# AutoTest Studio入门系列7：数据字典

AutoTest Studio提供一个性能非常卓越的数据字典模块，与其他测试框架使用的文本方式存储数据不同，AutoTest Stduio使用了Sqlite作为内部存储引擎，Sqlite的性能和稳定性是业界公认的。按照官方的介绍，AutoTest Stduio即便是存储上百万条记录，依然能提供良好的读取性能。

需要指出的是，AutoTest Studio的数据字典在任务运行期间，是只读的，即用户不能通过函数接口向数据字典中写入或者修改数据，只能通过Data Dictionary工具预先写入或者修改数据，这种设计保证任务运行期间不污染测试数据。

AutoTest Studio的数据字典以目录结构组织数据层次关系，这种设计使得数据字典组织关系具有良好的可读性，在数据字典中，有两个概念，数据集(Dataset)和数据项（Dataitem）。

**Dataset**，同一类数据的集合，如一个测试用户的基本信息，Dataset在全局是唯一的，即同一个数据字典中，数据集的名称不能重复，即便是不同的目录下也不行。

**Dataitem**，数据集下的一条数据记录，即一个key-value记录。

在项目创建的时候，AutoTest Studio默认为每个项目创建一个空的数据字典。

数据字典模块提供两个操作接口。

读取整个数据集，返回一个python字典结构。

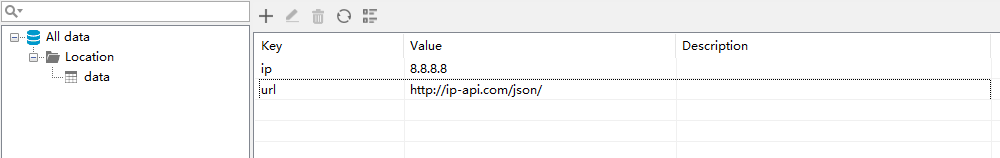
*GetDataset (dataset)*

读取数据集中key对应的value。

*GetDataValue(dataset, key)*

我们以前面章节的demo为例，将demo中直接定义ip和ipQueryUrl两个变量的值存放到数据字典中，然后通过数据字典接口来读取这两个变量的值。

首先，我们先建立一个目录（Location），再建立一个数据集(data),在“data”下新建两个数据项ip，ipQueryUrl，如下图所示。



修改原来的代码代码。

原来的代码：

import requests

from autotest import \*

#set test case information

SetCase("TEST-1","Get location by ip","1")

ip="8.8.8.8"

ipQueryUrl="http://ip-api.com/json/{0}".format(ip)

LogInfo("ipQueryUrl:{0}".format(ipQueryUrl))

response=requests.get(ipQueryUrl)

AssertEqual(response.status\_code,200,"Query ip is successful.")

修改后的代码：

import requests

from autotest import \*

#set test case information

SetCase("TEST-1","Get location by ip","1")

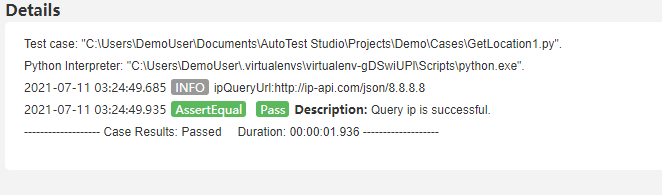
ipQueryUrl="{0}{1}".format(GetDataValue("data","url"),GetDataValue("data","ip"))

LogInfo("ipQueryUrl:{0}".format(ipQueryUrl))

response=requests.get(ipQueryUrl)

AssertEqual(response.status\_code,200,"Query ip is successful.")

运行结果如下，从日志看出，与原来的代码运行结果一致。



再次修改代码，这次，我们将整个数据集读取出来。

import requests

from autotest import \*

#set test case information

SetCase("TEST-1","Get location by ip","1")

data=GetDataset("data")

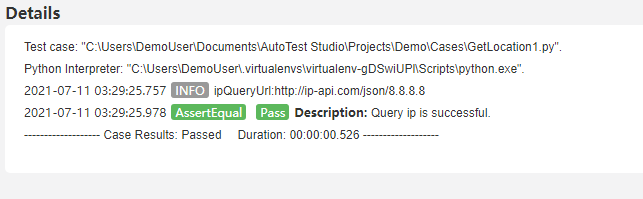
ipQueryUrl="{0}{1}".format(data["url"],data["ip"])

LogInfo("ipQueryUrl:{0}".format(ipQueryUrl))

response=requests.get(ipQueryUrl)

AssertEqual(response.status\_code,200,"Query ip is successful.")

运行结果：



从上面的demo可以看出，数据字典的使用非常简单。