contextcomponent-scan标签的use-default-filters属性的作用以及原理分析

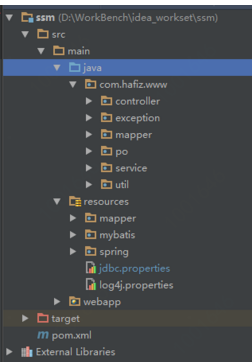
# 背景

　　我们在**Spring+SpringMVC+Mybatis**的集成开发中，经常会遇到**事务配置**不起作用等问题，那么本文就来分析下出现这种问题可能的原因以及解决方式。

SSM

# 分析及原理窥探

## 项目结构



## 我们在spring-mvc.xml文件中进行如下配置，这种方式会成功扫描到带有@Controller注解的Bean，不会扫描带有@Service/@Repository注解的Bean，是正确的。

<context:component-scan base-package="**com.hafiz.www.controller**">

**<context:include-filter type="annotation" expression="org.springframework.stereotype.Controller"/>**

</context:component-scan>

## 但是如下方式，不仅仅扫描到带有@Controller注解的Bean，还扫描到带有@Service/@Repository注解的Bean，可能造成事务不起作用等问题。

<context:component-scan base-package="**com.hafiz.www**">

**<context:include-filter type="annotation" expression="org.springframework.stereotype.Controller"/>**

</context:component-scan>

# 这是因为什么呢？下面让我们来从源码进行分析。

## 1.<context:component-scan>会交给org.springframework.context.config.ContextNamespaceHandler处理.

registerBeanDefinitionParser("component-scan", new ComponentScanBeanDefinitionParser());

## 2.ComponentScanBeanDefinitionParser会读取配置文件信息并组装成org.springframework.context.annotation.ClassPathBeanDefinitionScanner进行处理。

## 3.如果没有配置<context:component-scan>的use-default-filters属性，则默认为true，在创建ClassPathBeanDefinitionScanner时会根据use-default-filters是否为true来调用如下代码：

protected void registerDefaultFilters() {

　　this.includeFilters.add(new AnnotationTypeFilter(Component.class));

　　ClassLoader cl = ClassPathScanningCandidateComponentProvider.class.getClassLoader();

　　try {

　　　　this.includeFilters.add(new AnnotationTypeFilter(

　　　　　　((Class<? extends Annotation>) cl.loadClass("javax.annotation.ManagedBean")), false));

　　　 logger.info("JSR-250 'javax.annotation.ManagedBean' found and supported for component scanning");

　　}

　　catch (ClassNotFoundException ex) {

　　　　// JSR-250 1.1 API (as included in Java EE 6) not available - simply skip.

　　}

　　try {

　　　　this.includeFilters.add(new AnnotationTypeFilter(

　　　　　　((Class<? extends Annotation>) cl.loadClass("javax.inject.Named")), false));

　　　　logger.info("JSR-330 'javax.inject.Named' annotation found and supported for component scanning");

　　}

　　catch (ClassNotFoundException ex) {

　　　　// JSR-330 API not available - simply skip.

　　}

}

从以上源码我们可以看出**默认ClassPathBeanDefinitionScanner会自动注册对@Component、@ManagedBean、@Named注解的Bean进行扫描**。

## 4.在进行扫描时会通过include-filter/exclude-filter来判断你的Bean类是否是合法的：

protected boolean isCandidateComponent(MetadataReader metadataReader) throws IOException {

for (TypeFilter tf : this.excludeFilters) {

if (tf.match(metadataReader, this.metadataReaderFactory)) {

return false;

}

}

for (TypeFilter tf : this.includeFilters) {

if (tf.match(metadataReader, this.metadataReaderFactory)) {

AnnotationMetadata metadata = metadataReader.getAnnotationMetadata();

if (!metadata.isAnnotated(Profile.class.getName())) {

return true;

}

AnnotationAttributes profile = MetadataUtils.attributesFor(metadata, Profile.class);

return this.environment.acceptsProfiles(profile.getStringArray("value"));

}

}

return false;

}

从以上源码可看出：扫描时首先**通过exclude-filter 进行黑名单过滤**，然后通过include-filter 进行白名单过滤，否则默认排除。

# 三、结论

　　在spring-mvc.xml中进行如下配置：

<context:component-scan base-package="com.hafiz.www">

**<context:include-filter type="annotation" expression="org.springframework.stereotype.Controller"/>**

</context:component-scan>

则SpringMVC容器不仅仅扫描并注册带有@Controller注解的Bean，而且还扫描并注册了带有@Component的子注解@Service、@Reposity的Bean。**因为use-default-filters默认为true。**所以如果不需要默认的，则use-default-filters=“false”禁用掉。

当我们进行上面的配置时，SpringMVC容器会把service、dao层的bean重新加载，从而造成新加载的bean覆盖了老的bean，但事务的AOP代理没有配置在spring-mvc.xml配置文件中，造成事务失效。**解决办法是：在spring-mvc.xml配置文件中的context:component-scan标签中使用use-default-filters=“false”禁用掉默认的行为。**

**最终：**

<!-- Spring容器中注册非@controller注解的Bean -->

<context:component-scan base-package="com.hafiz.www">

<**context:exclude-filter** type="annotation" expression="org.springframework.stereotype.Controller"/>

</context:component-scan>

　　2.applicationContext-MVC.xml中配置

<!-- SpringMVC容器中只注册带有@controller注解的Bean -->

<context:component-scan base-package="com.hafiz.www" **use-default-filters="false"**>

**<context:include-filter** type="annotation" expression="org.springframework.stereotype.Controller" />

</context:component-scan>