



# 02: Spring Bean

Trần Văn Thịnh



# Spring bean

Spring bean là gì

IoC và DI

- Khái niệm IoC và DI
- Các cách thực hiện DI trong Spring

Spring IoC Container

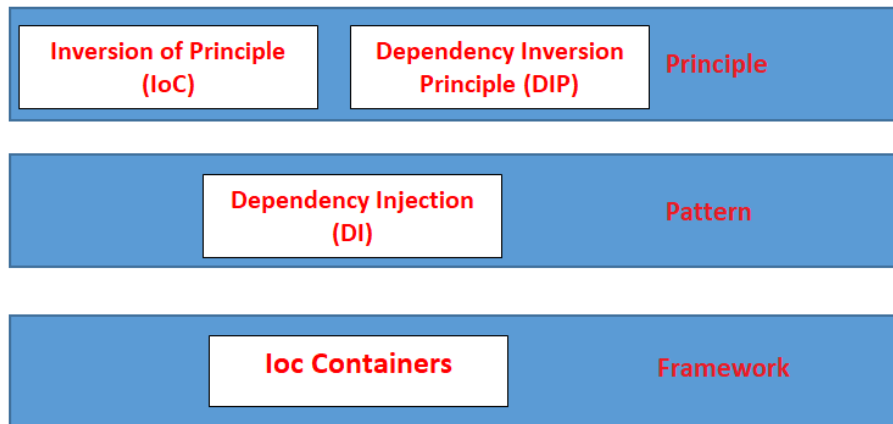
# Spring bean là gì



WHAT IS SPRING BEAN

- Khái niệm quan trọng trong Spring
- Là một object được khởi tạo, lắp ráp, hủy bỏ và quản lý bởi Spring IoC Container
- Beans và các dependencies của nó được phản ánh trong dữ liệu cấu hình metadata, sử dụng bởi container

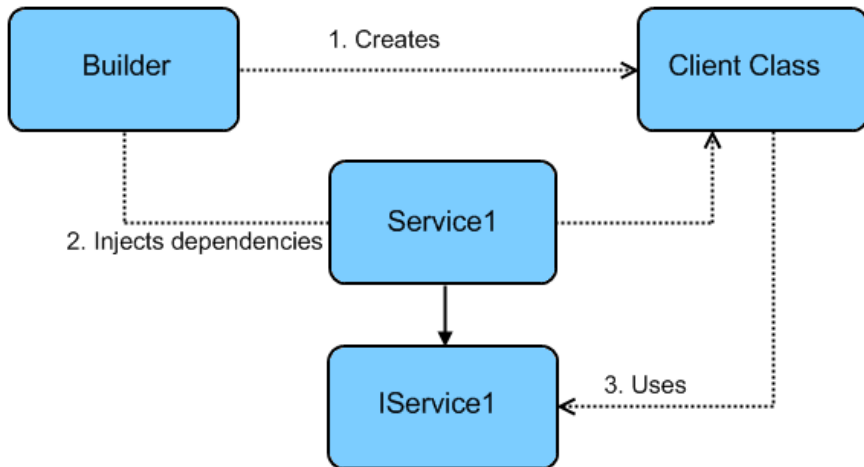
# IoC và DI



Container về mặt kỹ thuật, mô tả các thành phần (component) cung cấp cơ sở hạ tầng cho các thành phần khác

- IoC: một nguyên tắc (principle) trong phần mềm, chuyển quyền kiểm soát các objects hoặc các thành phần của một chương trình sang một container hay 1 framework
- Lợi ích:
  - Tách việc thực hiện một nhiệm vụ ra khỏi việc triển khai (implementation)
  - Chuyển đổi giữa các implementation khác nhau dễ dàng
  - Giúp chương trình đóng gói module dễ dàng
  - Giúp việc kiểm thử từng thành phần dễ dàng

