pybotlib

Release 0.0.1

May 29, 2019

Contents

CHAPTER 1

Introduction

`pybotlib` is a python library for developing Robotic Process Automation projects easily with python primarily on Linux. It contains the code for a central object that will maintain the state and behavior of the RPA. The focus of the project is to accelerate RPA development at scale using open source technology. The package is split into two parts the pybotlib.VirtualAgent object that will control the overall behavior of your RPA as well as pybotlib.utils that contain many useful functions for RPA developmet purposes. Check out the API Documentation for more details. The project is centered around open technologies and the believe that the future of digital transformation across business is rooted in a shared knowledge economy.

`pybotlib`'s offical version is only supported on Linux systems however there is a version for Windows 10 that can be cloned and used in github. The main branch of the pybotlib github repo is the Windows branch while the branch that is linked to PyPI is the ubuntu-client-37 branch.

To start using `pybotlib` on any linux machine you must make sure the machine can run GUI apps and has python 3.7+ installed.

We recommend to run light weight virtual desktops within docker containers and develop cloud native "destructible" RPA armies that can be summoned at mass scale.

To learn more about RPA development best practices in general refer to the following Medium article here: https://itnext.io/the-modern-enterprise-business-in-code-c6a5e0f4ed7e.

To install `pybotlib` you can install via pip or setup.py. We recommend to use pip for the latest stable version.

```
pip install pybotlib
```

To install from source:

```
git clone -b ubuntu-client-37 pybotlib && cd pybotlib
python setup.py install
```

As mentioned above, we recomend to use lightweight docker containers to run your RPA's. To spin up a fully functioning Ubuntu with GUI desktop run the following command, then access it via any VNC client or through the browser. More details on this container image can be found here: https://github.com/fcwu/docker-ubuntu-vnc-desktop

```
docker run -p 6080:80 -p 5900:5900 -v /dev/shm:/dev/shm dorowu/ubuntu-desktop-lxde-vnc
```

CHAPTER 2

Getting Started

Open, scalable, distributed processing is the future of computing and RPA's is general. `pybotlib` considers these factors as key to its development philosophy.

To specifically develop RPAs for scale deployments on any infrastructure follow the tutorial provided here: https://github.com/dkatz23238/pybotlib-tutorial

CHAPTER 3

API Documentation

class pybotlib.**VirtualAgent**(*bot_name*, *downloads_directory*, *df=None*, *firefoxProfile=None*)

Core class of pybotlib. Creates an 'RPA' object for business process automation.

RPA objects can be used to create a virtual assistant that will cary out a series of event-based or strictly scheduled tasks.

It is always recommended to provide a firefox path to have cookies and preferences persisted.

Parameters

- **bot_name** (*str*) – Name of the RPA. Used for logging and identification purposes.
- **downloads_directory** (*str*) – Name of the subfolder to which all file downloads from the internet will be downloaded to.
- **df** (*Pandas.DataFrame, optional*) – Used to embed a table within the VirtualAgent object and store call it via VirtualAgent.df
- **firefoxProfile** (*str, optional*) – Specific name of the Firefox profile settings subfolder to use when using the geckodriver. Usefull to retain cookies and other web based data. It is also very usefull to store specific accepted settings such as MIME types of resources you don't want to be prompted to download (direct download). By default the VirtualAgent will include most MIME file types to directly download them on the host machine without prompting. Profiles are found under: "~/.mozilla/firefox/profiles.ini".

Returns An instance of VirtualAgent ready to be used and deployed to automate business processes.

Return type *pybotlib.VirtualAgent*

create_log_file()

Creates a log csv under "./pybotlib_logs".

You can log transactional or execution logs once the file has been created.

find_by_tag_and_attr(*tag*, *attribute*, *evaluation_string*, *sleep_secs*, *return_many=True*)

Returns an Selenium webelement object.

