**Springboot框架中，它实现了自动配置，他的原理是在启动类上加了@SpringbootConfiguration注解，所以springBoot框架实现自动配置主要依靠@SpringbootConfiguration这个注解，以下内容是对这个注解的原理解剖：**

**打开@SpringbootConfiguration,可以看到这个注解上面有两个元注解以及一个组件扫描注解，自动配置主要依靠的就是这三个元注解**



**1.@SpringbootConfiguration：**

1.先看字面意思，字面意思就是---springboot配置

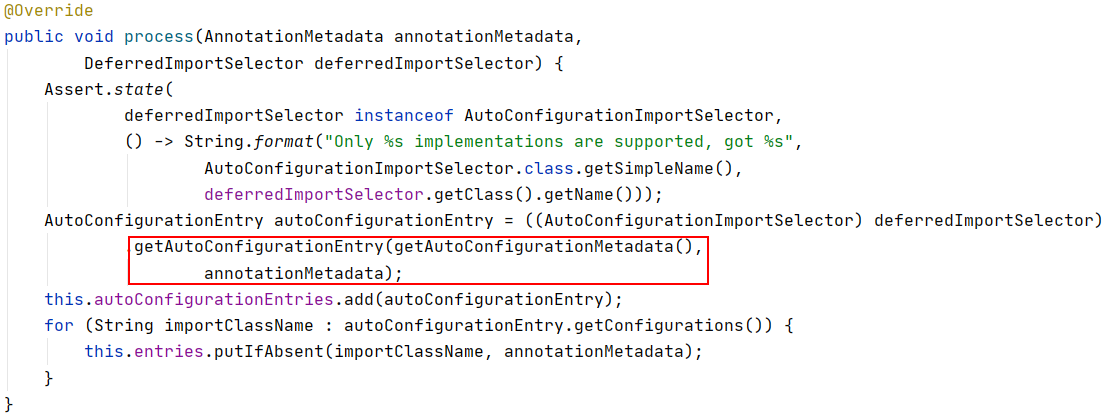
2.再打开@SpringbootConfiguration，看到该注解上面有@Configuration注解，所以理解为@SpringbootConfiguration将启动类修饰成一个配置类，这时启动类就等价于spring中的applicationContext.xml文件，用启动类来加载配置使用。

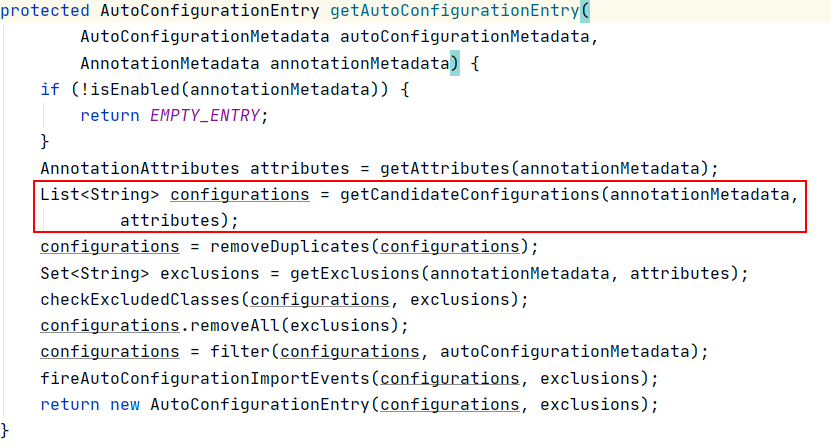
**2.@EnableAutoConfiguration:**

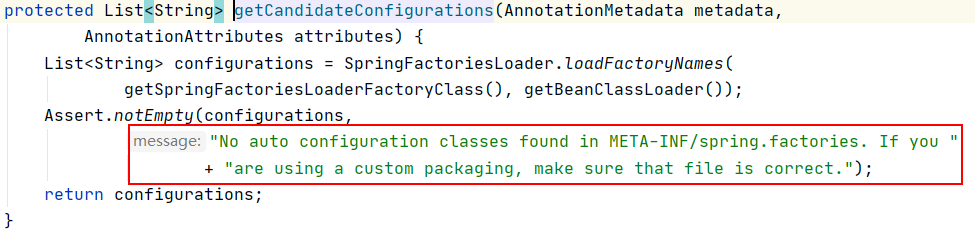
1.字面意思是---允许自动配置，这个注解就是起自动装配的作用；

2.打开注解SpringbootConfiguration上面的元注解@EnableAutoConfiguration，看到该注解上导入了一个类，@Import(AutoConfigurationImportSelector.class)，在自动配置导入选择器这个类中，有一个方法process() ，（注:老版本里面是执行selectImports()方法）

