

课程设计论文

题 目： Python 实现基于 Selenium
的自动健康打卡

学 校： 吉林大学

院 系： 汽车工程学院

作 者： 杨扬

学 号： 15191226

指导老师： 刘通

目录

课程设计论文	1
前言	3
实验源码.....	4
实验原理.....	7
实验过程.....	8
代码分析.....	9
参考文献.....	18

前言

疫情期间，在学校的积极应对政策下，学生必须进行每日健康打卡来汇报自身及家人的健康状况。基于锻炼 Python 应用程序设计能力的目的，在刘通老师课程《Python 语言程序设计》的进程中，产生了利用 Python 的 Selenium 库来实现自动健康打卡的想法，并予以了实现。

并有如下声明：

- 1.健康打卡如果出现任何问题请自负责任，该项目为自选，如果学生在完成项目过程出现因自动填报信息错误而导致的后续问题，本课程及教师不负责任。
- 2.为了积极配合学校防疫，在完成该项目后不再使用该项目打卡，并不再有意推广代码。

实验源码

```
1. # -*- coding:utf-8 -*-
2. from selenium import webdriver
3. import time
4. import requests
5. # ===== 信息 =====
6. username = "yangyang1519" # 账号
7. pwd = "Yang882323" # 密码
8. recv_qq = "2925539854" # 接收反馈的 QQ
9. loading_time = 5 # 预留页面加载时间 5 秒
10. # ===== url =====
11. login_url = "https://ehall.jlu.edu.cn/jlu_portal/login" # 登录 url
12. # ===== xpath =====
13. uid_xpath = '//*[@name="username"]' # 账号 xpath
14. pwd_xpath = '//*[@name="password"]' # 密码 xpath
15. button_xpath = '//*[@name="login_submit"]' # 登录按钮 xpath
16. stu_xpath = '//*[@id="cont_one_1"]/li[4]/a' # 本科生 xpath
17. handle_xpath = '//*[@id="service_guide"]/div/div/input[3]' # 我要办理 xpath
18. checkbox_xpath = '//*[@id="V1_CTRL82"]' # 承诺按钮 xpath
19. confirm_xpath = '//*[@id="form_command_bar"]/li[1]' # 确认按钮 xpath
20. ok_xpath = '//*[@button[1]]' # 好按钮 xpath
21. # ===== 驱动 =====
22. global driver
23. driver = webdriver.Chrome(executable_path="C:\\Program Files (x86)\\Google\\Chrome\\Application\\chromedriver.exe") # 设置 Chrome 驱动
24.
25.
26. def web_init():
27.     driver.maximize_window() # 最大化窗口
28.     driver.implicitly_wait(6) # 隐式等待
29.     driver.get(login_url) # 获取网页
30.
31.
32. def web_close():
33.     driver.quit() # 关闭所有关联的窗口
34.
35.
36. def web_load():
37.     time.sleep(loading_time) # 预留页面加载时间 5 秒
38.
39.
40. def auto_login(username_, pwd_, uid_xpath_, pwd_xpath_, button_xpath_):
41.     web_load()
```

```

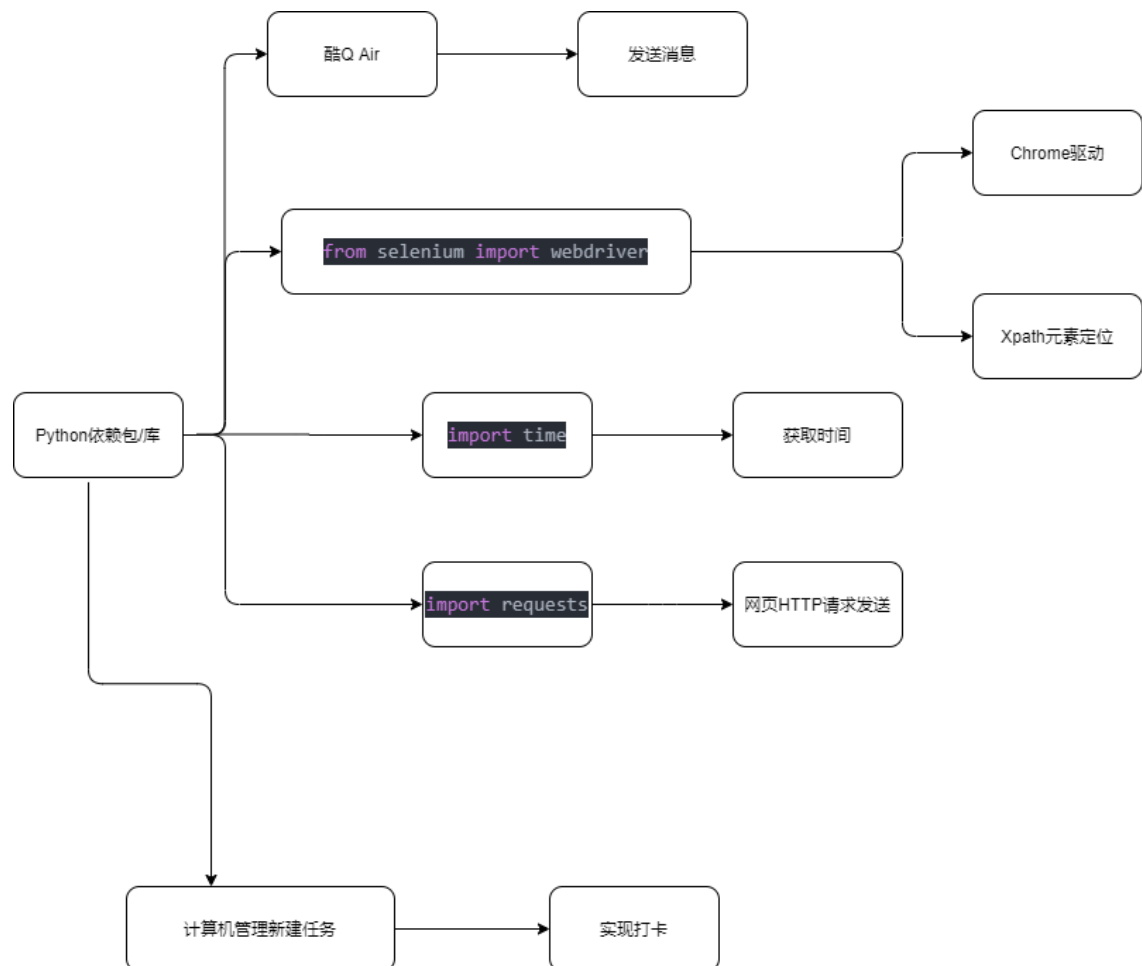
