

04、配置文件

1、文件类型

1.1、properties

同以前的properties用法

1.2、yaml

1.2.1、简介

YAML 是 "YAML Ain't Markup Language" (YAML 不是一种标记语言) 的递归缩写。在开发的这种语言时，YAML 的意思其实是："Yet Another Markup Language" (仍是一种标记语言)。

非常适合用来做以数据为中心的配置文件

1.2.2、基本语法

- key: value; kv之间有空格
- 大小写敏感
- 使用缩进表示层级关系
- 缩进不允许使用tab, 只允许空格
- 缩进的空格数不重要, 只要相同层级的元素左对齐即可
- '#'表示注释
- 字符串无需加引号, 如果要加, "与"表示字符串内容 会被 转义/不转义

如 /n 表示换行, 如果使用 '/'n'则转义该字符串, 直接输出/n这个字符串, 而不是输出换行;
如果使用"/n"则表示不转义该字符串, 直接输出换行;

1.2.3、数据类型

- 字面量: 单个的、不可再分的值。date、boolean、string、number、null

```
1 k: v
```

- 对象: 键值对的集合。map、hash、set、object

```
1 行内写法: k: {k1:v1,k2:v2,k3:v3}
2 #或
3 k:
4   k1: v1
5   k2: v2
6   k3: v3
```

- 数组: 一组按次序排列的值。array、list、queue

```
1 行内写法: k: [v1,v2,v3]
2 #或者
3 k:
4   - v1
5   - v2
6   - v3
```

1.2.4、示例

```
1 @Data
2 public class Person {
```

```
3
4     private String userName;
5     private Boolean boss;
6     private Date birth;
7     private Integer age;
8     private Pet pet;
9     private String[] interests;
10    private List<String> animal;
11    private Map<String, Object> score;
12    private Set<Double> salarys;
13    private Map<String, List<Pet>> allPets;
14 }
15
16 @Data
17 public class Pet {
18     private String name;
19     private Double weight;
20 }
```

```
1 # yaml表示以上对象
2 person:
3   userName: zhangsan
4   boss: false
5   birth: 2019/12/12 20:12:33
6   age: 18
7   pet:
8     name: tomcat
9     weight: 23.4
10  interests: [篮球, 游泳]
11  animal:
12    - jerry
13    - mario
14  score:
15    english:
16      first: 30
17      second: 40
18      third: 50
19    math: [131,140,148]
20    chinese: {first: 128,second: 136}
21  salarys: [3999,4999.98,5999.99]
22  allPets:
23    sick:
24      - {name: tom}
25      - {name: jerry,weight: 47}
26  health: [{name: mario,weight: 47}]
```

2、配置提示

自定义的类和配置文件绑定一般没有提示。


当我们将自定义的pojo与配置文件进行绑定时，可以通过配置文件来为pojo赋值，但是配置文件中不会像其他默认属性一样有提示，这是因为没有配置处理器，因此我们需要添加该依赖spring-boot-configuration-processor

```
1     <dependency>
2       <groupId>org.springframework.boot</groupId>
3       <artifactId>spring-boot-configuration-processor</artifactId>
4       <optional>true</optional>
5     </dependency>
6
7
8   <build>
9     <plugins>
10       <plugin>
11         <groupId>org.springframework.boot</groupId>
12         <artifactId>spring-boot-maven-plugin</artifactId>
```

同时，该配置处理器依赖并不是业务上的需求，只是我们在开发过程中的需要，因此在打包的jar包中并不需要该依赖，因此我们需要在打包插件中将该依赖排除。


```
13         <configuration>
14             <excludes>
15                 <exclude>
16                     <groupId>org.springframework.boot</groupId>
17                     <artifactId>spring-boot-configuration-processor</artifactId>
18                 </exclude>
19             </excludes>
20         </configuration>
21     </plugin>
22 </plugins>
23 </build>
```

关注作者和知识库后续更新



尚硅谷
程序员标配，人手一套尚硅谷教程，自学一样...

已关注



SpringBoot2核心技术与响应式编程
基于SpringBoot2.3与2.4版本

关注

推荐阅读

05、Web开发
1、SpringMVC自动配置概览Spring Boot provides auto-configur...

03、了解自动配置原理
1、SpringBoot特点1.1、依赖管理父项目做依赖管理依赖管理 <...

01、Spring与SpringBoot
1、Spring能做什么1.1、Spring的能力1.2、Spring的生态https://s...