PyClerk

Release 0.0.1

Ryan Giarusso

May 08, 2020

CONTENTS

1 Trying Out PyClerk 3

2 Getting More Advanced 5

3 Expanding PyClerk 7

4 Documentation 9

5 Future Growth 11

6 Indices and tables 13

Use python to access all U.S. caselaw through the Harvard Law School Library Caselaw Access Project.

PyClerk is a Python package that simplifies accessing the Caselaw Access Project's Web API (CAPAPI). Its goal it to reduce the necessary overhead to accessing CAPAPI from reading their detailed but dense documentation to simply importing a python package and trying out a few lines of code.

The current alpha version provides this simplicity for the Cases endpoint, especially for the single case API. While this is somewhat limited functionality compared to the full CAPAPI, this initial release will enable users to access the core data of CAP without leaving Python.

CHAPTER

ONE

TRYING OUT PYCLERK

The the completed project is hosted on the Python Package Index (PyPi).

Installing and trying PyClerk is easy. You'll need Python 3 installed as well as its package manager 'pip'. Some machines come with Python 2 installed as default as an OS dependency. On these, you may need to replace `pip` in the instructions with `pip3`. In a terminal window, or in a python virtual environment if you prefer:

- install PyClerk: `pip install pyclerk`
- start a Python console: `python3`
- import pyclerk and create a PyClerk instance:

```
import pyclerk
pc = pyclerk.PyClerk()
json, body = pc.cases.single_case(435800)  # This returns a specific case, with␣
↪internal id # 435800
```

- the final command will return two data structures `json` and `body`
- `json` is the content reply from the Caselaw Access Project API, assuming the API returns a valid status code. If it doesn't, the appropriate error is raised. This contains case metadata as a json object, with the content of the case as an entry in this structure represented as a bytestring.
- `body` is the content of the case deparsed to be easier to manipulate in Python.

CHAPTER

TWO

GETTING MORE ADVANCED

- from there, explore the docs to expand to include new parameters, new types of searches, and more!
- or, browse the CAPAPI root to identify additional functionality that you need for your project.
- you can also interface directly with the API through PyClerk without using the custom functions or classes:

```
import pyclerk
pc = pyclerk.PyClerk()
# Write out a custom API Query
custom\_url = "https://api.case.law/v1/YOUR CUSTOM REQUEST HERE"
# Send that request to CAPAPI and get a text response
response = pc.custom\_endpoint.send\_request(custom\_url)
# Parse that response into json and a custom body class
```

CHAPTER

THREE

EXPANDING PYCLERK

PyClerk is still under active development. That means you might find a bug or identify new functionality your project needs. The Caselaw Access Project might also update their API to a new version or change various functionality.

- If you find a bug, please file an issue here.
- If you need a new feature, you can file a feature request as an issue, or you can go implement it yourself! Just fork the project, add the feature, and submit a pull-request. I'd love some help.
- If the problem is with the CAPAPI itself, please let them know here.

CHAPTER

FOUR

DOCUMENTATION

Documentation is super important to this project–the whole goal is ease of use for new coders. That requires good documentation!

Rendered versions of documentation are available through ReadTheDocs. It includes both high-level descriptions and overviews (like the installation and first uses instructions above) and rendered versions of the docstrings that accompany the classes and functions in the package.

To rebulid the documentation: - Generate latest raw API docs:

```
sphinx-apidoc -e -o docs/source pyclerk
```

- Build the docs: `make html`

CHAPTER

FIVE

FUTURE GROWTH

The obvious case for future growth is the inclusion of all available endpoints, such as: - bulk - citations - courts - jurisdictions - ngrams - reporters - user_history - volumes - and any others CAP may choose to unveil.

Additionally, as we better define the best uses for this kind of data, PyClerk should grow to include pipelines for processing the API data into formats users want most. This might be in line with some of the sample processing functions I've outlined, or it could be something endusers create that I could never have imagined.

PyClerk will also probably need an update whenever CAP decides to move to v2 of their API.

Areas in the source code ripe for expansion are marked with `#FUTURE`.

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search