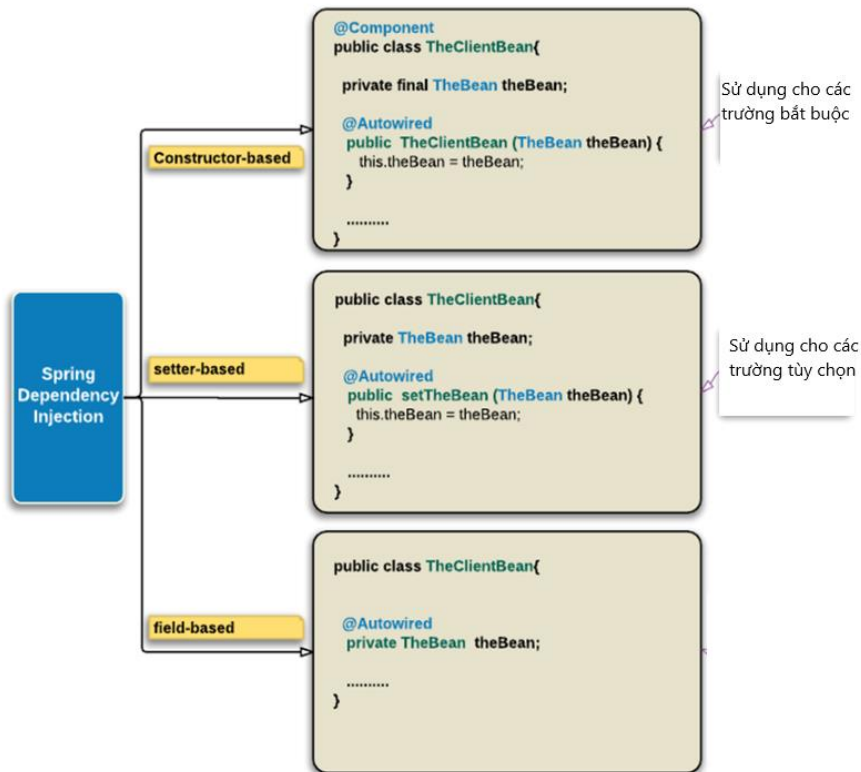
# IoC và DI



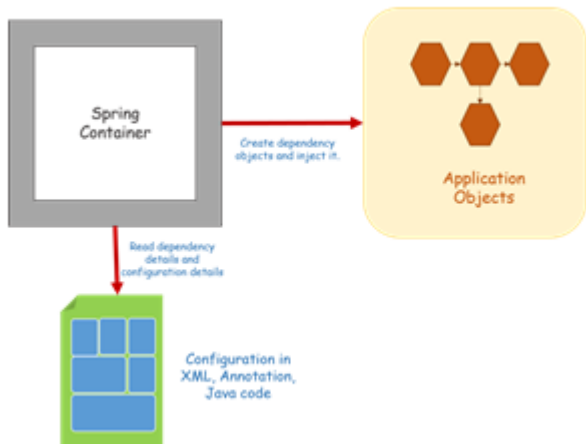
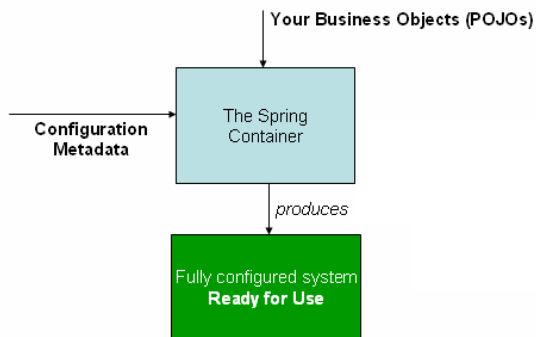
- DI (chèn, tiêm phụ thuộc) là một design pattern (mẫu thiết kế) thực thi IoC, trong đó điều khiển được đảo ngược là việc thiết lập các dependencies của một object
- Việc kết nối các đối tượng với các đối tượng khác, hoặc "tiêm" (inject) đối tượng vào các đối tượng khác, được thực hiện bởi một trình lắp ráp chứ không phải bởi chính các đối tượng.

