chrw Documentation

Release 1.0.1

plausibility

December 19, 2016

Contents

It’s a bird; it’s a plane! No, it’s the chrw module!

chrw is a wrapper for the chr url shortener, and follows all API rules, with included rate limiting.

CHAPTER 1

Contents

1.1 APIv1 Wrapper

Here is some info related to the chr wrapper for chr APIv1.

1.1.1 APIv1 Wrapper Usage

It's fairly simple to use the wrapper, so here's how you do it.

If you want a more thorough run down of any of the API calls, visit the chr docs, and read the API section.

Initialising

It's fairly easy to setup a wrapper instance, but you will need some things:

- The URL to the chr instance (in this case we'll use: `chr.so`)
- Your API key (in this case, it's just: `...`)

Here's how you set up your wrapper:

```
>>># Obviously v1 is for APIv1
>>> import chrw.v1

>>> # chr.so is our url, ... is our API key
>>> w = chrw.v1.wrapper("chr.so", "...")
```

Shortening

Shortening URLs can be done with a call to the *`chrw.v1.wrapper.shorten()`* method.

```
>>> reply = w.shorten("http://foo-bar.example.com/baz", custom="foo", give_delete=True)
>>> pprint.pprint(reply)
{u'delete': u'XXXXXXXXXX',
 u'enum': 0,
 u'error': u'false',
 u'given': u'http://foo-bar.example.com/baz',
 u'message': u'Successfully shrunk your URL!',
 u'short': u'+foo',
 u'url': u'http://chr.so/+foo'}
```

Deletion

If you want to delete a URL, you'll need to call *chrw.v1.wrapper.delete()* with the deletion key you got when you made the URL.

```
>>> reply = w.delete("+foo", "XXXXXXXXXX")
>>> pprint.pprint(reply)
{u'enum': 0,
 u'error': u'false',
 u'message': u'Successfully deleted http://chr.so/+foo'}
```

Statistics

Pulling up information about a given URL could not be easier than calling *chrw.v1.wrapper.stats()*.

```
>>> reply = w.stats("+foo")
>>> pprint.pprint(reply)
{u'clicks': {u'browsers': {u'unknown': 0},
             u'pd': {u'01/08': 0,
                     u'01/09': 0,
                     ...
                     u'02/05': 0,
                     u'02/06': 0},
             u'platforms': {u'unknown': 0}},
 u'enum': 0,
 u'error': u'false',
 u'hits': {u'all': 0, u'ratio': u'0', u'return': 0, u'unique': 1},
 u'long': u'http://foo-bar.example.com/baz',
 u'long_clip': u'http://foo-bar.example.com/baz',
 u'message': u'Here are your stats, piping hot from the oven!',
 u'short': u'+foo',
 u'short_url': u'http://chr.so/+foo'}
```

This simply tells us that there hasn't been any clicks yet. Not a single damn one. :(

Expansion

If you have a shortened URL you'd like info on, you can simply call *chrw.v1.wrapper.expand()*, and all of your questions will be answered.

```
>>> reply = w.expand("+foo")
>>> pprint.pprint(reply)
{u'enum': 0,
 u'error': u'false',
 u'long': u'http://foo-bar.example.com/baz',
 u'message': u"Here's the expanded URL",
 u'short': u'+foo'}
```

Exceptions

At some point in time, you're bound to come across some exceptions.

The API is turned off, your API key is nolonger valid, you gave a bad URL; it could be anything!

These are easy to handle with `try: ... except` clauses in Python, and you can find the exception list at *chrw.exceptions*.

A quick example, which involves catching `chrw.exceptions.RequestRateTooHigh`.

```
>>> import chrw.v1
>>> w = chrw.v1.wrapper("chr.so", "...")
>>> try:
...     for x in xrange(0, 10):
...         reply = w.shorten("http://some-damn-website.com/{0}".format(x))
... except chrw.exceptions.RequestRateTooHigh:
...     print "Girl, you requestin' too darn fast!"
...     print "Slow yo' self down!"
...     raise
...
Girl, you requestin' too darn fast!
Slow yo' self down!
Traceback (most recent call last):
  File "<stdin>", line 3, in <module>
  File "/home/chris/github/chrw/chrw/v1.py", line 93, in shorten
    "delete": "true" if give_delete else ""
  File "/home/chris/github/chrw/chrw/v1.py", line 33, in rate_limited_function
    raise chrw.exceptions.RequestRateTooHigh, msg
chrw.exceptions.RequestRateTooHigh: Last request was 0:00:00.000016, should be at least 0:00:01
```

1.2 Automated Documentation

It's a bird; it's a plane! No, it's the chrw module!

chrw is a wrapper for the chr url shortener, and follows all API rules, with included rate limiting.

