

Spring注解驱动开发第21讲——你还不会使用@Resource注解和@Inject注解吗？那你就out了！！

写在前面

在前一讲中，我介绍了如何使用@Autowired、@Qualifier 以及@Primary这三个注解自动装配Spring组件。那除了这三个注解以外，还有没有其他的注解可以自动装配组件呢？那必须有啊！今天，我们就一起来说说@Resource注解和@Inject注解。

简单介绍一下@Resource和@Inject 这俩注解

@Resource注解

@Resource注解是Java规范里面的，也可以说它是JSR250规范里面定义的一个注解。该注解默认按照名称进行装配，名称可以通过name属性进行指定，如果没有指定name属性，当注解写在字段上时，那么默认取字段名将其作为组件的名称在IOC容器中进行查找，如果注解写在setter方法上，那么默认取属性名进行装配。当找不到与名称匹配的bean时才按照类型进行装配。但是需要注意的一点是，如果name属性一旦指定，那么就只会按照名称进行装配。

我们先看一下@Resource注解的源码，如下所示。

```
* Copyright (c) 2005, 2011, Oracle and/or its affiliates. All rights reserved.

package javax.annotation;

import java.lang.annotation.*;
import static java.lang.annotation.ElementType.*;
import static java.lang.annotation.RetentionPolicy.*;

* The Resource annotation marks a resource that is needed.
@Target({TYPE, FIELD, METHOD})
@Retention(RUNTIME)
public @interface Resource {
    * The JNDI name of the resource. For field annotations,
    String name() default "";

    * The name of the resource that the reference points to. It can

    String lookup() default "";

    * The Java type of the resource. For field annotations,
    Class<?> type() default java.lang.Object.class;

    * The two possible authentication types for a resource.
    enum AuthenticationType {
        CONTAINER,
        APPLICATION
    }

    * The authentication type to use for this resource.
    AuthenticationType authenticationType() default AuthenticationType.CONTAINER;

    * Indicates whether this resource can be shared between
    boolean shareable() default true;

    * A product specific name that this resource should be mapped to.
    String mappedName() default "";

    * Description of this resource. The description is expected
    String description() default "";
}
```

https://blog.csdn.net/yerenyuan_pku

@Inject注解

@Inject注解也是Java规范里面的，也可以说它是JSR330规范里面定义的一个注解。该注解默认是根据参数名去寻找bean注入，支持Spring的@Primary注解优先注入，@Inject注解还可以增加@Named注解指定要注入的bean。

我们先看一下@Inject注解的源码，如下所示。

```
Resource.class Inject.class
2+ * Copyright (C) 2009 The JSR-330 Expert Group
16
17 package javax.inject;
18
19 import java.lang.annotation.Target;
20 import java.lang.annotation.Retention;
21 import java.lang.annotation.Documented;
22 import static java.lang.annotation.RetentionPolicy.RUNTIME;
23 import static java.lang.annotation.ElementType.METHOD;
24 import static java.lang.annotation.ElementType.CONSTRUCTOR;
25 import static java.lang.annotation.ElementType.FIELD;
26
28+ * Identifies injectable constructors, methods, and fields. May apply to static
182 @Target({ METHOD, CONSTRUCTOR, FIELD })
183 @Retention(RUNTIME)
184 @Documented
185 public @interface Inject {}
186
```

温馨提示，要想使用@Inject注解，需要在项目的pom.xml文件中添加如下依赖，即导入javax.inject这个包。

```
1 <dependency>
2   <groupId>javax.inject</groupId>
3   <artifactId>javax.inject</artifactId>
4   <version>1</version>
5 </dependency>
AI写代码xml
```