42.     driver.find_element_by_xpath(uid_xpath_).clear() # 清空已存在内容
43.     driver.find_element_by_xpath(uid_xpath_).send_keys(username_) # 用户名
44.     web_load()
45.     driver.find_element_by_xpath(pwd_xpath_).clear() # 清空已存在内容
46.     driver.find_element_by_xpath(pwd_xpath_).send_keys(pwd_) # 密码
47.     web_load()
48.     driver.find_element_by_xpath(button_xpath_).click() # 登录
49. def submit(stu_xpath_, handle_xpath_, checkbox_xpath_, confirm_xpath_, ok_xp
    ath_):
50.     driver.find_element_by_xpath(stu_xpath_).click() # 本科生申报
51.     web_load()
52.     driver.find_element_by_xpath(handle_xpath_).click() # 我要办理
53.     web_load()
54.     driver.switch_to.window(driver.window_handles[1]) # 切换至最新窗口
55.     web_load()
56.     driver.find_element_by_xpath(checkboxbox_xpath_).click() # 本人承诺
57.     web_load()
58.     driver.find_element_by_xpath(confirm_xpath_).click() # 确认填报
59.     web_load()
60.     time.sleep(4)
61.     driver.find_element_by_xpath(ok_xpath_).click() # 好
62.     web_load()
63.
64.
65. def remind( recv_qq_):
66.     today = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime(time.time()))
67.     result_msg = today+'杨扬打卡完成，本人和亲属未有计划 48 小时内从境外返回长春
    '
68.     print(result_msg)
69.     send_private_msg(recv_qq_, result_msg) # 发送一条打卡记录
70. def send_private_msg(qq, msg):
71.     func = "send_private_msg"
72.     qq_url = "http://localhost:5700/" + func
73.     params = {
74.         'user_id': qq,
75.         'message': msg,
76.     }
77.     req = requests.get(qq_url, params) # 调用酷 Q 发送消息
78.     print(req.status_code)
79.
80.
81. if __name__ == '__main__':

```

```
82.     web_init()
      # 驱动初始化
83.     auto_login(username, pwd, uid_xpath, pwd_xpath, button_xpath)
      # 登录操作
84.     submit(stu_xpath, handle_xpath, checkbox_xpath, confirm_xpath, ok_xpath)
      # 提交操作
85.     remind(recv_qq)
      # 发送提醒信息
86.     web_close()
      # 测试结束
```

