<https://mp.weixin.qq.com/s/vd6MR-UXi-OK2Z-uR15BTQ>

<https://mp.weixin.qq.com/s/dbSBzFICIDPLkj5Tuv2-yA>

# Spring Boot 的两种配置文件

# 理解配置文件 application.properties

在 Spring Boot 中，配置文件有两种不同的格式，一个是 properties ，另一个是 yaml 。

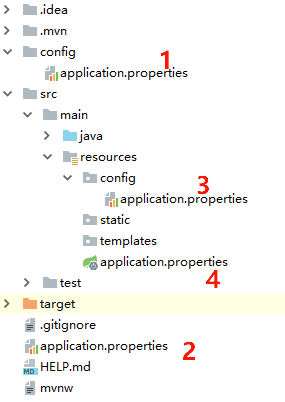
虽然 properties 文件比较常见，但是相对于 properties 而言，yaml 更加简洁明了，而且使用的场景也更多，很多开源项目都是使用 yaml 进行配置（例如 Hexo）。除了简洁，yaml 还有另外一个特点，就是 yaml 中的数据是有序的，properties 中的数据是无序的，在一些需要路径匹配的配置中，顺序就显得尤为重要（例如我们在 Spring Cloud Zuul 中的配置），此时我们一般采用 yaml。关于 yaml ，松哥之前写过一篇文章：

## 位置问题

首先，当我们创建一个 Spring Boot 工程时，默认 resources 目录下就有一个 application.properties 文件，可以在 application.properties 文件中进行项目配置，但是这个文件并非唯一的配置文件，在 Spring Boot 中，一共有 4 个地方可以存放 application.properties 文件。

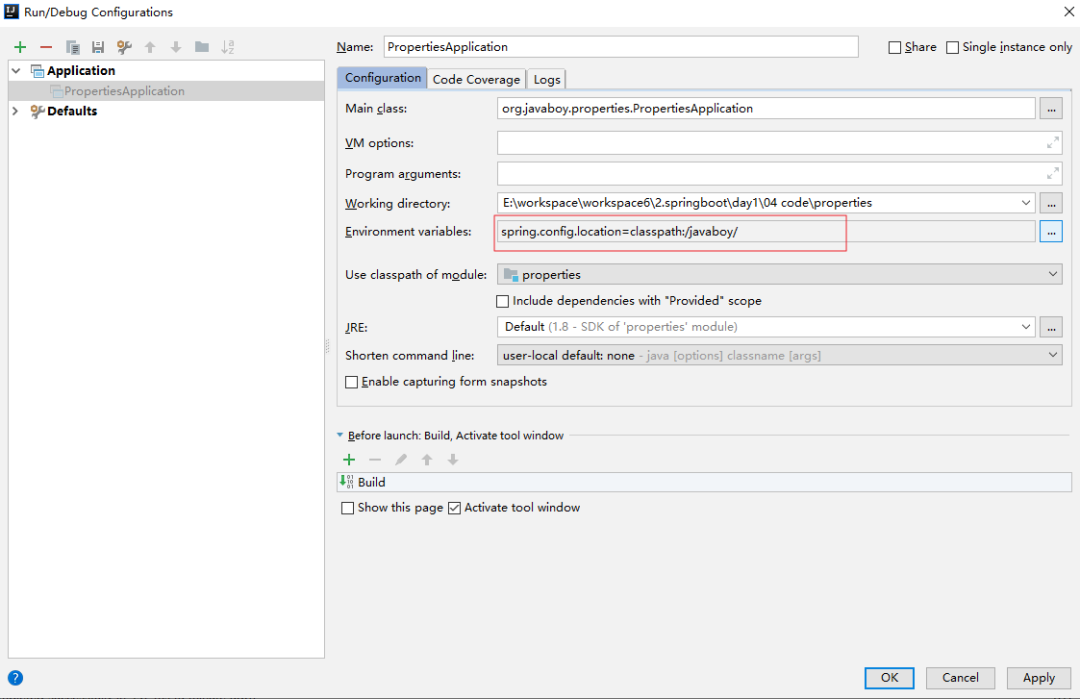
1. 当前项目根目录下的 config 目录下
2. 当前项目的根目录下
3. resources 目录下的 config 目录下
4. resources 目录下

