机器人流程自动化RPA

最佳实践

最佳实践

收发邮件

应用场景

在编写RPA流程时，我们经常需要在流程中收发邮件，而最常用的收发邮件方式是使用SMTP和POP3，下面我们来说明一下如何在RPA中收发邮件。

发送邮件

SMTP（Simple Mail Transfer Protocol）即简单邮件传输协议,它是一组用于由源地址到目的地址传送邮件的规则，由它来控制信件的中转方式。
python的smtplib提供了一种很方便的途径发送电子邮件。它对smtp协议进行了简单的封装。
Python创建 SMTP 对象语法如下：

```
import smtplib
smtpObj = smtplib.SMTP( [host [, port [, local_hostname]]] )
```

参数说明：

- host: SMTP 服务器主机。 你可以指定主机的ip地址或者域名如: runoob.com，这个是可选参数。
- port: 如果你提供了 host 参数, 你需要指定 SMTP 服务使用的端口号，一般情况下 SMTP 端口号为 25。
- local_hostname: 如果 SMTP 在你的本机上，你只需要指定服务器地址为 localhost 即可。

Python SMTP 对象使用 sendmail 方法发送邮件，语法如下：

```
SMTP.sendmail(from_addr, to_addrs, msg[, mail_options, rcpt_options])
```

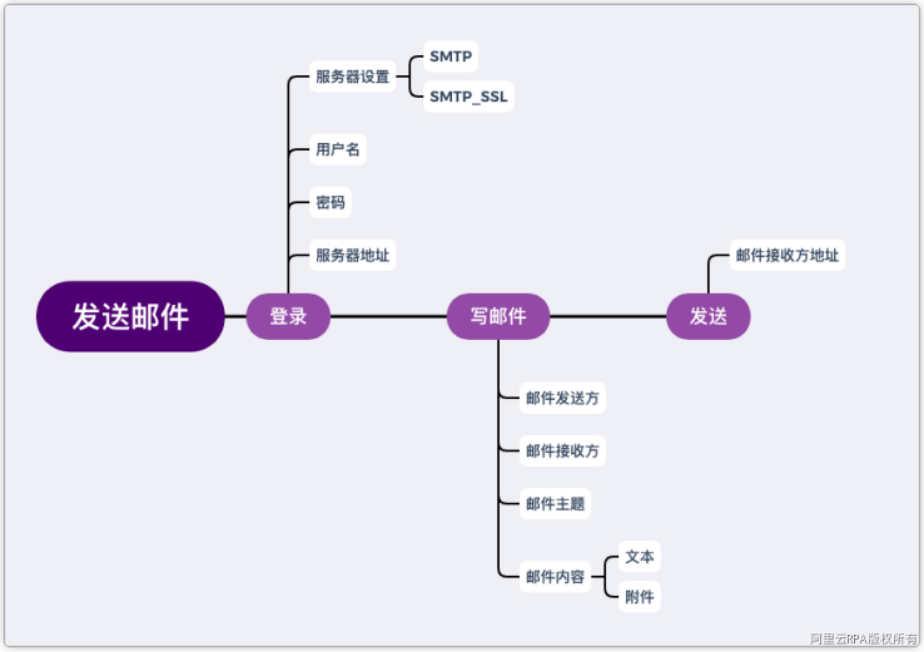
参数说明：

- from_addr: 邮件发送者地址。
- to_addrs: 字符串列表，邮件发送地址。
- msg: 发送消息

这里要注意一下第三个参数，msg 是字符串，表示邮件。我们知道邮件一般由标题，发信人，收件人，邮件内容，附件等构成，发送邮件的时候，要注意 msg 的格式。这个格式就是 smtp 协议中定义的格式。

示例

使用脚本发送邮件的思路其实和客户端发送邮件一样，过程都是：**登录 —> 写邮件 —> 发送**
只不过通过脚本发送时我们需要考虑到整个过程的方方面面。以下为思路导图：



发送简单邮件

基本思路是：

1. 设置好服务器端信息
2. 邮件主体信息
3. 登录发送

另外在发送简单文本邮件时，我们需要使用 MIMEText 类作为邮件主体。

```
from rpa.core import *
from rpa.utils import *
import rpa3 as rpa # 使用V3引擎
import smtplib
from email.mime.text import MIMEText
```

```
def start():
# 设置登录及服务器信息
# 邮箱服务器地址
mail_host = 'smtp.xxxx.com'
# 邮件用户名，这里我们推荐使用RPA的云变量(资产变量)存储敏感信息
mail_user = rpa.asset.value("mail_user")
# 密码(部分邮箱为授权码) ，这里我们推荐使用RPA的云变量(资产变量)存储敏感信息
mail_password = rpa.asset.value("mail_password")
# 邮件发送方邮箱地址
sender = 'xxxx@xxxx.com'
# 邮件接受方邮箱地址，注意这里是一个列表<list>，这意味着你可以写多个邮件地址群发
receivers = ['xxxx@xxxx.com']

# 设置email信息
# 邮件内容设置
message = MIMEText('使用RPA测试邮件发送','plain','utf-8')
# 邮件主题
message['Subject'] = '邮件测试'
# 发送方信息
message['From'] = sender
# 接受方信息
message['To'] = receivers[0]

# 登录并发送邮件
try:
smtpObj = smtplib.SMTP_SSL()
# 连接到服务器
smtpObj.connect(mail_host, 465)
# 登录到服务器
smtpObj.login(mail_user, mail_password)
# 发送
smtpObj.sendmail(
sender, receivers, message.as_string())
# 退出
smtpObj.quit()
print('success')
except smtplib.SMTPException as e:
print('error',e) # 打印错误
```

