链滴

以 OneBlog 开源博客为模板的个人博客自动加友链脚本 -python 版

作者：wylc

原文链接：https://ld246.com/article/1604282512844

来源网站：链滴

使用背景

个人SEO小工具，作用解放双手不在Ctrl C Ctrl V，知道OneBlog博客的links页地址就行。

上代码：

1. 自动友链业务代码

```
#!usr/bin/env python
# -*- coding: utf-8 -*-
#!文件类型：  python
#!创建时间：  2020/10/22 9:12
#!作者：  SongBin
#!文件名称：  AutoFriendLinks.py
#!简介：自动添加OneBlog博客友链
# coding:utf8
# python3
import re

from selenium import webdriver
import time
import random
import linecache
from selenium.webdriver.chrome.options import Options
from utils.MySqlConn import MyPymysqlPool
import logging
logging.basicConfig(level=logging.INFO,#控制台打印的日志级别
                    filename='C:\links\logs\output.log',
                    filemode='a',##模式，有w和a，w就是写模式，每次都会重新写日志，覆盖之前的日志
                    #a是追加模式，默认如果不写的话，就是追加模式
                    format=
                    '%(asctime)s - %(pathname)s[line:%(lineno)d] - %(levelname)s: %(message)s'
                    #日志格式
                    )
# 打开浏览器
logging.error("程序开始运行############")
print("程序开始运行############")
options = webdriver.FirefoxOptions()
# options.set_headless()
# chrome_options = Options()
# chrome_options.add_argument("--headless")
browser = webdriver.Firefox(options=options)
# browser = webdriver.Chrome(executable_path='C:\\softs\\chrome\\chromedriver.exe',optio
s=chrome_options)  # .Firefox()  # .PhantomJS(desired_capabilities=dcap) # executable_path=
/usr/local/bin/phantomjs'  phantomjs没有设置环境变量时可加参数
count = 1
while count > 0:
    url_input = input('请输入OneBlog友链添加地址: ')
    browser.get(url_input)
    linkStr = browser.find_element_by_tag_name("blockquote").text
    print(linkStr)
    text1 = re.findall('复制添加本站链接：.+.</a>', linkStr)[0]
```

```
    url = re.findall('https?://(?:[-\w.]|(?:%[\da-fA-F]{2}))+', text1)[0]
    print(url)
    descStr = re.findall('title.+." target', text1)[0]
    arg = descStr.split('"')
    desc = arg[1]
    print(desc)
    titleStr = re.findall('>.+.<',text1)[0]
    title = titleStr[1:-1]
    print(title)
    text2 = re.findall('复制添加本站图标：.+', linkStr)[0]
    favicon = re.findall('https?://.+.',text2)[0]
    print(favicon)
    # 获取要推广的博客列表（mysql中获取）
    mysql = MyPymysqlPool("MyBlogMysql")
    sql = "select * FROM mto_links where url = '" +url+ "' limit 1"
    links = mysql.getAll(sql)
    mysql.dispose()
    # num = len(links)
    # print(num)
    if links==False:
        # //添加他人友链到博客
        nowtime =time.strftime('%Y-%m-%d %H:%M:%S',time.localtime(time.time()))
        mysql = MyPymysqlPool("MyBlogMysql")
        insertSql = "insert into mto_links (url,name,description,favicon,status,home_page_display
source,create_time,update_time,created,updated) values (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s
"
        vals = (url,title,desc,favicon,1,1,'OTHER',nowtime,nowtime,nowtime,nowtime)
        newID = mysql.insertOneGetId(insertSql, vals)
        print(newID)
        mysql.dispose()
        # 添加自己友链到别人博客
        browser.find_element_by_class_name("auto-link-btn").click()
        browser.find_element_by_name("name").send_keys("IT源点")
        browser.find_element_by_name("url").send_keys("https://www.daxueyiwu.com/links/all")
        browser.find_elements_by_name("description")[1].send_keys("技术源泉,入行源点。分享自
，学习别人。")
        browser.find_element_by_name("email").send_keys("1370811553@qq.com")
        browser.find_element_by_name("favicon").send_keys("https://www.daxueyiwu.com/dist/
mages/logo/m.png")
        browser.find_element_by_class_name("autoLink").click()
        time.sleep(10)
        # 删除友链首页显示
        mysql = MyPymysqlPool("MyBlogMysql")
        upsql = 'update mto_links set home_page_display = 0 where id = ' +  str(newID)
        flag = mysql.update(upsql)
        mysql.dispose()
    else:
        print("该网站已经添加过友链！")
```