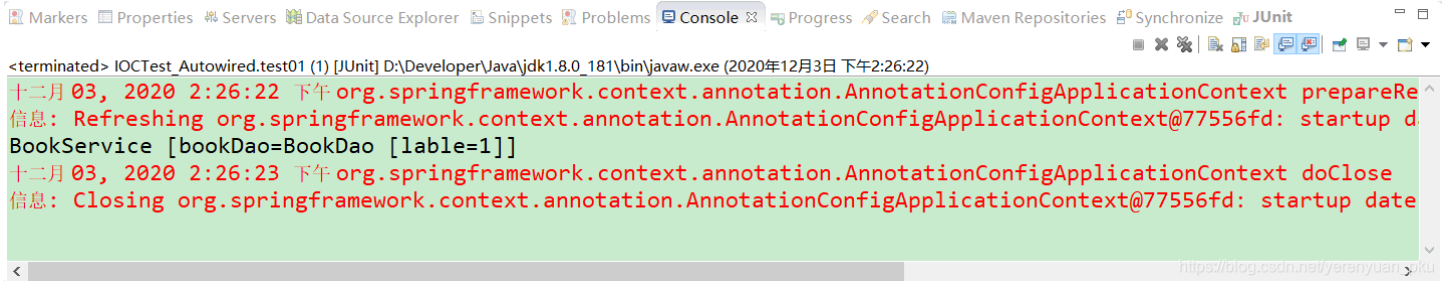
实战案例

测试@Resource注解

首先，我们将项目中的BookService类标注在bookDao字段上的@Autowired注解和@Qualifier注解注释掉，然后添加上@Resource注解，如下所示。

```
1 package com.meimeixia.service;
2
3 import javax.annotation.Resource;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.beans.factory.annotation.Qualifier;
7 import org.springframework.stereotype.Service;
8
9 import com.meimeixia.dao.BookDao;
10
11 @Service
12 public class BookService {
13
14 // @Qualifier("bookDao") // 要让首选装配起效果，@Qualifier自然就不能用了
15 // @Autowired(required=false)
16 @Resource
17 private BookDao bookDao;
18
19 public void print() {
20     System.out.println(bookDao);
21 }
22
23 @Override
24 public String toString() {
25     return "BookService [bookDao=" + bookDao + "];
26 }
27 }
28 }
AI写代码java运行
```

然后，我们运行一下IOCTest_Autowired类中的test01()方法，输出的结果信息如下所示。



可以看到，使用@Resource注解也能够 **自动装配** 组件，只不过此时自动装配的是lable为1的bookDao，而不是我们在MainConfigOfAutowired配置类中配置的优先装配的lable为2的bookDao。MainConfigOfAutowired配置类中配置的lable为2的bookDao如下所示。

```

1 package com.meimeixia.config;
2
3 import org.springframework.context.annotation.Bean;
4 import org.springframework.context.annotation.ComponentScan;
5 import org.springframework.context.annotation.Configuration;
6 import org.springframework.context.annotation.Primary;
7
8 import com.meimeixia.dao.BookDao;
9
10 /**
11  *
12  * @author Liayun
13  *
14  */
15 @Configuration
16 @ComponentScan({"com.meimeixia.service", "com.meimeixia.dao", "com.meimeixia.controller"})
17 public class MainConfigOfAutowired {
18
19     @Primary
20     @Bean("bookDao2")
21     public BookDao bookDao() {
22         BookDao bookDao = new BookDao();
23         bookDao.setLable("2");
24         return bookDao;
25     }
26
27 }

```

AI写代码java运行



这也进一步说明，@Resource注解和@Autowired注解的功能是一样的，都能实现自动装配，只不过@Resource注解默认是按照组件名称（即属性的名称）进行装配的。虽然@Resource注解具备自动装配这一功能，但是它是不支持@Primary注解优先注入的功能的，而且也不能像@Autowired注解一样能添加 **required=false** 属性。

我们在使用@Resource注解时，可以通过@Resource注解的name属性显示指定要装配的组件的名称。例如，我们要想装配lable为2的bookDao，只需要为@Resource注解添加 **name="bookDao2"** 属性即可，如下所示。

```

1 package com.meimeixia.service;
2
3 import javax.annotation.Resource;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.beans.factory.annotation.Qualifier;
7 import org.springframework.stereotype.Service;
8
9 import com.meimeixia.dao.BookDao;
10
11 @Service
12 public class BookService {
13
14     // @Qualifier("bookDao") // 要让首选装配起效果，@Qualifier自然就不能用了
15     // @Autowired(required=false)
16     @Resource(name="bookDao2")
17     private BookDao bookDao;
18
19     public void print() {
20         System.out.println(bookDao);
21     }
22
23     @Override
24     public String toString() {
25
26

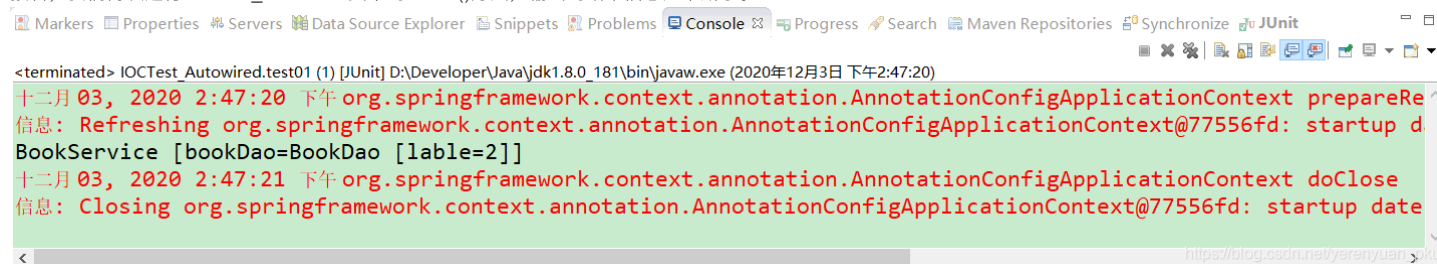
```

```
25         return "BookService [bookDao=" + bookDao + "];"
26     }
27 }
28 }
```

