15_依赖查找中的经典异常：Bean 找不到？Bean 不是唯一的？

1、开篇

上节课通过源码分析的方式介绍了：获取单个 Bean 类型实例；获取集合 Bean 类型实例；获取集合 Bean 类型名称。本节课，会带给大家依赖查找中的经典异常问题的讨论：Bean 找不到的情况以及 Bean 不是唯一的情况，看看在这种情况下应该如何处理。今天主要介绍 Spring IOC 依赖查找，有如下内容：

- BeansException 的子接口
- NoSuchBeanDefinitionException 不存在要查找的 Bean
- NoUniqueBeanDefinitionException 容器中存在多个同类型的 Bean

2、BeansException 的子接口

如图 1 所示，通常来说在操作 Bean 的时候会出现异常，这些异常都来自于 BeansException 的子接口。例如查找的 Bean 不存在于 IoC 容器中；类型以来查找时 IoC 容器存在多个 Bean 实例；当 Bean 对应的类型非具体类时；当 Bean 初始化过程中出现异常；当 BeanDefinition 配置元信息时出现异常。这些异常都会在不同的场景中出现。

异常类型	触发条件(举例)	场景举例
NoSuchBeanDefinitionException	当查找 Bean 不存在于 IoC 容器时	BeanFactory#getBean ObjectFactory#getObject
NoUniqueBeanDefinitionException	类型依赖查找时，IoC 容器存在多 个 Bean 实例	BeanFactory#getBean(Class)
BeanInstantiationException	当 Bean 所对应的类型非具体类时	BeanFactory#getBean
BeanCreationException	当 Bean 初始化过程中	Bean 初始化方法执行异常 时
BeanDefinitionStoreException	当 BeanDefinition 配置元信息非法 时	XML 配置资源无法打开时

图 1BeansException 的子接口

这里我们着重讨论 NoSuchBeanDefinitionException（Bean 不存在）和 NoUniqueBeanDefinitionException（Bean 不唯一）两种异常的使用场景。

3、NoSuchBeanDefinitionException 不存在要查找的 Bean

如图 2 所示，为了演示 NoSuchBeanDefinitionException 和 NoUniqueBeanDefinitionException 异常的场景。这里建立了 Rumenz 类作为测试类，类中出了定义 id 和 name 的变量和属性以外，还定义了 afterPropertiesSet 方法用来抛出异常。

```
public class Rumenz implements InitializingBean {
    private Integer id;
    private String name;
    @Override
    public void afterPropertiesSet() throws Exception {
        throw new Exception("初始化异常。。。。");
    }
    @Override
    public String toString() {
        return "Rumenz{" +
                "id=" + id +
                ", name='" + name + '\'' +
                '}';
    }
    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

图 2 Rumenz 类定义

有了测试类以后，在 main 函数中对其进行测试。

如图 3 所示，通过 AnnotationConfigApplicationContext 注册 Bean，不过这里没有注册 Rumenz，而是注册了 DemoApplication.class。随后，通过 byBeanFactory 方法获取 Bean 实例，其中使用了 AnnotationConfigApplicationContext 中的 getBean 方法获取 Rumenz.class 的实例，由于之前没有注册该类，因此这里会抛出异常。

```
public class DemoApplication {

    public static void main(String[] args) {
        AnnotationConfigApplicationContext ac=new AnnotationConfigApplicationContext();
        ac.register(DemoApplication.class); //不注册Rumenz
        ac.refresh();

        //NoSuchBeanDefinitionException不存在要查找的Bean
        byBeanFactory(ac);
        ac.close();
    }
     //NoSuchBeanDefinitionException不存在要查找的Bean
    private static void byBeanFactory(AnnotationConfigApplicationContext ac) {
        ac.getBean(Rumenz.class);
    }

}
```

图 3 NoSuchBeanDefinitionException 演示

运行上述代码抛出如下异常：

```
Exception in thread "main" org.springframework.beans.factory.NoSuchBeanDefinitionException:
No qualifying bean of type 'com.rumenz.Rumenz' available
```

3、NoUniqueBeanDefinitionException 容器中存在多个同类型的 Bean

上面讲到了 NoSuchBeanDefinitionException 处理不存在 Bean 的场景，在没有注册 Bean 的时候，依旧通过 getBean 获取实例就会抛出这个异常。接下来演示 NoUniqueBeanDefinitionException 的使用场景，从异常的字面意思上理解如果容器中存在多个同类型的 Bean 时就会抛出这个异常。如图 4 所示，在 Config 中实例化两个 Rumenz，它们除了 id 不同以后，name 和类型都是相同的。

```java
public class Config {
    @Bean
    public Rumenz rumenz1(){
        Rumenz r=new Rumenz();
        r.setId(456);
        r.setName("名字");
        return r;
    }

    @Bean
    public Rumenz rumenz(){
        Rumenz r=new Rumenz();
        r.setId(123);
        r.setName("名字");
        return r;
    }
}
```

图 4 定义两个相同名字的 Bean

然后就是测试代码了，其中依旧通过 AnnoationConfigApplicationContext 注册 Config.class ，之后通过 byBeanFactory 方法是图获取 Rumenz.class。由于在 Config 类中定义了两个类型相同的 Rumenz 实例，因此这个调用会抛出异常。

```
public class DemoApplication {

    public static void main(String[] args) {
        AnnotationConfigApplicationContext ac=new AnnotationConfigApplicationContext();
        ac.register(Config.class); //注册Rumenz
        ac.refresh();

        // NoUniqueBeanDefinitionException要查找的Bean不惟一
        byBeanFactory(ac);
        ac.close();
    }
     // NoUniqueBeanDefinitionException要查找的Bean不惟一
    private static void byBeanFactory(AnnotationConfigApplicationContext ac) {
        ac.getBean(Rumenz.class);
    }

}
```

图 5 NoUniqueBeanDefinitionException 测试代码

抛出的 NoUniqueBeanDefinitionException 异常如下展示：

Exception in thread "main" org.springframework.beans.factory.NoUniqueBeanDefinitionException: No qualifying bean of type 'com.rumenz.Rumenz' available: expected single matching bean but found 2: rumenz1,rumenz

4、总结

本节课介绍了 Spring IOC 查找 Bean 出现的异常，这些异常都是 BeansException 的子接口。其中 NoSuchBeanDefinitionException（Bean 不存在）和 NoUniqueBeanDefinitionException（Bean 不唯一）两种异常的使用场景是最为常见的，我们通过测试代码的方式将两种场景的异常做了介绍。下节课，通过 Spring IoC 容器初始化 Bean 的流程，下期见，拜拜。