# Python处理excel技巧

原创：Snake 安蜀黍 python爱好部落 2017-12-23

> 在测试中，我们需要对一些数据存取。存取的方式有很多，用txt就可以。但是用csv,excel等表单格式来存取，更方便直观。连大名鼎鼎的微软也考虑将 Python 作为 Excel 的一种官方脚本语言。

Python结合Excel进行数据处理，是一种比较流行的数据处理方式。

微软，最近也宣布：微软考虑将 Python 作为 Excel 的一种官方脚本语言

Python处理Excel,并不方便像普通文本文件一样直接进行读写，需要借助第三方库来实现。

常用的python-excel 系列有：

```
xlrd、xlwt、xlutils
xlrd — 读取 Excel 文件
xlwt — 写入 Excel 文件
xlutils — 操作 Excel 文件的实用工具，如复制、分割、筛选等
```

这些package可以直接用pip来安装，非常方便。

**excel读取**

结合一段简单的代码来看：

```
import xlrd
# 打开 xls 文件
book = xlrd.open_workbook("test.xls")
print "表单数量:", book.nsheets
print "表单名称:", book.sheet_names()
# 获取第1个表单
sh = book.sheet_by_index(0)
print u"表单 %s 共 %d 行 %d 列" % (sh.name, sh.nrows, sh.ncols)
print "第二行第三列:", sh.cell_value(1, 2)
# 遍历所有表单
for s in book.sheets():
    for r in range(s.nrows):
        # 输出指定行
        print s.row(r)
```

读取excel常用的方法如下：

```
open_workbook 打开文件
sheet_by_index 获取某一个表单
sheets 获取所有表单
cell_value 获取指定单元格的数据
```

当然，我们获取表单有几种方式：

```
data = xlrd.open_workbook('excelname')
table = data.sheets()[0] #通过索引顺序获取
table = data.sheet_by_index(0) #通过索引顺序获取
table = data.sheet_by_name('sheetname') #通过名称获取
```

## excel写入

结合一段简单的代码来看：

```
import xlwt
# 创建 xls 文件对象
wb = xlwt.Workbook()
# 新增一个表单
sh = wb.add_sheet('A Test Sheet')
# 按位置添加数据
sh.write(0, 0, 1234.56)
sh.write(1, 0, 8888)
sh.write(2, 0, 'hello')
sh.write(2, 1, 'world')
# 保存文件
wb.save('example.xls')
```

