**BÁO CÁO EX04\_SPRING BOOT BASIC**

1. Tight Coupling và Loosely Coupling
2. Tight Coupling

* Tight Coupling (kết nối chặt): Đây là một kiểu thiết kế trong đó các thành phần hoạt động cùng nhau một cách ràng buộc và phụ thuộc mạnh mẽ. Khi một phần thay đổi, các thành phần khác có thể bị ảnh hưởng và phải thay đổi tương ứng. Điều này làm cho mã nguồn khó tái sử dụng, bảo trì và kiểm thử.

1. Loosely Coupling

* Loosely Coupling (kết nối lỏng lẻo): Đây là một kiểu thiết kế mà các thành phần hoạt động độc lập và không phụ thuộc mạnh mẽ vào nhau. Các thành phần giao tiếp thông qua giao diện hoặc hợp đồng chung và không cần biết chi tiết về nhau. Điều này giúp tăng tính linh hoạt, tái sử dụng và bảo trì của mã nguồn.

1. Dependency Injection
2. Dependency Injection là gì?

* Dependency Injection là một nguyên tác thiết kế phần mềm giúp giảm sự phụ thuộc mạnh mẽ giữa các thành phần và tạo sự kết nối lỏng lẻo.
* Dependency Injection cung cấp các cơ chế để cung cấp các phụ thuộc (dependencies) cho một đối tượng mà không cần nó tự tạo ra chúng. Thay vào đó các phụ thuộc được cung cấp từ bên ngoài (thông qua constructor, setter hoặc các phương thức khác).
* Điều này giúp giảm sự phụ thuộc giữa các thành phần, tăng tính linh hoạt và tái sử dụng của mà nguồn.

1. Ưu và khuyết điểm của Dependency Injection

* Ưu điểm:
* Reduced dependencies: giảm sự kết dính giữa các module.
* Reusable: code dễ bảo trì, dễ tái sử dụng, thay thế module. Giảm boiler-plate code do việc tạo các biến phụ thuộc đã được injector thực hiện.
* Testable: rất dễ test và viết Unit Test.
* Readable: dễ dàng thấy quan hệ giữa các module vì các dependency đều được inject vào constructor.
* Khuyết điểm:
* Khái niệm Dependency Injection khá khó hiểu đối với người mới tìm hiểu..
* 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.
* Có thể gặp lỗi ở run-time thay vì compile-time.

1. Sử dụng Dependency Injection khi nào?

* Khi cần inject các giá trị từ một cấu hình cho một hoặc nhiều module khác nhau.
* Khi cần inject một dependency cho nhiều module khác nhau.
* Khi cần một vài service được cung cấp bởi container.
* Khi cần tách biệt các dependency giữa các môi trường phát triển khác nhau.

1. File cấu hình và profile
2. File cấu hình

* Là nơi chứa các thông tin cấu hình cho ứng dụng, chẳng hạn như các bean, các tài nguyên, các thuộc tính và các thiết lập khác. File cấu hình giúp tác riêng phần cấu hình ra khỏi mã nguồn chín, làm cho ứng dụng dễ dàng thay đổi cấu hình mà không cần thay đổi mã nguồn
* Trong Spring có hai loại file cấu hình phổ biến là application.properties và application.yml
* Application.properties: là một file cấu hình được sử dụng trong Spring Framework để cấu hình các thuộc tính của ứng dụng. Định dạng file properties sử dụng cặp key=value để cấu hình thuộc tính.
* Application.yml: là một file cấu hình YAML được sử dụng trong Spring Framework để cấu hình các thuộc tính của ứng dụng. YAML sử dụng cú pháp cấu trúc rõ ràng.

1. Profile

* Là một tính năng trong Spring Framework cho phép chúng ra cấu hình và chạy ứng dụng trên môi trường khác nhau (như local, development, production) mà không cần thay đổi mã nguồn. Profile cho phép chúng ta xác định các cấu hình và bean chỉ hoạt động trong môi trường cụ thể.