pypi-simple

Release 1.6.0.dev1

John T. Wodder II

2024 May 02

CONTENTS

GitHub | PyPI | Documentation | Issues | *Changelog*

CHAPTER

ONE

API

1.1 Client

class pypi_simple.**PyPISimple**(*endpoint: str = 'https://pypi.org/simple/'*, *auth: Any = None*, *session: Session | None = None*, *accept: str = 'application/vnd.pypi.simple.v1+json, application/vnd.pypi.simple.v1+html, text/html;q=0.01'*)

A client for fetching package information from a Python simple package repository.

If necessary, login/authentication details for the repository can be specified at initialization by setting the `auth` parameter to either a `(username, password)` pair or another authentication object accepted by requests.

If more complicated session configuration is desired (e.g., setting up caching), the user must create & configure a `requests.Session` object appropriately and pass it to the constructor as the `session` parameter.

A *PyPISimple* instance can be used as a context manager that will automatically close its session on exit, regardless of where the session object came from.

Changed in version 1.0.0: `accept` parameter added

Parameters

- **endpoint** (*str*) – The base URL of the simple API instance to query; defaults to the base URL for PyPI's simple API
- **auth** – Optional login/authentication details for the repository; either a `(username, password)` pair or another authentication object accepted by requests
- **session** – Optional `requests.Session` object to use instead of creating a fresh one
- **accept** (*str*) – The *Accept* header to send in requests in order to specify what serialization format the server should return; defaults to *ACCEPT_ANY*

get_index_page(*timeout: float | tuple[float, float] | None = None*, *accept: str | None = None*, *headers: dict[str, str] | None = None*) → *IndexPage*

Fetches the index/root page from the simple repository and returns an *IndexPage* instance.

> **Warning:** PyPI's project index file is very large and takes several seconds to parse. Use this method sparingly.

Changed in version 1.0.0: `accept` parameter added

Changed in version 1.5.0: `headers` parameter added

Parameters

- **timeout** (*float | tuple[float,float] | None*) – optional timeout to pass to the `requests` call
- **accept** (*Optional[str]*) – The *Accept* header to send in order to specify what serialization format the server should return; defaults to the value supplied on client instantiation
- **headers** (*Optional[dict[str, str]]*) – Custom headers to provide for the request.

Return type
IndexPage

Raises

- **requests.HTTPError** – if the repository responds with an HTTP error code
- ***UnsupportedContentTypeError*** – if the repository responds with an unsupported *Content-Type*
- ***UnsupportedRepoVersionError*** – if the repository version has a greater major component than the supported repository version

stream_project_names(*chunk_size: int = 65535*, *timeout: float | tuple[float, float] | None = None*, *accept: str | None = None*, *headers: dict[str, str] | None = None*) → Iterator[str]

Returns a generator of names of projects available in the repository. The names are not normalized.

Unlike *get_index_page()*, this function makes a streaming request to the server and parses the document in chunks. It is intended to be faster than the other methods, especially when the complete document is very large.

> **Warning:** This function is rather experimental. It does not have full support for web encodings, encoding detection, or handling invalid HTML.

Note: If the server responds with a JSON representation of the Simple API rather than an HTML representation, the response body will be loaded & parsed in its entirety before yielding anything.

Changed in version 1.0.0: `accept` parameter added

Changed in version 1.5.0: `headers` parameter added

Parameters

- **chunk_size** (*int*) – how many bytes to read from the response at a time
- **timeout** (*float | tuple[float,float] | None*) – optional timeout to pass to the `requests` call
- **accept** (*Optional[str]*) – The *Accept* header to send in order to specify what serialization format the server should return; defaults to the value supplied on client instantiation
- **headers** (*Optional[dict[str, str]]*) – Custom headers to provide for the request.

Return type
Iterator[str]

Raises

- **requests.HTTPError** – if the repository responds with an HTTP error code
- ***UnsupportedContentTypeError*** – if the repository responds with an unsupported *Content-Type*

- *UnsupportedRepoVersionError* – if the repository version has a greater major component than the supported repository version

get_project_page(*project: str*, *timeout: float | tuple[float, float] | None = None*, *accept: str | None = None*, *headers: dict[str, str] | None = None*) → *ProjectPage*

Fetches the page for the given project from the simple repository and returns a *ProjectPage* instance. Raises *NoSuchProjectError* if the repository responds with a 404. All other HTTP errors cause a `requests.HTTPError` to be raised.

