**SPRING IN ACTION FOURTH EDITION**

**Part 1 Core Spring**

1 Spring into action

* 1. **Simplifying Java development**
  2. **Containing your bean**
  3. Surveying the Spring landscape
  4. What’s new in Spring
  5. Sumary

1. Wiring beans
   1. Exploring Spring’s configuration options

Có 3 sự lựa chọn để wiring beans in your project

* Explicit configuration in XML ( cấu hình tường minh trong file XML )
* Explicit configuration in Java ( Cấu tường minh bằng Java)
* Implicit bean discovery and automatic wiring ( để các bean tự động liên kết )
  1. **Automatically wiring beans**
* Có @componentScan trong file config
* Có @Component truoc moi class
* Có @Autowire trong class test, hoặcmain de Spring tu dong tim trong Application context
  1. **Wiring beans with Java**
* Không can co @Component
* Các bean khai báo trong class Config với @Bean
* Có @Autowire

@contextConfiguration chỉ dùng cho lớp test, Với hàm main trong java phải khai báo tường minh ApplicationContext context = **new** AnnotationConfigApplicationContext(

* AppJavaBasedConfig.**class**);
  1. **Wiring beans with XML**
  2. Importing and **mixing** configurations
  3. **Sumary**

|  |  |  |  |
| --- | --- | --- | --- |
| **Anotation Name** | **Description** | **Type** | **Example** |
| @Componment | Xác định một Bean được tạo ra cho một Class và được quản lý trong Application Context | Automatically | @Component  public class SgtPeppers implements CompactDisc {} |
| @Configuration | Chú thích cho một lớp cấu hình | Automatically, java, Mix | @Configuration  @ComponentScan  public class CDPlayerConfig {  } |
| @ComponmentScan | Quét và phát hiện các Componment trong Spring | Automatically |
| @RunWith | Dùng trong lớp Test, và truyền vào class SpringJUnit4ClassRunner của Spring. | Automatically, java | @RunWith(SpringJUnit4ClassRunner.class)  @ContextConfiguration(classes=CDPlayerConfig.class)  public class CDPlayerTest {  } |
| @ContextConfiguration(classes=CDPlayerConfig.class) | Dùng trong lớp Test, truyền vào lớp cấu hình đã khai báo. | Automatically, java |
| @Test | Chú thích trước một hàm test | Automatically, java, xml | @Test  public void cdShouldNotBeNull() {  assertNotNull(cd);  } |
| @Named | Giống @Componment, nhưng thuộc Java Dependency Injection | Automatically | @Named  public class CDPlayer {  ...  @Inject  public CDPlayer(CompactDisc cd) {  this.cd = cd;  }  } |
| @Inject | Giống @Autowired, nhưng thuộc Java Dependency Injection | Automatically |
| @Autowired | Dùng để tự động tìm một bean trong Application Context để đáp ứng phụ thuộc của một Bean ở thời điểm tạo. Được sử dụng không giới hạn. | Automatically, XML | @Autowired  public CDPlayer(CompactDisc cd) {  this.cd = cd;  } |
| @Autowired(  required=false) | Dùng để tránh những lỗi ngoại lệ khi Spring chưa match được các Bean. | Automatically | @Autowired(required=false)  public CDPlayer(CompactDisc cd) {  this.cd = cd;  } |
| @Rule | Khai báo trong body của lớp Test | Automatically | @Rule public final StandardOutputStreamLog log =  new StandardOutputStreamLog(); |
|  |  |  |  |
| @Bean | Nói cho Spring biết một hàm sẽ trả về một đối tượng được hiểu như là một Bean trong Spring application context | Java, Mix | @Bean  public CompactDisc sgtPeppers() {  return new SgtPeppers();  } |
| <bean id=" " class="">  <constructor-arg ref=" " />  </bean> | Khai báo một Bean có một ID trong file cấu hình XML, với tham số truyền vào nằm trong thành phần <constructor-arg ref=" "/> | XML, Mix | <bean id="cdPlayer" class="soundsystem.CDPlayer">  <constructor-arg ref="compactDisc" />  </bean> |
