

---

# Letterboxd Documentation

*Release 0.3.0*

**Mark Boszko**

**May 04, 2020**



---

## Contents:

---

<b>1</b>	<b>Letterboxd</b>	<b>3</b>
1.1	Letterboxd API Access . . . . .	3
1.2	Acknowledgements . . . . .	4
<b>2</b>	<b>Installation</b>	<b>5</b>
2.1	Stable release . . . . .	5
2.2	From sources . . . . .	5
<b>3</b>	<b>Usage</b>	<b>7</b>
<b>4</b>	<b>Modules reference</b>	<b>9</b>
<b>5</b>	<b>letterboxd package reference</b>	<b>11</b>
5.1	letterboxd module . . . . .	11
5.2	api module . . . . .	12
5.3	user module . . . . .	12
5.4	config module . . . . .	13
<b>6</b>	<b>services package reference</b>	<b>15</b>
6.1	auth module . . . . .	15
6.2	film module . . . . .	16
6.3	list module . . . . .	18
6.4	member module . . . . .	20
6.5	search module . . . . .	21
<b>7</b>	<b>Contributing</b>	<b>23</b>
7.1	Types of Contributions . . . . .	23
7.2	Get Started! . . . . .	24
7.3	Pull Request Guidelines . . . . .	25
7.4	Tips . . . . .	25
7.5	Deploying . . . . .	25
<b>8</b>	<b>Credits</b>	<b>27</b>
8.1	Development Lead . . . . .	27
8.2	Contributors . . . . .	27
<b>9</b>	<b>Changelog</b>	<b>29</b>
9.1	[Unreleased] . . . . .	29

9.2	[0.3.0] - 2018-07-22	29
9.3	[0.2.6] - 2018-07-04	30
9.4	[0.2.5] - 2018-07-04 [YANKED]	30
9.5	[0.2.4] - 2018-07-04	30
9.6	[0.2.3] - 2018-07-04 [YANKED]	31
9.7	[0.2.2] - 2018-07-04 [YANKED]	31
9.8	[0.2.1] - 2018-07-04 [YANKED]	31
9.9	[0.2.0] - 2018-07-04	31
9.10	[0.1.0] - 2018-06-24	31
9.11	Changelog format	32
<b>10</b>	<b>Indices and tables</b>	<b>33</b>
	<b>Python Module Index</b>	<b>35</b>
	<b>Index</b>	<b>37</b>

This software is very much in an alpha state, and things that are working now may break in the future. Developer beware.



Python 3 implementation of the Letterboxd API v0.

- PyPI package: <https://pypi.org/project/letterboxd/>
- GitHub repo: <https://github.com/bobtiki/letterboxd>
- Documentation: <https://letterboxd.readthedocs.io>
- Free software: MIT license

Python 3.6 is required.

**Warning: THIS PROJECT IS CURRENTLY IN ALPHA:**

- It may be broken.
- What is working now may break between now and v1.0
- Initial focus is on implementing endpoints related to retrieving watchlists and other lists for users.

## 1.1 Letterboxd API Access

Letterboxd has posted an [example Ruby client](#), but as they say in the readme there:

Although the Letterboxd API isn't public yet (as at 2017-06-12), we have seeded some developers with API keys.

If you need more information about API access, please see <https://letterboxd.com/api-coming-soon/>.

## 1.2 Acknowledgements

Thanks to Kevin Ndung'u Gathuku of Semaphore for [this article on writing and testing an API wrapper](#) from which I learned many of the techniques I used to build this wrapper.

And of course, many thanks to Matthew Buchanan and Grant Berridge at Letterboxd for allowing me beta access to the Letterboxd API, and providing helpful assistance when I stumbled.



## 2.1 Stable release

To install Letterboxd, run this command in your terminal:

```
$ pip install letterboxd
```

This is the preferred method to install Letterboxd, as it will always install the most recent stable release.

If you don't have `pip` installed, this [Python installation guide](#) can guide you through the process.

