Ministry of SMEs and Startups × KOSME Korea SMEs and Startups Agency × LIKELION

Gradle, Maven, Ant

Nguyễn Anh Tuấn

K TECH COLLEGE

Nội dung bài giảng

1 Lịch sử của các công cụ phát triển

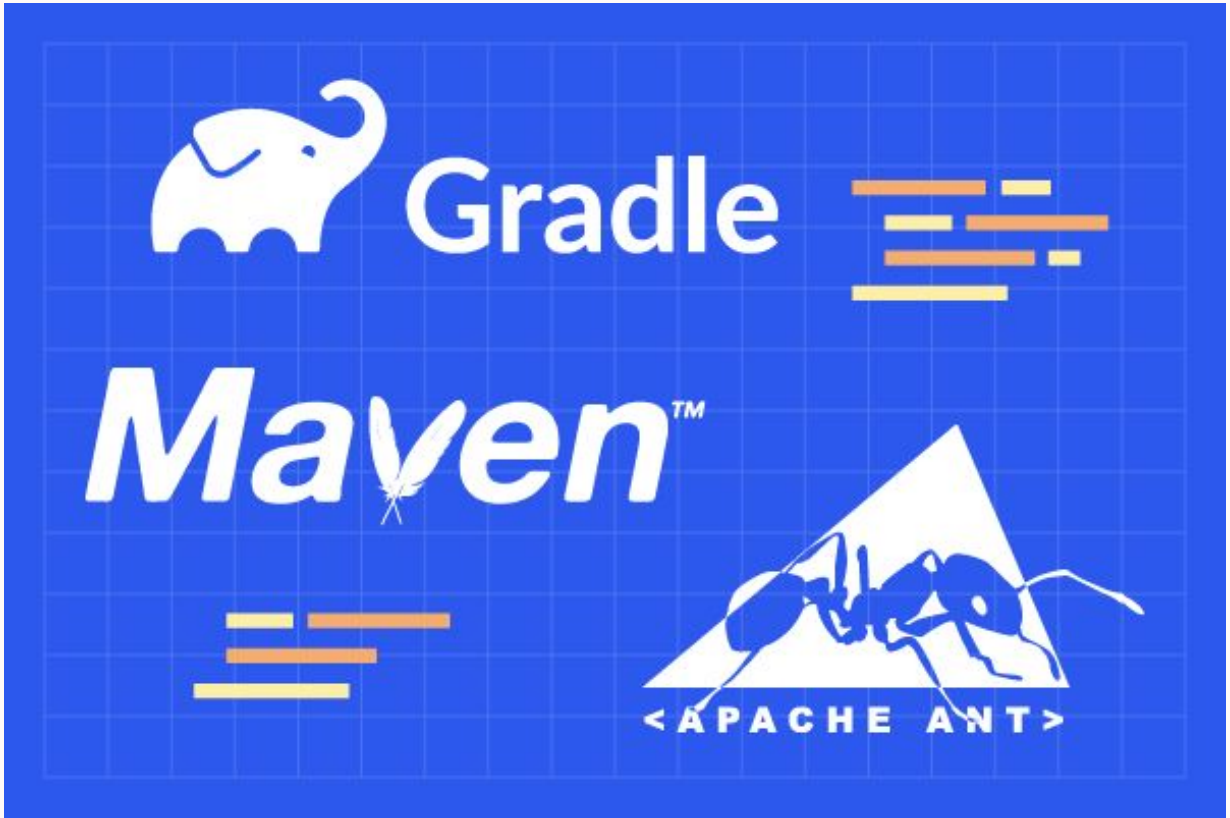
2 Công cụ Gradle

3 Công cụ Apache Maven

4 Công cụ Apache Ant

Lịch sử của các công cụ

Lịch sử ra đời của Ant, Maven, Gradle



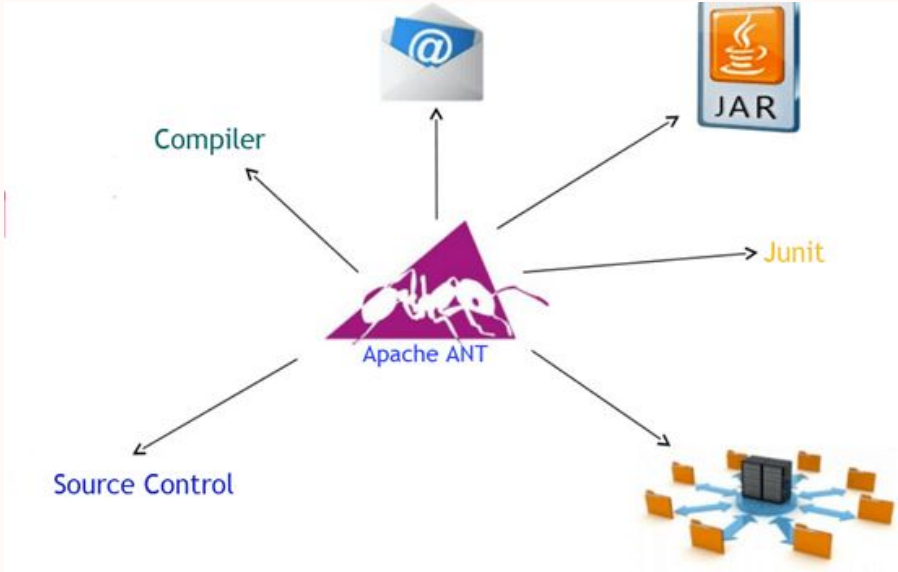
Lịch sử ra đời của Ant, Maven, Gradle

- **Apache Ant** ra mắt năm 2000, cho phép mô tả meta-data của dự án cũng như các hoạt động trong quá trình build thông qua bộ cú pháp XML. Tuy nhiên tiêu chuẩn code của Ant quá dài và phức tạp.
- **Apache Maven** ra mắt vào năm 2004, đưa ra một cấu trúc dự án tiêu chuẩn.
  - Maven cho phép sử dụng các plugin thay vì phải code toàn bộ quá trình đóng gói.
