# 极速开发框架SpringBoot及源码分析

主讲：Cat 老师

**北京动力节点教育科技有限公司**  
**2020 • 北京**

**动力节点版权所有，禁止私自传播**

## 快速回顾

What is SpringBoot？

SpringBoot开发运行环境要求

Springboot程序的几种创建方式

SpringBoot开发java程序

SpringBoot开发web程序

SpringBoot main方法分析

SpringBoot项目约定的代码结构

SpringBoot 项目pom.xml继承结构

SpringBoot 项目pom.xml修改默认版本

SpringBoot 整合mybatis

SpringBoot 整合jsp视图展示（也支持一些模板技术）

SpringBoot 项目pom.xml热部署插件devtools

JRebel （这是idea插件，收费的，需要破解）

SpringBoot开发最佳实践

SpringBoot敏感配置信息加密

SpringBoot数据库连接池

SpringBoot跨域资源共享CORS

SpringBoot静态资源处理

SpringBoot核心配置文件（Duration/DataSize单位类型、随机值、占位符、多环境、4种配置文件读取方式）

SpringBoot线程池异步调用（@Async、手动编码）

SpringBoot集成Swagger（OAS、springfox、swagger的关系）

SpringBoot集成websocket

注册Servlets, Filters, Listeners作为Spring Beans

SpringBoot使用拦截器

SpringBoot内嵌Web服务器

SpringBoot打包部署

SpringBoot配置SSL（https）

SpringBoot全局异常处理

SpringBoot 404页面处理

SpringBoot源码分析

课堂上未讲解部分：

SpringBoot多数据源

SpringBoot服务监控 springboot actuator、springboot admin；

springboot整合thymeleaf模板

SpringBoot RestTemplate

SpringBoot WebClient

Validation验证，@Vilidate注解验证

SpringBoot统一缓存处理

SpringBoot定时任务quartz

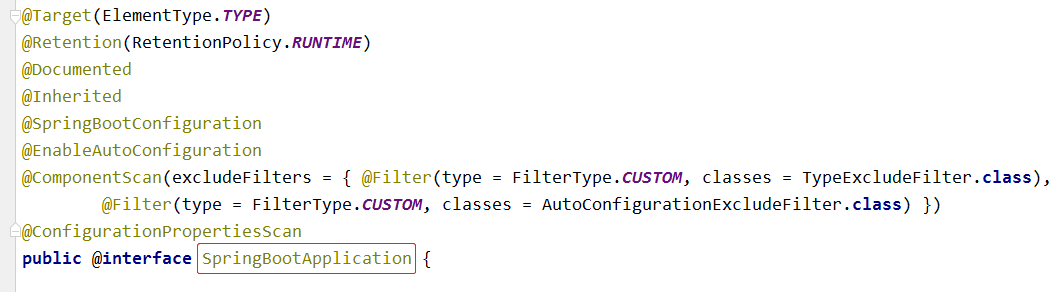
## SpringBoot自动配置原理

SpringBoot自动配置原理（也叫 SpringBoot自动装配原理，SpringBoot starter原理）

流程图：

<https://www.processon.com/view/link/5efc4ffa6376891e81f3895d>

在编写SpringBoot项目时，入口类上都会使用@SpringBootApplication注解了，我们可以看一下源代码



1. @SpringBootApplication

我们先来看该注解本身，它本身还可以指定如下属性：

exclude : 排除指定的类，比如：

@SpringBootApplication(exclude = RedisAutoConfiguration.**class**)

可以排除自定义配置等等，根据自己的需要进行定制；

excludeName：排除指定的类名，通过bean name来进行排除指定的类，如下：

@SpringBootApplication(excludeName = "org.springframework.boot.autoconfigure.data.redis.RedisAutoConfiguration")

scanBasePackages: 扫描指定的包添加到Spring容器中，参数为数组类型：

@SpringBootApplication(scanBasePackages="com.bjpowernode.boot.component")

