# scrapelib Documentation

***Release 1.0.0***

**Michael Stephens and James Turk**

January 11, 2016

Contents

CHAPTER 1

# Overview

scrapelib is a library for making requests to websites, particularly those that may be less-than-reliable.

scrapelib originated as part of the Open States project to scrape the websites of all 50 state legislatures and as a result was therefore designed with features desirable when dealing with sites that have intermittent errors or require rate-limiting.

As of version 0.7 scrapelib has been retooled to take advantage of the superb requests library.

Advantages of using scrapelib over alternatives like httplib2 simply using requests as-is:

- All of the power of the suberb requests library.
- HTTP(S) and FTP requests via an identical API
- support for simple caching with pluggable cache backends
- request throtting
- configurable retries for non-permanent site failures

CHAPTER 2

# Contents

## 2.1 scrapelib overview

scrapelib is configured by instantiating an instance of a *Scraper* with the desired options and paths.

### 2.1.1 Scraper object

**class** scrapelib.**Scraper**(*raise_errors=True*, *requests_per_minute=60*, *retry_attempts=0*, *retry_wait_seconds=5*, *header_func=None*)

Scraper is the most important class provided by scrapelib (and generally the only one to be instantiated directly). It provides a large number of options allowing for customization.

Usage is generally just creating an instance with the desired options and then using the `urlopen()` & *urlretrieve()* methods of that instance.

**Parameters**

- **raise_errors** – set to True to raise a *HTTPError* on 4xx or 5xx response
- **requests_per_minute** – maximum requests per minute (0 for unlimited, defaults to 60)
- **retry_attempts** – number of times to retry if timeout occurs or page returns a (non-404) error
- **retry_wait_seconds** – number of seconds to retry after first failure, subsequent retries will double this wait

**urlretrieve**(*url*, *filename=None*, *method='GET'*, *body=None*, *dir=None*, ***kwargs*)

Save result of a request to a file, similarly to `urllib.urlretrieve()`.

If an error is encountered may raise any of the scrapelib *exceptions*.

A filename may be provided or *urlretrieve()* will safely create a temporary file. If a directory is provided, a file will be given a random name within the specified directory. Either way, it is the responsibility of the caller to ensure that the temporary file is deleted when it is no longer needed.

**Parameters**

- **url** – URL for request
- **filename** – optional name for file
- **method** – any valid HTTP method, but generally GET or POST

- **body** – optional body for request, to turn parameters into an appropriate string use `urllib.urlencode()`
- **dir** – optional directory to place file in

**Returns filename, response** tuple with filename for saved response (will be same as given filename if one was given, otherwise will be a temp file in the OS temp directory) and a `Response` object that can be used to inspect the response headers.

### 2.1.2 Response objects

### 2.1.3 Exceptions

All scrapelib exceptions are a subclass of `ScrapeError`.

**class** `scrapelib.`**`HTTPMethodUnavailableError`**(*message*, *method*)
Raised when the supplied HTTP method is invalid or not supported by the HTTP backend.

**class** `scrapelib.`**`HTTPError`**(*response*, *body=None*)
Raised when urlopen encounters a 4xx or 5xx error code and the raise_errors option is true.

## 2.2 scrapeshell

Many times, especially during development, it is useful to open an interactive shell to tinker with a page. Often the HTML being returned is slightly out of sync with what is being seen in the browser, and it can be difficult to detect these differences without firing up an interactive python shell and inspecting what the request is returning.

If scrapelib is installed on your path it provides **`scrapeshell`**, an entrypoint that will open an IPython shell. It will present the user with an instance of requests.Response with the contents of the scraped page and if lxml is installed, an `lxml.html.HtmlElement` instance as well.

### 2.2.1 scrapeshell arguments

**`url`**
scrapeshell requires a URL, which will then be retrieved via a `get()` call.

**`--ua`** `user_agent`
Set a custom user agent (useful for seeing if a site is returning different results based on UA).

