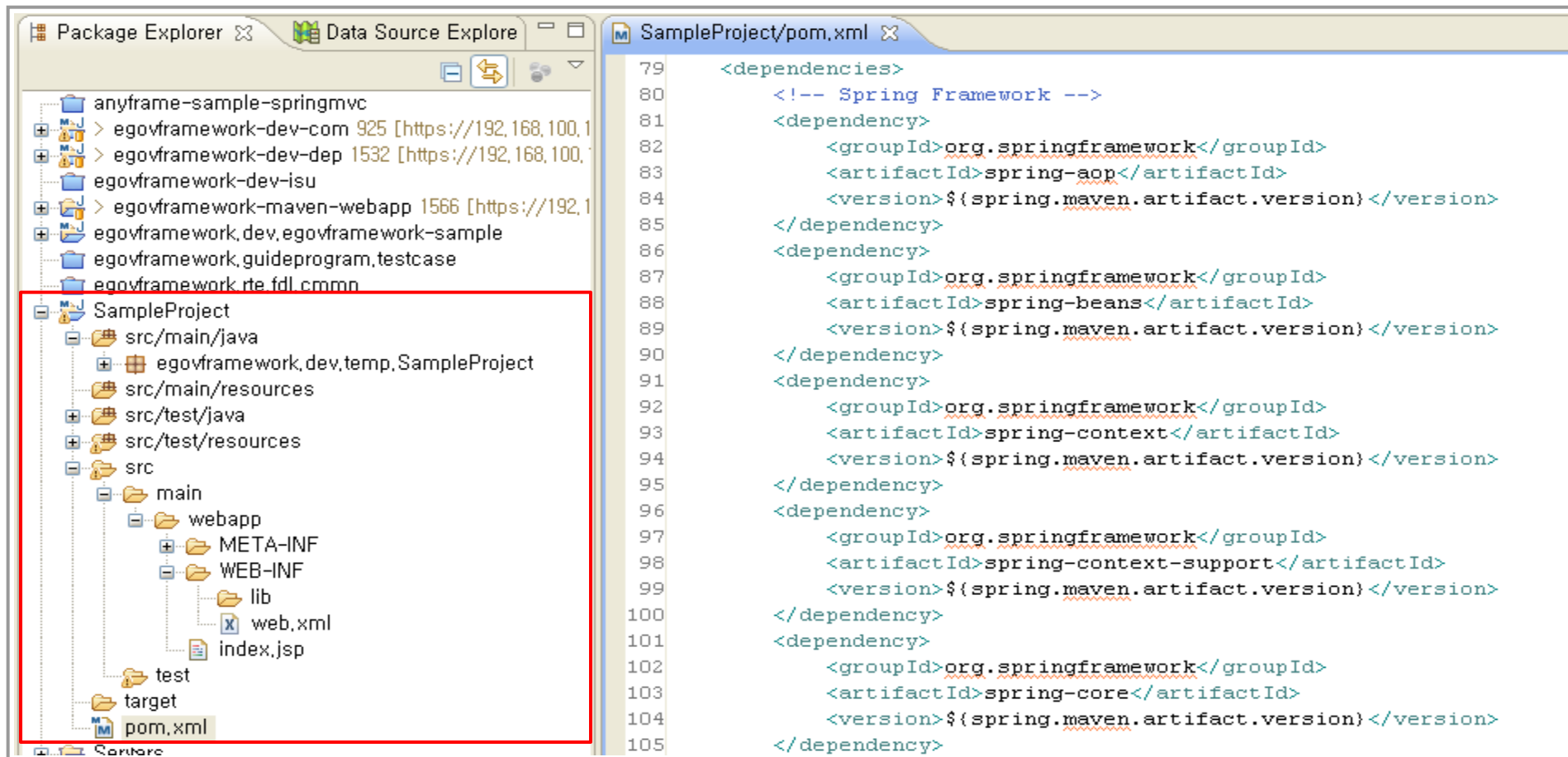
Configure...

Filter:

Group Id	Artifact Id	Version
egovframework,maven,archetypes	egovframework-maven-webapp	1.0

configure.. 버튼 클릭으로 제공된 archetype을 선택

❑ eGovFramework Archetype을 이용한 Maven 프로젝트



The screenshot displays the Eclipse IDE interface. On the left, the 'Package Explorer' shows a project named 'SampleProject' with a red box highlighting its internal structure. The structure includes 'src/main/java', 'src/main/resources', 'src/test/java', 'src/test/resources', 'src', 'main' (containing 'webapp' with 'META-INF', 'WEB-INF', 'lib', 'web.xml', and 'index.jsp'), 'test', and 'target'. The 'pom.xml' file is selected at the bottom. On the right, the 'SampleProject/pom.xml' file is open, showing a list of dependencies for Spring Framework artifacts, including 'spring-aop', 'spring-beans', 'spring-context', 'spring-context-support', and 'spring-core'. The dependencies are defined within a <dependencies> block, with each dependency specifying the groupId, artifactId, and version (using a placeholder for the spring.maven.artifact.version).

