**BÁO CÁO VỀ SPRINGBOOT**

**I/ Springboot là gì?**

1. **Spring framework**

Framework là một bộ công cụ và thư viện phát triển được thiết kế để giúp việc xây dựng ứng dụng trở nên dễ dàng hơn. Nó cung cấp một cấu trúc, các tiêu chuẩn và các công cụ giúp lập trình viên tập trung vào việc triển khai logic kinh doanh thay vì phải viết lại các thành phần cốt lõi của ứng dụng.

Spring Framework là một trong những framework phát triển ứng dụng Java phổ biến nhất. Nó cung cấp một cách tiếp cận mô hình phát triển linh hoạt và mạnh mẽ, giúp xây dựng các ứng dụng doanh nghiệp và ứng dụng web hiệu quả. Với Spring Framework, ta có thể sử dụng các module như Spring Boot, Spring MVC, Spring Data, Spring Security và nhiều module khác để xây dựng các ứng dụng phức tạp và chất lượng cao.

Spring Framework được xây dựng dựa trên 2 nguyên tắc design chính:

* Dependency Injection (DI)

***IoC là gì?***

IoC (Inversion of Control) là một nguyên tắc trong lập trình, trong đó quyền kiểm soát các đối tượng và luồng của chương trình được chuyển từ phần mềm chính tới một framework hoặc container. Thay vì phần mềm gọi các hàm hoặc khởi tạo các đối tượng, thì framework sẽ gọi các hàm và khởi tạo các đối tượng để tiếp quản và điều khiển luồng của chương trình. Điều này giúp tăng tính linh hoạt, dễ bảo trì và tái sử dụng mã nguồn

IoC Container là thành phần thực hiện IoC. Trong Spring, Spring IoC (IoC Container) là một phần quan trọng trong Spring để quản lý vòng đời và dependencies của các đối tượng trong ứng dụng. Nó có vai trò quản lý đối tượng, có trách nhiệm tạo ra, cấu hình và quản lý các đối tượng bean của ứng dụng. Thay vì tái tạo đối tượng bằng cách gọi trực tiếp từ mã, ta chỉ cần định nghĩa các đối tượng trong cấu hình (tệp XML hoặc thông qua Java Config) và container sẽ tự động khởi tạo, khởi tạo dependencies, quản lý và giải phóng tài nguyên khi không cần thiết nữa. Nó cũng đảm bảo dependencies được đưa vào bean một cách tự động, dễ dàng thiết lập và thay thế.

IoC giúp tách biệt và quản lý phụ thuộc giữa các thành phần trong ứng dụng, làm cho mã của bạn linh hoạt hơn và dễ dàng thay đổi hoặc kiểm thử.

***DI là gì?***

DI (Dependency Injection) là một cách triển khai của IoC. Nó tập trung vào việc cung cấp các phụ thuộc (dependencies) cho một đối tượng từ bên ngoài. Thay vì đối tượng tự tạo hoặc tìm kiếm những đối tượng phụ thuộc của nó, các phụ thuộc được chuyển vào từ bên ngoài. Các phụ thuộc này có thể là các đối tượng, các giao diện, hoặc các giá trị cấu hình. DI giúp tách biệt các thành phần và làm cho chúng dễ dàng thay thế và kiểm soát.

|  |  |
| --- | --- |
| Thay vì dùng như này: | Nên dùng như này: |
| *public class UserRepository {*  *private DatabaseConnection dbConnection;*  *public UserRepository () {*  *this.dbConnection = new DatabaseConnection();*  *}*  *}*  *UserRepository trực tiếp tạo ra và nắm quyền điều khiển DatabaseConnection. Gây bất tiện khi muốn thay đổi* | *public class UserRepository {*  *private DatabaseConnection dbConnection;*  *public UserRepository (DatabaseConnection dbConnection) {*  *this.dbConnection = dbConnection;*  *}*  *}*  *UserRepository không tạo trực tiếp một đối tượng DatabaseConnection, nhưng nó được cung cấp từ bên ngoài qua constructor của UserRepository. Điều này làm linh hoạt trong việc thay đổi mà không ảnh hưởng đến UserRepository.* |

* Aspect Oriented Programming (AOP)

1. **Springboot**

***Tại sao phải sử dụng springboot?***

Vấn đề đặt ra như sau: Khi 1 project có quá nhiều dependency và ta phải inject chúng, điều này gây ra sự khó khăn, vất vả. Việc xuất hiện của springboot như 1 kho chứa các dependencies và sẽ tự động inject (thông qua @Autowired), giúp rút ngắn thời gian lập trình, giảm sự rườm rà.

Spring Boot là một module của Spring Framework giúp các lập trình viên loại bỏ các bước cấu hình phức tạp mà Spring bắt buộc. Spring Boot là dự án phát triển bởi ngôn ngữ Java (JAV) trong hệ sinh thái Spring framework.

Một số tính năng nổi bật của Spring Boot đó là:

* Tạo ra các ứng dụng Spring mang tính độc lập.
* Nhúng trực tiếp Tomcat, Jetty hoặc Undertow mà không cần phải triển khai ra file WAR.
* Starter dependency giúp cho việc chạy cấu hình Maven trở nên đơn giản hơn.
* Tự động chạy cấu hình Spring khi cần thiết.
* Không sinh code cấu hình, đồng thời không yêu cầu phải cấu hình bằng XML.

Spring có kích thước rất nhỏ, trong suốt và nhẹ trong quá trình chạy. Chính vì kích thước nhỏ, nên đây là một giải pháp khá gọn, nhẹ với khả năng hỗ trợ để tạo ra và phát triển các ứng dụng web có mã hiệu suất cao. Hơn hết, có thể dễ dàng kiểm tra, thử nghiệm hoặc sử dụng lại code.

Những tính năng cốt lõi của Spring framework có thể được sử dụng trong việc phát triển java desktop, java web, … Mục tiêu chính là dễ dàng phát triển các ứng dụng J2EE dựa trên mô hình sử dụng POJO

**II/ Caching dữ liệu**

1. **Giới thiệu**

**@Cacheable**

@Cacheable được sử dụng để chỉ định rằng kết quả của việc gọi một phương thức sẽ được lưu vào bộ nhớ cache và được tái sử dụng nếu cùng một phương thức được gọi với cùng các đối số.

**@CacheEvict**

@CacheEvict được sử dụng để xóa một hoặc nhiều mục khỏi bộ nhớ cache. Nó có thể được sử dụng để xóa cache dựa trên các điều kiện cụ thể hoặc khi một phương thức được gọi.

**@CachePut**

@CachePut được sử dụng để chỉ định rằng giá trị trả về của một phương thức luôn được lưu vào bộ nhớ cache, bất kể phương thức đã được gọi trước đó hay không. Điều này hữu ích khi bạn muốn đảm bảo rằng kết quả của một phương thức luôn được lưu trữ trong cache.

**@Caching**

@Caching được sử dụng để nhóm nhiều chú thích caching trên một phương thức duy nhất. Nó cho phép bạn áp dụng nhiều hành vi caching khác nhau cho một phương thức, kết hợp hiệu ứng của @Cacheable, @CachePut, và @CacheEvict.