https://www.jianshu.com/p/e0b4437547f5

**SpringBoot 的自动化配置**

[BeStronger30](https://www.jianshu.com/u/e6e2cc431352) 已关注

2018.04.18 17:42\* 字数 747 阅读 416评论 1喜欢 0

熟悉SpringBoot的同学，应该知道SpringBoot的自动配置是基于Spring4.0以后的条件化配置，采用约定优于配置，它可以在运行时判断这个配置是否要运用。Spring Boot运用条件化配置的方法是，定义多个特殊的条件化注解。大部分**@ConditionalOnMissingBean注解**是我们要对SpringBoot的自动化配置进行覆盖的关键。下图是SpringBoot中定义的用于自动化配置相关的条件化注解（图片截取自**SpringBoot实战**）：

SpringBoot条件化注解.png

这些注解是如何使用的呢？先看一下SpringBoot自动配置库中的一部分：  
@Configuration  
@ConditionalOnClass({ DataSource.class, EmbeddedDatabaseType.class })  
@EnableConfigurationProperties(DataSourceProperties.class)  
@Import({ Registrar.class, DataSourcePoolMetadataProvidersConfiguration.class })  
public class DataSourceAutoConfiguration {  
...  
}  
使用@Configuration 注解表明它是一个配置类，重要的是使用了**@ConditionalOnClass**注解，要求Classpath里必须要有DataSource和EmbeddedDatabaseType。如果它们不存在，条件就不成立，DataSourceAutoConfiguration提供的配置都会被忽略掉。DataSourceAutoConfiguration 中定义的JdbcTemplate Bean中演示了**@ConditionalOnMissingBean注解**是如何工作的  
@Bean  
@ConditionalOnMissingBean(JdbcOperations.class)  
public JdbcTemplate jdbcTemplate() {  
return new JdbcTemplate(this.dataSource);  
}   
dbcTemplate()方法上添加了@Bean注解，在需要时可以配置出一个JdbcTemplateBean。但它上面还加了@ConditionalOnMissingBean注解，要求当前**不存在JdbcOperations类型**（JdbcTemplate实现了该接口）的Bean时才生效，如果有了该接口的实现Bean，则不会创建。

**怎么覆盖掉自动化配置？什么时候出现JdbcOperations类型的Bean？**

SpringBoot的设计是加载应用级别的配置，如果应用级别没有，则在考虑自动化的配置，如果我们我们自己定义一个JdbcTemplate,那么就会忽略掉自动化配配配置。

**如何覆盖一些需要微调的自动化配置？**

1. 通过属性文件外置配置  
   当我们只需要微调一些细节的时候，比如端口号，设置数据库得Url,我们可以通过只环境变量、 Java系统属性、 JNDI（Java Naming and Directory Interface）、命令行参数或者属性文件里进行指定就好了。 Spring Boot应用程序有多种设置途径。 Spring Boot能从多种属性源获得属性，包括如下几处。  
   (1) 命令行参数  
   (2) java:comp/env里的JNDI属性  
   (3) JVM系统属性  
   (4) 操作系统环境变量  
   (5) 随机生成的带random.\*前缀的属性（在设置其他属性时，可以引用它们，比如${random.  
   long}）  
   (6) 应用程序以外的application.properties或者appliaction.yml文件  
   (7) 打包在应用程序内的application.properties或者appliaction.yml文件  
   (8) 通过@PropertySource标注的属性源  
   (9) 默认属性

这个列表 **按照优先级排序**，也就是说，任何在高优先级属性源里设置的属性都会覆盖低优先级的相同属性，例如，命令行参数会覆盖其他属性源里的属性。application.properties和application.yml文件能放在以下四个位置。  
(1) 外置，在相对于应用程序运行目录的/config子目录里。  
(2) 外置，在应用程序运行的目录里。  
(3) 内置，在config包内。  
(4) 内置，在Classpath根目录  
同样，这个列表**按照优先级排序**。也就是说，**相对于应用程序运行目录 /config**子目录里的application.properties会覆盖应用程序Classpath里的application.properties中的相同属性。  
此外，如果你在**同一优先级位置**同时有application.properties和application.yml，那么**application.  
yml**里的属性**会覆盖application.properties**里的属性。

**参考书籍：**

**Spring In Action (Spring实战，张卫滨译)**  
**SpringBoot In Action(SpringBoot实战，丁雪丰译)**

**小礼物走一走，来简书关注我**