**`--noredirect`**
Don't follow redirects.

## 2.3 cache overview

Assign a *`MemoryCache`*, *`FileCache`*, or *`SQLiteCache`* to the `cache_storage` property of a *`scrapelib.Scraper`* to cache responses:

```
from scrapelib import Scraper
from scrapelib.cache import FileCache
cache = FileCache('cache-directory')
scraper = Scraper()
scraper.cache_storage = cache
scraper.cache_write_only = False
```

### 2.3.1 MemoryCache object

**class** scrapelib.cache.**MemoryCache**
In memory cache for request responses.

**get**(*key*)
Get cache entry for key, or return None.

**set**(*key*, *response*)
Set cache entry for key with contents of response.

### 2.3.2 FileCache object

**class** scrapelib.cache.**FileCache**(*cache_dir*, *check_last_modified=False*)
File-based cache for request responses.

**Parameters**

- **cache_dir** – directory for storing responses
- **check_last_modified** – set to True to compare last-modified timestamp in cached response with value from HEAD request

**get**(*orig_key*)
Get cache entry for key, or return None.

**set**(*key*, *response*)
Set cache entry for key with contents of response.

### 2.3.3 SQLiteCache object

**class** scrapelib.cache.**SQLiteCache**(*cache_path*, *check_last_modified=False*)
SQLite cache for request responses.

**Parameters**

- **cache_path** – path for SQLite database file
- **check_last_modified** – set to True to compare last-modified timestamp in cached response with value from HEAD request

**clear**()
Remove all records from cache.

**get**(*key*)
Get cache entry for key, or return None.

**set**(*key*, *response*)
Set cache entry for key with contents of response.

## 2.4 Migrating from 0.x to 1.x

Scrapelib 1.x drops some of the deprecated interfaces from early versions of scrapelib that did not depend upon requests.

The most noticable of these is the absence of `urlopen()`. Because scrapelib is implemented as a wrapper around `requests.Session` it is recommended people use `get()`, `post()` and the other request methods.

The return type of `urlopen` was a `ResultStr`, which is no longer returned, instead a standard requests.Response will be returned.

### 2.4.1 Notable Changes to Responses

To migrate code using urlopen from an old-style `scrapelib.ResultStr` to a `requests.Response`:

- It is no longer possible to treat a response like a string. Instead use `resp.text`, `resp.bytes` or `resp.json()` as needed.
- `resp.bytes` is now `resp.content`
- `resp.response` is now just `resp`
- `resp.response.code` is now `resp.status_code`
- `resp.response.requested_url` is no longer available, you can use the idiom `resp.history[0].url if resp.history else resp.url`
- `resp.encoding` is still available as `resp.encoding`
- `resp._scraper` is no longer available (was an internal API and hopefully not relied upon)

If you were already using `get()`, `post()` , and similar the response format has not changed.

### 2.4.2 Examples

#### A GET request

With urlopen:

```
s = Scraper()
resp = s.urlopen('http://example.com/json')
assert resp.response.code == 200
data = json.loads(resp)

# or for a non-JSON response
print(resp)
```

With get:

```
s = Scraper()
resp = s.get('http://example.com/json')
assert resp.status_code == 200
data = resp.json()      # uses requests built-in JSON support

# or for a non-JSON request
print(resp.text)
```

#### A POST request

With urlopen:

```
s = Scraper()
resp = s.urlopen('http://example.com/form', 'POST', {'param': 'abc'})
```

With get:

```
s = Scraper()
resp = s.post('http://example.com/form', {'param': 'abc'})
```

## 2.5 scrapelib changelog

### 2.5.1 1.0.0

**20 March 2015**

- drop deprecated urlopen interface, see http://scrapelib.readthedocs.org/en/latest/migration.html
- documentation thanks to poliquin
- sqlite cache thanks to poliquin
- fix for SSLError retries pointed out by Eric Mill

### 2.5.2 0.10.1

**22 January 2015**

- SQLite cache backend (thanks Chris Poliquin!)
- test and fix for header merging bug

### 2.5.3 0.10.0

**15 July 2014**

- added kwarg to use last modified headers when using caching -thanks to Kaitlin Devine
- fix for chardet bug when opening large PDFs (& other binary files) with urlretrieve - thanks to Katilin Devine
- deprecation of urlopen in favor of Requests's request(), get(), post(), etc.
- removal of robots.txt code
- switch tests to py.test
- addition of wheel for release

### 2.5.4 0.9.1

**28 March 2014**

- support kwargs in request paths -thanks to Drew Vogel
- allow_cookies removal and documentation fix -thanks to Joe Germuska
- add dir param to urlretrieve -thanks to Alison Rowland

