

---

# **viapy Documentation**

***Release 0.2.0***

**CDH @ Princeton**

**Dec 02, 2020**



---

## Contents:

---

<b>1</b>	<b>viapy Code Documentation</b>	<b>1</b>
1.1	API . . . . .	1
1.2	Views . . . . .	2
1.3	Widgets . . . . .	3
<b>2</b>	<b>CHANGELOG</b>	<b>5</b>
2.1	0.2 . . . . .	5
2.2	0.1.4 . . . . .	5
2.3	0.1.3 . . . . .	5
2.4	0.1.2 . . . . .	5
2.5	0.1.1 . . . . .	5
2.6	0.1 . . . . .	5
<b>3</b>	<b>Installation</b>	<b>7</b>
<b>4</b>	<b>Configuration for use with Django</b>	<b>9</b>
<b>5</b>	<b>Development instructions</b>	<b>11</b>
5.1	Unit Testing . . . . .	11
5.2	Documentation . . . . .	12
<b>6</b>	<b>License</b>	<b>13</b>
<b>7</b>	<b>Indices and tables</b>	<b>15</b>
	<b>Python Module Index</b>	<b>17</b>
	<b>Index</b>	<b>19</b>



### 1.1 API

```
class viapy.api.SRUItem(items=None, sequence_type=<type 'tuple'>)
    Single item returned by a SRU search, for use with ViafAPI.search() and SRUResult.

    label
        first main heading for this item

    nametype
        type of name (personal, corporate, title, etc)

    uri
        VIAF URI for this result

    viaf_id
        VIAF numeric identifier

class viapy.api.SRUResult(data)
    SRU search result object, for use with ViafAPI.search().

    records
        list of results as SRUItem.

    total_results
        number of records matching the query

class viapy.api.ViafAPI
    Wrapper for VIAF API.

    https://platform.worldcat.org/api-explorer/apis/VIAF

    api_base = 'https://www.viaf.org/viaf'
        base url for VIAF API methods

    find_corporate(name)
        Search VIAF for local.corporateNames
```

**find\_person** (*name*)  
Search VIAF for local.personalNames

**find\_place** (*name*)  
Search VIAF for local.geographicNames

**search** (*query*)  
Query VIAF search interface. Returns a list of *SRUItem*:param query: CQL query in viaf syntax (e.g.,  
cql.any all "term")

**suggest** (*term*)  
Query autosuggest API. Returns a list of results, or an empty list if no suggestions are found or if something went wrong

**uri\_base** = 'http://viaf.org/viaf'  
base url for VIAF URIs

**classmethod uri\_from\_id** (*viaf\_id*)  
Generate a canonical VIAF URI for the specified id

**class** viapy.api.ViafEntity (*viaf\_id*)  
Object for working with a single VIAF entity.

**Parameters** **viaf\_id** – viaf identifier (either integer or uri)

**birthdate**  
schema birthdate as rdflib.Literal

**birthyear**  
birth year

**deathdate**  
schema deathdate as rdflib.Literal

**deathyear**  
death year

**rdf**  
VIAF data for this entity as rdflib.Graph

**uriref**  
VIAF URI reference as instance of rdflib.URIRef

**classmethod year\_from\_isodate** (*date*)  
Return just the year portion of an ISO8601 date. Expects a string, returns an integer

## 1.2 Views

**class** viapy.views.ViafLookup (\*\*kwargs)  
View to provide VIAF suggestions for autocomplete lookup. Based on dal.  
autocompleteSelect2ListView. Expects search term as query string parameter *q*. Returns viaf  
URI as identifier and display form as text.

**get** (*request*, \*args, \*\*kwargs)  
Return JSON with suggested VIAF ids and display names.

**class** viapy.views.ViafSearch (\*\*kwargs)  
View to provide VIAF suggestions for autocomplete lookup. Based on dal.  
autocompleteSelect2ListView. Expects search term as query string parameter *q*. Returns viaf  
URI as identifier and display form as text.

**get** (*request*, \**args*, \*\**kwargs*)

Return JSON with suggested VIAF ids and display names.

## 1.3 Widgets

**class** viapy.widgets.**ViafWidget** (*url=None*, *forward=None*, \**args*, \*\**kwargs*)

Custom autocomplete select widget that displays VIAF id as a link. Extends `dal.autocomplete.Select2`.

**render** (*name*, *value*, *renderer=None*, *attrs=None*)

Call Django render together with *render\_forward\_conf*.





### 2.1 0.2

- Now supports Django versions 1.11 through 3.0.

### 2.2 0.1.4

- Fix Travis-CI build for building with and without Django.

### 2.3 0.1.3

- Fix GitHub repository name in sphinx documentation config file.

### 2.4 0.1.2

- Update sphinx configuration to support building documentation on readthedocs.org

### 2.5 0.1.1

- Document permissions.

### 2.6 0.1

Initial release.

