apeye

Release 1.4.1

Handy tools for working with URLs and APIs.

Dominic Davis-Foster

May 15, 2024

Contents

Chapter

ONE

Overview

apeye provides:

- *apeye.url*: pathlib.Path-like objects to represent URLs
- *Cache*: A JSON-backed cache decorator for functions
- *RateLimitAdapter*: A CacheControl adapter to limit the rate of requests

Chapter

TWO

Installation

2.1 from PyPI

```
$ python3 -m pip install apeye --user
```

2.2 from Anaconda

First add the required channels

```
$ conda config --add channels https://conda.anaconda.org/conda-forge
$ conda config --add channels https://conda.anaconda.org/domdfcoding
```

Then install

```
$ conda install apeye
```

2.3 from GitHub

```
$ python3 -m pip install git+https://github.com/domdfcoding/apeye@master --user
```

Attention: In v0.9.0 and above the *rate_limiter* module requires the `limiter` extra to be installed:

```
$ python -m pip install apeye[limiter]
```

Chapter

THREE

apeye.url

Source code: apeye_core/__init__.py

---

pathlib-like approach to URLs.

Changed in version 0.2.0: *SlumberURL* and *RequestsURL* moved to *apeye.slumber_url* and *apeye.requests_url* respectively.

Note: The classes in this module can instead be imported from the apeye_core module instead.

Classes:

Domain(subdomain, domain, suffix)	typing.NamedTuple of a URL's subdomain, domain, and suffix.
URL([url])	pathlib-like class for URLs.
URLPath(*args)	Represents the path part of a URL.

Data:

URLPathType	Invariant TypeVar bound to *apeye.url.URLPath*.
URLType	Invariant TypeVar bound to *apeye.url.URL*.

URLType = TypeVar(URLType, bound=URL)

Type: TypeVar

Invariant TypeVar bound to *apeye.url.URL*.

URLPathType = TypeVar(URLPathType, bound=URLPath)

Type: TypeVar

Invariant TypeVar bound to *apeye.url.URLPath*.

class URL(*url=''*)

Bases: PathLike

pathlib-like class for URLs.

Parameters **url** (Union[str, *URL*]) – The URL to construct the *URL* object from. Default ''.

Changed in version 0.3.0: The url parameter can now be a string or a *URL*.

Changed in version 1.1.0: Added support for sorting and rich comparisons (<, <=, > and >=).

Methods:

__eq__(other)	Return self == other.
__fspath__()	Returns the file system path representation of the *URL*.
__repr__()	Returns the string representation of the *URL*.
__str__()	Returns the *URL* as a string.
__truediv__(key)	Construct a new *URL* object for the given child of this *URL*.
from_parts(scheme, netloc, path[, query, ...])	Construct a *URL* from a scheme, netloc and path.
joinurl(*args)	Construct a new *URL* object by combining the given arguments with this instance's path part.
relative_to(other)	Returns a version of this URL's path relative to other.
strict_compare(other)	Return self ≡ other, comparing the scheme, netloc, path, fragment and query parameters.
with_name(name[, inherit])	Return a new *URL* with the file name changed.
with_suffix(suffix[, inherit])	Returns a new *URL* with the file suffix changed.

Attributes:

base_url	Returns a *apeye.url.URL* object representing the URL without query strings or URL fragments.
domain	Returns a *apeye.url.Domain* object representing the domain part of the URL.
fqdn	Returns the Fully Qualified Domain Name of the *URL* .
fragment	The URL fragment, used to identify a part of the document.
name	The final path component, if any.
netloc	Network location part of the URL
parent	The logical parent of the *URL*.
parents	An immutable sequence providing access to the logical ancestors of the *URL*.
parts	An object providing sequence-like access to the components in the URL.
path	The hierarchical path of the URL
port	The port of number of the URL as an integer, if present.
query	The query parameters of the URL, if present.
scheme	URL scheme specifier
stem	The final path component, minus its last suffix.
suffix	The final component's last suffix, if any.
suffixes	A list of the final component's suffixes, if any.

__class_getitem__ = <bound method GenericAlias of <class 'apeye.url.URL'>>

Type: MethodType

__eq__(*other*)
Return `self == other`.

> **Attention:** URL fragments and query parameters are not compared.
>
> **See also:** *URL.strict_compare()*, which *does* consider those attributes.

Return type bool

__fspath__()
Returns the file system path representation of the *URL*.

This is comprised of the `netloc` and `path` attributes.

Return type str

__repr__()
Returns the string representation of the *URL*.

Return type str

__str__()
Returns the *URL* as a string.

Return type str

__truediv__(*key*)
Construct a new *URL* object for the given child of this *URL*.

Return type *~URLType*

Changed in version 0.7.0:

- Added support for division by integers.
- Now officially supports the new path having a URL fragment and/or query parameters. Any URL fragment or query parameters from the parent URL are not inherited by its children.

property base_url: apeye.url.URLType
Returns a *apeye.url.URL* object representing the URL without query strings or URL fragments.

New in version 0.7.0.

Return type *~URLType*

property domain: apeye.url.Domain
Returns a *apeye.url.Domain* object representing the domain part of the URL.

Return type *Domain*

property fqdn: str
Returns the Fully Qualified Domain Name of the *URL* .