## 2.2 From sources

The sources for Letterboxd can be downloaded from the [Github repo](#).

You can either clone the public repository:

```
$ git clone git://github.com/bobtiki/letterboxd
```

Or download the [tarball](#):

```
$ curl -OL https://github.com/bobtiki/letterboxd/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```



To use Letterboxd in a project:

```
import letterboxd
lboxd = letterboxd.new()
```

You can pass the `api_base` URL for the API, your `api_key`, and your `api_secret` as arguments in `.new()`, or add it to your environment variables as:

```
export LBXD_API_KEY="YOUR_KEY_HERE"
export LBXD_API_SECRET="YOUR_SECRET_HERE"
```

If you do not provide an `api_base`, it currently defaults to `https://api.letterboxd.com/api/v0`.

For testing, you can also add environment variables for an account user name and password, and the location of your Charles proxy SSL certificate.

```
export LBXD_USERNAME="YOUR_USERNAME"
export LBXD_PASSWORD="YOUR_PASSWORD"
export CHARLES_CERTIFICATE="/path/to/charles-ssl-proxying-certificate.pem"
```

Set the `CHARLES` environment variable to `True` to turn the proxy settings on. For example, if you want to use it on a case-by-case basis for testing, run the tests like this:

```
$ CHARLES="True" pytest
```

Please see [Modules](#) for more details on usage of the classes.



## CHAPTER 4

---

Modules reference

---



## 5.1 letterboxd module

Python 3 wrapper for Version 0 of the Letterboxd API

```
class letterboxd.letterboxd.Letterboxd (api_base='https://api.letterboxd.com/api/v0',  
                                         api_key="", api_secret="")
```

Loads the API base URL, API key, and API shared secret, and connects with all of the other classes.

If the key and secret are not passed as arguments, it looks for them as environment variables, as LBXD\_API\_KEY and LBXD\_API\_SECRET.

```
auth ()
```

**Returns** services.auth.Authentication object

```
film (film_id)
```

**Parameters** **film\_id** – str - the LID of a film on Letterboxd

**Returns** services.film.Film object

```
film_collection (film_collection_id, film_collection_request)  
/film-collection/{id}
```

Get details about a film collection by ID. The response will include the film relationships for the signed-in member and the member indicated by the member LID if specified.

**Parameters**

- **film\_collection\_id** – str - LID of the FilmCollection
- **film\_collection\_request** – dict - FilmCollectionRequest

**Returns** dict - FilmCollection

