python-documentcloud Documentation

Release 1.0.4

Los Angeles Times Data Desk

Dec 03, 2018

Contents

A simple Python wrapper for the DocumentCloud API

CHAPTER 1

Features

- Retrieve and edit documents and projects, both public and private, from documentcloud.org
- Upload PDFs into your documentcloud.org account and organize them into projects
- Download text, images and entities extracted from your PDFs by DocumentCloud

CHAPTER 2

Documentation

2.1 Getting started

This tutorial will walk you through the process of installing python-documentcloud and making your first requests.

2.1.1 Installation

Provided that you have pip installed, you can install the library like so:

```
$ pip install python-documentcloud
```

2.1.2 Creating a client

Before you can interact with DocumentCloud, you first must import the library and initialize a client to talk with the site on your behalf.

```
>>> from documentcloud import DocumentCloud
>>> client = DocumentCloud()
```

Since we didn't provide any log-in credentials, the client above will only be able to access published documents. If have an account at DocumentCloud and want to use that, you can provide the crendentials to the client.

```
>>> client = DocumentCloud(USERNAME, PASSWORD)
```

You can also specify a custom uri if you have installed your own version of DocumentCloud

```
>>> client = DocumentCloud(USERNAME, PASSWORD, base_uri="https://your.documentcloud.
↪domain/api/")
```

2.1.3 Searching for documents

You can now you use client to interact with DocumentCloud. A search for documents about journalist Ruben Salazar would look like this:

```
>>> obj_list = client.documents.search("Ruben Salazar")
>>> # Let's grab the first one and look at it
>>> obj = obj_list[0]
>>> obj
<Document: Final OIR Report>
```

2.1.4 Interacting with a document

Once you have you hands on a document object, you can interact with the metadata stored at documentcloud.org. Here's a sample:

```
>>> print obj.title
Final OIR Report
>>> print obj.id
71072-oir-final-report
>>> print obj.contributor_organization
Los Angeles Times
>>> print obj.canonical_url
http://www.documentcloud.org/documents/71072-oir-final-report.html
```

You can even download the PDF, page images and full text.

```
>>> obj.large_image_url
...
>>> obj.large_image
...
>>> obj.full_text
...
>>> obj.pdf
...
```

2.1.5 Uploading a document

You can upload a PDF document from your local machine to documentcloud.org. Here's how:

```
>>> from documentcloud import DocumentCloud
>>> client = DocumentCloud(DOCUMENTCLOUD_USERNAME, DOCUMENTCLOUD_PASSWORD)
>>> obj = client.documents.upload("/home/ben/pdfs/myfile.pdf")
```

And you don't have to provide a path, you can also upload a file object.

```
>>> from documentcloud import DocumentCloud
>>> client = DocumentCloud(DOCUMENTCLOUD_USERNAME, DOCUMENTCLOUD_PASSWORD)
>>> pdf = open("/home/ben/pdfs/myfile.pdf", "rb")
>>> obj = client.documents.upload(pdf)
```

You can also provide URLs that link to PDFs, if that's the kind of thing you're into.

```
>>> client.documents.upload("http://ord.legistar.com/Chicago/attachments/e3a0cbcb-
↪044d-4ec3-9848-23c5692b1943.pdf")
```

2.1.6 Interacting with a newly uploaded public document

When you first upload a document, your local document object will not reflect some of the metadata and processing that happens in the first few seconds it is on the server. Documents set to public will be shown as private during that short interval. To interact with a document as soon as it is available, you can write a short loop to check whether it is ready.

First upload the document as normal.

```
>>> import time
>>> from documentcloud import DocumentCloud
>>> client = DocumentCloud(DOCUMENTCLOUD_USERNAME, DOCUMENTCLOUD_PASSWORD)
>>> obj = client.documents.upload("/home/ben/pdfs/myfile.pdf", access='public')
```

Then refresh your local document object from the server. If it is does not show up as public, then it is still processing, and you'll have to check again.

```
>>> obj = client.documents.get(obj.id)
>>> while obj.access != 'public':
>>>     time.sleep(5)
>>>     obj = client.documents.get(obj.id)
```

