链滴

Spring IOC 的核心机制：实例化与注入

作者：zyjImmortal
原文链接：https://ld246.com/article/1588579543438
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

上文我们介绍了IOC和DI，IOC是一种设计模式，DI是它的具体实现，有很多的框架都有这样的实现本文主要以spring框架的实现，来看具体的注入实现逻辑。

spring是如何将对象加入容器的

spring将对象加入容器的方式有很多种，最主要的是xml和注解的形式，而当下注解的形式应用更加广泛，所以这里我们也主要介绍注解注入模式下的相关知识点。

spring下的注解也是有很多种的，其中应用最为广泛的就是模式注解。

模式注解 stereotype annotations

咋一看，这是啥，说实话我也不太清楚，官方说的就叫这哈，其实他说的就是平时最常用的那些 @Component @Service @Controller @Repository @Configuration 注解。

如果一个类被打上 @Component注解，那他就会被容器扫描到并加入到容器里去。那这里我们用spring的方式再来实现一下上篇文章的获取英雄的需求。

先给 Diana这个英雄打上一个 @Component注解：

```
@Component
public class Diana{
    private String skillName = "Diana R";

    public Diana() {
    }

    public void q(){
        System.out.println("Diana Q");
    }
```

```
    public void w(){
        System.out.println("Diana W");
    }

    public void e(){
        System.out.println("Diana E");
    }

    public void r(){
        System.out.println(this.skillName);
    }
}
```

假设现在有一个Controller需要用到 Diana

```
@RestController
public class BannerController {

    @Autowired
    private Diana diana;

    @RequestMapping(value = "/v2/banner", method = {RequestMethod.GET})
    public String test() {
        diana.r();
        return "Hello SpringBoot";
    }
}
```

@Autowired的意思就是说在这里注入一个对象，然后我们启动应用，

通过浏览器访问 http://localhost:8081/v2/banner，会发现在控制台会输出 Diana的r方法的打印内，说明这里确实拿到了 Diana这个对象。

对象的注入时机

如果我们把 Diana的 @Component注解去掉，其他地方不变，再运行程序会发生什么，试试看。

重新启动的时候会提示我们下面的错误：

```
***************************
APPLICATION FAILED TO START
***************************

Description:

Field diana in com.zyj.missyou.api.v2.BannerController required a bean of type 'com.zyj.missyou.api.v2.Diana' that could not be found.

The injection point has the following annotations:
    - @org.springframework.beans.factory.annotation.Autowired(required=true)
```

看错误提示，说程序没有找到对应的bean，就是说我这里要注入bean了，但是在容器里没有找到。

这里就会得出一个结论：**容器在初始化后就会给相应的代码片段进行对象的注入。**

从提示信息中可以看到Autowired(required=true)，这里设置的默认是true，表示注入的时候，该对

必须存在，否则就会注入失败。当然这里可设置不报错，Autowired(required=false),这个时候再启就不会报错了，但是在访问的时候会报空指针异常。

那这里spring的容器究竟是在什么时候实例化的对象呢,是在访问controller的时候呢，还是在容器启的时候呢？我们可以通过下面的手段检测一下，在 Diana的无参构造方法中放一个打印语句，然后再动程序：

```
public Diana() {
    System.out.println("I am Diana");
}
```

程序启动以后，会看到在控制台会显示出构造方法中打印的语句

```
2020-05-04 14:36:09.636  INFO 96760 --- [  restartedMain] com.zyj.missyou.MissyouApplication       : Starting MissyouApplication on zhouyajundeMacBook
2020-05-04 14:36:09.638  INFO 96760 --- [  restartedMain] com.zyj.missyou.MissyouApplication       : No active profile set, falling back to default pr
2020-05-04 14:36:09.682  INFO 96760 --- [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.de
2020-05-04 14:36:09.682  INFO 96760 --- [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setti
2020-05-04 14:36:10.486  INFO 96760 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8081 (http)
2020-05-04 14:36:10.495  INFO 96760 --- [  restartedMain] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2020-05-04 14:36:10.495  INFO 96760 --- [  restartedMain] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.29]
2020-05-04 14:36:10.547  INFO 96760 --- [  restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContex
2020-05-04 14:36:10.547  INFO 96760 --- [  restartedMain] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization comple
I am Diana
2020-05-04 14:36:10.675  INFO 96760 --- [  restartedMain] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExec
2020-05-04 14:36:10.792  INFO 96760 --- [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
2020-05-04 14:36:10.832  INFO 96760 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8081 (http) with conte
2020-05-04 14:36:10.835  INFO 96760 --- [  restartedMain] com.zyj.missyou.MissyouApplication       : Started MissyouApplication in 1.487 seconds (JVM
```

这就说明在容器启动后把类加载到容器以后，就会实例化一个对象出来。这就又得出一个结论：**容器始化以后就会对bean进行实例化**

对象的延迟注入

上面提到容器在确定后就会把对象(也就是bean)实例化，那有没有办法让他不马上实例化？

spring提供了@Lazy这个注解，用以表明某个bean可以延迟加载，用之前的 Diana延时一下，给他上@Lazy这个注解：

```
@Component
@Lazy
public class Diana{
    private String skillName = "Diana R";

    public Diana() {
        System.out.println("I am Diana");
    }

    public void q(){
        System.out.println("Diana Q");
    }

    public void w(){
        System.out.println("Diana W");
    }

    public void e(){
        System.out.println("Diana E");
```

```
    }

    public void r(){
        System.out.println(this.skillName);
    }
}
```