扫描的包能注册识别，没有扫描的包将不能注册识别；

scanBasePackageClasses: 扫描指定包的类并添加到Spring容器中，参数为Class类型数组格式：

@SpringBootApplication(scanBasePackageClasses=MyComponent.class)

注册的类能识别，在同级包下或子包下的都能注册，否则不能识别；

@SpringBootApplication注解本身又是一个复合注解，它等效于如下四个注解：

@SpringBootConfiguration

@EnableAutoConfiguration

@ComponentScan

@ConfigurationPropertiesScan

**@SpringBootConfiguration**

该注解等效于@Configuration，那也就是说这个注解相当于@Configuration，所以这两个注解作用是一样的，它是让我们能够去注册一些额外的Bean，或者导入一些额外的配置，@Configuration还表示该类是一个配置类，不需要额外的xml进行配置，同时该类也是Spring容器中的一个bean。

@EnableAutoConfiguration （重点）

该注解是Spring Boot自动配置注解，Spring Boot中的自动配置主要是@EnableAutoConfiguration的功劳，该注解可以让Spring Boot根据类路径中的jar包依赖为当前项目进行自动配置，该注解由如下两个注解构成：

@AutoConfigurationPackage

@Import(EnableAutoConfigurationImportSelector.class)

@AutoConfigurationPackage

让包中的类以及子包中的类能够被自动扫描到Spring容器中；

@Import(AutoConfigurationPackages.Registrar.**class**)

*register*(registry, **new** PackageImport(metadata).getPackageName());

@Import注解，这个注解就是通过import的方式将bean添加到Spring容器中；

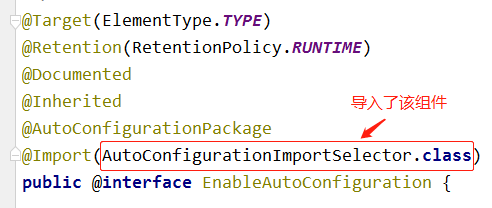
接下来我们重点看看注解：

@EnableAutoConfiguration

这是Spring Boot自动配置的核心注解；

Spring框架本身也提供了几个名字为@Enable开头的Annotation定义。比如@EnableScheduling、@EnableCaching等，@EnableAutoConfiguration的理念和这些注解其实是一脉相承的。

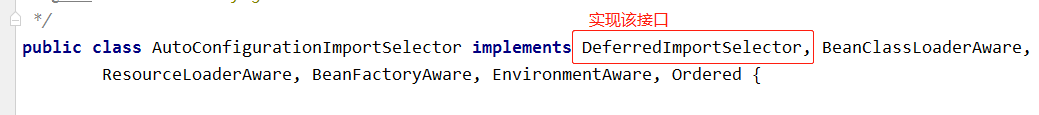
@EnableAutoConfiguration是借助@Import的帮助，将所有符合自动配置条件的bean定义加载到IoC容器；

