PyZillow Documentation

Release 0.7.0

Timo Cornelius Metzger

Jul 06, 2020

Contents

CHAPTER 1

A Python library to access the Zillow API

PyZillow is a Python wrapper for Zillow's API. With PyZillow, you can use a physical address or a Zillow ID to access real estate data from the Zillow database.

Currently, PyZillow supports the **GetDeepSearchResults** and **GetUpdatedPropertyDetails** API endpoints.

CHAPTER 2

Scope of this document

This documentation describes how to use PyZillow to access data through the Zillow API in Python. You will learn how to install PyZillow, initialize the API wrapper with the ZillowConnect class, and how to use the classes GetDeepSearchResults and GetUpdatedPropertyDetails to request and parse data from the Zillow API.

CHAPTER 3

Table of contents

3.1 Installation

3.1.1 Install PyZillow

Use `pip` on a command line to download PyZillow from PyPI and install it on your system:

```
$ pip install pyzillow
```

3.1.2 Getting the code

You can download the most recent version of PyZillow from GitHub:

```
$ git clone https://github.com/hanneshapke/pyzillow.git
$ python setup.py install
```

3.2 Getting started

3.2.1 Obtaining an API key (Zillow Web Service Identifier)

You need an API key from Zillow to request data from the Zillow API. You can apply for an API key by following these instructions: https://www.zillow.com/howto/api/APIOverview.htm. Zillow calls API keys 'Zillow Web Service Identifier'.

3.2.2 Initializing the API

To be able to communicate with the API, you first need to initialize a ZillowWrapper object with your API key. For example:

```
>>> from pyzillow.pyzillow import ZillowWrapper
>>> zillow_data = ZillowWrapper(YOUR_ZILLOW_API_KEY)
```

3.2.3 Accessing the GetDeepSearchResults API

The GetDeepSearchResults API queries the Zillow database for information on a specific address. The endpoint requires the following arguments:

- A street address (e.g. `'2114 Bigelow Ave'`)
- A ZIP code or city and state combination (e.g. `'98109'` or `'Seattle, WA'`)
- Optional: Enabling or disabling Zillow Rentzestimate information in API results (`True`/`False`)

To query the GetDeepSearchResults API:

```
>>> from pyzillow.pyzillow import ZillowWrapper, GetDeepSearchResults
>>> zillow_data = ZillowWrapper(YOUR_ZILLOW_API_KEY)
>>> deep_search_response = zillow_data.get_deep_search_results('2114 Bigelow Ave',
↪'98109', True)
>>> result = GetDeepSearchResults(deep_search_response)
```

An instance of `GetDeepSearchResults` has the following attributes: `.bathrooms` . `bedrooms` `.city` `.fips_county` `.graph_data_link` `.home_detail_link` . `home_size` `.home_type` `.last_sold_date` `.last_sold_price` `.latitude` . `longitude` `.map_this_home_link` `.property_size` `.rentzestimate_amount` `.rentzestimate_last_updated` `.rentzestimate_valuation_range_high` . `rentzestimate_valuation_range_low` `.rentzestimate_value_change` `.state` `.street` `.tax_value` `.tax_year` `.total_rooms` `.use_code` `.year_built` `.zestimate_amount` . `zestimate_last_updated` `.zestimate_percentile` `.zestimate_valuation_range_high` `.zestimate_valuation_range_low` `.zestimate_value_change` `.zillow_id` `.zipcode`

Access the information by reading the `GetDeepSearchResults` object's attributes. For example:

```
>>> print(result.zillow_id)
48749425
>>> print(result.bathrooms)
3.0
```

3.2.4 Accessing the GetUpdatedPropertyDetails API

The GetUpdatedPropertyDetails API endpoint requires a Zillow Property ID (ZPID) as an argument. To find this identifier, you can read the attribute `.zillow_id` of a GetDeepSearchResults object.