1.2.1 APIv1

class `chrw.v1.`**`wrapper`**(*url*, *api_key*, *https=False*, *user_agent=None*, *require_200=True*)

A wrapper to the chr url shortening service APIv1.

Obeys the 1r/s rule with rate limiting, so you can't accidentally hammer the API without knowing.

View online @ http://pypi.python.org/pypi/chrw

`__init__`(*url*, *api_key*, *https=False*, *user_agent=None*, *require_200=True*)

Pull together the various settings of the wrapper.

Parameters

- **`url`** (*`str`*) – the chr instance url, minus trailing slash (e.g. chr.so)
- **`api_key`** (*`str`*) – your API key, as written in the database
- **`https`** (*`bool`*) – are we using https to access this instance?
- **`user_agent`** (*`str or None`*) – what user agent will we use to access the API?
- **`require_200`** (*`bool`*) – should we raise an exception on a non-200OK reply?

`delete`(*url*, *code*)

Request a URL be deleted.

This will only work if you supply the valid deletion code.

Parameters

- **`url`** (*`str`*) – the shortened url to delete

- **code** (*str*) – the deletion code given to you on URL shorten

Returns the deletion request's reply dict

Return type dict

expand(*url*)

This will simply expand a shortened url (e.g, `+foo`) to the larger url (e.g, `http://foo.example.com/bar`)

Parameters **url** (*str*) – the shortened URL to expand

Returns the expansion request reply dict

Return type dict

fetch(**args*, ***kwargs*)

This does the bulk of the work for the wrapper.

It will send a POST request, to the API URL, with all required data, as well as the api_key given, and will handle various replies, raising exceptions as required.

Parameters

- **url** (*str*) – the url segment to POST to (unbuilt url, e.g., /submit, /expand)
- **pdata** (*dict*) – a dictionary of data to POST
- **store_to_self** (*bool*) – should we store the reply (if any) to self.reply?

Returns the API reply data

Return type dict

Raises chrw.exceptions.ApiDisabled, chrw.exceptions.InvalidApiKey, chrw.exceptions.PartialFormData, chrw.exceptions.NonZeroException

shorten(*url*, *custom=None*, *give_delete=True*)

Sends a URL shorten request to the API.

Parameters

- **url** (*str*) – the URL to shrink
- **custom** (*str*) – a custom URL to request
- **give_delete** (*bool*) – would we like a deletion key to be returned?

Returns the API response JSON dict

Return type dict

stats(*url*)

Request the stats for a given URL.

Parameters **url** (*str*) – the shortened url to get stats for

Returns the statistics JSON dict for the URL

Return type dict

1.2.2 Exceptions

chrw.exceptions contains all the exceptions relating to the execution of the chr wrapper.

exception `chrw.exceptions.`**ApiDisabled**

The chr API is disabled, so we can't do anything.

exception `chrw.exceptions.`**`ChrwException`**

Something went wrong when trying to work with the chr API.

exception `chrw.exceptions.`**`InvalidApiKey`**

We sent a request with an invalid API key.

exception `chrw.exceptions.`**`InvalidDataReturned`**

The reply we got didn’t appear to be valid JSON data.

exception `chrw.exceptions.`**`NonZeroReply`**

The reply returned a non-zero reply error number.

exception `chrw.exceptions.`**`PartialFormData`**

Something is missing from the form data.

This might suggest an API update, and incompatibilities.

exception `chrw.exceptions.`**`RequestFailed`**

A POST request we sent didn’t complete as 200 OK.

exception `chrw.exceptions.`**`RequestRateTooHigh`**

We’re attempting to poll the API too darn frequently.

exception `chrw.exceptions.`**`TimeIsBackToFront`**

We somehow traveled back in time, and there was no delorean.

Python Module Index