Hawk – Data Scraper

Hawk is data extracting tool designed in python and built as a standalone executable for windows. It doesn't require any external dependencies to run. The tool has been designed to ease the process of extracting data from the dynamic websites (website contains large amount of structured data). It uses inbuilt excel library that can read input, regex library to perform extraction and manipulate excel columns.

Hawk works based on an "ini" configuration file to read and process website extraction. A hawk configuration file is normal text file that contains 5 sections namely, **Main, Login, Get, Generate,** and **Put**.

A. Main

Main section contains 8 different configurations.

1. **keys**
   This configuration is used to get key information from the excel file. If you want data from page numbers 1 to 10, put 1 to 10 in an excel file and select that file as excel input file in hawk. If name of the request variable is "page", the configuration is, keys = {'page': 0}, here 0 is column of the key in excel file.
   Ex: **keys = { 'page':1}**

2. **start_row**
   Set the starting row of the input excel to be processed.
   Ex: **start_row = 0** (Excel rows and columns start with 0 (not 1))

3. **url**
   Url to be processed
   Ex: **url = http://hcad.org/Records/SelectRecord.asp**

4. **request_type**
   Url request type. options: get/post
   Ex: **request_type = post**

5. **request_vars**
   Other request variables needed. Leave it empty if nothing needed
   Ex: **request_vars = {'tab':'1', 'searchtype':'strap', 'taxyear':'2014'}**

6. **referer**
   Referer for url request. Leave empty if nothing needed
   Ex: **referer = http://hcad.org/includes/s_acct.asp**

7. **require_login**
   Whether logging in is needed. If login is needed, login module must be specified in login section below. Options: yes/no
   Ex: **require_login = yes**

8. **sleep**
   Wait time in milli seconds (1000 per second) for each request.
   Ex: **sleep = 0**

B. Login

Login section contains 2 configurations, module and login.

1. **module**
   Python module name to find login script that return login cookie information.
   Ex: **module = hcad** (if hcad.py has login information)

2. **function**
   Function name that return cookies in python dictionary format.
   Ex: **function = hcad.login()**

C. Get

Section contains capture configuration in regex format assigned to names.

Ex: **address = '.*?Property Address:.*?<th valign="top" align="left">(.*?)</th>'**

D. Generate

Contains generated variables including functions from captured names in "Get" section.

Ex:

address_splits = address.strip().title().split('<Br />', 1) if address_groups else False
address = address_splits[0] if address_splits else ''

You can even specify groups when multiple groups is captured in Get section.

Ex: **building_sf = building_sf_groups.group(2).strip() if building_sf_groups else ''**

E. Put

Where to put the generated values in output excel.

Ex:

0 = key
1 = address

hcad.ini

```
[Main]
keys = { 'searchval': 0 }
start_row = 0
url = http://hcad.org/Records/SelectRecord.asp
request_type = post
request_vars  = {'tab':'1', 'searchtype':'strap', 'taxyear':'2014'}
referer = http://hcad.org/includes/s_acct.asp
require_login = yes
sleep = 0

[Login]
module = hcad
function = hcad.login()

[Get]
address = '.*?Property Address:.*?<th valign="top" align="left">(.*?)</th>'
entity = '.*?<td valign="top" noWrap>Owner Name &<br />Mailing Address:</td>.*?-->(.*?)<br />'
building_sf = '.*?<td class="sub_header" noWrap>Land Area</td>.*?<tr align="center"
valign="top">.*?<td class="data">(.*?)</td>.*?<td class="data">(.*?)</td>'
val_land = '.*?<td class="data" align="left">Land</td>.*?<td class="data"
align="right">(.*?)</td>.*?<td class="data" align="left">Land</td>.*?<td class="data"
align="right">(.*?)</td>'
val_imp = '.*?<td class="data" align="left">Improvement</td>.*?<td class="data"
align="right">(.*?)</td>.*?<td class="data" align="left">Improvement</td>.*?<td class="data"
align="right">(.*?)</td>'
val_tot = '.*?<td class="data" align="left">Total</td>.*?<td class="data"
align="right">(.*?)</td>.*?<td class="data" align="left">Total</td>.*?<td class="data"
align="right">(.*?)</td>'


[Generate]
address_splits = address.strip().title().split('<Br />', 1) if address_groups else False
address = address_splits[0] if address_splits else ''
land_sf = building_sf_groups.group(1).strip() if building_sf_groups else ''
building_sf = building_sf_groups.group(2).strip() if building_sf_groups else ''
owner = entity_groups.group(1).strip() if entity_groups else ''
val_land_2013 = val_land_groups.group(1).strip() if val_land_groups else ''
val_land_2014 = val_land_groups.group(2).strip() if val_land_groups else ''
val_imp_2013 = val_imp_groups.group(1).strip() if val_imp_groups else ''
val_imp_2014 = val_imp_groups.group(2).strip() if val_imp_groups else ''
val_tot_2013 = val_tot_groups.group(1).strip() if val_tot_groups else ''
val_tot_2014 = val_tot_groups.group(2).strip() if val_tot_groups else ''
```

```
[Put]
0 = keys_with_value['searchval']
1 = address
2 = land_sf
3 = building_sf
4 = owner
5 = val_land_2013
6 = val_imp_2013
7 = val_tot_2013
8 = val_land_2014
9 = val_imp_2014
10 = val_tot_2014
```