|  |  |
| --- | --- |
|  |  |

|  |
| --- |
|  |
| Maven |
| Innovative Web |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

목차

[1 개요 2](#_Toc456257531)

[1.1 Maven 이란 2](#_Toc456257532)

[1.2 디렉터리 구조 2](#_Toc456257533)

[1.3 Maven Lifecycle 3](#_Toc456257534)

[1.3.1 Default Lifecycle 3](#_Toc456257535)

[1.3.2 Clean Lifecycle 3](#_Toc456257536)

[1.3.3 Site Lifecycle 4](#_Toc456257537)

[1.4 Dependency Management 4](#_Toc456257538)

[1.5 Plugin 4](#_Toc456257539)

[1.6 Maven Repository 5](#_Toc456257540)

[2 POM 설정 6](#_Toc456257541)

[2.1 POM 이란 6](#_Toc456257542)

[2.2 속성 설명 6](#_Toc456257543)

[3 Maven 환경설정 8](#_Toc456257544)

[3.1 setting.xml 설정 8](#_Toc456257545)

[3.1.1 Eclipse에서 Settings.xml 설정 8](#_Toc456257546)

[3.2 속성 설명 8](#_Toc456257547)

# 개요

## Maven 이란

Maven(메이븐)은 기본적으로 Build Tool이다. 우리가 생성한 Source 파일을 Build, Deploy를 자동화한 도구이다.

그리고 중요한 기능이 Dependency(의존성) 관리 기능이다. 선언적 설정에 의해 라이브러리를 자동으로 다운로드하고 관리한다.

CoC(Convention over Configuration, 설정 보다 관례) 개념으로 편하게 Build할 수 있는 환경을 제공한다.

1. CoC란?  
   개발자들이 결정해야 되는 부분을 줄여서 개발을 용이하고 단순하게 하면서 유연성을 읽지 않게 하는 소프트웨어 디자인 패러다임이다. 개발 시 관례적으로 하던 설정을 생략하여 생산성을 높이도록 한다.

## 디렉터리 구조

Maven은 아래와 같은 디렉터리 구조 정해져 있으며 변경은 가능하나 변경하지 않는 것이 좋다. (CoC 의 개념을 이해해 보면 된다.)

| **디렉터리 구조** | | | | **설명** |
| --- | --- | --- | --- | --- |
| [*Project\_root]* | | | | 프로젝트 |
|  | src | | |  |
|  |  | main | |  |
|  |  |  | java | Source Code |
|  |  |  | resources | 자원 |
|  |  |  | webapp | Web Source Code |
|  |  | test | |  |
|  |  |  | java | 테스트 소스 |
|  |  |  | resources | 테스트 자원 |
|  | target | | | Build 파일 |
|  | pom.xml | | | 설정 파일 |

## Maven Lifecycle

Maven은 정해진 일련의 Phase에 연계된 Goal을 실행하며 그 논리적인 작업 흐름 집합이 Lifecycle이다.

Maven은 아래와 같이 3개의 표준 Lifecycle을 제공한다.

* **default**: 일반적인 Build 프로세스
* **clean**: Build를 통한 생성된 산출물 제거
* **site**: 프로젝트 문서와 Reporting 작업 수행