实验原理

利用 Selenium 库的 webdriver 包，基于网页元素对于网页源码中 Xpath 的依赖，实现对 Chrome 浏览器的自动控制。利用自动软件酷 Q 实现腾讯 QQ 自动发送消息给相关联系人。并通过电脑的计算机管理新建定时任务实现自动操作。流程图如下（通过 drawio 绘制）：



实验过程

1. 将 chrome 驱动存放至 chrome 根目录下 Application 文件夹
2. 新增系统环境变量
3. 解压酷 Q 打开 CQA.exe,登录, 打开应用管理, 在 GitHub 下载 CQHTTP
4. 安装 selenium 和 requests[使用 pip 命令安装]
5. 查找相关网页的 url 以及定位按钮的 Xpath
6. 编写代码
7. 进行调试与完善
8. 新建自动任务, 并测试

代码分析

(一) 代码架构

采用面向过程的设计思维，将功能的实现划分为多个函数：

1. `web_init()`
2. `auto_login(username, pwd, uid_xpath, pwd_xpath, button_xpath)`
3. `submit(stu_xpath, handle_xpath, checkbox_xpath, confirm_xpath, ok_xpath)`
4. `remind(remind_url, recv_qq)`
5. `web_close()`

从而实现了网页的初始化，自动登录，提交打卡信息，发送打卡成功消息，关闭网页。利用 Selenium 库，只需要确定相关按钮的 *Xpath*，便能实现自动控制流程。

(二) 代码分析

如下代码，调用了相关的库以及包：

```
1. from selenium import webdriver
2. import time
3. import requests
```

如下代码，定义了登陆页面填充的相关信息（密码已做处理）：

```
1. username = "yangyang1519"
2. pwd = "*****"
3. recv_qq = "2925539854"
4. loading_time = 5
```

如下代码，提供了所需要的页面 url 以及，自动控制下需要点击的按钮 xpath：

```
1. login_url = "https://ehall.jlu.edu.cn/jlu_portal/login" # 登录 url
2. uid_xpath = '//*[@name="username"]' # 账号 xpath
3. pwd_xpath = '//*[@name="password"]' # 密码 xpath
4. button_xpath = '//*[@name="login_submit"]' # 登录按钮 xpath
5. stu_xpath = '//*[@id="cont_one_1"]/li[4]/a' # 本科生打卡按钮
6. handle_xpath = '//*[@id="service_guide"]/div/div/input[3]' # 我要办理按钮
7. checkbox_xpath = '//*[@id="V1_CTRL82"]' # 承诺按钮 xpath
8. confirm_xpath = '//*[@id="form_command_bar"]/li[1]' # 确认按钮 xpath
9. ok_xpath = '//*[@button[1]]' # 好按钮 xpath
```

此处需要再浏览器中，利用 F12 查看网页源码，并定位到能实现点击的元素。上述多个 Xpath 中，唯一不同的是最后一个按钮，源码中复制的 Xpath 所定位的这个元素是动态变化的，每一次打开该页面到此处，该元素的 Xpath 会自动更新，因而导致无法定位。故采取相对定位法，此按钮的特性在于 button[1]，故通过该 Xpath 便可以定位。

如下代码，通过浏览器驱动来实现浏览器的打开：

```
1. global driver
2. driver=webdriver.Chrome(executable_path="C:\\ProgramFiles(x86)\\Google\\Chrome\\
  Application\\chromedriver.exe")
```

此处，已经将该路径添加到系统的环境变量中。

如下代码实现了网页初始化功能：

```
1. def web_init():
2.     driver.maximize_window()           # 最大化窗口
3.     driver.implicitly_wait(6)          # 隐式等待
4.     driver.get(login_url)              # 获取网页
```