  - Maven còn cho phép tải về các thư viện thông qua internet, giúp cho việc chia sẻ cũng như quản lý phiên bản trở nên dễ dàng hơn.
  - Maven sử dụng XML làm cho file POM (Project Object Model) trở nên dài dòng và phức tạp.
- **Gradle** là bộ công cụ mã nguồn mở do cộng đồng phát triển có nguồn gốc từ Apache, nhằm kết hợp ưu điểm của cả Ant và Maven, đồng thời cải thiện tốc độ cũng như tính linh hoạt so với Ant và Maven.

Apache Ant

Công cụ Apache Ant

- **Mô tả**:
  - **Ant** là một công cụ xây dựng phổ biến trước Maven và Gradle, chủ yếu được sử dụng để tự động hóa các tác vụ xây dựng dự án.
  - **Ant** sử dụng các file cấu hình XML (build.xml) để định nghĩa các tác vụ xây dựng



Công cụ Apache Ant

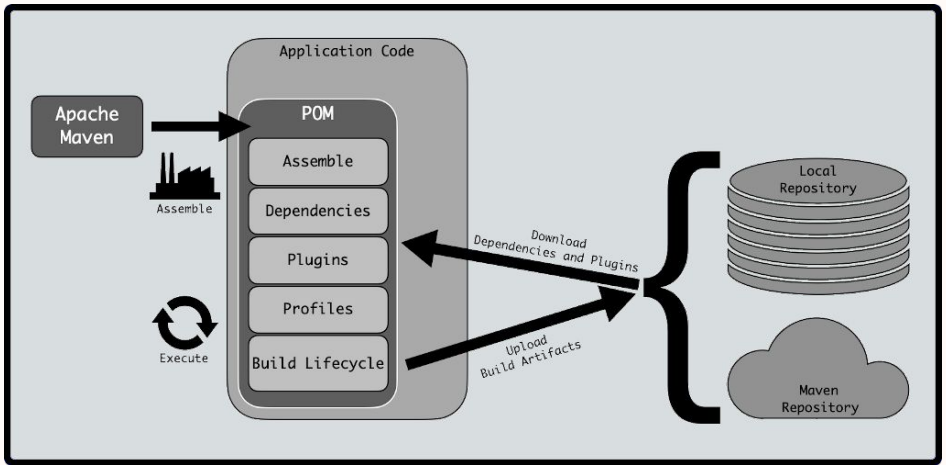
- **Ưu điểm:**
  - Rất linh hoạt, có thể tùy biến mọi tác vụ.
  - Đơn giản và dễ hiểu với các tác vụ cơ bản.