2. Mysql数据库连接工具类

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

```
#!文件类型：  python
#!创建时间：  2018/12/3 16:46
#!作者：  SongBin
#!文件名称：  MySqlConn.py

import pymysql, os, configparser
from pymysql.cursors import DictCursor
from DBUtils.PooledDB import PooledDB


class Config(object):
    """
    # Config().get_content("user_information")

    配置文件里面的参数
    [notdbMysql]
    host = 192.168.1.101
    port = 3306
    user = root
    password = python123
    """

    def __init__(self, config_filename="mysqlConfig.cfg"):#C:\\softs\\csdn_pl\\utils\\
        file_path = os.path.join(os.path.dirname(__file__), config_filename)
        self.cf = configparser.ConfigParser()
        # 读取中文配置文件要加 encoding="utf-8" 或 encoding="utf-8-sig"
        self.cf.read(file_path,encoding="utf-8")

    def get_sections(self):
        return self.cf.sections()

    def get_options(self, section):
        return self.cf.options(section)

    def get_content(self, section):
        result = {}
        for option in self.get_options(section):
            value = self.cf.get(section, option)
            result[option] = int(value) if value.isdigit() else value
        return result


class BasePymysqlPool(object):
    def __init__(self, host, port, user, password, db_name=None):
        self.db_host = host
        self.db_port = int(port)
        self.user = user
        self.password = str(password)
        self.db = db_name
        self.conn = None
        self.cursor = None


class MyPymysqlPool(BasePymysqlPool):
```

```
    """
    MYSQL数据库对象，负责产生数据库连接，此类中的连接采用连接池实现获取连接对象：conn = ysql.getConn()
            释放连接对象;conn.close()或del conn
    """
    # 连接池对象
    __pool = None

    def __init__(self, conf_name=None):
        self.conf = Config().get_content(conf_name)
        super(MyPymysqlPool, self).__init__(**self.conf)
        # 数据库构造函数，从连接池中取出连接，并生成操作游标
        self._conn = self.__getConn()
        self._cursor = self._conn.cursor()

    def __getConn(self):
        """
        @summary: 静态方法，从连接池中取出连接
        @return MySQLdb.connection
        """
        if MyPymysqlPool.__pool is None:
            __pool = PooledDB(creator=pymysql,
                              mincached=1,
                              maxcached=20,
                              host=self.db_host,
                              port=self.db_port,
                              user=self.user,
                              passwd=self.password,
                              db=self.db,
                              use_unicode=True,
                              charset="utf8",
                              cursorclass=DictCursor)
        return __pool.connection()

    def getTabCount(self, sql, param=None):
        """
        @summary: 获取表中相关条件下的数据条数
        @param sql:查询 S Q L，如果有查询条件，请只指定条件列表，并将条件值使用参数[param]递进来
        @param param: 可选参数，条件列表值（元组/列表）
        @return: result list(字典对象)/boolean 查询到的结果集
        """
        if param is None:
            self._cursor.execute(sql)
        else:
            self._cursor.execute(sql, param)
        count = self._cursor.fetchone()
        return count
    def getAll(self, sql, param=None):
        """
        @summary: 执行查询，并取出所有结果集
        @param sql:查询 S Q L，如果有查询条件，请只指定条件列表，并将条件值使用参数[param]递进来
        @param param: 可选参数，条件列表值（元组/列表）
```

```
        @return: result list(字典对象)/boolean 查询到的结果集
        """
        if param is None:
            count = self._cursor.execute(sql)
        else:
            count = self._cursor.execute(sql, param)
        if count > 0:
            result = self._cursor.fetchall()
        else:
            result = False
        return result

    def getOne(self, sql, param=None):
        """
        @summary: 执行查询，并取出第一条
        @param sql:查询 S Q L，如果有查询条件，请只指定条件列表，并将条件值使用参数[param]
递进来
        @param param: 可选参数，条件列表值（元组/列表）
        @return: result list/boolean 查询到的结果集
        """
        if param is None:
            count = self._cursor.execute(sql)
        else:
            count = self._cursor.execute(sql, param)
        if count > 0:
            result = self._cursor.fetchone()
        else:
            result = False
        return result

    def getMany(self, sql, num, param=None):
        """
        @summary: 执行查询，并取出num条结果
        @param sql:查询 S Q L，如果有查询条件，请只指定条件列表，并将条件值使用参数[param]
递进来
        @param num:取得的结果条数
        @param param: 可选参数，条件列表值（元组/列表）
        @return: result list/boolean 查询到的结果集
        """
        if param is None:
            count = self._cursor.execute(sql)
        else:
            count = self._cursor.execute(sql, param)
        if count > 0:
            result = self._cursor.fetchmany(num)
        else:
            result = False
        return result
    def insertOneGetId(self,sql, param=None):
        """
        @summary: 向数据表插入一条记录，并返回该记录的ID
        @param sql:要插入的 S Q L 格式，使用(%s,%s)
        @param values:要插入的记录数据值 tuple/list
        @return: newId 新插入记录的ID
```

```
    """
    self.__query(sql, param)
    # 获取最新自增ID 获取最新自增ID 获取最新自增ID
    newId = self._cursor.lastrowid
    return newId

def insertMany(self, sql, values):
    """
    @summary: 向数据表插入多条记录
    @param sql:要插入的 S Q L 格式
    @param values:要插入的记录数据tuple(tuple)/list[list]
    @return: count 受影响的行数
    """
    count = self._cursor.executemany(sql, values)
    return count

def __query(self, sql, param=None):
    if param is None:
        count = self._cursor.execute(sql)
    else:
        count = self._cursor.execute(sql, param)
    return count

def update(self, sql, param=None):
    """
    @summary: 更新数据表记录
    @param sql:  S Q L 格式及条件，使用(%s,%s)
    @param param: 要更新的  值 tuple/list
    @return: count 受影响的行数
    """
    return self.__query(sql, param)

def insert(self, sql, param=None):
    """
    @summary: 更新数据表记录
    @param sql:  S Q L 格式及条件，使用(%s,%s)
    @param param: 要更新的  值 tuple/list
    @return: count 受影响的行数
    """
    return self.__query(sql, param)

def delete(self, sql, param=None):
    """
    @summary: 删除数据表记录
    @param sql:  S Q L 格式及条件，使用(%s,%s)
    @param param: 要删除的条件 值 tuple/list
    @return: count 受影响的行数
    """
    return self.__query(sql, param)

def begin(self):
    """
    @summary: 开启事务
    """
```

```
        self._conn.autocommit(0)

    def end(self, option='commit'):
        """
        @summary: 结束事务
        """
        if option == 'commit':
            self._conn.commit()
        else:
            self._conn.rollback()

    def dispose(self, isEnd=1):
        """
        @summary: 释放连接池资源
        """
        if isEnd == 1:
            self.end('commit')
        else:
            self.end('rollback')
        self._cursor.close()
        self._conn.close()


if __name__ == '__main__':
    import datetime
    def getNowTime():
        # 格式化字符串
        now_time = datetime.datetime.now()
        now_time_str = datetime.datetime.strftime(now_time, '%Y-%m-%d %H:%M:%S')
        # now_time.strftime('%Y-%m-%d %H:%M:%S')
        return now_time_str

    print(getNowTime())

    mysql = MyPymysqlPool("notdbMysql")

    # sql = "select * from article where title = %s ORDER BY CreateTime DESC"
    # result9 = mysql.getOne(sql, '我们')
    # print(result9['id'])



    # sqlinMany = "insert into mgroup (groupname,Status,CreateUser,Createtime) values (%s,
s,%s,%s)"
    # # vals =[[1,'http://www.songbin.top/1.jpg',getNowTime()],[2,'http://www.songbin.top/2.j
g',getNowTime()]]
    # # vals =[(1,'http://www.songbin.top/3.jpg',getNowTime()),(2,'http://www.songbin.top/3.j
g',getNowTime())]
    # vals = ('美女图片', 1, 'root',getNowTime())
    # result4 = mysql.insertOneGetId(sqlinMany, vals)
    # print(result4)

    # 执行查询，并取出所有结果集
```

```
    # sqlAll = "SELECT * FROM article WHERE info like  '%%%s%%'" % '我'
    # result = mysql.getAll(sqlAll)
    # print(result)
    # 执行查询，并取出第一条
    # sqlOne = "select * from article"
    # result2 = mysql.getAll(sqlOne)
    # print(result2)

    # 执行查询，并取出num条结果
    # sqlMany = "select * from article"
    # result3 = mysql.getMany(sqlMany, 1)
    # print(result3)

    # 向数据表插入多条记录

    # sqlinMany = "insert into imglist (aid,imgurl,createtime) values (%s,%s,%s)"
    # # vals =[[1,'http://www.songbin.top/1.jpg',getNowTime()],[2,'http://www.songbin.top/2.j
g',getNowTime()]]
    # # vals =[(1,'http://www.songbin.top/3.jpg',getNowTime()),(2,'http://www.songbin.top/3.j
g',getNowTime())]
    # vals = ((1, 'http://www.songbin.top/4.jpg', getNowTime()), (2, 'http://www.songbin.top/4.j
g', getNowTime()))
    # result4 = mysql.insertMany(sqlinMany, vals)
    # print(result4)
    sqlC = "select count(*) num from mto_posts"
    num = mysql.getTabCount(sqlC)
    print(num["num"])

    upsql = 'update mto_posts set tgstatus = 1 where id = 3'
    flag = mysql.update(upsql)
    print('行:'+str(flag))
    # 释放资源
    mysql.dispose()
```

3. 数据库配置文件

```
#db_name可以不设置，实现多数据库连接
[MyBlogMysql]
host = 你自己的数据库连接地址
port = 3306
user = root
password = 自己的mysql数据库密码
db_name = 自己的blog数据库名称
```

获取整体项目可访问：IT源点