[HDDT-Intern] Tìm hiểu khái niệm DI, Service – Mai Trung Tiến

Dependency inversion principle

Service object design pattern

# Tìm hiểu khái niệm DI, Service

## Tìm hiểu khái niệm Dependency inversion principle

Là một phương pháp cụ thể cho các mo-dun phần mềm được kết nối lỏng lẻo. Khi tuân theo nguyên tắc này, các mối quan hệ phụ thuộc thông thường được thiết lập từ các mô-đun thiết lập chính sách cấp cao đến các mô-đun phụ thuộc cấp thấp sẽ bị đảo ngược, do đó hiển thị các mô-đun cấp cao độc lập với các chi tiết triển khai mô-đun cấp thấp.

Nguyên tắc nêu rõ:

* Các mô-đun cấp cao không được nhập bất cứ thứ gì từ các mô-đun cấp thấp. Cả hai nên phụ thuộc vào sự trừu tượng (ví dụ: giao diện).
* Sự trừu tượng không nên phụ thuộc vào chi tiết. Chi tiết (triển khai cụ thể) sẽ phụ thuộc vào sự trừu tượng.

Bằng cách quy định rằng cả đối tượng cấp cao và cấp thấp đều phải phụ thuộc vào cùng một sự trừu tượng.

Dependency inversion principle được coi là một khái niệm duy nhất cần được khái quát hóa, tức là áp dụng cho tất cả các giao diện giữa các mô-đun phần mềm.

* Sẽ đơn giản hơn khi xem một nguyên tắc tư duy tốt như một mẫu mã hóa. Khi một lớp trừu tượng hoặc một giao diện đã được mã hóa.
* Bởi vì nhiều công cụ kiểm tra đơn vị dựa vào tính kế thừa để thực hiện mô phỏng , nên việc sử dụng giao diện chung giữa các lớp (không chỉ giữa các mô-đun khi sử dụng tính tổng quát là hợp lý) đã trở thành quy tắc.

## Tìm hiểu khái niệm Dependency inversion (DI)

Trong công nghệ phần mềm DI là một kỹ thuật lập trình trong đó một dối tượng hoặc hàm nhận các đối tượng hoặc hàm khác mà nó yêu cầu, trái ngược với việc tạo chúng từ bên trong.

Việc chèn DI nhằm mục đích tách biệt các mối quan tâm trong việc xây dựng các đối tượng sử dụng chúng, dấn đến các chương trình được liên kết lỏng lẻo.

Mẫu đảm bảo rằng một đối tượng hoặc chức năng muốn sử dụng một dịch vụ nhất định không cần phải biết cách xây dựng các dịch vụ đó. Thay vào đó, ' client ' (đối tượng hoặc hàm) nhận được cung cấp các phần phụ thuộc của nó bằng mã bên ngoài (một 'bộ tiêm') mà nó không biết.

## Vai trò của Dependency inversion

Dependency inversion có 4 vai trò chính: service, clients, interface and injectors

### Services and clients

Dịch vụ là bất kỳ lớp nào có chứa chức năng hữu ích. Đổi lại, khách hàng là bất kỳ lớp nào sử dụng dịch vụ. Các dịch vụ mà khách hàng yêu cầu là các dịch vụ phụ thuộc của khách hàng .

### Interfaces

Khách hàng không nên biết các phần phụ thuộc của họ được triển khai như thế nào, chỉ biết tên và API của họ.

### Injectors

Vai trò của Injectors là xây dựng và kết nối các biểu đồ đối tượng phức tạp, trong đó các đối tượng có thể vừa là khách hàng vừa là dịch vụ. Bản thân bộ tiêm có thể có nhiều đối tượng làm việc cùng nhau nhưng không được là máy khách, vì điều này sẽ tạo ra sự phụ thuộc vòng tròn .

## Khái niệm Service Objects Design Pattern

Service Objects giúp giảm tính xử lý ở Controller và Models .

Một Service Object là một PORO (Plain Old Ruby Object), có nghĩa là tách các vấn đề có tính nghiệp vụ vào một thành các Class và methods để hỗ trợ việc quản lý.

### Nguyên tắc chính:

* Chỉ có một public method: Service object chỉ nên có một method public để cung cấp cho đối tượng. Thường gặp tên method public là call, perform, run…
* Quản lý thuộc tính rõ ràng: Tùy thuộc vào tổ chức thuộc tính của từng project, nhưng quy tắc chung là khởi tạo thuộc tính đồng thời với khởi tạo đối tượng.

Link tham khảo.

<https://en.wikipedia.org/wiki/Dependency_inversion_principle>  
<https://en.wikipedia.org/wiki/Dependency_injection>