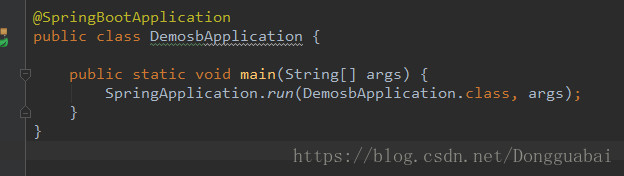
先看看SpringBoot的主配置类：

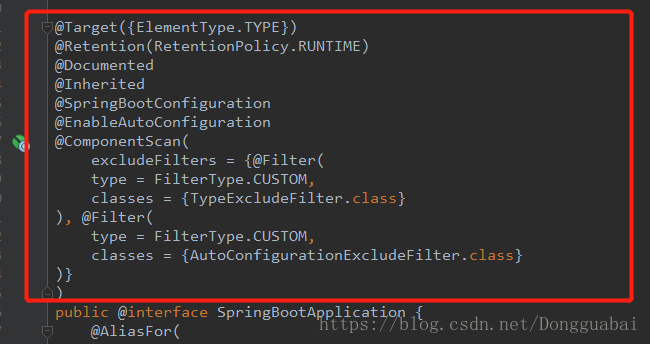


里面有一个main方法运行了一个run()方法，在run方法中必须要传入一个被@SpringBootApplication注解的类。

@SpringBootApplication

SpringBoot应用标注在某个类上说明这个类是SpringBoot的主配置类，SpringBoot就会运行这个类的main方法来启动SpringBoot项目。