2.1.7 Uploading a directory of documents as a project

Here's how to upload a directory full of documents and add them all to a new project. Be warned, this will upload any documents in directories inside the path you specify.

```
>>> # Connect to documentcloud
>>> from documentcloud import DocumentCloud
>>> client = DocumentCloud(DOCUMENTCLOUD_USERNAME, DOCUMENTCLOUD_PASSWORD)
>>> # Create the project
>>> project, created = client.projects.get_or_create_by_title("Groucho Marx's FBI file
↪")
>>> # Upload all the pdfs
>>> obj_list = client.documents.upload_directory('/home/ben/pdfs/groucho_marx/')
>>> # Add the newly created documents to the project
>>> project.document_list = obj_list
>>> # Save the changes to the project
>>> project.put()
```

2.1.8 Securely uploading a document

How to upload a document, but prevent it from being sent to DocumentCloud's third-party services like OpenCalais.

```
>>> from documentcloud import DocumentCloud
>>> client = DocumentCloud(DOCUMENTCLOUD_USERNAME, DOCUMENTCLOUD_PASSWORD)
>>> obj = client.documents.upload("/home/ben/pdfs/myfile.pdf", secure=True)
```

2.1.9 Uploading a PDF from a URL

How to read a PDF document from a URL on the World Wide Web and upload it to DocumentCloud without saving it to your local hard drive.

```
>>> from documentcloud import DocumentCloud
>>> url = "http://myhost.org/interesting-doc.pdf"
>>> client = DocumentCloud(DOCUMENTCLOUD_USERNAME, DOCUMENTCLOUD_PASSWORD)
>>> # Upload the specified URL to the given client
>>> obj = client.documents.upload(url)
```

2.2 Documents

Methods for drawing down, editing and uploading data about documents.

2.2.1 Retrieval

`client.documents.`**`get`**(*id*)

Return the document with the provided DocumentCloud identifer.

```
>>> from documentcloud import DocumentCloud
>>> client = DocumentCloud(USERNAME, PASSWORD)
>>> client.documents.get('71072-oir-final-report')
<Document: Final OIR Report>
```

`client.documents.`**`search`**(*keyword*, *page=None*, *per_page=1000*, *mentions=3*, *data=False*)

Return a list of documents that match the provided keyword.

```
>>> from documentcloud import DocumentCloud
>>> client = DocumentCloud()
>>> obj_list = client.documents.search('Ruben Salazar')
>>> obj_list[0]
<Document: Final OIR Report>
```

DocumentCloud paginates search results. By default the search methods returns all pages. If you want to restrict the number of pages that are searched or retrieve a specific page you should provide some combination of the following keyword arguments.

```
>>> obj_list = client.documents.search('Ruben Salazar', page=1, per_page=10)
>>> # You can guess that will do.
>>> len(obj_list) == 10
>>> True
```

By default, the search returns three mentions of the result in each document. You can increase that number up to 10 by modifying the keyword argument.

```
>>> client.documents.search('Ruben Salazar', mentions=10)
```

Unlike when you get a document directly via id, the key/value dictionaries they can be assigned are not provided by default in search results.

To have them included, switch the following keyword argument.

```
>>> client.documents.search('Ruben Salazar', data=True)
```

2.2.2 Editing

`document_obj.put()`
Save changes to a document back to DocumentCloud. You must be authorized to make these changes. Only the `title`, `source`, `description`, `related_article`, `published_url`, `access` and `data` attributes may be edited.

```
>>> # Grab a document
>>> obj = client.documents.get('71072-oir-final-report')
>>> print obj.title
Draft OIR Report
>>> # Change its title
>>> obj.title = "Brand new title"
>>> print obj.title
Brand New Title
>>> # Save those changes
>>> obj.put()
```

`document_obj.delete()`
Delete a document from DocumentCloud. You must be authorized to make these changes.

```
>>> obj = client.documents.get('71072-oir-final-report')
>>> obj.delete()
```

`document_obj.save()`
An alias for `put` that saves changes back to DocumentCloud.

2.2.3 Uploading

`client.documents.upload(pdf, title=None, source=None, description=None, related_article=None, published_url=None, access='private', project=None, data=None, secure=False, force_ocr=False)`
Upload a PDF to DocumentCloud. You must be authorized to do this. Returns the object representing the new record you've created. You can submit either a file path or a file object.

```
>>> from documentcloud import DocumentCloud
>>> client = DocumentCloud(USERNAME, PASSWORD)
>>> new_id = client.documents.upload("/home/ben/test.pdf", "Test PDF")
>>> # Now fetch it
>>> client.documents.get(new_id)
<Document: Test PDF>
```

You can also URLs link to PDFs, if that's the kind of thing you want to do.

```
>>> client.documents.upload("http://ord.legistar.com/Chicago/attachments/
↪e3a0cbcb-044d-4ec3-9848-23c5692b1943.pdf")
```