一些邮箱登录比如阿里企业邮箱、QQ 邮箱需要 SSL 认证，所以 SMTP 已经不能满足要求，而需要 SMTP_SSL，解决办法为：

```
smtpObj = smtplib.SMTP()
smtpObj.connect(mail_host, 25)
#######替换为########
smtpObj = smtplib.SMTP_SSL()
smtpObj.connect(mail_host, 465) # 注意要修改端口号
```

常用邮箱端口说明：

邮箱	SMTP服务器	SSL协议端口	非SSL协议端口

163	smtp.163.com	465或者994	25
qq	smtp.qq.com	465或587	25

发送带有附件的邮件

基本思路就是，使用MIMEMultipart来标示这个邮件是多个部分组成的，然后attach各个部分。如果是附件，则add_header加入附件的声明。
在python中，MIME的这些对象的继承关系如下。
MIMEBase
|— MIMENonMultipart
|— MIMEApplication
|— MIMEAudio
|— MIMEImage
|— MIMEMessage
|— MIMEText
|— MIMEMultipart
一般来说，不会用到MIMEBase，而是直接使用它的继承类。MIMEMultipart有attach方法，而MIMENonMultipart没有，只能被attach。MIME有很多种类型，这个略麻烦，如果附件是图片格式，我要用MIMEImage，如果是音频，要用MIMEAudio，如果遇到word、excel等我们不确定该用哪种MIME类型时，可以不管什么类型的附件，都用MIMEApplication，MIMEApplication默认子类型是application/octet-stream。
application/octet-stream表明“这是个二进制的文件，希望对端那边知道怎么处理”，然后客户端，比如阿里企业邮箱，收到这个声明后，会根据文件扩展名来猜测。

```
from rpa.core import *
from rpa.utils import *
import rpa3 as rpa # 使用V3引擎
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from email.mime.image import MIMEImage
from email.mime.application import MIMEApplication


def start():

# 设置登录及服务器信息
# 邮箱服务器地址
mail_host = 'smtp.xxxx.com'
# 邮件用户名，这里我们推荐使用RPA的云变量(资产变量)存储敏感信息
mail_user = rpa.asset.value("mail_user")
# 密码(部分邮箱为授权码)，这里我们推荐使用RPA的云变量(资产变量)存储敏感信息
mail_password = rpa.asset.value("mail_password")
# 邮件发送方邮箱地址
sender = 'xxxx@xxxx.com'
# 邮件接受方邮箱地址，注意这里是一个列表<list>，这意味着你可以写多个邮件地址群发
receivers = ['xxxx@xxxx.com']
```

```
# 设置eamil信息
# 添加一个MIMEmultipart类，处理正文及附件
message = MIMEMultipart()
message['From'] = sender
message['To'] = receivers[0]
message['Subject'] = '邮件测试'

# ---这是文字部分---
# 推荐使用html格式的正文内容，这样比较灵活，可以附加图片地址，调整格式等
with open(r'C:\temp\测试HTML.html','rb') as hf:
html_content = hf.read()
# 设置html格式参数
html_part = MIMEText(html_content,'html','utf-8')
message.attach(html_part)

# ---这是附件部分---
# 添加一个txt文本附件
with open(r'C:\temp\测试文本.txt','rb') as tf:
txt_content = tf.read()
txt_part = MIMEApplication(txt_content)
txt_part.add_header('Content-Disposition', 'attachment', filename="测试.txt")
message.attach(txt_part)

# 添加一个xlsx类型附件
with open(r'C:\temp\测试Excel.xlsx','rb') as ef:
excle_content = ef.read()
excle_part = MIMEApplication(excle_content)
excle_part.add_header('Content-Disposition', 'attachment', filename="测试Excel.xlsx")
message.attach(excle_part)

# 添加一个docx类型附件
with open(r'C:\temp\测试word.docx','rb') as df:
word_content = df.read()
word_part = MIMEApplication(word_content)
word_part.add_header('Content-Disposition', 'attachment', filename="测试word.docx")
message.attach(word_part)

# 添加一个pdf类型附件
with open(r'C:\temp\测试PDF.pdf','rb') as pf:
pdf_content = pf.read()
pdf_part = MIMEApplication(pdf_content)
pdf_part.add_header('Content-Disposition', 'attachment', filename="测试PDF.pdf")
message.attach(pdf_part)

# 添加一个png类型附件
with open(r'C:\temp\测试图片.png','rb') as gf:
png_content = gf.read()
png_part = MIMEApplication(png_content)
png_part.add_header('Content-Disposition', 'attachment', filename="测试图片.png")
message.attach(png_part)

# 登录并发送
try:
smtpObj = smtplib.SMTP_SSL()
# 连接到服务器
```

```
smtpObj.connect(mail_host, 465)
# 登录到服务器
smtpObj.login(mail_user, mail_password)
# 发送
smtpObj.sendmail(
sender, receivers, message.as_string())
# 退出
smtpObj.quit()
print('success')
except smtplib.SMTPException as e:
print('error',e)
```

接收Email

POP3和IMAP

POP是指邮局协议，目的是让用户可以访问邮箱服务器中的邮件，允许用户从服务器上把邮件存储到本地主机（即自己的计算机）上,同时删除保存在邮件服务器上的邮件，而POP3服务器则是遵循POP3协议的接收邮件服务器，用来接收电子邮件的。

