apypie Documentation

Release 0.4.0

Evgeni Golov

Jun 14, 2023

Contents:

CHAPTER 1

Developer Interface

class apypie.**Api**(***kwargs*)

Apipie API bindings

Parameters

- **uri** – base URL of the server
- **username** – username to access the API
- **password** – username to access the API
- **api_version** – version of the API. Defaults to *1*
- **language** – prefered locale for the API description
- **apidoc_cache_base_dir** – base directory for building apidoc_cache_dir. Defaults to *~/.cache/apipie_bindings*.
- **apidoc_cache_dir** – where to cache the JSON description of the API. Defaults to *apidoc_cache_base_dir/<URI>*.
- **apidoc_cache_name** – name of the cache file. If there is cache in the *apidoc_cache_dir*, it is used. Defaults to *default*.
- **verify_ssl** – should the SSL certificate be verified. Defaults to *True*.
- **session** – a *requests.Session* compatible object. Defaults to *requests.Session()*.

Usage:

```
>>> import apypie
>>> api = apypie.Api(uri='https://api.example.com', username='admin', password=
↪'changeme')
```

apidoc

The full apidoc.

The apidoc will be fetched from the server, if that didn't happen yet.

Returns The apidoc.

apidoc_cache_file
Full local path to the cached apidoc.

cache_extension
File extension for the local cache file.

Will include the language if set.

call(*resource_name*, *action_name*, *params=None*, *headers=None*, *options=None*, *data=None*, *files=None*)
Call an action in the API.

It finds most fitting route based on given parameters with other attributes necessary to do an API call.

Parameters

- **resource_name** – name of the resource
- **action_name** – action_name name of the action
- **params** – Dict of parameters to be sent in the request
- **headers** – Dict of headers to be sent in the request
- **options** – Dict of options to influence the how the call is processed * *skip_validation* (Bool) *false* - skip validation of parameters
- **data** – Binary data to be sent in the request
- **files** – Binary files to be sent in the request

Returns `dict` object

Return type dict

Usage:

```
>>> api.call('users', 'show', {'id': 1})
```

clean_cache()
Remove any locally cached apidocs.

http_call(*http_method*, *path*, *params=None*, *headers=None*, *data=None*, *files=None*)
Execute an HTTP request.

Parameters

- **params** – Dict of parameters to be sent in the request
- **headers** – Dict of headers to be sent in the request
- **data** – Binary data to be sent in the request
- **files** – Binary files to be sent in the request

Returns `dict` object

Return type dict

resource(*name*)
Get a resource.

Parameters **name** – the name of the resource to load

Returns *Resource* object

Return type *apypie.Resource*

Usage:

```
>>> api.resource('users')
```

resources
List of available resources.

Usage:

```
>>> api.resources
['comments', 'users']
```

validate_cache(*cache_name=None*)
Ensure the cached apidoc matches the one presented by the server.

Parameters **cache_name** – The name of the apidoc on the server.

class apypie.**Resource**(*api*, *name*)
Apipie Resource

action(*name*)
Get an *Action* for this resource.

Parameters **name** – The name of the action.

actions
Actions available for this resource.

Returns The actions.

call(*action*, *params=None*, *headers=None*, *options=None*, *data=None*, *files=None*)
Call the API to execute an action for this resource.

Parameters

- **action** – The action to call.
- **params** – The params that should be passed to the API.
- **headers** – Additional headers to be passed to the API.
- **options** – Options
- **data** – Binary data to be submitted to the API.
- **files** – Files to be submitted to the API.

Returns The API response.

has_action(*name*)
Check whether the resource has a given action.

Parameters **name** – The name of the action.

class apypie.**Action**(*name*, *resource*, *api*)
Apipie Action

apidoc
The apidoc of this action.

Returns The apidoc.

call(*params=None*, *headers=None*, *options=None*, *data=None*, *files=None*)
Call the API to execute the action.

Parameters

- **params** – The params that should be passed to the API.
- **headers** – Additional headers to be passed to the API.
- **options** – Options
- **data** – Binary data to be submitted to the API.
- **files** – Files to be submitted to the API.

Returns The API response.

examples
The examples of this action.

Returns The examples.

static filter_empty_params(*params=None*)
Filter out any params that have no value.

Parameters **params** – The params to filter.

Returns The filtered params.

find_route(*params=None*)
Find the best matching route for a given set of params.

Parameters **params** – Params that should be submitted to the API.

Returns The best route.

params
The params accepted by this action.

Returns The params.

prepare_params(*input_dict*)
Transform a dict with data into one that can be accepted as params for calling the action.

This will ignore any keys that are not accepted as params when calling the action. It also allows generating nested params without forcing the user to care about them.

Parameters **input_dict** – a dict with data that should be used to fill in the params

Returns `dict` object

Return type dict

Usage:

```
>>> action.prepare_params({'id': 1})
{'user': {'id': 1}}
```

routes
The routes this action can be invoked by.

Returns The routes

validate(*values*, *data=None*, *files=None*)
Validate a given set of parameter values against the required set of parameters.

Parameters

- **values** – The values to validate.
- **data** – Additional binary data to validate.
- **files** – Additional files to validate.

class apypie.**Param**(***kwargs*)
Apipie Param

class apypie.**Route**(*path*, *method*, *description=''*)
Apipie Route

params_in_path
Params that can be passed in the path (URL) of the route.

Returns The params.

path_with_params(*params=None*)
Fill in the params into the path.

Returns The path with params.

class apypie.**Example**(*http_method*, *path*, *args*, *status*, *response*)
Apipie Example

classmethod parse(*example*)
Parse an example from an apidoc string

Returns The parsed *Example*

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index