**信控自动化底层函数封装API说明**

版本：v1.1

目录

[1 . dbPool 3](#_Toc483500312)

[1.1 setPool(db\_info) 3](#_Toc483500313)

[1.2 \_\_init\_\_( db\_info={},switch=1) 3](#_Toc483500314)

[1.3 Columns(table) 4](#_Toc483500315)

[1.4 create\_params(table,args={}) 4](#_Toc483500316)

[1.5 get\_rows(size=None, is\_dict = True) 4](#_Toc483500317)

[1.6 execute(sql,param={}) 5](#_Toc483500318)

[1.7 executemany(column,value={}) 5](#_Toc483500319)

[1.8 execQuery(sql,param={}) 5](#_Toc483500320)

[1.9 executeUD(sql,param={}) 6](#_Toc483500321)

[1.10 execQuery\_pages(sql,args={},page=1,page\_size=50) 6](#_Toc483500322)

[1.11 execUpdate(table,set\_dict={}, where\_dict ={}) 6](#_Toc483500323)

[1.12 execDelete(table, table ,where\_dict ={}) 7](#_Toc483500324)

[1.13 getRowsNum() 7](#_Toc483500325)

[1.14 commit() 7](#_Toc483500326)

[1.15 rollback() 7](#_Toc483500327)

[1.16 close() 8](#_Toc483500328)

[1.17 distroyCache() 8](#_Toc483500329)

[1.18 execProc(proc\_name,param=[]) 8](#_Toc483500330)

[2 . analyzeConf 9](#_Toc483500331)

[2.1 getSect(m\_sectionname) 9](#_Toc483500332)

[2.2 getChildSect(m\_sectionname, m\_childsecname) 9](#_Toc483500333)

[2.3 getSectInfo(m\_sectionname, m\_key=None) 10](#_Toc483500334)

[2.4 insertStr(m\_sectionname, m\_tgtstring, m\_insertstr) 10](#_Toc483500335)

[3 . dataDeal.py 11](#_Toc483500336)

[3.1 find\_sqlFile(self) 11](#_Toc483500337)

[3.2 get\_allFiles(self) 11](#_Toc483500338)

[3.3 execute\_single(self) 11](#_Toc483500339)

[3.4 execute\_batchSqlFiles (self) 11](#_Toc483500340)

[4 . ACTinit.py 12](#_Toc483500341)

[4.1 path\_dict(self) 12](#_Toc483500342)

[4.2 constant\_dict(self) 12](#_Toc483500343)

[5 . baseHelper.py 13](#_Toc483500344)

[5.1 get\_dbInfoByConf() 13](#_Toc483500345)

[5.2 set\_regionOfDbInfo(region, db\_info) 13](#_Toc483500346)

# 1 . dbPool

dbPool.py主要用于创建数据库缓存池，采用多线程并发链接方式与单例设计模式。还提供对数据的增删改查操作等十多个调用接口。

## 1.1 setPool(db\_info)

功能：采用多线程设置数据库缓存池，定义了缓存池提供大小、数据库最大连接数、链接最多共享数等数据库属性

参数说明：db\_info是一个字典，内容是数据库的属性值，db\_info = db\_info{}。

使用说明：

# -\*- coding:utf-8 -\*-

db\_info = {

'dbType' : "cx\_Oracle",

'host' : "10.124.0.41",

'user' : "autotest",

'passwd' : "autotest",

'port' : "1521",

'dbName' : "AUTOTEST",

'charset' : "utf-8"

}

pool = dbPool.dbPool.get\_instance(db\_info)#后边示例代码均用这个pool实例

pool.setPool(db\_info = db\_info)

备注：此函数是静态方法@staticmethod

## 1.2 \_\_init\_\_( db\_info={},switch=1)

功能：（1）初始化dbPool类，设置数据库链接方法\_conn、数据库操作游标方法\_cursor

（2）创建数据库私有链接和共享链接

参数说明：db\_info同上，switch是共享链接与私有链接的开关。默认创建共享链接

@switch=1 : get shareConn共享链接

@switch=0 : get privateConn私有链接

使用说明：

# -\*- coding:utf-8 -\*-

print(u'创建共享链接')