按如上顺序，四个配置文件的优先级依次降低。如下：



这四个位置是默认位置，即 Spring Boot 启动，默认会从这四个位置按顺序去查找相关属性并加载。但是，这也不是绝对的，我们也可以在项目启动时自定义配置文件位置。

例如，现在在 resources 目录下创建一个 javaboy 目录，目录中存放一个 application.properties 文件，那么正常情况下，当我们启动 Spring Boot 项目时，这个配置文件是不会被自动加载的。我们可以通过 spring.config.location 属性来手动的指定配置文件位置，指定完成后，系统就会自动去指定目录下查找 application.properties 文件。



此时启动项目，就会发现，项目以 classpath:/javaboy/application.propertie 配置文件启动。

这是在开发工具中配置了启动位置，如果项目已经打包成 jar ，在启动命令中加入位置参数即可：

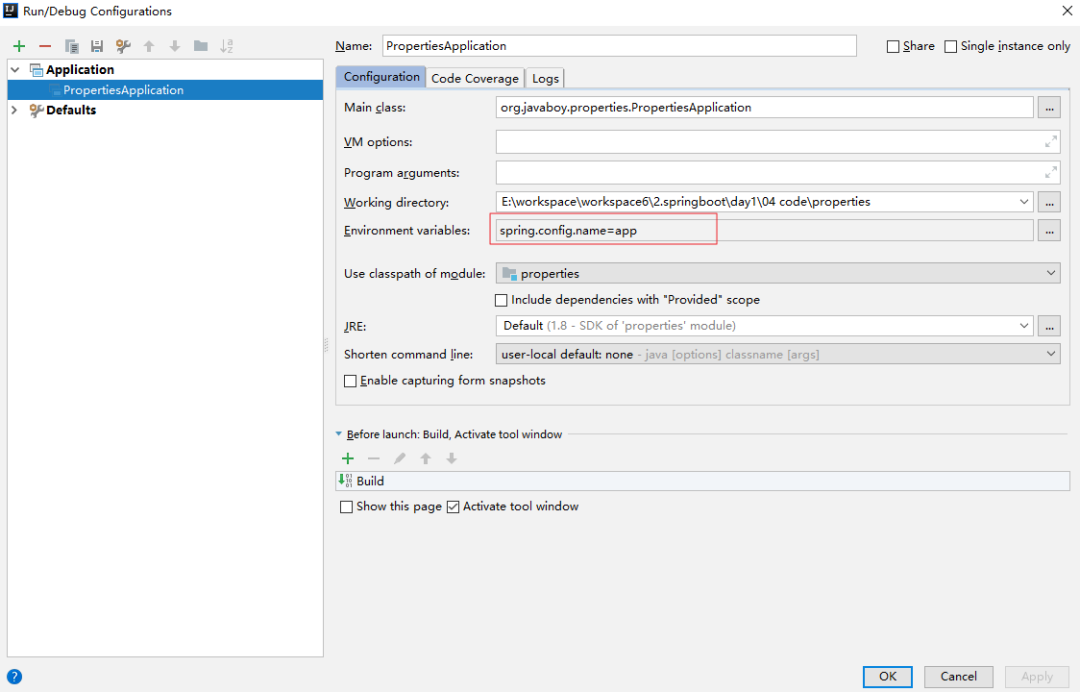


## 文件名问题

对于 application.properties 而言，它不一定非要叫 application ，但是项目默认是去加载名为 application 的配置文件，如果我们的配置文件不叫 application ，也是可以的，但是，需要明确指定配置文件的文件名。

方式和指定路径一致，只不过此时的 key 是 spring.config.name 。

首先我们在 resources 目录下创建一个 app.properties 文件，然后在 IDEA 中指定配置文件的文件名：



指定完配置文件名之后，再次启动项目，此时系统会自动去默认的四个位置下面分别查找名为 app.properties 的配置文件。当然，允许自定义文件名的配置文件不放在四个默认位置，而是放在自定义目录下，此时就需要明确指定 spring.config.location 。

配置文件位置和文件名称可以同时自定义。

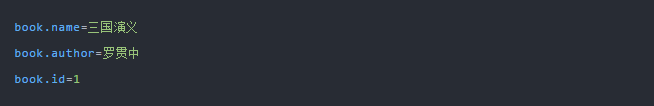
## 普通的属性注入

由于 Spring Boot 源自 Spring ，所以 Spring 中存在的属性注入，在 Spring Boot 中一样也存在。由于 Spring Boot 中，默认会自动加载 application.properties 文件，所以简单的属性注入可以直接在这个配置文件中写。

