# ****Spring Boot 属性配置和使用****

Spring Boot 允许通过外部配置让你在不同的环境使用同一应用程序的代码，简单说就是可以通过配置文件来注入属性或者修改默认的配置。

## ****Spring Boot 系列****

[Spring Boot 入门](http://blog.csdn.net/isea533/article/details/50278205" \t "http://blog.csdn.net/isea533/article/details/_blank)

[Spring Boot 属性配置和使用](http://blog.csdn.net/isea533/article/details/50281151" \t "http://blog.csdn.net/isea533/article/details/_blank)

[Spring Boot 集成MyBatis](http://blog.csdn.net/isea533/article/details/50359390" \t "http://blog.csdn.net/isea533/article/details/_blank)

[Spring Boot 静态资源处理](http://blog.csdn.net/isea533/article/details/50412212" \t "http://blog.csdn.net/isea533/article/details/_blank)

[Spring Boot - 配置排序依赖技巧](http://blog.csdn.net/isea533/article/details/53975720" \t "http://blog.csdn.net/isea533/article/details/_blank)

[Spring Boot - DevTools 介绍](http://blog.csdn.net/isea533/article/details/70495714" \t "http://blog.csdn.net/isea533/article/details/_blank)

## ****Spring Boot 支持多种外部配置方式****

这些方式优先级如下：

命令行参数

来自java:comp/env的JNDI属性

Java系统属性（System.getProperties()）

操作系统环境变量

RandomValuePropertySource配置的random.\*属性值

jar包外部的application-{profile}.properties或application.yml(带spring.profile)配置文件

jar包内部的application-{profile}.properties或application.yml(带spring.profile)配置文件

jar包外部的application.properties或application.yml(不带spring.profile)配置文件

jar包内部的application.properties或application.yml(不带spring.profile)配置文件

@Configuration注解类上的@PropertySource

通过SpringApplication.setDefaultProperties指定的默认属性

### ****命令行参数****

通过java -jar app.jar --name="Spring" --server.port=9090方式来传递参数。

参数用--xxx=xxx的形式传递。

可以使用的参数可以是我们自己定义的，也可以是Spring Boot中默认的参数。

很多人可能会关心如web端口如何配置这样的问题，这些都是Spring Boot中提供的参数，部分可用参数如下：

# LOGGING

logging.path=/var/logs

logging.file=myapp.log

logging.config= # location of config file (default classpath:logback.xml for logback)

logging.level.\*= # levels for loggers, e.g. "logging.level.org.springframework=DEBUG" (TRACE, DEBUG, INFO, WARN, ERROR, FATAL, OFF)

# EMBEDDED SERVER CONFIGURATION (ServerProperties)

server.port=8080

server.address= # bind to a specific NIC

server.session-timeout= # session timeout in seconds

server.context-parameters.\*= # Servlet context init parameters, e.g. server.context-parameters.a=alpha

server.context-path= # the context path, defaults to '/'

server.servlet-path= # the servlet path, defaults to '/'

更多常见的应用属性请浏览[这里](http://docs.spring.io/spring-boot/docs/1.2.3.RELEASE/reference/html/common-application-properties.html" \t "http://blog.csdn.net/isea533/article/details/_blank)

**注意：**命令行参数在app.jar的后面！

可以通过SpringApplication.setAddCommandLineProperties(false)禁用命令行配置。

### ****Java系统属性****

注意Java系统属性位置java -Dname="isea533" -jar app.jar，可以配置的属性都是一样的，优先级不同。

例如java -Dname="isea533" -jar app.jar --name="Spring!"中name值为Spring!

### ****操作系统环境变量****

配置过JAVA\_HOME的应该都了解这一个。

这里需要注意的地方，有些OS可以不支持使用.这种名字，如server.port，这种情况可以使用SERVER\_PORT来配置。

具体名字如何匹配，看本文后面。

### ****RandomValuePropertySource****

系统中用到随机数的地方，例如：

my.secret=${random.value}

my.number=${random.int}

my.bignumber=${random.long}

my.number.less.than.ten=${random.int(10)}

my.number.in.range=${random.int[1024,65536]}

random.int\*支持value参数和,max参数，当提供max参数的时候，value就是最小值。

### ****应用配置文件（.properties或.yml）****

在配置文件中直接写：

name=Isea533

server.port=8080

.yml格式的配置文件如：

name: Isea533server:

port: 8080

当有前缀的情况下，使用.yml格式的配置文件更简单。关于.yml配置文件用法请看[这里](http://docs.spring.io/spring-boot/docs/1.2.3.RELEASE/reference/html/boot-features-external-config.html" \l "boot-features-external-config-yaml" \t "http://blog.csdn.net/isea533/article/details/_blank)