# IoC và DI

## Different ways of DI in Spring

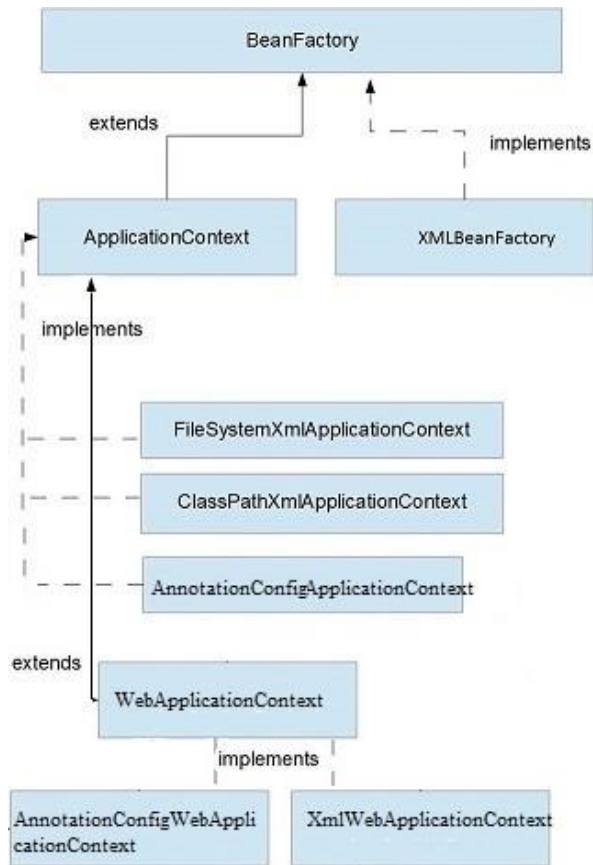


# Spring (IoC) Container



- Spring (IoC) Container là trung tâm của hệ thống Spring, quản lý vòng đời hoàn chỉnh của mỗi bean từ khi tạo ra đến khi bị phá hủy
- Spring container sử dụng DI để quản lý bean

# Spring (IoC) Container



- Đại diện cho Spring IoC Container:

- **BeanFactory**: root interface, thao tác cơ bản để quản lý bean trong ứng dụng
- **ApplicationContext**: sub-interface với thêm một số tính năng quan trọng:
  - Bean instantiation/wiring
  - Automatic **BeanPostProcessor** registration
  - Automatic **BeanFactoryPostProcessor** registration
  - Convenient **MessageSource** access (for i18n)
  - **ApplicationEvent** publication



# CẤU HÌNH SPRING BEAN

- Làm thế nào để định nghĩa một bean trong Spring
  - Các cách khác nhau để định nghĩa Spring bean
  - Bean annotation trong Spring
- Spring bean properties
  - Làm thế nào để định nghĩa nhiều bean của cùng một class
- IoC tạo một instance của bean như thế nào

# Làm thế nào để định nghĩa một bean trong Spring

- Sử dụng Annotations trong Spring – stereotype annotation with @ComponentScan (auto scan)

@Component

Generic annotation chỉ ra java class là một bean. Spring sử dụng cơ chế auto scan để tìm và đăng kí bean này vào context

@Repository

Kế thừa từ Component, chỉ ra class này là repository(tương tác với DB)

@Service

Kế thừa từ Component, chỉ ra class này là service, chứa tất cả business logic

@Controller

Sử dụng ở class level trong spring MVC, đánh dấu class này là một spring web controller, xử lý HTTP request

# Làm thế nào để định nghĩa một bean trong Spring

- Sử dụng Annotations trong Spring – stereotype annotation with @ComponentScan (auto scan)

```
@Component
public class Student {
    private int Id;
    private String firstname;
    private String lastname;
    private String email;
    public Student() {
    }
    // Getters + Setters
}
```

# Làm thế nào để định nghĩa một bean trong Spring

- Sử dụng Annotations trong Spring – @Bean với @Configuration

```
@Configuration
public class SpringConfig {
    @Bean
    public Student studentBean() {
        return new Student();
    }
    @Bean
    public Employee employeeBean() {
        return new Employee();
    }
    // Other spring beans ...
}
```

# Spring bean properties

- class : java class của bean, bắt buộc
- id: id duy nhất của bean (unique bean identifier)
- name: cách khác để xác định bean identifier. Spring sử dụng id/name để xác định bean. Nếu các bean có cùng class thì sẽ cần sử dụng thêm
- Scope: scope của bean object
- Property: sử dụng chủ yếu để quản lý và xác định các bean dependencies, và giá trị của chúng thông qua setter method
- Initialization: initialization mode, lazily or eagerly

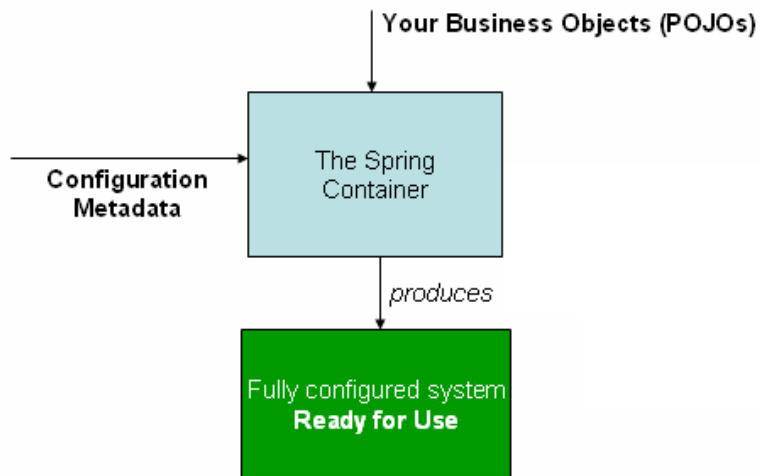
# Định nghĩa nhiều bean của cùng 1 class