再次启动程序，会发现在控制台还是会执行构造方法里的打印语句，这是什么情况，不是明明已经加延迟加载了么？

这里会设计到spring的一个机制，就是说如果某个bean没有设置延迟加载，这个bean会在容器启动实例化，并且这个bean里面所依赖的其他bean都会进行过实例化，即使设置了懒加载。

我们在BannerController里用到了 Diana这个bean，并把它设置了延迟加载，但是并没有把BannerController也设置成延迟加载，所以容器再实例化BannerController的时候同样会 Diana这个bean进行例化。如果要真正做到延迟加载，需要让BannerController也要延迟加载。给它加上@Lazy以后，再动看一下：

```
2020-05-04 14:55:14.792  INFO 97562 --- [  restartedMain] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2020-05-04 14:55:14.792  INFO 97562 --- [  restartedMain] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.29]
2020-05-04 14:55:14.854  INFO 97562 --- [  restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2020-05-04 14:55:14.854  INFO 97562 --- [  restartedMain] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed
2020-05-04 14:55:15.006  INFO 97562 --- [  restartedMain] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecuto
2020-05-04 14:55:15.122  INFO 97562 --- [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
2020-05-04 14:55:15.169  INFO 97562 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8081 (http) with context
2020-05-04 14:55:15.172  INFO 97562 --- [  restartedMain] com.zyj.missyou.MissyouApplication       : Started MissyouApplication in 1.421 seconds (JVM run
2020-05-04 14:56:52.211  INFO 97562 --- [nio-8081-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherSer
2020-05-04 14:56:52.211  INFO 97562 --- [nio-8081-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2020-05-04 14:56:52.217  INFO 97562 --- [nio-8081-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 6 ms
I am Diana
Diana R
```

这个时候控制就没有那条打印语句的输出内容了，此时我们通过浏览器来访问BannerController的路，此时就会在控制台输出构造方法里打印语句的输出内容了。

对象的注入逻辑

上文 Diana的就是一个单独的类，没有实现任何接口，这种实现方式他是有问题的，很难进行扩展，里不应该一类具体的实现类，而是要应该依赖抽象，这样才能满足开闭原则，让程序具有良好的扩展。

这里让 Diana实现了一 Skill接口，

```
public interface Skill {
    void q();
    void w();
    void e();
    void r();
}
```

在BannerController注入这里也要改一下：

```
@Autowired
private Skill diana;
```

运行程序，发先能够运行成功,如果新添加一个英雄的实现类：

```
@Component
public class Irelia implements Skill {

    public Irelia() {
        System.out.println("Hello, Irelia");
    }
    public void q(){
        System.out.println("Irelia Q");
    }

    public void w(){
        System.out.println("Irelia W");
    }

    public void e(){
        System.out.println("Irelia E");
    }

    public void r(){
        System.out.println("Irelia R");
    }
}
```

我们再去运行程序，发现还是能运行成功，并且发现注入的是 Diana而不是 Irelia,那这里我们可能会为什么呀，前面好像也没指定注入哪个实现类。

难道是因为 private Skill diana;这里写了 diana么？你还别说，还真是这个原因，这里涉及到spring注入机制了。

当spring的IOC容器注入bean的时候，如果发现有多个相同类型的bean时，就会去看它们的名称(name)，如果名称符合要求，就会注入这个名称的bean。每个被容器扫描到的bean都会有一个默认的name，就是它的类名，首字母会小写，比如我们实例中的：BannerController的name就是bannerController，Irelia就是irelia。

可以试着把 private Skill diana改成 private Skill skill，再次运行，就会发生报错：

```
***************************
APPLICATION FAILED TO START
***************************

Description:

Field skill in com.zyj.missyou.api.v2.BannerController required a single bean, but 2 were found:
    - diana: defined in file [/Users/zhouyajun/workspace/imooc/sleeve/missyou/target/classes/com/zyj/missyou/api/v2/Diana.class]
    - irelia: defined in file [/Users/zhouyajun/workspace/imooc/sleeve/missyou/target/classes/com/zyj/missyou/api/v2/Irelia.class]


Action:

Consider marking one of the beans as @Primary, updating the consumer to accept multiple beans, or using @Qualifier to identify the bean that should be consumed
```

提示你找不到响应的bean。这里要明确一点哈，虽然报错了，但是这种写法是标准的，上面那个直写实现类的名字不是标准的写法。

那怎么解决这个找不到的问题呢？

spring提供了一个@Qualifier的注解，给他传入要注入bean的名字，就可以指定注入的bean，@Qua

ifier("irelia")。

总结

Autowired的注入方式有两种，一种是按照类型注入，一种是按照名称注入，默认是按照类型注入，果只有一个实现类 Diana，这里写成 private Skill skill，运行程序，你会发现是能够注入成功的。因Skill类型下只有一个实现类，自然就是注入它了。如果有多个实现类，那就会先按照这个类型查找，个类型下所有到的实现类再按照名称查找，直至找到符合要求的，找不到就报错。