YAML快速入门

叩丁狼教育 (/u/231b43e2c05f) + 关注
2018.02.18 19:19* 字数 1776 阅读 24345 评论 3 喜欢 27
(/u/231b43e2c05f)

【原创文章，转载请注明原文章地址，谢谢！】

我们学习Java，都是先介绍properties文件，使用properties文件配合Properties对象能够很方便的适用于应用配置上。然后在引入XML的时候，我们介绍properties格式在表现层级关系和结构关系的时候，十分欠缺，而XML在数据格式描述和较复杂数据内容展示方面，更加优秀。到后面介绍JSON格式的时候，我们发现JSON格式比较XML格式，更加方便（除去数据格式限制之外），所以现在很多配置文件（比如Nginx和大部分脚本语言的配置文件）都习惯使用JSON的方式来完成，包括Springboot的出现目的也是在一定程度上去掉XML的繁琐配置。

在Springboot中，推荐使用properties或者YAML文件来完成配置，但是对于较复杂的数据结构来说，YAML又远远优于properties。本文就快速介绍YAML的常见语法格式。

先来看一个Springboot中的properties文件和对应YAML文件的对比：

```
#properties(示例来源于Springboot User guide):

environments.dev.url=http://dev.bar.com
environments.dev.name=Developer Setup
environments.prod.url=http://foo.bar.com
environments.prod.name=My Cool App
my.servers[0]=dev.bar.com
my.servers[1]=foo.bar.com
```

可以明显的看到，在处理层级关系的时候，properties需要使用大量的路径来描述层级（或者属性），比如environments.dev.url和environments.dev.name。其次，对于较为复杂的结构，比如数组（my.servers），写起来更为复杂。而对应的YAML格式文件就简单很多：

```
#YAML格式
environments:
    dev:
        url: http://dev.bar.com
        name: Developer Setup
    prod:
        url: http://foo.bar.com
        name: My Cool App
my:
    servers:
        - dev.bar.com
        - foo.bar.com
```

可以直观的看到，YAML使用冒号加缩进的方式代表层级（属性）关系，使用短横杠(-)代表数组元素。
经过这个示例的演示，可以很明显的看到YAML针对properties文件的优异之处。

快速入门

下面立刻展示YAML最基本，最常用的一些使用格式：
首先YAML中允许表示三种格式，分别是常量值，对象和数组
例如：

```
#即表示url属性值；
url: http://www.wolfcode.cn
#即表示server.host属性的值；
server:
    host: http://www.wolfcode.cn
#数组，即表示server为[a,b,c]
server:
    - 120.168.117.21
    - 120.168.117.22
    - 120.168.117.23
#常量
pi: 3.14   #定义一个数值3.14
hasChild: true  #定义一个boolean值
name: '你好YAML'   #定义一个字符串
```

注释

和properties相同，使用#作为注释，YAML中只有行注释。

基本格式要求

1，YAML大小写敏感；
2，使用缩进代表层级关系；
3，缩进只能使用空格，不能使用TAB，不要求空格个数，只需要相同层级左对齐（一般2个或4个空格）

对象

使用冒号代表，格式为key: value。冒号后面要加一个空格：

```
key: value
```

可以使用缩进表示层级关系；

```
key:
    child-key: value
    child-key2: value2
```

YAML中还支持流式(flow)语法表示对象，比如上面例子可以写为：

```
key: {child-key: value, child-key2: value2}
```

较为复杂的对象格式，可以使用问号加一个空格代表一个复杂的key，配合一个冒号加一个空格代表一个value：

```
?
    - complexkey1
    - complexkey2
:
    - complexvalue1
    - complexvalue2
```

意思即对象的属性是一个数组[complexkey1,complexkey2]，对应的值也是一个数组[complexvalue1,complexvalue2]

数组

使用一个短横线加一个空格代表一个数组项：

```
hobby:
    - Java
    - LOL
```

当然也可以有这样的写法：

```
-
    - Java
    - LOL
```

可以简单理解为：[[Java,LOL]]
一个相对复杂的例子：

```
companies:
    -
        id: 1
        name: company1
        price: 200W
    -
        id: 2
        name: company2
        price: 500W
```

意思是companies属性是一个数组，每一个数组元素又是由id,name,price三个属性构成；
数组也可以使用流式(flow)的方式表示：

```
companies: [{id: 1,name: company1,price: 200W},{id: 2,name: company2,price: 500W}]
```

常量

YAML中提供了多种常量结构，包括：整数，浮点数，字符串，NULL，日期，布尔，时间。下面使用一个例子来快速了解常量的基本使用：

```
boolean:
    - TRUE  #true,True都可以
    - FALSE  #false，False都可以
float:
    - 3.14
    - 6.8523015e+5  #可以使用科学计数法
int:
    - 123
    - 0b1010_0111_0100_1010_1110    #二进制表示
null:
    nodeName: 'node'
    parent: ~  #使用~表示null
string:
    - 哈哈
    - 'Hello world'  #可以使用双引号或者单引号包裹特殊字符
    - newline
      newline2    #字符串可以拆成多行，每一行会被转化成一个空格
date:
    - 2018-02-17    #日期必须使用ISO 8601格式，即yyyy-MM-dd
datetime:
    -  2018-02-17T15:02:31+08:00    #时间使用ISO 8601格式，时间和日期之间使用T连接，最后
```