后来又出现了IMAP协议(Interactive Mail Access Protocol),即交互式邮件访问协议，与POP3的不同在于：开启了IMAP后，在电子邮件客户端收取的邮件仍然保留在服务器上，同时在客户端上的操作都会反馈到服务器上，如：删除邮件，标记已读等，服务器上的邮件也会做相应的动作

使用POP3

python的poplib模块支持POP3,基本步骤:

1. 连接到服务器
2. 登录
3. 发出服务请求
4. 退出

poplib的常用方法:

方法	描述
POP3(server)	实例化POP3对象，server是pop服务器地址
user(username)	发送用户名到服务器，等待服务器返回信息
pass_(password)	密码
stat()	返回邮箱的状态,返回2元祖(消息的数量,消息的总字节)
list([msgnum])	stat()的扩展，返回一个3元祖(返回信息, 消息列表, 消息的大小)，如果指定msgnum，就只返回指定消息的数据
retr(msgnum)	获取详细msgnum，设置为已读，返回3元组(返回信息, 消息msgnum的所以内容, 消息的字节数)，如果指定msgnum，就只返回指定消息的数据
dele(msgnum)	将指定消息标记为删除

quit()	登出，保存修改，解锁邮箱，结束连接，退出

```
from poplib import POP3
p = POP3('pop.xxxx.com')
p.user('xxxx@xxxx.com')
p.pass_('xxxxxxxx')
p.stat()
...
p.quit()
```

使用IMAP

python中的imaplib包支持IMAP4
常用方法:

方法	描述
IMAP4(server)	与IMAP服务器建立连接
login(user, pass)	用户密码登录
list()	查看所有的文件夹(IMAP可以支持创建文件夹)
select()	选择文件夹默认是”INBOX”
search() 三个参数，第一的是CHARSET,通常为None(ASCII),第二个参数不知到是干什么官方没解释	

Example IMAOP4接收

```
import getpass, imaplib
M = imaplib.IMAP4()
M.login(getpass.getuser(), getpass.getpass())
M.select()
typ, data = M.search(None, 'ALL')
for num in data[0].split():
typ, data = M.fetch(num, '(RFC822)')
print('Message %s\n%s\n' % (num, data[0][1]))
M.close()
M.logout()
```

相关连接

- smtplib模块:https://docs.python.org/3/library/smtplib.html

- email模块:https://docs.python.org/3/library/email.html
- poplib模块:https://docs.python.org/3/library/poplib.html
- imaplib模块:https://docs.python.org/3/library/imaplib.html

数据库

1.RPA连接MySQL数据库

RPA连接MySQL数据库

阿里云RPA可以使用进行MySQL数据库操作的主要第三方库主要四个，mysqlclient、PyMySQL、peewee和SQLAlchemy。下面我们主要介绍如何在RPA中使用PyMySQL操作MySQL数据库。

1. 环境准备

首先需要使用阿里云RPA的第三方库打包工具 rpapack 将本地python环境中的 PyMySQL 打包成RPA可识别的 rpax 文件，然后在阿里云RPA开发工具 studio 中就打包好的 PyMySQL.rpax 文件导入。

执行的命令：

```
pip install PyMySQL # 先在本地的python环境中安装PyMySQL库
pip install rpapack # 安装阿里云RPA的第三方库打包工具
python -m rpapack PyMySQL # 打包PyMySQL
```

2.使用

2.1 使用方法说明

数据库连接

```
connection = pymysql.connect(host='localhost',
user='user',
password='passwd',
db='db',
charset='utf8mb4',
cursorclass=pymysql.cursors.DictCursor)
```

详细的参数说明请查看Connection Object。

游标

连接完数据库，接着就该获取游标，之后才能进行执行、提交等操作
cursor = connection.cursor()
查询时，默认返回的数据类型为元组，可以修改返回类型
几种常用游标类型：

- Cursor: 默认，元组类型
- DictCursor: 字典类型
- SSCursor: 无缓冲元组类型
- SSDictCursor: 无缓冲字典类型

无缓冲游标类型，适用于数据量很大，一次性返回太慢，或者服务端带宽较小
创建连接时，通过cursorclass 参数指定类型：

```
connection = pymysql.connect(host='localhost',
user='root',
password='root',
db='db',
charset='utf8',
cursorclass=pymysql.cursors.DictCursor)
```

也可以在创建游标时指定类型：

```
cursor = connection.cursor(cursor=pymysql.cursors.DictCursor)
```

游标移动

所有的数据查询操作均基于游标，我们可以通过cursor.scroll(num, mode)控制游标的位置。

```
cursor.scroll(1, mode='relative') # 相对当前位置移动
cursor.scroll(2, mode='absolute') # 相对绝对位置移动
```

sql语句

我的习惯是先写好要操作的语句，如插入、更新、删除等，同时也要注意 **pymysql** 中所有的有关**更新**数据（**insert**，**update**，**delete**）的操作都需要 **commit**，否则无法将数据提交到数据库。

```
# 插入语句
insert_sql = "insert into `jd_huawei` (`pid`, `url`, `price`, " \
"`refprice`, `name`, `comment_num`, `comment_type`)values('%s','%s','%s'," \
"'%s','%s','%s','%s')" % (info_id, info_url, price, refprice, name, comment_num, comment_types)
# 查询语句
select_sql = "select `pid` from `jd_huawei` where `pid`='%s'" % info_id
```