Changed in version 1.0.0:

- A 404 now causes *NoSuchProjectError* to be raised instead of returning `None`
- `accept` parameter added

Changed in version 1.5.0: `headers` parameter added

Parameters

- **project** (*str*) – The name of the project to fetch information on. The name does not need to be normalized.
- **timeout** (*float | tuple[float,float] | None*) – optional timeout to pass to the `requests` call
- **accept** (*Optional[str]*) – The *Accept* header to send in order to specify what serialization format the server should return; defaults to the value supplied on client instantiation
- **headers** (*Optional[dict[str, str]]*) – Custom headers to provide for the request.

Return type
ProjectPage

Raises

- *NoSuchProjectError* – if the repository responds with a 404 error code
- **requests.HTTPError** – if the repository responds with an HTTP error code other than 404
- *UnsupportedContentTypeError* – if the repository responds with an unsupported *Content-Type*
- *UnsupportedRepoVersionError* – if the repository version has a greater major component than the supported repository version

get_project_url(*project: str*) → str

Returns the URL for the given project's page in the repository.

Parameters
project (*str*) – The name of the project to build a URL for. The name does not need to be normalized.

Return type
str

download_package(*pkg: DistributionPackage*, *path: AnyStr | PathLike*, *verify: bool = True*, *keep_on_error: bool = False*, *progress: Callable[[int | None], ProgressTracker] | None = None*, *timeout: float | tuple[float, float] | None = None*, *headers: dict[str, str] | None = None*) → None

Download the given *DistributionPackage* to the given path.

If an error occurs while downloading or verifying digests, and `keep_on_error` is not true, the downloaded file is not saved.

Download progress can be tracked (e.g., for display by a progress bar) by passing an appropriate callable as the `progress` argument. This callable will be passed the length of the downloaded file, if known, and it must return a *ProgressTracker* — a context manager with an `update(increment: int)` method that will be passed the size of each downloaded chunk as each chunk is received.

Changed in version 1.5.0: `headers` parameter added

Parameters

- **pkg** (DistributionPackage) – the distribution package to download
- **path** – the path at which to save the downloaded file; any parent directories of this path will be created as needed
- **verify** (*bool*) – whether to verify the package's digests against the downloaded file
- **keep_on_error** (*bool*) – whether to keep (true) or delete (false) the downloaded file if an error occurs
- **progress** – a callable for constructing a progress tracker
- **timeout** (*float | tuple[float,float] | None*) – optional timeout to pass to the `requests` call
- **headers** (*Optional[dict[str, str]]*) – Custom headers to provide for the request.

Raises

- **requests.HTTPError** – if the repository responds with an HTTP error code
- ***NoDigestsError*** – if `verify` is true and the given package does not have any digests with known algorithms
- ***DigestMismatchError*** – if `verify` is true and the digest of the downloaded file does not match the expected value

get_package_metadata_bytes(*pkg:* DistributionPackage, *verify: bool = True*, *timeout: float | tuple[float, float] | None = None*, *headers: dict[str, str] | None = None*) → bytes

Added in version 1.5.0.

Retrieve the distribution metadata for the given *DistributionPackage* as raw bytes. This method is lower-level than *PyPISimple.get_package_metadata()* and is most appropriate if you want to defer interpretation of the data (e.g., if you're just writing to a file) or want to customize the handling of non-UTF-8 data.

Not all packages have distribution metadata available for download; the *DistributionPackage.has_metadata* attribute can be used to check whether the repository reported the availability of the metadata. This method will always attempt to download metadata regardless of the value of *has_metadata*; if the server replies with a 404, a *NoMetadataError* is raised.

Parameters

- **pkg** (DistributionPackage) – the distribution package to retrieve the metadata of
- **verify** (*bool*) – whether to verify the metadata's digests against the retrieved data
- **timeout** (*float | tuple[float,float] | None*) – optional timeout to pass to the `requests` call
- **headers** (*Optional[dict[str, str]]*) – Custom headers to provide for the request.

Return type
bytes

Raises

- *NoMetadataError* – if the repository responds with a 404 error code
- requests.HTTPError – if the repository responds with an HTTP error code other than 404
- *NoDigestsError* – if verify is true and the given package's metadata does not have any digests with known algorithms
- *DigestMismatchError* – if verify is true and the digest of the downloaded data does not match the expected value

get_package_metadata(*pkg:* DistributionPackage, *verify: bool = True*, *timeout: float | tuple[float, float] | None = None*, *headers: dict[str, str] | None = None*) → str

Added in version 1.3.0.

Retrieve the distribution metadata for the given *DistributionPackage* and decode it as UTF-8. The metadata can then be parsed with, for example, the packaging package.