Compared to the GetDeepSearchResults API endpoint described above, the GetUpdatedPropertyDetails API endpoint delivers more details about the object, such as `.heating_system` or `.school_district`. However, GetUpdatedPropertyDetails data is not available for all valid Zillow Property IDs.

To query the GetUpdatedPropertyDetails API:

```
>>> from pyzillow.pyzillow import ZillowWrapper, GetUpdatedPropertyDetails
>>> zillow_data = ZillowWrapper(YOUR_ZILLOW_API_KEY)
>>> updated_property_details_response = zillow_data.get_updated_property_details(
↪'48749425')
>>> result = GetUpdatedPropertyDetails(updated_property_details_response)
```

An instance of `GetDeepSearchResults` has the following attributes: `.agent_name` `.agent_profile_url` `.appliances` `.basement` `.bathrooms` `.bedrooms` `.brokerage` `.city` `.cooling_system` `.elementary_school` `.exterior_material` `.floor_material` `.heating_sources` `.heating_system` `.high_school` `.home_description` `.home_detail_link` `.home_info` `.home_size` `.home_type` `.latitude` `.longitude` `.middle_school` `.neighborhood` `.num_floors` `.num_rooms` `.page_view_count_this_month` `.page_view_count_total` `.parking_type` `.photo_gallery` `.posting_agent` `.posting_last_update` `.posting_mls` `.posting_status` `.posting_type` `.price` `.property_size` `.roof` `.rooms` `.school_district` `.state` `.street` `.view` `.year_built` `.year_updated` `.zillow_id` `.zipcode`

Access the information by reading the `GetUpdatedPropertyDetails` object's attributes. For example:

```
>>> print(result.home_type)
SingleFamily
>>> print(result.parking_type)
Off-street
```

3.3 Modules Documentation

3.3.1 pyzillow.pyzillow module

The `ZillowWrapper` class

class pyzillow.pyzillow.**ZillowWrapper**(*api_key: str = None*)

This class provides an interface into the Zillow API. An API key is required to create an instance of this class:

```
>>> from pyzillow.pyzillow import ZillowWrapper
>>> zillow_data = ZillowWrapper(YOUR_ZILLOW_API_KEY)
```

To request data from Zillow, you can choose between:

1. The GetDeepSearchResults API endpoint (*pyzillow.pyzillow.GetDeepSearchResults*) which requires the following arguments:
   - A street address (e.g. `'2114 Bigelow Ave'`)
   - A ZIP code or city and state combination (e.g. `'98109'` or `'Seattle, WA'`)
   - Optional: Enabling or disabling Zillow Rentzestimate information in API results (`True`/`False`)

   Example:

   ```
   >>> from pyzillow.pyzillow import ZillowWrapper, GetDeepSearchResults
   >>> zillow_data = ZillowWrapper(YOUR_ZILLOW_API_KEY)
   >>> deep_search_response = zillow_data.get_deep_search_results(address,
                                                                  zipcode,
                                                                  ␣
   ↪rentzestimate)
   >>> result = GetDeepSearchResults(deep_search_response)
   ```

2. The GetUpdatedPropertyDetails API endpoint (*pyzillow.pyzillow.GetUpdatedPropertyDetails*) which requires a Zillow Property ID (ZPID) as an argument. You can acquire this identifier by accessing `.zillow_id` from a *pyzillow.pyzillow.GetDeepSearchResults* object. GetUpdatedPropertyDetails data is not available for all valid Zillow IDs.

Example:

```
>>> from pyzillow.pyzillow import ZillowWrapper,␣
↪GetUpdatedPropertyDetails
>>> zillow_data = ZillowWrapper(YOUR_ZILLOW_API_KEY)
>>> updated_property_details_response =                    zillow_data.
↪get_updated_property_details(zillow_id)
>>> result = GetUpdatedPropertyDetails(updated_property_details_
↪response)
```

get_data (*url: str*, *params: dict*)

This method requests data from the API endpoint specified in the url argument. It uses parameters from the params argument.