Return type str

fragment
Type: Optional[str]

The URL fragment, used to identify a part of the document. None if absent from the URL.

New in version 0.7.0.

classmethod from_parts(*scheme*, *netloc*, *path*, *query=None*, *fragment=None*)
Construct a *URL* from a scheme, netloc and path.

Parameters

- **scheme** (str) – The scheme of the URL, e.g 'http'.
- **netloc** (str) – The netloc of the URl, e.g. 'bbc.co.uk:80'.
- **path** (Union[str, Path, PathLike]) – The path of the URL, e.g. '/news'.
- **query** (Optional[Mapping[Any, List]]) – The query parameters of the URL, if present. Default None.
- **fragment** (Optional[str]) – The URL fragment, used to identify a part of the document. None if absent from the URL. Default None.

Put together, the resulting path would be 'http://bbc.co.uk:80/news'

Return type *~URLType*

Changed in version 0.7.0: Added the query and fragment arguments.

joinurl(**args*)
Construct a new *URL* object by combining the given arguments with this instance's path part.

New in version 1.1.0.

Except for the final path element any queries and fragments are ignored.

Return type *~URLType*

Returns A new *URL* representing either a subpath (if all arguments are relative paths) or a totally different path (if one of the arguments is absolute).

property name: str
The final path component, if any.

Return type str

netloc
Type: str

Network location part of the URL

property parent: apeye.url.URLType
The logical parent of the *URL*.

Return type *~URLType*

property parents: Tuple[apeye.url.URLType, ...]
An immutable sequence providing access to the logical ancestors of the *URL*.

Return type Tuple[*~URLType*, ...]

property parts: Tuple[str, ...]
An object providing sequence-like access to the components in the URL.

To retrieve only the parts of the path, use *URL.path.parts*.

Return type Tuple[str, ...]

path
Type: *URLPath*

The hierarchical path of the URL

property port: Optional[int]
The port of number of the URL as an integer, if present. Default None.

New in version 0.7.0.

Return type Optional[int]

query
Type: Dict[str, List[str]]

The query parameters of the URL, if present.

New in version 0.7.0.

relative_to(*other*)
Returns a version of this URL's path relative to `other`.

New in version 1.1.0.

Parameters **other** (Union[str, *URL*, *URLPath*]) – Either a *URL*, or a string or *URLPath* representing an *absolute* path. If a *URL*, the *netloc* must match this URL's.

Raises **ValueError** – if the operation is not possible (i.e. because this URL's path is not a subpath of the other path)

Return type *URLPath*

scheme
Type: str

URL scheme specifier

property stem: str
The final path component, minus its last suffix.

Return type str

strict_compare(*other*)
Return `self` $\equiv$ `other`, comparing the scheme, netloc, path, fragment and query parameters.

New in version 0.7.0.

Return type bool

property suffix: str
The final component's last suffix, if any.

This includes the leading period. For example: '.txt'.

Return type str

property suffixes: List[str]
A list of the final component's suffixes, if any.

These include the leading periods. For example: ['.tar', '.gz'].

Return type List[str]

with_name(*name*, *inherit=True*)
Return a new *URL* with the file name changed.

Parameters

- **name** (str)
- **inherit** (bool) – Whether the new *URL* should inherit the query string and fragment from this *URL*. Default True.

Return type *~URLType*

Changed in version 0.7.0: Added the inherit parameter.

with_suffix(*suffix*, *inherit=True*)
Returns a new *URL* with the file suffix changed.

If the *URL* has no suffix, add the given suffix.

If the given suffix is an empty string, remove the suffix from the *URL*.

Parameters

- **suffix** (str)
- **inherit** (bool) – Whether the new *URL* should inherit the query string and fragment from this *URL*. Default True.

Return type *~URLType*

Changed in version 0.7.0: Added the inherit parameter.

class URLPath(**args*)
Bases: PurePosixPath

Represents the path part of a URL.

Subclass of pathlib.PurePosixPath that provides a subset of its methods.

Changed in version 1.1.0: Implemented *is_absolute()*, *joinpath()*, *relative_to()*, match(), anchor, drive, and support for rich comparisons (<, <=, > and >=), which previously raised NotImplementedError.

Methods:

__bytes__()	Return the bytes representation of the path.
__eq__(other)	Return `self == other`.
__repr__()	Return a string representation of the *URLPath*.
__rtruediv__(key)	Return `value / self`.
__str__()	Return the string representation of the path, suitable for passing to system calls.
__truediv__(key)	Return `self / value`.
is_absolute()	Returns whether the path is absolute (i.e.
is_relative_to(*other)	Return True if the path is relative to another path or False.
is_reserved()	Return True if the path contains one of the special names reserved by the system, if any.
joinpath(*args)	Combine this *URLPath* with one or several arguments.
relative_to(*other)	Returns the relative path to another path identified by the passed arguments.
with_name(name)	Return a new path with the file name changed.
with_stem(stem)	Return a new path with the stem changed.
with_suffix(suffix)	Return a new path with the file suffix changed.

Attributes:

name	The final path component, if any.
parent	The logical parent of the path.
parents	A sequence of this path's logical parents.
parts	An object providing sequence-like access to the components in the filesystem path.
root	The root of the path, if any.
stem	The final path component, minus its last suffix.
suffix	The final component's last suffix, if any.
suffixes	A list of the final component's suffixes, if any.