Not all packages have distribution metadata available for download; the *DistributionPackage.has_metadata* attribute can be used to check whether the repository reported the availability of the metadata. This method will always attempt to download metadata regardless of the value of *has_metadata*; if the server replies with a 404, a *NoMetadataError* is raised.

Changed in version 1.5.0: headers parameter added

Parameters

- **pkg** (DistributionPackage) – the distribution package to retrieve the metadata of
- **verify** (*bool*) – whether to verify the metadata's digests against the retrieved data
- **timeout** (*float | tuple[float,float] | None*) – optional timeout to pass to the requests call
- **headers** (*Optional[dict[str, str]]*) – Custom headers to provide for the request.

Return type

str

Raises

- *NoMetadataError* – if the repository responds with a 404 error code
- requests.HTTPError – if the repository responds with an HTTP error code other than 404
- *NoDigestsError* – if verify is true and the given package's metadata does not have any digests with known algorithms
- *DigestMismatchError* – if verify is true and the digest of the downloaded data does not match the expected value

1.2 Core Classes

class pypi_simple.**IndexPage**

A parsed index/root page from a simple repository

projects: `list[str]`

The project names listed in the index. The names are not normalized.

repository_version: `str | None`

The repository version reported by the page, or `None` if not specified

last_serial: `str | None`

The value of the *X-PyPI-Last-Serial* response header returned when fetching the page, or `None` if not specified

classmethod from_html(*html: str | bytes*, *from_encoding: str | None = None*) → *IndexPage*

Added in version 1.0.0.

Parse an HTML index/root page from a simple repository into an *IndexPage*. Note that the *last_serial* attribute will be `None`.

Parameters

- **html** (*str or bytes*) – the HTML to parse
- **from_encoding** (*Optional[str]*) – an optional hint to Beautiful Soup as to the encoding of `html` when it is `bytes` (usually the `charset` parameter of the response's *Content-Type* header)

Return type
IndexPage

Raises
UnsupportedRepoVersionError – if the repository version has a greater major component than the supported repository version

classmethod from_json_data(*data: Any*) → *IndexPage*

Added in version 1.0.0.

Parse an object decoded from an *application/vnd.pypi.simple.v1+json* response (See **PEP 691**) into an *IndexPage*. The *last_serial* attribute will be set to the value of the `.meta._last-serial` field, if any.

Parameters
data – The decoded body of the JSON response

Return type
IndexPage

Raises

- ***UnsupportedRepoVersionError*** – if the repository version has a greater major component than the supported repository version
- **ValueError** – if `data` is not a `dict`

classmethod from_response(*r: Response*) → *IndexPage*

Added in version 1.0.0.

Parse an index page from a `requests.Response` returned from a (non-streaming) request to a simple repository, and return an *IndexPage*.

Parameters
r (*requests.Response*) – the response object to parse

Return type
IndexPage

Raises

- ***UnsupportedRepoVersionError*** – if the repository version has a greater major component than the supported repository version
- ***UnsupportedContentTypeError*** – if the response has an unsupported *Content-Type*

class pypi_simple.**ProjectPage**

A parsed project page from a simple repository

project: `str`

The name of the project the page is for

packages: `list[DistributionPackage]`

A list of packages (as *DistributionPackage* objects) listed on the project page

repository_version: `str | None`

The repository version reported by the page, or `None` if not specified

last_serial: `str | None`

The value of the *X-PyPI-Last-Serial* response header returned when fetching the page, or `None` if not specified

versions: `list[str] | None = None`

Added in version 1.1.0.

A list of the project's versions, or `None` if not specified[1].

tracks: `list[str]`

Added in version 1.4.0.

Repository "tracks" metadata. See PEP 708.

alternate_locations: `list[str]`

Added in version 1.4.0.

Repository "alternate locations" metadata. See PEP 708.

classmethod from_html(*project: str*, *html: str | bytes*, *base_url: str | None = None*, *from_encoding: str | None = None*) → *ProjectPage*

Added in version 1.0.0.

Parse an HTML project page from a simple repository into a *ProjectPage*. Note that the *last_serial* attribute will be `None`.

Parameters

- **project** (*str*) – The name of the project whose page is being parsed
- **html** (*str or bytes*) – the HTML to parse
- **base_url** (*Optional[str]*) – an optional URL to join to the front of the packages' URLs (usually the URL of the page being parsed)

[1] The `versions`, `size`, and `upload_time` fields are only populated if the response was JSON from a server supporting **PEP 700**.

- **from_encoding** (*Optional[str]*) – an optional hint to Beautiful Soup as to the encoding of `html` when it is `bytes` (usually the `charset` parameter of the response's *Content-Type* header)

Return type
ProjectPage

Raises
UnsupportedRepoVersionError – if the repository version has a greater major component than the supported repository version

classmethod from_json_data(*data: Any*, *base_url: str | None = None*) → *ProjectPage*

Added in version 1.0.0.