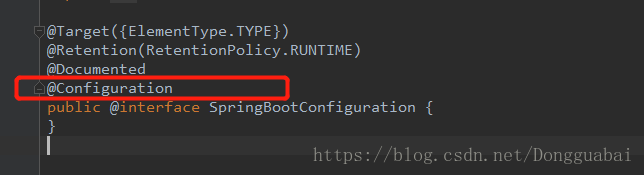
那@SpringBootApplication注解到底是什么呢，点进去看看：



发现@SpringBootApplication是一个组合注解。

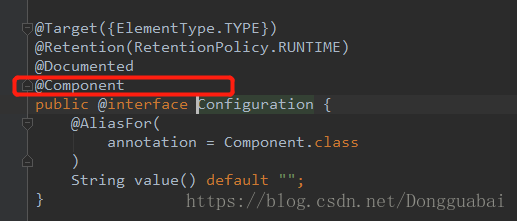
@SpringBootConfiguration

先看看@SpringBootConfiguration注解：



这个注解很简单，表明该类是一个Spring的配置类。

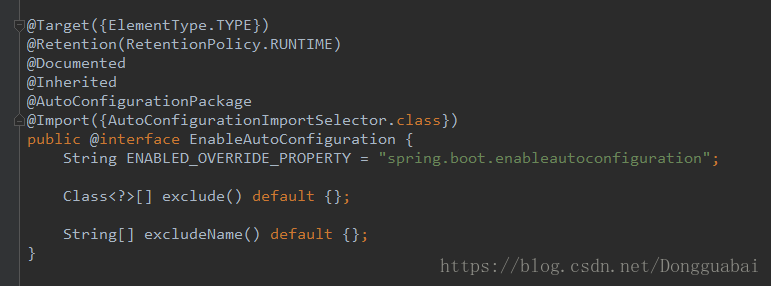
再进去看看@Configuration：



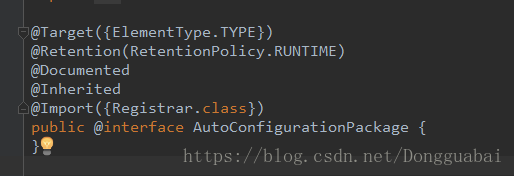
说明Spring的配置类也是Spring的一个组件。

@EnableAutoConfiguration

这个注解是开启自动配置的功能。

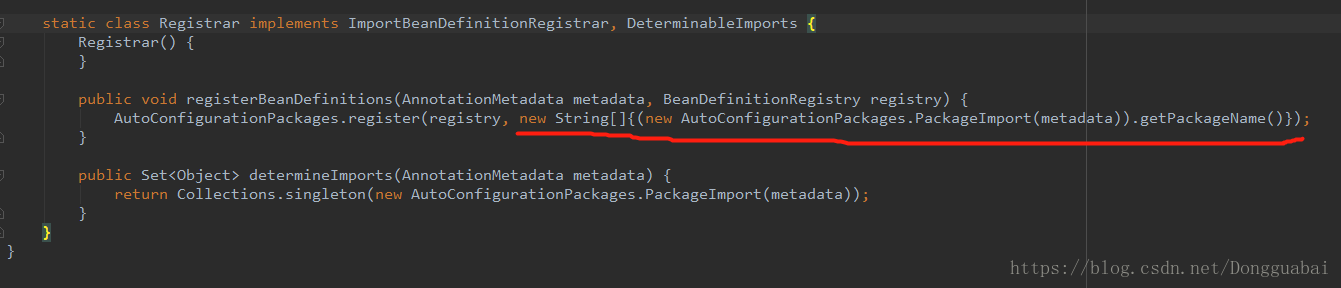


先看看@AutoConfigurationPackage注解：



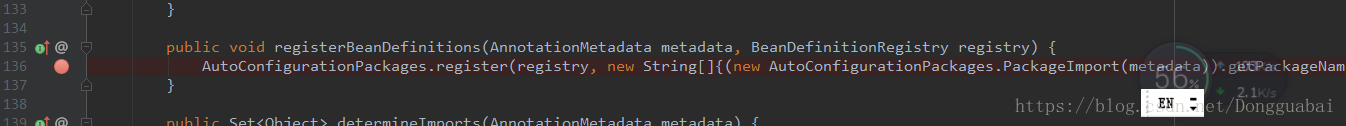
这个注解是自动配置包，主要是使用的@Import来给Spring容器中导入一个组件 ，这里导入的是Registrar.class。

