devourer Documentation

Release 0.1

Bonnier Business Polska

Dec 24, 2018

Contents

Contents: Devourer is a generic API client. It features an object-oriented, declarative approach to simplify the communication. It depends on the brilliant requests package as the gateway to API server. A simple example:

```
>>> from devourer import GenericAPI, APIMethod, APIError
>>>
>>>
>>> class TestApi(GenericAPI):
>>>     posts = APIMethod('get', 'posts/')
>>>     comments = APIMethod('get', 'posts/{id}/comments')
>>>     post = APIMethod('get', 'posts/{id}/')
>>>     add_post = APIMethod('post', 'posts/')
>>>
>>>     def __init__(self,
>>>                  url=None,
>>>                  auth=None,
>>>                  throw_on_error=True,
>>>                  load_json=True):
>>>         params = (url or 'http://jsonplaceholder.typicode.com/',
>>>                   auth or None,  # this can be ('user', 'password')
>>>                                  # or requests auth object
>>>                   load_json=load_json,
>>>                   throw_on_error=throw_on_error)
>>>         super(TestApi, self).__init__(*params)
>>>
>>> api = TestApi()
>>> posts = api.posts()
>>> post = api.post(id=posts[0]['id'])
>>> comments = api.comments(id=post['id'])
>>> new_post_id = api.add_post(userId=1,
>>>                            title='Breaking news',
>>>                            body='I just got devoured.')
>>> try:
>>>     post = api.post(id=new_post_id)
>>> except APIError:
>>>     print('Oops, this API is not persistent!')
```

The init function gives details so you don't need to repeat them elsewhere, enables parsing json responses and raising exceptions on error. You can also obtain raw string with *load_json=False* and silence errors getting None instead when they happen with *throw_on_error=False*.

class devourer.**GenericAPIBase**(*url*, *auth*, *throw_on_error=False*, *load_json=False*, *headers=None*)

This is the base API representation class without declarative syntax.

Requires GenericAPICreator metaclass to work.

call(*name*, **args*, ***kwargs*)

This function invokes the API method from the class declaration according to the name parameter along with all the hooks.

Parameters

- **name** – name of method to call.
- **args** – non-keyword arguments of API method call.
- **kwargs** – keyword arguments of API method call.

Returns Result of finalize_method call, by default content of API's response.

finalize(*name*, *result*, **args*, ***kwargs*)
Post-request hook. By default it takes care of throw_on_error and returns response content.

Parameters

- **name** – name of the called method.
- **result** – requests' response object.
- **args** – non-keyword arguments of API method call.
- **kwargs** – keyword arguments of API method call.

Returns result.content

invoke(*http_method*, *url*, *params*, *data=None*, *payload=None*, *headers=None*, *requests_kwargs=None*)
This method makes a request to given API address concatenating the method path and passing along authentication data.

Parameters

- **http_method** – http method to be used for this call.
- **url** – exact address to be concatenated to API address.
- **data** – dict or encoded string to be sent as request body.
- **payload** – the payload dictionary to be sent in body of the request, encoded as JSON.
- **headers** – the headers to be sent with http request.

Returns response object as in requests.

classmethod outer_call(*name*)
This is a wrapper creating anonymous function invoking call with correct method name.

Parameters **name** – Name of method for which call wrapper has to be created.

Returns drop-in call replacement lambda.

prepare(*name*, **args*, ***kwargs*)
This function is a pre-request hook. It receives the exact same parameters as the API method call and the name of the method. It should return a PrepareCallArgs instance.

By default it doesn't change the args and selects 'name' method from the class declaration as the callable to execute.

Parameters

- **name** – name of API method to call.
- **args** – non-keyword arguments of API method call.
- **kwargs** – keyword arguments of API method call.

Returns PrepareCallArgs instance

class devourer.**GenericAPI**(*url*, *auth*, *throw_on_error=False*, *load_json=False*, *headers=None*)
This is the base API representation class.