Parse an object decoded from an *application/vnd.pypi.simple.v1+json* response (See **PEP 691**) into a *ProjectPage*. The *last_serial* attribute will be set to the value of the `.meta._last-serial` field, if any.

Parameters

- **data** – The decoded body of the JSON response
- **base_url** (*Optional[str]*) – an optional URL to join to the front of any relative file URLs (usually the URL of the page being parsed)

Return type
ProjectPage

Raises

- **ValueError** – if `data` is not a `dict`
- ***UnsupportedRepoVersionError*** – if the repository version has a greater major component than the supported repository version

classmethod from_response(*r: Response*, *project: str*) → *ProjectPage*

Added in version 1.0.0.

Parse a project page from a `requests.Response` returned from a (non-streaming) request to a simple repository, and return a *ProjectPage*.

Parameters

- **r** (*requests.Response*) – the response object to parse
- **project** (*str*) – the name of the project whose page is being parsed

Return type
ProjectPage

Raises

- ***UnsupportedRepoVersionError*** – if the repository version has a greater major component than the supported repository version
- ***UnsupportedContentTypeError*** – if the response has an unsupported *Content-Type*

class pypi_simple.**DistributionPackage**

Information about a versioned archive file from which a Python project release can be installed

Changed in version 1.0.0: `yanked` field replaced with *is_yanked* and *yanked_reason*

filename: **str**

The basename of the package file

url: `str`

The URL from which the package file can be downloaded, with any hash digest fragment removed

project: `str` | `None`

The name of the project (as extracted from the filename), or `None` if the filename cannot be parsed

version: `str` | `None`

The project version (as extracted from the filename), or `None` if the filename cannot be parsed

package_type: `str` | `None`

The type of the package, or `None` if the filename cannot be parsed. The recognized package types are:

- `'dumb'`
- `'egg'`
- `'msi'`
- `'rpm'`
- `'sdist'`
- `'wheel'`
- `'wininst'`

digests: `dict[str, str]`

A collection of hash digests for the file as a `dict` mapping hash algorithm names to hex-encoded digest strings

requires_python: `str` | `None`

An optional version specifier string declaring the Python version(s) in which the package can be installed

has_sig: `bool` | `None`

Whether the package file is accompanied by a PGP signature file. This is `None` if the package repository does not report such information.

is_yanked: `bool` = **False**

Whether the package file has been "yanked" from the package repository (meaning that it should only be installed when that specific version is requested)

yanked_reason: `str` | `None` = **None**

If the package file has been "yanked" and a reason is given, this attribute will contain that (possibly empty) reason

has_metadata: `bool` | `None` = **None**

Whether the package file is accompanied by a Core Metadata file. This is `None` if the package repository does not report such information.

metadata_digests: `dict[str, str]` | `None` = **None**

If the package repository provides a Core Metadata file for the package, this is a (possibly empty) `dict` of digests of the file, given as a mapping from hash algorithm names to hex-encoded digest strings; otherwise, it is `None`

size: `int` | `None` = **None**

Added in version 1.1.0.

The size of the package file in bytes, or `None` if not specified[Page 9, 1].

upload_time: *datetime* | *None* = None

Added in version 1.1.0.

The time at which the package file was uploaded to the server, or None if not specified[Page 9, 1].

property sig_url: *str*

The URL of the package file's PGP signature file, if it exists; cf. *has_sig*

property metadata_url: *str*

The URL of the package file's Core Metadata file, if it exists; cf. *has_metadata*

classmethod from_link(*link:* Link, *project_hint:* *str* | *None* = *None*) → *DistributionPackage*

Construct a *DistributionPackage* from a *Link* on a project page.

Parameters

- **link** (Link) – a link parsed from a project page
- **project_hint** (*Optional[str]*) – Optionally, the expected value for the project name (usually the name of the project page on which the link was found). The name does not need to be normalized.

Return type
DistributionPackage

classmethod from_json_data(*data:* *Any*, *project_hint:* *str* | *None* = *None*, *base_url:* *str* | *None* = *None*) → *DistributionPackage*

Construct a *DistributionPackage* from an object taken from the "files" field of a **PEP 691** project detail JSON response.