| <bean id="" class=""  c:cd-ref=" " /> | Khai báo kiểu c-namespace introduced in Spring 3.0 | XML | <bean id="cdPlayer" class= "soundsystem.CDPlayer"  c:cd-ref="compactDisc" /> |
| <bean id=" "  class=" ">  <constructor-arg value="" />  </bean> | Khai báo một Bean nhưng Bean đó không có Tham số phụ thuộc , chỉ có các Value. | XML | <bean id="compactDisc"  class="soundsystem.BlankDisc">  <constructor-arg value="Sgt. Pepper" />  <constructor-arg value="The Beatles" />  </bean> |
| <bean id=""  class=""  c:\_0 =""  c:\_1=”" /> | Khai báo một Bean nhưng Bean đó không có Tham số phụ thuộc , chỉ có các Value. | XML, Mix | <bean id="compactDisc"  class="soundsystem.BlankDisc"  c:\_title="Sgt. Pepper"  c:\_artist="The Beatles" /> |
| <list> | Là một thành phần con của <constructor-arg> và chỉ ra rằng một danh sách các thuộc tính được truyền vào trong hàm dựng. | XML | <bean id="" class="">  <constructor-arg value="Sgt. Pepper" />  <constructor-arg value="The Beatles" />  <constructor-arg>  <list>  <value>Sgt. Pepper's Lonely</value>  <value>With a Little Help</value>  <value>……………</value>  </list>  </constructor-arg>  </bean> |
| <set> | Sử dụng giống <List> |  |  |
|  |  |  |  |
| <property name=" " ref=" " />  </bean> | Chú thích @Autopwired ở phụ thuộc của một lớp (PROPERTY INJECTION ) | XML | <bean id="cdPlayer"  class="soundsystem.CDPlayer">  <property name="compactDisc" ref="compactDisc" />  </bean> |
| <bean id=""  class=" "  p:name-ref=" " /> | Sử dụng p-namespace tương tự <property name=”” ref=””/> | XML | <bean id="cdPlayer"  class="soundsystem.CDPlayer"  p:compactDisc-ref="compactDisc" /> |
| <property name=" " value=" " />  </bean> | Khai báo các Values của một lớp trong file cấu hình XML | XML | <bean id="compactDisc"  class="soundsystem.BlankDisc">  <property name="title" value="Pepper/>  <property name="art"value="Beatles" />  <property name="tracks">  <list>  <value>Sgt. Pepper's Lonely </value>  <value>With a Little Help</value>  </list>  </property>  </bean> |
| @Import(CDConfig.class) | Chú thích khai báo lớp Config | Mix (Referencing XML configuration in JavaConfig) | @Configuration  @Import(CDConfig.class)  public class CDPlayerConfig {  @Bean  public CDPlayer cdPlayer(CompactDisc compactDisc) {  return new CDPlayer(compactDisc);  }  } |
| @ImportResource("  classpath:cd-config.xml") | Chú thích khai báo file cấu hình XMl trong lớp config | Mix (Referencing XML configuration in JavaConfig) |  |
| <importresource="" /> |  | Mix (Referencing JavaConfig in XML configuration) | <?xml version="1.0" encoding="UTF-8"?>  <beans …  <import resource="cd-config.xml" />  <bean id="cdPlayer"  class="soundsystem.CDPlayer"  c:cd-ref="compactDisc" />  </beans> |
| <bean class = “” />  <import resource = “ConfigFile.xml” /> | Khai báo trong file config sử dụng trong cách Mix | Mix (Referencing JavaConfig in XML configuration) | <bean class="soundsystem.CDConfig" />  <import resource="cdplayer-config.xml" /> |

**3 Advanced wiring**

**3.3 Addressing ambiguity in autowiring**

Autowiring là một sự trợ giúp rất lớn bởi vì nó giảm khai báo tường minh cần thiết trong các thành phần ứng dụng. Nhưng autowiring chỉ làm việc chính xác khi mà một bean được liên kết với kết quả mong muốn.Khi mà có nhiều hơn một bean được matching lúc này sẽ có sự mập mờ trong sự liên kết giữa các bean, sẽ ngăn cản Spring tự động autowiring các thuộc tính, tham số hàm dựng, hoặc tham số truyền vào hàm.

