**Koin**

[**https://www.raywenderlich.com/9457-dependency-injection-with-koin**](https://www.raywenderlich.com/9457-dependency-injection-with-koin)

1. Dependency Injection là gì ?

**Dependency injection (DI)** là một kỹ thuật lập trình giúp tách một class độc lập với các biến phụ thuộc. Với lập trình hướng đối tượng, chúng ta hầu như luôn phải làm việc với rất nhiều class trong một chương trình. Các class được liên kết với nhau theo một mối quan hệ nào đó. **Dependency** là một loại quan hệ giữa 2 class mà trong đó một class hoạt động độc lập và class còn lại phụ thuộc bởi class kia.

Hiểu một cách đơn giản Giả sử chúng ta có 2 Class là Class A và Class B, khi chúng ta sử dụng Class A tham chiếu với Class B với việc sử dụng các method của Class B, lúc này sinh ra quan hệ dependency giữa A và B. Để Class A có thể gọi method của B thì trước tiên nó phải tạo một instance của Class B. Vậy ta có thể hiểu, việc chuyển giao nhiệm vụ khởi tạo object đó cho một ai khác và trực tiếp sử dụng các **dependency** đó được gọi là **dependency injection**.

Trong hướng dẫn này, bạn sẽ làm quen với [Koin](https://insert-koin.io/) , một trong những framework mới phổ biến nhất cho DI. Bạn sẽ bắt đầu bằng cách tìm hiểu những kiến ​​thức cơ bản về DI và cách các dự án Android của bạn có thể tận dụng nó. Sau đó, bạn sẽ áp dụng nó cho một ứng dụng mẫu sử dụng Koin làm khung DI và minh họa các lợi ích của nó.

1. Koin là gì

Được ra mắt vào năm 2018 Koin là Dependency injection framework mạnh mẽ dành cho Kotlin. Koin là một lightweight dependency injection framework được viết bằng Kotli thuần túy sử dụng functional nên không proxy, không sinh code, không ánh xạ.

Vì vậy, tại sao lại sử dụng Koin thay vì một trong các khung DI khác? Câu trả lời: Koin ngắn gọn và thẳng thắn hơn những người khác.

Lấy Dagger 2 phổ biến làm ví dụ. Để sử dụng Dagger 2 , trước tiên bạn cần làm quen với các khái niệm như module và component và các annotations như @Inject . Mặc dù đường cong học tập dốc cho khung công tác này cuối cùng cũng thành công, nhưng để tận dụng tối đa nó, bạn vẫn phải học một số khái niệm nâng cao như phạm vi và các thành phần con.

Ngược lại, Koin cho phép bạn chỉ cần khai báo các mô-đun, bao gồm các phụ thuộc tiềm năng, sẽ được sử dụng trong dự án và trực tiếp đưa chúng vào lớp quan tâm.

#### Hiểu các thuật ngữ trong Koin

Trong khi làm việc với Koin, chúng tôi cần hiểu một số thuật ngữ trước khi bắt đầu.

* **module**  - nó tạo ra một module trong Koin sẽ được Koin sử dụng để cung cấp tất cả các phụ thuộc.
* **single** - nó tạo ra một singleton có thể được sử dụng trên toàn ứng dụng như một instance.
* **factory** - nó cung sẽ cung cấp một đối tượng mới cho mỗi lần chúng ta yêu cầu một thể hiện.
* **get ()** - nó được sử dụng trong hàm tạo của một lớp để cung cấp phụ thuộc bắt buộc.

1. Demo
2. Kết luận

Một hạn chế khi sử dụng Koin là bạn chỉ có thể đưa các phụ thuộc vào Activity các lớp ra khỏi hộp. Để đưa các phụ thuộc vào các kiểu lớp khác, bạn phải thực hiện việc đó thông qua các hàm tạo tương ứng. Để giải quyết vấn đề này, Koin cho phép các lớp tuân theo KoinComponent interface, do đó có thể tiêm vào các Activity lớp không phải lớp. Bạn sẽ thấy một ví dụ về điều này sau này.