Parameters

- **data** – a file dictionary
- **project_hint** (*Optional[str]*) – Optionally, the expected value for the project name (usually the name of the project page on which the link was found). The name does not need to be normalized.
- **base_url** (*Optional[str]*) – an optional URL to join to the front of a relative file URL (usually the URL of the page being parsed)

Return type
DistributionPackage

Raises
ValueError – if data is not a dict

1.3 Progress Trackers

class pypi_simple.**ProgressTracker**

A typing.Protocol for progress trackers. A progress tracker must be usable as a context manager whose __enter__ method performs startup & returns itself and whose __exit__ method performs shutdown/cleanup. In addition, a progress tracker must have an update(increment: int) method that will be called with the size of each downloaded file chunk.

__enter__() → Self

__exit__(*exc_type: type[BaseException] | None*, *exc_val: BaseException | None*, *exc_tb: TracebackType | None*) → bool | None

update(*increment: int*) → None

pypi_simple.**tqdm_progress_factory**(***kwargs: Any*) → Callable[[int | None], *ProgressTracker*]

A function for displaying a progress bar with tqdm during a download. Naturally, using this requires tqdm to be installed alongside `pypi-simple`.

Call *tqdm_progress_factory()* with any arguments you wish to pass to the `tqdm.tqdm` constructor, and pass the result as the `progress` argument to *PyPISimple.download_package()*.

Example:

```
with PyPISimple() as client:
    page = client.get_project_page("pypi-simple")
    pkg = page.packages[-1]
    client.download_package(
        pkg,
        path=pkg.filename,
        progress=tqdm_progress_factory(desc="Downloading ..."),
    )
```

1.4 Parsing Filenames

pypi_simple.**parse_filename**(*filename: str*, *project_hint: str | None = None*) → tuple[str, str, str]

Given the filename of a distribution package, returns a triple of the project name, project version, and package type. The name and version are spelled the same as they appear in the filename; no normalization is performed.

The package type may be any of the following strings:

- `'dumb'`
- `'egg'`
- `'msi'`
- `'rpm'`
- `'sdist'`
- `'wheel'`
- `'wininst'`

Note that some filenames (e.g., `1-2-3.tar.gz`) may be ambiguous as to which part is the project name and which is the version. In order to resolve the ambiguity, the expected value for the project name (*modulo* normalization) can be supplied as the `project_name` argument to the function. If the filename can be parsed with the given string in the role of the project name, the results of that parse will be returned; otherwise, the function will fall back to breaking the project & version apart at an unspecified point.

Changed in version 1.0.0: Now raises *UnparsableFilenameError* for unparsable filenames instead of returning all `None`s

Parameters

- **filename** (*str*) – The package filename to parse

- **project_hint** (*Optional[str]*) – Optionally, the expected value for the project name (usually the name of the project page on which the filename was found). The name does not need to be normalized.

Return type
tuple[str, str, str]

Raises
UnparsableFilenameError – if the filename cannot be parsed

1.5 Parsing Simple Repository HTML Pages

class pypi_simple.**RepositoryPage**

Added in version 1.0.0.

A parsed HTML page from a **PEP 503** simple repository

repository_version: `str | None`

The repository version, if any, reported by the page in accordance with **PEP 629**

links: `list[Link]`

A list of hyperlinks found on the page

pypi_meta: `dict[str, list[str]]`

Added in version 1.4.0.

<meta/> tags found on the page whose `name` attributes start with `pypi:`. This is a dict in which the keys are `name` attributes with leading `"pypi:"` removed and in which the values are the corresponding `content` attributes.

property tracks: `list[str]`

Added in version 1.4.0.

Repository "tracks" metadata. See PEP 708.

property alternate_locations: `list[str]`

Added in version 1.4.0.

Repository "alternate locations" metadata. See PEP 708.

classmethod from_html(*html: str | bytes*, *base_url: str | None = None*, *from_encoding: str | None = None*) → *RepositoryPage*

Parse an HTML page from a simple repository into a *RepositoryPage*.

Parameters

- **html** (*str or bytes*) – the HTML to parse
- **base_url** (*Optional[str]*) – an optional URL to join to the front of the links' URLs (usually the URL of the page being parsed)
- **from_encoding** (*Optional[str]*) – an optional hint to Beautiful Soup as to the encoding of `html` when it is `bytes` (usually the `charset` parameter of the response's *Content-Type* header)

Return type
RepositoryPage

Raises
UnsupportedRepoVersionError – if the repository version has a greater major component than the supported repository version

class pypi_simple.**Link**

A hyperlink extracted from an HTML page

text: str

The text inside the link tag, with leading & trailing whitespace removed and with any tags nested inside the link tags ignored

url: str

The URL that the link points to, resolved relative to the URL of the source HTML page and relative to the page's <base> href value, if any

attrs: dict[str, str | list[str]]

A dictionary of attributes set on the link tag (including the unmodified `href` attribute). Keys are converted to lowercase. Most attributes have `str` values, but some (referred to as "CDATA list attributes" by the HTML spec; e.g., `"class"`) have values of type `list[str]` instead.