Usefull function to scan a web site for elements that satisfy specific conditions. This function is accelerated with javascript. For example: `my_bot.find_by_tag_and_attr("a","class", "special_class",0.2)`

Parameters

- **tag** (*str*) – HTML tag to begin search for. If the element we seek is an <input> we would pass the argument "input".
- **attribute** (*str*) – Which attribute of the HTML element do we evaluate in order to interact with a webpage. To name a few: "class", "id", or "placeholder", are all possible examples.
- **evaluation_string** (*str*) – What text should we evaluate when searching the elements on the page. If our attribute is "id" and evaluation string is "001" we will reduce our search the the elements that id == "001".
- **sleep_secs** (*float*) – How many seconds to sleep before executing search. Used to contemplate for slow webpages
- **return_many** (*bool, optional*) – Should the method return a list or an individual element

Returns

Either returns a list of webelement objects or an individual webelement object depending on the return_many argument.

Return type list or selenium.webdriver.remote.webelement

get(*url*)
Access a website via the browser.

Given a URL the VirtualAgent will navigate to this website via the webdriver. The webdriver must be instantited and running by using the initialize_driver method.

Raises `Exception` – If no driver is initialized.

initialize_driver()
Instantiates a geckodriver firefox instance.

This method is used to initialize a webdriver instance to automate browser based tasks.

Raises `FileNotFoundError` – If the geckodriver is not in CWD. You can use pybotlib.utils.get_geckodriver() to download and place it in the CWD.

log(*message*, *tag='transaction'*)
Logs a message to the currently active log file.

Used to log messages to the currently active log file.

Parameters

- **message** (*str*) – Message to be logged.
- **tag** (*str, optional*) – Tag associated to message. Defaults to "transactional".

log_bot_completion()
Log completion of RPA.

Logs that the RPA has successfully completed. To be used at the very end of the RPA.

quit_driver()
Quits out of the web driver.

set_DataFrame(*df*)
Setter for a pandas.DataFrame. :param df: Embed a dataframe within the RPA after instantiation :type df: pandas.DataFrame

use_javascript(*script*)
Executes javascript code into the current running webpage. :raises: `Exception` – If no driver is initialized.

pybotlib.**generate_js**(*tag*, *atr*, *evalString*)
Generates js string for web element searching.

Parameters

- **tag** (*str*) – HTML tag to search for.
- **atr** (*str*) – HTML attribute for which to evaluate when searching the DOM.
- **evalString** (*str*) – Used to determine if attribute of HTML is element is equal to this string.

Returns Javascript code to loop through HTML elements and find a subset satisfying a specific evaluated condition.

Return type str

CHAPTER 4

pybotlib.utils

pybotlib.utils.**create_minio_bucket**(*host_uri*, *minio_access_key*, *minio_secret_key*, *bucket_name*)

Creates a minio bucket.

pybotlib.utils.**dt_parse**(*t*)

Parses out datetime from email msg format.

Parameters **t** (*str*) – t is a string containing an RFC 2822 date, such as "Mon, 20 Nov 1995 19:12:08 -0500".

Returns A datetime.datetime object.

Return type datetime.datetime

pybotlib.utils.**get_geckodriver**()

Fetches latest version of geckodriver to automate firefox via the Selenium webdriver.

This function uses GNU wget. Make sure it is installed on the system before calling get_geckodriver()

pybotlib.utils.**pandas_read_google_sheets**(*sheet_id*)

Returns a pandas DataFrame from a spreadsheet in google sheets. Make sure the spreadsheet has a "view" link and only contains one tab of data.

Parameters **sheet_id** (*str*) – Individual googlesheet ID extracted from URL with view access.

Returns A dataframe with the data from the googlesheets.

Return type pandas.DataFrame

pybotlib.utils.**return_emails_from_IMAP**(*email_account*, *password*, *email_folder*, *search_term='ALL'*, *url='imap.gmail.com'*)

Returns a list of mailparser.MailParser objects from an email address using IMAP.