执行sql语句

cursor.execute(sql, args)
这里的参数 args 可以是，tuple，list，dict，另外 execute 还能返回受影响的行数。

```
influence_num = cursor.execute(sql, args)
print(type(influence_num)) # int
```

cursor.executemany(sql, args)可以批量执行SQL语句

查询获取数据

取出全部的数据，可以返回一个结果集

```
cursor.fetchall()
```

取出一定数量的数据

```
cursor.fetchmany(size)
```

取出一条数据

```
cursor.fetchone()
```

事务处理

事务开始

```
connection.begin()
```

提交修改

```
connection.commit()
```

事务回滚

```
connection.rollback()
```

关闭连接

关闭游标

```
cursor.close()
```

关闭数据库连接

```
connection.close()
```

2.2 PyMySQL示例

接下来我们来看一下PyMySQL的示例：
示例使用一个简单的表：

```
# 表结构
CREATE TABLE `users` (
`id` int(11) NOT NULL AUTO_INCREMENT, # id 整型 不能为空 自动增长
`email` varchar(255) COLLATE utf8_bin NOT NULL, # 邮箱 可变字符串 区分大小写，不能为空
`password` varchar(255) COLLATE utf8_bin NOT NULL, # 密码 可变字符串 区分大小写，不能为空
PRIMARY KEY (`id`) # id 为主键
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin
AUTO_INCREMENT=1 ;
# InnoDB 引擎 默认 utf-8 编码 区分大小写 自动增长从1开始


import pymysql.cursors

# 连接数据库
connection = pymysql.connect(host='localhost',
user='user',
password='passwd',
db='db',
```

```
charset='utf8mb4',
cursorclass=pymysql.cursors.DictCursor)

try:
with connection.cursor() as cursor:
# 创建一条新的记录
sql = "INSERT INTO `users` (`email`, `password`) VALUES (%s, %s)"
cursor.execute(sql, ('webmaster@python.org', 'very-secret'))

# 连接完数据库并不会自动提交，所以需要手动 commit 你的改动
connection.commit()

with connection.cursor() as cursor:
# 读取单条记录
sql = "SELECT `id`, `password` FROM `users` WHERE `email`=%s"
cursor.execute(sql, ('webmaster@python.org',))
result = cursor.fetchone()
print(result)
finally:
connection.close()
```

这里注意连续用了两处 **with** 好处就在于 **with** 结束后会自动 close cursor 而免去了 cursor.close()

该示例输出：

```
{'password': 'very-secret', 'id': 1}
```

相关链接

PyMySQL官方文档

2.SQL Server

```
conn = pymssql.connect(server, user, password, "连接默认数据库名称")  #获取连接
```

Parameters/参数说明

参数名称	参数类型	参数说明
server	string	IP地址

user	string	用户名称
password	string	密码
DbName	string	连接默认数据库名称

Return Value/返回信息

暂无

Remarks/备注

暂无

Related/相关

暂无

Example/实例

Example SqlServer 连接

```
import pymssql

server = "187.32.43.13" # 连接服务器地址
user = "root" # 连接帐号
password = "1234" # 连接密码

conn = pymssql.connect(server, user, password, "连接默认数据库名称") #获取连接

cursor = conn.cursor() # 获取光标

# 创建表
cursor.execute("""
IF OBJECT_ID('persons', 'U') IS NOT NULL
DROP TABLE persons
CREATE TABLE persons (
id INT NOT NULL,
```

```
name VARCHAR(100),
salesrep VARCHAR(100),
PRIMARY KEY(id)
)
""")

# 插入多行数据
cursor.executemany(
"INSERT INTO persons VALUES (%d, %s, %s)",
[(1, 'John Smith', 'John Doe'),
(2, 'Jane Doe', 'Joe Dog'),
(3, 'Mike T.', 'Sarah H.')])
# 你必须调用 commit() 来保持你数据的提交如果你没有将自动提交设置为true
conn.commit()

# 查询数据
cursor.execute('SELECT * FROM persons WHERE salesrep=%s', 'John Doe')

# 遍历数据（存放到元组中）方式1
row = cursor.fetchone()
while row:
print("ID=%d, Name=%s" % (row[0], row[1]))
row = cursor.fetchone()
# 遍历数据（存放到元组中）方式2
for row in cursor:
print('row = %r' % (row,))

# 关闭连接
conn.close()
```

Example 存储过程

```
import pymssql

server = "187.32.43.13" # 连接服务器地址
user = "root" # 连接帐号
password = "1234" # 连接密码

with pymssql.connect(server, user, password, "你的连接默认数据库名称") as conn:
with conn.cursor(as_dict=True) as cursor: # 数据存放到字典中
cursor.execute('SELECT * FROM persons WHERE salesrep=%s', 'John Doe')
for row in cursor:
print("ID=%d, Name=%s" % (row['id'], row['name']))
```

3.Oracle

```
#方法一：用户名、密码和监听分开写
import cx_Oracle
db=cx_Oracle.connect('username/password@host/orcl')
db.close()

#方法二：用户名、密码和监听写在一起
import cx_Oracle
db=cx_Oracle.connect('username','password','host/orcl')
db.close()

#方法三：配置监听并连接
import cx_Oracle
tns=cx_Oracle.makedsn('host',1521,'orcl')
db=cx_Oracle.connect('username','password',tns)
db.close()
```