- Thêm thuộc tính name của @Bean để phân biệt

```
@Configuration
public class MyBeansConfig {
    @Bean(name="simpleBeanA")
    public SimpleBean simpleBeanA() {
        return new SimpleBean();
    }
    @Bean(name="simpleBeanB")
    public SimpleBean simpleBeanB() {
        return new SimpleBean();
    }
}
```

# Spring IoC khởi tạo bean thế nào

- Khi chạy ứng dụng, spring trước tiên tạo IoC container
- Sau đó, container sẽ đọc cấu hình metadata được cung cấp bằng java hoặc xml, sau đó khởi tạo các beans được yêu cầu phụ thuộc vào các properties và các dependencies của chúng



# Bean Dependencies



# Bean Dependencies

- Dependencies là các object của các class mà là data member trong các class khác

```
public class Address {  
    private String street;  
    private String city;  
    public Address() {  
    }  
    // Getters + Setters  
}  
  
public class Employee {  
    private String firstname;  
    private String lastname;  
    // Dependency  
    private Address address  
    public Employee() {  
    }  
    // Getters + Setters  
}
```

# Bean Dependencies

- Định nghĩa một bean dependency
  - Không cần suy nghĩ đến việc làm thế nào để khởi tạo các objects hoặc quản lý tất cả các dependencies giữa chúng- Spring làm hết
  - Ở ví dụ dưới đây, Spring sử dụng constructor arguments như là bean dependencies

```
@Bean
public Employee empBean(Address address) {
    return new Employee(address);
}
```

```
@Component
public class Employee {
    private Address address;
    public Employee(Address address) {
        this.address = address;
    }
}
```

# Bean Dependencies

- Wiring trong Spring
  - @Autowired
  - @Resource (thuộc [JSR-250](#) annotation collection)
  - @Inject (thuộc [JSR-330](#) annotations collection)
- @Autowired thường dùng
- Độ ưu tiên
  - Match by Type
  - Match by Qualifier
  - Match by Name

# Bean Scope trong Spring

Bean Scope

Làm thế nào để định nghĩa  
Bean Scope

Spring bean life cycle

# Bean Scope

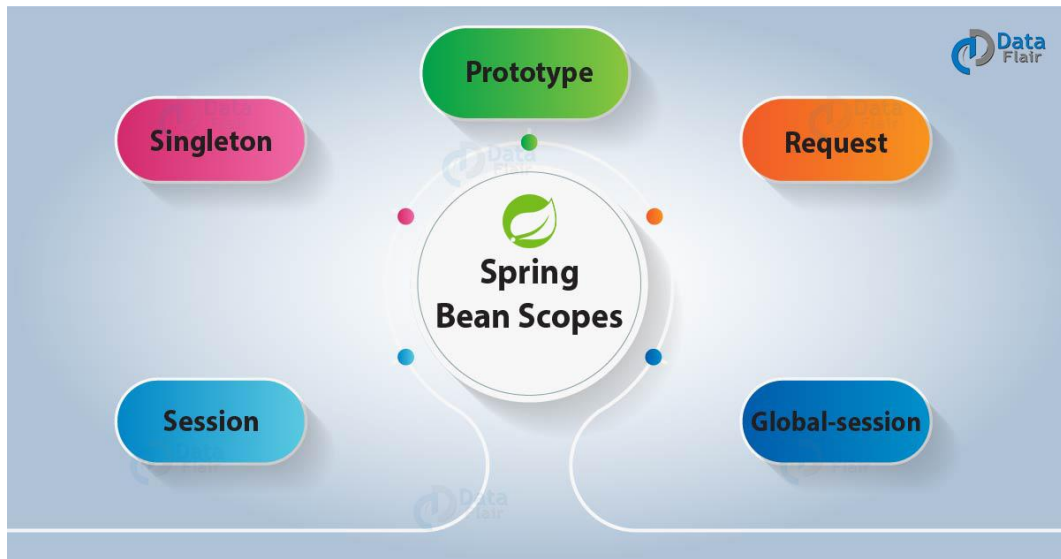
Scope	Description
singleton	This is the default scope, it means only one single instance of the bean will be created in the container!
prototype	Each time the bean is requested, a new instance will be created.
request	A new instance of the bean will be returned for each HTTP request.
session	This is same as request scope, a single instance for every HTTP session.
application	This value creates a single bean instance shared accross the same web-aware Spring ApplicationContext
websocket	Only one instance will be created during complete lifecycle of WebSocket.