- **Nhược điểm:**
  - Không có hệ thống quản lý phụ thuộc tích hợp.
  - Cấu hình thủ công, không chuẩn hóa cấu trúc dự án.
  - Cú pháp XML phức tạp và dài dòng.

Apache Maven

Công cụ Apache Maven

- **Mô tả:**
  - **Maven** là công cụ quản lý và thiết lập tự động 1 dự án. Chủ yếu dùng cho ngôn ngữ java, nhưng nó cũng có thể được dùng để xây dựng và quản lý các dự án dùng C#, Ruby, Scala hay ngôn ngữ khác.
  - **Maven** dùng khái niệm Project Object Model (POM) để mô tả việc build project, các thành phần phụ thuộc và các module.
  - **Maven** tải các thư viện, plug-in từ 1 hoặc nhiều repositories.



Công cụ Apache Maven

- **Các tiêu chí của Maven:**
  - Tạo ra một quy trình build dễ dàng.
  - Cung cấp một hệ thống build chuẩn.
  - Cung cấp thông tin project chất lượng.
  - Cung cấp hướng dẫn tốt nhất cho việc thực hiện phát triển.
  - Cho phép chuyển đổi tới những tính năng mới.

Công cụ Apache Maven

- **Ưu điểm:**
  - Quản lý phụ thuộc dễ dàng.
  - Chuẩn hóa cấu trúc dự án.
  - Hỗ trợ hệ sinh thái plugin rộng lớn.
  - Tích hợp tốt với nhiều công cụ CI/CD.

- **Nhược điểm:**
  - Định dạng XML có thể phức tạp và khó đọc.
  - Cấu hình phức tạp đối với dự án lớn.
  - Khả năng tùy biến hạn chế so với Gradle.

Công cụ Apache Maven

- **Cài đặt Apache Maven trên Windows:**
  - Link tải file cài đặt: http://maven.apache.org/download.cgi
  - Cấu hình môi trường trên Windows:
    - Vào **Environment Variables**
    - Trong **System variables** chọn New
      - Variable name: **M2_HOME**
      - Variable value: Đường dẫn đến thư mục Maven, ví dụ **C:\apache-maven**
    - Trong **System variables** tìm Path
      - Thêm **%M2_HOME%\bin**
  - Kiểm tra phiên bản: "mvn -version"

Công cụ Apache Maven

- **Cài đặt Apache Maven trên Mac:**
  - Cài đặt HomeBrew thông qua terminal:
    - /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
  - Cài đặt Maven
    - brew install maven
  - Kiểm tra phiên bản: "mvn -version"