`client.documents.upload_directory(pdf, source=None, description=None, related_article=None, published_url=None, access='private', project=None, data=None, secure=False, force_ocr=False)`
Searches through the provided path and attempts to upload all the PDFs it can find. Metadata provided to the other keyword arguments will be recorded for all uploads. Returns a list of document objects that are created. Be warned, this will upload any documents in directories inside the path you specify.

```
>>> from documentcloud import DocumentCloud
>>> client = DocumentCloud(DOCUMENTCLOUD_USERNAME, DOCUMENTCLOUD_PASSWORD)
>>> obj_list = client.documents.upload_directory('/home/ben/pdfs/groucho_marx/')
```

2.2.4 Metadata

`document_obj.`**`access`**
: The privacy level of the resource within the DocumentCloud system. It will be either `public`, `private` or `organization`, the last of which means the is only visible to members of the contributors organization. Can be edited and saved with a put command.

`document_obj.`**`annotations`**
: A list of the annotations users have left on the document. The data are modeled by their own Python class, defined in the *Annotations* section.

```
>>> obj = client.documents.get('83251-fbi-file-on-christopher-biggie-smalls-
↪wallace')
>>> obj.annotations
[<Annotation>, <Annotation>, <Annotation>, <Annotation>, <Annotation>]
```

`document_obj.`**`canonical_url`**
: The URL where the document is hosted at documentcloud.org.

`document_obj.`**`contributor`**
: The user who originally uploaded the document.

`document_obj.`**`contributor_organization`**
: The organizational affiliation of the user who originally uploaded the document.

`document_obj.`**`created_at`**
: The date and time that the document was created, in Python's datetime format.

`document_obj.`**`data`**
: A dictionary containing supplementary data linked to the document. This can any old thing. It's useful if you'd like to store additional metadata. Can be edited and saved with a put command.

  Some keywords are reserved by DocumentCloud and you'll get an error if you try to submit them here. They are: person, organization, place, term, email, phone, city, state, country, title, description, source, account, group, project, projectid, document, access, filter.

```
>>> obj = client.documents.get('83251-fbi-file-on-christopher-biggie-smalls-
↪wallace')
>>> obj.data
{'category': 'hip-hop', 'byline': 'Ben Welsh', 'pub_date': datetime.date(2011, 3,␣
↪1)}
```

  Keys and values also must be strings. No integers or other numbers.

```
>>> obj.data = dict(number=1)
TypeError: data attribute values must be strings
```

`document_obj.`**`description`**
: A summary of the document. Can be edited and saved with a put command.

`document_obj.`**`entities`**
: A list of the entities extracted from the document by OpenCalais. The data are modeled by their own Python class, defined in the *Entities* section.

```
>>> obj = client.documents.get('83251-fbi-file-on-christopher-biggie-smalls-
↪wallace')
>>> obj.entities
[<Entity: Angeles>, <Entity: FD>, <Entity: OO>, <Entity: Los Angeles>, ...
```

document_obj.**file_hash**
A hash representation of the raw PDF data as a hexadecimal string.

```
>>> obj = client.documents.get('1021571-lafd-2013-hiring-statistics')
>>> obj.file_hash
'872b9b858f5f3e6bb6086fec7f05dd464b60eb26'
```

You could recreate this hexadecimal hash yourself using the SHA-1 algorithm.

```
>>> import hashlib
>>> hashlib.sha1(obj.pdf).hexdigest()
'872b9b858f5f3e6bb6086fec7f05dd464b60eb26'
```

document_obj.**full_text**
Returns the full text of the document, as extracted from the original PDF by DocumentCloud. Results may vary, but this will give you what they got. Currently, DocumentCloud only makes this available for public documents.

```
>>> obj = client.documents.get('71072-oir-final-report')
>>> obj.full_text
"Review of the Los Angeles County Sheriff's\nDepartment's Investigation into␣
↪the\nHomicide of Ruben Salazar\nA Special Report by the\nLos Angeles County␣
↪Office of Independent Review\n ...
```

document_obj.**full_text_url**
Returns the URL that contains the full text of the document, as extracted from the original PDF by DocumentCloud.

