



통합 구현(Spring, Django)

# Maven 활용하기



한국기술교육대학교  
온라인평생교육원

## 학습내용

- Maven 개념 파악하기
- POM과 Plugin
- Lifecycle과 Profile

## 학습목표

- Maven의 개념을 파악할 수 있다.
- Maven의 핵심 기술인 POM과 Plugin에 대해 설명할 수 있다.
- Maven 기반의 Profile을 구성하는 방법에 대해 설명할 수 있다.

# Maven 개념 파악하기

## 1 Maven이란

### 1》Maven은?

- 1 Apache 프로젝트
- 2 소스 코드로부터 배포 가능한 산출물(Artifact)을 빌드(Build)하는 '빌드 툴(Build Tool)'
- 3 편리한 '프로젝트 관리 툴'

### 2》Maven이 없다면?

라이브러리를 직접 다운받아서 프로젝트 경로에 복사해야 함

Java의 경우 Classpath에 라이브러리를 지정해야 함

다운받은 라이브러리가 프로젝트 팀원에게도 전달되어야 함

# Maven 개념 파악하기

## 1 Maven이란

### 1 Maven은?

Build

소스 코드, 테스트 코드 등을 컴파일

Package

배포 가능한 jar, war 파일 등을 생성

Test

단위 테스트(Unit Test) 등을 실행하고,  
빌드 결과가 정상적인지 점검

Report

빌드 / 패키지 / 테스트 결과를 정리하고,  
빌드 수행 리포트를 생성

Release

빌드 후 생성된 산출물(Artifact)을 로컬 혹은 원격 저장소에  
저장(배포)

## Maven 개념 파악하기

### 2 의존성 관리

#### 1 라이브러리 다운로드 자동화

1 더 이상 필요한(의존성 있는) 라이브러리를 하나씩 다운로드 받을 필요가 없음

2 필요하다고 선언만 하면 Maven이 자동으로 다운로드해줌

#### 2 Maven은 선언적이다?

Maven은 명령식(명령어를 입력)이 아님

사용되는 jar 파일들을 어디서 다운로드 받고, 어느 릴리즈(버전)인지 명시하면, 코딩을 하지 않아도 Maven이 알아서 관리함

재 다운로드, 최신 버전 설치까지 관리해 줌

### 기존 방식과 Maven방식의 차이

기존 방식	Maven 방식
<ul style="list-style-type: none"> <li>• 사용자가 외부 라이브러리를 위한 lib 폴더를 직접 생성</li> <li>• 사용자가 라이브러리 다운로드 후 Classpath 지정</li> </ul>	<ul style="list-style-type: none"> <li>• 선언만 하면 자동으로 생성</li> </ul>

기존 방식과 Maven 방식

# POM과 Plugin

## 1 POM

### 1》POM(Project Object Model)

- 1 POM은 프로젝트 자체와 의존성에 대한 설정 및 정보 포함
- 2 Maven은 pom.xml을 읽어 프로젝트를 가공하는 방법 이해
- 3 각각의 프로젝트는 pom.xml 파일을 하나씩 가짐

### 2》Maven의 자원 식별 형식

그룹 식별자(Group ID) : com.spring

산출물 식별자(Artifact ID) : projectname

버전(Version) : 001

# POM과 Plugin

## 1 POM

### 3 pom.xml의 구조

pom.xml의 구조	설명
<name>	어플리케이션의 명칭
<packaging>	프로젝트 산출물(Artifact) 패키징 유형 POM, jar, WAR, EAR, EJB, bundle, ... 중에서 선택
<parent>	프로젝트의 계층 정보
<dependencies>	의존성 정의 및 설정 영역
<dependency>	하나의 의존성 정의
<groupId>	일반적으로 프로젝트의 패키지 명칭
<artifactId>	산출물의 명칭(Artifact's name), groupId 범위 내에서 유일해야 함
<version>	산출물(Artifact)의 현재 버전

## 2 Plugin




### 1 Plugin

- 1 Maven은 플러그인 실행 프레임워크
- 2 Maven의 모든 작업은 플러그인이 수행
- 3 플러그인은 다른 산출물(Artifacts)과 같이 저장소에서 관리

# POM과 Plugin

## 2 Plugin

### 2 Goal

-  Maven은 여러 가지 플러그인으로 구성되어 있으며, 각각의 플러그인은 하나 이상의 Goal(명령, 작업)을 포함하고 있음
-  goal은 플러그인과 goal 명칭의 조합으로 실행할 수 있음
  - 1) 형식 : <plugin>:<goal>
  - 2) 예시 : mvn archetype:generate
-  Maven은 여러 Goal을 묶어 “라이프사이클 단계(Lifecycle Phases)”로 만들고 실행
  - 1) 형식 : mvn <phase name>
  - 2) 예시 : mvn install