Parameters/参数说明

参数名称	参数类型	参数说明

Return Value/返回信息

暂无

Remarks/备注

暂无

Related/相关

1.引用模块cx_Oracle
2.连接数据库
3.获取cursor
4.使用cursor进行各种操作
5.关闭cursor
6.关闭连接

Example/实例

Example Oracle 连接

```
import cx_Oracle                                    #引用模块cx_Oracle
conn=cx_Oracle.connect('load/123456@localhost/ora11g') #连接数据库
c=conn.cursor() #获取cursor
x=c.execute('select * from Rpa_Info') #使用cursor进行操作
x.fetchone()
c.close() #关闭cursor
conn.close() #关闭连接
```

Example Oracle 连接操作

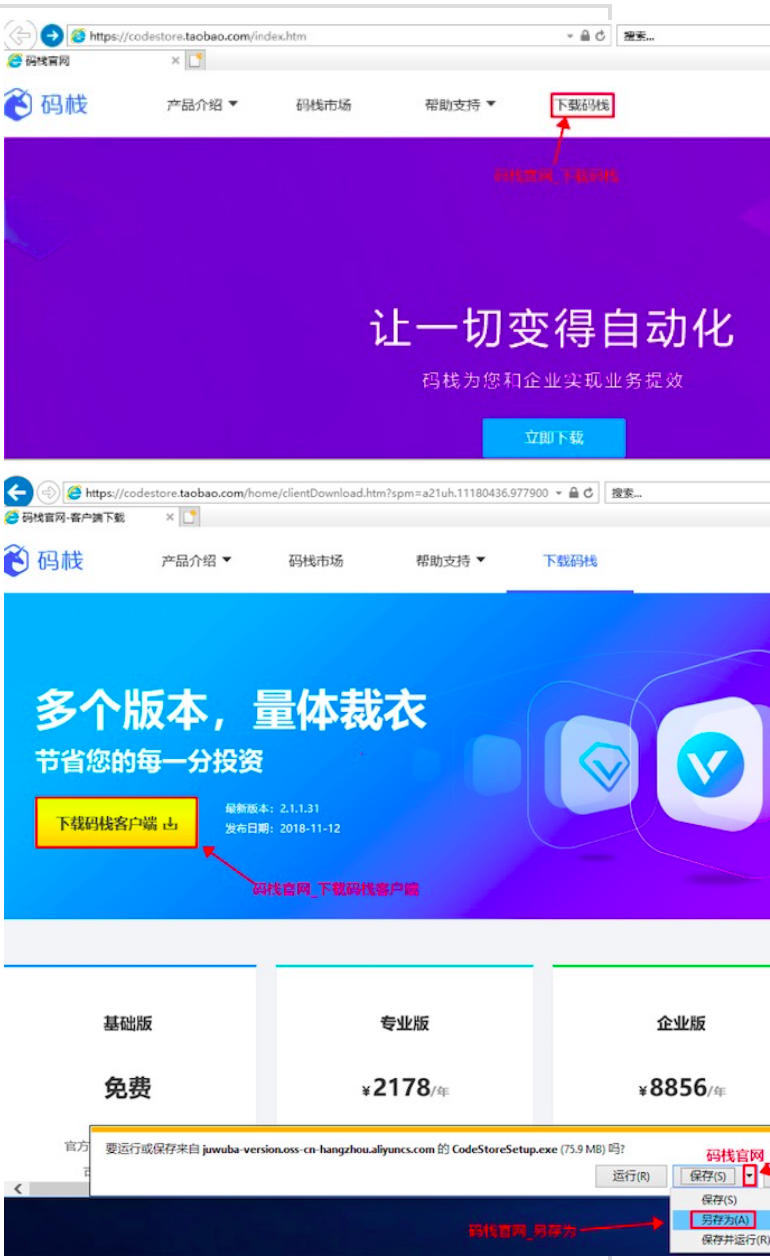
```
import cx_Oracle
def Oracle_Connet():
db=cx_Oracle.connect('username/password@host/orcl')
cr=db.cursor()
Strsql='select * from Rpa_info'
rs=execute(Strsql)
#返回所有结果集
print('all:(%s)'%rs)
for x in rs
print (x)

#返回行
while(1):
rs=cr.fetchone()
if rs==None:break
db.close()

#参数
#字典
pr={'ID':'123'}
Strsql=Strsql + ' where ID=:ID'
cr.execute(Strsql,pr)
#传参
Strsql=Strsql + ' where ID=456'
cr.execute(Strsql,pr)
cr.close()
db.close()
pass
```

IE上传下载

1.IE download

Example IE9 IE10 IE11

版本	网页控件录制图示
IE9	
IE10	
IE11	




```
import rpa
import os
def uploaddownload():
    page = rpa.ie.create('https://codestore.taobao.com/index.htm')
    page.click("码栈官网下载码栈",simulate=True)
    page.wait('码栈官网下载码栈客户端') #等待控件加载
    page.click("码栈官网下载码栈客户端") #通过点击代替直接使用download
    rpa.win32.wait("码栈官网保存下拉", timeout=20) #wait方法暂时只支持桌面控件录制(图像)录制的控件
    rpa.win32.click("码栈官网保存下拉")
    rpa.win32.click("码栈官网_另存为") #自定义存储路径时需要录制保存下拉框内的另存为控件
```

rpa.win32.save_file_dialog(r’ C:\test\codestore.exe’ ,title = ‘另存为’) sleep(3) page.close()	

