# SpringMvc框架

## 常用的HttpMessageConverter

|  |
| --- |
| ByteArrayHttpMessageConverter converts byte arrays. StringHttpMessageConverter converts strings. ResourceHttpMessageConverter converts to/from org.springframework.core.io.Resource for all media types. SourceHttpMessageConverter converts to/from a javax.xml.transform.Source. FormHttpMessageConverter converts form data to/from a MultiValueMap<String, String>. Jaxb2RootElementHttpMessageConverter converts Java objects to/from XML — added if JAXB2 is present on the classpath. MappingJacksonHttpMessageConverter converts to/from JSON — added if Jackson is present on the classpath. AtomFeedHttpMessageConverter converts Atom feeds — added if Rome is present on the classpath. RssChannelHttpMessageConverter converts RSS feeds — added if Rome is present on the classpath.  ByteArrayHttpMessageConverter: 负责读取二进制格式的数据和写出二进制格式的数据；  StringHttpMessageConverter：负责读取字符串格式的数据和写出二进制格式的数据；  ResourceHttpMessageConverter：负责读取资源文件和写出资源文件数据；  FormHttpMessageConverter：负责读取form提交的数据（能读取的数据格式为 application/x-www-form-urlencoded，不能读取multipart/form-data格式数据）；负责写入application/x-www-from-urlencoded和multipart/form-data格式的数据；  MappingJacksonHttpMessageConverter: 负责读取和写入json格式的数据；  SouceHttpMessageConverter：负责读取和写入 xml 中javax.xml.transform.Source定义的数据；  Jaxb2RootElementHttpMessageConverter：负责读取和写入xml 标签格式的数据；  AtomFeedHttpMessageConverter：负责读取和写入Atom格式的数据；  RssChannelHttpMessageConverter：负责读取和写入RSS格式的数据； |

## [spring mvc 注解配置StringHttpMessageConverter不生效](http://blog.csdn.net/sen19910708/article/details/49933213)

### 方法一

当使用@ResponseBody返回的字符串带有中文时，返回类型为String会被StringHttpMessageConverter处理，当时查看源码发现默认的Charset DEFAULT\_CHARSET使用的是ISO-8859-1，这里除了修改配置外，可以使用覆盖类的方式来修改这个类的源码，即在自己的项目下新建org.springframework.http.converter包，然后新建类StringHttpMessageConverter，修改里面的

**public** **static** **final** Charset DEFAULT\_CHARSET = Charset.forName("utf-8");

这样当项目启动后，工程的类会优先被加载，这样可以达到我们想要的效果。

### 方法二

 另外一种方式就是直接配置DefaultAnnotationHandlerMapping和AnnotationMethodHandlerAdapter这两个类，当我们使用 mvc:annotation-driven 这个标签时，其实也是来配置这两个类的，直接采用下面的配置

|  |
| --- |
| 1. <!-- 手动配置Springmvc注解驱动 --> 2. **<bean** class="org.springframework.web.servlet.mvc.annotation.DefaultAnnotationHandlerMapping"**/>** 4. <!-- 启动Spring MVC的注解功能,设置编码方式，防止乱码 --> 5. **<bean** class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter"**>** 6. **<property** name="messageConverters"**>** 7. **<list>** 8. **<bean** class="org.springframework.http.converter.ByteArrayHttpMessageConverter" **/>** 9. **<bean** class = "org.springframework.http.converter.StringHttpMessageConverter"**>** 10. **<property** name="supportedMediaTypes" value="text/plain;charset=UTF-8" **/>** 11. **</bean>** 12. **<bean** class="org.springframework.http.converter.json.MappingJackson2HttpMessageConverter"**>** 13. **<property** name="supportedMediaTypes"**>** 14. **<list>** 15. **<value>**text/html;charset=UTF-8**</value>** 16. **</list>** 17. **</property>** 18. **</bean>** 19. **</list>** 20. **</property>** 21. **</bean>** |

# [Spring中的<context:annotation-config/>配置](http://www.cnblogs.com/iuranus/archive/2012/07/19/2599084.html)

### <beans>加入命名空间

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns="http://www.springframework.org/schema/beans"  xmlns:context="http://www.springframework.org/schema/context"  ...  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans-3.0.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context-3.0.xsd  ..  >  <context:annotation-config/>  </beans> |

### Processor

使用@Autowired注解，必须事先在Spring容器中声明AutowiredAnnotationBeanPostProcessor的Bean：

|  |
| --- |
| <bean class="org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor "/> |