#pool.shareOrPrivate()

pool = dbPool.dbPool.get\_instance(db\_info)

print('\n')

print(u'创建私有链接')

pool = dbPool.dbPool.get\_instance(db\_info)

print('\n')

print(u'创建数据库操作游标构造器')#\_conn,\_cursor

pool = dbPool.dbPool.get\_instance(db\_info)

## 1.3 Columns(table)

功能：获取数据库某个表所有的字段（列），返回值是list

参数说明：table表名

使用说明：

# -\*- coding:utf-8 -\*-

column = pool.columns("tf\_f\_user")

print(column)

## 1.4 create\_params(table,args={})

功能：自动创建参数字典，供其他将字典作为参数的函数调用。也可用于校验传入的表字段是否正确。

参数说明：table表名，args表字段组成的字典

使用说明：

# -\*- coding:utf-8 -\*-

dt = pool.create\_params('tf\_f\_user',{'user\_id':'b','serial\_number1':'a'})

print(dt)

print('\n')

由于tf\_f\_user没有serial\_number1字段，所以返回{'user\_id': 'b'}

## 1.5 get\_rows(size=None, is\_dict = True)

功能：主要用于加工从数据库捞取的数据，格式化大小写，返回值是字典组成的列表。

参数说明：size是sql执行得到结果集的大小，默认为空，捞取所有的符合条件的结果。

is\_dict：ture时将每条数据结果，存入字典中。False时原格式，数据格式是list

使用说明：

# -\*- coding:utf-8 -\*-

#由于字典形式方便我们获取和操作数据，所以使用时is\_dict使用默认值即可

pool. get\_rows()

## 1.6 execute(sql,param={})

功能：执行sql语句

参数说明：sql语句，param{}用于手工绑定变量组成的字典

使用说明：

# -\*- coding:utf-8 -\*-

sql = "select \* from tf\_f\_user where user\_id= :user\_id"

param = {':user\_id':'1113110500000803'}#变量绑定

rst = pool.execute(sql,param)

print("%s\n" %rst)

print(rst[0]['user\_id'],rst[0]['serial\_number'])#取值

## 1.7 executemany(column,value={})

功能：执行多条sql语句

参数说明：column表字段名，value{}用值列表字典

使用说明：

# -\*- coding:utf-8 -\*-

主要用户批量插入，自动绑定变量模式

## 1.8 execQuery(sql,param={})

功能：执行查询sql语句

参数说明：sql语句，param{}用于手工绑定变量组成的字典

使用说明：

# -\*- coding:utf-8 -\*-

sql = "select \* from tf\_f\_user where user\_id= :user\_id"

param = {':user\_id':'1113110500000803'}#变量绑定

rst = pool.execute(sql,param)

print("%s\n" %rst)

print(rst[0]['user\_id'],rst[0]['serial\_number'])#取值

注：本方法是1.6的子集，这里只是便于使用者理解单独创建查询函数

## 1.9 executeUD(sql,param={})

功能：执行更新和插入sql语句，返回值是本次操作影响的行数

参数说明：sql语句，param{}用于手工绑定变量组成的字典

使用说明：

# -\*- coding:utf-8 -\*-

sql = 'UPDATE tf\_f\_resource set validtag=:validtag,totalaccount=:totalaccount where resourceinsid=:resourceinsid'

param = {'resourceinsid': '1000000000000013', 'totalaccount': 101010, 'validtag': 0}

pool.executeUD(sql,param)

pool.commit()

## 1.10 execQuery\_pages(sql,args={},page=1,page\_size=50)

功能：分页查询，为做测试报告报表使用

参数说明：sql语句，args{}参数字典{'字段1'：'值1','字段2':'值2'}，page页码，page\_size分页大小

使用说明：

# -\*- coding:utf-8 -\*-

与1.6类似

## 1.11 execUpdate(table,set\_dict={}, where\_dict ={})

功能：自动绑定变量，更新操作。返回值是本次操作影响的行数

参数说明：table：表名

set\_dict ：set c1=v1 and c2=v2 等价于 {'c1':'v1','c2':'v2',...}

where\_dict：where c1=v1 and c2=v2 等价于 {'c1':'v1','c2':'v2',...}

使用说明：

# -\*- coding:utf-8 -\*-

print(u'更新功能（自动绑定变量模式）')

set\_dict = {'totalaccount':101010,'validtag':0}

where\_dict = {'resourceinsid':'1000000000000013'}

a = pool.execUpdate(table,set\_dict,where\_dict)

pool.commit()

print(u"本次操作成功数: ",a)

print('down!\n')

