# **TÌM HIỂU VỀ CÁC CÔNG CỤ QUẢN LÝ CẤU HÌNH**

# **(GRADLE)**

1. **Giới thiệu:**
   1. **Lịch sử hình thành:**

Trong quá trình phát triển phần mềm, việc tự động hóa quá trình build và quản lý các thư viện rất quan trọng, nhằm tối ưu hóa công đoạn biên dịch, đóng gói; thuận tiện hơn cho việc phát triển phần mềm, cũng như maven và gradle cũng là một trong những công cụ đó.

* Năm 2000, Apache Ant ra đời như là một trong những tool đóng gói hiện đại và nhanh chóng được sử dụng trong các dự án sử dụng ngôn ngữ Java. Ant cho phép nhà phát triển mô tả meta-data của dự án cũng như các hoạt động trong quá trình build thông qua bộ cú pháp XML. Tuy nhiên những đoạn mã viết theo tiêu chuẩn của Ant quá dài và phức tạp.
* Năm 2004, Apache Maven ra đời và đưa ra các cải thiện so với Ant. Bằng việc đưa ra một cấu trúc dự án tiêu chuẩn, Maven cho phép nhà phát triển sử dụng một trong các plugin của mình thay vì phải viết toàn bộ các dòng code mô tả quá trình đóng gói. Ngoài ra Maven còn cho phép tải về các thư viện phụ thuộc thông qua internet, điều này giúp cho việc chia sẻ cũng như quản lý phiên bản trở nên dễ dàng hơn. Đổi lại, nhà phát triển sẽ mất khá nhiều công sức để tùy chỉnh nếu cấu trúc dự án hiện tại không tuân theo Maven Standard Directory Layout. Và việc vẫn sử dụng XML làm cho file POM (Project Object Model) trở nên dài dòng và phức tạp để nắm bắt logic.
* Gradle kết hợp các ưu điểm của Ant và Maven đồng thời thêm vào các cải thiện mới và sử dụng Groovy trong việc giải quyết bài toán đóng gói.
  1. **Các ưu điểm của Grandle:**

Gradle kết hợp các ưu điểm của Ant và Maven đồng thời đem lại các cải thiện mới thông qua việc sử dụng Groovy (một trong những ngôn ngữ JVM) trong việc giải quyết bài toán đóng gói. Nhờ vậy build script viết bằng Gradle trở nên ngắn gọn và trực quan dễ đọc hơn.

Ngoài việc sử dụng Groovy làm DSL, Gradle cũng chú ý hỗ trợ thêm snhiều tính năng khác để cải thiện quá trình đóng gói cũng như tăng năng suất cho nhà phát triển. Đối với các dự án lớn, việc có nhiều module bên trong phụ thuộc lẫn nhau là hết sức phổ biến. Khi sử dụng Maven, quá trình build diễn ra lâu do quá trình clean toàn bộ các artifact và sau đó là quá trình đóng gói lại từ đầu, đồng thời thời gian chạy các đoạn Unit Test cũng tiêu tốn khá nhiều thời gian. Đối với Gradle, do một task build (một đơn vị logic của một quá trình build) đều được định nghĩa input và output, nên khi có sự thay đổi nhỏ trong source code, Gradle sẽ chỉ thực hiện build lại một phần cần thiết. Đặc biệt, Gradle sẽ tự tìm kiếm các task khác lệ thuộc task này và thực hiện quá trình build lại. Như vậy, tốc độ build của Gradle sẽ được rút ngắn. Hơn thế nữa, Gradle cho phép chạy parallel các đoạn code test nhờ đó rút ngắn thời gian của quá trình build. Trong tương lai, Gradle cho phép thực hiện việc phân phối các đoạn code test qua nhiều machine khác nhau để giảm thiểu thời gian chờ đợi. Đồng thời điểm khác biệt lớn so với Maven là thay vì dừng quá trình build ngay khi gặp lỗi, Gradle sẽ thực thi hết tất cả các task có thể và tổng hợp toàn bộ các lỗi gặp phải trong một lần đóng gói. Nhà phát triển sẽ không phải chờ đợi quá lâu để build và gặp lỗi, sửa lỗi và build lại.

* + 1. **Build cho nhiều ngôn ngữ:**

Gradle hỗ trợ build cho nhiều ngôn ngữ lập trình khác nhau bao gồm Java, Scala, Python, C/C++, Android, iOS.

Được thiết kế để chạy trên JVM, Gradle hỗ trợ mạnh mẽ cho các dự án viết bằng Java và các ngôn ngữ JVM khác như Scala, Groovy,... với việc hỗ trợ nhiều loại ứng dụng và công cụ phát triển thiết yếu như test frameworks (JUnit, TestNG, Spock), application frameworks (Java EE, Grails, Play!), embedded web servers (Jetty) and code analysis cops (Checkstyle, CodeNarc, FindBugs, JDepend, PMD, JaCoCo, Sonar).

Tích hợp với Android Studio hỗ trợ quản lý dependency, multi-project. Đồng thời tích hợp các test services phổ biến như AppThwack, TestDroid và Manymo. Ngoài ra, có thể tạo nhiều APK từ 1 module và upload chúng lên Google Play và các app store khác với các phiên bản khác nhau (free, paid,...)

Hỗ trợ compiler, assembler và linker; cho phép phân phối các thư viện, file thực thi với nhiều loại (debug, release), platforms (x86, x86\_64), phiên bản (community, enterprise) khác nhau. Tích hợp GCC, CLANG, XCODE, VISUAL C++

* + 1. **Quản lý dependency mạnh mẽ:**

Có thể nhận từ bất kỳ repository nào được biết đến : từ public hay private đến local; pha trộn và kết hợp các loại repository, Maven repositories, public (Maven Central, jCenter), third-party and in-house (Bintray, Artifactory), local Maven repository, Ivy repositories, Custom & legacy repositories

* + 1. **Giải pháp dependency tối ưu:**
* Không bao giờ tải 1 artifact nhiều lần
* Cache
* Xác minh qua checksum
* Track nguồn gốc dependency đảm bảo tính hợp lệ và ngăn chặn lỗi ...
  + 1. **So sánh với Maven:**
       - **Maven:**
* Quá trình build diễn ra lâu do quá trình clean toàn bộ các artifact và sau đó là quá trình đóng gói lại từ đầu.
* Các đoạn Unit Test cũng tiêu tốn khá nhiều thời gian.
* Dừng quá trình build ngay khi gặp lỗi
* **Grandle:**
* Một task build đều được định nghĩa input và output, nên khi có sự thay đổi nhỏ trong source code, Gradle sẽ chỉ thực hiện build lại một phần cần thiết
* Gradle sẽ tự tìm kiếm các task khác lệ thuộc task này và thực hiện quá trình build lại
* Gradle cho phép chạy parallel các đoạn code test nhờ đó rút ngắn thời gian của quá trình build
* Ngoài ra Gradle cho phép thực hiện việc phân phối các đoạn code test qua nhiều machine khác nhau để giảm thiểu thời gian chờ đợi.
* Thực thi hết tất cả các task có thể.
* Tổng hợp toàn bộ các lỗi gặp phải trong một lần đóng gói.