
eater Documentation

Release 0.4.0

Alex Hayes

Apr 20, 2021

CONTENTS

1	Contents	3
2	License	13
3	Author	15
	Python Module Index	17
	Index	19

Python library to consume APIs and hold them to account.



Not every API provider has an API as nice as [Stripe](#) and often their documentation is incorrect or (arguably better) non-existent.

Not every API provider can manage their release cycle, like inform customers, conform to semantic version numbers etc..

Chances are, if you're reading this, you're trying to talk to an API just like this.

You want to be sure, when you send them something, they give you the right thing back. That is, it contains the write shape of data and the correct data types and if you're really persnickety, that it validates against defined rules.

Well, this Python library, with the help of [schematics](#), will allow you to do just that.

CONTENTS

1.1 Installation

Note this library only supports Python 3.5+ - if you want to add support for other versions, please see the [Developer Documentation](#) docs.

You can install eater either via the Python Package Index (PyPI) or from github.

To install using pip;

```
$ pip install eater
```

From github;

```
$ pip install git+https://github.com/alexhayes/eater.git
```

1.2 Usage

1.2.1 Quickstart

Let's get started with a trivial example, this usage assumes you are familiar with [schematics](#) (v2.x) and somewhat familiar with [requests](#).

Firstly, define models that represent the API you have the fortunate pleasure of communicating with.

```
from schematics import Model
from schematics.types import ListType, ModelType, StringType

class Book(Model):
    title = StringType(required=True, min_length=3)

class BookListResponse(Model):
    """
    Represents, as a Python object, the JSON response returned by the API.
    """
    books = ListType(ModelType(Book))
```

Once you've defined models that represent the APIs input/output you then create a class that inherits `eater.HTTP eater`. Your class must define a `url`, `request_cls` and `response_cls`.

```
import eater

class BookListAPI(eater.HTTP eater):
    url = 'http://example.com/books/'
    response_cls = BookListResponse
```

You can then consume the API;

```
api = BookListAPI()

response = api()

for book in response.books:
    print(book.title)
```

Note that you *call* the actual instance of your API class.

1.2.2 Holding the API to account

That's right, we were concerned our API wasn't going to do what it said it would. That would be hard to imagine for the trivial example we have above however accidents do happen, developers are only human right?!

Remember our definition of a book?

```
class Book(Model):
    title = StringType(required=True, min_length=3)
```

If for some reason the endpoint at <https://example.com/books/> returned a book that contained a title less than three characters in length schematics would kindly raise a `DataError` for us.

For example;

```
from schematics.exceptions import DataError

try:
    response = api()
except DataError as e:
    # Oh no, our API provider didn't give us back what they said they would
    # e would now contain something like:
    # schematics.exceptions.DataError: {'title': ValidationError("String value is too
    ↪short.") }
```

1.2.3 HTTP request type

By default `HTTP eater` performs a HTTP GET request however you can change this by setting `method` on your API class;

```
class BookCreateAPI(eater.HTTP eater):
    method = 'post'
    ...
```

Any request method supported by `requests` are supported, ie... `get`, `post`, `put`, `delete`, `head`, `options`.

1.2.4 Post Data

You can POST a JSON object over the wire by defining a `request_cls` on your API class, as follows;

```
class BookCreateAPI(eater.HTTP eater):
    url = 'http://example.com/books/'
    method = 'post'
    request_cls = Book
    response_cls = Book
```

You can then call your API as follows;

```
api = BookCreateAPI(name='Awesome Book')
response = api()
```

Which would result in the following JSON payload being sent to the server;

```
{
    name: "Awesome Book"
}
```

It's also possible to pass in an instance of your `request_cls` as the first (and only) parameter.

```
book = Book({'name': 'Awesome Book'})
api = BookCreateAPI(book)
response = api()
```

1.2.5 Dynamic URL

The `url` can contain string formatting that refers the request model, like so;

```
class GetBookRequest(Model):
    id = IntType(required=True, min_value=1)

class GetBookAPI(eater.HTTP eater):
    url = 'http://path.to.awesome/{request_model.id}'
    request_cls = GetBookRequest
    response_cls = Book
```