- Basic support for VIAP API use: autocomplete, SRU search, information about a single VIAF entity.
- Basic Django integration (optional); django-autocomplete-light lookup view and a VIAF url widget.

#### *VIAF via Python*

Python module for interacting with [VIAF](#) (the Virtual International Authority File) data and APIs.

**viapy** provides optional Django integration; this currently includes a django-autocomplete-light lookup view and a VIAF url widget.

## CHAPTER 3

---

### Installation

---

Use pip to install from GitHub. Use a branch or tag name, e.g. @develop or @1.0 if you want to install a specific tagged release or branch:

```
pip install git+https://github.com/Princeton-CDH/viapy.git@develop#egg=viapy
```



---

### Configuration for use with Django

---

Using *viapy* with Django requires additional configuration. Add *viapy* to installed applications along with the needed *django-autocomplete-light* modules:

```
INSTALLED_APPS = (  
    ...  
    'dal',  
    'dal_select2',  
    'viapy',  
    ...  
)
```

Include the *viapy* urls at the desired base url with the namespace:

```
urlpatterns = [  
    ...  
    url(r'^viaf/', include('viapy.urls', namespace='viaf')),  
    ...  
)
```



---

## Development instructions

---

This git repository uses [git flow](#) branching conventions.

Initial setup and installation:

- Recommended: create and activate a python 3.5 virtualenv:

```
virtualenv viapy -p python3.5
source viapy/bin/activate
```

- pip install the package with its python dependencies:

```
pip install -e .
pip install -e ".[django]"
```

### 5.1 Unit Testing

Unit tests are set up to be run with [py.test](#)

- Copy sample test settings and add a **SECRET\_KEY**:

```
cp ci/testsettings.py testsettings.py
```

- To run the tests, either use the configured setup.py test command:

```
python setup.py test
```

- Or install test requirements and use py.test directly:

```
pip install -e '.[test_all]'
py.test
```

## 5.2 Documentation

Documentation is generated using [sphinx](#). To generate documentation, first install development requirements:

```
pip install -e "[docs]"
```

Then build the documentation using the customized make file in the *docs* directory:

```
cd sphinx-docs  
make html
```

When building documentation for a production release, use *make docs* to update the published documentation on GitHub Pages.



## CHAPTER 6

---

### License

---

**viapy** is distributed under the Apache 2.0 License.

©2017 Trustees of Princeton University. Permission granted via Princeton Docket #18-3449-1 for distribution online under a standard Open Source license. Ownership rights transferred to Rebecca Koeser provided software is distributed online via open source.



## CHAPTER 7

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



### V

`viapy.api`, [1](#)  
`viapy.views`, [2](#)  
`viapy.widgets`, [3](#)



## A

`api_base` (*viapy.api.ViafAPI attribute*), 1

## B

`birthdate` (*viapy.api.ViafEntity attribute*), 2

`birthyear` (*viapy.api.ViafEntity attribute*), 2

## D

`deathdate` (*viapy.api.ViafEntity attribute*), 2

`deathyear` (*viapy.api.ViafEntity attribute*), 2

## F

`find_corporate()` (*viapy.api.ViafAPI method*), 1

`find_person()` (*viapy.api.ViafAPI method*), 1

`find_place()` (*viapy.api.ViafAPI method*), 2

## G

`get()` (*viapy.views.ViafLookup method*), 2

`get()` (*viapy.views.ViafSearch method*), 2

## L

`label` (*viapy.api.SRUItem attribute*), 1

## N

`nametype` (*viapy.api.SRUItem attribute*), 1

## R

`rdf` (*viapy.api.ViafEntity attribute*), 2

`records` (*viapy.api.SRUResult attribute*), 1

`render()` (*viapy.widgets.ViafWidget method*), 3

## S

`search()` (*viapy.api.ViafAPI method*), 2

`SRUItem` (*class in viapy.api*), 1

`SRUResult` (*class in viapy.api*), 1

`suggest()` (*viapy.api.ViafAPI method*), 2

## T

`total_results` (*viapy.api.SRUResult attribute*), 1

## U

`uri` (*viapy.api.SRUItem attribute*), 1

`uri_base` (*viapy.api.ViafAPI attribute*), 2

`uri_from_id()` (*viapy.api.ViafAPI class method*), 2

`uriref` (*viapy.api.ViafEntity attribute*), 2

## V

`viaf_id` (*viapy.api.SRUItem attribute*), 1

`ViafAPI` (*class in viapy.api*), 1

`ViafEntity` (*class in viapy.api*), 2

`ViafLookup` (*class in viapy.views*), 2

`ViafSearch` (*class in viapy.views*), 2

`ViafWidget` (*class in viapy.widgets*), 3

`viapy.api` (*module*), 1

`viapy.views` (*module*), 2

`viapy.widgets` (*module*), 3

## Y

`year_from_isodate()` (*viapy.api.ViafEntity class method*), 2