

# Bean Lifecycle

Nguyễn Anh Tuấn

**KTECH**  
COLLEGE



# Nội dung bài giảng

- 1 Cơ chế component scan
- 2 Cơ chế Lazy
- 3 Vòng đời của Bean

# Cơ chế Component scan

# Cơ chế Component scan

- **Component scan:**
  - Cho phép Spring Boot tự động tìm kiếm và quản lý các bean trong ứng dụng.
  - Mặc định, Spring Boot sẽ quét toàn bộ các package và các package con của package chứa hàm main.
  - Cú pháp: **@ComponentScan("domain.package")**
- **Thay đổi phạm vi scan:**
  - Trường hợp muốn tùy chỉnh cấu hình cho Component scan để chỉ tìm kiếm các bean trong 1 package nhất định, có 2 cách sau:
    - Chỉ định trực tiếp package trong **@ComponentScan**
    - Sử dụng **scanBasePackages**: 1 thuộc tính trong **@SpringBootApplication**

# Cơ chế Component scan

## Spring JavaConfig Component Scanning

It will cause Spring to scan the package "com.example" and will discover beans annotated with `@Component` or other stereotype annotations. This is alternative to using `<context:component-scan>`.

```
@Configuration
@ComponentScan("com.example")
public class MyAppConfig{
    //no bean definitions using @Bean methods here
}
```

Declaring a bean component which is to be scanned, we can use any of the followings:

- ▶ `@Component`
- ▶ `@Controller`
- ▶ `@Repository`
- ▶ `@Service`

SCAN

Package com.example

```
@Component
public class MyBean1{
    .....
}
```

```
@Component
public class MyBean2{
    @Autowired
    private MyBean1 bean1;
    .....
}
```

Injecting other beans  
just like before

```
@Component
@Lazy
public class MyBean3{
    .....
}
```

We can use most of the annotation which we use with `@Bean` methods :

- ▶ `@Scope`
- ▶ `@Lazy`
- ▶ `@DependsOn`
- etc

# Cơ chế Lazy

# Cơ chế Lazy

- **Eager:**
  - Bean được tạo ra ngay khi chạy chương trình.
  - Ví dụ: **singleton scope**.
- **Lazy:**
  - Bean được tạo ra khi gọi đến nó.
  - Ví dụ 1: **prototype scope**.
  - Ví dụ 2: Sử dụng **@Lazy**.

# Cơ chế Lazy

## Lazy Loading of Spring Beans

This bean is loaded when Spring container starts up, typically at:

```
ApplicationContext context =  
new AnnotationConfigApplicationContext(  
    Config.class);
```

This bean is loaded when used first time, typically at:

```
ALazyBean theBean =  
context.getBean(ALazyBean.class);
```

Or injecting this bean into another non-lazy bean.

```
@Configuration  
public class Config{  
  
    @Bean  
    public bean1() {  
        return new AnEagerBean();  
    }  
  
    @Bean  
    @Lazy  
    public bean2() {  
        return new ALazyBean();  
    }  
    .....  
}
```

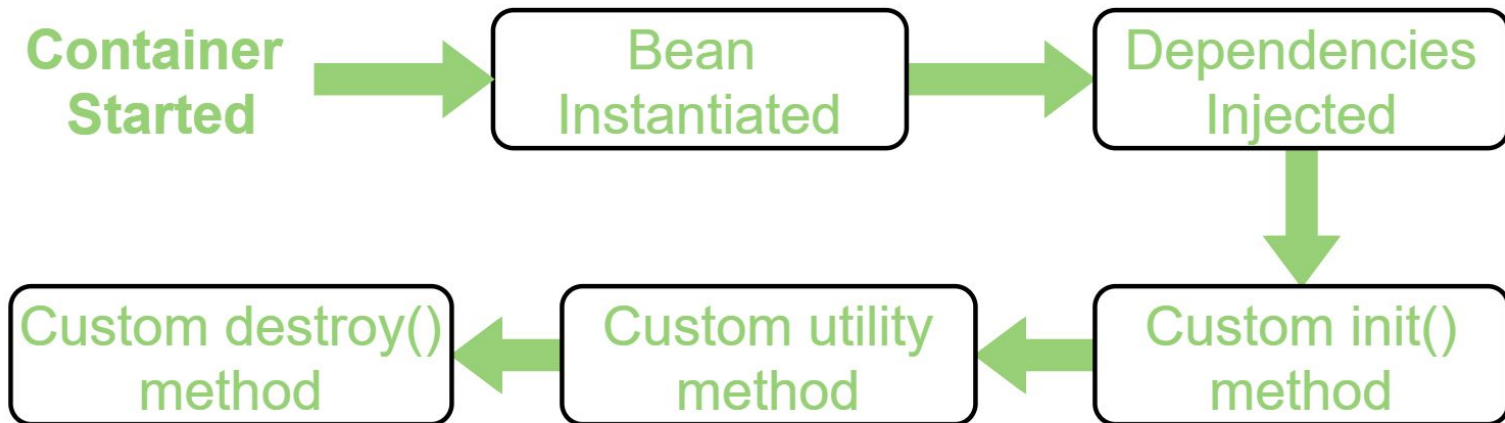


# Vòng đời của Bean

# Vòng đời của Bean

- **@PostConstruct** và **@PreDestroy**:
  - Là 1 Annotation đánh dấu trên 1 method bên trong 1 Bean.
- **@PostConstruct**:
  - Spring IoC Container hoặc ApplicationContext sẽ gọi method này sau khi Bean đó được tạo ra và quản lý.
- **@PreDestroy**:
  - Spring IoC Container hoặc ApplicationContext sẽ gọi method này trước hi Bean đó bị xoá hoặc không được quản lý nữa.

# Vòng đời của Bean



# ROAD TO KOREA

Nếu có bất kỳ thắc mắc nào, hãy đặt câu hỏi qua

**[mail@mail.com](mailto:mail@mail.com) hoặc Zalo 0xxx xxx xxx**