来看下这个Registrar：



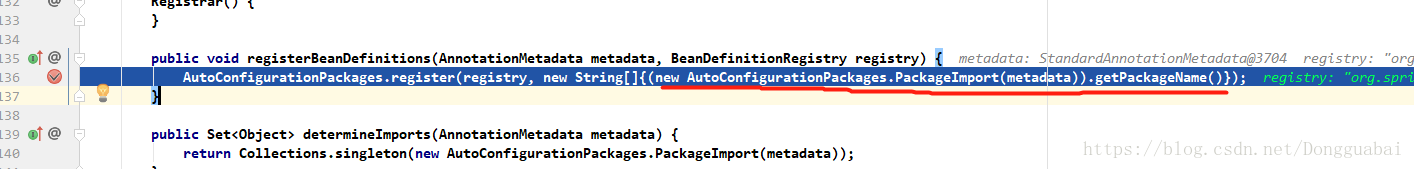
就是通过这个方法获取扫描的包路径，可以debug看看：

在这行代码上打了一个断点：

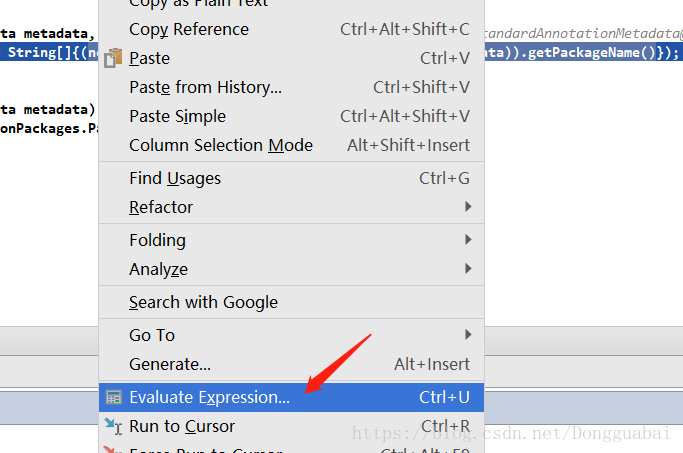


启动项目：

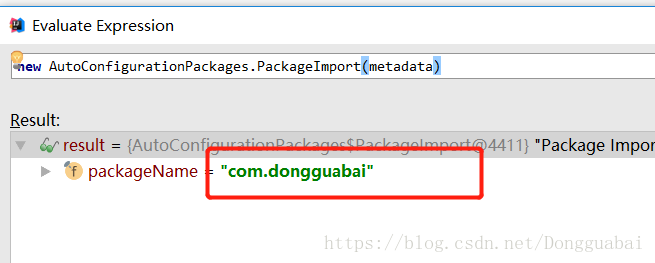
进入断点处：



看看能否获取扫描的包路径：

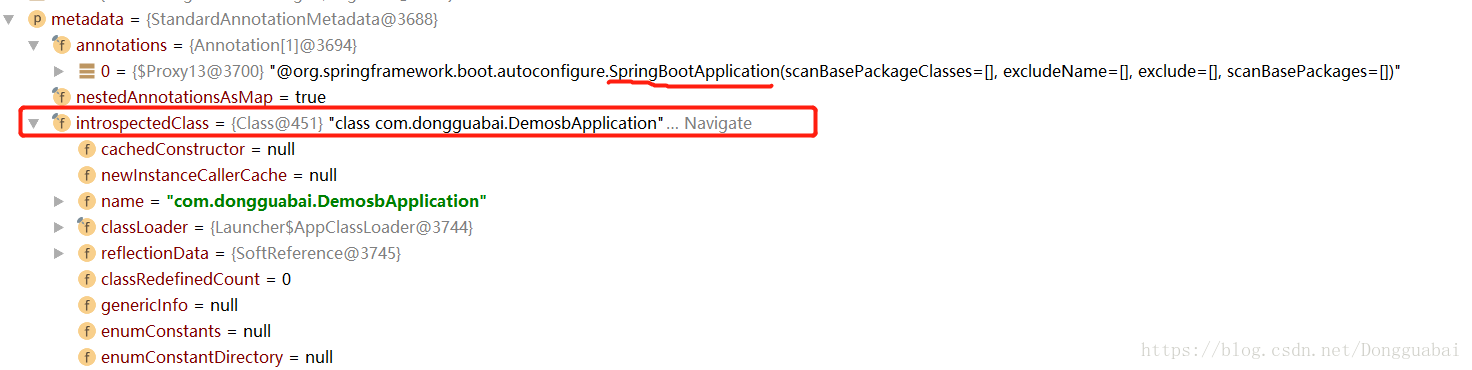


已经获取到了包路径：



那那个metadata是什么呢：

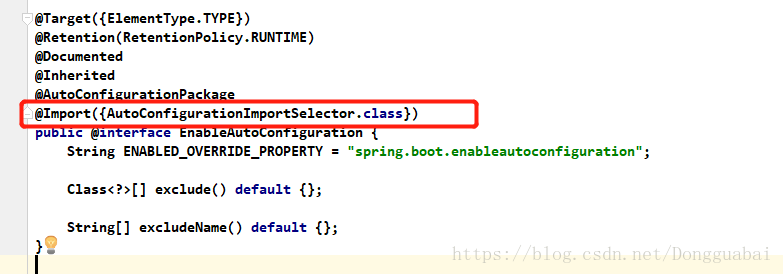
可以看到是标注在@SpringBootApplication注解上的DemosbApplication，也就是我们的主配置类：



