Dùng bean có thể sử dụng lại được đối tượng ở 2 class khác nhau mà không phải khởi tạo 1 đối tượng mới. Dùng từ khóa new sẽ khởi tạo 1 đối tượng mới hoàn toàn.

Việc sử dụng bean trong spring, spring sẽ quản lý vòng đời của đối tượng được tạo ra và các phụ thuộc của nó tự động, trong khi đó nếu dùng new sẽ phải tự mình quản lý vòng đời của đối tượng và các phụ thuộc của nó. Xóa obj trong java bằng cách gán obj bằng null hoặc gán bằng new và không làm gì.

Ioc giúp quản lý sự phụ thuộc mà không phải động chạm vào source code. Vd khi thay đổi đường dẫn hoặc cách làm việc với db, ta không phải động chạm vào source code mà chỉ việc thay đổi trong file cấu hình…

**Vòng đời bean:**

* Tự khởi tạo tất cả các bean khi gọi applicationcontext
* Khi gọi context.registerShutdownHook(), các bean sẽ bị hủy
* Trong web app khi server tắt các bean sẽ tự động hủy

**Sử dụng annotation : @Required, @Autowired**

* Required: đặt trong setter, yêu cầu phải có setter đó khi khởi tạo bean.
* Autowired
* On property: ưu tiên nối với bean theo type, nếu có nhiều bean có type giống nhau no sẽ tự động bắt bean theo name.
* On setter : giống on property

@Configuration và @Bean annotation

* Sử dụng @Configurtion trong class chỉ ra class có thể được sử dụng bởi Spring IoC container như 1 nguồn định nghĩa bean. 1 phương thức có @Bean sẽ trả về 1 đối tượng mà được coi như là 1 bean trong Spring application context. Ví dụ đơn giản:

@Configuration

public class AppConfig {

@Bean

public HelloWorld helloWorld(){

return new HelloWorld();

}

}

Code trên tương tự như đoạn cấu hình XML dưới:

<beans>

<bean id = "helloWorld" class = "java.HelloWorld" />

</beans>

Tên của phương thức được đinh nghĩa với @Bean làm việc như bean ID, nó tao ra và trả về 1 bean. Class cấu hình có thể có 1 hoặc nhiều @Bean. Khi các class cấu hình được định nghĩa, có thể load nó bằng cách cung cấp chúng vào Spring container sử dụng:

ApplicationContext ctx = new AnnotationConfigApplicationContext(AppConfig.class);

Có thể load nhiều class cấu hình bằng cách:

AnnotationConfigApplicationContext ctx = new AnnotationConfigApplicationContext();

ctx.register(AppConfig.class, OtherConfig.class);

ctx.register(AdditionalConfig.class);

ctx.refresh();

**@Import annotation**

@Import cho phép tải các định nghĩa @Bean từ các class cấu hình khác. Vd:

@Configuration

@Import(ConfigA.class)

public class ConfigB {

@Bean

…

}

Thay vì chỉ rõ cả 2 ConfigA.class và ConfigB.class trong application context, giờ chỉ việc thêm vào ConfigB là được.