AI写代码java运行



接着，我们再次运行IOCTest_Autowired类中的test01()方法，输出的结果信息如下所示。



```
<terminated> IOCTest_Autowired.test01 (1) [JUnit] D:\Developer\Java\jdk1.8.0_181\bin\javaw.exe (2020年12月3日 下午2:47:20)
十二月 03, 2020 2:47:20 下午 org.springframework.context.annotation.AnnotationConfigApplicationContext prepareRe
信息: Refreshing org.springframework.context.annotation.AnnotationConfigApplicationContext@77556fd: startup d
BookService [bookDao=BookDao [lable=2]]
十二月 03, 2020 2:47:21 下午 org.springframework.context.annotation.AnnotationConfigApplicationContext doClose
信息: Closing org.springframework.context.annotation.AnnotationConfigApplicationContext@77556fd: startup date
```

可以看到，此时输出了lable为2的bookDao，说明@Resource注解可以通过name属性显示指定要装配的bean。

测试@Inject注解

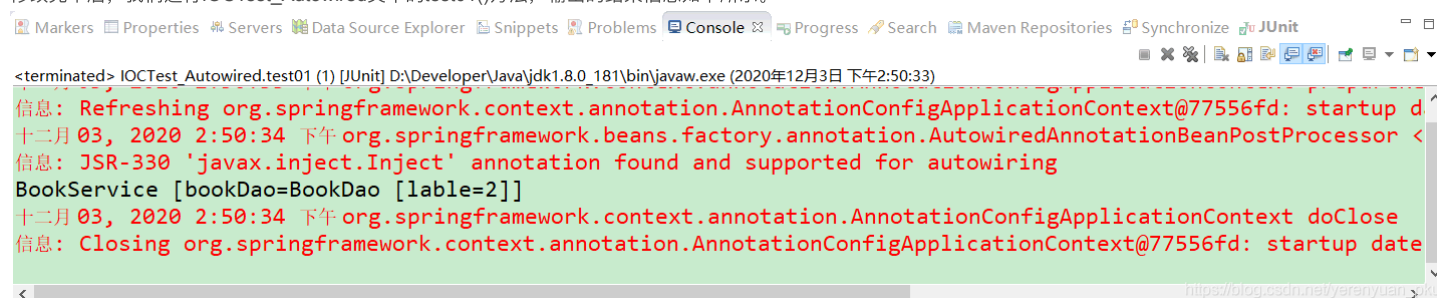
在BookService类中，将@Resource注解注释掉，添加@Inject注解，如下所示。

```
1 package com.meimeixia.service;
2
3 import javax.annotation.Resource;
4 import javax.inject.Inject;
5
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.beans.factory.annotation.Qualifier;
8 import org.springframework.stereotype.Service;
9
10 import com.meimeixia.dao.BookDao;
11
12 @Service
13 public class BookService {
14
15     // @Qualifier("bookDao") // 要让首选装配起效果，@Qualifier自然就不能用了
16     // @Autowired(required=false)
17     // @Resource(name="bookDao2")
18     @Inject
19     private BookDao bookDao;
20
21     public void print() {
22         System.out.println(bookDao);
23     }
24
25     @Override
26     public String toString() {
27         return "BookService [bookDao=" + bookDao + "];"
28     }
29 }
30 }
```

AI写代码java运行



修改完毕后，我们运行IOCTest_Autowired类中的test01()方法，输出的结果信息如下所示。



```
<terminated> IOCTest_Autowired.test01 (1) [JUnit] D:\Developer\Java\jdk1.8.0_181\bin\javaw.exe (2020年12月3日 下午2:50:33)
信息: Refreshing org.springframework.context.annotation.AnnotationConfigApplicationContext@77556fd: startup d
十二月 03, 2020 2:50:34 下午 org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor <
信息: JSR-330 'javax.inject.Inject' annotation found and supported for autowiring
BookService [bookDao=BookDao [lable=2]]
十二月 03, 2020 2:50:34 下午 org.springframework.context.annotation.AnnotationConfigApplicationContext doClose
信息: Closing org.springframework.context.annotation.AnnotationConfigApplicationContext@77556fd: startup date
```

可以看到，使用@Inject注解默认输出的是lable为2的bookDao。这是因为@Inject注解和@Autowired注解一样，默认优先装配使用了@Primary注解标注的组件。

其实，这也进一步说明了，**@Inject注解和@Autowired注解的功能是一样的**，都能实现自动装配，而且它俩都支持**@Primary注解优先注入的功能**。只不过，**@Inject注解不能像@Autowired注解一样能添加 `required=false` 属性**，因为它里面没啥属性。

@Resource和@Inject这俩注解与@Autowired注解的区别

不同点

- @Autowired是Spring中的专有注解，而@Resource是Java中JSR250规范里面定义的一个注解，@Inject是Java中JSR330规范里面定义的一个注解
- @Autowired支持参数required=false，而@Resource和@Inject都不支持
- @Autowired和@Inject支持@Primary注解优先注入，而@Resource不支持
- @Autowired通过@Qualifier指定注入特定bean，@Resource可以通过参数name指定注入bean，而@Inject需要通过@Named注解指定注入bean

相同点

三种注解都可以实现bean的自动装配。