__bytes__()
Return the bytes representation of the path. This is only recommended to use under Unix.

__eq__(*other*)
Return `self == other`.

Return type `bool`

__repr__()
Return a string representation of the *URLPath*.

Return type `str`

__rtruediv__(*key*)
Return `value / self`.

__str__()
Return the string representation of the path, suitable for passing to system calls.

Return type `str`

__truediv__(*key*)
Return `self / value`.

is_absolute()
Returns whether the path is absolute (i.e. starts with `/`).

New in version 1.1.0: previously raised `NotImplementedError`.

Return type `bool`

is_relative_to(**other*)
Return True if the path is relative to another path or False.

is_reserved()
Return True if the path contains one of the special names reserved by the system, if any.

joinpath(**args*)
Combine this *URLPath* with one or several arguments.

New in version 1.1.0: previously raised `NotImplementedError`.

Return type *~URLPathType*

Returns A new *URLPath* representing either a subpath (if all arguments are relative paths) or a totally different path (if one of the arguments is absolute).

property name
The final path component, if any.

property parent
The logical parent of the path.

property parents
A sequence of this path's logical parents.

property parts
An object providing sequence-like access to the components in the filesystem path.

relative_to(**other*)
Returns the relative path to another path identified by the passed arguments.

The arguments are joined together to form a single path, and therefore the following behave identically:

```
>>> URLPath("/news/sport").relative_to("/", "news")
URLPath('sport')
>>> URLPath("/news/sport").relative_to("/news")
URLPath('sport')
```

New in version 1.1.0: previously raised `NotImplementedError`.

Parameters ***other** (`Union`[`str`, `Path`, `PathLike`])

Raises `ValueError` – if the operation is not possible (because this is not a subpath of the other path)

See also:

relative_to(), which is recommended when constructing a relative path from a *URL*. This method cannot correctly handle some cases, such as:

```
>>> URL("https://github.com/domdfcoding").path.relative_to(URL("https://github.
↪com").path)
Traceback (most recent call last):
ValueError: '/domdfcoding' does not start with ''
```

Since `URL("https://github.com").path` is `URLPath('')`.

Instead, use:

```
>>> URL("https://github.com/domdfcoding").relative_to(URL("https://github.com"))
URLPath('domdfcoding')
```

Return type *~URLPathType*

property root
The root of the path, if any.

property stem
The final path component, minus its last suffix.

property suffix
The final component's last suffix, if any.

This includes the leading period. For example: '.txt'

property suffixes
A list of the final component's suffixes, if any.

These include the leading periods. For example: ['.tar', '.gz']

with_name(*name*)
Return a new path with the file name changed.

with_stem(*stem*)
Return a new path with the stem changed.

with_suffix(*suffix*)
Return a new path with the file suffix changed. If the path has no suffix, add given suffix. If the given suffix is an empty string, remove the suffix from the path.

namedtuple Domain(*subdomain*, *domain*, *suffix*)
`typing.NamedTuple` of a URL's subdomain, domain, and suffix.

Fields

0) **subdomain** (`str`) – Alias for field number 0
1) **domain** (`str`) – Alias for field number 1
2) **suffix** (`str`) – Alias for field number 2

__repr__()
Return a string representation of the *Domain*.

Return type str

property fqdn
Returns a Fully Qualified Domain Name, if there is a proper domain/suffix.

```
>>> URL('https://forums.bbc.co.uk/path/to/file').domain.fqdn
'forums.bbc.co.uk'
>>> URL('https://localhost:8080').domain.fqdn
''
```

property ipv4: Optional[ipaddress.IPv4Address]
Returns the ipv4 if that is what the presented domain/url is.

```
>>> URL('https://127.0.0.1/path/to/file').domain.ipv4
IPv4Address('127.0.0.1')
>>> URL('https://127.0.0.1.1/path/to/file').domain.ipv4
>>> URL('https://256.1.1.1').domain.ipv4
```

Return type Optional[IPv4Address]

property registered_domain
Joins the domain and suffix fields with a dot, if they're both set.

```
>>> URL('https://forums.bbc.co.uk').domain.registered_domain
'bbc.co.uk'
>>> URL('https://localhost:8080').domain.registered_domain
''
```

Chapter

FOUR

apeye.requests_url

Extension of *URL* with support for interacting with the website using the Requests library.

New in version 0.2.0.

Classes:

RequestsURL([url])	Extension of *URL* with support for interacting with the website using the Requests library.
TrailingRequestsURL([url])	Extension of *RequestsURL* which adds a trailing slash to the end of the URL.

Data:

_R	Invariant TypeVar bound to *apeye.requests_url.RequestsURL*.

class RequestsURL(*url=''*)

Bases: *URL*

Extension of *URL* with support for interacting with the website using the Requests library.

The `requests.Session` used for this object – and all objects created using the `/` or `.parent` operations – can be accessed using the *session* attribute. If desired, this can be replaced with a different session object, such as one using caching.

Parameters **url** (Union[str, *URL*]) – The url to construct the *URL* object from. Default `''`.

Changed in version 0.3.0: The `url` parameter can now be a string or a *URL*.

Changed in version 1.1.0: When a *RequestsURL* object is deleted or garbage collected, the underlying `requests.Session` object it only closed if no objects hold references to the session. This prevents the session object of a global object from being inadvertently closed when one of its children is garbage collected.

