# Spring Boot核心-Web配置

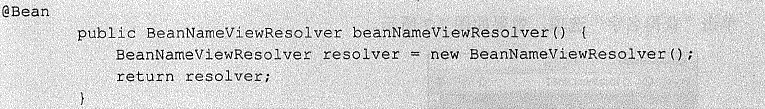
## Spring Boot 提供的自动配置

分析WebMvcAutoConfiguration 及WebMvcProperties 的源码，发现Spring Boot为我们提供了如下自动配置功能。

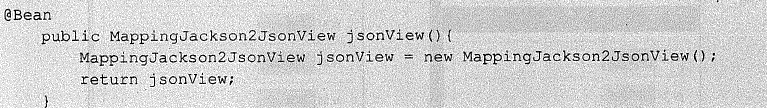
### 自动配置ViewResolver

1. ContentNegotiatingViewResolver：这是一个特殊的ViewResolver, ContentNegotiatingViewResolver 不自己处理View ，而是代理给不同的ViewResolver 来处理，它有最高的优先级。
2. BeanNameViewResolver：根据Bean的名称来解析视图，如控制器（ @Controller ）中的一个方法的返回值的字符串（视图名）为X，BeanNameView Resolver 去查找名为X的View 来渲染视图。

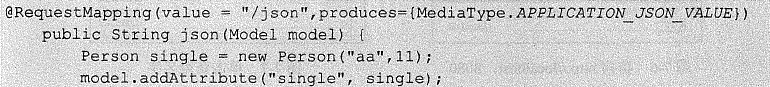
定义BeanNameView Resolver 的Bean:



定义一个View 的Bean ，名称为jsonView



在控制器中，返回值为字符串jsonView ，它会找Bean 的名称为jsonView 的视图来渲染：





1. IntemalResource View Resolver:

这个是一个极为常用的ViewResolver，主要通过设置前缀、后缀，以及控制器中方法来返回视图名的字符串，以得到实际页面， Spring Boot 的源码如下：

|  |
| --- |
| @Bean @ConditionalOnMissingBean **public** InternalResourceViewResolver defaultViewResolver() {  InternalResourceViewResolver resolver = **new** InternalResourceViewResolver();  resolver.setPrefix(**this**.**mvcProperties**.getView().getPrefix());  resolver.setSuffix(**this**.**mvcProperties**.getView().getSuffix());  **return** resolver; } |

### 自动配置的静态资源

在自动自己置类的addResourceHandlers 方法中定义了以下静态资源的自动配置。

( 1) 类路径文件：把类路径下的/static 、/public 、/resources 和/META-INF/resources 文件夹下的静态文件直接映射为/\*\*，可以通过<http://localhost:8080>/\*\*来访问。

(2) webjar：webjar 就是将我们常用的脚本框架封装在jar 包中的jar 包，把webjar 的/META-INF/resources/webjars/下的静态文件映射为/webjar/\*\*，可以通过http://localhost: 8080/webjar/\*\*来访问。

### 自动配置Formatter和Converter

从WebMvcAutoConfiguration代码中可以看到，只要我们定义了Converter 、GenericConverter 和Formatter 接口的实现类的Bean, 这些 Bean 就会自动注册到Spring 容器中。

|  |
| --- |
| @Override **public void** addFormatters(FormatterRegistry registry) {  **for** (Converter<?, ?> converter : getBeansOfType(Converter.**class**)) {  registry.addConverter(converter);  }  **for** (GenericConverter converter : getBeansOfType(GenericConverter.**class**)) {  registry.addConverter(converter);  }  **for** (Formatter<?> formatter : getBeansOfType(Formatter.**class**)) {  registry.addFormatter(formatter);  } } |

