**Báo cáo**

1. Tight coupled

là liên kết ràng buộc giữa các class quá chặt chẽ. Khi thay đổi 1 class thì sẽ ảnh hưởng đến class khác. Điều này sẽ không tốt trong quá trình phát triển và làm việc nhóm do khi thay đổi 1 class thì phải sửa tất cả các class liên quan.

1. Loosely coupled

là giảm sự liên kết giữa các class.

1. Dependency injection

3.1 Tổng quan

* là một design pattern cho phép chúng ta loại bỏ sự phụ thuộc cứng nhắc giữa các phần tử và làm cho ứng dụng trở nên linh hoạt mềm dẻo hơn, dễ mở rộng và bảo trì hơn
* cung cấp cho một đối tượng các thể hiện phụ thuộc (dependencies) của nó từ bên ngoài truyền vào mà không phải khởi tạo trực tiếp từ trong class sử dụng.
  1. Nhiệm vụ
* Tạo các đối tượng
* Quản lý sự phụ thuộc giữa các đối tượng
* Cung cấp các phụ thuộc được yêu cầu cho đối tượng
  1. Nguyên tắc hoạt động
* 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.
* Việc khởi tạo các module cấp thấp sẽ do DI Container/ IoC Container thực hiện.
* Việc Module nào gắn với interface nào sẽ được config trong file properties, trong file XML hoặc thông qua Annotation. Annotation là một cách thường được sử dụng trong các Framework, chẳng hạn như @Autowired với [Spring](https://spring.io/) .
  1. Cách cài đặt

Ảnh có chứa biểu đồ

Mô tả được tạo tự động

* Client : là 1 class cần sử dụng Service
* Service : là 1 class/interface cung cấp service/dependency cho client
* ServiceImpl : cài đặt các phương thức cụ thể của Service
* Injector : là 1 lớp chịu trách nhiệm khởi tạo các service và inject các thể hiện này cho Client
  1. Các 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.
* Fields/ properties: Các dependency sẽ được truyền vào 1 class một cách trực tiếp vào các field.
* **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 cũng ít được sử dụng.
* [**Service Locator**](https://gpcoder.com/4946-huong-dan-java-design-pattern-service-locator/): nó hoạt động như một mapper, cho phép thay đổi code tại thời điểm run-time mà không cần biên dịch lại ứng dụng hoặc phải khởi động lại.

1. File cấu hình và Profiles

* Spring Profiles là một core feature trong **Spring Framework**, cho phép cấu hình ứng dụng, active/deactive bean tùy theo môi trường.
* Spring sẽ nhận các tên file như sau trong thư mục resource:
* application.properties
* application.yml
* application-{tên file}.yml // properties
* các bean:
* @Component : khai báo 1 bean
* @Value : lấy giá trị từ file yml hoặc properties
* @Autowired : gọi đến phương thức khởi tạo
* @Primary: được ưu tiên gọi đến phương thức khởi tạo khi lớp cha có nhiều lớp con