Methods:

__del__()	Attempt to close session when garbage collected to avoid leaving connections open.
delete(**kwargs)	Send a DELETE request using Requests.
get([params])	Perform a GET request using Requests.
head(**kwargs)	Send a HEAD request using Requests.
options(**kwargs)	Send an OPTIONS request using Requests.
patch([data, json])	Send a PATCH request using Requests.
post([data, json])	Send a POST request using Requests.
put([data, json])	Send a PUT request using Requests.
resolve([timeout])	Resolves the URL into its canonical form.

Attributes:

session	The underlying requests session.

__del__()
Attempt to close session when garbage collected to avoid leaving connections open.

delete(***kwargs*)
Send a DELETE request using Requests.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/DELETE

Parameters ****kwargs** – Optional arguments that `requests.request()` takes.

Return type Response

get(*params=None*, ***kwargs*)
Perform a GET request using Requests.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/GET

Parameters

- **params** (Union[Mapping[Union[str, bytes, int, float], Union[str, bytes, int, float, Iterable[Union[str, bytes, int, float]]]], str, bytes, Tuple[Union[str, bytes, int, float], Union[str, bytes, int, float, Iterable[Union[str, bytes, int, float]]]], None]) – Dictionary, list of tuples or bytes to send in the query string for the `requests.Request`. Default None.
- ****kwargs** – Optional arguments that `requests.request()` takes.

Changed in version 0.7.0: If `params` is None but the URL has a query string, the query string will be parsed and used for `params`.

Return type Response

head(***kwargs*)
Send a HEAD request using Requests.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/HEAD

Parameters ****kwargs** – Optional arguments that `requests.request()` takes. If *allow_redirects* is not provided, it will be set to *False* (as opposed to the default `requests.request()` behavior).

Return type Response

options(***kwargs*)
Send an OPTIONS request using Requests.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/OPTIONS

Parameters ****kwargs** – Optional arguments that `requests.request()` takes.

Return type Response

patch(*data=None*, *json=None*, ***kwargs*)
Send a PATCH request using Requests.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PATCH

Parameters

- **data** (Union[None, str, bytes, MutableMapping[str, Any], List[Tuple[str, Optional[str]]], Tuple[Tuple[str, Optional[str]]], IO]) – Dictionary, list of tuples, bytes, or file-like object to send in the body of the requests.Request. Default None.
- **json** – json data to send in the body of the requests.Request. Default None.
- ****kwargs** – Optional arguments that requests.request() takes.

Return type Response

post(*data=None*, *json=None*, ***kwargs*)
Send a POST request using Requests.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST

Parameters

- **data** (Union[None, str, bytes, MutableMapping[str, Any], List[Tuple[str, Optional[str]]], Tuple[Tuple[str, Optional[str]]], IO]) – Dictionary, list of tuples, bytes, or file-like object to send in the body of the requests.Request. Default None.
- **json** – json data to send in the body of the requests.Request. Default None.
- ****kwargs** – Optional arguments that requests.request() takes.

Return type Response

put(*data=None*, *json=None*, ***kwargs*)
Send a PUT request using Requests.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PUT

Parameters

- **data** (Union[None, str, bytes, MutableMapping[str, Any], List[Tuple[str, Optional[str]]], Tuple[Tuple[str, Optional[str]]], IO]) – Dictionary, list of tuples, bytes, or file-like object to send in the body of the requests.Request. Default None.
- **json** – json data to send in the body of the requests.Request. Default None.
- ****kwargs** – Optional arguments that requests.request() takes.

Return type Response

resolve(*timeout=None*)
Resolves the URL into its canonical form.

This is done by making a HEAD request and following HTTP 302 redirects.

New in version 0.8.0.

Changed in version 1.1.0: Added the timeout argument.

Return type *~_R*

session
Type: Session

The underlying requests session.

class TrailingRequestsURL(*url=''*)
Bases: *RequestsURL*

Extension of *RequestsURL* which adds a trailing slash to the end of the URL.

New in version 0.5.0.

Parameters **url** (Union[str, *URL*]) – The url to construct the *URL* object from. Default ''.

__str__()
Returns the *TrailingRequestsURL* as a string.

Return type str

_R = TypeVar(_R, bound=RequestsURL)
Type: TypeVar

Invariant TypeVar bound to *apeye.requests_url.RequestsURL*.

Chapter

FIVE

apeye.slumber_url

Subclass of *URL* with support for interacting with REST APIs with Slumber and Requests.

New in version 0.2.0.

Classes:

SlumberURL([url, auth, format, ...])	Subclass of *URL* with support for interacting with REST APIs with Slumber and Requests.

class SlumberURL(*url=''*, *auth=None*, *format='json'*, *append_slash=True*, *session=None*, *serializer=None*, ***, *timeout=None*, *allow_redirects=True*, *proxies=None*, *verify=None*, *cert=None*)

Bases: *URL*

Subclass of *URL* with support for interacting with REST APIs with Slumber and Requests.