说白了就是将主配置类（即@SpringBootApplication标注的类）的所在包及子包里面所有组件扫描加载到Spring容器。所以包名一定要注意。

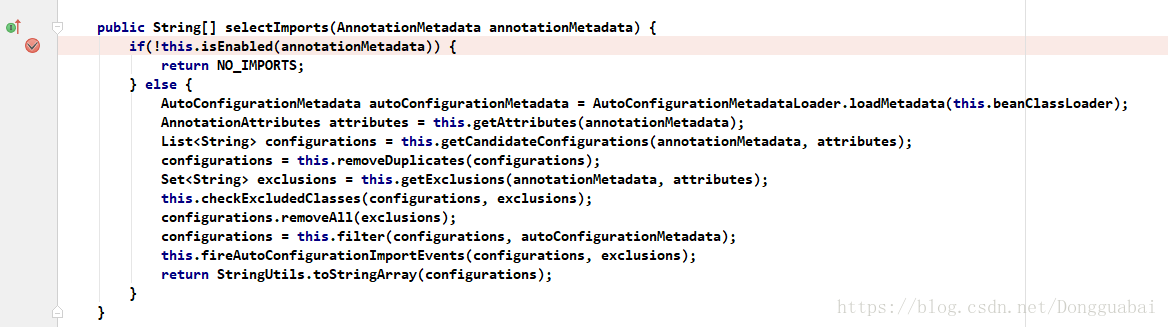
现在包扫描路径获取到了，那具体加载哪些组件呢，看看下面这个注解。

@Import({AutoConfigurationImportSelector.class})



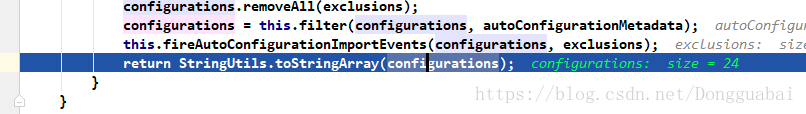
@Import注解就是给Spring容器中导入一些组件，这里传入了一个组件的选择器:AutoConfigurationImportSelector。

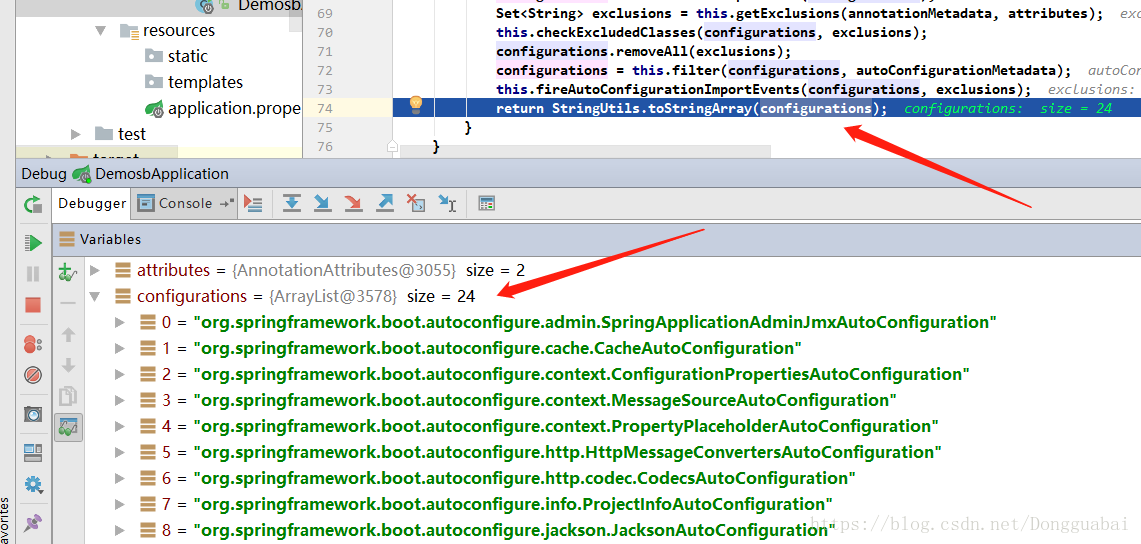
里面有一个selectImports方法，将所有需要导入的组件以全类名的方式返回；这些组件就会被添加到容器中。