document_obj.**get_page_text**(*page*)
Submit a page number and receive the raw text extracted from it by DocumentCloud.

```
>>> obj = client.documents.get('1088501-adventuretime-alta')
>>> txt = obj.get_page_text(1)
# Let's print just the first line
>>> print txt.decode().split("\n")[0]
STATE OF CALIFORNIA- HEALTH AND HUMAN SERVICES AGENCY
```

document_obj.**id**
The unique identifer of the document in DocumentCloud's system. Typically this is a string that begins with a number, like `83251-fbi-file-on-christopher-biggie-s.malls-wallace`

document_obj.**large_image**
Returns the binary data for the "large" sized image of the document's first page. If you would like the data for some other page, pass the page number into `document_obj.get_large_image(page)`. Currently, DocumentCloud only makes this available for public documents.

document_obj.**large_image_url**
Returns a URL containing the "large" sized image of the document's first page. If you would like the URL for some other page, pass the page number into `document_obj.get_large_image_url(page)`.

document_obj.**large_image_url_list**
Returns a list of URLs for the "large" sized image of every page in the document.

document_obj.**mentions**
When the document has been retrieved via a search, this returns a list of places the search keywords appear in the text. The data are modeled by their own Python class, defined in the *Mentions* section.

```
>>> obj_list = client.documents.search('Christopher Wallace')
>>> obj = obj_list[0]
>>> obj.mentions
[<Mention: Page 2>, <Mention: Page 3> ....
```

document_obj.**normal_image**
Returns the binary data for the "normal" sized image of the document's first page. If you would like the data for some other page, pass the page number into `document_obj.get_normal_image(page)`. Currently, DocumentCloud only makes this available for public documents.

document_obj.**normal_image_url**
Returns a URL containing the "normal" sized image of the document's first page. If you would like the URL for some other page, pass the page number into `document_obj.get_normal_image_url(page)`.

document_obj.**normal_image_url_list**
Returns a list of URLs for the "normal" sized image of every page in the document.

document_obj.**pages**
The number of pages in the document.

document_obj.**pdf**
Returns the binary data for document's original PDF file. Currently, DocumentCloud only makes this available for public documents.

document_obj.**pdf_url**
Returns a URL containing the binary data for document's original PDF file.

document_obj.**published_url**
Returns an URL outside of documentcloud.org where this document has been published.

document_obj.**related_article**
Returns an URL for a news story related to this document.

document_obj.**sections**
A list of the sections earmarked in the text by a user. The data are modeled by their own Python class, defined in the *Sections* section.

```
>>> obj = client.documents.get('74103-report-of-the-calpers-special-review')
>>> obj.sections
[<Section: Letter to Avraham Shemesh and Richard Resller of SIM Group>, <Section:␣
↪Letter to Ralph Whitworth, founder of Relational Investors>, ...
```

document_obj.**small_image**
Returns the binary data for the "small" sized image of the document's first page. If you would like the data for some other page, pass the page number into `document_obj.get_small_image(page)`. Currently, DocumentCloud only makes this available for public documents.

document_obj.**small_image_url**
Returns a URL containing the "small" sized image of the document's first page. If you would like the URL for some other page, pass the page number into `document_obj.get_small_image_url(page)`.

document_obj.**small_image_url_list**
Returns a list of URLs for the "small" sized image of every page in the document.

document_obj.**source**
The original source of the document. Can be edited and saved with a put command.

`document_obj.`**`thumbnail_image`**
: Returns the binary data for the "thumbnail" sized image of the document's first page. If you would like the data for some other page, pass the page number into `document_obj.get_thumbnail_image(page)`. Currently, DocumentCloud only makes this available for public documents.

`document_obj.`**`thumbnail_image_url`**
: Returns a URL containing the "thumbnail" sized image of the document's first page. If you would like the URL for some other page, pass the page number into `document_obj.get_small_thumbnail_url(page)`.

`document_obj.`**`thumbnail_image_url_list`**
: Returns a list of URLs for the "small" sized image of every page in the document.

`document_obj.`**`title`**
: The name of the document. Can be edited and saved with a put command.

`document_obj.`**`updated_at`**
: The date and time that the document was last updated, in Python's datetime format.

2.3 Projects

Methods for drawing down, editing and uploading data about DocumentCloud projects. A project is a group of documents.

2.3.1 Retrieval

`client.projects.`**`get`**`(`*id=None*, *title=None*`)`
: Return the project with the provided DocumentCloud identifer. You can retrieve projects using either the *id* or *title*.

```
>>> from documentcloud import DocumentCloud
>>> client = DocumentCloud(USERNAME, PASSWORD)
>>> # Fetch using the id
>>> obj = client.projects.get(id='816')
>>> obj
<Project: The Ruben Salazar Files>
>>> # Fetch using the title
>>> obj = client.projects.get(title='The Ruben Salazar Files')
>>> obj
<Project: The Ruben Salazar Files>
```

`client.projects.`**`get_by_id`**`(`*id*`)`
: Return the project with the provided id. Operates the same as *client.projects.get*.