Parameters

- **url** (Union[str, *URL*]) – The url to construct the *SlumberURL* object from. Default `''`.
- **auth** (Union[None, Tuple[str, str], AuthBase, Callable[[PreparedRequest], PreparedRequest]]) – Default None.
- **format** (str) – Default `'json'`.
- **append_slash** (bool) – Default True.
- **session** – Default None.
- **serializer** (Optional[*SerializerRegistry*]) – Default None.
- **timeout** (Union[None, float, Tuple[float, float], Tuple[float, None]]) – How long to wait for the server to send data before giving up. Default None.
- **allow_redirects** (Optional[bool]) – Whether to allow redirects. Default True.
- **proxies** (Optional[MutableMapping[str, str]]) – Dictionary mapping protocol or protocol and hostname to the URL of the proxy. Default None.
- **verify** (Union[None, bool, str]) – Either a boolean, in which case it controls whether we verify the server's TLS certificate, or a string, in which case it must be a path to a CA bundle to use. Default None.
- **cert** (Union[str, Tuple[str, str], None]) – Either the path to the SSL client cert file (`.pem`), or a tuple of `('cert', 'key')`. Default None.

`timeout`, `allow_redirects`, `proxies`, `verify` and `cert` are passed to Requests when making any HTTP requests, and are inherited by all children created from this URL.

Changed in version 0.3.0: The `url` parameter can now be a string or a *URL*.

Changed in version 1.1.0: When a *RequestsURL* object is deleted or garbage collected, the underlying requests.Session object it only closed if no objects hold references to the session. This prevents the session object of a global object from being inadvertently closed when one of its children is garbage collected.

Methods:

__del__()	Attempt to close session when garbage collected to avoid leaving connections open.
delete(**params)	Perform a DELETE request using Slumber.
get(**params)	Perform a GET request using Slumber.
head(**kwargs)	Send a HEAD request using Requests.
options(**kwargs)	Send an OPTIONS request using Requests.
patch([data, files])	Perform a PATCH request using Slumber.
post([data, files])	Perform a POST request using Slumber.
put([data, files])	Perform a PUT request using Slumber.
url()	Returns the URL as a string.

Attributes:

allow_redirects	Whether to allow redirects.
cert	The path to ssl client cert file or a tuple of ('cert', 'key').
proxies	Dictionary mapping protocol or protocol and hostname to the URL of the proxy.
serializer	The serializer used to (de)serialize the data when interacting with the API.
session	The underlying requests session.
timeout	How long to wait for the server to send data before giving up.
verify	Either a boolean, in which case it controls whether we verify the server's TLS certificate, or a string, in which case it must be a path to a CA bundle to use.

__del__()
Attempt to close session when garbage collected to avoid leaving connections open.

allow_redirects
Type: Optional[bool]

Whether to allow redirects.

cert
Type: Union[str, Tuple[str, str], None]

The path to ssl client cert file or a tuple of ('cert', 'key').

delete(**params*)
Perform a DELETE request using Slumber.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/DELETE

Parameters **params** – Parameters to send in the query string of the requests.Request.

Return type bool

Returns True if the DELETE request succeeded. False otherwise.

get(***params*)
Perform a GET request using Slumber.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/GET

Parameters **params** – Parameters to send in the query string of the requests.Request.

Return type Dict

head(***kwargs*)
Send a HEAD request using Requests.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/HEAD

Parameters **kwargs** – Optional arguments that requests.request() takes. If *allow_redirects* is not provided, it will be set to False (as opposed to the default requests.request() behavior).

Return type CaseInsensitiveDict

options(***kwargs*)
Send an OPTIONS request using Requests.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/OPTIONS

Parameters **kwargs** – Optional arguments that requests.request() takes.

Return type str

patch(*data=None*, *files=None*, ***params*)
Perform a PATCH request using Slumber.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PATCH

Parameters

- **data** – Dictionary, list of tuples, bytes, or file-like object to send in the body of the requests.Request. Default None.
- **files** – Dictionary of 'name': file-like-objects (or {'name': file-tuple}) for multipart encoding upload. file-tuple can be a 2-tuple ('filename', fileobj), 3-tuple ('filename', fileobj, 'content_type') or a 4-tuple ('filename', fileobj, 'content_type', custom_headers), where 'content-type' is a string defining the content type of the given file and custom_headers a dict-like object containing additional headers to add for the file. Default None.
- **params** – Parameters to send in the query string of the requests.Request.

Return type Dict

post(*data=None*, *files=None*, ***params*)
Perform a POST request using Slumber.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST

Parameters

- **data** (Union[None, str, bytes, MutableMapping[str, Any], List[Tuple[str, Optional[str]]], Tuple[Tuple[str, Optional[str]]], IO]) – Dictionary, list of tuples, bytes, or file-like object to send in the body of the requests.Request. Default None.
- **files** – Dictionary of 'name': file-like-objects (or {'name': file-tuple}) for multipart encoding upload. file-tuple can be a 2-tuple ('filename', fileobj), 3-tuple

('filename', fileobj, 'content_type') or a 4-tuple ('filename', fileobj, 'content_type', custom_headers), where 'content-type' is a string defining the content type of the given file and custom_headers a dict-like object containing additional headers to add for the file. Default None.

- **params** – Parameters to send in the query string of the requests.Request.

