Build

소스 코드 파일을 컴퓨터나 휴대폰에서 실행할 수 있는 독립 소프트웨어 가공물로 변환하는 과정이나 그에 대한 결과물.

Build Tool

쉽게 말해 빌드를 도와주는 도구.

종속성 다운로드(라이브러리를 자동 추가 및 관리), 컴파일(소스 코드 > 바이너리 코드), 바이너리 코드 패키징, 테스트 실행, 배포.

우선 필요한 라이브러리들이 많아졌다. 많은 라이브러리들을 직접 다운받고 추가가 가능하지만 번거로움과 버전관리, jar파일의 보안 위험 등의 문제가 발생하기 때문에 사용한다.

Make

초기 빌드 도구. C언어로 작성함 > 자바 빌드 도구로 사용하기 어려움이 있음 > Apache Ant를 많이 사용하게 된다.

Apache Ant

Make와 비슷하며 간단하다. XML 기반의 빌드 스크립트를 사용. 공식적인 규약을 가지고 있지 않기 때문에 유연성이 장점이다. 유연성이 좋은 만큼 정확한 코딩이 필요해 프로젝트 구조가 복잡해질수록 빌드 과정의 이해가 어려워진다 > 단점을 보완하기위해 Maven을 만듬.

Maven텍스트이(가) 표시된 사진

자동 생성된 설명

XML파일을 계속 사용. 공식적인 규약 추가> 표준화> 유지보수 가능. 그러나 유연성이 사라짐. 필요한 라이브러리를 자동으로 불러오고 관리하는 기능 추가. Pom.xml 파일(정적인 문서)에 작성하면 된다.

Gradle

Ant의 장점, Maven의 기능을 합침. XML파일을 사용하지 않고, JVM 위에서 동작하는 Groovy나 Kotlin을 통해 작성. 동적인 스크립트 설정 파일 작성으로 Ant의 유연함을 가져옴. Build.gradle(동적인 소스 파일) 로직을 넣을 수 있어 한계가 없다. 가독성이 좋아졌다> 유지보수가 좋아졌다. Configuration Injection(설정 주입) 방식을 사용해 필요한 모듈에만 주입할 수 있다. 성능이 Maven보다 두 배 이상 빠르다. Incremental Builds(빌드 실행 중 마지막 빌드 호출 이후 변경이 됐는지 확인> 변경이 없으면 빌드 실행 안함), Build Cache(다른 컴퓨터에서 실행된 경우 로컬 실행을 건너뛰고 빌드 캐시로부터 가져옴), Gradle Demon(빌드하는 인스턴스를 유지해 빌드가 끝나도 사라지지 않고 백그라운드에서 대기해 초기화하는 시간을 단축).

<https://gradle.org/>

<https://docs.gradle.org/current/userguide/userguide.html>

<https://www.slipp.net/wiki/display/IDE/Gradle>

<https://tecoble.techcourse.co.kr/post/2020-09-17-java-build-tool/>

<https://www.youtube.com/watch?v=ntOH2bWLWQs>

<https://jaeho214.tistory.com/27>