

YAML语法规则

参考了[YAML官网](#)，YAML 是专注于写配置文件的语言，这个名字的含义是 `YAML Ain't Markup Language` (YAML不是一种标记语言)，但是实际上 `YAML` 还是一种标记语言，只不过是更加聚焦于数据的标记语言。

- YAML的基本语法规则

1. 大小写敏感
2. 使用缩进表示层级关系（这个和Python很像了，估计是从Python借鉴的）缩进时绝对不允许使用Tab键，只允许使用空格键 缩进的空格数不重要，只要元素左侧能对齐就可以。字符串可以不用引号标注

YAML完全兼容Json的语法，可以把Json看做是YAML的一个子集

我们来看一下YAML支持的数据类型

- 对象：就是键值对的集合，和Json中对象的概念一样
- 列表：又称为序列，是一组有序排列的值，和Json中列表的概念一样
- scalars(纯量)：不可拆分的数据类型，和Json中的基本数据类型一样

来看一下YAML 具体的语法。

- 在一个 `yaml` 文件中，使用 `---` 来表示一段文档（或者一组配置）的开始，使用 `...` 来表示一段文档的结束。如果 `yaml` 中只有一组配置，则可以省略掉 `---`。
- 使用 `#` 来表示注解
- 使用 `-` 来表示单个的列表项，比如

```
- A
- B
- C
```

对应Json中的 `['A', 'B', 'C']`

```
-
- A
- B
- C
-
- D
- E
- F
```

对应于Json的 `[['A', 'B', 'C'], ['D', 'E', 'F']]`

尤其是要注意，由于YAML兼容Json的语法，所以我们直接在yaml文件中写 `[[A, B, C], [D, E, F]]` 也是可以的。

- 使用 `:` 来表示键值对

```
name: chico
age: 18
```

对应Json的 { name: 'chico', age: 18 }

```
-
  name: chico
  age: 18
-
  name: dong
  age: 19
```

对应Json中的 [{ name: 'chico', age: 18 }, { name: 'dong', age: 19 }]

看一个将列表和对象结合起来的

```
american:
- Boston Red Sox
- Detroit Tigers
- New York Yankees
national:
- New York Mets
- Chicago Cubs
- Atlanta Braves
```

对应Json { american: ['Boston Red Sox', 'Detroit Tigers', 'New York Yankees'], national: ['New York Mets', 'Chicago Cubs', 'Atlanta Braves'] }

与Json不同的是，YAML中键值对的键不要求一定是字符串，可以是一些复杂的类型，但是Json中要求键值对的键必须是字符串。

当YAML中键值对的键是复杂类型的时候，必须要用 ? 说明，比如

```
?[a,b,c]:color
#或者
?-a
-b
-c
:color
```

- YAML中 null 用 ~ 表示 比如 money:~
- YAML中段落用 | 和缩进表示，YAML会保留该段落中的回车换行符 比如

```
description: |
```

#注意: 和 | 之间的空格

这是一篇非常非常长的介绍YAML的文章
文章是非常有内容的

- YAML中用 > 和缩进表示把段落连城一句话, YAML会把该段落中的回车换行替换成空格, 最终连成一句话

```
description: >
```

这是一篇介绍YAML的文章, 非常长
但是把他们连成一句话了

- YAML中+ 表示保留段落末尾的换行, - 表示删除文本末尾的换行

```
a: |+
```

保留了换行符

```
b: |-
```

删除了换行符

- YAML中也会把 " 双引号和 ' 单引号中的字符串连成一行, 引号中可以包含转义字符

```
description:
```

"虽然表面上看这是两行,
但其实是一行"

#和下面这个是一样的

```
description:
```

虽然表面上看这是两行,
但其实是一行

- YAML中使用 ! 和 !! 来做强制类型转换 比如

```
#强行把123转换成str
```

```
e:!!str 123
```

```
#强行把boolean型转换为str
```

```
b:!!str true
```

- YAML中可以通过 & 来锚点, 通过 * 来引用

```
s1:
```

```
  name: chico
```

```
  favourite: &SS reading
```

```
s2:
```

```
  name: dong
```

```
  favourite: *SS
```

- << 表示追加 比如

```
default: &&default
  host: http
  path: index
server1:
  name: server1
  <<: *default
server2:
  name: server2
  <<: *default
```

等价于

```
default: &&default
  host: http
  path: index
server1:
  name: server1
  host: http
  path: index
server2:
  name: server2
  host: http
  path: index
```

- YAML还支持set和ordered map 在YAML中set就是一组值全部为null的键值对，比如

```
--- !!set #强行转换为set
? haha ha
? chico dong
? haha lala
```

对应的json是 { haha ha: null, 'chico dong': null, 'haha lala': null }

在YAML中ordered map就是一组有序的键值对（可以认为是放在list中的键值对），比如

```
--- !!omap #强行转换为有序键值对
- haha ha: haha
- chico dong: dong
- lala: "haha lala"
```

对应的json为 [{ haha ha: 'haha' }, { 'chico dong': 'dong' }, { lala: 'haha lala' }] 感觉这个和正常的yaml嵌套写法没什么区别