betterreads

Release 0.4.0

Apr 26, 2019

Contents:

This package provides a Python interface for the Goodreads API. Using it, you can do pretty much anything that Goodreads allows through their public API.

This package is largely Python 2 compatible, but is only officially supported for Python 3.

CHAPTER 1

Why BetterReads?

BetterReads is an expansion of the goodreads2 package available on PyPi. That package is no longer maintained and needed some updates to be usable. The name is just cheeky and cute, and not to imply that this project is substantially better than any other project. Someday I hope it's succeeded by a Python package called BestReads.

Major updates in this new project:

- Use https for Oauth - goodreads now requires this and previous packages' Oauth requests always fail
- Add convenience method to get all of a user's reviews for a specific shelf
- **Some opinionated development changes. For example**
  - No longer making live API calls in unit tests
  - Using black code style across the board
  - More robust documentation

CHAPTER 2

Dependencies

This package depends on the following packages:

- xmltodict
- requests
- rauth
- backports-datetime-fromisoformat

They can be installed using `pip`.

```
sudo pip install -r requirements.txt
```

If you want to contribute to this package, you will need to install the packages in `requirements-dev.txt` as well.

CHAPTER 3

Installation

To install, run the following command from the top-level package directory.

```
sudo python setup.py install
```

You can also install BetterReads using pip. The current version on PyPi is 0.4.1.

```
pip install betterreads
```

CHAPTER 4

Getting Started

The first thing is to request an API key from Goodreads here. Once you have it, you can create a client instance to query Goodreads.

```
from goodreads import client
gc = client.GoodreadsClient(<api_key>, <api_secret>)
```

To access some of the methods, you need OAuth for authorization.

```
gc.authenticate(<access_token>, <access_token_secret>)
```

Note that `access_token` and `access_token_secret` are different from developer key and secret. For the development step, you can call the same function with no parameters to get authorization. It will open a URL pointing a Goodreads page for OAuth permission. For your application, you can direct the user to that particular URL, ask them to authorize your app and save the returning `access_token` and `access_token_secret` in your database.

CHAPTER 5

Contribution

If you find an API method that is not supported by this package, feel free to create a Github issue. Also, you are more than welcome to submit a pull request for a bug fix or additional feature. For more detail on contributing to this project and setting up your local dev environment, check out our contribution guide.

CHAPTER 6

License

MIT License

CHAPTER 7

Acknowledgment

Thanks to Paul Shannon and Sefa Kilic for providing 'goodreads' package at PyPI, and to Tatiana and Rehan Khwaja for continuing the project as goodreads2. BetterReads couldn't exist without all of you.

7.1 Author

7.1.1 GoodreadsAuthor

GoodreadsAuthor is a BetterReads object for interfacing with author data from Goodreads.

Properties

- gid: Goodreads id of the author (type: int)
- name: Author's name (type: string)
- about: Goodreads "about the author" blurb (type: string)
- books: A list of GoodreadsBook objects for each of the author's credited works (type: list)
- born_at: Author's date of birth (type: datetime)
- died_at: Author's date of death (type: datetime)
- fans_count: Author's number of fans on Goodreads (type: int)
- gender: Author's gender (type: string)
- hometown: Author's hometown (type: string)
- link: Link to the author's Goodreads author page (type: string)
- image_url: Url for an author's featured image on their Goodreads page (type: string)
- small_image_url: Url for the small version of an author's featured image on their Goodreads page (type: string)

- influences: A list of other creators the author is influenced by (type: string)
- user: A GoodreadsUser object for the author's Goodreads user profile (type: GoodreadsUser)
- works_count: A count of the works an author is credited on (type: int)

Usage

You can query Author information from the GoodreadsClient by searching by id.

```
>>> from betterreads.client import GoodreadsClient

>>> gc = GoodreadsClient('GOODREADS_API_KEY', 'GOODREADS_SECRET')
>>> author = gc.author(2617)
>>> author.name
u'Jonathan Safran Foer'
>>> author.works_count
13
>>> author.books
[Extremely Loud and Incredibly Close, Everything Is Illuminated, ...]
```

7.2 Book

7.2.1 GoodreadsBook

A GoodreadsBook object represents the Goodreads concept of a book. It combines two concepts important to the Goodreads API. First, is the book itself. Books have page numbers, one specific edition, one specific publisher, etc.

