pyrobot Documentation

Release 0.1

Joshua Carp

February 02, 2014

Contents:

CHAPTER 1

PyRobot: Your friendly neighborhood web scraper

Browse the web from the comfort of your Python terminal.

```
import re
from pyrobot import RoboBrowser

# Browse to Rap Genius
browser = RoboBrowser(history=True)
browser.open('http://rapgenius.com/')

# Search for Queen
form = browser.get_form(action=re.compile(r'search'))
form['q'].value = 'queen'
browser.submit_form(form)

# Look up the first song
songs = browser.select('.song_name')
browser.follow_link(songs[0])
lyrics = browser.find(class_=re.compile(r'\blyrics\b'))
lyrics.text     # '\n[Intro]\nIs this the real life...

# Back to results page
browser.back()

# Look up my favorite song
browser.follow_link(text=re.compile(r'death on two legs', re.I))
lyrics = browser.find(class_=re.compile(r'\blyrics\b'))
lyrics.text     # '\n[Verse 1]\nYou suck my blood like a leech...
```

CHAPTER 2

Installation

At the command line:

```
$ easy_install pyrobot
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv pyrobot
$ pip install pyrobot
```

CHAPTER 3

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

3.1 Types of Contributions

3.1.1 Report Bugs

Report bugs at https://github.com/jmcarp/pyrobot/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

3.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

3.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

3.1.4 Write Documentation

pyrobot could always use more documentation, whether as part of the official pyrobot docs, in docstrings, or even on the web in blog posts, articles, and such.

3.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/jmcarp/pyrobot/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

3.2 Get Started!

Ready to contribute? Here's how to set up *pyrobot* for local development.

1. Fork the *pyrobot* repo on GitHub.
2. Clone your fork locally:

   ```
   $ git clone git@github.com:your_name_here/pyrobot.git
   ```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

   ```
   $ mkvirtualenv pyrobot
   $ cd pyrobot/
   $ python setup.py develop
   ```

4. Create a branch for local development:

   ```
   $ git checkout -b name-of-your-bugfix-or-feature
   ```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

   ```
   $ flake8 pyrobot tests
   $ python setup.py test
   $ tox
   ```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

   ```
   $ git add .
   $ git commit -m "Your detailed description of your changes."
   $ git push origin name-of-your-bugfix-or-feature
   ```

7. Submit a pull request through the GitHub website.

3.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/jmcarp/pyrobot/pull_requests and make sure that the tests pass for all supported Python versions.

3.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_pyrobot
```

CHAPTER 4

Credits

4.1 Development Lead

- Joshua Carp <jm.carp@gmail.com>

4.2 Contributors

None yet. Why not be the first?

CHAPTER 5

History

5.1 0.1.0

- First release on PyPI.

CHAPTER 6

API Reference

6.1 browser

Robotic browser

class pyrobot.browser.**RoboBrowser**(*auth=None*, *parser=None*, *headers=None*, *user_agent=None*, *history=True*)

Robotic web browser.

back(*n=1*)

Go back in browser history.

Parameters **n** (*int*) – Number of pages to go back

follow_link(*value=None*, **args*, ***kwargs*)

Find a click a link by tag, pattern, and/or BeautifulSoup arguments.

Parameters **value** – BeautifulSoup tag, string, or regex. If tag, follow its href; if string or regex, search parsed document for match

forward(*n=1*)

Go forward in browser history.

Parameters **n** (*int*) – Number of pages to go forward

get_form(*id=None*, **args*, ***kwargs*)

Find form by ID, as well as standard BeautifulSoup arguments.

Parameters **id** (*str*) – Form ID

Returns BeautifulSoup tag if found, else None

get_forms(**args*, ***kwargs*)

Find forms by standard BeautifulSoup arguments.

Returns List of BeautifulSoup tags

get_link(*text=None*, **args*, ***kwargs*)

Find an anchor or button by containing text, as well as standard BeautifulSoup arguments.

Parameters **text** – String or regex to be matched in link text

Returns BeautifulSoup tag if found, else None

get_links(*text=None*, **args*, ***kwargs*)

Find anchors or buttons by containing text, as well as standard BeautifulSoup arguments.

Parameters **text** – String or regex to be matched in link text

Returns List of BeautifulSoup tags

open(*url*)
Open a URL.

Parameters **url** (*str*) – URL

submit_form(*form*)
Submit a form.

Parameters **form** (*Form*) – Filled-out form object

class pyrobot.browser.**RoboState**(*browser*, *response*)
Representation of a browser state. Wraps the browser and response, and lazily parses the response content.

parsed
Lazily parse response content, using HTML parser specified by the browser.

6.2 form

HTML forms

class pyrobot.forms.form.**Form**(*parsed*)
Representation of an HTML form.

serialize()
Serialize each form field and collect the results in a dictionary of dictionaries. Different fields may serialize their contents to different sub-dictionaries: most serialize to data, but file inputs serialize to files. Sub-dictionary keys should correspond to parameters of requests.Request.

Return dict Dict-of-dicts of serialized data

6.3 fields

HTML form fields

class pyrobot.forms.fields.**BaseField**(*parsed*)
Abstract base class for form fields.

class pyrobot.forms.fields.**FieldMeta**(*name*, *bases*, *dct*)
Multiply inherit from ValueMeta and ABCMeta; classes with this meta- class are automatically assigned a value property and can use methods from ABCMeta (e.g. abstractmethod).

mro() → list
return a type's method resolution order

register(*subclass*)
Register a virtual subclass of an ABC.

class pyrobot.forms.fields.**ValueMeta**(*name*, *bases*, *dct*)
Meta-class that creates a value property on class creation. Classes with this meta-class should define _get_value and optionally _set_value methods.

mro() → list
return a type's method resolution order

CHAPTER 7

Indices and tables

- *genindex*
- *modindex*
- *search*

p