`client.projects.`**`get_by_title`**`(`*title*`)`
: Return the project with the provided title. Operates the same as *client.projects.get*.

`client.projects.`**`all`**`()`
: Return all projects for the authorized DocumentCloud account

```
>>> from documentcloud import DocumentCloud
>>> client = DocumentCloud(USERNAME, PASSWORD)
>>> obj_list = client.projects.all()
>>> obj_list[0]
<Project: Ruben Salazar>
```

2.3.2 Editing

`project_obj.put()`
Save changes to a project back to DocumentCloud. You must be authorized to make these changes. Only the *title*, *source*, *document_list* attributes may be edited.

```
>>> obj = client.projects.get('816')
>>> obj.title = "Brand new title"
>>> obj.put()
```

`project_obj.delete()`
Delete a project from DocumentCloud. You must be authorized to make these changes.

```
>>> obj = client.projects.get('816')
>>> obj.delete()
```

`project_obj.save()`
An alias for `put` that saves changes back to DocumentCloud.

2.3.3 Creation

`client.projects.create(title=None, description=None, document_ids=None)`
Create a new project on DocumentCloud. You must be authorized to do this. Returns the object representing the new record you've created.

```
>>> from documentcloud import DocumentCloud
>>> client = DocumentCloud(USERNAME, PASSWORD)
>>> obj = client.projects.create("New project")
>>> obj
<Project: New project>
```

`client.projects.get_or_create_by_title(title=None)`
Fetch the project with provided name, or create it if it does not exist. You must be authorized to do this. Returns a tuple. An object representing the record comes first. A boolean that reports whether or not the objects was created fresh comes second. It is true when the record was created, false when it was found on the site already.

```
>>> from documentcloud import DocumentCloud
>>> client = DocumentCloud(USERNAME, PASSWORD)
>>> # The first time it will be created and added to documentcloud.org
>>> obj, created = client.projects.get_or_create_by_title("New project")
>>> obj, created
<Project: New project>, True
>>> # The second time it will be fetched from documentcloud.org
>>> obj, created = client.projects.get_or_create_by_title("New project")
>>> obj, created
<Project: New project>, False
```

2.3.4 Metadata

`project_obj.description`
A summary of the project. Can be edited and saved with a put command.

`project_obj.document_ids`
A list that contains the unique identifier of the documents assigned to this project. Cannot be edited. Edit the document_list instead.

```
>>> obj = client.projects.get('816')
>>> obj.document_ids
[u'19419-times-columnist-ruben-salazar-killed-by-bullet', u'19420-usps-american-
↪journalists-stamp', u'19280-fbi-file-on-el-paso-investigations', u'19281-letter-
↪from-the-lapd-chief', ...
```

`project_obj.`**`document_list`**
: A list that documents assigned to this project. Can be expanded by appending new documents to the list or cleared by reassigning it as an empty list and then issuing the put command.

```
>>> obj = client.projects.get('816')
>>> obj.document_list
[<Document: Times Columnist Ruben Salazar Slain by Tear-gas Missile>, <Document:␣
↪Salazar's Legacy Lives On>, <Document: Cub Reporter Catches Attention of El␣
↪Paso FBI>, ...
```

`project_obj.`**`get_document`**`(`*id*`)`
: Retrieves a particular document from the project using the provided DocumentCloud identifer.

`project_obj.`**`id`**
: The unique identifer of the project in DocumentCloud's system. Typically this is a number.

`project_obj.`**`title`**
: The name of the project. Can be edited and saved with a put command.

2.4 Other objects

Other types of data provided by the DocumentCloud system.

2.4.1 Annotations

Notes left in documents.

`annotation_obj.`**`access`**
: The privacy level of the resource within the DocumentCloud system. It will be either `public` or `private`.

`annotation_obj.`**`description`**
: Space for a lengthy text block that will be published below the highlighted text in the DocumentCloud design.

`annotation_obj.`**`id`**
: The unique identifer of the document in DocumentCloud's system.

`annotation_obj.`**`location`**
: The location of where the annotation appears on the document's page. Defined by the *Locations* class.

`annotation_obj.`**`page`**
: The page where the annotation appears.