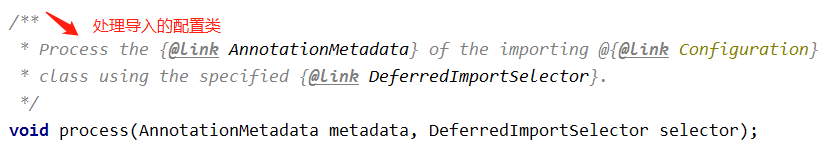


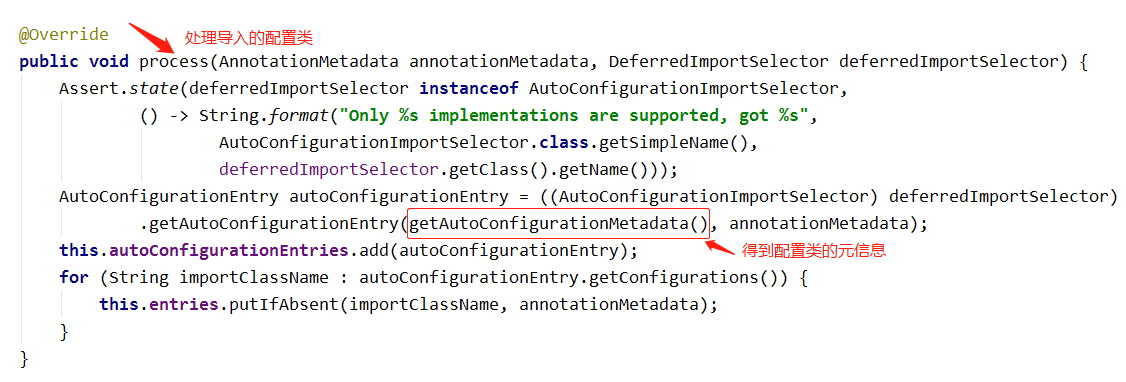
该注解通过@Import注解导入了一个组件：AutoConfigurationImportSelector

，该组件实现了接口：  
**public interface** DeferredImportSelector **extends** ImportSelector