1.5.1 Streaming Parsers

pypi_simple.**parse_links_stream**(*htmlseq: Iterable*, *base_url: str | None = None*, *http_charset: str | None = None*) → Iterator[*Link*]

Parse an HTML page given as an iterable of `bytes` or `str` and yield each hyperlink encountered in the document as a *Link* object.

This function consumes the elements of `htmlseq` one at a time and yields the links found in each segment before moving on to the next one. It is intended to be faster than *RepositoryPage.from_html()*, especially when the complete document is very large.

> **Warning:** This function is rather experimental. It does not have full support for web encodings, encoding detection, or handling invalid HTML. It also leaves CDATA list attributes on links as strings instead of converting them to lists.

Parameters

- **htmlseq** (*Iterable[AnyStr]*) – an iterable of either `bytes` or `str` that, when joined together, form an HTML document to parse
- **base_url** (*Optional[str]*) – an optional URL to join to the front of the links' URLs (usually the URL of the page being parsed)
- **http_charset** (*Optional[str]*) – the document's encoding as declared by the transport layer, if any; e.g., as declared in the `charset` parameter of the *Content-Type* header of the HTTP response that returned the document

Return type
Iterator[*Link*]

Raises
UnsupportedRepoVersionError – if the repository version has a greater major component than the supported repository version

pypi_simple.**parse_links_stream_response**(*r: Response*, *chunk_size: int = 65535*) → Iterator[*Link*]

Parse an HTML page from a streaming `requests.Response` object and yield each hyperlink encountered in the document as a *Link* object.

See *parse_links_stream()* for more information.

Parameters

- **r** (*requests.Response*) – the streaming response object to parse
- **chunk_size** (*int*) – how many bytes to read from the response at a time

Return type
Iterator[*Link*]

Raises
UnsupportedRepoVersionError – if the repository version has a greater major component than the supported repository version

1.6 Constants

pypi_simple.**PYPI_SIMPLE_ENDPOINT**: str = 'https://pypi.org/simple/'

The base URL for PyPI's simple API

pypi_simple.**SUPPORTED_REPOSITORY_VERSION**: str = '1.2'

The maximum supported simple repository version (See **PEP 629**)

1.6.1 *Accept* Header Values

The following constants can be passed as the `accept` parameter of *PyPISimple* and some of its methods in order to indicate to the server which serialization format of the Simple API it should return:

pypi_simple.**ACCEPT_ANY**: str = 'application/vnd.pypi.simple.v1+json, application/vnd.pypi.simple.v1+html, text/html;q=0.01'

Accept header value for accepting either the HTML or JSON serialization without a preference

pypi_simple.**ACCEPT_JSON_ONLY** = 'application/vnd.pypi.simple.v1+json'

Accept header value for accepting only the JSON serialization

pypi_simple.**ACCEPT_HTML_ONLY** = 'application/vnd.pypi.simple.v1+html, text/html;q=0.01'

Accept header value for accepting only the HTML serialization

pypi_simple.**ACCEPT_JSON_PREFERRED** = 'application/vnd.pypi.simple.v1+json, application/vnd.pypi.simple.v1+html;q=0.5, text/html;q=0.01'

Accept header value for accepting either the HTML or JSON serialization with a preference for JSON

pypi_simple.**ACCEPT_HTML_PREFERRED** = 'application/vnd.pypi.simple.v1+html, text/html;q=0.5, application/vnd.pypi.simple.v1+json;q=0.1'

Accept header value for accepting either the HTML or JSON serialization with a preference for HTML

1.7 Exceptions

exception pypi_simple.**DigestMismatchError**

Bases: ValueError

Raised by *PyPISimple.download_package()* and *PyPISimple.get_package_metadata()* with `verify=True` when the digest of the downloaded data does not match the expected value

algorithm

The name of the digest algorithm used

expected_digest

The expected digest

actual_digest

The digest of the data that was actually received

exception pypi_simple.**NoDigestsError**

Bases: ValueError

Raised by *PyPISimple.download_package()* and *PyPISimple.get_package_metadata()* with `verify=True` when the given package or package metadata does not have any digests with known algorithms

exception pypi_simple.**NoMetadataError**

Added in version 1.3.0.