Example IE7 IE8

版本	网页控件录制图示	
IE7(不支持)		
IE8		
import rpa import os def upload*download():* *page =* *rpa.ie.create(‘https://codestore.taobao.com/index.htm ’)* *page.click(“码栈官网*下载码栈”,simulate=True) page.wait(‘码栈官网 *下载码栈客户端’) #等待控件加载* *page.click(“码栈官网*下载码栈客户端”) #通过点击代替直接使用download rpa.win32.wait(“码栈官网保存”, timeout=20) #wait方法暂时只支持桌面控件录制(图像)录制的控件* *rpa.win32.click(“码栈官网保存”)* rpa.win32.save_file_dialog(r’		

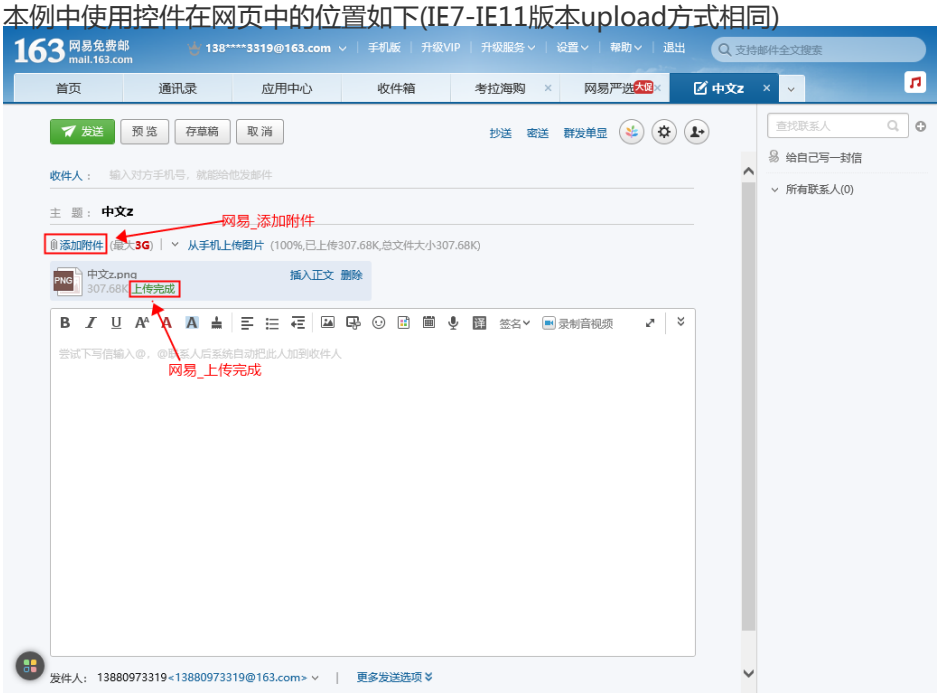
C:\test\codestore.exe’ ,title = ‘另存为’) sleep(3) page.close()		

#

2.IE upload

Example

本例中使用控件在网页中的位置如下(IE7-IE11版本upload方式相同)



请先手动登录163官网,并进入写信界面后再使用本例代码操作

```
import rpa
def ie11upload():
page = rpa.ie.catch('网易邮箱6.0版',mode = 'title') #catch用户手动打开的写信界面
page.upload("网易_添加附件", r"C:\test\中文z.png") #上传图片到附件
sleep(3) #上传过程等待
page.wait("网易_上传完成",timeout=30) #等待上传操作后,上传完成控件加载
text = page.value("网易_上传完成") #获取上传结果,显示是否上传成功
```

```
print(text)
if text == str("上传完成"):
print('upload上传成功！')
else:
print('upload上传失败！')
sleep(5)

page.close()
```

输入框输入最佳实践

输入框输入最佳实践

应用场景

我们对各种控件进行操作使用率最高的两个方法分别是 输入 和 点击 ，而输入这个操作分为两种方式：对输入框输入内容和输入键盘快捷键来搭配使用。我们对控件 输入 的SDK一共有4个分别是： input 、 key_send 、 input_text 和 input_hotkeys 其中 input 、 input_text 和 input_hotkeys 应用于 IE 、 Chrome 、 win32 和 Java 控件； key_send 应用于 win32 控件。我们在使用上述 输入 方法时，避免受中文法干扰，正调用SDK的输入 方法前先将系统输入输入法切换为英文输入法。

示例：

```
from rpa.core import *
from rpa.utils import *
import rpa3 as rpa # 使用V3引擎
from win32con import WM_INPUTLANGCHANGEREQUEST
import win32gui
import win32api

def start():
# 打开一个网页，使用input_text方法输入文字
page = rpa.ie.create("www.taobao.com")

# 设置输入法为英文输入法
set_English_input_method()
```

```
# 对页面输入框输入文字，如：在[搜索输入框]中输入'手机'
page.input_text("搜索输入框", "手机")


def set_English_input_method() -> bool:
"""
设置当前的前景窗口输入法为英文输入法。
语言代码：
0x0804: "Chinese (Simplified) (People's Republic of China)"
0x0409: 'English (United States)'
:return: 成功为True，失败为False
"""
# 获取前景窗口句柄
hwnd = win32gui.GetForegroundWindow()
# 设置键盘布局为英文
result = win32api.SendMessage(hwnd, WM_INPUTLANGCHANGEREQUEST, 0, 0x0409)
return True if result == 0 else False
```

