

MAVEN VS GRADLE UNAQUEL BATTLE

# Giới Thiệu về Gradle và Maven

Trong quá trình phát triển phần mềm hoặc tự động hóa, 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 mềmGradle va Maven là hai công cụ hổ trợ giải quyết các vấn đề đó.

Có một số khác biệt cơ bản trong cách xây dựng hai hệ thống. Gradle dựa trên một biểu đồ nhiệm vụ - trong đó các nhiệm vụ là những thứ thực hiện công việc - trong khi Maven dựa trên mô hình cố định mục tiêu và tuyến tính. Với Maven, các mục tiêu được gắn liền với các giai đoạn của dự án và các mục tiêu phục vụ một chức năng tương tự như các nhiệm vụ của Gradle.

Từ đó ta có những khác biệt chính giữa Gradle và Maven: Tính linh hoạt, hiệu suất, trải nghiệm người dùng và quản lý dependency.

# So sánh Gradle và Maven

## Tính linh hoạt

Gradle sử dụng Groovy (một trong những ngôn ngữ JVM). Nhờ vậy build script viết bằng Gradle trở nên ngắn gọn và trực quan dễ đọc hơn. Mô hình của Gradle cũng cho phép nó [sử dụng được C / C ++ để phát triển](https://github.com/gradle/gradle-native)  và có thể được mở rộng để bao quát bất kỳ hệ ngôn ngữ lập trình nào  Java, Scala, Python, C/C++, Android, iOS.

. Cả Gradle và Maven đều cung cấp quy ước về cấu hình. Tuy nhiên, Maven cung cấp một mô hình rất cứng nhắc khiến việc tùy biến trở nên tẻ nhạt và đôi khi không thể. Gradle, được xây dựng với khả năng tùy biến cấu trúc bản dựng dựa trên trí tưởng tượng của mỗi người bằng cách trao quyền và có trách nhiệm cho người dùng.

## Hiệu suất

***“Gradle nhanh hơn ít nhất gấp đôi cho gần như mọi kịch bản (nhanh hơn 100 lần [nà ní] đối với các bản dựng lớn sử dụng bộ đệm xây dựng)”***

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.

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.

Cả Gradle va Maven đều cho phép các bản dựng đa mô-đun chạy song song (Maven gần đây đã nâng cấp tính năng này) và giải quyết phụ thuộc song song. Sự khác biệt lớn nhất là các cơ chế né công việc của Gradle và cơ chế vận hành thông minh. Từ đó tổng kết được 3 tính năng hàng đầu giúp Gradle nhanh hơn nhiều so với Maven là:

[Incrementality](https://blog.gradle.org/introducing-incremental-build-support): Gradle tránh né các kiểu con đà điểu =)) công việc bằng cách theo dõi đầu vào và đầu ra của các tác vụ và chỉ chạy những gì cần thiết.

[Build Cache](https://blog.gradle.org/introducing-gradle-build-cache): Sử dụng lại các đầu ra xây dựng của bất kỳ bản dựng Gradle nào khác có cùng các đầu vào, bao gồm giữa các máy.

[Gradle Daemon](https://docs.gradle.org/current/userguide/gradle_daemon.html):  Một quá trình tồn tại lâu dài giúp xây dựng thông tin "nóng hổi vừa thổi vừa ăn" trong bộ nhớ (hiểu môm na là chạy tác vụ trong nền không cần mở các chương trình).

Một điểm sánh như ánh hào quang ~~ khác của Gradle 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.

## Trải nghiệm người dùng

Gradle được google sử dụng (COCC) =)))))

Một số lượng lớn người dùng thích thực hiện các hoạt động xây dựng thông qua giao diện dòng lệnh. Gradle cung cấp một CLI hiện đại có các tính năng như ‘gradle tasks’, cũng như ghi nhật ký logging  và [hoàn thành dòng lệnh](https://github.com/gradle/gradle-completion).

## Quản lý dependency mạnh mẽ

Khi nói đến việc quản lý các dependency, cả Gradle và Maven đều có thể xử lý các dependency: local, public or private. Sử dụng các dependency bắc cầu của bên thứ ba và để đọc định dạng siêu dữ liệu POM. Tuy nhiên, Gradle chiến thắng khi nói đến API và cách triển khai dependency. cũng như cho phép lưu trữ an toàn đồng thời các dependency, siêu dữ liệu của kho lưu trữ cùng với các phụ thuộc được lưu trong bộ nhớ cache, đảm bảo rằng hai hoặc nhiều dự án sử dụng cùng một bộ đệm sẽ không ghi đè lên nhau và nó có bộ đệm dựa trên tổng kiểm tra và có thể đồng bộ hóa bộ đệm với kho lưu trữ. (Điểm nhấn Gradle có thể cấu sờ hình dependency sử dụng cho mục sờ đíc của người sử dụng).

Maven cho phép một người ghi đè một dependency dựa theo phiên bản. Mới hơn (mập hơn) đè bẹp thằng củ hơn (ốm yếu). Gradle tương thích với Siêu dữ liệu IVY, cho phép bạn xác định các quy tắc tùy chỉnh để chỉ định một phiên bản cho sự phụ thuộc động và giải quyết xung đột phiên bản. Đây không phải là có sẵn trên Maven.

# Túm cái quần lại là:

Với Maven, bạn có thể dễ dàng xác định siêu dữ liệu và dependence của dự án, Maven cung cấp [quản lý phụ thuộc](https://stackify.com/application-dependency-mapping-performance/) đơn giản nhưng hiệu quả và vì nó có cấu trúc thư mục cho các dự án của bạn, bạn có một số cách bố trí tiêu chuẩn cho tất cả các dự án của bạn. Nó sử dụng tệp XML khai báo cho tệp POM của nó và có một loạt các plugin mà bạn có thể sử dụng. Nhưng việc tạo một bản dựng tùy biến cao có thể là cơn ác mộng đối với người dùng Maven. Cũng như tính linh hoạt và hiệu suất làm việc của Maven là tồi tệ.

# ------------------------------------------------------------------------------------🡪 Gradle