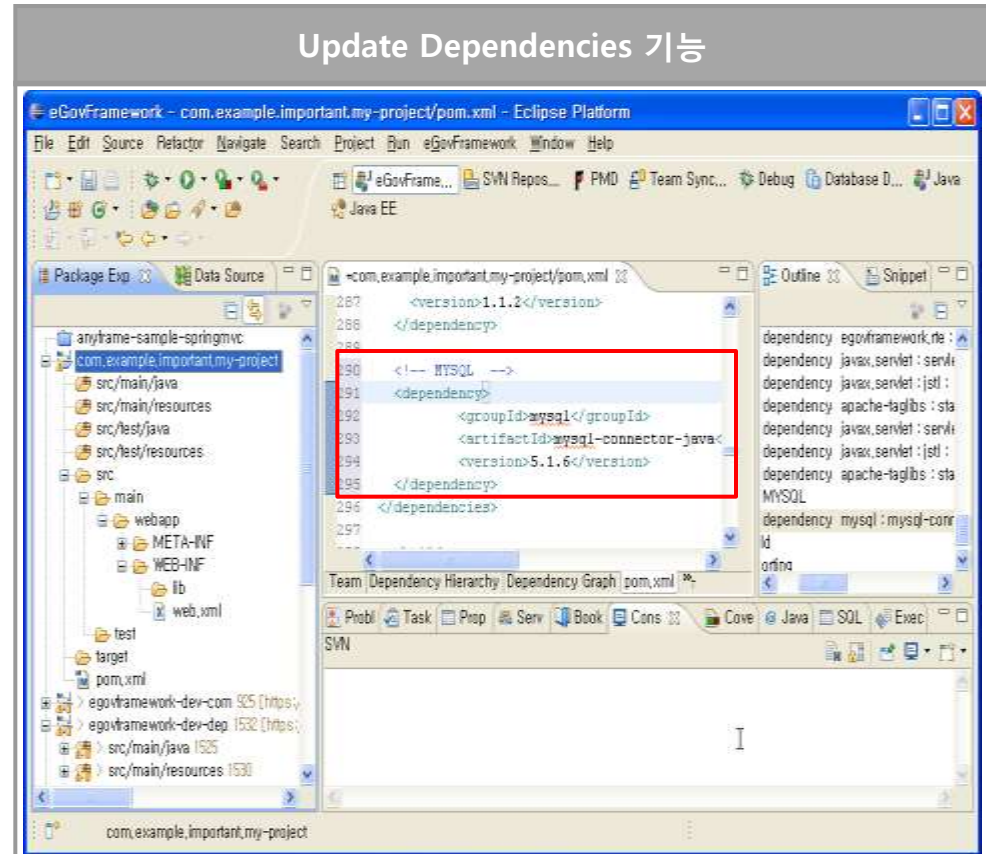
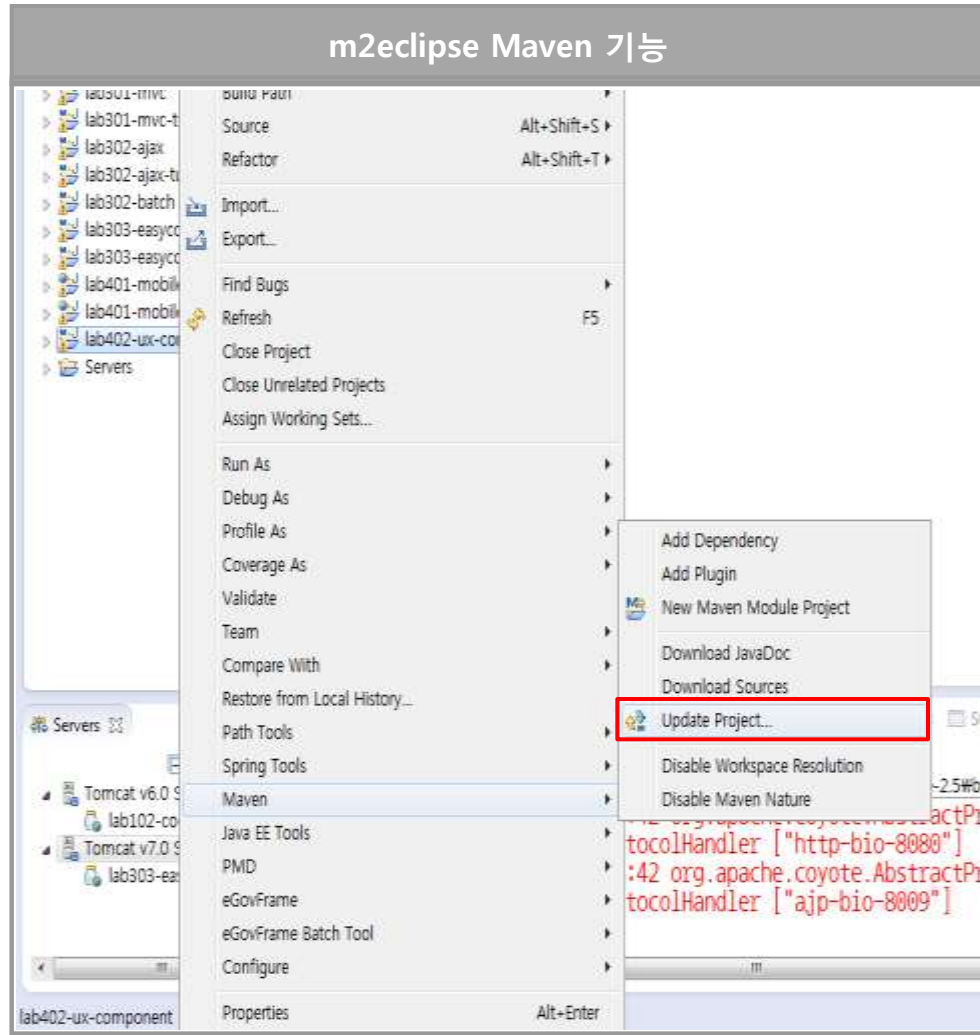
```

79 <dependencies>
80     <!-- Spring Framework -->
81     <dependency>
82         <groupId>org.springframework</groupId>
83         <artifactId>spring-aop</artifactId>
84         <version>${spring.maven.artifact.version}</version>
85     </dependency>
86     <dependency>
87         <groupId>org.springframework</groupId>
88         <artifactId>spring-beans</artifactId>
89         <version>${spring.maven.artifact.version}</version>
90     </dependency>
91     <dependency>
92         <groupId>org.springframework</groupId>
93         <artifactId>spring-context</artifactId>
94         <version>${spring.maven.artifact.version}</version>
95     </dependency>
96     <dependency>
97         <groupId>org.springframework</groupId>
98         <artifactId>spring-context-support</artifactId>
99         <version>${spring.maven.artifact.version}</version>
100    </dependency>
101    <dependency>
102        <groupId>org.springframework</groupId>
103        <artifactId>spring-core</artifactId>
104        <version>${spring.maven.artifact.version}</version>
105    </dependency>