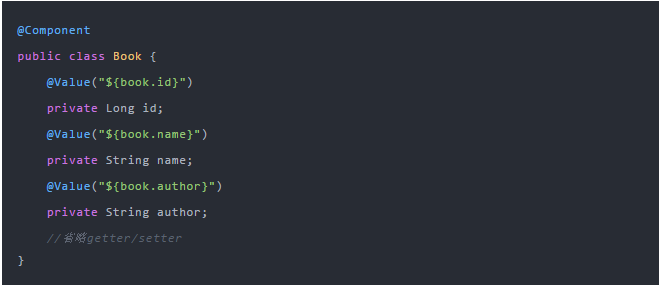
例如，现在定义一个 Book 类：



然后，在 application.properties 文件中定义属性：



按照传统的方式（Spring中的方式），可以直接通过 @Value 注解将这些属性注入到 Book 对象中：



**注意**

Book 对象本身也要交给 Spring 容器去管理，如果 Book 没有交给 Spring 容器，那么 Book 中的属性也无法从 Spring 容器中获取到值。

配置完成后，在 Controller 或者单元测试中注入 Book 对象，启动项目，就可以看到属性已经注入到对象中了。

一般来说，我们在 application.properties 文件中主要存放系统配置，这种自定义配置不建议放在该文件中，可以自定义 properties 文件来存在自定义配置。

例如在 resources 目录下，自定义 book.properties 文件，内容如下：



此时，项目启动并不会自动的加载该配置文件，如果是在 XML 配置中，可以通过如下方式引用该 properties 文件：



如果是在 Java 配置中，可以通过 @PropertySource 来引入配置：



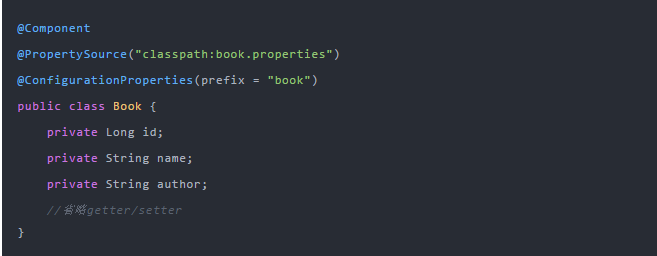
这样，当项目启动时，就会自动加载 book.properties 文件。

这只是 Spring 中属性注入的一个简单用法，和 Spring Boot 没有任何关系。

## 类型安全的属性注入

Spring Boot 引入了类型安全的属性注入，如果采用 Spring 中的配置方式，当配置的属性非常多的时候，工作量就很大了，而且容易出错。

使用类型安全的属性注入，可以有效的解决这个问题。



这里，主要是引入 @ConfigurationProperties(prefix = "book") 注解，并且配置了属性的前缀，此时会自动将 Spring 容器中对应的数据注入到对象对应的属性中，就不用通过 @Value 注解挨个注入了，减少工作量并且避免出错。

## 总结

application.properties 是 Spring Boot 中配置的一个重要载体，很多组件的属性都可以在这里定制。它的用法和 yaml 比较类似，关于 yaml 配置，大家可以参考 [**Spring Boot 中的 yaml 配置简介**](https://mp.weixin.qq.com/s?__biz=MzI1NDY0MTkzNQ==&mid=2247485055&idx=1&sn=54ebd48d58eee3b36c107a73bdc9f13f&scene=21#wechat_redirect) 。

# Spring Boot中的yaml配置简介

搞Spring Boot的小伙伴都知道，Spring Boot中的配置文件有两种格式，properties或者yaml，一般情况下，两者可以随意使用，选择自己顺手的就行了，那么这两者完全一样吗？肯定不是啦！本文就来和大伙重点介绍下yaml配置，最后再来看看yaml和properties配置有何区别。