相关连接

- Win32 API

打印文档

1.打印图片

```
for y in range(0, height, scale):  #根据缩放长度 遍历高度
for x in range(0, width, scale): #根据缩放长度 遍历宽度
choice =gray_img.getpixel((x, y)) * char_len // 255 #获取每个点的灰度 根据不同的灰度填写相应的 替换字符
if choice==char_len:
choice=char_len-1
sys.stdout.write(char_lst[choice]) #写入控制台
sys.stdout.write('\n')
sys.stdout.flush()
```

Parameters/参数说明

参数名称	参数类型	参数说明
StrFile	string	文件路径

Return Value/返回信息

暂无

Remarks/备注

拿到图片对象，并转换图片模式（ ‘’ L ‘’ ）,L模式可以去到图片的个像素的灰度参数；
定义图片缩放长度、替换字符的串（根据灰度值排序）；

遍历缩放后每一个点位置，并获取该点位置的灰度值，根据灰度值替换为相应的替换字符;
打印在控制台;

Related/相关

利用 python 的 image模块和简单用符号，再控制台打印出图片的图片轮廓

PIL 安装 pip install Pillow

Example/实例

Example 打印图片

```
from PIL import Image
import sys
import os

def PrintImg(StrFile):
try:
pic = os.path.abspath(StrFile) #获取图片路径参数
except:
print(StrFile)
img = Image.open(pic) #获取图片对象
width = img.width #获取图片宽度
height = img.height #获取图片高度

gray_img = img.convert('L') #图片转换为'L'模式 模式“L”为灰色图像，它的每个像素用8个bit表示，0表示黑，255表示白，其他数字表示不同的灰度

scale = width // 100 #图片缩放100长度
char_lst = ' .:-=+*#%@' #要替换的字符
```

```
char_len = len(char_lst) #替换字符的长度

for y in range(0, height, scale): #根据缩放长度 遍历高度
for x in range(0, width, scale): #根据缩放长度 遍历宽度
choice =gray_img.getpixel((x, y)) * char_len // 255 #获取每个点的灰度 根据不同的灰度填写相应的 替换字符
if choice==char_len:
choice=char_len-1
sys.stdout.write(char_lst[choice]) #写入控制台
sys.stdout.write('\n')
sys.stdout.flush()
pass
```

2.打印ExcelWord文档

```
#读取docx中的文本代码示例
import docx
#获取文档对象
file=docx.Document("E:\\test.docx")
print("段落数:"+str(len(file.paragraphs)))#段落数为13，每个回车隔离一段

#输出每一段的内容
for para in file.paragraphs:
print(para.text)

#输出段落编号及段落内容
for i in range(len(file.paragraphs)):
print("第"+str(i)+"段的内容是："+file.paragraphs[i].text)
```

Parameters/参数说明

参数名称	参数类型	参数说明
StrFile	string	docx 文件路径

Return Value/返回信息

```
暂无
```

Remarks/备注

python无法读取.doc文件

问题分析：python利用python-docx (0.8.6)库可以读取.docx文件或.txt文件，且一路畅通无阻，而对.doc文件本身python是无法操作.doc文件是他的先天不足，将.doc文档利用手动的方式“另存为”.docx文档，就能够成功打开转化后的.docx文档，于是我就尝试利用代码方式完成这个手动的“另存为”功能，问题得以解决。

利用python将大批.doc文件转化为.docx文件，再读写.docx文件

Related/相关

暂无

Example/实例

Example 打印word

```
import sys
import pickle
import re
import codecs
import string
import shutil
from win32com import client as wc
import docx

def start():
# 在此处开始编写您的应用
pass

def PrintWord(StrFile):
#获取文档对象
file=docx.Document(StrFile)
print("段落数:"+str(len(file.paragraphs)))#每个回车隔离一段

#输出每一段的内容
for para in file.paragraphs:
print(para.text)

#输出段落编号及段落内容
for i in range(len(file.paragraphs)):
print("第"+str(i)+"段的内容是："+file.paragraphs[i].text)
pass
```

```
def doSaveAas():
word = wc.Dispatch('Word.Application')
doc = word.Documents.Open(u'E:\\code\\test.doc') # 目标路径下的文件

doc.SaveAs(u'E:\\test.docx', 12, False, "", True, "", False, False, False, False) # 转化后路径下的文件
doc.Close()
word.Quit()
pass
```

打印Excel文档

```
for rownum in range(1,nrows):
row = table.row_values(rownum)
if row:
app = {}
for i in range(len(colnames)):
app[colnames[i]] = row[i]
list.append(app)
return list
```