```

- egovframework-webapp Archetype은 Spring 3.0.5 기반의 dependency들을 기본으로 제공함

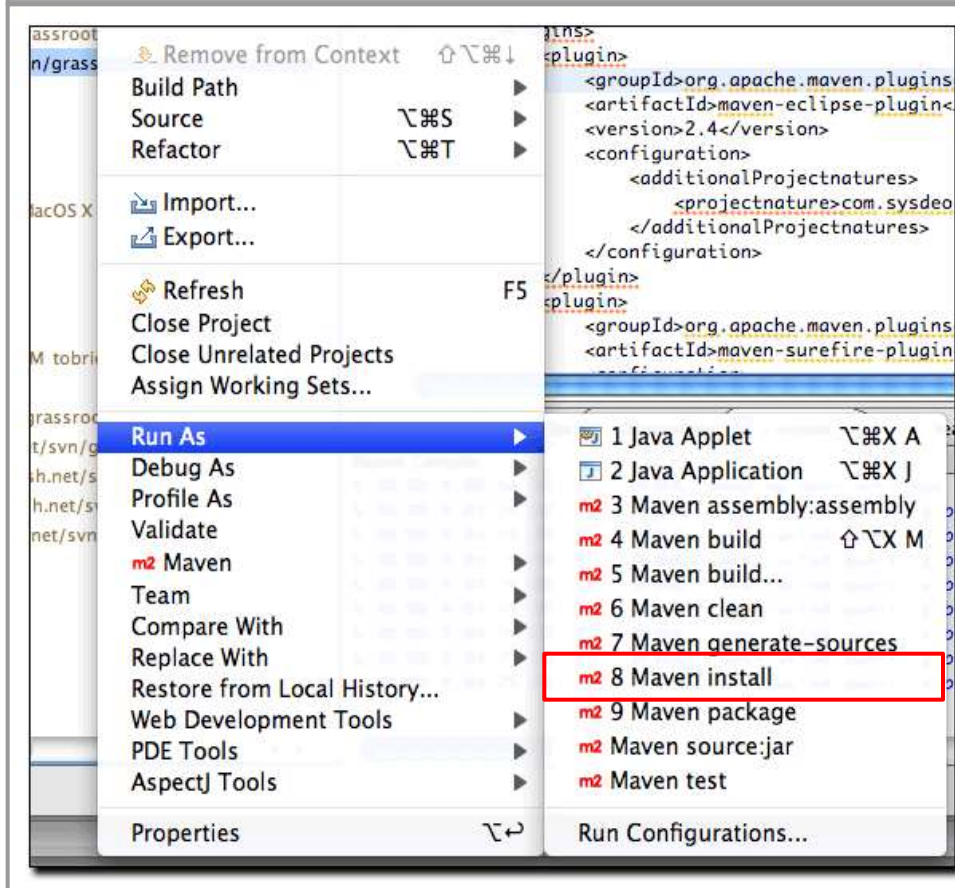
- ❑ m2eclipse에서 제공하는 메뉴에서 메이븐 설정 및 연동 기능을 호출할 수 있음