Used to search a specific IMAP email folder and return a list of individual mailparser.MailParser objects. Will return no values if the login, folder, or search fails. You can replace search_term with other fields such as "UnSeen" or "Seen".

Parameters

- **email_account** (*str*) – Email address to read inbox from in string format.

- **password** (*str*) – Password for associated email_account.
- **email_folder** (*str*) – Which IMAP folder to return emails from.
- **search_term** (*str*) – Term that is used to search in email folder. Defaults to all.
- **url** (*str*) – IMAP server url. Defaults to imap.gmail.com for use with google gmail accounts.

Returns A list of mailparser.MailParser objects retrieved from the email server.

Return type list

pybotlib.utils.**save_csv_from_googlesheets**(*service_file*, *sheet_url*, *filename*)

Save a google sheets table to .csv with name filename.

Needs JSON credentials file location, full google sheets URL and filename to be saved. Google Sheet must be shared with view access to service account listed in JSON credentials file under 'client_email'.

Parameters

- **service_file** (*str*) – Path to the google credentials json service file.
- **sheet_url** (*str*) – URL of the googlesheet to be downloaded.
- **filename** (*str*) – Name of the csv file to be saved without the '.csv'.

Returns True. The csv was saved.

Raises Exception – If the function does not return True.

pybotlib.utils.**save_emails_to_CWD**(*list_of_mails*)

Saves a list of mailparser.MailParser objects to CWD.

Takes as input a list of mailparser.MailParser objects and saves the emails to current working directory under a folder called pybotlib_emails. Headers and body are saved as individual txt files inside a folder named after the subject and date recieved. Attachments are also saved into said folder.

Parameters **list_of_mails** (*list*) – A list of mailparser.MailParser objects.

Returns None.

pybotlib.utils.**send_HTML_email_with_attachement**(*subject*, *body*, *sender_email*, *receiver_email*, *password*, *filename*, *watermark='pybotlib RPA'*)

Sends an visually pleasing HTML email with one attachment from a gmail account.

Parameters

- **subject** (*str*) – Subject of the email.
- **body** (*str*) – Body of the email.
- **sender_email** (*str*) – From field in the email.
- **reciever_email** (*str*) – The recipient emaill address.
- **password** (*str*) – Password of senders email.
- **filename** (*str*) – Absoloute path of file to send in email or the file name if the file is in CWD.

Returns None.

pybotlib.utils.**send_email_with_attachement**(*subject*, *body*, *sender_email*, *receiver_email*, *password*, *filename*)

Sends a simple with one attachment from a gmail account.

Parameters

- **subject** (*str*) – Subject of the email.
- **body** (*str*) – Body of the email.
- **sender_email** (*str*) – From field in the email.
- **reciever_email** (*str*) – The recipient emaill address.
- **password** (*str*) – Password of senders email.
- **filename** (*str*) – Absoloute path of file to send in email or the file name if the file is in CWD.

Returns None.

pybotlib.utils.**write_file_to_minio_bucket**(*host_uri*, *minio_access_key*, *minio_secret_key*, *bucket_name*, *filename*)

Write a file to a minio bucket.

Parameters

- **host_uri** (*str*) – URI to minio instance.
- **minio_access_key** (*str*) – Access key.
- **minio_secret_key** (*str*) – Secret key.
- **bucket_name** (*str*) – Bucket name to save.
- **filename** (*str*) – Name of file to save to bucket. Must be in CWD.

Returns None.

Raises Exception – If bucket does not exist or the file is not found.

CHAPTER 5

pybotlib.exceptions

exception pybotlib.exceptions.**Error**
Base class for other exceptions

exception pybotlib.exceptions.**NoElementsSatisfyConditions**
Raised when find_by_tag_and_attr results in an empty list

CHAPTER 6

Indices and tables

- genindex
- search

Python Module Index

p

Index