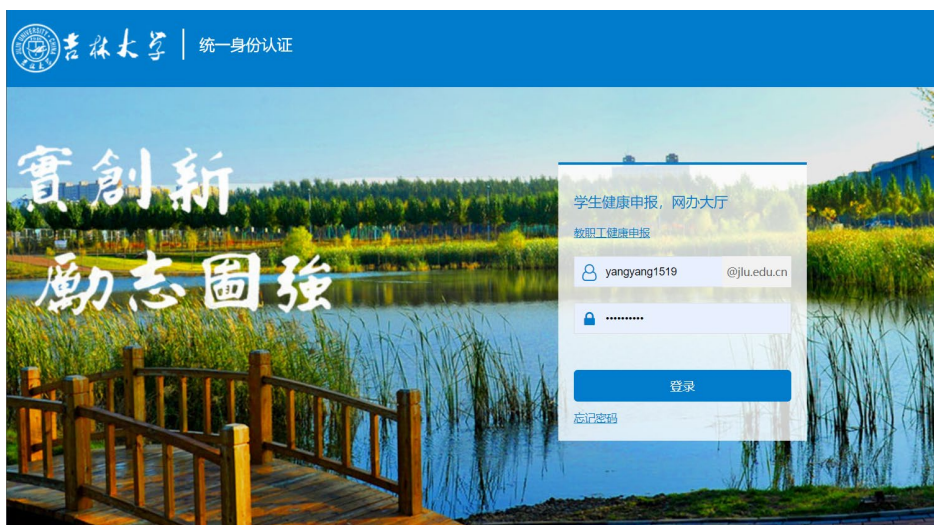
在打卡过程完成后，必然需要关闭相关页面，因而需要设计网页关闭功能：

```
1. def web_close():           # 关闭所有关联的窗口
2.     driver.quit()          # 获取网页
```

显然由于网络的原因，许多时候网页的加载速度比较缓慢，如果未进行加载，即进行相关的操作，则会产生问题，因此，预留一个显式的网页加载等待函数如下（加载时间已经设定）：

```
1. def web_load():
2.     time.sleep(loading_time)           # 预留页面加载时间 5 秒
```

在网页初始化后，便打开了登录界面，如下：



因而此处需要设计一个登录函数，来实现用户账号及密码的填充，函数如下：

```
1. def auto_login(username_, pwd_, uid_xpath_, pwd_xpath_, button_xpath_):
2.     web_load()
3.     driver.find_element_by_xpath(uid_xpath_).clear()      # 清空已存在内容
4.     driver.find_element_by_xpath(uid_xpath_).send_keys(username_) # 用户名
5.     web_load()
6.     driver.find_element_by_xpath(pwd_xpath_).clear()      # 清空已存在内容
7.     driver.find_element_by_xpath(pwd_xpath_).send_keys(pwd_)   # 密码
8.     web_load()
9.     driver.find_element_by_xpath(button_xpath_).click() # 通过登录按钮登录
```

考虑到浏览器的缓存，因而可能在账号密码的填充处已经有相关的信息，因而清空这两栏信息的功能显得十分合适。利用 `send_keys()` 函数，可以轻松的做到填充。最后一行代码，使用了该项目大幅度用到的函数 `find_element_by_xpath()`，即利用 Xpath 查找网页元素，并通过 `click()` 函数，实现了鼠标点击模拟。这种用法，在后续的函数中也经常出现。

通过模拟可知，此时进入了如下界面：



根据登陆界面的思想，利用 Xpath 定位元素，不难得出如下代码：

```

1. def submit(stu_xpath_, handle_xpath_, checkbox_xpath_, confirm_xpath_, ok_xpath_):
2.     driver.find_element_by_xpath(stu_xpath_).click() # 本科生健康状况申报
3.     web_load()
4.     driver.find_element_by_xpath(handle_xpath_).click() # 我要办理
5.     web_load()
6.     driver.switch_to.window(driver.window_handles[1]) # 切换至最新窗口
7.     web_load()
8.     driver.find_element_by_xpath(checkboxbox_xpath_).click() # 本人承诺
9.     web_load()
10.    driver.find_element_by_xpath(confirm_xpath_).click() # 确认填报
11.    web_load()
12.    driver.find_element_by_xpath(ok_xpath_).click() # 好
13.    web_load()