**注意：**使用.yml时，属性名的值和冒号中间必须有空格，如name: Isea533正确，name:Isea533就是错的。

#### ****属性配置文件的位置****

spring会从classpath下的/config目录或者classpath的根目录查找application.properties或application.yml。

/config优先于classpath根目录

### ****@PropertySource****

这个注解可以指定具体的属性配置文件，优先级比较低。

### ****SpringApplication.setDefaultProperties****

例如：

SpringApplication application = new SpringApplication(Application.class);

Map<String, Object> defaultMap = new HashMap<String, Object>();

defaultMap.put("name", "Isea-Blog");//还可以是Properties对象

application.setDefaultProperties(defaultMap);

application.run(args);

## ****应用（使用）属性****

### ****@Value(“${xxx}”)****

这种方式是最简单的，通过@Value注解可以将属性值注入进来。

### ****@ConfigurationProperties****

Spring Boot 可以方便的将属性注入到一个配置对象中。例如：

my.name=Isea533

my.port=8080

my.servers[0]=dev.bar.com

my.servers[1]=foo.bar.com

对应对象：

@ConfigurationProperties(prefix="my")public class Config {

private String name;

private Integer port;

private List<String> servers = new ArrayList<String>();

public String geName(){

return this.name;

}

public Integer gePort(){

return this.port;

}

public List<String> getServers() {

return this.servers;

}

}

Spring Boot 会自动将prefix="my"前缀为my的属性注入进来。

Spring Boot 会自动转换类型，当使用List的时候需要注意在配置中对List进行初始化！

Spring Boot 还支持嵌套属性注入，例如：

name=isea533

jdbc.username=root

jdbc.password=root...

对应的配置类：

@ConfigurationPropertiespublic class Config {

private String name;

private Jdbc jdbc;

class Jdbc {

private String username;

private String password;

//getter...

}

public Integer gePort(){

return this.port;

}

public Jdbc getJdbc() {

return this.jdbc;

}

}

jdbc开头的属性都会注入到Jdbc对象中。

### ****在@Bean方法上使用@ConfigurationProperties****

例如：

@ConfigurationProperties(prefix = "foo")@Beanpublic FooComponent fooComponent() {

...

}

Spring Boot 会将foo开头的属性按照名字匹配注入到FooComponent对象中。

## ****属性占位符****

例如：

app.name=MyApp

app.description=${app.name} is a Spring Boot application

可以在配置文件中引用****前面****配置过的属性（优先级前面配置过的这里都能用）。

通过如${app.name:默认名称}方法还可以设置默认值，当找不到引用的属性时，会使用默认的属性。

由于${}方式会被Maven处理。如果你pom继承的spring-boot-starter-parent，Spring Boot 已经将maven-resources-plugins默认的${}方式改为了@ @方式，例如@name@。

如果你是引入的Spring Boot，你可以修改使用[其他的分隔符](http://maven.apache.org/plugins/maven-resources-plugin/resources-mojo.html" \l "delimiters" \t "http://blog.csdn.net/isea533/article/details/_blank)

### ****通过属性占位符还能缩短命令参数****

例如修改web默认端口需要使用--server.port=9090方式，如果在配置中写上：

server.port=${port:8080}

* 1

那么就可以使用更短的--port=9090，当不提供该参数的时候使用默认值8080。

## ****属性名匹配规则****

例如有如下配置对象：

@Component@ConfigurationProperties(prefix="person")public class ConnectionSettings {

private String firstName;

}

firstName可以使用的属性名如下：

1. person.firstName，标准的驼峰式命名
2. person.first-name，虚线（-）分割方式，推荐在.properties和.yml配置文件中使用
3. PERSON\_FIRST\_NAME，大写下划线形式，建议在系统环境变量中使用

## ****属性验证****

可以使用JSR-303注解进行验证，例如：

@Component@ConfigurationProperties(prefix="connection")public class ConnectionSettings {

@NotNull

private InetAddress remoteAddress;

// ... getters and setters

* }

## ****最后****

以上是Spring Boot 属性配置和使用的内容，有些不全面的地方或者读者有更多疑问，可以查看[Spring Boot完整文档](http://docs.spring.io/spring-boot/docs/1.2.3.RELEASE/reference/html/index.html" \t "http://blog.csdn.net/isea533/article/details/_blank) 或 [Externalized Configuration](http://docs.spring.io/spring-boot/docs/1.2.3.RELEASE/reference/html/boot-features-external-config.html" \t "http://blog.csdn.net/isea533/article/details/_blank)。

关于Spring Boot更多的内容可以继续关注本博客。