Return type Dict

proxies
Type: Optional[MutableMapping[str, str]]

Dictionary mapping protocol or protocol and hostname to the URL of the proxy.

put(*data=None*, *files=None*, ***params*)
Perform a PUT request using Slumber.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PUT

Parameters

- **data** – Dictionary, list of tuples, bytes, or file-like object to send in the body of the requests.Request. Default None.
- **files** – Dictionary of 'name': file-like-objects (or {'name': file-tuple}) for multipart encoding upload. file-tuple can be a 2-tuple ('filename', fileobj), 3-tuple ('filename', fileobj, 'content_type') or a 4-tuple ('filename', fileobj, 'content_type', custom_headers), where 'content-type' is a string defining the content type of the given file and custom_headers a dict-like object containing additional headers to add for the file. Default None.
- **params** – Parameters to send in the query string of the requests.Request.

Return type Dict

serializer
Type: *SerializerRegistry*

The serializer used to (de)serialize the data when interacting with the API.

New in version 0.6.0.

session
Type: Session

The underlying requests session.

New in version 0.6.0.

timeout
Type: Union[None, float, Tuple[float, float], Tuple[float, None]]

How long to wait for the server to send data before giving up.

url()
Returns the URL as a string.

Return type str

verify
Type: Union[None, bool, str]

Either a boolean, in which case it controls whether we verify the server's TLS certificate, or a string, in which case it must be a path to a CA bundle to use.

5.2 slumber_url.serializers

JSON and YAML serializers for *SlumberURL*.

New in version 0.6.0.

Classes:

JsonSerializer()	Serializer for JSON data.
Serializer()	Base class for serializers.
SerializerRegistry([default_serializers])	Serializes and deserializes data for transfer to and from a REST API.
YamlSerializer()	Serializer for YAML data.

Exceptions:

SerializerNotAvailable(content_type)	The requested *Serializer* is not available.

class JsonSerializer
Bases: *Serializer*

Serializer for JSON data.

Changed in version 0.6.0: Moved to *apeye.slumber_url.serializers*

Methods:

dumps(data)	Serialize data using this *Serializer*.
loads(data)	Deserialize data using this *Serializer*.

dumps(*data*)
Serialize data using this *Serializer*.

Parameters **data** (Mapping[str, Any])

Return type str

loads(*data*)
Deserialize data using this *Serializer*.

Parameters **data** (str)

Return type MutableMapping[str, Any]

class Serializer
Bases: ABC

Base class for serializers.

Changed in version 0.6.0: Moved to *apeye.slumber_url.serializers*

Attributes:

content_types	List of supported content types.
key	An identifier for the supported data type.

Methods:

dumps(data)	Serialize data using this *Serializer*.
get_content_type()	Returns the first value from *content_types*.
loads(data)	Deserialize data using this *Serializer*.

abstract property content_types: List[str]
List of supported content types.

Return type List[str]

abstract dumps(*data*)
Serialize data using this *Serializer*.

Parameters **data** (Mapping[str, Any])

Return type str

get_content_type()
Returns the first value from *content_types*.

Return type str

abstract property key: str
An identifier for the supported data type.

For example, a YAML serializer would set this to 'yaml'.

Return type str

abstract loads(*data*)
Deserialize data using this *Serializer*.

Parameters **data** (str)

Return type MutableMapping[str, Any]

exception SerializerNotAvailable(*content_type*)
Bases: *apeye.slumber_url.exceptions.SlumberBaseException*

The chosen *Serializer* is not available.

Changed in version 0.6.0: Moved to *apeye.slumber_url.serializers*

class SerializerRegistry(*default='json'*, *serializers=None*)

Bases: object

Serializes and deserializes data for transfer to and from a REST API.

Parameters

- **default** (str) – The default serializer to use if none is specified. Corresponds to the *key* of a *Serializer*. Default 'json'.
- **serializers** (Optional[List[*Serializer*]]) – List of *Serializer* objects to use. Default None.

Changed in version 0.6.0: Moved to *apeye.slumber_url.serializers*

Attributes:

default	The default serializer to use if none is specified.
serializers	Mapping of formats to *Serializer* objects.

Methods:

dumps(data[, format])	Serialize data of the given format.
get_content_type([format])	Returns the content type for the serializer that supports the given format.
get_serializer([name, content_type])	Returns the first *Serializer* that supports either the given format or the given content type.
loads(data[, format])	Deserialize data of the given format.

default

Type: str

The default serializer to use if none is specified.

dumps(*data*, *format=None*)

Serialize data of the given format.

Parameters

- **data** (Mapping[str, Any])
- **format** (Optional[str]) – The serialization format to use. Default None.

Return type str

get_content_type(*format=None*)

Returns the content type for the serializer that supports the given format.

Parameters **format** (Optional[str]) – The desired serialization format. Default None.

get_serializer(*name=None*, *content_type=None*)

Returns the first *Serializer* that supports either the given format or the given content type.

Parameters

- **name** (Optional[str]) – Default None.
- **content_type** (Optional[str]) – Default None.

loads(*data*, *format=None*)
Deserialize data of the given format.

Parameters

- **data** (str)
- **format** (Optional[str]) – The serialization format to use. Default None.

Return type MutableMapping[str, Any]

serializers
Type: Dict[str, *Serializer*]

Mapping of formats to *Serializer* objects.

class YamlSerializer
Bases: *Serializer*

Serializer for YAML data.

Changed in version 0.6.0: Moved to *apeye.slumber_url.serializers*

Attention: Either PyYaml or ruamel.yaml must be installed to use this serializer.