# Định nghĩa bean scope

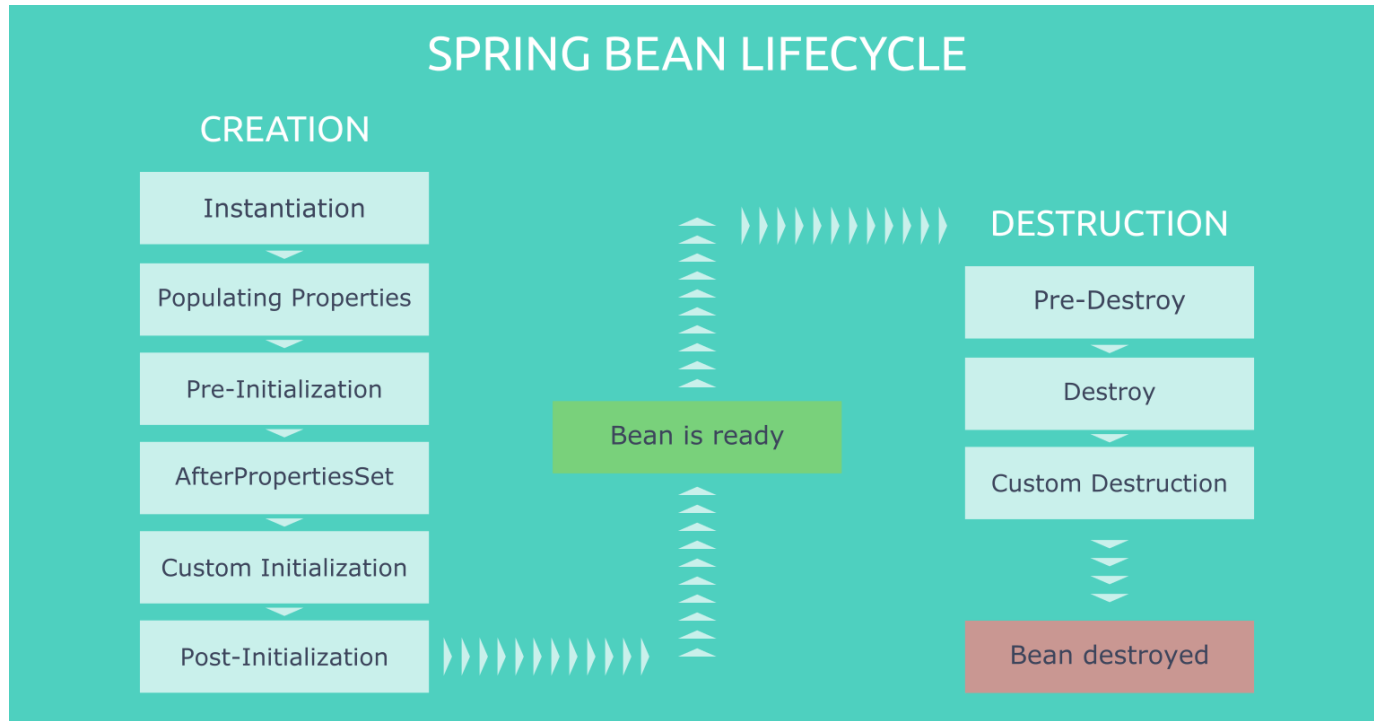
- @Scope

`@Scope(value = ConfigurableBeanFactory.SCOPE_SINGLETON)`

`@Scope(value = ConfigurableBeanFactory.SCOPE_PROTOTYPE)`



# Vòng đời của Spring bean



---

## **Get Bean với ApplicationContext : getBean() APIs**



# getBean() APIs

- Lấy Bean bằng Name

```
Object lion = context.getBean("lion");  
  
assertEquals(Lion.class, lion.getClass());
```

- Lấy Bean bằng Name và type

```
Lion lion = context.getBean("lion", Lion.class);
```

- Và một số method khác
- **Không nên sử dụng applicationContext.getBean() một cách trực tiếp**

# Sử dụng @SpringBootApplication

- Dùng để kích hoạt các tính năng sau
  - @EnableAutoConfiguration: Spring Boot's auto-configuration
  - @ComponentScan: Sử dụng để scan các component (@Component và các annotation thừa kế nó) trong các package
  - @Configuration: Cho phép đăng kí các bean vào trong context hoặc import thêm các configuration class

---

# DEMO