Books belong to a "Work" that contains aggregate information for all editions of the book. Works have attributes like average rating, or editions.

Properties

- gid: Goodreads id of the book (type: int)
- title: Title of the book (type: string)
- authors: List of GoodreadsAuthor objects for authors credited to the work (type: list)
- description: Description of the book (type: string)
- average_rating: Mean average star rating given to the work by Goodreads users. This is a float value between 1 and 5. (type: float)
- rating_dist: Ratings distribution for the work (type: string)
- ratings_count: Number of ratings given to the work (type: int)
- text_reviews_count: Number of text reviews left for the work (type: int)
- num_pages: Number of pages in the book (type: int)
- popular_shelves: Dict of shelf names and counts that represent the number of times this work appears on a shelf named *name* on Goodreads (type: dict)
- work: Information on the book's "original work" (type: dict)
- series_works: Returns information about related works in the same series (type: dict)

- publication_date: Publication date for the book (not necessarily the original publication date) (type: datetime)
- publisher: Published listed for the book (not necessarily the original publisher) (type: string)
- language_code: Language code for the book (not necessarily the original language) (type: string)
- edition_information: Information about this specific edition of the work (type: string)
- image_url: Url to the book's cover image (type: string)
- small_image_url: Url to a smaller version of the book's cover image (type: string)
- is_ebook: `True` or `False` value for whether this book is an ebook (type: boolean)
- **format: format of the book (type: string)**
  - Example: "HARDCOVER"
- isbn: ISBN-10 of the book (type: string)
- isbn13: ISBN-13 of the book (type: string)
- link: link to the book's goodreads page (type: string)
- reviews_widget: widget for reviews in HTML (type: string)
- similar_books: List of GoodreadsBook objects of books similar to this one (type: list)

Usage

You can query Book information from the GoodreadsClient by searching by id.

```
>>> from betterreads.client import GoodreadsClient

>>> gc = GoodreadsClient('GOODREADS_API_KEY', 'GOODREADS_SECRET')
>>> book = gc.book(1)

>>> book.title
u'Harry Potter and the Half-Blood Prince (Harry Potter, #6)'
>>> authors = book.authors
>>> authors[0].name
u'J.K. Rowling'
>>> book.average_rating
4.49
```

7.3 Client

7.3.1 GoodreadsClient

The GoodreadsClient is the wrapper that powers direct queries to the Goodreads API.

Fields

- client_key: Application's Goodreads API Client Key
- client_secret: Application's Goodreads API Client Secret

Properties

- query_dict: A dict containing a GoodreadsClient's client_key

Key Functions and Usage

Instantiate Client

```
>>> from betterreads.client import client
>>> gc = client.GoodreadsClient(<api_key>, <api_secret>)
```

`author`

Get information about an author, querying by Goodreads author id.

params:

- author_id (int)

return: GoodreadsAuthor object

```
>>> author = gc.author(2618)
>>> author.name
u'Jonathan Safran Foer'
```

`find_author`

Get information about an author querying by name.

params:

- author_name (string)

return: GoodreadsAuthor object or `None`

```
>>> author = gc.find_author("Stephen King")
>>> author.gid
'3389'
```

`book`

Get information about a book querying by Goodreads book id or ISBN.

params:

- book_id (int) (optional)
- book_isbn (int or string) (optional)

return: GoodreadsBook object

```
>>> book = gc.book(123)
>>> book.title
u'The Power of One (The Power of One, #1)'
>>> book.num_pages
'291'
```

search_books

Get the most popular books for a given query. Searches title, author, and genre.

params:

- q (string)
- page (int, deftault=1)
- **search_field (string, default="all")**
  - valid options: ['title', 'author', 'genre', 'all']

return: List of GoodreadsBook objects.

