## 使用随机数

在一些情况下，有些参数我们需要希望它不是一个固定的值，比如密钥、服务端口等。Spring Boot的属性配置文件中可以通过${random}来产生int值、long值或者string字符串，来支持属性的随机值。

# 随机字符串

com.didispace.blog.value=${random.value}

# 随机int

com.didispace.blog.number=${random.int}

# 随机long

com.didispace.blog.bignumber=${random.long}

# 10以内的随机数

com.didispace.blog.test1=${random.int(10)}

# 10-20的随机数

com.didispace.blog.test2=${random.int[10,20]}

## 通过命令行设置属性值

相信使用过一段时间Spring Boot的用户，一定知道这条命令：java -jar xxx.jar --server.port=8888，通过使用–server.port属性来设置xxx.jar应用的端口为8888。

在命令行运行时，连续的两个减号--就是对application.properties中的属性值进行赋值的标识。所以，java -jar xxx.jar --server.port=8888命令，等价于我们在application.properties中添加属性server.port=8888，该设置在样例工程中可见，读者可通过删除该值或使用命令行来设置该值来验证。

通过命令行来修改属性值固然提供了不错的便利性，但是通过命令行就能更改应用运行的参数，那岂不是很不安全？是的，所以Spring Boot也贴心的提供了屏蔽命令行访问属性的设置，只需要这句设置就能屏蔽：SpringApplication.setAddCommandLineProperties(false)。

## 多环境配置

我们在开发Spring Boot应用时，通常同一套程序会被应用和安装到几个不同的环境，比如：开发、测试、生产等。其中每个环境的数据库地址、服务器端口等等配置都会不同，如果在为不同环境打包时都要频繁修改配置文件的话，那必将是个非常繁琐且容易发生错误的事。

对于多环境的配置，各种项目构建工具或是框架的基本思路是一致的，通过配置多份不同环境的配置文件，再通过打包命令指定需要打包的内容之后进行区分打包，Spring Boot也不例外，或者说更加简单。

在Spring Boot中多环境配置文件名需要满足application-{profile}.properties的格式，其中{profile}对应你的环境标识，比如：

* application-dev.properties：开发环境
* application-test.properties：测试环境
* application-prod.properties：生产环境

至于哪个具体的配置文件会被加载，需要在application.properties文件中通过spring.profiles.active属性来设置，其值对应{profile}值。

如：spring.profiles.active=test就会加载application-test.properties配置文件内容

下面，以不同环境配置不同的服务端口为例，进行样例实验。

* 针对各环境新建不同的配置文件application-dev.properties、application-test.properties、application-prod.properties
* 在这三个文件均都设置不同的server.port属性，如：dev环境设置为1111，test环境设置为2222，prod环境设置为3333
* application.properties中设置spring.profiles.active=dev，就是说默认以dev环境设置
* 测试不同配置的加载
  + 执行java -jar xxx.jar，可以观察到服务端口被设置为1111，也就是默认的开发环境（dev）
  + 执行java -jar xxx.jar --spring.profiles.active=test，可以观察到服务端口被设置为2222，也就是测试环境的配置（test）
  + 执行java -jar xxx.jar --spring.profiles.active=prod，可以观察到服务端口被设置为3333，也就是生产环境的配置（prod）

按照上面的实验，可以如下总结多环境的配置思路：

* application.properties中配置通用内容，并设置spring.profiles.active=dev，以开发环境为默认配置
* application-{profile}.properties中配置各个环境不同的内容
* 通过命令行方式去激活不同环境的配置