You can build a concrete API by declaring methods while creating the class, ie.:

```
>>> class MyAPI(GenericAPI):
>>>     method1 = APIMethod('get', 'people/')
>>>     method2 = APIMethod('post', 'my/news/items/')
```

Hooks can be overridden globally:

```
>>> def prepare(self, name, **args, **kwargs):
>>>     return PrepareCallArgs(call=self._methods[name],
>>>                            args=args,
>>>                            kwargs=kwargs)
```

As well as for particular methods only:

```
>>> def prepare_method1(self, name, *args, **kwargs):
>>>     return PrepareCallArgs(call=self._methods[name],
>>>                            args=args,
>>>                            kwargs=kwargs)
```

```
>>> def call_method1(self, name, *args, **kwargs):
>>>     prepared = getattr(self, 'prepare_{}'.format(name))
>>>     prepared = prepared(name, *args, **kwargs)
>>>     callback = getattr(self, 'finalize_{}'.format(name))
>>>     return callback(name,
>>>                     prepared.call(*prepared.args,
>>>                                   **prepared.kwargs),
>>>                     *prepared.args,
>>>                     **prepared.kwargs)
```

```
>>> def finalize_method2(self, name, result, *args, **kwargs):
>>>     if self.throw_on_error and result.status_code >= 400:
>>>         error_msg = "Error when invoking {} with parameters {} {}: {}"
>>>         params = (name, args, kwargs, result.__dict__)
>>>         raise APIError(error_msg.format(*params))
>>>     if self.load_json:
>>>         return json.loads(result.content)
>>>     return result.content
```

class devourer.**AsyncAPI**(**args*, ***kwargs*)

This is the async API representation class.

You can build a concrete API by declaring methods while creating the class, ie.:

```
>>> class MyAPI(AsyncAPI):
>>>     method1 = APIMethod('get', 'people/')
>>>     method2 = APIMethod('post', 'my/news/items/')
```

Hooks can be overridden globally:

```
>>> def prepare(self, name, **args, **kwargs):
>>>     return PrepareCallArgs(call=self._methods[name],
>>>                            args=args,
>>>                            kwargs=kwargs)
```

As well as for particular methods only:

```
>>> def prepare_method1(self, name, *args, **kwargs):
>>>     return PrepareCallArgs(call=self._methods[name],
>>>                            args=args,
>>>                            kwargs=kwargs)
```

```
>>> def call_method1(self, name, *args, **kwargs):
>>>     prepared = getattr(self, 'prepare_{}'.format(name))
```

(continues on next page)

(continued from previous page)

```
>>>     prepared = prepared(name, *args, **kwargs)
>>>     callback = getattr(self, 'finalize_{}'.format(name))
>>>     return callback(name,
>>>                     prepared.call(*prepared.args,
>>>                                   **prepared.kwargs),
>>>                     *prepared.args,
>>>                     **prepared.kwargs)
```

```
>>> def finalize_method2(self, name, future, *args, **kwargs):
>>>     result = future.result()
>>>     if self.throw_on_error and result.status_code >= 400:
>>>         error_msg = "Error when invoking {} with parameters {} {}: {}"
>>>         params = (name, args, kwargs, result.__dict__)
>>>         raise APIError(error_msg.format(*params))
>>>     if self.load_json:
>>>         return json.loads(result.content)
>>>     return result.content
```

class devourer.**APIMethod**(*http_method*, *schema*, *requests_kwargs=None*)

This class represents a single method in an API. It's able to dynamically create request URL using schema and call parameters. The schema uses Python 3-style string formatting. Usually you don't need to call any methods by hand.

Example:

```
>>> post = APIMethod('get', 'post/{id}/')
```

params

List of available parameters for this method.

Returns List of available parameters for this method.

schema

Method's address relative to API address.

Returns Method's address relative to API address.

class devourer.**APIError**(*message*, ***kwargs*)

An error while querying the API.

class devourer.**GenericAPICreator**

This creator is a metaclass (it's a subclass of type, not object) responsible for creating and injecting helper methods as well as connecting APIMethods with GenericAPI.

class devourer.**PrepareCallArgs**(*call=None*, *args=None*, *kwargs=None*)

An inner class containing properties required to fire off a request to an API. It's not a namedtuple because it provides default values.

CHAPTER 1

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

Index