```
films ()
```

**Returns** services.film.Films object

**list** (*list\_id*)

**Parameters** **list\_id** – str - the LID of a list on Letterboxd

**Returns** services.list.List object

**member** (*member\_id*)

**Parameters** **member\_id** – str - LID for Letterboxd member

**Returns** services.member.Member object

**search** (*search\_request*)

/search

**Parameters** **search\_request** – dict - SearchRequest

**Returns** dict - SearchResponse

**user** (*username, password*)

Signs in the user, and adds the oAuth token to future API calls

**Parameters**

- **username** – str
- **password** – str

**Returns** user.User object

## 5.2 api module

**class** letterboxd.api.**API** (*api\_base, api\_key, api\_secret*)

Communication methods for the Letterboxd API

**api\_call** (*path, params={}, form=None, headers={}, method='get'*)

The workhorse method of calls to the Letterboxd API

**Parameters**

- **path** – str - URL endpoint path for the desired service
- **params** – dict - request parameters
- **form** – str - form information, likely from the auth.py call
- **headers** – dict - request parameters
- **method** – str - HTML methods, [get, post, put, patch, delete]

**Returns** requests.Response object

## 5.3 user module

User-based features of the Letterboxd API

**class** letterboxd.user.**User** (*api, username, password*)

Provides access token and shortcuts to user-focused methods



**me**

/me

Get details about the authenticated member.

Calls to this endpoint must include the access token for an authenticated member.

**Returns** dict - MemberAccount

**me\_update** (*member\_settings\_update\_request*)

/me

Update the profile settings for the authenticated member.

Calls to this endpoint must include the access token for an authenticated member

**Parameters** **member\_settings\_update\_request** – dict - MemberSettingsUpdateRequest

**Returns** dict - MemberSettingsUpdateResponse

**token**

Ask services.auth to get a token, and return the token string

**Returns** str - OAuth token

## 5.4 config module

Configuration

Constants for package use.



## 6.1 auth module

User authentication services for the Letterboxd API

Authentication API Documentation: <http://api-docs.letterbotokend.com/#auth>

**class** `letterboxd.services.auth.Authentication` (*api, username, password*)

User authentication services for Letterboxd

This token business mostly takes care of itself. Instantiate authentication with username and password, then call `token()`, and if there isn't a token already, or if it's expired, it will go and get one.

**static** `forgotten_password_request` (*api, forgotten\_password\_request*)  
`/auth/forgotten-password-request`

Request a link via email to reset the password for a member's account.

**Request** `forgotten_password_request` - `ForgottenPasswordRequest`

**Returns** `int` - HTTP status code

**login** (*username, password*)

User access to the Letterboxd API. Requests a token for the user.

**Parameters**

- **username** – str
- **password** – str

**Returns** `dict` - either an `AccessToken` or `OAuthError`

**refresh\_token** ()

Uses the current single-use `refresh_token` to request a new access token for the user

**Returns** `dict` - either an `AccessToken` or `OAuthError`

**token**

Checks if the user authentication token already exists. If not, it tries to get one. If it does exist, it checks to see if it is expired, and if so, it attempts to refresh the token.

**Returns** str - user token

## 6.2 film module

**class** letterboxd.services.film.**Film** (*api, film\_id=None*)  
/film/\* services for the Letterboxd API

**availability** (*film\_id=None*)  
/film/{id}/availability

Get availability data about a film by ID. If no film ID passed, uses the initialized film.

NOTE: This data is currently available to first-party only.

**Parameters** **film\_id** – str - LID of the film

**Returns** dict - FilmAvailabilityResponse

**details** (*film\_id=None*)  
/film/{id}

Get details about a film by ID. If no film ID passed, uses the initialized film.

**Parameters** **film\_id** – str - LID of the film

**Returns** dict - Film

**me** (*film\_id=None*)  
/film/{id}/me

Get details of the authenticated member's relationship with a film by ID. If no film ID passed, uses the initialized film.

**Parameters** **film\_id** – str - LID of the film

**Returns** dict - FilmRelationship

**me\_update** (*film\_relationship\_update\_request, film\_id=None*)  
/film/{id}/me [PATCH]

Update the authenticated member's relationship with a film by ID.

Calls to this endpoint must include the access token for an authenticated member

**Parameters**

- **film\_id** – str - LID of the film
- **film\_relationship\_update\_request** – dict - FilmRelationshipUpdateRequest

**Returns** dict - FilmRelationshipUpdateResponse

**members** (*film\_id=None, member\_film\_relationships\_request=None*)  
/film/{id}/members

Get details of members' relationships with a film by ID. If no film ID passed, uses the initialized film.

**Parameters**

- **film\_id** – str - LID of the film

- **member\_film\_relationships\_request** – dict - MemberFilmRelationshipsRequest

**Returns** dict - MemberFilmRelationshipsResponse

**report** (*film\_id=None, report\_film\_request=None*)  
/film/{id}/report

