@EnableAutoConfiguration和@SpringbootApplication的区别

Spring Boot采取了这些注解的优点，在@EnableAutoConfiguration注解中的这些注解被用来执行自动配置（auto-configuration）。让我们来近距离的看一下@EnableAutoConfiguration注解并看一看它背后的逻辑。

@Target({ElementType.TYPE})

@Retention(RetentionPolicy.RUNTIME)

@Documented

@Inherited

@AutoConfigurationPackage

@Import({EnableAutoConfigurationImportSelector.class})

public @interface EnableAutoConfiguration {

String ENABLED\_OVERRIDE\_PROPERTY = "spring.boot.enableautoconfiguration";

Class<?>[] exclude() default {};

String[] excludeName() default {};

}

我们知道，这个类会尝试去猜测和配置应用中需要的bean。auto-configuration类是基于classpath（pom.xml的配置）的配置和应用中定义的beans被使用，但是发挥这种魔力的是org.springframework.boot.autoconfigure.EnableAutoConfigurationImportSelector类，该类可以找到所有必须的配置类。

定位main应用类：建议将main应用类放在位于其他类上面的根包（root package） 中。通常使用 **@EnableAutoConfiguration 注解你的main类，** 并且暗地里为某些项定义了一个基础“search package”。 例如， 如果你正在编写一个JPA应用，在被 @EnableAutoConfiguration所注解的类的包中的所有@Entity 注解的项都将会被检索到。

使用root包允许在不需要指定basePackage属性的情况下使用@ComponentScan注解。如果main类位于根包中， 你也可以使用 @SpringBootApplication 注解。

 你可以添加@**SpringBootApplication**或@ComponentScan 注解而不需要任何参数。 你的所有应用程序组件（ @Component , @Service , @Repository , @Controller 等） 将被自动注册为Spring Beans。

很多Spring Boot开发者总是使用 **@Configuration ， @EnableAutoConfiguration 和 @ComponentScan 注解他们的main类。 由于这些注解被如此频繁地一块使用 ， Spring Boot提供一个方便的 @SpringBootApplication** 选择。@SpringBootApplication 注解等价于以默认属性使用 @Configuration ， @EnableAutoConfiguration 和 @ComponentScan 。