Parameters/参数说明

参数名称	参数类型	参数说明
file	string	docx 文件路径
colnameindex	int	行号
by_index	int	索引

Return Value/返回信息

```
暂无
```

Remarks/备注

```
暂无
```

Related/相关

确保安装好excle的处理包xlrd

安装过程如下：http://blog.csdn.net/weixin_40449300/article/details/79138741

Example/实例

Example 打印Excel

```
import  xdrlib ,sys
import xlrd
def open_excel(file= 'file.xls'):
try:
data = xlrd.open_workbook(file)
return data
except Exception as e:
print(str(e))
pass
#根据索引获取Excel表格中的数据 参数:file：Excel文件路径 colnameindex：表头列名所在行的所以 ，by_index：表的索引
def excel_table_byindex(file= 'E:\\test.xls',colnameindex=0,by_index=0):
data = open_excel(file)
table = data.sheets()[by_index]
nrows = table.nrows #行数
ncols = table.ncols #列数
colnames = table.row_values(colnameindex) #某一行数据
list =[]
for rownum in range(1,nrows):
row = table.row_values(rownum)
if row:
app = {}
for i in range(len(colnames)):
app[colnames[i]] = row[i]
list.append(app)
return list
pass

#根据名称获取Excel表格中的数据 参数:file：Excel文件路径 colnameindex：表头列名所在行的所以 ，by_name：Sheet1名称
def excel_table_byname(file= 'E:\\test.xls',colnameindex=0,by_name=u'电度'):
data = open_excel(file)
table = data.sheet_by_name(by_name)
nrows = table.nrows #行数
colnames = table.row_values(colnameindex) #某一行数据
list =[]
for rownum in range(1,nrows):
row = table.row_values(rownum)
if row:
```

```
app = {}
for i in range(len(colnames)):
app[colnames[i]] = row[i]
list.append(app)
return list
pass

def main():
tables = excel_table_byindex()
for row in tables:
print(row)

tables = excel_table_byname()
for row in tables:
print(row)

pass
```

持久化应用日志

本功能要求客户端版本必须为3.3.31及以上
rpa.logger提供了use_logger方法，用来设置自定义的日志记录程序，结合python的*logging*库即可实现日志持久化的功能。
示例代码:

```
from rpa.core import *
from rpa.utils import *
import rpa
import os
import logging
from logging.handlers import TimedRotatingFileHandler

def use_my_logger():
""" 设置自定义的日志记录程序
使用TimedRotatingFileHandler按日期循环记录日志，可替换为任意handler
"""
logger = logging.getLogger('mylogger')
logger.setLevel(logging.DEBUG) # 设置根日志级别
log_file = os.path.join(os.getenv('APPDATA'), r'AliRPA\log\myapp.log') # 设置日志文件存放位置
(%appdata%\AliRPA\log\app\myapp.log)
formatter = logging.Formatter('%(asctime)s[%(name)s][%(levelname)s] %(message)s')
handler = TimedRotatingFileHandler(log_file, when='D', interval=1)
handler.setFormatter(formatter)
handler.suffix = '%Y%m%d'
handler.setLevel(logging.DEBUG) # 设置当前handler日志级别
logger.addHandler(handler)
rpa.logger.use_logger(logger) # 替换为自定义的logger
```

```
def start():
# 在此处开始编写您的应用
use_my_logger()
print('message 1')
rpa.logger.info('info-message')
rpa.logger.warn('warn-message')
rpa.logger.error('error-message')
```

使用Python激活窗口

方法如下，可以通过修改match方法来改变匹配方式

```
from rpa.core import *
from rpa.utils import *
import rpa
import win32gui, win32con

def start():
# 在此处开始编写您的应用
win_active('记事本')

def win_active(winTitle):
def match(winText):
"""匹配方式：模糊匹配"""
return winText.find(winTitle) > -1
hwnds = match_windows(match)
if hwnds:
win32gui.ShowWindow(hwnds[0], win32con.SW_SHOWMAXIMIZED) # SW_SHOWDEFAULT 默认大小
，SW_SHOWMAXIMIZED 最大化显示
win32gui.SetForegroundWindow(hwnds[0])
win32gui.SetActiveWindow(hwnds[0])

def match_windows(match_func):
def callback(hwnd, hwnds):
if win32gui.IsWindowVisible(hwnd) and win32gui.IsWindowEnabled(hwnd):
winText = win32gui.GetWindowText(hwnd)
if match_func(winText):
hwnds.append(hwnd)
return True
hwnds = []
win32gui.EnumWindows(callback, hwnds)
return hwnds
```

透视表相关操作

注: 客户端版本需要在3.3.26及以上

创建数据视图

```
xls = rpa.excel.open(r'C:\data\案例数据.xlsx')
sheet = xls.get_sheet('口碑汇总数据')
pivot_settings = rpa.excel.PivotTableSettings('MyPivotTable')
pivot_settings.columns['税额'] = {} # 添加"列标签"
pivot_settings.filters['不含税金额'] = {} # 添加"筛选字段"
pivot_settings.rows['税率'] = {} # 添加"行标签"
pivot_settings.values['税额2'] = {"Function": "xlAverage"} # 添加"数值"
pivot_settings.values['购方纳税人识别号'] = {"Function": "xlSum"} # 添加"数值"
sheet.create_pivottable('口碑汇总数据', 'G1:J4952', 'Sanco', 'A1', pivot_settings)
xls.save()
```

Function取值范围参考https://docs.microsoft.com/zh-cn/dotnet/api/microsoft.office.interop.excel.xlconsolidationfunction?redirectedfrom=MSDN&view=excel-pia

刷新数据视图

```
xls = rpa.excel.open(r'C:\Work\案例数据.xlsx')
sheet = xls.get_sheet('Sanco')
sheet.refresh_pivottable(index=1)
xls.save()
```

获取透视表筛选列的所有项

```
xls = rpa.excel.open(r'C:\Work\案例数据.xlsx')
sheet = xls.get_sheet('Sanco')
items = sheet.get_all_pivot_field_items('不含税金额')
print(items)
```

选择/取消选择一组透视表筛选项

```
xls = rpa.excel.open(r'C:\Work\案例数据.xlsx')
```

```
sheet = xls.get_sheet('Sanco')
sheet.select_pivot_field_items('不含税金额', [0.02, 0.03, 0.07], select=False)
xls.save()
```