写入excel常用的方法：

```
Workbook 创建文件对象
add_sheet 新增一个表单
write 在指定单元格写入数据
```

我们可以将这些方法，写成一个类来调用。做成自己的包，做其他自动化的时候，直接调用即可。

```
#! /usr/bin/env python
# coding=utf-8

'''
FuncName: Excelhelper.py
Desc: operate excel
Author: Anderson
微信公众号: python爱好部落
'''
import xlwt
import xlrd
import os
from xlutils.copy import copy

class WriteExcel(object):
    """
    操作excel表格
    写入excel：1.无文件则自动创建文件，创建新文件和新工作表
              2.存在文件且存在指定工作表，直接操作该工作表
              3.存在文件且不存在指定工作表，新建指定工作表并写入该表，不影响已存在工作表数据
    """

    def __init__(self, fileName, sheetName):
        self.fileName = fileName
        self.sheetName = sheetName
        self.style = self.sheetStyle()

    def sheetStyle(self):
        """
        添加样式:
        http://nullege.com/codes/search/xlwt.Style
        https://xlwt.readthedocs.io/en/latest/
        :return: 各种配置参数
        """
        # style = xlwt.easyxf('font: color-index red, bold on')
        font = xlwt.Font()  # Create the Font
        # 字体
        font.name = 'Times New Roman'
```

```
        # 粗体
        # font.bold = True
        # 下划线
        # font.underline = True
        # 斜体
        # font.italic = True

        pattern = xlwt.Pattern()  # Create the Pattern
        pattern.pattern = xlwt.Pattern.SOLID_PATTERN  # May be: NO_PATTERN 关模式, SOLID_PATTERN 开模式, o
r 0x00 through 0x12
        pattern.pattern_fore_colour = 2  # May be: 8 through 63. 0 = Black, 1 = White, 2 = Red, 3 = Gree
n, 4 = Blue, 5 = Yellow, 6 = Magenta, 7 = Cyan, 16 = Maroon, 17 = Dark Green, 18 = Dark Blue, 19 = Dark
Yellow , almost brown), 20 = Dark Magenta, 21 = Teal, 22 = Light Gray, 23 = Dark Gray, the list goes on.
..

        style = xlwt.XFStyle()  # Create the Style 初始化样式
        style.font = font  # Apply the Font to the Style 为样式设置字体
        style.pattern = pattern  # Add Pattern to Style

        return style

    def sheetValue(self, row, col, value, style=False):
        """
        :param sheetName: 表名, fileName: 文件名, row:行, col：列, value:值, style:格式
        :return: 对文件的指定表进行写数据操作，支持重复写
        """
        # xls文件不存在则新创建文件和新创建工作表
        if os.path.exists(self.fileName) == False:
            writeExcel = xlwt.Workbook()
            # 注意：如果对同一个单元格重复操作，会引发overwrite Exception，想要取消该功能，需要在添加工作表时指定为
可覆盖，所以在打开时加cell_overwrite_ok=True解决
            writeExcel.add_sheet(self.sheetName, cell_overwrite_ok=True)
            # 这里只能保存扩展名为xls的，xlsx的格式不支持
            writeExcel.save(self.fileName)
        else:
            # open existed xls file 需要 newWb = copy('fileName')
            # 注意添加参数formatting_info=True，得以保存之前数据的格式
            readExcel_old = xlrd.open_workbook(self.fileName, formatting_info=True)
            # 判断sheetName的索引，将数据写入该表
            sheetNames = readExcel_old.sheet_names()
            readExcel_new = copy(readExcel_old)
            # 判断是否存在sheetName，不存在则新建该工作表，不影响其他表数据
            if self.sheetName not in sheetNames:
                readExcel_new.add_sheet(self.sheetName)
                readExcel_new.save(self.fileName)
                readExcel_new = xlrd.open_workbook(self.fileName, formatting_info=True)
                sheetNames = readExcel_new.sheet_names()
                readExcel_new = copy(readExcel_new)
            else:
                pass
            sheetNameIndex = sheetNames.index(self.sheetName)
            writeExcel = readExcel_new.get_sheet(sheetNameIndex)

            if style:
                writeExcel.write(row, col, value, self.style)
            else:
                writeExcel.write(row, col, value)
                # 这里只能保存扩展名为xls的，xlsx的格式不支持
            readExcel_new.save(self.fileName)

class ReadExcel(object):
    def __init__(self, fileName, sheetName):
        self.fileName = fileName
        self.sheetName = sheetName
        self.sheets = xlrd.open_workbook(fileName, formatting_info=True)
        self.tables = self.sheets.sheet_by_name(self.sheetName)
```

```
        self.tables = self.sheets.sheet_by_name(self.sheetName)
        self.handele = dict([(sheetName, self.sheets.sheet_by_name(sheetName))
                             for sheetName in self.sheets.sheet_names()])

        # 获取整行的值
    def sheetRowValue(self, row=0):
        if self.sheetName not in self.handele: return None
        return self.tables.row_values(row)

        # 获取整列的值
    def sheetColValue(self, col=0):
        if self.sheetName not in self.handele: return None
        return self.tables.row_values(col)

        # 获取所有表的名称
    def sheetNames(self):
        return self.sheets.sheet_names()

        # 获取单元格的值
    def sheetCellValue(self, row, col):
        if self.sheetName not in self.handele: return None
        return self.tables.cell(row, col).value

        # 获取工作表的行数
    def sheetRowNum(self):
        if self.sheetName not in self.handele: return None
        return self.tables.nrows

        # 获取工作表的列数
    def sheetColNum(self):
        if self.sheetName not in self.handele: return None
        return self.tables.ncols

        # 获取所有行的值
    def get_all_Row(self):
        row_value=[]
        for i in range(self.sheetRowNum()):
            #print(self.sheetRowValue(i))
            row_value.append(self.sheetRowValue(i))
        return(row_value)

        # 获取所有列的值
    def get_all_Col(self):
        col_value = []
        for i in range(self.sheetColNum()):
            #print(self.sheetColValue(i))
            col_value.append(self.sheetColValue(i))
        return col_value

#测试一下
if __name__ == "__main__":
    fileName = "helper.xlsx"
    sheetName = "Sheet1"

    myWrite = WriteExcel(fileName, sheetName)
    style = True

    for row in range(10):
        col = row
        value = "anderson"
        myWrite.sheetValue(row, col, value, style)

    myRead = ReadExcel(fileName, sheetName)
    print(myRead.get_all_Row())
```

封装好自己的类，在其它地方，直接调用。
妈妈在也不用担心我不会用python处理excel了。

作者简介：
Snake, 人称安蜀黍，专职软件测试10几年，测试界的老司机。

更多精彩，请关注微信公众号：**python爱好部落**