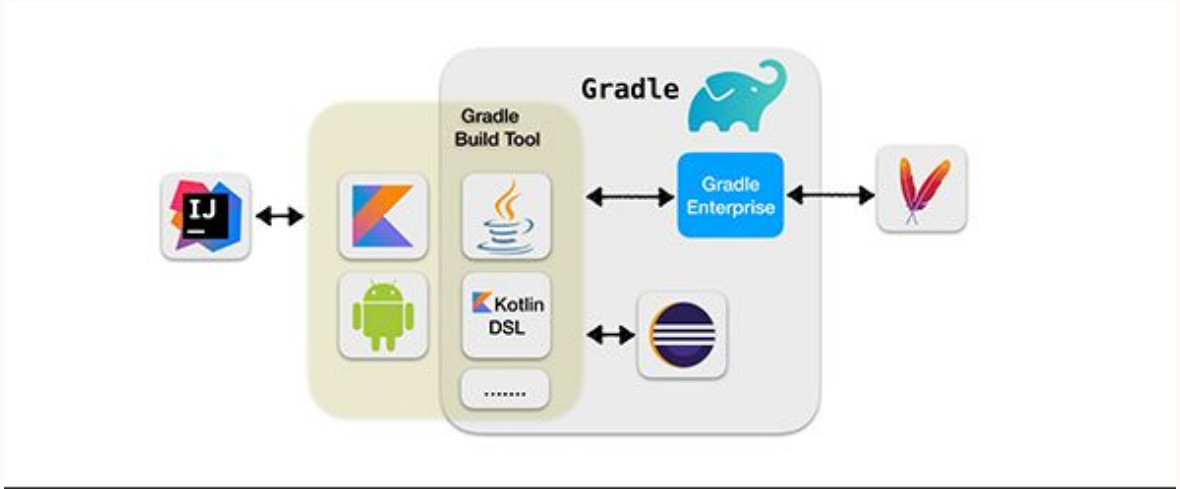
Công cụ Apache Maven

- **Một số câu lệnh thường dùng:**
  - **Maven** sử dụng các mục tiêu **<goal>** để thực thi các công việc cụ thể. Các mục tiêu này có thể được thực thi bằng cách sử dụng cú pháp **mvn <goal>**
    - **clean**: Xóa các tệp tạm thời và thư mục đang tạo bởi Maven.
    - **compile**: Biên dịch mã nguồn của dự án.
    - **mvn exec:java:** Biên dịch và thực thi ứng dụng Java của bạn từ Maven.
    - **test**: Chạy các bài kiểm tra (unit tests) của dự án.
    - **package**: Đóng gói mã nguồn thành một tệp JAR/WAR.
    - **install**: Đưa các artifact vào local repository để có thể sử dụng chung cho các dự án khác.
    - **deploy**: Sao lưu các artifact lên remote repository.

Gradle

Công cụ Gradle

- **Mô tả:**
  - **Gradle** kết hợp các ưu điểm của **Ant** và **Maven** đồng thời đem lại các cải thiện mới thông qua việc sử dụng DSL (Domain Specific Language) dựa trên Groovy hoặc Kotlin để định nghĩa cấu hình của dự án và các quy trình xây dựng.
  - **Gradle** hỗ trợ build cho nhiều ngôn ngữ lập trình khác nhau bao gồm Java, Scala, Python, C/C++, Android, iOS.



Công cụ Gradle

- **Gradle hỗ trợ hai loại DSL chính:**
  - **Groovy DSL (build.gradle):** Groovy DSL là ngôn ngữ đặc thù được sử dụng mặc định trong Gradle, cho phép bạn định nghĩa các tác vụ, dependencies, plugin và các cấu hình khác trong file build.gradle.

```
// File build.gradle
plugins {  id 'java' }

repositories { mavenCentral() }

dependencies { implementation 'com.google.guava:guava:30.1-jre' }

task runApp(type: JavaExec) {
        main = 'com.example.MainClass'
        // Định nghĩa lớp chính của ứng dụng
        classpath = sourceSets.main.runtimeClasspath
}
```

Công cụ Gradle

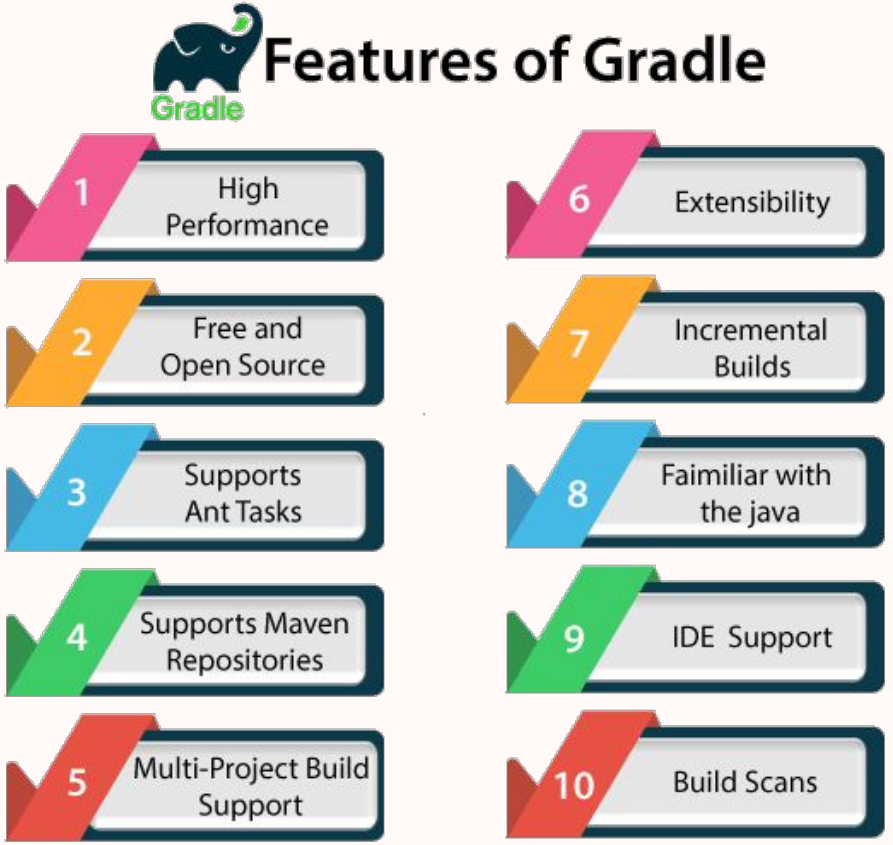
- **Gradle hỗ trợ hai loại DSL chính:**
  - **Kotlin DSL (build.gradle.kts):** Kotlin DSL là một lựa chọn thay thế cho Groovy DSL, cho phép sử dụng ngôn ngữ lập trình Kotlin để định nghĩa các cấu hình dự án và các task.

```
// File build.gradle.kts
plugins { id("java") }

repositories { mavenCentral() }

dependencies { implementation("com.google.guava:guava:30.1-jre") }

tasks.register<JavaExec>("runApp") {
        main = "com.example.MainClass"
        // Định nghĩa lớp chính của ứng dụng
        classpath = sourceSets.getByName("main").runtimeClasspath
}
```

