Spring的父子容器问题

本文以Spring和SpringMVC的父子容器为例介绍Spring中的父子容器问题。

# 背景：

最近由于项目的包扫描出现了问题，在解决问题的过程中，偶然发现了Spring和SpringMVC是有父子容器关系的，而且正是因为这个才往往会出现包扫描的问题，我们在此来分析和理解Spring和SpringMVC的父子容器关系并且给出Spring和SpringMVC配置文件中包扫描的官方推荐方式。

# 概念理解和只是铺垫

在Spring整体框架的核心概念中，容**器是核心思想**，就是用来管理Bean的整个生命周期的，而**在一个项目中，容器不一定只有一个，Spring中可以包括多个容器**，而且容器有上下层关系，目前最常见的一种场景就是**在一个项目中引入Spring和SpringMVC这两个框架，那么它其实就是两个容器，Spring是父容器，SpringMVC是其子容器**，并且在Spring父容器中注册的Bean对于SpringMVC容器中是可见的，而在SpringMVC容器中注册的Bean对于Spring父容器中是不可见的，也就是子容器可以看见父容器中的注册的Bean，反之就不行。

　　我们可以使用统一的如下注解配置来对Bean进行批量注册，而不需要再给每个Bean单独使用xml的方式进行配置。

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

　　从Spring提供的参考手册中我们得知该配置的功能是扫描配置的base-package包下的所有使用了@Component注解的类，并且将它们自动注册到容器中，同时也扫描**@Controller，@Service，@Respository这三个注解，因为他们是继承自@Component**。

　　在项目中我们经常见到还有如下这个配置，其实有了上面的配置，这个是可以省略掉的，因为上面的配置会默认打开以下配置。以下配置会默认声明了@Required、@Autowired、 @PostConstruct、@PersistenceContext、@Resource、@PreDestroy等注解。

**<context:annotation-config/>**

　　另外，还有一个和SpringMVC相关如下配置，经过验证，这个是SpringMVC必须要配置的，因为它声明了@RequestMapping、@RequestBody、@ResponseBody等。并且，该配置默认加载很多的参数绑定方法，比如json转换解析器等。

<mvc:annotation-driven />

而上面这句配置spring3.1之前的版本和以下配置方式等价

<!--配置注解控制器映射器,它是SpringMVC中用来将Request请求URL到映射到具体Controller-->

<bean class="org.springframework.web.servlet.mvc.annotation.**DefaultAnnotationHandlerMapping**"/>

<!--配置注解控制器映射器,它是SpringMVC中用来将具体请求映射到具体方法-->

<bean class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter"/>

spring3.1之后的版本和以下配置方式等价

<!--配置注解控制器映射器,它是SpringMVC中用来将Request请求URL到映射到具体Controller-->

<bean class="org.springframework.web.servlet.mvc.method.annotation.**RequestMappingHandlerMapping**"/>

<!--配置注解控制器映射器,它是SpringMVC中用来将具体请求映射到具体方法-->

<bean class="org.springframework.web.servlet.mvc.method.annotation.**RequestMappingHandlerAdapter**"/>

# 具体场景分析

下面让我们来详细扒一扒Spring与SpringMVC的容器冲突的原因到底在那里？

　　我们共有Spring和SpringMVC两个容器，它们的配置文件分别为applicationContext.xml和applicationContext-MVC.xml。

　　1.在applicationContext.xml中配置了<context:component-scan base-package=“com.hafiz.www" />，负责所有需要注册的Bean的扫描和注册工作。

　　2.在applicationContext-MVC.xml中配置<mvc:annotation-driven />，负责SpringMVC相关注解的使用。

　　3.启动项目我们发现SpringMVC无法进行跳转，将log的日志打印级别设置为DEBUG进行调试，发现SpringMVC容器中的请求好像没有映射到具体controller中。

　　4.在applicationContext-MVC.xml中配置<context:component-scan base-package=“com.hafiz.www" />，重启后，验证成功，springMVC跳转有效。

　　下面我们来查看具体原因，翻看源码，从SpringMVC的DispatcherServlet开始往下找，我们发现SpringMVC初始化时，会寻找SpringMVC容器中的所有使用了@Controller注解的Bean，来确定其是否是一个handler。1,2两步的配置使得当前springMVC容器中并没有注册带有@Controller注解的Bean，而是把所有带有@Controller注解的Bean都注册在Spring这个父容器中了，所以springMVC找不到处理器，不能进行跳转。核心源码如下:

protected void initHandlerMethods() {

　　if (logger.isDebugEnabled()) {

　　　　logger.debug("Looking for request mappings in application context: " + getApplicationContext());

　　}

　　String[] beanNames = (this.detectHandlerMethodsInAncestorContexts ?

　　　　　　 BeanFactoryUtils.beanNamesForTypeIncludingAncestors(getApplicationContext(), Object.class) :

　　　　　　　getApplicationContext().getBeanNamesForType(Object.class));

　　for (String beanName : beanNames) {

　　　　if (isHandler(getApplicationContext().getType(beanName))){

　　　　　　detectHandlerMethods(beanName);

　　　　}

　　}

　　handlerMethodsInitialized(getHandlerMethods());

}

在方法isHandler中会判断当前bean的注解是否是controller，源码如下：

protected boolean isHandler(Class<?> beanType) {

　　return AnnotationUtils.findAnnotation(beanType, Controller.class) != null;

}

而在第4步配置中，SpringMVC容器中也注册了所有带有@Controller注解的Bean，故SpringMVC能找到处理器进行处理，从而正常跳转。

我们找到了出现不能正确跳转的原因，那么它的解决办法是什么呢？

　　我们注意到在initHandlerMethods()方法中，detectHandlerMethodsInAncestorContexts这个Switch，它主要控制获取哪些容器中的bean以及是否包括父容器，默认是不包括的。所以解决办法就是在springMVC的配置文件中配置HandlerMapping的detectHandlerMethodsInAncestorContexts属性为true即可（这里需要根据具体项目看使用的是哪种HandlerMapping），让它检测父容器的bean。如下：

<bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping">

<property name="detectHandlerMethodsInAncestorContexts">

<value>true</value>

</property>

</bean>

但在实际工程中会包括很多配置，我们按照官方推荐根据不同的业务模块来划分不同容器中注册不同类型的Bean：Spring父容器负责所有其他非@Controller注解的Bean的注册，而SpringMVC只负责@Controller注解的Bean的注册，使得他们各负其责、明确边界。配置方式如下

1.在applicationContext.xml中配置:

<!-- 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>

关于use-default-filters="false"的作用，请参见另一篇博客：context:component-scan标签的use-default-filters属性的作用以及原理分析

<http://www.cnblogs.com/hafiz/p/5875770.html>

# 总结

这样我们在清楚了spring和springMVC的父子容器关系、以及扫描注册的原理以后，**根据官方建议我们就可以很好把不同类型的Bean分配到不同的容器中进行管理。**再出现Bean找不到或者SpringMVC不能跳转以及事务的配置失效的问题，我们就可以很快的定位以及解决问题了。很开心，有木有~