一些特殊符号

YAML中提供了很多特殊符号，在这里简单介绍常用的一些：
1，--- YAML可以在同一个文件中，使用---表示一个文档的开始；比如Springboot中profile的定义：

```
server:
    address: 192.168.1.100
---
spring:
    profiles: development
    server:
        address: 127.0.0.1
---
spring:
    profiles: production
    server:
        address: 192.168.1.120
```

代表定义了两个profile，一个是development，一个production；也常常使用---来分割不同的内容，比如记录日志：

```
---
Time: 2018-02-17T15:02:31+08:00
User: ed
Warning:
     This is an error message for the log file
---
Time: 2018-02-17T15:05:21+08:00
User: ed
Warning:
    A slightly different error message.
```


2，... 和---配合使用，在一个配置文件中代表一个文件的结束：

```
---
time: 20:03:20
player: Sammy Sosa
action: strike (miss)
...
---
time: 20:03:47
player: Sammy Sosa
action: grand slam
...
```

相当于在一个yaml文件中连续写了两个yaml配置项。

3，!! YAML中使用!!做类型强行转换：

```
string:
    - !!str 54321
    - !!str true
```

相当于把数字和布尔类型强转为字符串。当然允许转型的类型很多，比如：

```
--- !!set
- Mark McGwire: 65
- Sammy Sosa: 63
- Sammy Sosa: 63
- Ken Griffy: 58
```

将数组解析为set，简单理解，转化的内容就是：[{Ken Griffy=58}, {Mark McGwire=65}, {Sammy Sosa=63}]，重复的Sammy Sosa去掉；

4，>在字符串中折叠换行，| 保留换行符，这两个符号是YAML中字符串经常使用的符号，比如：

```
accomplishment: >
 Mark set a major league
 home run record in 1998.
stats: |
 65 Home Runs
 0.278 Batting Average
```

那么结果是：

```
stats=65 Home Runs
 0.278 Batting Average,
```

即| 符号保留了换行符，而accomplishment的结果为：

```
accomplishment=Mark set a major league home run record in 1998.
```

即将换行符转化成了空格；要注意一点的是，每行的文本前一定要有一个空格。
|符号常见用于在YAML中配置HTML片段：

```
phraseTemplate: |
  <p style="color: red">
    some template ${msg}
  </p>
```

5，引用。重复的内容在YAML中可以使用&来完成锚点定义，使用*来完成锚点引用，例如：

```
hr:
- Mark McGwire
- &SS Sammy Sosa
rbi:
- *SS
- Ken Griffey
```

可以看到，在hr中，使用&SS为Sammy Sosa设置了一个锚点（引用），名称为SS，在rbi中，使用*SS完成了锚点使用，那么结果为：

```
{rbi=[Sammy Sosa, Ken Griffey], hr=[Mark McGwire, Sammy Sosa]}
```

我们也可以这样定义：

```
SS: &SS Sammy Sosa
hr:
 - Mark McGwire
 - *SS
rbi:
 - *SS
 - Ken Griffey
```

注意，不能独立的定义锚点，比如不能直接这样写： &SS Sammy Sosa；另外，锚点能够定义更复杂的内容，比如：

```
default: &default
    - Mark McGwire
    - Sammy Sosa
hr: *default
```

那么hr相当于引用了default的数组，注意，hr: *default要写在同一行。

6，合并内容。主要和锚点配合使用，可以将一个锚点内容直接合并到一个对象中。来看一个示例：

```
merge:
  - &CENTER { x: 1, y: 2 }
  - &LEFT { x: 0, y: 2 }
  - &BIG { r: 10 }
  - &SMALL { r: 1 }

sample1:
    <<: *CENTER
    r: 10

sample2:
    << : [ *CENTER, *BIG ]
    other: haha

sample3:
    << : [ *CENTER, *BIG ]
    r: 100
```

在merge中，定义了四个锚点，分别在sample中使用。
sample1中，<<: *CENTER意思是引用{x: 1,y: 2}，并且合并到sample1中，那么合并的结果为：sample1={r=10, y=2, x=1}

sample2中，<<: [*CENTER, *BIG] 意思是联合引用{x: 1,y: 2}和{r: 10}，并且合并到sample2中，那么合并的结果为：sample2={other=haha, x=1, y=2, r=10}

sample3中，引入了*CENTER, *BIG，还使用了r: 100覆盖了引入的r: 10，所以sample3值为：sample3={r=100, y=2, x=1}

有了合并，我们就可以在配置中，把相同的基础配置抽取出来，在不同的子配置中合并引用即可。

以上示例均通过Snakeyaml测试通过

了解更多

如果要了解更多关于YAML的详情，可以了解（本文参考）：
1，YAML1.2规范：http://yaml.org/spec/1.2/spec.pdf (https://link.jianshu.com?t=http%3A%2F%2Fyaml.org%2Fspec%2F1.2%2Fspec.pdf)
2，JS-YAML在线示例：http://nodeca.github.io/js-yaml/ (https://link.jianshu.com?t=http%3A%2F%2Fnodeca.github.io%2Fjs-yaml%2F)