**3.3.1** Disignating a primary bean ( chỉ định một Bean ưu tiên đầu tiên)

Khi khai báo bean, bạn có thể tránh liên kết mập mờ bởi chỉ định một bean ưu tiên đầu tiên. Nếu xảy ra sự mập mờ thì Spring sẽ chọn một bean ưu tiên. Essentially(bản chất) là bạn đang khai báo bean mà bạn mong muốn.

Sử dụng ***@Primary*** để nhấn mạnh sự ưu tiên cho Bean mà bạn mong muốn.

+ Sử dụng ***@Primary*** cùng với ***@Componment*** cho bean mà được Scan

@Component

@Primary

public class IceCream implements Dessert { ... }

+ Sử dụng ***@Primary*** cùng với ***@Bean*** cho khai báo cấu hình java

@Bean

@Primary

public Dessert iceCream() {

return new IceCream();

}

+ Cấu hình trong XML:

<bean id="iceCream" class="com.desserteater.IceCream" primary="true" />

* Nếu chỉ định một Bean khác có chú thích @Primary thì Spring không thể chọn được Bean ưu tiên.

**3.3.2 Qualifying autowired beans**

Spring’s qualifers áp dụng một hoạt động thu hẹp tất cả các bean để cuối cùng chọn ra một bean đoen phù hợp với yêu cầu.

**@Qualifer** là cách chính để làm việc với qualifers

Ví dụ để đảm bảo bean IceCream được tiêm tới hàm setDessert ta làm như sau:

@Autowired

@Qualifier("iceCream")

public void setDessert(Dessert dessert) {

this.dessert = dessert;

}

Tham số được gửi tới @Qualifier là ID của bean mà bạn muốn Tiêm vào.

Bean iceCream được tiêm vào khi mà lớp IceCream đã được quét. Mọi sự thay đổi đối với class name sẽ dẫn đến qualifier làm việc không hiệu quả.

**CREATING CUSTOM QUALIFIERS**

**Thay vì dựa vào ID bean như vòng loại, banjc so thể gán your own qualifier to bean.**

@Component

@Qualifier("cold")

public class IceCream implements Dessert { ... }

Sau đó có thể thoải mái inject “cold” mà không lo về sự liên kết giữa các bean.

@Autowired

@Qualifier("cold")

public void setDessert(Dessert dessert) {

this.dessert = dessert;

}

Nếu khai báo theo Java config thì như sau:

@Bean

@Qualifier("cold")

public Dessert iceCream() {

return new IceCream();

}

**DEFINING CUSTOM QUALIFIER ANNOTATIONS**

**//khai báo một chú thích @Cold**

@Target({ElementType.CONSTRUCTOR, ElementType.FIELD,

ElementType.METHOD, ElementType.TYPE})

@Retention(RetentionPolicy.RUNTIME)

@Qualifier

public @interface Cold { }

**//khai báo chú thích vòng 2 @Creamy cho IceCream nếu @Cold bị trùng**

@Target({ElementType.CONSTRUCTOR, ElementType.FIELD,

ElementType.METHOD, ElementType.TYPE})

@Retention(RetentionPolicy.RUNTIME)

@Qualifier

public @interface Creamy { }

//class IceCream

@Component

@Cold

@Creamy

public class IceCream implements Dessert { ... }

//class Popsicle

@Component

@Cold

@Fruity

public class Popsicle implements Dessert { ... }

/.Inject Bean

@Autowired

@Cold

@Creamy

public void setDessert(Dessert dessert) {

this.dessert = dessert;

}

**Singleton:** 1 doi tuong chi co mot the hien duy nhat trong application, co the tao nhieu lan.