Python Congress Documentation

Release 0.3.2

Chris Amico

Mar 04, 2018

Contents:

A Python client for the ProPublica Congress API

CHAPTER 1

Install

From PyPI:

```
pip install python-congress
```

Download and run the install script:

```
git clone https://github.com/eyeseast/propublica-congress && cd propublica-congress
python setup.py install
```

CHAPTER 2

Usage

The main entrypoint for the API is the Congress class, which is instantiated with your API key. (Request an API key at ProPublica's data store.)

Endpoints are organized into subclients attached to the main Congress instance. For example:

```
>>> from congress import Congress
>>> congress = Congress(API_KEY)

# get member by bioguide ID
>>> pelosi = congress.members.get('P000197')
>>> pelosi['twitter_id']
'NancyPelosi'

# get recent House bills
# recent bills come in two types: 'introduced' and 'updated'
>>> introd = congress.bills.recent(
...     chamber='house',
...     congress=115,
...     type='introduced')

>>> len(introd['bills'])
20

>>> introd['chamber']
'House'

# or use a convenience function
>>> introd = congress.bills.introduced('house')
>>> introd['chamber']
'House'
>>> len(introd['bills'])
20
```

2.1 API

2.1.1 Congress

A Python client for the ProPublica Congress API

API docs: https://propublica.github.io/congress-api-docs/

class congress.**Congress**(*apikey=None*, *cache='.cache'*, *http=None*)

Implements the public interface for the ProPublica Congress API

Methods are namespaced by topic (though some have multiple access points). Everything returns decoded JSON, with fat trimmed.

In addition, the top-level namespace is itself a client, which can be used to fetch generic resources, using the API URIs included in responses. This is here so you don't have to write separate functions that add on your API key and trim fat off responses.

Create a new instance with your API key, or set an environment variable called `PROPUBLICA_API_KEY`.

Congress uses httplib2, and caching is pluggable. By default, it uses httplib2.FileCache, in a directory called `.cache`, but it should also work with memcache or anything else that exposes the same interface as FileCache (per httplib2 docs).

Example: Using a custom cache object

Redis is a good option for a cache that can be shared between processes.

```
>>> from redis import StrictRedis
>>> from congress import Congress
>>> db = StrictRedis()
>>> congress = Congress(API_KEY, cache=db)
>>> senate = congress.members.filter('senate') # hits the API, caching the result
>>> senate = congress.members.filter('senate') # uses the cache
```

2.1.2 Members

class congress.members.**MembersClient**(*apikey=None*, *cache='.cache'*, *http=None*)

bills(*member_id*, *type='introduced'*)

Same as BillsClient.by_member

compare(*first*, *second*, *chamber*, *type='votes'*, *congress=115*)

See how often two members voted together in a given Congress. Takes two member IDs, a chamber and a Congress number.

departing(*chamber*, *congress=115*)

Takes a chamber and congress and returns a list of departing members

filter(*chamber*, *congress=115*, ***kwargs*)

Takes a chamber and Congress, OR state and district, returning a list of members

get(*member_id*)

Takes a bioguide_id, returns a legislator

new(***kwargs*)

Returns a list of new members

party()
: Get state party counts for the current Congress

2.1.3 Bills

class congress.bills.**BillsClient**(*apikey=None*, *cache='.cache'*, *http=None*)

by_member(*member_id*, *type='introduced'*)
: Takes a bioguide ID and a type: (introduced|updated|cosponsored|withdrawn) Returns recent bills

introduced(*chamber*, *congress=115*)
: Shortcut for getting introduced bills

major(*chamber*, *congress=115*)
: Shortcut for major bills

passed(*chamber*, *congress=115*)
: Shortcut for passed bills

recent(*chamber*, *congress=115*, *type='introduced'*)
: Takes a chamber, Congress, and type: (introduced|updated) Returns a list of recent bills

upcoming(*chamber*, *congress=115*)
: Shortcut for upcoming bills

updated(*chamber*, *congress=115*)
: Shortcut for getting updated bills

2.1.4 Votes

class congress.votes.**VotesClient**(*apikey=None*, *cache='.cache'*, *http=None*)

by_date(*chamber*, *date*)
: Return votes cast in a chamber on a single day

by_month(*chamber*, *year=None*, *month=None*)
: Return votes for a single month, defaulting to the current month.

by_range(*chamber*, *start*, *end*)
: Return votes cast in a chamber between two dates, up to one month apart.

by_type(*chamber*, *type*, *congress=115*)
: Return votes by type: missed, party, lone no, perfect

get(*chamber*, *rollcall_num*, *session*, *congress=115*)
: Return a specific roll-call vote, including a complete list of member positions

loneno(*chamber*, *congress=115*)
: How often is each member the lone no vote?

missed(*chamber*, *congress=115*)
: Missed votes by member

nominations(*congress=115*)
: Return votes on nominations from a given Congress

party(*chamber*, *congress=115*)
: How often does each member vote with their party?

perfect (*chamber*, *congress=115*)
Who never misses a vote?

today (*chamber*)
Return today's votes in a given chamber

2.1.5 Committees

class congress.committees.**CommitteesClient** (*apikey=None*, *cache='.cache'*, *http=None*)

2.1.6 Nominations

class congress.nominations.**NominationsClient** (*apikey=None*, *cache='.cache'*, *http=None*)

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index