### HttpMessageConverters

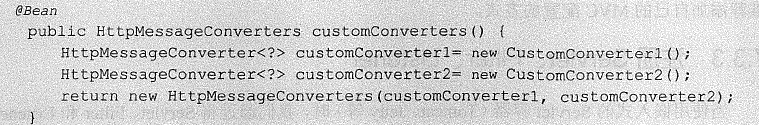
在WebMvcAutoConfiguration 中，注册了messageConverters

|  |
| --- |
| @Configuration @Import(EnableWebMvcConfiguration.**class**) @EnableConfigurationProperties({ WebMvcProperties.**class**, ResourceProperties.**class** }) **public static class** WebMvcAutoConfigurationAdapter **extends** WebMvcConfigurerAdapter {   **private final** HttpMessageConverters **messageConverters**;   @Override  **public void** configureMessageConverters(List<HttpMessageConverter<?>> converters) {  converters.addAll(**this**.**messageConverters**.getConverters());  }  } |

这里直接注入了HttpMessageConverters 的Bean ，这个Bean 是在HttpMessageConvertersAutoConfiguration 类中定义的，我们自动注册的HttpMessageConverter除了Spring MVC 默认的ByteArrayHttpMessageConverter 、StringHttpMessageConverter 、ResourceHttpMessageConverter、SourceHttpMessageConverter、AllEncompassingFormHttpMessageConverter外，在HttpMessageConverters AutoConfiguration 的自动配置文件里还引入了JacksonHttpMessageConverters Configuration 和GsonHttpMessage ConverterConfiguration 。

1. 若jackson 的jar 包在类路径上，则Spring Boot 通过JacksonHttpMessageConvertersConfiguration 增加MappingJackson2HttpMessage Converter 和Mapping Jackson2XmlHttpMessageConverter;
2. 若gson 的jar 包在类路径上，则Spring Boot 通过GsonHttpMessageConverterConfiguration 增加GsonHttpMessageConvertera

在Spring Boot 中，如果要新增自定义的HttpMessageConverter，则只需定义一个你自己的HttpMessageConverters的Bean ，然后在此Bean 中注册自定义HttpMessageConverterBean。



### 静态首页的支持

把静态index.html 文件放在如下目录。

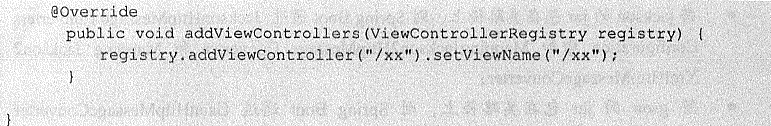
* classpath:/META-INF /resources/index.html
* classpath:/resources/index.html
* classpath:/static/index.html
* classpath:/public/index.html

当我们访问应用根目录http://localhost: 8080/时，会直接映射。

## 接管Spring Boot 的Web 配置

如果Spring Boot 提供的Spring MVC 不符合要求，则可以通过一个配置类（注解有@Configuration 的类）加上@EnableWebMvc 注解来实现完全自己控制的MVC 配置。如果既需要保留Spring Boot 提供的便利，又需要增加自己的额外的配置的时候，可以定义一个配置类并继承WebMvcConfigurer Adapter，无须使用＠EnableWebMvc 注解。



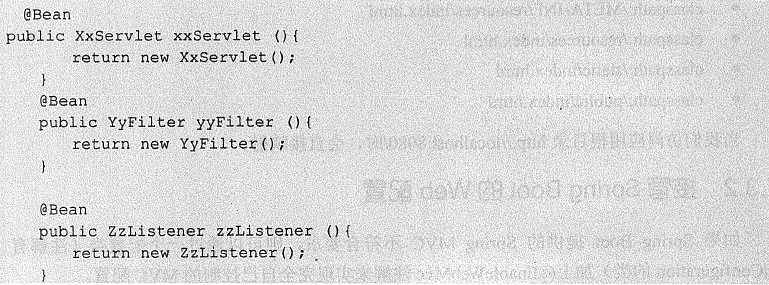


在这的addViewControllers 方法，并不会覆盖Web MvcAutoConfiguration 中的addViewControllers，也就是我们自己的配置和Spring Boot 的自动配置同时有效，这是推荐的MVC 配置的方式。

## 注册Servlet、Filter、Listener

可以通过将Servlet 、Filter 和Listener声明为Spring Bean 而达到注册的效果；或者注册ServletRegistrationBean、FilterRegistrationBean 和ServletListenerRegistrationBean 的Bean。

1. 直接注册Bean 示例



1. 通过RegistrationBean 示例