## 1.12 execDelete(table, table ,where\_dict ={})

功能：自动绑定变量，删除操作。返回值是本次操作影响的行数

参数说明：table：表名

where\_dict：where c1=v1 and c2=v2 等价于 {'c1':'v1','c2':'v2',...}

使用说明：

# -\*- coding:utf-8 -\*-

print(u'删除功能（自动绑定变量模式）')

table = 'tf\_f\_resource'

where\_dict = {'resourceinsid':'1000000000000017'}

a = pool.execDelete(table,where\_dict)

pool.commit()

print(u"本次操作成功数: ",a)

print('down!\n')

## 1.13 getRowsNum()

功能：辅助函数，返回sql执行后影响的行数

参数说明：

使用说明：

# -\*- coding:utf-8 -\*-

略

## 1.14 commit()

功能：事物提交

参数说明：

使用说明：

# -\*- coding:utf-8 -\*-

略

## 1.15 rollback()

功能：事物回滚

参数说明：

使用说明：

# -\*- coding:utf-8 -\*-

略

## 1.16 close()

功能：事物提交，关闭数据库操作游标，关闭数据库链接

参数说明：

使用说明：

# -\*- coding:utf-8 -\*-

略

## 1.17 distroyCache()

功能：清空缓存池和线程池

参数说明：

使用说明：

# -\*- coding:utf-8 -\*-

略

## 1.18 execProc(proc\_name,param=[])

功能：调用执行存储过程,返回影响数据条数

参数说明：proc\_name 存储过程名字，string。

param 被执行存储过程需要的参数，list。

使用说明：

# -\*- coding:utf-8 -\*-

proc\_name = “p\_example\_l”

param = [‘a’,’b’,’c’,’d’,……]

a = execProc(proc\_name, param)

print(a)

# 2 . analyzeConf

ParseConf类，用于解析配置文件、获取节点内容、节点字段值、插入字符串等，

使用前进行实例化，对象调用相应的方法实现功能。

## 2.1 getSect(m\_sectionname)

功能：获取节点内容

参数说明：节点名

使用说明：

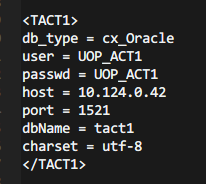
类ParseConf实例化，传入配置文件路径后，直接调用，参数传入需要获取内容的节点名，大小敏感

confpath可置空，默认配置文件地址为/ngbss/credit/practice/code/etc/autotest.conf

pc = ParseConf(confpath)

pc.getSect(‘TACT1’)返回值为节点TACT1内的所有内容，格式与配置文件保持一致，例如

（配置文件截图）



（返回值如下）

db\_type = cx\_Oracle

user = UOP\_ACT1

passwd = UOP\_ACT1

host = 10.124.0.42

port = 1521

dbName = tact1

charset = utf-8

## 2.2 getChildSect(m\_sectionname, m\_childsecname)

功能：获取子节点内容

参数说明：节点名、子节点名

使用说明：

类ParseConf实例化，传入配置文件路径后，直接调用，参数传入需要获取内容的节点名，大小敏感

confpath可置空，默认配置文件地址为/ngbss/credit/practice/code/etc/autotest.conf

pc = ParseConf(confpath)

pc.getChildSect(‘TACT1’,’TYPE’)返回值为子节点TYPE内的所有内容，类型为str，格式与配置文件保持一致

## 2.3 getSectInfo(m\_sectionname, m\_key=None)

功能：获取节点内字段值

参数说明：节点名，字段key(非必传参数)

使用说明：

类ParseConf实例化，传入配置文件路径后，直接调用，参数传入需要获取内容的节点名，大小敏感

confpath可置空，默认配置文件地址为/ngbss/credit/practice/code/etc/autotest.conf

pc = ParseConf(confpath)

pc.getSectInfo(‘TACT1’)返回值为节点TACT1内的所有内容，类型为dict；

pc.getSectInfo(‘TACT1’,’db\_type’)返回值为节点TACT1内字段db\_type的值，类型为str，如字段有误，返回No such key提示；