Raised by *PyPISimple.get_package_metadata()* when a request for distribution metadata fails with a 404 error code

filename

The filename of the package whose metadata was requested

exception pypi_simple.**NoSuchProjectError**

Raised by *PyPISimple.get_project_page()* when a request for a project fails with a 404 error code

project

The name of the project requested

url

The URL to which the failed request was made

exception pypi_simple.**UnsupportedContentTypeError**

Bases: ValueError

Raised when a response from a simple repository has an unsupported *Content-Type*

url

The URL that returned the response

content_type

The unsupported *Content-Type*

exception pypi_simple.**UnsupportedRepoVersionError**

Raised upon encountering a simple repository whose repository version (**PEP 629**) has a greater major component than the maximum supported repository version (*SUPPORTED_REPOSITORY_VERSION*)

declared_version: *str*

The version of the simple repository

supported_version: *str*

The maximum repository version that we support

exception pypi_simple.**UnexpectedRepoVersionWarning**

Bases: UserWarning

Emitted upon encountering a simple repository whose repository version (**PEP 629**) has a greater minor version components than the maximum supported repository version (*SUPPORTED_REPOSITORY_VERSION*).

This warning can be emitted by anything that can raise *UnsupportedRepoVersionError*.

exception pypi_simple.**UnparsableFilenameError**

Bases: ValueError

Added in version 1.0.0.

Raised when *parse_filename()* is passed an unparsable filename

filename

The unparsable filename

1.8 Footnotes

CHAPTER

TWO

EXAMPLES

2.1 Getting a Project's Dependencies

pypi_simple can be used to fetch a project's dependencies (specifically, the dependencies for each of the project's distribution packages) as follows.

Note that Warehouse only began storing the contents of package `METADATA` files in May 2023. Packages uploaded prior to that point are gradually having their metadata "backfilled" in; see <https://github.com/pypi/warehouse/issues/8254> for updates.

```
# Requirements:
#     Python 3.8+
#     packaging 23.1+
#     pypi_simple 1.3+

from packaging.metadata import parse_email
from pypi_simple import PyPISimple

with PyPISimple() as client:
    page = client.get_project_page("pypi-simple")
    for pkg in page.packages:
        if pkg.has_metadata:
            src = client.get_package_metadata(pkg)
            md, _ = parse_email(src)
            if deps := md.get("requires_dist"):
                print(f"Dependencies for {pkg.filename}:")
                for d in deps:
                    print(f"    {d}")
            else:
                print(f"Dependencies for {pkg.filename}: NONE")
        else:
            print(f"{pkg.filename}: No metadata available")
        print()
```

2.2 Downloading With a Rich Progress Bar

The `PyPISimple.download_package()` method can be passed a callable for constructing a progress bar to use when downloading. *`pypi_simple`* has built-in support for using a tqdm progress bar, but any progress bar can be used if you provide the right structure.

Here is an example of using a progress bar from rich. The progress bar uses the default settings; adding customization is left as an exercise to the reader.

```python
from __future__ import annotations
from dataclasses import InitVar, dataclass, field
from types import TracebackType
from pypi_simple import PyPISimple
from rich.progress import Progress, TaskID

@dataclass
class RichProgress:
    bar: Progress = field(init=False, default_factory=Progress)
    task_id: TaskID = field(init=False)
    size: InitVar[int | None]

    def __post_init__(self, size: int | None) -> None:
        self.task_id = self.bar.add_task("Downloading...", total=size)

    def __enter__(self) -> RichProgress:
        self.bar.start()
        return self

    def __exit__(
        self,
        _exc_type: type[BaseException] | None,
        _exc_val: BaseException | None,
        _exc_tb: TracebackType | None,
    ) -> None:
        self.bar.stop()

    def update(self, increment: int) -> None:
        self.bar.update(self.task_id, advance=increment)


with PyPISimple() as client:
    page = client.get_project_page("numpy")
    pkg = page.packages[-1]
    client.download_package(pkg, path=pkg.filename, progress=RichProgress)
```

CHAPTER

THREE

CHANGELOG

3.1 v1.6.0 (in development)

- Drop support for Python 3.7

3.2 v1.5.0 (2024-02-24)

- **Bugfix**: Fix parsing of "true" `data-core-metadata` attributes and handling of the attribute's absence (contributed by @thatch)
  - *`DistributionPackage.has_metadata`* will now be `None` if this attribute was absent in the HTML returned by PyPI. Previously, it would be `False` under this circumstance.
- Added *`PyPISimple.get_package_metadata_bytes()`* (contributed by @thatch)
- *`PyPISimple.get_package_metadata()`* now always decodes responses as UTF-8 (contributed by @thatch)
- Request methods now take optional `headers` arguments (contributed by @thatch)

3.3 v1.4.1 (2024-01-30)