### Default Lifecycle

| **Phase / Goal** | **설명** |
| --- | --- |
| validate | 프로젝트의 상태가 정상인기 여부와 Build에 필요한 정보 존재 여부 검사 |
| initialize | Build 상태 초기화. 속성 설정 및 작업 디렉터리 생성 등 |
| generate-sources | Compile에 필요한 Source를 생성 |
| process-sources | Filtering 처리 등 Source Code 가공 |
| generate-resources | 패키지에 포함된 자원 생성 |
| process-resources | 패키지 작업을 준비하기 위해 자원들을 대상 디렉터리로 복사 및 가공 |
| **compile** | Source Code Compile |
| process-classes | Compile된 파일에 대한 후 처리 수행 |
| generate-test-sources | Compile에 필요한 테스트 Source를 생성 |
| process-test-sources | Filtering 처리 등 테스트 Source Code 가공 |
| generate-test-resources | 패키지에 포함된 테스트 자원 생성 |
| process-test-resources | 패키지 작업을 준비하기 위해 테스트 자원들을 대상 디렉터리로 복사 및 가공 |
| test-compile | 테스트 Source Code Compile |
| process-test-classes | Compile된 테스트 파일에 대한 후 처리 수행 |
| **test** | 단위 테스트 수행 |
| prepare-package | 패키지 생성을 이전의 사전 작업 수행 |
| **package** | jar, war 등 패키지 작업 수행 |
| pre-integration-test | 통합 테스트를 실행하기 위한 사전 작업 수행 |
| integration-test | 통합 테스트 수행 |
| post-integration-test | 통합 테스트 이후 후처리 작업 수행 |
| verify | 패키지의 품질 기준에 적합한지 여부를 검사 |
| **install** | 패키지를 로컬 저장소에 설치 |
| **deploy** | 패키지를 원격 저장소에 배포 |

### Clean Lifecycle

| **Phase / Goal** | **설명** |
| --- | --- |
| pre-clean | 사전 작업 수행 |
| **clean** | Build에서 생성된 모든 파일을 삭제 |
| post-clean | 사후 작업 수행 |

### Site Lifecycle

| **Phase / Goal** | **설명** |
| --- | --- |
| pre-site | 사전 작업 수행 |
| **site** | 프로젝트 사이트 문서 생성 |
| post-site | 사후 작업 수행 |
| **site-deploy** | 생성된 사이트 문서를 웹 서버로 배포 |

## Dependency Management

Maven은 POM에서 선언한 Dependency를 관리하고 Repository에서 라이브러리를 다운로드 하여 Local Repository에 저장한다.

* 분석 순서
  1. Local Repository 확인
  2. Remote Repository (Default. Maven Central Repository) 확인
  3. 확인 실패 시 에러
* 선언 속성
  1. groupId: 부분적인 프로젝트나 조직에서의 라이브러리 집합
  2. artifactId: 프로젝트의 실제 이름으로 groupId와 합쳐져 프로젝트 식별
  3. version: artifact의 버전
  4. scope: 라이브러리 범위 지정(default: compile)
* scope
  1. compile: 기본값으로 모든 클래스 패스에서 사용한다.
  2. provided: 컴파일과 유사하나 패키지에는 포함되지 않는다.
  3. runtime: 컴파일에서 사용하지 않고 런타임에서만 사용한다.
  4. test: 테스트 단계에서만 사용한다.
  5. system: provided와 유사하나 직접 jar파일을 제공해야 하고 Repository에서 찾지 않는다.

## Plugin

Maven의 모든 Phase는 Plugin(<https://maven.apache.org/plugins/>)으로 구성되어 있다. Lifecycle에 따라 필요한 Plugin을 다운로드 받아서 수행하게 된다.