Report problems with a film by ID. Does NOT default to the initialized Film instance LID, so as to not submit unnecessary reports.

**Parameters**

- **film\_id** – str - the LID of the film
- **report\_film\_request** – dict - ReportFilmRequest

**Returns** requests.Response.status\_code

**statistics** (*film\_id=None*)  
/film/{id}/statistics

Get statistical data about a film by ID.

**Parameters** **film\_id** – str - the LID of the film

**Returns** dict - FilmStatistics

**class** letterboxd.services.film.**FilmCollection** (*api*)  
/film-collection service for the Letterboxd API

**film\_collection** (*film\_collection\_id=None, film\_collection\_request=None*)  
/film-collection/{id}

Get details about a film collection by ID. The response will include the film relationships for the signed-in member and the member indicated by the member LID if specified.

**Parameters**

- **film\_collection\_id** – str - The LID of the film collection.
- **film\_collection\_request** – dict - FilmCollectionRequest

**Returns** dict - FilmCollection

**class** letterboxd.services.film.**Films** (*api*)  
/films/\* services for the Letterboxd API

**films** (*films\_request=None*)  
/films

A cursored window over the list of films.

Use the ‘next’ cursor to move through the list. The response will include the film relationships for the signed-in member and the member indicated by the member LID if specified.

**Parameters** **films\_request** – dict - FilmsRequest

**Returns** dict

**genres** ()  
/films/genres

Get a list of genres supported by the /films endpoint.

Genres are returned in alphabetical order.

**Returns** dict - GenresResponse

**services** ()

/films/film-services

Get a list of services supported by the /films endpoint.

Services are returned in alphabetical order. Some services are only available to paying members, so results will vary based on the authenticated member's status.

**Returns** dict - FilmServicesResponse

## 6.3 list module

**class** letterboxd.services.list.**List** (*api, list\_id=None*)

/list/\* services for the Letterboxd API

**comments** (*list\_id=None, comments\_request=None*)

/list/{id}/comments

A cursored window over the comments for a list. Use the 'next' cursor to move through the comments.

### Parameters

- **list\_id** – str - LID of the list
- **comments\_request** – dict - CommentsRequest

### Returns

**create\_comment** (*list\_id=None, comment\_creation\_request=None*)

/list/{id}/comments

Create a comment on a list.

Calls to this endpoint must include the access token for an authenticated member.

### Parameters

- **list\_id** – str - LID for the list
- **comment\_creation\_request** – dict - CommentCreationRequest

**Returns** dict - ListComment

**delete** (*list\_id=None*)

/list/{id}

Delete a list by ID. Does NOT default to the initialized List instance LID, so as to not accidentally delete the instance list.

Calls to this endpoint must include the access token for an authenticated member, and the authenticated member must be the owner of the specified list.

**Parameters** **list\_id** – str - LID of the list

**Returns** bool - Success

**details** (*list\_id=None*)

/list/{id}

Get details of a list by LID. If no list ID passed, uses the initialized list.

**Parameters** **list\_id** – str - LID of the film

**Returns** dict - Film

**entries** (*list\_id=None, list\_entries\_request=None*)  
/list/{id}/entries

Get entries for a list by ID.

**Parameters**

- **list\_id** – str - LID of the list
- **list\_entries\_request** – dict - ListEntriesRequest

**Returns** dict - ListEntriesResponse

**me** (*list\_id=None*)  
/list/{id}/me

Get details of the authenticated member's relationship with a list by ID.

Calls to this endpoint must include the access token for an authenticated member.

**Parameters** **list\_id** – str - LID of the list

**Returns** dict - ListRelationship

**me\_update** (*list\_id=None, list\_relationship\_update\_request=None*)  
/list/{id}/me

Update the authenticated member's relationship with a list by ID.

Calls to this endpoint must include the access token for an authenticated member.

**Parameters**

- **list\_id** – str - LID for the list
- **list\_relationship\_update\_request** – dict - ListRelationshipUpdateRequest

**Returns** dict - ListRelationshipUpdateResponse

**report** (*list\_id=None, report\_list\_request=None*)  
/list/{id}/report

Report a list by ID. Does NOT default to the initialized List instance LID, so as to not submit erroneous reports.

Calls to this endpoint must include the access token for an authenticated member.

**Parameters**

- **list\_id** – str - LID of the list
- **report\_list\_request** – dict - ReportListRequest

**Returns** bool - Success

**statistics** (*list\_id=None*)  
/list/{id}/statistics

Get statistical data about a list by ID.

**Parameters** **list\_id** – str - LID of the list

**Returns** dict - ListStatistics

**update** (*list\_id=None, list\_update\_request=None*)  
/list/{id} [PATCH]

Update a list by ID.

Calls to this endpoint must include the access token for an authenticated member.

**Parameters**

- **list\_id** – str - LID of the list
- **list\_update\_request** – dict - ListUpdateRequest

**Returns** dict - ListUpdateResponse