Parameters

- **url** (*str*) – URL of API endpoint
- **params** (*dict*) – Parameters for API query

Raises

- ***ZillowFail*** – The API endpoint could not be reached or the request did not return valid XML
- ***ZillowError*** – The API endpoint responded with an error code
- ***ZillowNoResults*** – The request did not return any results

Returns Result from API query

Return type xml.etree.ElementTree.Element

get_deep_search_results (*address: str*, *zipcode: str*, *rentzestimate: bool = False*)

This method provides results from the GetDeepSearchResults API endpoint as an XML object.

Parameters

- **address** (*str*) – Street address to look up
- **zipcode** (*str*) – ZIP code to look up
- **rentzestimate** (*bool, optional*) – Add Rent Zestimate information to result (True/False), defaults to False

Returns Result from API query

Return type xml.etree.ElementTree.Element

get_updated_property_details (*zpid: str*)

This method provides results from the GetUpdatedPropertyDetails API endpoint as an XML object.

Parameters **zpid** (*str*) – Zillow Web Service Identifier

Returns Result from API query

Return type xml.etree.ElementTree.Element

The GetDeepSearchResults class

class pyzillow.pyzillow.**GetDeepSearchResults** (*data*, **args*, ***kwargs*)

Bases: pyzillow.pyzillow.ZillowResults

Maps results from the XML data array into attributes of an instance of GetDeepSearchResults.

An instance of GetDeepSearchResults has the following attributes: .bathrooms . bedrooms .city .fips_county .graph_data_link .home_detail_link . home_size .home_type .last_sold_date .last_sold_price .latitude . longitude .map_this_home_link .property_size .rentzestimate_amount .rentzestimate_last_updated .rentzestimate_valuation_range_high . rentzestimate_valuation_range_low .rentzestimate_value_change .state .street .tax_value .tax_year .total_rooms .use_code .year_built . zestimate_amount .zestimate_last_updated .zestimate_percentile . zestimate_valuation_range_high .zestimate_valuation_range_low . zestimate_value_change .zillow_id .zipcode

The GetUpdatedPropertyDetails class

class pyzillow.pyzillow.**GetUpdatedPropertyDetails**(*data*, **args*, ***kwargs*)
Bases: pyzillow.pyzillow.ZillowResults

Maps results from the XML data array into attributes of an instance of GetUpdatedPropertyDetails.

An instance of GetUpdatedPropertyDetails has the following attributes: .agent_name .agent_profile_url .appliances .basement .bathrooms .bedrooms . brokerage .city .cooling_system .elementary_school .exterior_material .floor_material .heating_sources .heating_system .high_school . home_description .home_detail_link .home_info .home_size .home_type . latitude .longitude .middle_school .neighborhood .num_floors .num_rooms .page_view_count_this_month .page_view_count_total .parking_type . photo_gallery .posting_agent .posting_last_update .posting_mls . posting_status .posting_type .price .property_size .roof .rooms . school_district .state .street .view .year_built .year_updated .zillow_id .zipcode

3.3.2 pyzillow.pyzillowerrors module

exception pyzillow.pyzillowerrors.**ZillowError**(*status*, *url=None*, *response=None*)
Bases: Exception

A ZillowError exception is raised if the API endpoint responded with an error code (http://www.zillow.com/howto/api/GetDeepSearchResults.htm).

exception pyzillow.pyzillowerrors.**ZillowFail**
Bases: Exception

A ZillowFail exception is raised if the API endpoint could not be reached or the request did not return valid XML.

exception pyzillow.pyzillowerrors.**ZillowNoResults**
Bases: Exception

A ZillowNoResults exception is raised if the request did not return any results.

3.4 Contributing

Contributions are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

3.4.1 Types of contributions

Reporting bugs

Report bugs at https://github.com/hanneshapke/pyzillow/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fixing bugs

Fixes are always welcome! Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to fix it.