### 3 Plugin 선언 예시(pom.xml)

```
<build>
  <plugins>
    <plugin>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>2.1</version>
      <configuration>
        <source>1.8</source>
        <target>1.8</target>
      </configuration>
    </plugin>
  </plugins>
</build>
```



# Lifecycle과 Profile

## 1 Lifecycle

일반적으로 소프트웨어 프로젝트는 컴파일, 테스트, 패키징, 인스톨과 같은 비슷한 전형적인 단계들을 가짐

이러한 단계는 Goal들로 구성됨

- 1) Goal은 특정 작업, 최소한의 실행 단위(Task)

Maven에서는 이러한 단계를 Build Life Cycle이라 부름

- 1) 빌드 단계(Build Phases)들은 사전 정의된 순서대로 실행
- 2) 모든 빌드 단계는 이전 단계가 성공적으로 실행되었을 때 실행

### Phase

- 단계(Phase)는 논리적인 개념
- 실질적인 작업을 수행하는 것은 각각의 단계에 연결(Associate)된 플러그인 Goal
- 패키지 타입(Package Type : jar, war 등)에 따라 각 단계에서 수행되는 Goal이 달라질 수 있음

## Phase와 Goals 비교

Phases	Goals
process-resources	resources:resources
Compile	compiler:compile
process-classes	
process-test-resources	resources:testResources
test-compile	compiler:testCompile
test	surefire:test
prepare-package	
package	jar:jar

# Lifecycle과 Profile

## 1 Lifecycle

### 표준 Lifecycle

1 clean : 빌드 시 생성되었던 산출물을 지움

- clean : 이전 빌드에서 생성된 모든 파일 삭제

2 default : 일반적인 빌드 프로세스를 위한 모델

- test : 적합한 단위 테스트 프레임워크를 이용해 테스트를 수행하고, 테스트 코드는 패키지 되거나 배포된 패키지 없이 실행될 수 있어야 함
- package : 컴파일 된 코드 등을 이용해 war 등의 패키지 파일 생성
- deploy : 다른 개발자 혹은 프로젝트와 공유할 수 있도록 원격 저장소에 최종 패키지를 배포

3 site : 프로젝트 문서와 사이트 작성 수행

- site : 프로젝트 사이트 문서 생성

# Lifecycle과 Profile

## 2 Profile

서로 다른 대상 환경(Target Environment)를 위한 다른 빌드 설정

- 1) 다른 운영체제
- 2) 다른 배포 환경

동작 방식(Activation)

- 1) (-P) 명령행 실행환경 옵션
- 2) 예 : mvn package -P prod

Maven은 정상 절차(Step) 이외에 프로파일을 위한 절차를 추가로 수행

Profile, 빌드 선언 예시

```
<profiles>
  <profile>
    <id>prod</id>
    <properties>
      <env>prod</env>
    </properties>
  </profile>
  <profile>
    <id>dev</id>
    <properties>
      <env>dev</env>
    </properties>
    <activation>
      <activeByDefault>true</activeByDefault>
    </activation>
  </profile>
</profiles>
```

# Lifecycle과 Profile

## 2 Profile

 Profile, 빌드 선언 예시

```
<build>
  <resources>
    <resource>
      <directory>src/main/resources/common</directory>
    </resource>
    <resource>
      <directory>src/main/resources/${env}</directory>
    </resource>
  </resources>
</build>
```

### 1. Maven 개념 파악하기

- Maven이란
  - Apache 프로젝트
  - 소스 코드로부터 배포 가능한 산출물 (Artifact)을 빌드(Build)하는 '빌드 툴(Build Tool)'
  - 편리한 '프로젝트 관리 툴'
- 의존성
  - 라이브러리 다운로드 자동화
  - 선언적인 Maven

### 2. POM과 Plugin

- POM
  - 프로젝트 자체와 의존성에 대한 설정 및 정보를 포함
  - Maven은 pom.xml 을 읽어 프로젝트를 가공하는 방법 이해
  - 각각의 프로젝트는 pom.xml 파일을 하나씩 가짐
- Plugin
  - Maven은 플러그인 실행 프레임워크임
  - Maven의 모든 작업은 플러그인이 수행
  - 플러그인은 다른 산출물(Artifacts)과 같이 저장소에서 관리

## 2. POM과 Plugin

- Lifecycle
  - 소프트웨어 프로젝트는 컴파일, 테스트, 패키징, 인스톨과 같은 비슷한 전형적인 단계들을 가짐
  - 이러한 단계는 Goal(특정 작업, 최소한의 실행 단위)들로 구성
  - Maven에서는 이러한 단계를 Build Life Cycle이라 부름
- Plugin
  - 서로 다른 대상 환경(Target Environment)를 위한 다른 빌드 설정
  - Maven은 정상 절차(Step) 이외에 프로파일을 위한 절차를 추가로 수행