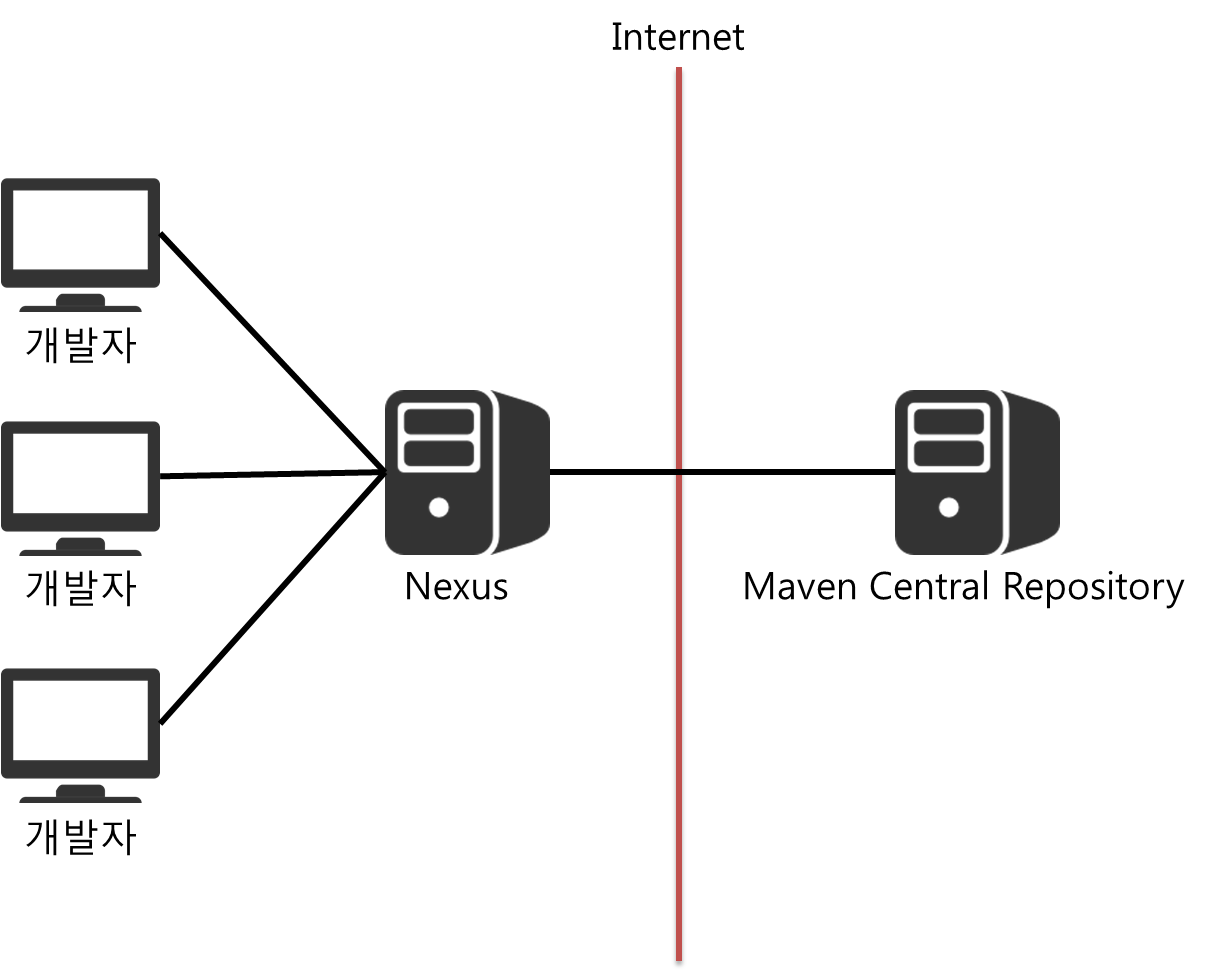
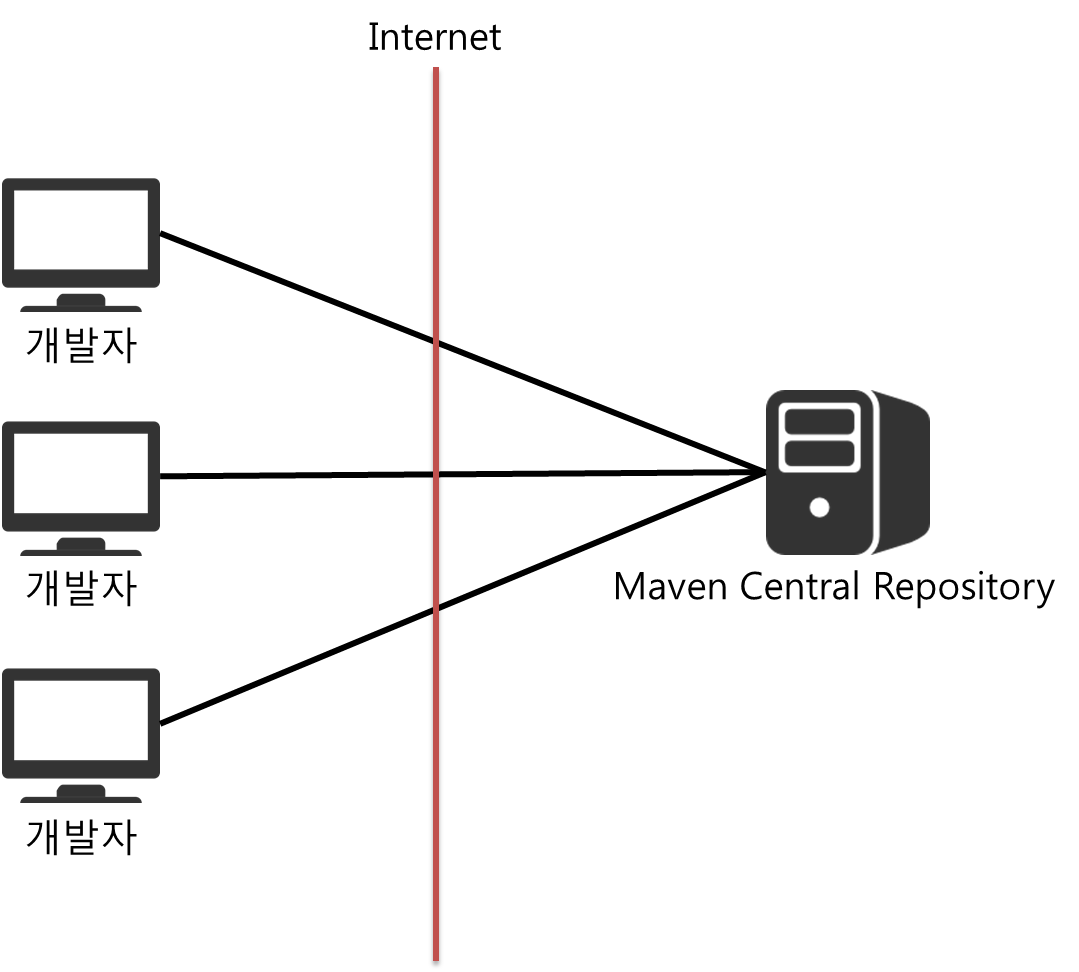
Phase의 처리 중에 설정이 필요하거나 Build 방식을 변경, 기능을 추가하고 싶은 경우 Plugin을 설정하여 사용한다.