```

至此，打卡环节已经完成。下面需要发送一条打卡完成消息到相关的 QQ 用户：

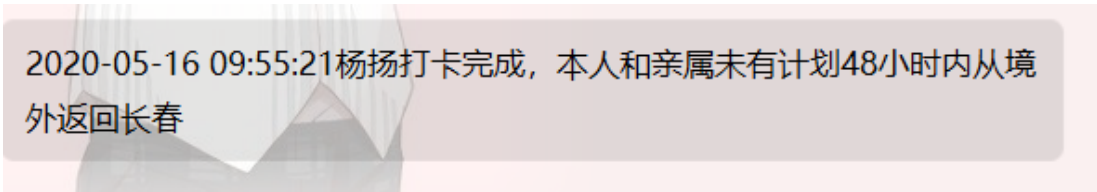
可以写出如下代码：

```

1. def remind( recv_qq_):
2.     today = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime(time.time()))
3.     result_msg = today+'杨扬打卡完成，本人和亲属未有计划 48 小时内从境外返回长春'
4.     print(result_msg)
5.     send_private_msg(recv_qq_, result_msg) # 发送一条打卡记录
6. def send_private_msg(qq, msg):
7.     func = "send_private_msg"
8.     qq_url = "http://localhost:5700/" + func
9.     params = {
10.         'user_id': qq,
11.         'message': msg,
12.     }
13.     req = requests.get(qq_url, params) # 调用酷 Q 发送消息
14.     print(req.status_code)

```

实现结果如下：

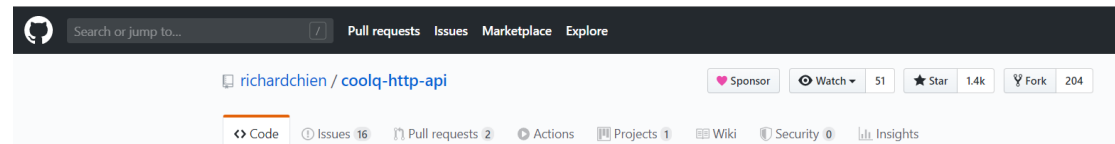


2020-05-16 09:55:21杨扬打卡完成，本人和亲属未有计划48小时内从境外返回长春

QHTTP 的日志如下：

```
[2020-05-16 09:55:22.683] [I] [HTTP] 已成功处理一个 API 请求: /send_private_msg
```

利用引入的 time 包，可以轻松的获取今天的时间，利用字符串的拼接，便可得到所需要发送的消息。发送消息的函数需要调用软件酷 Q，本质上是一个托管 QQ 的自动机器人。此处需要配置酷 Q 的 QHTTP 应用，具体下载需要访问 GitHub：



QHTTP 的加载需要许多动态链接库(即.dll 文件)，若提示缺少，需按照 VC++2017，因为此电脑已经事先安装 VisualStudio2017，相关的库文件完整，因而不会出现该问题。至此，整个打卡流程化设计已经完成，因而主函数的写法将异常简单：

```
1. if __name__ == '__main__':
2.     web_init()
3.     # 驱动初始化
4.     auto_login(username, pwd, uid_xpath, pwd_xpath, button_xpath)
5.     # 登录操作
6.     submit(stu_xpath, handle_xpath, checkbox_xpath, confirm_xpath, ok_xpath)
7.     # 提交操作
8.     remind(recv_qq)
9.     # 发送提醒信息
10.    web_close()
11.    # 测试结束
```

(三) 调试与异常分析

初次进行调试时,在函数,程序断在了主函数内的submit()函数处,几经尝试,错误日志如下:

```
debug.log
1 [0508/155512.155:ERROR:process_reader_win.cc(123)] NtOpenThread: {无法访问} 过程已请求访问一对象,但未给访问权限。 (0xc0000022)
2 [0508/155512.158:ERROR:exception_snapshot_win.cc(98)] thread ID 16408 not found in process
3 [0508/155512.237:ERROR:process_reader_win.cc(151)] SuspendThread: 拒绝访问。 (0x5)
4 [0508/155512.237:ERROR:process_reader_win.cc(123)] NtOpenThread: {无法访问} 过程已请求访问一对象,但未给访问权限。 (0xc0000022)
5 [0508/155512.237:ERROR:exception_snapshot_win.cc(98)] thread ID 7828 not found in process
```

在多次断点定位调试下,锁定到 bug 的位置在初始定义区,该程序无法通过所提供的 ok_xpath 定位到相关按钮,该处原来的代码如下:

```
ok_xpath = '//*[@id="dialog_container_401963"]/div[2]/button[1]'
```

在,重新调试,并再次获取 Xpath 后代码如下:

```
ok_xpath = '//*[@id="dialog_container_420706"]/div[2]/button[1]'
```

从而找到了问题所在,此界面的 Xpath 并不是静态的,每一次网页的打开,此元素都会获得动态分配的 Xpath,因而直接从源码复制过来的 Xpath 是无法定位元素的。在查找资料后发现,并不一定要使用 Xpath 的完整绝对路径或者相对路径,只要相关标识能唯一定位元素即可,因而,将代码改写为:

```
ok_xpath = './button[1]'
```

