https://mp.weixin.qq.com/s/WyP6Jl5-HEQU3KmAqWjl\_w

# Spring Boot 中自定义 SpringMVC 配置

用过 Spring Boot 的小伙伴都知道，我们只需要在项目中引入 spring-boot-starter-web 依赖，SpringMVC 的一整套东西就会自动给我们配置好，但是，真实的项目环境比较复杂，系统自带的配置不一定满足我们的需求，往往我们还需要结合实际情况自定义配置。

自定义配置就有讲究了，由于 Spring Boot 的版本变迁，加上这一块本身就有几个不同写法，很多小伙伴在这里容易搞混，今天松哥就来和大家说一说这个问题。

## 概览

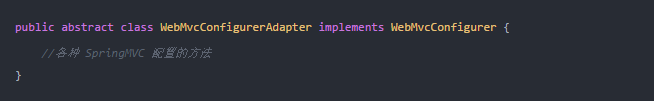
首先我们需要明确，跟自定义 SpringMVC 相关的类和注解主要有如下四个：

* WebMvcConfigurerAdapter
* WebMvcConfigurer
* WebMvcConfigurationSupport
* @EnableWebMvc

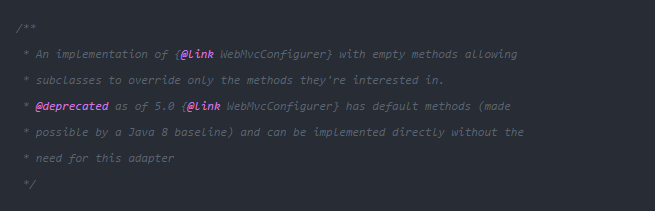
这四个中，除了第四个是注解，另外三个两个类一个接口，里边的方法看起来好像都类似，但是实际使用效果却大不相同，因此很多小伙伴容易搞混，今天松哥就来和大家聊一聊这个问题。

## WebMvcConfigurerAdapter

我们先来看 WebMvcConfigurerAdapter，这个是在 Spring Boot 1.x 中我们自定义 SpringMVC 时继承的一个抽象类，这个抽象类本身是实现了 WebMvcConfigurer 接口，然后抽象类里边都是空方法，我们来看一下这个类的声明：



再来看看这个类的注释：



这段注释关于这个类说的很明白了。同时我们也看到，从 Spring5 开始，由于我们要使用 Java8，而 Java8 中的接口允许存在 default 方法，因此官方建议我们直接实现 WebMvcConfigurer 接口，而不是继承 WebMvcConfigurerAdapter 。

也就是说，在 Spring Boot 1.x 的时代，如果我们需要自定义 SpringMVC 配置，直接继承 WebMvcConfigurerAdapter 类即可。

## WebMvcConfigurer

根据上一小节的解释，小伙伴们已经明白了，WebMvcConfigurer 是我们在 Spring Boot 2.x 中实现自定义配置的方案。

WebMvcConfigurer 是一个接口，接口中的方法和 WebMvcConfigurerAdapter 中定义的空方法其实一样，所以用法上来说，基本上没有差别，从 Spring Boot 1.x 切换到 Spring Boot 2.x ，只需要把继承类改成实现接口即可。

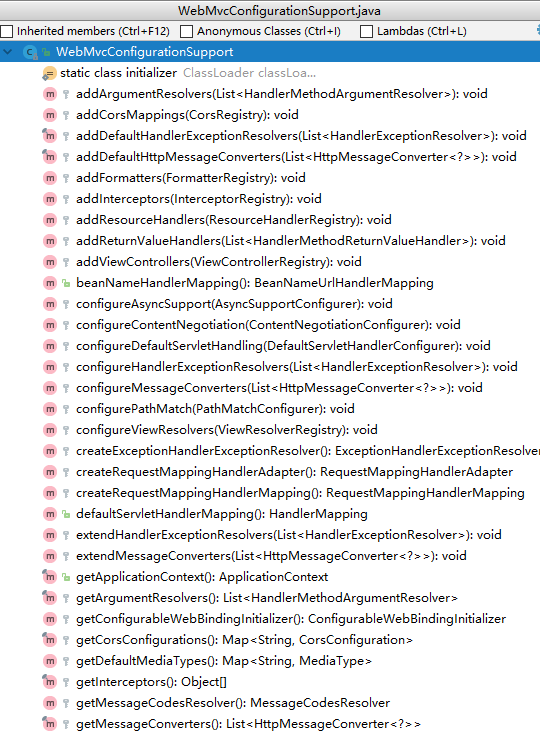
**WebMvcConfigurationSupport**

前面两个都好理解，还有一个 WebMvcConfigurationSupport ，这个又是干什么用的呢？

松哥之前有一篇文章中用过这个类，不知道小伙伴们有没有留意，就是下面这篇：

* [**纯 Java 代码搭建 SSM 环境**](https://mp.weixin.qq.com/s?__biz=MzI1NDY0MTkzNQ==&mid=2247485596&idx=1&sn=14a5b1e07b09df838328388ce77cbba5&scene=21#wechat_redirect)（第一篇）