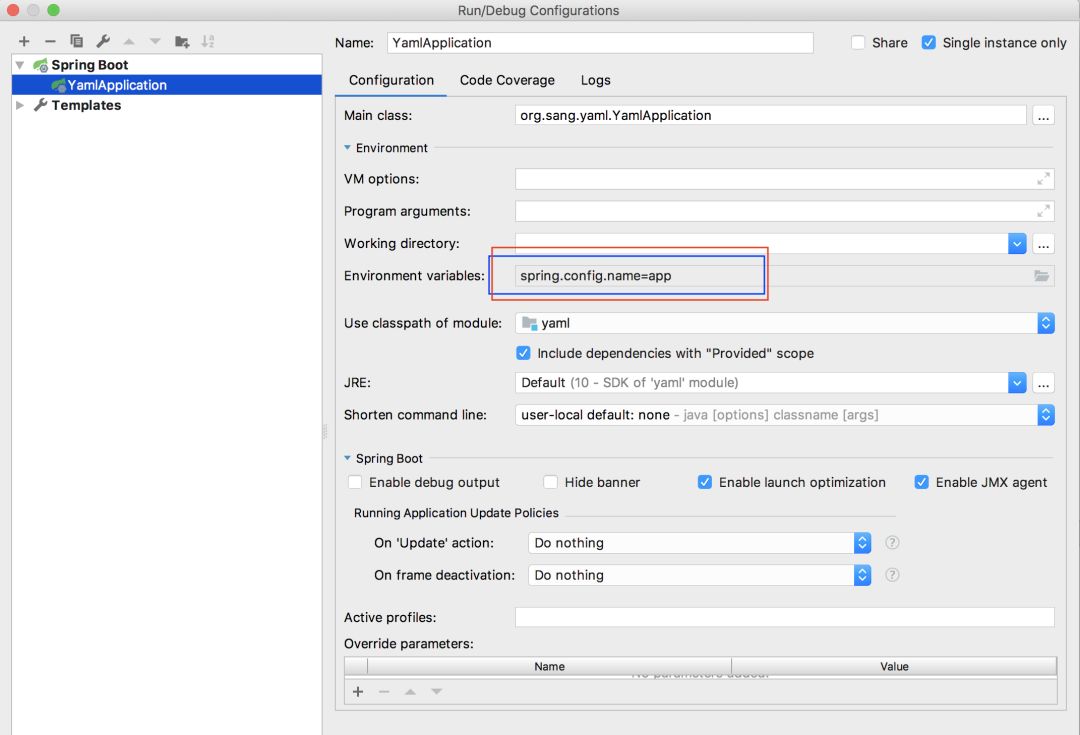
#### 狡兔三窟

首先application.yaml在Spring Boot中可以写在四个不同的位置，分别是如下位置：

1. 项目根目录下的config目录中
2. 项目根目录下
3. classpath下的config目录中
4. classpath目录下

四个位置中的application.yaml文件的优先级按照上面列出的顺序依次降低。即如果有同一个属性在四个文件中都出现了，以优先级高的为准。

那么application.yaml是不是必须叫application.yaml这个名字呢？当然不是必须的。开发者可以自己定义yaml名字，自己定义的话，需要在项目启动时指定配置文件的名字，像下面这样：

