Spring Boot中处理同名Bean冲突的解决办法

在Spring Boot项目中，`同名Bean`的问题时有发生，尤其是在接手第三方库或进行项目继承开发时。这种情况会导致Spring容器在创建Bean时产生冲突，进而抛出`ConflictingBeanDefinitionException`异常。本文将介绍几种解决这种同名Bean冲突方案。

方案一：更换类名

在两个类都没有手动设置Bean的名称的时候，最简单的办法就是修改其中一个冲突的类的名字，这样就能避免名称冲突。

方案二：手动设置Bean的名称

手动设置@Bean注解的名称来避免冲突。例如，将原来的`@Bean`注解改为`@Bean(”bean1”)`，这样就可以将Bean的名称修改为`bean1`，避免和自动配置的Bean名称冲突。

方案三：使用`@Primary`注解

`@Primary`注解可以用来指定当存在多个同类型的Bean时，哪个Bean应该被优先考虑。如果不想改变类名也不想手动设置注解中的名字的话，可以采用这种方案。

```
@Service
@Primary
public class CustomAuthCodeServiceImpl implements AuthCodeService {
    @Override
    public boolean check() {
        // 自定义认证逻辑
        return true;
    }
```

}

```

在上述代码中，`@Primary`注解确保了`CustomAuthCodeServiceImpl`会覆盖原有的实现。然而，这种方法有时可能不起作用，特别是当Bean是通过类路径扫描自动配置时。

方案四：自定义`@ComponentScan`排除规则
---

`@ComponentScan`注解可以用来指定Spring在启动时扫描哪些包，并排除某些特定的类。通过自定义排除规则，我们可以阻止Spring创建不需要的Bean。

```
@SpringBootApplication
@ComponentScan(basePackages = ”com.yourcompany”,
               excludeFilters = {@Filter(type =
FilterType.ASSIGNABLE_TYPE, classes = AuthCodeServiceImpl.class)})
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}

```

在上述代码中，我们排除了`AuthCodeServiceImpl`类，这样Spring就不会为其创建Bean。

方案五：自定义`TypeExcludeFilter`
---

如果`@ComponentScan`的排除规则与`@SpringBootApplication`的默认排除规则冲突，我们可以通过自定义`TypeExcludeFilter`来解决问题。

0. 创建自定义的`TypeExcludeFilter`类，继承自`TypeExcludeFilter`并重写`match`方法。
```

```
public class CustomTypeExcludeFilter extends TypeExcludeFilter {
    @Override
    public boolean match(MetadataReader metadataReader,
MetadataReaderFactory metadataReaderFactory) throws IOException {
        String className =
metadataReader.getClassMetadata().getClassName();
        return AuthCodeServiceImpl.class.getName().equals(className);
    }
}

```

2. 实现`ApplicationContextInitializer`接口，将自定义的`TypeExcludeFilter`注册到Spring容器中。

```
public class CustomTypeExcludeFilterApplicationContextInitializer
implements ApplicationContextInitializer {
    @Override
    public void initialize(ConfigurableApplicationContext
applicationContext) {
        BeanFactory beanFactory = applicationContext.getBeanFactory();
        BeanDefinitionBuilder beanDefinitionBuilder =
BeanDefinitionBuilder.genericBeanDefinition(CustomTypeExcludeFilter.cl
ass);
        beanFactory.registerBeanDefinition("customTypeExcludeFilter",
beanDefinitionBuilder.getBeanDefinition());
    }
}

```

3. 在`/META-INF/spring.factories`中创建配置文件，指定自定义的`ApplicationContextInitializer`。

```
org.springframework.context.ApplicationContextInitializer=\
com.yourcompany.context.CustomTypeExcludeFilterApplicationContextIn
itializer

```

通过上述步骤，我们可以有效地排除不需要的Bean，解决同名Bean冲突问题。

总结

以上提供了5中方案来解决Spring Boot中的同名Bean冲突，希望对你有帮助。实际工作中可以根据具体的情况来进行选择。

另外，对GPT4.0有需求的同学，请移步[GPT4.0教程](http://cxyroad.com/”https://7we.cn/subscribe-chatgpt-plus-in-five-minutes/”)。