Công cụ Gradle

- **Ưu điểm:**
  - DSL dễ đọc và dễ viết.
  - Khả năng tùy biến cao.
  - Hiệu suất xây dựng nhanh hơn nhờ cơ chế caching và build incremental.
  - Tích hợp dễ dàng với các công cụ và nền tảng khác.

- **Nhược điểm:**
  - Cú pháp mới có thể khó khăn cho người mới bắt đầu.
  - Hệ sinh thái plugin còn hạn chế so với Maven.
  - Cần thêm thời gian để làm quen và học cách cấu hình.



Công cụ Gradle

- **Một số câu lệnh thường dùng:**
  - **Gradle** sử dụng các mục tiêu **<goal>** để thực thi các công việc cụ thể. Các mục tiêu này có thể được thực thi bằng cách sử dụng cú pháp **gradle <goal>**
    - **clean**: Xóa các file và thư mục được tạo ra trong quá trình xây dựng.
    - **build**: Thực hiện toàn bộ quy trình biên dịch và đóng gói của dự án.
    - **tasks**: Hiển thị danh sách các task có sẵn để thực thi trong dự án.
    - **<taskName>**: Thực thi 1 task <taskName> cụ thể.
    - **compileJava**: Biên dịch mã nguồn của dự án.
    - **test**: Chạy các bài kiểm tra (unit tests) của dự án.
    - **jar**: Đóng gói mã nguồn thành một tệp JAR/WAR.
    - **dependencyUpdates**: Kiểm tra và hiển thị các cập nhật mới nhất cho các phụ thuộc trong dự án.
    - **javadoc**: Tạo tài liệu Javadoc cho dự án.

So sánh giữa 3 công cụ

So sánh giữa 3 công cụ phát triển

Nội dung	Ant	Maven	Gradle
Cấu hình	Dựa trên XML, không chuẩn hóa, yêu cầu cấu hình thủ công.	Dựa trên XML, có cấu trúc chuẩn hóa.	Dựa trên Groovy hoặc Kotlin DSL, dễ đọc và viết hơn.
Hiệu suất	Tốc độ phụ thuộc vào cấu hình và kịch bản xây dựng.	Tốc độ xây dựng nhanh hơn Ant, chậm hơn Gradle.	Tốc độ xây dựng nhanh hơn nhờ caching và incremental build.
Linh hoạt	Rất linh hoạt nhưng yêu cầu cấu hình phức tạp.	Cấu hình đơn giản nhưng hạn chế hơn so với Gradle.	Linh hoạt và tùy biến cao.
Lỗi complie		Phát hiện lỗi sẽ dừng quá trình đóng gói	Tổng hợp toàn bộ các lỗi gặp phải trong một lần đóng gói

ROAD TO KOREA

Nếu có bất kỳ thắc mắc nào, hãy đặt câu hỏi qua

mail@mail.com hoặc Zalo 0xxx xxx xxx