Methods:

dumps(data)	Serialize data using this *Serializer*.
loads(data)	Deserialize data using this *Serializer*.

dumps(*data*)
Serialize data using this *Serializer*.

Parameters **data** (Mapping[str, Any])

Return type str

loads(*data*)
Deserialize data using this *Serializer*.

Parameters **data** (str)

Return type MutableMapping[str, Any]

5.3 slumber_url.exceptions

Exceptions for *SlumberURL*.

New in version 0.6.0.

Exceptions:

HttpClientError(*args, **kwargs)	Raised when the server tells us there was a client error (4xx).
HttpNotFoundError(*args, **kwargs)	Raised when the server sends a 404 error.
HttpServerError(*args, **kwargs)	Raised when the server tells us there was a server error (5xx).
SlumberBaseException	All Slumber exceptions inherit from this exception.
SlumberHttpBaseException(*args, **kwargs)	All Slumber HTTP Exceptions inherit from this exception.

exception HttpClientError(**args*, ***kwargs*)

Bases: *apeye.slumber_url.exceptions.SlumberHttpBaseException*

Raised when the server tells us there was a client error (4xx).

Changed in version 0.6.0: Moved to *apeye.slumber_url.exceptions*

exception HttpNotFoundError(**args*, ***kwargs*)

Bases: *apeye.slumber_url.exceptions.HttpClientError*

Raised when the server sends a 404 error.

Changed in version 0.6.0: Moved to *apeye.slumber_url.exceptions*

exception HttpServerError(**args*, ***kwargs*)

Bases: *apeye.slumber_url.exceptions.SlumberHttpBaseException*

Raised when the server tells us there was a server error (5xx).

Changed in version 0.6.0: Moved to *apeye.slumber_url.exceptions*

exception SlumberBaseException

Bases: Exception

All Slumber exceptions inherit from this exception.

Changed in version 0.6.0: Moved to *apeye.slumber_url.exceptions*

exception SlumberHttpBaseException(**args*, ***kwargs*)

Bases: *apeye.slumber_url.exceptions.SlumberBaseException*

All Slumber HTTP Exceptions inherit from this exception.

Changed in version 0.6.0: Moved to *apeye.slumber_url.exceptions*

Chapter

SIX

apeye.cache

Caching functions for functions.

See also:

- The cachier project
- DiskCache

class Cache(*app_name*)

Bases: object

Cache function arguments to and in-memory dictionary and a JSON file.

Parameters **app_name** (str) – The name of the app. This dictates the name of the cache directory.

Methods:

__call__(func)	Decorator to cache the return values of a function based on its inputs.
clear([func])	Clear the cache.
load_cache(func)	Loads the cache for the given function.

Attributes:

app_name	The name of the app.
cache_dir	The location of the cache directory on disk.
caches	Mapping of function names to their caches.

__call__(*func*)

Decorator to cache the return values of a function based on its inputs.

Parameters **func** (Callable)

app_name

Type: str

The name of the app. This dictates the name of the cache directory.

cache_dir

Type: PathPlus

The location of the cache directory on disk.

caches
Type: Dict[str, Dict[str, Any]]

Mapping of function names to their caches.

clear(*func=None*)
Clear the cache.

Parameters **func** (Optional[Callable]) – Optional function to clear the cache for. By default, the whole cache is cleared.

Return type bool

Returns True to indicate success. False otherwise.

load_cache(*func*)
Loads the cache for the given function.

Parameters **func** (Callable)

Chapter

SEVEN

apeye.email_validator

Source code: apeye_core/email_validator.py

Email address validation functions.

New in version 1.0.0.

This module is a subset of `https://pypi.org/project/email-validator`

Note: The classes in this module can instead be imported from the apeye_core.email_validator module instead.

Exceptions:

EmailSyntaxError	Exception raised when an email address fails validation because of its form.

Classes:

ValidatedEmail(original_email, email, ..., [, ...])	Represents the return type of the *validate_email()* function.

Functions:

validate_email(email[, allow_smtputf8, ...])	Validates an email address.
validate_email_domain_part(domain)	Validate the domain part of an email address (the part after the @-sign).
validate_email_local_part(local[, ...])	Validates the local part of an email address (the part before the @-sign).

exception EmailSyntaxError

Bases: ValueError

Exception raised when an email address fails validation because of its form.

class ValidatedEmail(*original_email*, *email*, *local_part*, *domain*, ***, *ascii_email=None*, *ascii_local_part=None*, *ascii_domain=None*, *smtputf8=None*)

Bases: object

Represents the return type of the *validate_email()* function.

This class holds the normalized form of the email address alongside other information.

Parameters

- **original_email** (str) – The original, unnormalized email address.
- **email** (str) – The normalized email address, which should always be used in preference to the original address.
- **local_part** (str) – The local part of the email address after Unicode normalization.
- **domain** (str) – The domain part of the email address after Unicode normalization or conversion to Unicode from IDNA ascii.
- **ascii_email** (Optional[str]) – If not None, a form of the email address that uses 7-bit ASCII characters only. Default None.
- **ascii_local_part** (Optional[str]) – If not None, the local part of the email address using 7-bit ASCII characters only. Default None.
- **ascii_domain** (Optional[str]) – If not None, a form of the domain name that uses 7-bit ASCII characters only. Default None.
- **smtputf8** (Optional[bool]) – Indicates whether SMTPUTF8 will be required to transmit messages to this address. Default None.