return: list of matching GoodreadsBook objects

```
>>> books = gc.search_books("Shakespeare King")
>>> print(books)
[King Lear,
 Hamlet,
 Manga Shakespeare: King Lear,
 Macbeth,
 A Midsummer Night's Dream,
 Julius Caesar,
 The Merchant of Venice,
 Twelfth Night,
 King Henry IV, Part 1,
 Shakespeare's King Lear (Cliffs Notes),
 The Complete Works of Shakespeare,
 Richard III,
 Henry V,
 Prefaces to Shakespeare: King Lear,
 Four Great Tragedies: Hamlet / Othello / King Lear / Macbeth,
 The Winter's Tale,
 The Comedy of Errors,
 William Shakespeare, "King Lear",
 King Richard II (The Arden Shakespeare),
 Shakespeare's Ovid: Being Arthur Golding's Translation Of The Metamorphoses]
```

search_books_total_pages

Get the total number of pages for a book search. Accepts text for the query param and searches title, author, and genre.

params:

- q (string)
- page (int, default=1)
- **search_field (string, default='all')**

 - valid options: ['title', 'author', 'genre', 'all']

return: integer number of results pages for the query string

```
>>> gc.search_books_total_pages("Shakespeare King")
41
```

search_books_all_pages

Get all the books for a given query. This will return all books where the title/author/genre fields show matches. Sorted by popularity on Goodreads. Note that if you use a broad search term this operation could take a while. Also bear in mind that the Goodreads API terms and conditions limit each application to 1 request/second per endpoint. This client function does not throttle requests.

params:

- q (string)
- page (int, default=1)
- **search_field (string, default='all')**
  - valid options ['title', 'author', 'genre', 'all']

return: List of GoodreadsBook objects

```
>>> books = gc.search_books_all_pages("Demon in my View", search_field="title")
>>> print(books)
[Demon in My View,
 A Demon in My View,
 Demon in My View,
 Of a Demon in My View,
 Of a Demon in My View,
 A Demon in My View,
 The Tree of Hands / A Demon in My View,
 A Demon In My View,
 The face of trespass: A judgement in stone ; A demon in my view,
 A Demon in My View (Prose series) (Prose series),
 A Demon in My View (Prose Series 68),
 Novels by Amelia Atwater-Rhodes: Falcondance, Demon in My View, Hawksong, Wyvernhail,
↪ Snakecharm, Shattered Mirror, Midnight Predator,
 Nyeusigrube: The Kiesha'ra Series, Amelia Atwater-Rhodes, Demon in My View,␣
↪Shattered Mirror, in the Forests of the Night,
 Articles on Nyeusigrube, Including: In the Forests of the Night, Demon in My View,␣
↪Shattered Mirror, Midnight Predator, Hawksong, Snakecharm, Falcondance, Wyvernhail,␣
↪Zane Cobriana, Amelia Atwater-Rhodes, the Kiesha'ra Series,
 The Ruth Rendell Omnibus: "Face of Trespass", "Judgement in Stone", "Demon in My View
↪" v. 1]
```

group

Get information about a group. Queries the Goodreads API by group id.

params:

- group_id (int or string)

return: GoodreadsGroup

```
>>> gc.group(8095)
u'Goodreads Developers'
```

owned_book

Get info about an owned book. Queries the Goodreads API by id. This method requires user authentication.

params:

- owned_book_id (int or string)

return: GoodreadsOwnedBook object

find_groups

Find groups based based on a text query.

params:

- query (string)
- page (int, default=1)

return: List of OrderedDicts

request

create a GoodreadsRequest object and make a request to the Goodreads API.

user

Get information on a goodreads user, querying either by Goodreads id or username. Returns a GoodreadsUser object.

```
>>> user = gc.user(user_id=12345)
>>> user.name
u'Example McTesterson'

>>> user = gc.user(username="test_username")
>>> user.name
u'Test Name Person III'
```

7.3.2 GoodreadsClientException

An `Exception` that is raised when the `GoodreadsClient` encounters an error executing a request

7.4 Comment

7.4.1 GoodreadsComment

GoodreadsComment is a BetterReads object for interfacing with comment data from Goodreads.

