
PyScrawl Documentation

Release 1.2

Nils Corver

Jan 06, 2017

Contents

1	Quickstart	3
2	API Reference	5
2.1	How to Scrawl	5
2.2	Example	5
2.3	API	6
	Python Module Index	9

API client for using scrawl.nl.

CHAPTER 1

Quickstart

Quickstart with a pre-build zipfile:

```
# -*- coding: utf-8 -*-
from pyscrawl import Scrawl

# setup the Scrawl client with your API key
scrawl = Scrawl('MY-API-KEY')

# upload ./document.zip to Scrawl and wait for the conversion.
result = scrawl.upload_zipfile('PyScrawl test', './document.zip')
result.sleep_until_ready()

# direct link to the generated PDF file
pdf_file = result.pdf
```


2.1 How to Scrawl

Sorry, we still need to write this. Scrawl is still in beta, so this is likely to change a bit. When the implementation cools down a bit, the actual setup of a Scrawl document will be explained here.

2.2 Example

Quickstart with a pre-build zipfile:

```
# -*- coding: utf-8 -*-
from pyscrawl import Scrawl

# setup the Scrawl client with your API key
scrawl = Scrawl('MY-API-KEY')

# upload ./document.zip to Scrawl and wait for the conversion.
result = scrawl.upload_zipfile('PyScrawl test', './document.zip')
result.sleep_until_ready()

# direct link to the generated PDF file
pdf_file = result.pdf
```

You can also build a zipfile on the fly using *ScrawlDocument*.

```
# -*- coding: utf-8 -*-
from pyscrawl import Scrawl, ScrawlDocument

# setup the Scrawl client with your API key
scrawl = Scrawl('MY-API-KEY')

# define an index_html using a template engine (for example)
```

```
index_html = '''
<html>
  <head>
    <link rel="stylesheet" href="./styles.css">
  </head>
  <body>
    
    <h1>Hello World</h1>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla
    accumsan, metus ultrices eleifend gravida, nulla nunc varius lectus,
    nec rutrum justo nibh eu lectus. Ut vulputate semper dui. Fusce erat
    odio, sollicitudin vel erat vel, interdum mattis neque.</p>
  </body>
</html>
'''

# build a new document
document = (ScrawlDocument()
            .add_file('./styles.css')
            .add_file('./awesome-logo.png')
            .add_content('index.html', index_html))

# upload to Scrawl and wait for the conversion.
result = scrawl.upload_document('PyScrawl document', document)
result.sleep_until_ready()

# direct link to the generated PDF file
pdf_file = result.pdf
```

2.3 API

You can build you own tests, validators and sanitizers using these classes.

class `pyscrawl.Scrawl` (*api_key*, *server*='https://scrawl.nl')

Scrawl API service.

This allows you to call our API service with ease.

Parameters

- **api_key** (*str*) – Your Scrawl API key.
- **server** (*str*) – The Scrawl server location.

```
from pyscrawl import Scrawl
scrawl = Scrawl('my-api-key')
```

upload_zipfile (*title*, *filename*, *override*=*False*)

Upload a new document to scrawl.

Parameters

- **title** (*str*) – The title of the document.
- **filename** (*str*) – Name of the zip-file to upload.
- **override** (*bool*) – Override if document with the same name exists. Defaults to *False*.

Return type *ScrawlResultDocument*

```
result = scrawl.upload_zipfile('PyScrawl test', './document.zip')
result.sleep_until_ready()
```

upload_container (*title, container, override=False*)

Upload a new document to scrawl using ScrawlContainer.

You can use this to construct a zip-file on the fly.

Parameters

- **title** (*str*) – The title of the document.
- **container** (*ScrawlContainer*) – The container to upload.
- **override** (*bool*) – Override if document with the same name exists. Defaults to *False*.

Return type *ScrawlResultDocument*

upload_stream (*title, stream, override=False*)

Upload a new document using a BytesIO stream.

We recommend you use *upload_zipfile()* or *upload_container()*, but you can use this when you want to do something more advanced.

Parameters

- **title** (*str*) – The title of the document.
- **stream** (*_IOBase*) – The stream to upload.
- **override** (*bool*) – Override if document with the same name exists. Defaults to *False*.

Return type *ScrawlResultDocument*

class `pyscrawl.ScrawlContainer`

Scrawl Container (in-memory ZipFile).

This allows you to programmatically build a ZipFile in memory. Use *add_file()* and *add_content()* to build the ZipFile. When you are done, you can upload it by calling *Scrawl.upload_container()*.

Parameters **api_key** (*str*) – Your Scrawl API key.

```
from pyscrawl import Scrawl, ScrawlContainer
scrawl = Scrawl('my-api-key')

# define an index_html using a template engine (for example)
index_html = '''<html>...</html>'''

# build a new document
container = (ScrawlContainer()
             .add_file('./styles.css')
             .add_file('./awesome-logo.png')
             .add_content('index.html', index_html))

# upload to Scrawl and wait for the conversion.
result = scrawl.upload_container('PyScrawl test', container)
result.sleep_until_ready()
```

add_file (*filename*)

Add a file from disk to the container.

Parameters **filename** (*str*) – The filename on the disk.

Return type *ScrawlContainer*

add_content (*filename, content*)

Add content to the container.

Parameters

- **filename** (*str*) – How to represent the content.
- **content** (*str*) – The actual content

Return type *ScrawlContainer*

get_stream ()

Build a BytesIO in-memory ZipFile based on the files.

Use *add_file()* and *add_content()* to prepare the container. When done, you can call *get_stream()* to build a in-memory ZipFile based on the current configuration.

When using *Scrawl.upload_container()*, this method is automatically called.

Return type BytesIO

class `pyscrawl.ScrawlResultDocument` (*result, api_key*)

Scrawl Result Document.

When calling the API, this result document is returned. It contains information about your uploaded document.

See *Scrawl.upload_zipfile()* or *Scrawl.upload_container()* for more information.

Variables

- **api_key** (*str*) – The API key.
- **info** (*str*) – Information URL.
- **pdf** (*str*) – URL, for downloading or viewing the PDF.
- **html** (*str*) – URL, for online viewing the uploaded HTML document.
- **converted** (*bool*) – *True* when Scrawl has converted the document to PDF.
- **placed_on** (*datetime.datetime*) – DateTime when the document was added to the Scrawler conversion queue.
- **processed_on** (*datetime.datetime*) – DateTime when the document was processed by the Scrawler (*converted* is *True*, when this is set).

sleep_until_ready (*delay=1*)

Sleep (in delays) until Scrawl has converted the document.

Until the *converted* bit is set to *True*. You can always store the pdf link, even if the conversion has not yet been completed.

Parameters **delay** (*int*) – The amount of seconds to sleep (at least 1 second).

Return type bool

Python Module Index

p

`pyscrawl`, 6

A

`add_content()` (`pyscawl.ScrawlContainer` method), 7

`add_file()` (`pyscawl.ScrawlContainer` method), 7

G

`get_stream()` (`pyscawl.ScrawlContainer` method), 8

P

`pyscawl` (module), 6

S

`Scrawl` (class in `pyscawl`), 6

`ScrawlContainer` (class in `pyscawl`), 7

`ScrawlResultDocument` (class in `pyscawl`), 8

`sleep_until_ready()` (`pyscawl.ScrawlResultDocument` method), 8

U

`upload_container()` (`pyscawl.Scrawl` method), 7

`upload_stream()` (`pyscawl.Scrawl` method), 7

`upload_zipfile()` (`pyscawl.Scrawl` method), 6