Methods:

__eq__(other)	Return `self == other`.
__repr__()	Return a string representation of the *ValidatedEmail* object.
__str__()	Return a string representation of the *ValidatedEmail* object.
as_dict()	Convenience method for accessing the *ValidatedEmail* as a dict.

__eq__(*other*)

Return `self == other`.

Return type bool

__repr__()

Return a string representation of the *ValidatedEmail* object.

Return type str

__str__()

Return a string representation of the *ValidatedEmail* object.

Return type str

as_dict()

Convenience method for accessing the *ValidatedEmail* as a dict.

Return type Dict[str, Any]

validate_email(*email*, *allow_smtputf8=True*, *allow_empty_local=False*)
Validates an email address.

Parameters

- **email** (Union[str, bytes]) – Either a string, or ASCII-encoded bytes.
- **allow_smtputf8** (bool) – Default True.
- **allow_empty_local** (bool) – Whether to allow the local part (the bit before the @-sign) to be empty. Default False.

Raises *EmailSyntaxError* – if the address is not valid

Return type *ValidatedEmail*

validate_email_domain_part(*domain*)
Validate the domain part of an email address (the part after the @-sign).

Parameters **domain** (str)

Return type Dict[str, str]

validate_email_local_part(*local*, *allow_smtputf8=True*, *allow_empty_local=False*)
Validates the local part of an email address (the part before the @-sign).

Parameters

- **local** (str)
- **allow_smtputf8** (bool) – Default True.
- **allow_empty_local** (bool) – Whether to allow the local part to be empty/. Default False.

Return type Dict[str, Any]

Chapter

EIGHT

apeye.rate_limiter

Rate limiters for making calls to external APIs in a polite manner.

Attention: This module has the following additional requirements:

```
cachecontrol[filecache]>=0.12.6
filelock>=3.8.0; python_version >= "3.7"
lockfile>=0.12.2; python_version < "3.7"
```

These can be installed as follows:

```
$ python -m pip install apeye[limiter]
```

Classes:

HTTPCache(app_name[, expires_after])	Cache HTTP requests for up to 28 days and limit the rate of requests to no more than 5/second.
RateLimitAdapter([cache, cache_etags, ...])	Custom `cachecontrol.adapter.CacheControlAdapter` to limit the rate of requests to 5 per second.

Functions:

rate_limit([min_time, logger])	Decorator to force a function to run no less than `min_time` seconds after it last ran.

class HTTPCache(*app_name*, *expires_after=datetime.timedelta(days=28)*)

Cache HTTP requests for up to 28 days and limit the rate of requests to no more than 5/second.

Parameters

- **app_name** (*str*) – The name of the app. This dictates the name of the cache directory.
- **expires_after** (*timedelta*) – The maximum time to cache responses for. Default `datetime.timedelta(days=28)`.

Attributes:

app_name	The name of the app.
cache_dir	The location of the cache directory on disk.
caches	Mapping of function names to their caches.

Methods:

clear()	Clear the cache.

app_name
Type: str

The name of the app. This dictates the name of the cache directory.

cache_dir
Type: PathPlus

The location of the cache directory on disk.

caches
Type: Dict[str, Dict[str, Any]]

Mapping of function names to their caches.

clear()
Clear the cache.

Return type bool

Returns True to indicate success. False otherwise.

rate_limit(*min_time=0.2*, *logger=None*)
Decorator to force a function to run no less than `min_time` seconds after it last ran. Used for rate limiting.

Parameters

- **min_time** (float) – The minimum interval between subsequent runs of the decorated function. Default `0.2`, which gives a maximum rate of 5 calls per second.
- **logger** (Optional[Logger]) – Optional logger to log information about requests to. Defaults to the root logger.

Return type Callable[[Callable], Any]

class RateLimitAdapter(*cache=None*, *cache_etags=True*, *controller_class=None*, *serializer=None*, *heuristic=None*, *cacheable_methods=None*, **args*, ***kw*)
Bases: CacheControlAdapter

Custom `cachecontrol.adapter.CacheControlAdapter` to limit the rate of requests to 5 per second.

Parameters

- **cache** (BaseCache | None) – Default None.
- **cache_etags** (bool) – Default True.
- **controller_class** (type[CacheController] | None) – Default None.
- **serializer** (Serializer | None) – Default None.
- **heuristic** (BaseHeuristic | None) – Default None.
- **cacheable_methods** (Collection[str] | None) – Default None.

Methods:

rate_limited_send(*args, **kwargs)	Wrapper around `CacheControlAdapter.send` to limit the rate of requests.
send(request[, cacheable_methods])	Send a request.

rate_limited_send(**args*, ***kwargs*)

Wrapper around `CacheControlAdapter.send` to limit the rate of requests.

Return type Response

send(*request*, *cacheable_methods=None*, ***kwargs*)

Send a request.

Use the request information to see if it exists in the cache and cache the response if we need to and can.

Parameters

- **request** (PreparedRequest) – The `requests.PreparedRequest` being sent.
- **cacheable_methods** (Optional[Collection[str]]) – Default None.
- ****kwargs** – Additional arguments taken by `requests.adapters.HTTPAdapter.send()` (e.g. `timeout`).

Return type Response

Python Module Index

a

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Y