```
class letterboxd.services.list.Lists (api)  
/lists service for the Letterboxd API
```

```
create_list (list_creation_request=None)  
[POST] /lists
```

Create a list.

Calls to this endpoint must include the access token for an authenticated member.

**Parameters** **list\_creation\_request** – dict - ListCreationRequest

**Returns** dict - ListCreateResponse

```
lists (lists_request=None)  
[GET] /lists
```

A cursored window over a list of lists.

Use the ‘next’ cursor to move through the list.

**Parameters** **lists\_request** – dict - ListsRequest

**Returns** dict - ListsResponse

## 6.4 member module

```
class letterboxd.services.member.Member (api, member_id=None)  
/member/* services for the Letterboxd API
```

```
details (member_id=None)  
/member/{id}
```

Get details about a member by ID.

# TODO: Write this function

**Parameters** **member\_id** – str - The LID of the member.

**Returns** dict - Member

```
watchlist (member_id=None, watchlist_request=None)  
/member/{id}/watchlist
```

Get details of a member’s public watchlist by ID.

The response will include the film relationships for the signed-in member, the watchlist’s owner, and the member indicated by the member LID if specified (the member and memberRelationship parameters are optional, and can be used to perform comparisons between the watchlist owner and another member). Use the /film/{id}/me endpoint to add or remove films from a member’s watchlist.

**Parameters**

- **member\_id** – str - The LID of the member.



- **watchlist\_request** – dict - WatchlistRequest

**Returns** dict - FilmsResponse

## 6.5 search module

**class** letterboxd.services.search.**Search** (*api*)

*/search for the Letterboxd API*

**search** (*search\_request=None*)

*/search*

**Parameters** **search\_request** – dict - SearchRequest

**Returns** dict - SearchResponse



Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given. You can contribute in many ways:

## 7.1 Types of Contributions

### 7.1.1 Report Bugs

Report bugs at <https://github.com/bobtiki/letterboxd/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.

### 7.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” and “help wanted” is open to whoever wants to implement it.

### 7.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “enhancement” and “help wanted” is open to whoever wants to implement it.

## 7.1.4 Write Documentation

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

## 7.1.5 Submit Feedback

The best way to send feedback is to file an issue at <https://github.com/bobtki/letterboxd/issues>.

If you are proposing a feature:

- Explain in detail how it 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 :)

## 7.2 Get Started!

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

1. Fork the *letterboxd* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/letterboxd.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv letterboxd
$ cd letterboxd/
$ python setup.py develop
```

4. Create a branch for local development:

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

Now you can make your changes locally.

5. When you're done making changes, run it through the Black code reformatter, and check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ black letterboxd
$ flake8 letterboxd tests
$ python setup.py test or pytest
$ tox
```

To get flake8 and tox, just pip install them into your virtualenv.

6. 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
```

7. Submit a pull request through the GitHub website.

## 7.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.6 and 3.7. (Once PyPy adds support for Python 3.6, the pull request should also work for pypy3.) Check [https://travis-ci.org/bobtiki/letterboxd/pull\\_requests](https://travis-ci.org/bobtiki/letterboxd/pull_requests) and make sure that the tests pass for all supported Python versions.
4. Use the pull request template to format the description and changelog information for your pull request.

## 7.4 Tips

To run a subset of tests:

```
$ pytest tests.test_letterboxd
```

## 7.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in CHANGELOG.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.



