Maven vs Gradle

> 필요한 라이브러리를 불러오고 빌드하는 life Cycle을 관리해주는 툴
>
> 레거시한 프로젝트는 Maven이지만,
> 최근 Gradle로 넘어오는 추세

빌드

: 소스코드 파일을 컴퓨터에서 실행할 수 있는 독립적인 형태로 변환하는 과정과 결과

- spring boot
  - 작성한 소스코드(.java) + 프로젝트에 사용한 파일 및 자원(.xml, .properties, .jpa, .jpg)을
    jvm이나 톰캣 같은 was가 인식할 수 있도록 패키징하는 과정 및 결과물
  - 빌드를 하면, 소스코드를 컴파일 해서 .class로 변환하고, resource를 .class가 참조할 수 있는 적절한 위치로 옮기고 META-INF와 MANIFEST.MF들을 하나로 압축하는 과정을 의미

빌드 관리 도구

: 소스코드에서 애플리케이션을 생성하면서 사용하는 필요한 라이브러리를 자동으로 관리해줌.

빌드 관리 도구가 수행하는 작업

- 종속성 다운로드 - **전처리(preprocessing)**
- 소스코드를 바이너리 코드로 **컴파일(Compile)**
- 바이너리 코드를 **패키징(Packaging)**
- **테스트** 실행(Testing)
- 프로덕션 시스템에 **배포(distribution)**

Maven

정의

: Java전용 프로젝트 관리 도구로, Lifecycle 관리 목적 빌드 도구이며, Apache Ant의 대안으로 만들어짐. Apache라이센스로 배포되는 오픈 소스 소프트웨어.

특징

- Lifecycle 관리 도구로, 정해진 Lifecycle에 의해 작업을 수행하며, 전반적인 프로젝트 관리 기능을 포함
- 필요한 라이브러리를 pom.xml에 정의 ⇒ 프로젝트 모델링
  - pom : Project Object Model
    - 프로젝트 정보(프로젝트 이름, 라이센스, ...)
    - 빌드 설정(소스, 리소스, Lifecycle별 실행한 plugin 등 빌드 관련 설정)
    - 빌드 환경(사용자 환경 별로 달라질 수 있는 프로필 정보)
    - pom 연관 정보(의존 프로젝트, 모듈, 상위 프로젝트 등)
  - pom.xml을 이용한 정형화된 빌드 시스템으로 다양한 라이브러리를 관리하며 네트워크를 통해 자동으로 다운을 받아줌.
- 기존 Apache Ant (Java로 개발된 어플리케이션으로 커맨드 라인 형태의 빌드, 배포를 위한 도구)라는 빌드 도구를 많이 사용했지만 Maven이 Ant를 넘어서 개발자들이 많이 사용)

Lifecycle 순서

1. clean : 빌드 시 생성되어있었던 파일들을 삭제
2. validate : 프로젝트가 올바른지 확인하고 필요한 모든 정보를 사용할 수 있는지 확인하는 단계
3. compile : 프로젝트 소스코드를 컴파일 하는 단계
4. test : 단위 테스트를 수행하는 단계. 테스트 실패 시 빌드 실패로 처리하며, 스킵이 가능하다.
5. package : 실제 컴파일된 소스 코드와 리소스들을 jar, war등의 파일을 배포를 위한 패키지로 만든다.
6. verify : 통합 테스트 결과에 대한 검사를 실행하여 품질 기준을 충족하는지 확인한다.
7. site : 프로젝트 문서와 사이트 작성, 생성하는 단계
8. deploy : 만들어진 package를 원격 저장소에 release하는 단계

Apache Ant vs Maven

Apache Ant는 비교적 자유도가 높은 편이고,
Maven은 정해진 Lifecycle에 의해 작업을 수행하며, 전반적인 프로젝트 관리 기능까지 포함하는 차이점이 있다.

Gradle

정의

: Maven을 대체할 수 있는 프로젝트 구성 관리 및 범용 빌드 툴

- Ant Builder와 Groovy script를 기반으로 구축되어 기존 Ant의 역할과 배포스크립트의 기능을 모두 사용가능
- 스프링 부트와 안드로이드에서도 사용
- 빌드 속도가 Maven에 비해 10~100배 가량 빠르다
- Java, C/C++, Python 등을 지원

Groovy

: JVM에서 실행되는 스크립트 언어

- JVM에서 동작하지만 소스코드를 컴파일할 필요 없다.
- Java와 호환되며, Java class file들을 Groovy class로 사용 가능하다.
- Java 문법과 유사하여 빌드 처리를 관리할 수 있다.

특징

- 가독성이 좋다 : 코딩에 의한 간결한 정의가 가능
- 재사용에 용이 : 설정 주입 방식(Configuration Injection)을 사용하므로 재사용에 용이
- 구조적인 장점 : Build Script를 Groovy기반 DSL(Domail Specific Language)를 사용해 코드로서 설정 정보를 구성하므로 구조적인 장점이 있다.
- 편리함 : Gradle 설치 없이 Gradle wrapper를 이용해 빌드 지원
- 멀티 프로젝트 : Gradle은 멀티 프로젝트 빌드를 지원하기 위해 설계된 빌드 관리 도구
- 지원 : Maven을 완전 지원

Maven vs Gradle

1. 스크립트 길이와 가독성 측면에서 Gradle이 우세
2. 빌드와 테스트 실행 결과 Gradle이 더 빠름
    a. gradle은 캐시를 사용하므로 테스트 반복 시 실행 결과 시간의 차이가 더 커진다.
3. 의존성이 늘어날 수록 스크립트 품질의 차이가 커진다.
    a. Maven은 멀티 프로젝트에서 특정 설정을 다른 모듈에서 사용하려면 상속을 받아야 한다.
    b. Gradle은 설정 주입 방식을 사용해 멀티 프로젝트에 적합하다.