`annotation_obj.`**`title`**
: The name of the annotation, which appears in the table of contents and above the highlighted text when published by DocumentCloud.

2.4.2 Entities

Keywords extracted from documents with OpenCalais.

`location_obj.`**`relevance`**
: The weighting associated with this connection by OpenCalais. Higher numbers are supposed to be more relevant.

`location_obj.`**`type`**
: The category of entity the value belongs to.

`location_obj.`**`value`**
: The name of the entity extracted from the document (i.e. "Los Angeles" or "Museum of Modern Art")

2.4.3 Locations

The location where *Annotations* are placed within a document.

`location_obj.`**`bottom`**
: The value of the bottom edge of an annotation.

`location_obj.`**`left`**
: The value of the left edge of an annotation.

`location_obj.`**`right`**
: The value of the right edge of an annotation.

`location_obj.`**`top`**
: The value of the top edge of an annotation.

2.4.4 Mentions

Mentions of a search keyword found in one of the documents.

`mention_obj.`**`page`**
: The page where the mention occurs.

`mention_obj.`**`text`**
: The text surrounding the mention of the keyword.

2.4.5 Sections

Sections of the documents earmarked by users.

`section_obj.`**`title`**
: The name of the section.

`section_obj.`**`page`**
: The page where the section begins.

2.5 Changelog

2.5.1 1.0.4

- Throw an error when integers or other non-strings are included in Document metadata dictionaries
- Added a number of keyword arguments to documents searches to pull a single page, change page size and request document metadata in result

- Temporarily removed SSL from image and text URLs to workaround bugs in underlying dependencies

2.5.2 1.0.3

- Encoding bug fix for metadata associated with documents via API

2.5.3 1.0.2

- URLs to PDFs can now be submitted for upload
- Refactored setup.py and tests to be less complex

2.5.4 1.0.1

- Python 3.4 testing
- 400MB upload limit to match DocumentCloud's API restrictions

2.5.5 1.0.0

- Adopted semantic versioning without breaking existing packages on PyPI
- Fixed bugs with `get_page_text`
- Added keyword argument during initialization that allows you to override the `BASE_URI` and connect with independent clones of DocumentCloud. Contributed by Adi Eyal.
- Refactored unit tests to run more quickly and require fewer web requests
- Documentation moved from the gh-pages branch to master and refactored to be published via ReadTheDocs.

2.5.6 0.2

- Python 3 support
- PEP8 and PyFlakes compliance
- Coverage reports on testing via coveralls.io

2.5.7 0.16

- Continuous integration testing with TravisCI
- Fixed bug with empty strings in Document descriptions
- Raise errors when a user tries to save a data keyword reserved by DocumentCloud
- Allow all-caps file extensions
- Retry requests that fail with an increasing backoff delay
- Fixed a bug in how titles are assigned to a file object
- Added access checks when retrieving txt, pdf, img about a document

2.5.8 0.15

- File objects can now be submitted for uploading
- Added more support for unicode data thanks to contributions by Shane Shifflet.
- Smarter lazy loading of Document attributes missing from a search

2.5.9 0.14

- Added `data` attribute on Document for storing dictionaries of arbitrary metadata
- Added `secure` option for Document uploads to prevent data from being sent to OpenCalais
- Added `save` alias on Document and Project objects that uses the pre-existing `put` command
- Fixed to url encoding to makes the system more unicode friendly
- Added all Document upload arguments to `upload_directory` method

2.5.10 0.13

- `upload_directory` method for documents

2.5.11 0.12

- `get_or_create_by_title` method for projects
- Document and project creation methods now return an object, not the new id.
- Projects can pulled by id or by title

2.5.12 0.11

- Document search now returns `mentions` of the keyword in the documents
- `related_url` and `published_url` attributes now more easily accessible
- `normal` sized images now available

2.6 Credits

The lead developer of this project is Ben Welsh.

But it would not be possible without:

- The DocumentCloud team.
- Chris Amico, Christopher Groskopf and Mitchell Kotler, who broke ground with code that I've adapted.
- Fixes from friendly people like Joe Germuska, Shane Shifflet and Adi Eyal.

CHAPTER 3

Contributing

- Code repository: https://github.com/datadesk/python-documentcloud
- Issues: https://github.com/datadesk/python-documentcloud/issues
- Packaging: https://pypi.python.org/pypi/python-documentcloud
- Testing: https://travis-ci.org/datadesk/python-documentcloud
- Coverage: https://coveralls.io/r/datadesk/python-documentcloud

Index

R

S

T

U

V