## 思考问题

前面我们配置Capability时，各个参数都是在代码里面写死的，比如：desired\_caps['platformVersion']='5.1.1' 一旦设备和测试的app发生改变则需要去代码里面一个个修改，要么同时根据不同设备不同App来维护多套代码，这样显示是不符合规范而且是低效的！违背了自动化的初衷，那么如何改进这样的现状呢？

## 解决思路

针对这种可能频繁变动的部分，可以将数据和代码分离。将数据单独抽离出来放在配置文件里面， 代码直接从配置文件去读取数据，这样能够减少代码冗余，提高效率。PS：类似的Web前端的html标签和css分离。

配置数据该如何管理？这里我们推荐使用**yaml**来管理配置数据。

## yaml概述

#### yaml简介

正如YAML所表示的YAML Ain’t Markup Language，YAML 是一种简洁的非标记语言。YAML以数据为中心，使用空白，缩进，分行组织数据，从而使得表示更加简洁易读。

由于实现简单，解析成本很低，YAML特别适合在脚本语言中使用。列一下现有的语言实现：Ruby，Java，Perl，Python，PHP，JavaScript等。

YAML 是专门用来写配置文件的语言，非常简洁和强大，远比 JSON 格式方便。比如同一段数据Josn和Yaml的表示分别如下：

##### Json

{ name: 'Tom Smith',age: 37,spouse: { name: 'Jane Smith', age: 25 },children: [ { name: 'Jimmy Smith', age: 15 },{ name: 'Jenny Smith', age: 12 } ] }

##### yaml

name: Tom Smith

age: 37

spouse:

name: Jane Smith

age: 25

children:

- name: Jimmy Smith

age: 15

- name: Jenny Smith

age: 12

#### 语法特点

* 大小写敏感
* 使用缩进表示层级关系
* 缩进时不允许使用Tab键，只允许使用空格。
* 缩进的空格数目不重要，只要相同层级的元素左侧对齐即可

#### yaml下载安装

* 下载地址：<http://pyyaml.org/wiki/PyYAML>
* 安装：下载完成后点击运行安装包即可。
* 安装完成后在python引入yaml检测是否安装成功。
* [PyYAML官方文档](http://pyyaml.org/wiki/PyYAMLDocumentation)

## yaml数据类型详解

### 支持数据类型

1. **纯量**（scalars）：单个的、不可再分的值
2. **数组**：一组按次序排列的值，又称为序列（sequence） / 列表（list）
3. **对象**：键值对的集合，又称为映射（mapping）/ 哈希（hashes） / 字典（dictionary）

### 纯量

数据最小的单位，不可以再分割。类似于Python中单个变量

flag

### list数组

与Python的list数组结构类似，数组元素使用“-”开头，也可以根据缩进进行数组嵌套。

- Jack

- Harry

- Sunny

# 也可以写成一行

[Jack,Harry,Sunny]

对应到python的list写法如下：

['Jack','Harry','Sunny']

扩展学习：[Python数据类型视频教程](http://www.51zxw.net/list.aspx?cid=615)

### 对象

对象的一组键值对，使用冒号结构表示。类似Python中的字典数据结构。

platformName: Android

platformVersion: 6.0.1

# **Yaml** 也允许另一种写法，将所有键值对写成一个行内对象。

{platformName: Android,platformVersion: 6.0.1}

注意：冒号后面一定要有空格！对应到python字典的写法如下：

{'platformName': 'Android', 'platformVersion': '6.0.1'}

### 数据嵌套

yaml数据嵌套表示可以将上面的各类数据根据实际场景进行组合嵌套。

###### 数据场景

Tom Smith 37岁，他有一个妻子叫 Jane Smith，25岁。 另外他有2个孩子，一个叫Jimmy Smith，15岁；另外一个叫Jenny Smith ，12岁。

yaml语法表示如下：

**familyInfo.yaml**

name: Tom Smith

age: 37

spouse:

name: Jane Smith

age: 25

children:

- name: Jimmy Smith

age: 15

- name: Jenny Smith

age: 12

转化为Python的写法为：

{'name':'Tom Smith','age':37,'spouse':{'name':'Jane Smith','age':25},'childern':[{'name':'Jimmy Smith','age':15},{'name':'Jenny Smith','age':12}]}

## yaml数据操作

### 数据读取

##### 测试场景

* 读取配置中的所有信息
* 读取yaml数据表中Tom Smith的姓名、年龄、信息
* 单独读取配偶的姓名和年龄信息
* 分别读取两个孩子的姓名、年龄信息

##### load方法

load(stream, Loader=Loader) 解析文件流中的第一个YAML文档并生成相应的Python对象。

##### 代码实现

yaml\_load.py

import yaml

file=open('familyInfo.yaml','r')

data=yaml.load(file)

print(data)

print(data['name'])

print(data['age'])

print(data['spouse'])

print(data['spouse']['name'])

print(data['spouse']['age'])

print(data['children'])

print(data['children'][0]['name'])

print(data['children'][0]['age'])

print(data['children'][1]['name'])

print(data['children'][1]['age'])

### 数据修改

如果想改变某个数据，可以使用如下方法：

data['name']='51zxw'

**print**(data['name'])

注意：此处只是变量类型的数据变更，不会真正修改到yaml配置表中的数据。

### 数据转化

方法：dump()可以将Python对象序列化成YAML流。如果stream为None，则返回生成的字符串。

##### 测试场景

将下面python数据类型转化为yaml数据类型

slogan=['welcome','to','51zxw']

website={'url':'www.51zxw.net'}

##### 代码实践

import yaml

slogan=['welcome','to','51zxw']

website={'url':'www.51zxw.net'}

**#python** data

**print**(slogan)

**print**(website)

**#yaml** data

**print**(yaml.dump(slogan))

**print**(yaml.dump(website))

运行结果：

C:\Python35\python.exe E:/AppiumScript/advance/yaml/yaml\_down.py

['welcome', 'to', '51zxw']

{'url': 'www.51zxw.net'}

[welcome, to, 51zxw]

{url: www.51zxw.net}

Process finished with **exit** code 0

### Capability配置数据分离实践

#### 测试场景

将capability的各项参数值与代码进行分离。

#### 场景分析

结合前面所学习的知识，我们可以把之前capability中各项写死的配置信息来抽离出来，存放在一个yaml配置文件中，使用 对象数据类型来存储数据；然后调用load()方法读取数据，从而实现数据和代码的分离。

#### 代码实现

1.参数配置表：desired\_caps.yaml

**platformName**: **Android**

**platformVersion**: 5.1.1

**deviceName**: 127.0.0.1:62025

**app**: **C**:\**Users**\**Shuqing**\**Desktop**\**Appium** **software**\**chapter4**\**App**\**kaoyan3**.1.0.apk

**noReset**: **False**

**appPackage**: **com**.tal.kaoyan

**appActivity**: **com**.tal.kaoyan.ui.activity.SplashActivity

**ip**: 127.0.0.1

**port**: 4723

capability\_yaml.py

from appium import webdriver

import yaml

file=open('desired\_caps.yaml','r')

data=yaml.load(file)

desired\_caps={}

desired\_caps['platformName']=data['platformName']

desired\_caps['platformVersion']=data['platformVersion']

desired\_caps['deviceName']=data['deviceName']

desired\_caps['app']=data['app']

desired\_caps['noReset']=data['noReset']

desired\_caps['appPackage']=data['appPackage']

desired\_caps['appActivity']=data['appActivity']

driver = webdriver.Remote('http://'+str(data['ip'])+':'+str(data['port'])+'/wd/hub', desired\_caps)

##### 报错分析

yaml.scanner.ScannerError: mapping values are not allowed here

该报错说明map对象数据类型写法错误，一般为“：”后面没有留空格。如：platformName:Android

#### 参考资料

* <https://blog.csdn.net/Marksinoberg/article/details/52979419>
* <http://www.ruanyifeng.com/blog/2016/07/yaml.html>