### 8.1 Development Lead

- Mark Boszko <mboszko@mac.com>

### 8.2 Contributors

- Chris Pruitt, advice and feedback





All notable changes to this project will be documented in this file.

The format is based on [Keep a Changelog](#) and this project adheres to [Semantic Versioning](#).

## 9.1 [Unreleased]

## 9.2 [0.3.0] - 2018-07-22

### 9.2.1 ADDED

- Issue and pull request templates
- Code of Conduct
- **Coverage for Letterboxd API endpoints**
  - /auth/forgotten-password-request
  - /auth/username-check
  - /list/{id}
  - /list/{id} [PATCH]
  - /list/{id} [DELETE]
  - /list/{id}/comments
  - /list/{id}/comments [POST]
  - /list/{id}/entries
  - /list/{id}/me
  - /list/{id}/me [PATCH]

- /list/{id}/report [POST]
- /list/{id}/statistics
- /lists
- /lists [POST]
- /me [PATCH]

- A TON more pytest unit tests, and made existing ones more comprehensive

## 9.2.2 CHANGED

- Internally refactored API object definitions into their own file, for pytest
- Internally changed some variable names to better reflect Letterboxd API nomenclature (I don't *think* this affects any method arguments.)

## 9.2.3 FIXED

- Removed mutable default arguments on several methods

## 9.3 [0.2.6] - 2018-07-04

### 9.3.1 CHANGED

- Getting the Travis CI integration with PyPI to work properly.

## 9.4 [0.2.5] - 2018-07-04 [YANKED]

## 9.5 [0.2.4] - 2018-07-04

### 9.5.1 CHANGED

- Getting `bumpversion` to work properly.

## 9.6 [0.2.3] - 2018-07-04 [YANKED]

## 9.7 [0.2.2] - 2018-07-04 [YANKED]

## 9.8 [0.2.1] - 2018-07-04 [YANKED]

## 9.9 [0.2.0] - 2018-07-04

### 9.9.1 Added

- This `CHANGELOG.rst`
- Converted `README.md` to `.rst`
- Documentation written with reStructuredText and Sphinx, being built to [Read the Docs](#)
- Added a number of defaults and tests as provided in [cookiecutter-pypackage](#)
- Added an easy initializer with `import letterboxd` and then `letterboxd.new()`
- `User.refresh_token()` to refresh the user authentication OAuth token
- **Coverage for Letterboxd API endpoints:**
  - `/film/{id}/members`
  - `/film/{id}/report`
  - `/film/{id}/statistics`
  - `/films`
  - `/films/film-services`
  - `/films/genres`
  - `/film-collection/{id}`
  - `/search`

### 9.9.2 Changed

- All api-calling methods now return the dictionary from the response JSON, instead of the entire `requests.Response`.

## 9.10 [0.1.0] - 2018-06-24

### 9.10.1 Added

- First public version! Version 0.1.0a tagged on [GitHub](#), and posted to PyPI.
- `letterboxd`, `api`, `user`, `auth`, `member`, and `film` modules.
- **coverage for Letterboxd API endpoints:**
  - `film`

- /film/{id}
- /film/{id}/availability — this data is first-party only
- /film/{id}/me
- /me
- /member/{id}/watchlist

## 9.11 Changelog format

- **Each version should:**
  - List its release date in ISO 8601 format (YYYY-MM-DD).
  - **Group changes to describe their impact on the project, as follows:**
    - \* `Added` for new features.
    - \* `Changed` for changes in existing functionality.
    - \* `Deprecated` for once-stable features removed in upcoming releases.
    - \* `Removed` for deprecated features removed in this release.
    - \* `Fixed` for any bug fixes.
    - \* `Security` to invite users to upgrade in case of vulnerabilities.
  - Take a look at [this checklist for packaging a new version](#), and [this one](#)

# CHAPTER 10

---

## Indices and tables

---

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



|

letterboxd.api, 12  
letterboxd.config, 13  
letterboxd.letterboxd, 11  
letterboxd.user, 12