　使用 @Required注解，就必须声明RequiredAnnotationBeanPostProcessor的Bean：

|  |
| --- |
| <bean class="org.springframework.beans.factory.annotation.RequiredAnnotationBeanPostProcessor"/> |

类似地，使用

@Resource、

@PostConstruct、

@PreDestroy

等注解就必须声明 CommonAnnotationBeanPostProcessor；

使用@PersistenceContext注解，就必须声明 PersistenceAnnotationBeanPostProcessor的Bean。

这样的声明未免太不优雅，而Spring为我们提供了一种极为方便注册这些BeanPostProcessor的方式，即使用<context:annotation- config/>隐式地向 Spring容器注册AutowiredAnnotationBeanPostProcessor、RequiredAnnotationBeanPostProcessor、CommonAnnotationBeanPostProcessor以及PersistenceAnnotationBeanPostProcessor这4个BeanPostProcessor。

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

### component-scan

在我们使用注解时一般都会配置扫描包路径选项：

|  |
| --- |
| <context:component-scan base-package="pack.pack"/>  该配置项其实也包含了自动注入上述processor的功能，因此当使用<context:component-scan/>后，即可将<context:annotation-config/>省去  <context:annotation-config/>还提供了两个子标签  1.        <context:include-filter>  2.       <context:exclude-filter>  <context:component-scan>有一个use-default-filters属性，该属性默认为true  这就意味着会扫描指定包下的全部的标有@Component的类，并注册成bean.也就是@Component的子注解@Service,@Reposity等  Use-default-filter此时为true那么会对base-package包或者子包下的所有的进行java类进行扫描,并把匹配的java类注册成bean。  可以发现这种扫描的粒度有点太大，如果你只想扫描指定包下面的Controller，该怎么办？此时子标签<context:incluce-filter>就起到了勇武之地。如下所示  <context:component-scan base-package="tv.huan.weisp.web .controller">  <context:include-filter type="annotation" expression="org.springframework.stereotype.Controller"/>  </context:component-scan>  这样就会只扫描base-package指定下的有@Controller下的java类，并注册成bean  指定的include-filter没有起到作用，只要把use-default-filter设置成false就可以了  Use-dafault-filters=”false”的情况下：<context:exclude-filter>指定的不扫描，<context:include-filter>指定的扫描 |

# 处理在类级别上的@RequestMapping注解

## DefaultAnnotationHandlerMapping

|  |
| --- |
| <!-- 处理在类级别上的@RequestMapping注解 -->  <bean      class="org.springframework.web.servlet.mvc.annotation.DefaultAnnotationHandlerMapping">  <property name="interceptors">  <list>  <!-- 多个拦截器,顺序执行 -->  <ref bean="SpringMVCInterceptor" />  <ref bean="OpenSessionInViewInterceptor"/>  </list>  </property>  </bean> |

# 处理方法级别上的@RequestMapping注解

## AnnotationMethodHandlerAdapter

|  |
| --- |
| <!-- 处理方法级别上的@RequestMapping注解 -->  <bean id="annotationMethodHandlerAdapter"  class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter">  <property name="messageConverters">  <list>  <bean      class="org.springframework.http.converter.StringHttpMessageConverter">      <property name="supportedMediaTypes">  <list>  <value>text/html;charset=utf-8</value>  <value>text/plain;charset=utf-8</value>  </list>      </property>  </bean>  <bean  class="org.springframework.http.converter.json.MappingJacksonHttpMessageConverterv2">  <property name="objectMapper">  <bean class="net.pm.misc.Hibernate4AwareObjectMapper" />  </property>  </bean>  </list>  </property>  </bean> |

# 配置文件的读取与PropertyPlaceholderConfigurer的用法

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <beans           xmlns="http://www.springframework.org/schema/beans"           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"           xmlns:p="http://www.springframework.org/schema/p"           xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/**spring-beans-2.5.xsd**">  <!-- 使用spring提供的**PropertyPlaceholderConfigurer**读取数据库配置信息**.properties** -->           <bean id="propertyConfigurer" class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">                     <property name="**locations**">                              <list>  <!--  这里的classpath可以认为是项目中的src属性名是 **locations，使用子标签<list></list>可以指定多个数据库的配置文件，这里指定了一个**  -->                                       <value>classpath:resource/config/jdbc.properties</value>                              </list>                     </property>           </bean>  </beans> |