## 2.4 insertStr(m\_sectionname, m\_tgtstring, m\_insertstr)

功能：指定节点的指定位置插入指定字符串

参数说明：节点名，指定位置（后面插入），带插入字符串

使用说明：

类ParseConf实例化，传入配置文件路径后，直接调用，参数传入需要获取内容的节点名，大小敏感

confpath可置空，默认配置文件地址为/ngbss/credit/practice/code/etc/autotest.conf

pc = ParseConf(confpath)

pc.insertStr(‘JF\_INTERFACE’, ‘user\_id in (’, ‘123124232123123’)插入成功返回1，失败返回0，插入值如已存在，提示This string has already existed!，返回失败0

# 3 . dataDeal.py

信控自动化数据分为两大类：第一类是基础数据，包括三户资料等信息，第二类是业务数据，包括各类工单表数据等信息。导入第二类数据采用单个进程操作，导入第一类数据采用动态计算多进程并行导入操作。基于此dataDeal.py通过前台wiki参数控制实现以上两大类功能，具体某个用例所需的基础数据与业务数据导入到oracle；做冒烟测试时候，所有用例基础数据与业务数据批量导入oracle。

## 3.1 find\_sqlFile(self)

功能：递归查找基础数据目录下以及子目录下目标文件。基础数据目录是在配置文件中配置的。根据实例化时传入的自动化用例名称去模糊匹配sql文件，找出匹配到的并返回。

参数说明：无

使用说明：

c = dataDeal('jkAndSend', '6')

print(c. find\_sqlFile())#成功找到会返回文件名

## 3.2 get\_allFiles(self)

功能：获取基础数据目录下所有文件

参数说明：无

使用说明：

c = dataDeal('jkAndSend', '6')

print(c.get\_allFiles())

## 3.3 execute\_single(self)

功能：执行指定的sql文件，导入指定用例基础数据。根据3.1获取的sql文件来执行。

参数说明：无

使用说明：

c = dataDeal('jkAndSend', '6')

c. execute\_single ()

## 3.4 execute\_batchSqlFiles (self)

功能：执行指定目录下所有sql文件，根据配置文件设置了每个进程执行的sql文件数量m、以及和合理性阀值n。通过获取目录下sql文件数量s，然后计算创建进程池，然后并行执行导入sql文件。

参数说明：无

使用说明：

c = dataDeal('6')

print(c. execute\_single ())

注：批量执行只需要传入导入域即可，所以实例化时不需要传入测试用例名称

# 4 . ACTinit.py

用来存储信控自动化后台，所有用到的初始化值。目的是为了方便后续维护！

内容包括：

# 1.路径 ，比如：配置文件路径、常用目录路径等

# 2.全局常量，比如：事务控制常量、时间间隔常量等

# 等,可以随时增加。

#增加参数规范：【初始化路径专用字典】

# 1.key:需要初始化文件名\_\*\*\*\*

# 2.value:string、int等类型变量

注意：追加内容时，一定注意加上“,”！

## 4.1 path\_dict(self)

功能：初始化路径专用字典

使用方法：

path\_dict = {

'analyzeConf\_path' : '/ngbss/credit/practice/code/etc/autotest.conf', #解析配置文件类使用初始化值

'dataDeal\_path' : '/ngbss/credit/practice/code/basesql/credit/' #信控自动化数据脚本sql文件目录

}

## 4.2 constant\_dict(self)

功能：初始化常量专用字典

使用方法：

constant\_dict = {

'TRANSACTION\_SQL' : 50, #数据库事务控制阀值

'SUBPROCESS\_LIMIT' : 500, #批量导入基础sql文件，动态控制进程池大小，即一个进程执行SUBPROCESS\_LIMIT个sql文件

'PROCESS\_SPECIAL' : 750 #特殊处理，当基础数据sql文件少于PROCESS\_SPECIAL时没必要创建进程池

}

# 5 . baseHelper.py

辅助baseauto底层功能实现的方法集合

## 5.1 get\_dbInfoByConf()

功能：获取一个默认节点，作为数据库默认链接库，作为模板供替换为目标库

参数：无

使用说明：

Utils = baseUtils()

Utils. get\_dbInfoByConf()

## 5.2 set\_regionOfDbInfo(region, db\_info)

功能：根据参数传入域，替换5.1获取的模板库得到目标库

参数：region数据库域 db\_info 模板数据库tns

使用说明：

Utils = baseUtils()

Utils. set\_regionOfDbInfo(‘3’, Utils. get\_dbInfoByConf())