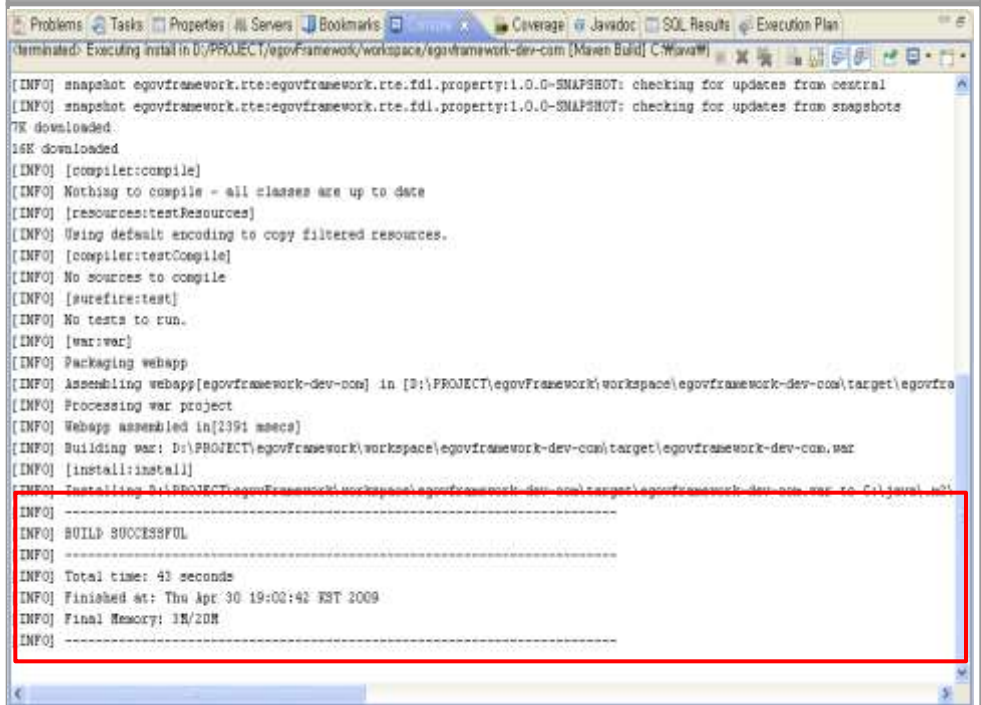
pom.xml에 dependency를 추가하고 **update dependencies** 메뉴를 선택하면 저장소에서 해당 artifact를 찾아 프로젝트에 Cache함

m2eclipse는 이클립스의 Run As 메뉴에 Maven 기본 생명주기 단계를 추가하여 빌드 편의성을 제공함

m2eclipse Maven 빌드 기능



Maven install 기능



Maven install 메뉴를 선택하여 생명 주기의 각 단계를 실행하고 프로젝트를 패키징하여 로컬 repository에 저장함

9. 참고 자료

☐ Apache Maven

- <http://maven.apache.org/>

☐ Maven Central Repository

- <http://mvnrepository.com/>

☐ M2eclipse

- <http://m2eclipse.codehaus.org/>

☐ Nexus

- <http://www.sonatype.com/books/nexus-book/reference/>