debug运行看看：

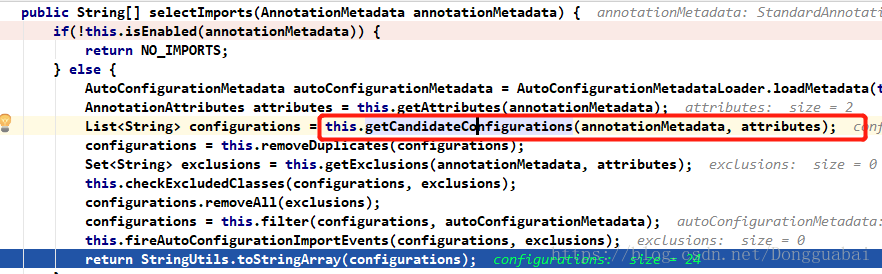
会给容器中导入非常多的自动配置类（xxxAutoConfiguration）；就是给容器中导入这个场景需要的所有组件，并配置好这些组件:

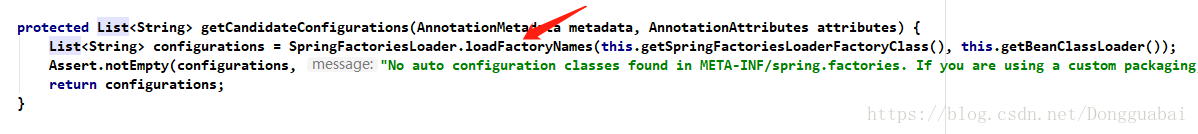


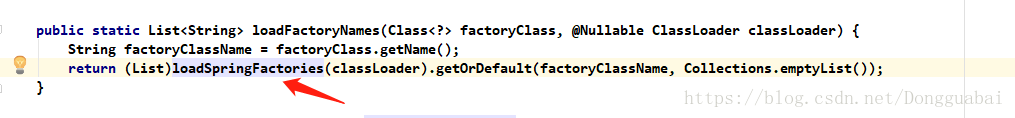


有了自动配置类，免去了我们手动编写配置注入功能组件等的工作。

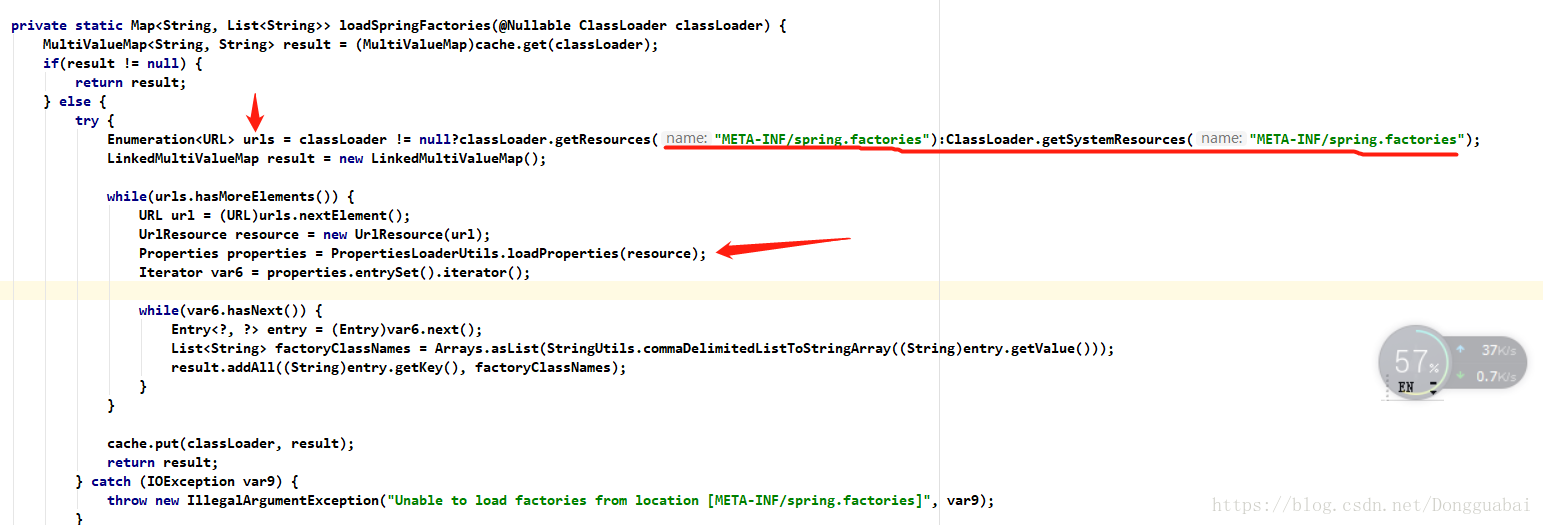
那他是如何获取到这些配置类的呢，看看上面这个方法：



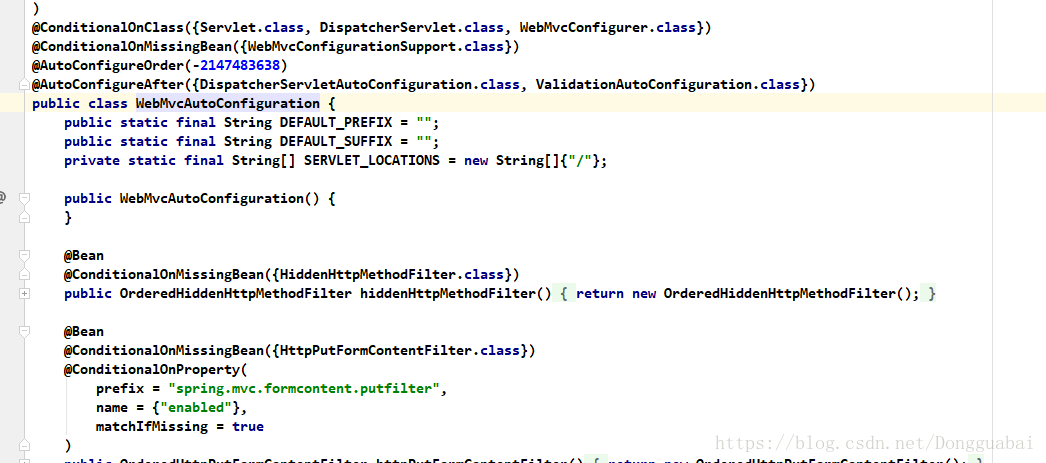