To retrieve the formatted URL you can call `.url` on the instance and it will give you the formatted URL.

```
api = GetBookAPI(id=1234)
print(api.url)
# prints: http://path.to.awesome/1324
```

If you need to get the unformatted URL you must call `.url` on the class:

```
print(GetBookAPI.url)
# prints: http://path.to.awesome/{request_model.id}
```

For more control you can also override the `get_url` method;

```
class GetBookAPI(eater.HTTP eater):
    url = 'http://path.to.awesome/{request_model.id}'
    request_cls = GetBookRequest
```

(continues on next page)

(continued from previous page)

```
response_cls = Book

def get_url(self) -> str:
    if self.request_model.id < 100:
        url = 'http://path.to.less.awesome/{request_model.id}'
    else:
        url = type(self).url
    return url.format(request_model=request_model)
```

It's important to note that in your `get_url` method you should use `type(self).url` rather than `self.url`. This is because `self.url` is replaced with the formatted URL within HTTPeater's `__init__` function.

1.2.6 More Control

You can control any kwarg supplied to `requests` by defining a `get_request_kwargs` method in your API class.

For instance, if you want to pass some parameters in the URL;

```
class BookListAPI(HTTPeater):

    def get_request_kwargs(self, request_model: BookListRequest, **kwargs) -> dict:
        """
        Returns a dict of kwargs to supply to requests.
        """
        kwargs['params'] = {
            'in_print': request_model.in_print
        }
        return kwargs
```

However, a better way of setting `kwargs['params']` above would be;

```
kwargs['params'] = request_model.to_primitive()
```

Calling `to_primitive()` on your model returns a dict of native python types suitable for sending over the wire. See the [schematics docs](#) for more information.

1.2.7 Auth, Headers & Sessions

Under the covers HTTPeater automatically creates a `requests.Session` for you.

When you create an instance of your API class that inherits HTTPeater you can pass through kwargs that will be applied to this generated session, or optionally you can pass in a session object of your creation.

```
api = BookListAPI(_requests={'auth': ('john', 's3cr3t')})
```

Need to set a custom header?

```
api = BookListAPI(_requests={'headers': {'EGGS': 'Sausage'}})
```

Or need to do something really special with your own custom session?

```
session = requests.Session()
api = BookListAPI(_requests={'session': session})
```

Alternatively a nicer approach than supplying `_requests` every time you instantiate your API is to subclass `HTTPEater`, define a `create_session` method and have your `BookListAPI` class inherit from your subclass.

```
class AwesomeAPI(eater.HTTPEater):

    def create_session(self, **kwargs):
        """
        Ensure we set auth for all API calls...
        """
        self.session = requests.Session()
        # Get auth details from settings, or if you're feeling reckless just hard_
↪code them...
        self.session.auth = ('john', 's3cr3t')
        self.session.headers.update({'EGGS', 'Sausage'})
        return self.session

class BookListAPI(AwesomeAPI):
    url = 'https://example.com/books/'
    request_cls = BookListRequest
    response_cls = Response
```

This way, whenever you use the `BookListAPI` it will automatically have your auth details set.

1.2.8 Control everything!

You can break into all aspects of eater's lifecycle by overriding methods on your API class;

- `HTTPEater.get_url()` - Modify the URL
- `HTTPEater.create_request_model()` - Modify the creation of your `request_model`
- `HTTPEater.get_request_kwargs()` - Modify the kwargs supplied to `requests`
- `HTTPEater.create_response_model()` - Modify the creation of the `response_model` from the requests response.
- `HTTPEater.create_session()` - Modify the creation of the session.

See the *Internal Module Reference* for more details.

1.3 Developer Documentation

1.3.1 Contributions

Contributions are more than welcome!

