Spring Boot AutoConfigure

Spring Boot根据Classpath中的依赖内容自动配置bean到IoC容器。Auto-Configuration会推断哪些bean是用户可能需要的，比如HSQLDB包在当前classpath下，并且用户没有配置其他数据库连接，则Auto-Configure功能会自动注入一个基于内存数据库连接给应用的IOC容器，启自动配置功能需要添加@EanbleAutoConfigure注解。

自动配置中使用的条件化注解如下表所示：

|  |  |
| --- | --- |
| 条件化注解 | 配置生效条件 |
| @ConditionOnBean | 配置了某个特定Bean |
| @ConditionOnMissingBean | 没有配置特定Bean |
| @ConditionOnClass | ClassPath中有指定的类 |
| @ConditionalOnMissingClass | ClassPath中没有指定的类 |
| @ConditionOnExpression | 给定的Spring Expression Language计算式为true |
| @ConditionOnJava | Java的版本匹配特定值或者范围值 |
| @ConditionalOnJndi | 参数中给定的JDNI位置必须存在一个 |
| @ConditionalOnProperty | 指定的配置属性要有一个明确的值 |
| @ConditionalOnResource | Classpath中有指定的资源 |
| @ConditionalOnWebApplication | 这是一个Web程序 |
| @ConditionalOnNotWebApplication | 这不是一个Web程序 |

使用示例如下：

*@Configuration*

*@EnableConfigurationProperties(HelloProperties.class)*

*@ConditionalOnClass(HelloService.class)*

*@ConditionalOnProperty(*

*prefix = "hello",*

*value = "enabled",*

*matchIfMissing = true*

*)*

*public class HelloAutoConfiguration {*

*@Autowired*

*private HelloProperties helloProperties;*

*@Bean*

*@ConditionalOnMissingBean(HelloService.class)*

*public HelloService helloService() {*

*System.out.println(">>>> The HelloService Not Found, Execute Create New Bean.");*

*HelloService helloService = new HelloService();*

*helloService.setMsg(helloProperties.getMsg());*

*helloService.setShow(helloProperties.isShow());*

*return helloService;}}*