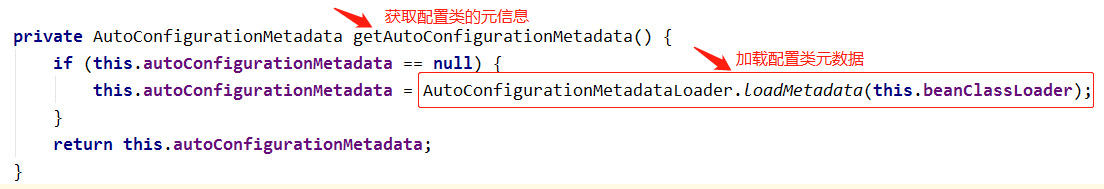
该接口主要是为了导入@Configuration的配置项；

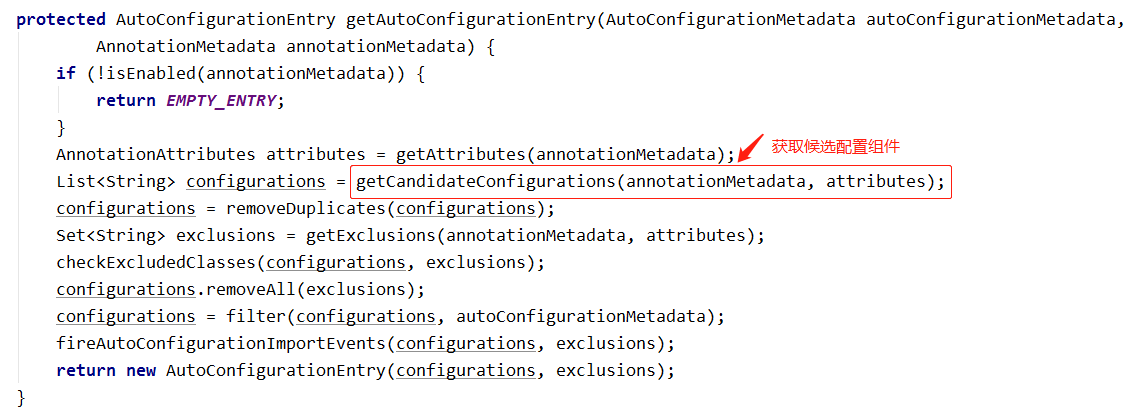


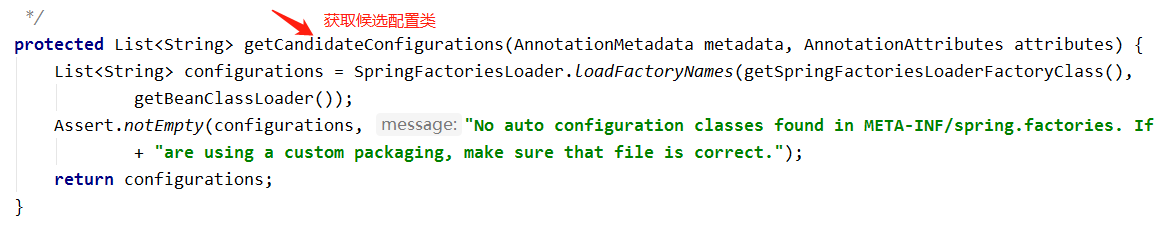




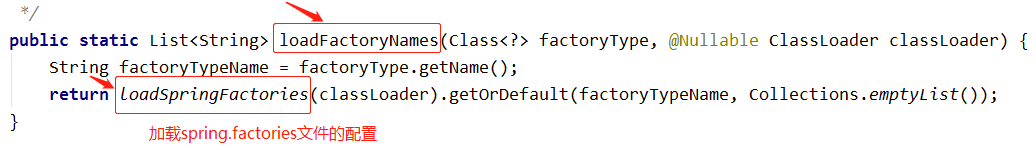
会调用getCandidateConfigurations方法：

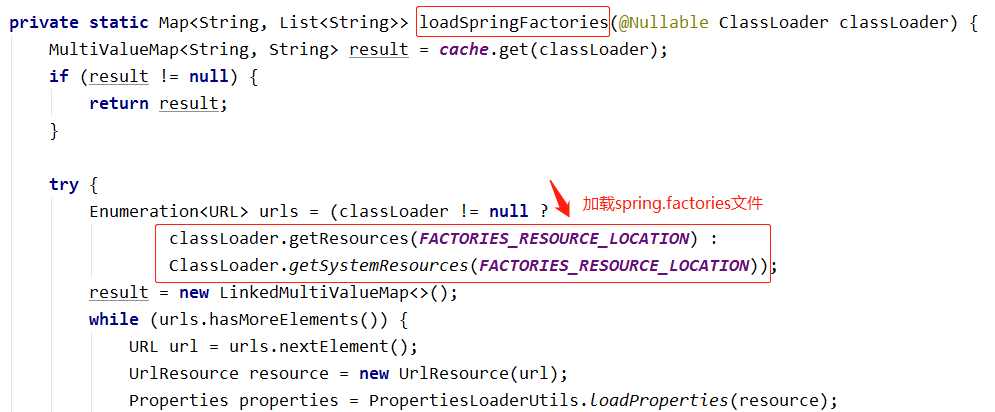


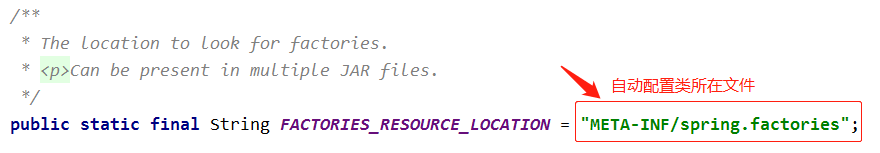




在这里遇到我们熟悉的SpringFactoryiesLoader, 它会读取META-INF/spring.factories下的EnableAutoConfiguration的配置，紧接着在进行排除与过滤，进而得到需要装配的类；







@EnableAutoConfiguration与@Conditional

@EnableAutoConfiguration自动加载配置，@Conditional根据环境决定是否解析处理配置，这两个注解的配合完成了自动化配置功能；

## Spring Boot中如何自定义starter？

1. 写一个starter项目，一般是一个空壳，里面不写代码，主要是起到依赖其他项目的作用；
2. 写一个真正的实现自动装配逻辑的项目，在项目classpath下写一个META-INF/spring.factories文件；

3、在spring.factories中添加配置：

org.springframework.boot.autoconfigure.EnableAutoConfiguration =

com.bjpowernode.xxxConfiguration（该类我们自己去实现，里面主要是要做自动化的配置，给使用者把该配的配置好，让别人可以直接使用）

4、xxxConfiguration类需要添加注解@Configuration；

5、xxxConfiguration类一般也会使用@Conditional来适应不同的环境；

6、在xxxConfiguration类中实现自动化配置；

有了SpringBoot的自动化配置，我们可以灵活的自定义我们自己的自动配置，当应用需要该功能时，只需要简单的依赖该starter即可，同时Spring Boot为我们提供的条件注解，同样的代码可以灵活适应各种环境；