|  |
| --- |
| Ex) Eclipse 개발 환경 구성을 지원하는 Plugin |
| <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-eclipse-plugin</artifactId>  <version>2.10</version>  <configuration>  <wtpContextName>/</wtpContextName>  </configuration>  </plugin> |

## Maven Repository

Dependency로 설정된 라이브러리 및 배포하는 패키지를 저장하고 있는 곳이 Repository이다. Maven은 기본적으로 Maven Central Repository(<https://repo1.maven.org/maven2/>)에서 라이브러리를 다운로드 받는다.

회사 내에서 모든 개발자가 Maven Central Repository에서 라이브러리를 다운로드 받는 것이 좋은 구성이 아닐 수도 있다. 회사에서 생성되는 패키지의 배포도 문제가 될 수도 있다. 이러한 문제를 해결하기 위해서 Nexus(<http://www.sonatype.com/download-oss-sonatype>)를 사용하여 Maven Repository를 구성하고 모든 개발자는 Nexus Repository를 바라보게 하여 효율적인 관리를 할 수 있다.



* 현재 구축된 Nexus
* Nexus 2.x (향후 3.x 업그레이드 필요)
* <http://172.16.34.78:8081/nexus/> admin/admin123

# POM 설정

## POM 이란

POM(Project Object Model, 프로젝트 객체 모델)로 프로젝트 당 하나의 pom.xml 파일이 존재한다. pom.xml 파일에 설정 및 의존성 정보가 포함되어 있다.

Maven은 기본으로 Super POM을 가지고 있으며 관례적으로 설정하는 부분이 명시되어 있다. (<https://maven.apache.org/guides/introduction/introduction-to-the-pom.html#Super_POM>)

## 속성 설명

| **Attribute** | **설명** |
| --- | --- |
| **POM Relationships** | |
| groupId | 프로젝트나 조직을 구분하기 위한 식별자 |
| artifactId | 프로젝트의 실제 이름 |
| version | 프로젝트의 버전 |
| parent | 하위 Module인 경우 상위 Project를 설정 |
| modules | Parent Project인 경우 하위 Module의 정보 설정 |
| dependencies | 의존성 설정 |
| dependencyManagement | Parent Project인 경우 의존성 관리를 위한 전체 설정 |
| **Project Information** | |
| name | 프로젝트 이름 |
| description | 프로젝트 설명 |
| licenses | 라이선스 정보 |
| developers | 개발자 정보 |
| organization | 조직 정보 |
| url | 프로젝트 URL |
| contributors | 참여자 정보 |
| inceptionYear | 제작 년도 |
| **Build Settings** | |
| properties | 환경변수 설정 |
| packaging | 패키지 할 파일 형식 (jar, war) |
| build | Build 환경 설정 및 구성 |
| reporting | 프로젝트 사이트 문서 생성 관련 설정 |
| **Build Environment – Information** | |
| issueManagement | 이슈 관리 정보 |
| ciManagement | 프로젝트 지속적 통합 정보 |
| mailingLists | 메일 목록 |
| scm | 프로젝트의 SCM 사양(SVN,CVS) |
| **Build Environment – Maven Environment** | |
| profiles | Build Profile 설정 |
| repositories | Dependency 라이브러리를 다운로드 받을 정보 |
| pluginRepositories | Plugin 라이브러리를 다운로드 받을 정보 |
| prerequisites | 전제조건 설정 |
| destribtionManagement | 배포 Repository |

# Maven 환경설정

## setting.xml 설정

Maven 실행을 위한 설정 파일로 대부분은 pom.xml에서 설정 가능하다.

### Eclipse에서 Settings.xml 설정

1. [Window] > [Preferences] 메뉴 선택
2. “Preferences” 창의 왼쪽 트리 메뉴에서 [Maven] > [User Settings] 메뉴 선택
3. “User Settings” 항목에서 “Browse……” 버튼을 클릭하여 설정하고자 하는 “setting.xml” 선택
4. [OK] 또는 [Apply] 버튼 클릭

## 속성 설명

| **Attribute** | **설명** |
| --- | --- |
| **localRepository** | Local Repository 경로 설정 (Default: ${user.home}/.m2/repository) |
| pluginGroups | 플러그인 조회에 사용하는 또 다른 그룹 ID를 지정합니다. |
| proxies | 네트워크에서 사용할 Proxy 서버 설정 |
| **servers** | 특정 서버 접속에 필요한 인증 정보 |
| mirrors | 다른 지정된 Repository 사이트를 지정 |
| profiles | Build 시 사용되는 Profile 목록 |

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http://maven.apache.org/xsd/settings-1.0.0.xsd">    <localRepository>D:\MavenRepository</localRepository>  <servers>  <server>  <id>releases</id>  <username>deploy</username>  <password>deploy123</password>  </server>  <server>  <id>snapshots</id>  <username>deploy</username>  <password>deploy123</password>  </server>  </servers>  </settings> |