To get setup do the following;

```
mkvirtualenv --python=/usr/bin/python3.5 eater
git clone https://github.com/alexhayes/eater.git
cd eater
pip install -r requirements/dev.txt
```

1.3.2 Running Tests

Once you've checked out you should be able to run the tests;

```
tox
```

Or run all environments at once using detox;

```
detox
```

Or simply run with py.test;

```
py.test
```

1.3.3 Linting

Running pylint is easy and is part of the CI;

```
pylint eater
```

1.3.4 Creating Documentation

```
cd docs
make apidoc clean html
```

1.4 Internal Module Reference

Release 0.4.0

Date Apr 20, 2021

1.4.1 eater.api package

Submodules

eater.api.base module

eater.api.base

Base Eater API classes and utilities.

```
class eater.api.base.BaseEater
    Bases: abc.ABC
```

Base Eater class.

```
abstract property request_cls
    A schematics model that represents the API request.
```

```
abstract property response_cls
    A schematics model that represents the API response.
```

eater.api.http module

eater.api

Eater HTTP API classes.

```
class eater.api.http.HTTPEater (request_model: Optional[schematics.models.Model] = None, *,  
                                _requests: dict = {}, **kwargs)  
    Bases: eater.api.base.BaseEater
```

Eat JSON HTTP APIs for breakfast.

Instances of this class can't be created directly, you must subclass this class and set `url` and `response_cls`.

See *Usage* for more details.

Initialise instance of HTTPEater.

Parameters

- **request_model** (*Model*) – An instance of a schematics model
- **_requests** (*dict*) – A dict of kwargs to be supplied when creating a requests session.
- **kwargs** (*dict*) – If `request_model` is not defined a dict of kwargs to be supplied as the first argument `raw_data` when creating an instance of `request_cls`.

```
__init__ (request_model: Optional[schematics.models.Model] = None, *, _requests: dict = {},  
          **kwargs)
```

Initialise instance of HTTPEater.

Parameters

- **request_model** (*Model*) – An instance of a schematics model
- **_requests** (*dict*) – A dict of kwargs to be supplied when creating a requests session.
- **kwargs** (*dict*) – If `request_model` is not defined a dict of kwargs to be supplied as the first argument `raw_data` when creating an instance of `request_cls`.

```
create_request_model (request_model: Optional[schematics.models.Model] = None, **kwargs)  
                      → schematics.models.Model
```

Create the request model either from `kwargs` or `request_model`.

Parameters

- **request_model** (*Model / None*) – An instance of `request_cls` or `None`.
- **kwargs** (*dict*) – kwargs to be supplied as the `raw_data` parameter when instantiating `request_cls`.

Returns An instance of `request_cls`.

Return type `schematics.Model`

```
create_response_model (response: requests.models.Response, request_model: schematics.models.Model) → schematics.models.Model
```

Given a requests `Response` object, return the response model.

Parameters

- **response** (*requests.Response*) – A `requests.Response` object representing the response from the API.
- **request_model** (*schematics.Model*) – The model used to generate the request - an instance of `request_cls`.

create_session (*session: Optional[requests.sessions.Session] = None, auth: Optional[tuple] = None, headers: Optional[requests.structures.CaseInsensitiveDict] = None*) → *requests.sessions.Session*

Create and return an instance of a requests Session.

Parameters

- **auth** (*tuple|None*) – The auth kwarg when to supply when instantiating *requests.Session*.
- **headers** (*requests.structures.CaseInsensitiveDict*) – A dict of headers to be supplied as the headers kwarg when instantiating *requests.Session*.

Returns An instance of *requests.Session*

Return type *requests.Session*

get_request_kwargs (*request_model: Optional[schematics.models.Model], **kwargs*) → *dict*

Retrieve a dict of kwargs to supply to requests.

Parameters

- **request_model** (*Model|None*) – An instance of *request_cls* or None.
- **kwargs** (*dict*) – kwargs to be supplied as the *raw_data* parameter when instantiating *request_cls*.

Returns A dict of kwargs to be supplied to requests when making a HTTP call.

Return type *dict*

get_url () → *str*

Retrieve the URL to be used for the request.

Note that this method should always use *type(self).url* to access the *url* property defined on the class. This is necessary because the *url* property is replaced in *HTTPEater.__init__()*.

Returns The URL to the API endpoint.

Return type *str*

method = 'get'

The HTTP method to use to make the API call.

request (***kwargs*) → *schematics.models.Model*

Make a HTTP request of of type *method*.