3.在process（）方法中，进行一系列的调用

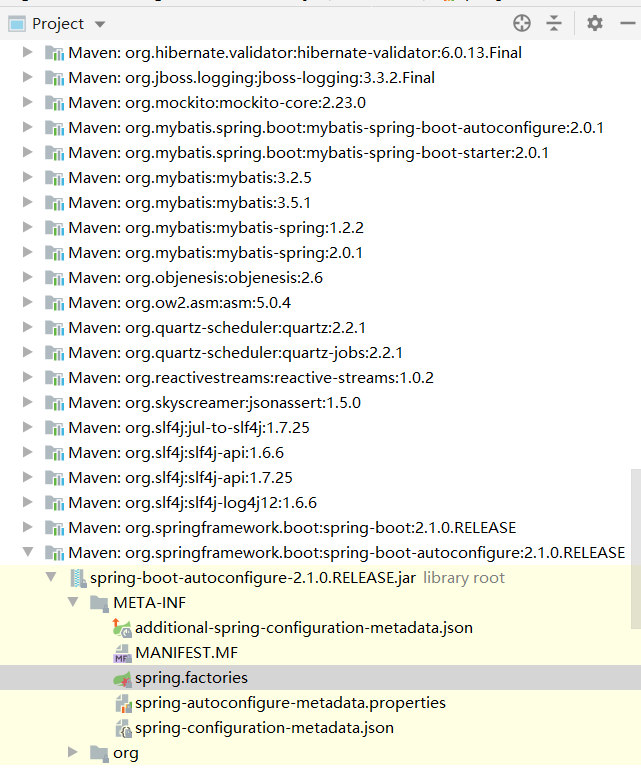




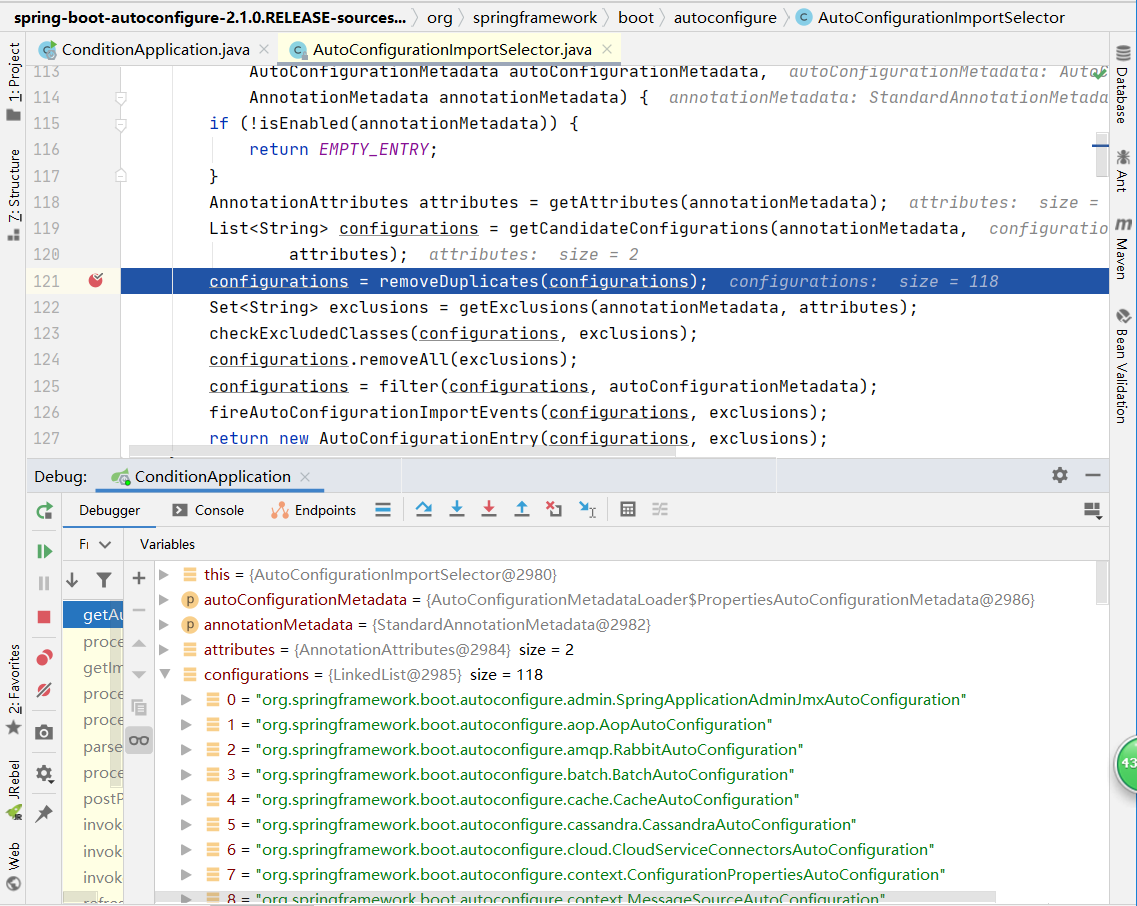


4.在META-INF/spring.factories中没有发现自动配置类。如果你是使用自定义包装，确保文件是正确的。

通过信息可以了解，自动配置文件最后是通过SpringFactoriesLoader.loadFactoryNames 方法实现的，这个方法的作用就是从META-INF/spring.factories文件中读取指定类对应的类名称列表，配置文件位置：META-INF/spring.factories，该配置文件中定义了大量的配置类，当 SpringBoot 应用启动时，AutoConfigurationImportSelector内部会自动加载这些配置类，初始化Bean



5.在AutoConfigurationImportSelector类中加断点解析获得 List<String> configurations的值（如下图所示）:



**2.@ComponentScan**

1.@ComponentScan和前两个注解都位于@SpringBootApplication注解上，字面意思是---组成部分扫描

2.@ComponentScan注解的作用是用于扫描组件包，类似于context-componet-scan,

注意：如果不指定扫描路径，默认扫描该注解修饰的启动类所在的包以及子包