**A**

API (class in *letterboxd.api*), 12  
 api\_call() (*letterboxd.api.API* method), 12  
 auth() (*letterboxd.letterboxd.Letterboxd* method), 11  
 Authentication (class in *letterboxd.services.auth*), 15  
 availability() (*letterboxd.services.film.Film* method), 16

**C**

comments() (*letterboxd.services.list.List* method), 18  
 create\_comment() (*letterboxd.services.list.List* method), 18  
 create\_list() (*letterboxd.services.list.Lists* method), 20

**D**

delete() (*letterboxd.services.list.List* method), 18  
 details() (*letterboxd.services.film.Film* method), 16  
 details() (*letterboxd.services.list.List* method), 18  
 details() (*letterboxd.services.member.Member* method), 20

**E**

entries() (*letterboxd.services.list.List* method), 18

**F**

Film (class in *letterboxd.services.film*), 16  
 film() (*letterboxd.letterboxd.Letterboxd* method), 11  
 film\_collection() (*letterboxd.letterboxd.Letterboxd* method), 11  
 film\_collection() (*letterboxd.services.film.FilmCollection* method), 17  
 FilmCollection (class in *letterboxd.services.film*), 17  
 Films (class in *letterboxd.services.film*), 17  
 films() (*letterboxd.letterboxd.Letterboxd* method), 11  
 films() (*letterboxd.services.film.Films* method), 17

forgotten\_password\_request() (*letterboxd.services.auth.Authentication* static method), 15

**G**

genres() (*letterboxd.services.film.Films* method), 17

**L**

Letterboxd (class in *letterboxd.letterboxd*), 11  
 letterboxd.api (module), 12  
 letterboxd.config (module), 13  
 letterboxd.letterboxd (module), 11  
 letterboxd.services.auth (module), 15  
 letterboxd.services.film (module), 16  
 letterboxd.services.list (module), 18  
 letterboxd.services.member (module), 20  
 letterboxd.services.search (module), 21  
 letterboxd.user (module), 12  
 List (class in *letterboxd.services.list*), 18  
 list() (*letterboxd.letterboxd.Letterboxd* method), 11  
 Lists (class in *letterboxd.services.list*), 20  
 lists() (*letterboxd.services.list.Lists* method), 20  
 login() (*letterboxd.services.auth.Authentication* method), 15

**M**

me (*letterboxd.user.User* attribute), 12  
 me() (*letterboxd.services.film.Film* method), 16  
 me() (*letterboxd.services.list.List* method), 19  
 me\_update() (*letterboxd.services.film.Film* method), 16  
 me\_update() (*letterboxd.services.list.List* method), 19  
 me\_update() (*letterboxd.user.User* method), 13  
 Member (class in *letterboxd.services.member*), 20  
 member() (*letterboxd.letterboxd.Letterboxd* method), 12  
 members() (*letterboxd.services.film.Film* method), 16

## R

`refresh_token()` (*letterboxd.services.auth.Authentication method*), 15

`report()` (*letterboxd.services.film.Film method*), 17

`report()` (*letterboxd.services.list.List method*), 19

## S

`Search` (*class in letterboxd.services.search*), 21

`search()` (*letterboxd.letterboxd.Letterboxd method*), 12

`search()` (*letterboxd.services.search.Search method*), 21

`services()` (*letterboxd.services.film.Films method*), 17

`statistics()` (*letterboxd.services.film.Film method*), 17

`statistics()` (*letterboxd.services.list.List method*), 19

## T

`token` (*letterboxd.services.auth.Authentication attribute*), 15

`token` (*letterboxd.user.User attribute*), 13

## U

`update()` (*letterboxd.services.list.List method*), 19

`User` (*class in letterboxd.user*), 12

`user()` (*letterboxd.letterboxd.Letterboxd method*), 12

## W

`watchlist()` (*letterboxd.services.member.Member method*), 20