You should generally leave this method alone. If you need to customise the behaviour use the methods that this method uses.

request_cls = None

Default *request_cls* to None

session = None

An instance of requests Session

abstract property url

Returns the URL to the endpoint - property must be defined by a subclass.

Note that this property is replaced with the value of *HTTPEater.get_url()* within *HTTPEater.__init__()*.

Module contents

1.4.2 eater package

Subpackages

eater.tests package

Subpackages

eater.tests.api package

Submodules

eater.tests.api.test_base module

eater.tests.api.test_http module

Module contents

Module contents

Submodules

eater.errors module

eater.errors

A place for errors that are raised by Eater.

exception eater.errors.**EaterConnectError**

Bases: *eater.errors.EaterError*

Raised if there is a connection error.

exception eater.errors.**EaterError**

Bases: *Exception*

Base Eater error.

exception eater.errors.**EaterTimeoutError**

Bases: *eater.errors.EaterError*

Raised if something times out.

exception eater.errors.**EaterUnexpectedError**

Bases: *eater.errors.EaterError*

Raised when something unexpected happens.

exception eater.errors.**EaterUnexpectedResponseError**

Bases: *eater.errors.EaterUnexpectedError*

Raised when a response from an API is unexpected.

Module contents

Consume APIs and hold them to account.

```
class eater.VersionInfo (major, minor, micro, releaselevel, serial)
    Bases: tuple

    Create new instance of VersionInfo(major, minor, micro, releaselevel, serial)

    property major
        Alias for field number 0

    property micro
        Alias for field number 2

    property minor
        Alias for field number 1

    property releaselevel
        Alias for field number 3

    property serial
        Alias for field number 4
```

1.4.3 eater

LICENSE

This software is licensed under the *MIT License*. See the [LICENSE](#).

CHAPTER
THREE

AUTHOR

Alex Hayes <alex@alution.com>

PYTHON MODULE INDEX

e

- `eater`, [12](#)
- `eater.api`, [11](#)
- `eater.api.base`, [8](#)
- `eater.api.http`, [9](#)
- `eater.errors`, [11](#)
- `eater.tests`, [11](#)
- `eater.tests.api`, [11](#)

Symbols

`__init__()` (*eater.api.http.HTTPEater method*), 9

B

`BaseEater` (*class in eater.api.base*), 8

C

`create_request_model()`
 (*eater.api.http.HTTPEater method*), 9
`create_response_model()`
 (*eater.api.http.HTTPEater method*), 9
`create_session()` (*eater.api.http.HTTPEater method*), 9

E

`eater`
 module, 12
`eater.api`
 module, 11
`eater.api.base`
 module, 8
`eater.api.http`
 module, 9
`eater.errors`
 module, 11
`eater.tests`
 module, 11
`eater.tests.api`
 module, 11
`EaterConnectError`, 11
`EaterError`, 11
`EaterTimeoutError`, 11
`EaterUnexpectedError`, 11
`EaterUnexpectedResponseError`, 11

G

`get_request_kwargs()` (*eater.api.http.HTTPEater method*), 10
`get_url()` (*eater.api.http.HTTPEater method*), 10

H

`HTTPEater` (*class in eater.api.http*), 9

M

`major()` (*eater.VersionInfo property*), 12
`method` (*eater.api.http.HTTPEater attribute*), 10
`micro()` (*eater.VersionInfo property*), 12
`minor()` (*eater.VersionInfo property*), 12
module
 eater, 12
 eater.api, 11
 eater.api.base, 8
 eater.api.http, 9
 eater.errors, 11
 eater.tests, 11
 eater.tests.api, 11

R

`releaselevel()` (*eater.VersionInfo property*), 12
`request()` (*eater.api.http.HTTPEater method*), 10
`request_cls` (*eater.api.http.HTTPEater attribute*), 10
`request_cls()` (*eater.api.base.BaseEater property*), 8
`response_cls()` (*eater.api.base.BaseEater property*), 8

S

`serial()` (*eater.VersionInfo property*), 12
`session` (*eater.api.http.HTTPEater attribute*), 10

U

`url()` (*eater.api.http.HTTPEater property*), 10

V

`VersionInfo` (*class in eater*), 12