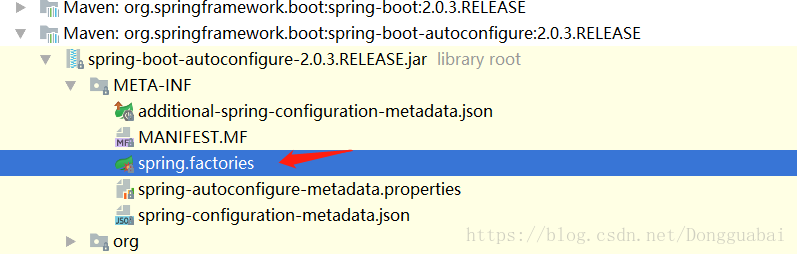
会从META-INF/spring.factories中获取资源，然后通过Properties加载资源：



Spring Boot在启动的时候从类路径下的META-INF/spring.factories中获取EnableAutoConfiguration指定的值，将这些值作为自动配置类导入到容器中，自动配置类就生效，帮我们进行自动配置工作。以前我们需要自己配置的东西，自动配置类都帮我们完成了。



J2EE的整体整合解决方案和自动配置都在spring-boot-autoconfigure-2.0.3.RELEASE.jar：





比如看看WebMvcAutoConfiguration：

都已经帮我们配置好了，我们不用再单独配置了：