Implementing features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

Writing documentation

PyZillow could always use more documentation, whether as part of the official PyZillow docs, in docstrings, or even on the web in blog posts, articles, or tweets.

Submitting feedback

The best way to send feedback is to file an issue at https://github.com/hanneshapke/pyzillow/issues.

If you are proposing a feature:

- Explain in detail how the feature would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

3.4.2 Getting started

Ready to contribute? Here's how to set up PyZillow for local development.

Fork the *pyzillow* repo on GitHub.

Clone your fork locally:

```
$ git clone git@github.com:hanneshapke/pyzillow.git
```

Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Create a virtualenv to separate your Python dependencies:

```
$ virtualenv .pyzillow-env && source .pyzillow-env/bin/activate
```

Configure development requirements:

```
$ make develop
```

Now you can make your changes locally.

When you're done making changes, use Pytest to check that your changes pass style and unit tests, including testing other Python versions. You can run all tests by running pytest:

```
$ pytest
```

Please lint your code before committing your changes:

```
$ make lint
```

Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

3.4.3 Submitting a pull request

Check that your pull request meets these guidelines before you submit it:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs need to be updated. Include docstrings with your new functionality (Sphinx reStructuredText) and check if you need to update the information in the /docs/ folder.
3. The pull request should work with Python 3.6, 3.7 and 3.8. Make sure that all tests run by pytest pass.

3.4.4 Running a subset of tests

Use pytest in combination with a substring in case you want to run only specific tests instead of all available tests. pytest will only run tests with names matching the substring:

```
$ pytest -k <substring> -v
```

3.5 Credits

3.5.1 Maintainers

- Timo Cornelius Metzger <pyzillow@tcmetzger.net>
- Hannes Hapke <hannes@renooble.com>

3.5.2 Contributors

- Alvaro Feito <alvaro@renooble.com>
- Bryce Boe <bbzbryce@gmail.com>
- Hannes Hapke <hannes@renooble.com>
- Miguel Paolino <miguel@renooble.com>
- Timo Cornelius Metzger <pyzillow@tcmetzger.net>

3.6 Version history

3.6.1 0.7.0 (2020-05-30)

- Updated error handling, too many request error, Github issue 18
- Updated error handling, error 6 (Github issue 6)
- Pinned python-coveralls to latest version 2.9.3 (#27)
- Added posting details to GetUpdatedPropertyDetails result (#10)
- Updated pytest version (#32)
- Updated coverage version (#28)
- Added support for additional API fields (#16)

Thanks to Alexandra M. Chace (#16), Marilyn Chace, Evan Pete Walsh (#11), Stephen Holsapple (#10), ZAD-Man (Issue #6)

3.6.2 0.6.0 (2020-05-28)

- Updated tests, incl. complete API mocking
- Updated test dependencies
- Removed Python 2 support

3.6.3 0.5.0 (2015-09-12)

- Removed Django dependency, mocked tests, Python 3.4 support

3.6.4 0.4.0 (2014-12-20)

- Zestimate extracted from Zillow's GetDeepSearchResults API.

3.6.5 0.3.1 (2014-12-20)

- Added test cases and increased test coverage setup.

3.6.6 0.3.0 (2014-12-19)

- Refactored structure, travis CI compliance, coverage setup.

3.6.7 0.2.7

- Bug fix: Missing ParseError, numRooms now read from UpdatedProperty

3.6.8 0.2.6

- Bug fix

3.6.9 0.2.5

- Using markdown as README file for setup.py

3.6.10 0.2.4

- Coordinates provides as GEOS point

3.6.11 0.2.3

- New attributes added: home_description, num_floors, floor_material, parking_type

3.6.12 0.2.2

- Licence changed to MIT

3.6.13 0.2.1

- pip created and code refactured

3.6.14 0.2

- API Wrapper for the GetDeepSearchResults and GetUpdatedPropertyDetails API. test.py and setup.py created.

3.6.15 0.1

- Project created

Python Module Index

p

Index

G

P

Z