

狂神说SpringBoot04：JSR303数据校验及多环境切换

秦疆 狂神说 2020-03-11

狂神说SpringBoot系列连载课程，通俗易懂，基于SpringBoot2.2.5版本，欢迎各位狂粉转发关注学习。未经作者授权，禁止转载

JSR303数据校验

先看看如何使用

Springboot中可以用@validated来校验数据，如果数据异常则会统一抛出异常，方便异常中心统一处理。我们这里来写个注解让我们的name只能支持Email格式；

```
@Component //注册bean
@ConfigurationProperties(prefix = "person")@Validated
//数据校验
public class Person {
    @Email(message="邮箱格式错误") //name必须是邮箱格式
    private String name;
}
```

运行结果：default message [不是一个合法的电子邮件地址];

```
org.springframework.boot.context.properties.bind.validation.BindValidationException: Binding validation errors
in object 'person' on field 'name': rejected value [qinjiang]; codes [Email.person.name,Email.name,Email]
;framework.boot.context.properties.bind.validation.ValidationBindHandler.getBindValidationException(Validat
```

使用数据校验，可以保证数据的正确性；

常见参数

```
@NotNull(message="名字不能为空")private String userName;@Max(value=120,message="年龄最大不能查过120")private int age;@Email(message="邮箱格式错误")private String email;
```

空检查@Null 验证对象是否为null@NotNull 验证对象是否不为null, 无法查检长度为0的字符串@NotBlank 检查约束字符串是不是Null还有被Trim的长度是否大于0,只对字符串,且会去掉前后空格.@NotEmpty 检查约束元素是否为NULL或者是EMPTY. Boolean检查@AssertTrue 验证 Boolean 对象是否为 true @AssertFalse 验证 Boolean 对象是否为 false

长度检查@Size(min=, max=) 验证对象 (Array,Collection,Map,String) 长度是否在给定的范围之内 @Length(min=, max=) string is between min and max included.

日期检查@Past 验证 Date 和 Calendar 对象是否在当前时间之前 @Future 验证 Date 和 Calendar 对象是否在当前时间之后 @Pattern 验证 String 对象是否符合正则表达式的规则

.....等等除此以外,我们还可以自定义一些数据校验规则

多环境切换

profile是Spring对不同环境提供不同配置功能的支持,可以通过激活不同的环境版本,实现快速切换环境;

多配置文件

我们在主配置文件编写的时候,文件名可以是 application-{profile}.properties/yml,用来指定多个环境版本;

例如:

application-test.properties 代表测试环境配置

application-dev.properties 代表开发环境配置

但是Springboot并不会直接启动这些配置文件，它默认使用**application.properties**主配置文件；

我们需要通过一个配置来选择需要激活的环境：

```
#比如在配置文件中指定使用dev环境，我们可以通过设置不同的端口号进行测试；#我们启动SpringBoot，就可以看到已经切换到dev下的配置了；spring.profiles.active=dev
```

yaml的多文档块

和properties配置文件中一样，但是使用yaml去实现不需要创建多个配置文件，更加方便了！

```
server:  port: 8081#选择要激活那个环境块spring:  profiles:    active: prod
---server:  port: 8083spring:  profiles: dev #配置环境的名称

---
server:  port: 8084spring:  profiles: prod #配置环境的名称
```

注意：如果yaml和properties同时都配置了端口，并且没有激活其他环境，默认会使用properties配置文件的！

配置文件加载位置

外部加载配置文件的方式十分多，我们选择最常用的即可，在开发的资源文件中进行配置！

Config locations are searched in reverse order. By default, the configured locations are `classpath:/,classpath:/config/,file:./,file:./config/`. The resulting search order is the following:

1. `file:./config/`
2. `file:./`
3. `classpath:/config/`
4. `classpath:/`



springboot 启动会扫描以下位置的`application.properties`或者`application.yml`文件作为Spring boot的默认配置文件:

优先级1: 项目路径下的`config`文件夹配置文件
优先级2: 项目路径下配置文件
优先级3: 资源路径下的`config`文件夹配置文件
优先级4: 资源路径下配置文件

优先级由高到底, 高优先级的配置会覆盖低优先级的配置;

SpringBoot会从这四个位置全部加载主配置文件; 互补配置;

我们在最低级的配置文件中设置一个项目访问路径的配置来测试互补问题;

```
#配置项目的访问路径server.servlet.context-path=/kuang
```

拓展, 运维小技巧

指定位置加载配置文件

我们还可以通过`spring.config.location`来改变默认的配置文件位置

项目打包好以后, 我们可以使用命令行参数的形式, 启动项目的时候来指定配置文件的新位置; 这种情况, 一般是后期运维做的多, 相同配置, 外部指定的配置文件优先级最高

```
java -jar spring-boot-config.jar --spring.config.location=F:/application.properties
```



🔊 长按关注

据说关注小狂神的人都
走向人生巅峰了，还不
长按关注一下？

×

×

