**[HDDT-Intern] Tìm hiểu DI – Nguyễn Đặng Nhật Hào**

1. **Nguyên tắc thiết kế hướng đối tượng**

**SOLID** là viết tắt của 5 chữ cái đầu trong 5 nguyên tắc thiết kế hướng đối tượng. Giúp cho lập trình viên viết ra những đoạn code dễ đọc, dễ hiểu, dễ maintain. Nó được đưa ra bởi **Robert C. Martin** và **Michael Feathers**. 5 nguyên tắc đó bao gồm:

* **S**ingle responsibility priciple (SRP)
* *Một class chỉ nên giữ 1 trách nhiệm duy nhất (Chỉ có thể sửa đổi class với 1 lý do duy nhất)*

Open/Closed principle (OCP)

* *Có thể thoải mái mở rộng 1 class, nhưng không được sửa đổi bên trong class đó (open for extension but closed for modification).*
* **L**iskov substitution principe (LSP)
* *Trong một chương trình, các object của class con có thể thay thế class cha mà không làm thay đổi tính đúng đắn của chương trình*
* **I**nterface segregation principle (ISP)
* *Thay vì dùng 1 interface lớn, ta nên tách thành nhiều interface nhỏ, với nhiều mục đích cụ thể*
* **D**ependency inversion principle (DIP)
* *Các module cấp cao không nên phụ thuộc vào các modules cấp thấp. Cả 2 nên phụ thuộc vào abstraction.*
* *Interface (abstraction) không nên phụ thuộc vào chi tiết, mà ngược lại. (Các class giao tiếp với nhau thông qua interface, không phải thông qua implementation.)*

1. **Định nghĩa và khái niệm DI**

* **Dependency Inversion:** Đây là một nguyên lý để thiết kế và viết code.
* **Inversion of Control:** Đây là một design pattern được tạo ra để code có thể tuân thủ nguyên lý Dependency Inversion. Có nhiều cách hiện thực pattern này: ServiceLocator, Event, Delegate, … Dependency Injection là một trong các cách đó.
* **Dependency Injection:** Đây là một cách để hiện thực Inversion of Control Pattern (Có thể coi nó là một design pattern riêng cũng được). Các module phụ thuộc (dependency) sẽ được inject vào module cấp cao.

Có thể hiểu Dependency Injection một cách đơn giản như sau:

* Các module không giao tiếp trực tiếp với nhau, mà thông qua interface. Module cấp thấp sẽ implement interface, module cấp cao sẽ gọi module cấp thấp thông qua interface.
  + *Ví dụ: Để giao tiếp với database, ta có interface IDatabase, các module cấp thấp là XMLDatabase, SQLDatabase. Module cấp cao là CustomerBusiness sẽ chỉ sử dụng interface IDatabase.*
* Việc khởi tạo các module cấp thấp sẽ do DI Container thực hiện. Ví dụ: Trong module CustomerBusiness, ta sẽ không khởi tạo IDatabase db = new XMLDatabase(), việc này sẽ do DI Container thực hiện. Module CustomerBusiness sẽ không biết gì về module XMLDatabase hay SQLDatabase.
* Việc Module nào gắn với interface nào sẽ được config trong code hoặc trong file XML.
* DI được dùng để làm giảm sự phụ thuộc giữa các module, dễ dàng hơn trong việc thay đổi module, bảo trì code và testing.
  1. **Các dạng của DI**

Có 3 dạng Dependency Injection:

* **Constructor Injection:** Các dependency sẽ được container truyền vào (inject vào) 1 class thông qua constructor của class đó. Đây là cách thông dụng nhất.
* **Setter Injection:** Các dependency sẽ được truyền vào 1 class thông qua các hàm Setter.
* **Interface Injection:** Class cần inject sẽ implement 1 interface. Interface này chứa 1 hàm tên Inject. Container sẽ injection dependency vào 1 class thông qua việc gọi hàm Inject của interface đó. Đây là cách rườm rà và ít được sử dụng nhất.
  1. **Ưu nhược điểm của DI**

|  |  |
| --- | --- |
| Ưu điểm | Nhược điểm |
| * Giảm sự kết dính giữa các module * Code dễ bảo trì, dễ thay thế module * Rất dễ test và viết Unit Test * Dễ dàng thấy quan hệ giữa các module (Vì các dependency đều được inject vào constructor) | * Khái niệm DI khá khó hiểu, các developer mới sẽ gặp khó khăn khi học. * Sử dụng interface nên đôi khi sẽ khó debug, do không biết chính xác module nào được gọi. * Các object được khởi tạo toàn bộ ngay từ đầu, có thể làm giảm performance. * Làm tăng độ phức tạp của code. |

1. **Áp dụng DI vào bài tập quản lí sinh viên**

[*https://github.com/cauhamau/SDS\_Training/tree/master/Practice3\_ServiceDI\_StudentManagement*](https://github.com/cauhamau/SDS_Training/tree/master/Practice3_ServiceDI_StudentManagement)

Link tham khảo:

[*https://topdev.vn/blog/solid-la-gi*](https://topdev.vn/blog/solid-la-gi)

[*https://toidicodedao.com/2015/03/24/solid-la-gi-ap-dung-cac-nguyen-ly-solid-de-tro-thanh-lap-trinh-vien-code-cung/*](https://toidicodedao.com/2015/03/24/solid-la-gi-ap-dung-cac-nguyen-ly-solid-de-tro-thanh-lap-trinh-vien-code-cung/)

[*https://toidicodedao.com/2015/11/03/dependency-injection-va-inversion-of-control-phan-1-dinh-nghia/*](https://toidicodedao.com/2015/11/03/dependency-injection-va-inversion-of-control-phan-1-dinh-nghia/)