**Cài đặt**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<packaging>pom</packaging>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.0.5.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<groupId>me.loda.spring</groupId>

<artifactId>spring-boot-learning</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-boot-learning</name>

<description>Everything about Spring Boot</description>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<!--spring mvc, rest-->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

Cấu trúc thư mục:



[**#**](https://loda.me/spring-boot-2-autowired-primary-qualifier-loda1557561089057#c%C3%A1ch-inject-bean-c%E1%BB%A7a-spring)**Cách inject Bean của Spring**

@Autowired đánh dấu cho Spring biết rằng sẽ tự động inject bean tương ứng vào vị trí được đánh dấu.

@Component

public class Girl {

// Đánh dấu để Spring inject một đối tượng Outfit vào đây

@Autowired

Outfit outfit;

// public Girl(Outfit outfit) {

// this.outfit = outfit;

// }

// GET

// SET

}

Sau khi tìm thấy một class đánh dấu @Component. thì quá trình inject Bean xảy ra theo cách như sau:

1. Nếu Class không có hàm Constructor hay Setter. Thì sẽ sử dụng [Java Reflection](https://loda.me/articles/huong-dan-java-reflection) để đưa đối tượng vào thuộc tính có đánh dấu @Autowired.
2. Nếu có hàm Constructor thì sẽ inject Bean vào bởi tham số của hàm
3. Nếu có hàm Setter thì sẽ inject Bean vào bởi tham số của hàm

Như ví dụ ở trên tôi đã sử dụng cách Java Reflection để inject Bean vào class Girl. Nếu không sử dụng @Autowired thì bạn phải có một Constructor thay thế, hoặc một Setter tương ứng.

@Component

public class Girl {

// Đánh dấu để Spring inject một đối tượng Outfit vào đây

@Autowired

Outfit outfit;

// Spring sẽ inject outfit thông qua Constructor trước

public Girl() { }

// Nếu không tìm thấy Constructor thoả mãn, nó sẽ thông qua setter

public void setOutfit(Outfit outfit) {

this.outfit = outfit;

}

// GET

// SET

}

Bạn cũng có thể gắn @Autowired lên trên method, thay vì thuộc tính, chức năng cũng vẫn tương tự, nó sẽ tìm Bean phù hợp với method đó và truyền vào.

@Component

public class Girl {

// Đánh dấu để Spring inject một đối tượng Outfit vào đây

Outfit outfit;

// Spring sẽ inject outfit thông qua Constructor trước

public Girl() { }

@Autowired

// Nếu không tìm thấy Constructor thoả mãn, nó sẽ thông qua setter

public void setOutfit(Outfit outfit) {

this.outfit = outfit;

}

// GET

// SET

}

**Vấn đề của @Autowired**

Trong thực tế, sẽ có trường hợp chúng ta sử dụng @Autowired khi **Spring Boot** có chứa 2 Bean cùng loại trong Context.

Lúc này thì **Spring** sẽ bối rối và không biết sử dụng Bean nào để inject vào đối tượng.

Ví dụ:

Class Outfit có 2 kế thừa là Bikini và Naked

import org.springframework.stereotype.Component;

public interface Outfit {

public void wear();

}

/\*

Đánh dấu class bằng @Component

Class này sẽ được Spring Boot hiểu là một Bean (hoặc dependency)

Và sẽ được Spring Boot quản lý

\*/

@Component

public class Bikini implements Outfit {

@Override

public void wear() {

System.out.println("Mặc bikini");

}

}

@Component

public class Naked implements Outfit {

@Override

public void wear() {

System.out.println("Đang không mặc gì");

}

}

Class Girl yêu cầu inject một Outfit vào cho mình.

@Component

public class Girl {

@Autowired

Outfit outfit;

// GET

// SET

}

Lúc này khi chạy chương trình. **Spring Boot** sẽ báo lỗi như sau.

Output:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

APPLICATION FAILED TO START

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Description:

Parameter 0 of constructor in me.loda.spring.helloprimaryqualifier.Girl required a single bean, but 2 were found:

- bikini: defined in file [/Users/lv00141/Documents/WORKING\_SPACE/GITHUB/spring-boot-learning/spring-boot-helloworld-@Primary - @Qualifier/target/classes/me/loda/spring/helloprimaryqualifier/Bikini.class]

- naked: defined in file [/Users/lv00141/Documents/WORKING\_SPACE/GITHUB/spring-boot-learning/spring-boot-helloworld-@Primary - @Qualifier/target/classes/me/loda/spring/helloprimaryqualifier/Naked.class]

Đại khái là, trong quá trình cài đặt, nó tìm thấy tới 2 đối tượng thoả mãn Outfit. Giờ nó không biết sử dụng cái nào để inject vào trong Girl

**@Primary**

Cách giải quyết thứ nhất là sử dụng Annotation @Primary.

@Primary là annotation đánh dấu trên một Bean, giúp nó **luôn được ưu tiên lựa** chọn trong trường hợp có nhiều Bean cùng loại trong Context.

Trong ví dụ ở trên, nếu chúng ta để Naked là primary. Thì chương trình sẽ chạy bình thường.

Và hiển nhiên Outfit bên trong Girl sẽ là Naked.

@Component

@Primary

public class Naked implements Outfit {

@Override

public void wear() {

System.out.println("Đang không mặc gì");

}

}

Chạy thử chương trình:

@SpringBootApplication

public class App {

public static void main(String[] args) {

// ApplicationContext chính là container, chứa toàn bộ các Bean

ApplicationContext context = SpringApplication.run(App.class, args);

// Khi chạy xong, lúc này context sẽ chứa các Bean có đánh

// dấu @Component.

Girl girl = context.getBean(Girl.class);

System.out.println("Girl Instance: " + girl);

System.out.println("Girl Outfit: " + girl.outfit);

girl.outfit.wear();

}

}

Output:

Girl Instance: me.loda.spring.helloprimaryqualifier.Girl@eb9a089

Girl Outfit: me.loda.spring.helloprimaryqualifier.Naked@1688653c

Đang không mặc gì

**Spring Boot** đã ưu tiên Naked và inject nó vào Girl.

**@Qualifier**

Cách thứ hai, là sử dụng Annotation @Qualifier.

@Qualifier xác định tên của một Bean mà bạn muốn chỉ định inject.

Ví dụ:

@Component("bikini")

public class Bikini implements Outfit {

@Override

public void wear() {

System.out.println("Mặc bikini");

}

}

@Component("naked")

public class Naked implements Outfit {

@Override

public void wear() {

System.out.println("Đang không mặc gì");

}

}

@Component

public class Girl {

Outfit outfit;

// Đánh dấu để Spring inject một đối tượng Outfit vào đây

public Girl(@Qualifier("naked") Outfit outfit) {

// Spring sẽ inject outfit thông qua Constructor đầu tiên

// Ngoài ra, nó sẽ tìm Bean có @Qualifier("naked") trong context để ịnject

this.outfit = outfit;

}

// GET

// SET

}

**Kết**

@Primary và @Qualifier là một trong những tính năng bạn nên biết trong Spring để có thể xử lý vấn đề nhiều Bean cùng loại trong một Project.