7.4.2 Properties

- gid: Goodreads id of the comment (type: int)
- body: Text body of the comment (type: string)
- user: GoodreadsUser that authored the comment (type: GoodreadsUser)
- created_at: String representation of the date when the comment was created (type: datetime)
- updated_at: String representation of the date when the comment was updated (type: datetime)

7.4.3 Usage

You can query Comments on a resource from the GoodreadsClient by searching by id.

```
>>> from betterreads.client import GoodreadsClient

>>> gc = GoodreadsClient('GOODREADS_API_KEY', 'GOODREADS_SECRET')
>>> comments = gc.list_comments("review", 123456)
>>> comment = comments[0]
>>> comment.body
u'This was a really good review'
```

When querying for a comment, you must specify which type of resource you're grabbing comments for. Valid resource types are:

- `author_blog_post`
- `blog`
- `book_news_post`
- `chapter`
- `comment`
- `community_answer`
- `event_response`
- `fanship`
- `friend`
- `giveaway`
- `giveaway_request`
- `group_user`
- `interview`
- `librarian_note`
- `link_collection`
- `list`
- `owned_book`
- `photo`
- `poll`
- `poll_vote`

- `queued_item`
- `question`
- `question_user_stat`
- `quiz`
- `quiz_score`
- `rating`
- `read_status`
- `recommendation`
- `recommendation_request`
- `review`
- `topic`
- `user`
- `user_challenge`
- `user_following`
- `user_list_challenge`
- `user_list_vote`
- `user_quote`
- `user_status`
- `video`

7.5 Event

7.5.1 GoodreadsEvent

GoodreadsEvent is a BetterReads object for interfacing with event data from Goodreads.

7.5.2 Properties

- gid: Goodreads id for the event (type: int)
- title: Event's title (type: string)
- description: event's description (type: string)
- link: Link to the event on Goodreads (type: string)
- venue: Venue information for the event (type: string)
- address: Street address of the event (type: string)
- city: City where the event takes place (type: string)
- postal_code: Postal code for the event (type: string)
- state_code: The state code of the event (type: string)
- country_code: The country code for the event (type: string)

- access: indicates whether the event is public or private (type: string)
- **event_type: indicates the type of event (type: string)**
  - Example: author reading
- added_by: user id for who added the event (type: int)
- image_url: link to the event image (type: string)
- created_at: String representation of the event's created time (type: datetime)
- updated_at: String representation of the event's updated time (type: datetime)
- reminder_at: String representation of the time that attendees will be reminded about the event (type: datetime)
- rsvp_end_at: String representation of the time that RSVPs will close (type: datetime)
- start_at: String representation of the time the event begins (type: datetime)
- end_at: String representation of the time the event ends (type: datetime)
- attending_count: Number of users who have RSVPd (type: int)
- responses_count: Number of responses (type: int)
- **resource: Resource that the event is supporting (type: tuple)**
  - Example: author or book that the event is promoting

7.5.3 Usage

You can query Events within or near a zip code with the GoodreadsClient by searching by zip.

```
>>> from betterreads.client import GoodreadsClient

>>> gc = GoodreadsClient('GOODREADS_API_KEY', 'GOODREADS_SECRET')
>>> events = gc.list_events(80126)
>>> events[0]
>>> event.title
u'Meet and Greet with Mary Sue'
>>> event.description
u'Learn all about Mary Sue and her epic adventures!'
```

7.6 Group

7.6.1 GoodreadsGroup

GoodreadsGroup is a BetterReads object for interfacing with group data from Goodreads.