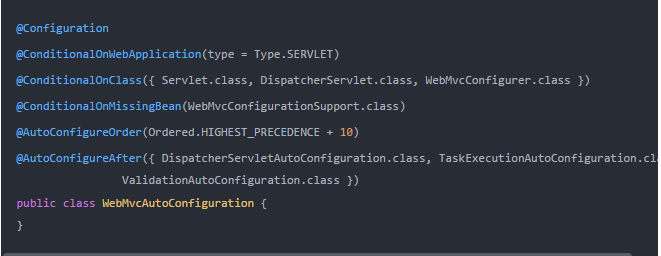
这篇文章我放弃了 Spring 和 SpringMVC 的 xml 配置文件，转而用 Java 代替这两个 xml 配置。那么在这里我自定义 SpringMVC 配置的时候，就是通过继承 WebMvcConfigurationSupport 类来实现的。在 WebMvcConfigurationSupport 类中，提供了用 Java 配置 SpringMVC 所需要的所有方法。我们来看一下这个方法的摘要：



有一点眼熟，可能有小伙伴发现了，这里的方法其实和前面两个类中的方法基本是一样的。

在这里首先大家需要明确的是，WebMvcConfigurationSupport 类本身是没有问题的，我们自定义 SpringMVC 的配置是可以通过继承 WebMvcConfigurationSupport 来实现的。但是继承 WebMvcConfigurationSupport 这种操作我们一般只在 Java 配置的 SSM 项目中使用，Spring Boot 中基本上不会这么写，为什么呢？

小伙伴们知道，Spring Boot 中，SpringMVC 相关的自动化配置是在 WebMvcAutoConfiguration 配置类中实现的，那么我们来看看这个配置类的生效条件：

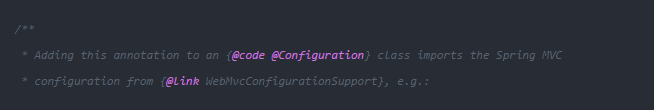


我们从这个类的注解中可以看到，它的生效条件有一条，就是当不存在 WebMvcConfigurationSupport 的实例时，这个自动化配置才会生生效。因此，如果我们在 Spring Boot 中自定义 SpringMVC 配置时选择了继承 WebMvcConfigurationSupport，就会导致 Spring Boot 中 SpringMVC 的自动化配置失效。

**Spring Boot 给我们提供了很多自动化配置，很多时候当我们修改这些配置的时候，并不是要全盘否定 Spring Boot 提供的自动化配置，我们可能只是针对某一个配置做出修改，其他的配置还是按照 Spring Boot 默认的自动化配置来，而继承 WebMvcConfigurationSupport 来实现对 SpringMVC 的配置会导致所有的 SpringMVC 自动化配置失效，因此，一般情况下我们不选择这种方案。**

## @EnableWebMvc

最后还有一个 @EnableWebMvc 注解，这个注解很好理解，它的作用就是启用 WebMvcConfigurationSupport。我们来看看这个注解的定义：



可以看到，加了这个注解，就会自动导入 WebMvcConfigurationSupport，所以在 Spring Boot 中，我们也不建议使用 @EnableWebMvc 注解，因为它一样会导致 Spring Boot 中的 SpringMVC 自动化配置失效。

**总结**

不知道上面的解释小伙伴有没有看懂？我再简单总结一下：

1. Spring Boot 1.x 中，自定义 SpringMVC 配置可以通过继承 WebMvcConfigurerAdapter 来实现。
2. Spring Boot 2.x 中，自定义 SpringMVC 配置可以通过实现 WebMvcConfigurer 接口来完成。
3. 如果在 Spring Boot 中使用继承 WebMvcConfigurationSupport 来实现自定义 SpringMVC 配置，或者在 Spring Boot 中使用了 @EnableWebMvc 注解，都会导致 Spring Boot 中默认的 SpringMVC 自动化配置失效。
4. 在纯 Java 配置的 SSM 环境中，如果我们要自定义 SpringMVC 配置，有两种办法，第一种就是直接继承自 WebMvcConfigurationSupport 来完成 SpringMVC 配置，还有一种方案就是实现 WebMvcConfigurer 接口来完成自定义 SpringMVC 配置，如果使用第二种方式，则需要给 SpringMVC 的配置类上额外添加 @EnableWebMvc 注解，表示启用 WebMvcConfigurationSupport，这样配置才会生效。换句话说，在纯 Java 配置的 SSM 中，如果你需要自定义 SpringMVC 配置，你离不开 WebMvcConfigurationSupport ，所以在这种情况下建议通过继承 WebMvcConfigurationSupport 来实现自动化配置。