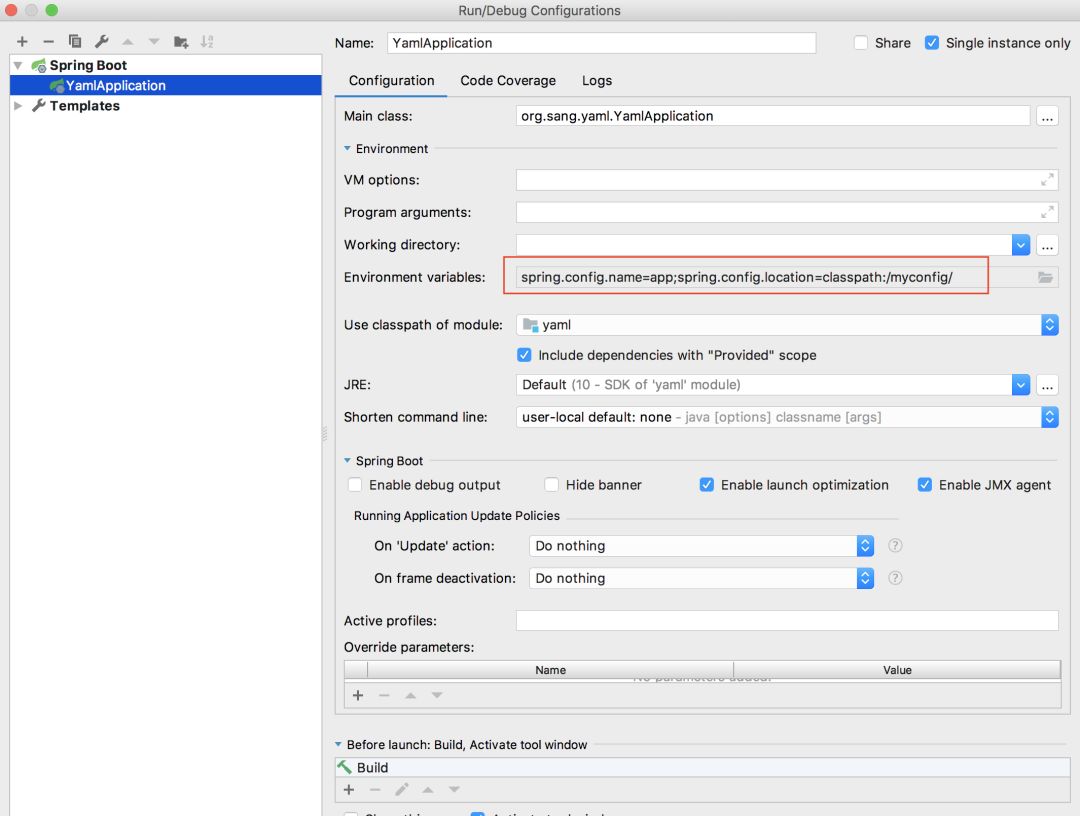


当然这是在IntelliJ IDEA中直接配置的，如果项目已经打成jar包了，则在项目启动时加入如下参数：



这样配置之后，在项目启动时，就会按照上面所说的四个位置按顺序去查找一个名为app.yaml的文件。当然这四个位置也不是一成不变的，也可以自己定义，有两种方式，一个是使用 spring.config.location属性，另一个则是使用 spring.config.additional-location这个属性，在第一个属性中，表示自己重新定义配置文件的位置，项目启动时就按照定义的位置去查找配置文件，这种定义方式会覆盖掉默认的四个位置，也可以使用第二种方式，第二种方式则表示在四个位置的基础上，再添加几个位置，**新添加的位置的优先级大于原本的位置。**

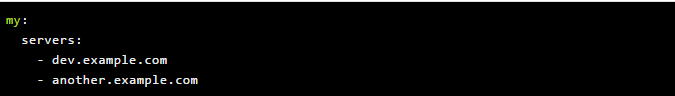
配置方式如下：



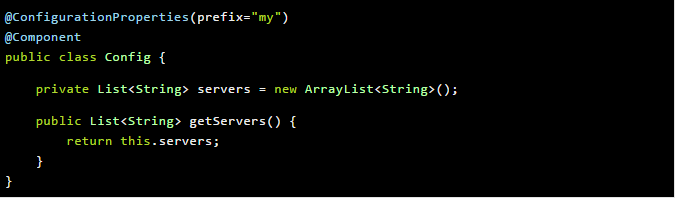
**这里要注意，配置文件位置时，值一定要以/结尾。**

#### 数组注入

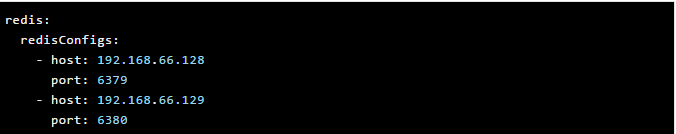
yaml也支持数组注入，例如



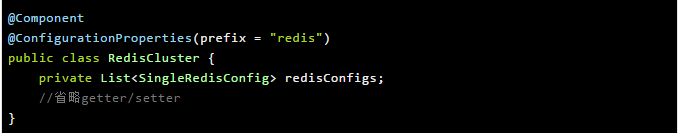
这段数据可以绑定到一个带Bean的数组中：



项目启动后，配置中的数组会自动存储到servers集合中。当然，yaml不仅可以存储这种简单数据，也可以在集合中存储对象。例如下面这种：



这个可以被注入到如下类中：



#### 优缺点

不同于properties文件的无序，yaml配置是有序的，这一点在有些配置中是非常有用的，例如在Spring Cloud Zuul的配置中，当我们配置代理规则时，顺序就显得尤为重要了。当然yaml配置也不是万能的，例如，**yaml配置目前不支持@PropertySource注解**。

好了，关于YAML的配置就先说这么多，有问题欢迎留言讨论。