* **Dependency Injection (DI)** là một mô hình lập trình và thiết kế phần mềm, đây là một phương pháp giảm sự phụ thuộc giữa các thành phần (hoặc lớp)
* Trong DI các phụ thuộc một đối tượng không được tạo bên trong đối tượng đó, mà được cung cấp từ bên ngoài
* DI thường được thể hiện thông qua 3 cách chính:
* **Constructor Injection**
* Setter Injection
* Interface Injection

**IoC (Inversion of Control)** là một nguyên tắc lập trình trong việc quản lý sự phụ thuộc giữa các thành phần trong một hệ thống phần mềm. Thay vì các thành phần tự mình tạo ra và quản lý sự phụ thuộc của chúng, IoC chuyển trách nhiệm này cho một thực thể bên ngoài được gọi là Container hay Framework.

**IoC container** là một cơ chế giúp các đối tượng trong ứng dụng không cần tự khởi tạo các phụ thuộc của mình. IoC Container sẽ tạo ra các đối tượng, nối chúng lại với nhau, cấu hình chúng, và quản lý vòng đời của chúng từ khi tạo ra đến khi bị hủy. IoC Container sử dụng DI để quản lý các thành phần tạo nên một ứng dụng. Những đối tượng này được gọi là Spring Bean.

Những object mà được tạo và quản lý bởi **Spring IoC Container** thì được gọi là **Bean**.

Bean được quản lý như thế nào?

**ApplicationContext** là giao diện (interface) đại diện cho Spring IoC Container trong Spring Boot. Nó chịu trách nhiệm quản lý vòng đời của các Bean, cung cấp các dịch vụ như Dependency Injection, quản lý cấu hình, và hỗ trợ các tính năng nâng cao như i18n (quốc tế hóa) và event publishing.

Ngoài ApplicationContext, bạn có thể gặp khái niệm BeanFactory. BeanFactory là một IoC container cơ bản, cung cấp các chức năng cốt lõi để quản lý Bean. ApplicationContext mở rộng BeanFactory, cung cấp nhiều tính năng hơn và phù hợp hơn cho các ứng dụng Spring Boot hiện đại.

Khi ứng dụng Spring Boot khởi chạy, Spring IoC Container sẽ thực hiện Component Scan, quét các packages để tìm kiếm các class được đánh dấu là Bean và đưa chúng vào ApplicationContext.

Spring IoC Container quản lý bean thông qua các bước sau:

Định nghĩa bean: Bean được định nghĩa trong ứng dụng thông qua:

* File XML: Ví dụ <bean id="myBean" class="com.example.MyBean"/>.
* Java Config: Sử dụng @Configuration và @Bean.
* Annotation: Sử dụng @Component, @Service, @Repository, @Controller kết hợp với @ComponentScan.

Trong IoC Container có hai loại chính:

1. **2.1 BeanFactory**

* **BeanFactory** là một IoC container đơn giản và cơ bản trong **Spring Framework**. Nó cung cấp các khả năng quản lý các **bean** (các đối tượng Java), bao gồm việc khởi tạo và xử lý phụ thuộc.
* BeanFactory khởi tạo bean khi được yêu cầu (lazy loading). Điều này có nghĩa là bean chỉ được khởi tạo khi nó thực sự cần thiết, giúp tiết kiệm tài nguyên, đặc biệt trong các ứng dụng có nhiều bean mà không phải tất cả đều được sử dụng cùng lúc.

**Ưu điểm:**

* Phù hợp với các ứng dụng nhỏ, nơi bạn không cần quản lý phức tạp.
* Hiệu quả với việc quản lý tài nguyên vì bean chỉ được tạo khi cần thiết.

**Hạn chế:**

* Không hỗ trợ một số tính năng cao cấp như xử lý các event hoặc quản lý các lifecycle hook (callback method).

1. **2.2 ApplicationContext**

* **ApplicationContext** là một loại IoC container tiên tiến hơn. Nó mở rộng BeanFactory và cung cấp thêm nhiều tính năng bổ sung như event propagation, hỗ trợ internationalization (i18n), quản lý các event lifecycle, và khả năng tương tác với các resource file.
* ApplicationContext khởi tạo **tất cả các bean** tại thời điểm container được load (eager loading), tức là các bean sẽ được khởi tạo ngay lập tức khi container khởi động, thay vì đợi đến lúc được sử dụng như trong BeanFactory.

**Ưu điểm:**

* Hỗ trợ quản lý lifecycle cho các bean thông qua các callback method như init và destroy.
* Cho phép lắng nghe và phát các event trong ứng dụng, giúp ứng dụng có thể phản hồi nhanh chóng khi có sự thay đổi.

**Hạn chế:**

* Khởi tạo các bean ngay khi ứng dụng bắt đầu (eager loading), có thể làm tăng thời gian khởi động nếu số lượng bean lớn.

1. **2.3 Sự khác biệt chính**

* **BeanFactory** sử dụng lazy loading, chỉ khởi tạo bean khi cần, còn ApplicationContext sử dụng eager loading, khởi tạo tất cả các bean ngay khi container khởi chạy.
* **ApplicationContext** hỗ trợ nhiều tính năng hơn như quản lý event, resource, và lifecycle, trong khi BeanFactory chỉ cung cấp các tính năng cơ bản.

Tùy vào nhu cầu và độ phức tạp của ứng dụng, bạn có thể chọn loại IoC container phù hợp. Trong các ứng dụng Spring hiện đại, ApplicationContext được sử dụng phổ biến hơn do tính năng phong phú và khả năng mở rộng tốt hơn.