7.6.2 Properties

- gid: Goodreads id for the Group (type: int)
- title: Title of the group (type: string)
- description: Short text description of the group (type: string)
- **category: Text category for filtering groups (type: string)**

  - Example: Organizations

- **subcategory: Text category for further filtering groups (type: string)**
  - Example: Companies
- rules: Text description of the rules for participating in the group (type: string)
- image_url: URL for the group's featured image (type: string)
- last_activity_at: String representation of the time and date of the group's last activity (type: datetime)
- access: Indicates whether a group is public or private (type: string)
- users_count: Number of users in the group (type: int)
- members: List of dicts containing information about group members. Member ids can be used to query GoodreadsUsers using the GoodreadsClient (type: list)

7.6.3 Usage

You can query Group information from the GoodreadsClient by searching by id.

```
>>> g = gc.find_groups("Python")
>>> g = groups[0]
>>> g['title']
u'The Computer Scientists'
>>> group = gc.group(g['id'])
>>> group.description
u'Only for Committed Self Learners and Computer Scientists Who are Starving for
Information, and Want to Advance their Skills Through: Reading, Practicing and
Discussion Computer Science and Programming Books.'
```

7.7 Owned Book

7.7.1 GoodreadsOwnedBook

GoodreadsOwnedBook is the BetterReads representation of the Goodreads concept of an "owned book."

An owned book is a representation of a physical copy of a book that is owned by a person or organization.

7.7.2 Properties

- gid: Goodreads id for the owned book (type: int)
- book: GoodreadsBook object for the book that is "owned" (type: GoodreadsBook)
- review: GoodreadsReview associated with the book (type: GoodreadsReview)
- current_owner: User id of the book's current owner. Can be used to query GoodreadsUser data. (type: int)
- original_purchase_date: String representation of the date that the book was purchased (type: datetime)
- original_purchase_location: String for where the owner purchased the book (type: string)
- condition: Condition of the book (ex: Brand new) (type: string)
- link: Link to the Owned Book record on Goodreads (type: string)

7.7.3 Usage

You can query for an owned book using the GoodreadsClient. You must have an authenticated user session in order to query for owned books.

7.8 Request

7.8.1 GoodreadsRequest

Betterreads interface object for making requests to the Goodreads API.

7.8.2 GoodreadsRequestException

Betterreads exception for when the Goodreads API returns an unexpected status code.

7.9 Review

7.9.1 GoodreadsReview

Betterreads interface for interacting with Goodreads review data.

7.9.2 Properties

- gid: Goodreads review id (type: int)
- book: Dict with information on the book the review belongs to (type: dict)
- rating: Review rating from 1 - 5 "stars" (type: int)
- shelves: Most common shelf names that contain this book (type: list)
- recommended_for: dict of users recommended for (type: dict)
- recommended_by: dict of users recommended by (type: dict)
- started_at: String representation of the date the user started reading the book (type: datetime)
- read_at: String representation of the date the user finished reading the book (type: datetime)
- body: Text body of the review (type: string)
- comments_count: Number of comments on the review (type: int)
- url: Link to the review on Goodreads (type: string)
- owned: boolean value (expressed as "0" or "1") whether the user owns this book (type: boolean)

7.9.3 Usage

You can query Reviews by id using the GoodreadsClient.

```
>>> from betterreads.client import GoodreadsClient

>>> gc = GoodreadsClient('GOODREADS_API_KEY', 'GOODREADS_SECRET')
>>> review = gc.review(1)
>>> review.body
u'I was lukewarm on this book'
>>> review.owned
False
```

7.10 Session

7.10.1 GoodreadsSession

GoodreadsSession is the class that controls the flow of authenticating a user so that the GoodreadsClient can make requests requiring user auth.

7.11 Shelf

7.11.1 GoodreadsUserShelf

`GoodreadsUserShelf` is a class interface for the Goodreads concept of shelf. Shelves are collections of books that belong to users.

7.11.2 Properties

- gid: Goodreads id of the shelf (type: int)
- name: Name of the shelf (type: string)
- exclusive: boolean flag indicating whether this shelf is mutually exclusive with other exclusive shelves (type: boolean)
- count: number of books on a shelf (type: integer)
- sticky: boolean flag indicating whether this shelf is "sticky" to the top of the list in the Goodreads UI (type: boolean)
- description: text description of the shelf (type: string)
- featured: boolean flag indicating whether this is a GoodreadsUser's featured shelf. (type: boolean)

7.11.3 Usage

There is no method available for querying a shelf individually. Shelf data is obtained when querying for a GoodreadsUser

7.12 User

7.12.1 GoodreadsUser

GoodreadsUser is a BetterReads object for interfacing with user data from Goodreads.