### 2.5.5 0.9.0

**22 May 2013**

- replace FTPSession with FTPAdapter
- fixes for latest requests

### 2.5.6 0.8.0

**18 March 2013**

- **requests 1.0 compatibility**
  - removal of requests pass-throughs
  - deprecation of setting parameters via constructor

### 2.5.7 0.7.4

**20 December 2012**

- bugfix for status_code coming from a cache
- bugfix for setting user-agent from headers
- fix requests version at <1.0

### 2.5.8 0.7.3

**21 June 2012**

- fix for combination of FTP and caching
- drop unnecessary ScrapelibSession
- bytes fix for scrapeshell
- use UTF8 if encoding guess fails

### 2.5.9 0.7.2

**9 May 2012**

- bugfix for user-agent check
- bugfix for cached content with r characters
- bugfix for requests >= 0.12
- cache_dir deprecation is total

### 2.5.10 0.7.1

**27 April 2012**

- breaking change: no longer accept URLs without a scheme
- deprecation of error_dir & context-manager mode
- addition of overridable accept_response hook
- bugfix: retry on more requests errors
- bugfix: unicode cached content no longer incorrectly encoded
- implement various requests enhancements separately for ease of reuse
- convert more Scraper parameters to properties

### 2.5.11 0.7.0

**23 April 2012**

- rewritten internals to use requests, dropping httplib2
- as a result of rewrite, caching behavior no longer attempts to be compliant with the HTTP specification but is much more configurable
- added cache_write_only option
- deprecation of accept_cookies, use_cache_first, cache_dir parameter
- improved tests
- improved Python 3 support

### 2.5.12 0.6.2

**20 April 2012**

- bugfix for POST-redirects
- drastically improved test coverage
- add encoding to ResultStr

### 2.5.13 0.6.1

**19 April 2012**

- add .bytes attribute to ResultStr
- bugfix related to bytes in urlretrieve

### 2.5.14 0.6.0

**19 April 2012**

- remove urllib2 fallback for HTTP
- rework entire test suite to not rely on Flask

- Unicode & Str unification
- experimental Python 3.2 support

### 2.5.15 0.5.8

**15 February 2012**

- fix to test suite from Alex Chiang

### 2.5.16 0.5.7

**2 February 2012**

- -p, –postdata parameter
- argv fix for IPython <= 0.10 from Joe Germuska
- treat FTP 550 errors as HTTP 404s
- use_cache_first improvements

### 2.5.17 0.5.6

**9 November 2011**

- scrapeshell fix for IPython >= 0.11
- scrapelib.urlopen can take method/body params too

### 2.5.18 0.5.5

**27 September 2011**

- use None for no timeout, never create non-blocking socket
- documentation and owernship changes

### 2.5.19 0.5.4

**7 June 2011**

- actually fix reinstantiation of Http object

### 2.5.20 0.5.3

**7 June 2011**

- bugfix for reinstantiation of Http object

### 2.5.21 0.5.2

**16 May 2011**

- support timeout for urllib2 requests

### 2.5.22 0.5.1

**6 April 2011**

- bugfix for exception handling on retry
- fix a deprecation warning for Python 2.6+

### 2.5.23 0.5.0

**18 March 2011**

- sphinx documentation
- addition of scrapeshell
- addition of retry_on_404 parameter to urlopen
- bugfix to exception handling scope issue
- bugfix within tests to avoid false negative

### 2.5.24 0.4.3

**11 February 2011**

- fix retry on certain httplib2 errors
- add a top-level urlopen function

### 2.5.25 0.4.2

**8 February 2011**

- fix retry on socket errors
- close temporary file handle

### 2.5.26 0.4.1

**7 December 2010**

- support retry of requests that produce socket timeouts
- increased test coverage

### 2.5.27 0.4.0

**8 November 2010**

- bugfix: tests require unittest2 or python 2.7
- configurable retry handling for random failures

### 2.5.28 0.3.0

**5 October 2010**

- bugfixes for cookie handling
- better test suite
- follow redirects even after a POST
- change several configuration variables into properties
- request timeout argument

### 2.5.29 0.2.0

**9 July 2010**

- use_cache_first option to avoid extra HTTP HEAD requests
- raise_errors option to treat HTTP errors as exceptions
- addition of urlretrieve

CHAPTER 3

# Indices and tables

- genindex
- modindex
- search

## S

## Symbols



## C



## F



## G



## H



## M



## S



## U