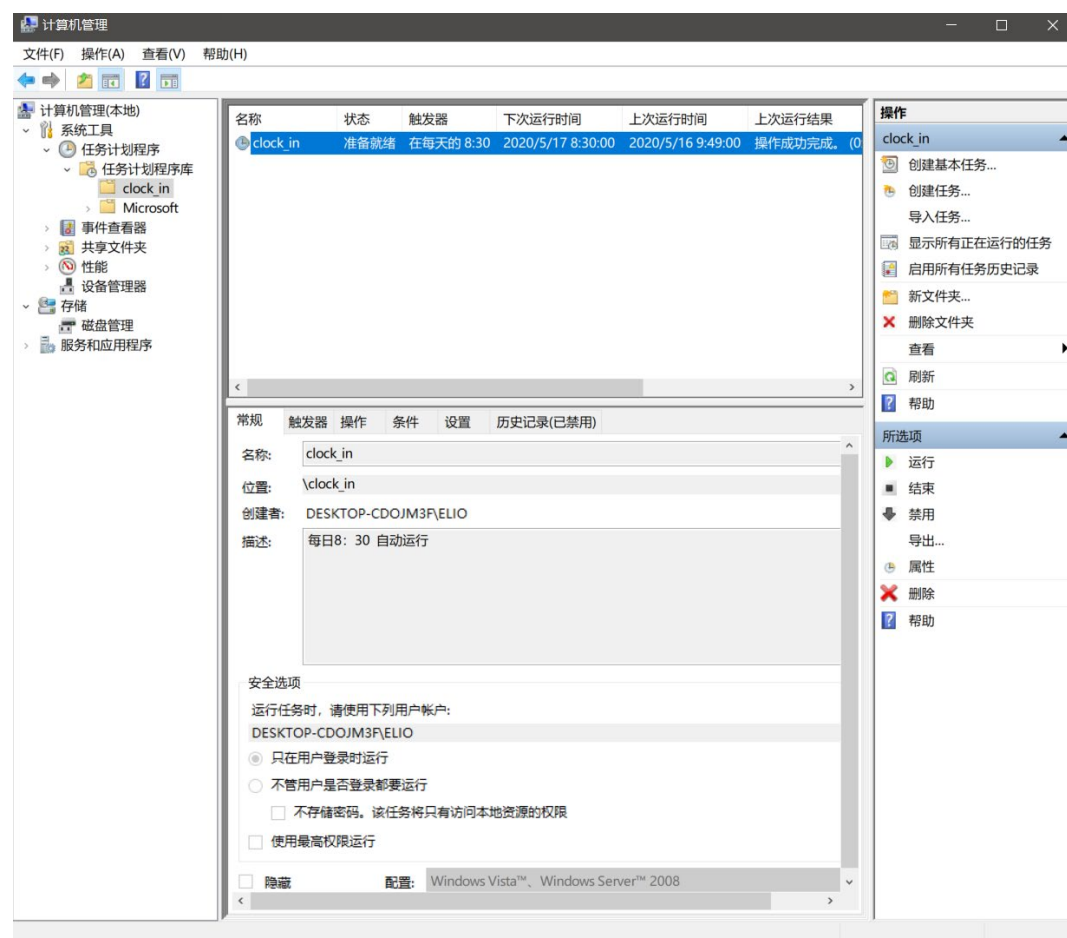
再次调试,无异常抛出,程序成功运行,并且发送了相关打卡信息。

(四) 实现计算机自动完成任务

如果每次仍然需要打开源文件运行才能实现该功能，并未实现真正意义上的自动打卡。因而，通过计算机的任务管理，可以新建一个定时任务，自动触发该源文件。实现过程如下：

1. win+R 键入 compmgmt.msc 打开计算机管理
2. 任务计划程序库新建定时打卡的任务
3. 通过源文件的绝对路径，定位源文件
4. 设置完成

最终的结果截图如下：



(五) 优化策略

1. 该程序部署在个人客户机上，局限性较大，可以通过远端的服务器进行远端部署，实现全过程自动完成。
2. 代码只考虑了个人情况，可以改进成为交互模式，交互地填充信息，保存后进行自动打卡过程。

参考文献

[1]zoly.python selenium 实现简单网页测试流程 - 健康打卡 [EB/OL].
https://blog.csdn.net/qg_41854291/article/details/105469603,2020-4-12.

附录

1. 本论文源代码即论文 word 与 pdf 格式所在 GitHub 仓库链接: <https://github.com/Elio-yang/Python->