7.12.2 Properties

- gid: Goodreads user id (type: int)
- user_name: Goodreads handle for the user (type: string)
- name: User's name (type: string)
- link: URL for the user profile on Goodreads (type: string)
- image_url: URL for the user's profile image (type: string)
- small_image_url: URL for a smaller version of the user's profile image (type: string)

7.12.3 Key Functions and Usage

Get a user using the GoodreadsClient

```
>>> from betterreads.client import GoodreadsClient
>>> gc = GoodreadsClient(os.environ.get("GOODREADS_KEY"), os.environ.get("GOODREADS_
↪SECRET"))
>>> user = gc.user(user_id=12345)
>>> user.name
u'Example McTesterson'
>>> user.user_name
u'FlyMcTesterson63'
```

`list_groups`

Get a list of all the GoodreadsGroups to which a user belongs

params:

- page (int, default=1)

return: list of GoodreadsGroup objects

`owned_books`

Get a list of GoodreadsOwnedBook belonging to a user

params:

- page (int, default=1)

return: list of GoodreadsOwnedBook objects

`reviews`

Get a list of a user's GoodreadsReview

params:

- page (int, default=1)

return: list of GoodreadsReview objects

`shelves`

Get a list of user's GoodreadsUserShelf objects

params:

- page (int, default=1)

return: list of GoodreadsUserShelf objects

`per_shelf_reviews`

Get a list of GoodreadsBook objects belonging to a particular GoodreadsUserShelf

params:

- page (int, default=1)
- per_page (int, default=200)
- shelf_name (string, default="read)

return: list of GoodreadsReview objects

7.13 Contribution Guide

7.13.1 Submitting an Issue

If you notice something is broken, a feature is missing, or you have a request to make the project better, please feel free to submit an issue via the issue tracker. You do not need to have a solution or commit to fixing the problem in order to create an issue. However, if you would like to contribute to the project, we would love to have your contribution!

7.13.2 Fork the Repo and Create a Branch

To contribute to this project, first make a fork of the repo.

Then create a branch on your local fork. This branch should be based off of the `develop` branch, NOT the `master` branch. Active development is merged into the `develop` branch and `develop` is merged into `master` when there is a new release or a hotfix to be released on PyPi.

If your change corresponds to an issue your branch name should start with to that issue number. Your name branch should be descriptive.

Example: `13-add-delete-to-owned-book`

Once you've made your changes and the unit test suite is passing, create a pull request. Make sure that your pull request is pointing to `develop` and not `master`.

7.13.3 Pre-Commit

This project uses pre-commit hooks to ensure consistent code style throughout the repo. We use black for Python files and Python code within documentation. We use prettier for all other filetypes.

Make sure you've installed all the packages listed in both `requirements.txt` and `requirements-dev.txt`. This will install pre-commit for you. Then run `pre-commit install` to set up the local pre-commit environment.

Pre-commit will run each time you attempt to commit staged changes. You can run the pre-commit checks at any time using `pre-commit run`.

7.13.4 Running Tests

You will not need your own developer keys to run the unit tests. However, you will need developer keys for the Goodreads API in order to run the integration test suite.

Any changes you make likely shouldn't impact the integration tests, but if for some reason you do need to adjust them and run them, set your developer keys as `GOODREADS_KEY` and `GOODREADS_SECRET` environment variables.

You can obtain a Goodreads developer key here.

To run the test suite, make sure you've installed the packages listed in `requirements.txt` and `requirements-dev.txt`. Then run `pytest --cov=betterreads`

Pull requests that cause the repository's overall test coverage to drop below 85% or cause a decrease in coverage of %5 or more will be rejected. Please make sure to update tests in accordance with your changes.

7.13.5 Community Standards

In general, PRs will be acknowledged within one week of receipt. I wish I could say that they would all be reviewed and merge in in this time frame, but sometimes life gets the better of us. I'll do my best.

All contributions and discussions in this repo should abide by the Code of Conduct.

- search