- Migrated from setuptools to hatch
- **Bugfix**: When no `accept` argument is passed to *`PyPISimple.get_project_page()`*, actually fall back to the value supplied on client instantiation, as documented (contributed by @thatch)

3.4 v1.4.0 (2023-11-01)

- Support **PEP 708**
  - *`tracks`* and *`alternate_locations`* attributes added to *`ProjectPage`*
  - *`pypi_meta`*, *`tracks`*, and *`alternate_locations`* attributes added to *`RepositoryPage`*
  - *`SUPPORTED_REPOSITORY_VERSION`* increased to `"1.2"`

3.5 v1.3.0 (2023-11-01)

- Support Python 3.12
- Update for PEP 714
- Gave *PyPISimple* a *get_package_metadata()* method
- Added an examples page to the documentation

3.6 v1.2.0 (2023-09-23)

- Update pydantic to v2.0

3.7 v1.1.0 (2023-02-19)

- Support **PEP 700**
  - `versions` field added to *ProjectPage*
  - `size` and `upload_time` fields added to *DistributionPackage*
  - *SUPPORTED_REPOSITORY_VERSION* increased to `"1.1"`

3.8 v1.0.0 (2022-10-31)

- Removed deprecated functionality:
  - `DistributionPackage.get_digests()`
  - `PyPISimple.get_projects()`
  - `PyPISimple.get_project_files()`
  - `parse_simple_index()`
  - `parse_project_page()`
  - `parse_links()`
- Drop support for Python 3.6
- Support Python 3.11
- *IndexPage*, *ProjectPage*, *DistributionPackage*, and *Link* have been changed from NamedTuples to dataclasses
- Replaced `DistributionPackage.yanked` with separate *is_yanked* and *yanked_reason* attributes
- *parse_filename()* now raises an *UnparsableFilenameError* on unparsable filenames instead of returning a triple of `None`s
- *PyPISimple.get_project_page()* now raises a *NoSuchProjectError* on 404 responses instead of returning `None`
- The functions for parsing data into *IndexPage* and *ProjectPage* instances have been replaced with classmethods:

  - `parse_repo_index_page()` → *IndexPage.from_html()*
  - `parse_repo_index_json()` → *IndexPage.from_json_data()*
  - `parse_repo_index_response()` → *IndexPage.from_response()*
  - `parse_repo_links()` → *RepositoryPage.from_html()*
  - `parse_repo_project_page()` → *ProjectPage.from_html()*
  - `parse_repo_project_json()` → *ProjectPage.from_json_data()*
  - `parse_repo_project_response()` → *ProjectPage.from_response()*
- Add a *RepositoryPage* class for representing the return value of `parse_repo_links()` (now called *RepositoryPage.from_html()*)
- Renamed `DistributionPackage.from_pep691_details()` to *from_json_data()*
- *PyPISimple.stream_project_names()* now accepts JSON responses
- Use pydantic internally to parse JSON responses
- Added constants for passing to *PyPISimple* and its methods in order to specify the *Accept* header to send

3.9 v0.10.0 (2022-06-30)

- Support Python 3.10
- Support **PEP 691**
  - Send *Accept* headers in requests (except for *stream_project_names()*) listing both the new JSON format and the old HTML format
  - `parse_repo_project_response()` and `parse_repo_index_response()` now support both the JSON and HTML formats
  - Add `parse_repo_index_json()` and `parse_repo_project_json()` functions
  - Gave *DistributionPackage* a `from_pep691_details()` classmethod
  - *DistributionPackage.has_metadata* will now be `None` if not specified by a JSON response
  - *DistributionPackage.metadata_url* is now always non-`None`
- Gave *DistributionPackage* a *digests* attribute
  - The `get_digests()` method of *DistributionPackage* is now deprecated; use *digests* instead
  - Digest fragments are now removed from *DistributionPackage.url* when parsing HTML responses
- Warn on encountering a repository version with a greater minor version than expected
- Gave *PyPISimple* a *download_package()* method

3.10 v0.9.0 (2021-08-26)

- Support **PEP 658** by adding *has_metadata*, *metadata_url*, and *metadata_digests* attributes to *DistributionPackage*

3.11 v0.8.0 (2020-12-13)

- Support Python 3.9
- *PyPISimple* is now usable as a context manager that will close the session on exit

3.12 v0.7.0 (2020-10-15)

- Drop support for Python 2.7, Python 3.4, and Python 3.5
- *DistributionPackage.has_sig* is now `None` if the package repository does not report this information
- Added type annotations
- Moved documentation from README file to a Read the Docs site
- Added new methods to *PyPISimple*:
  - *get_index_page()* — Returns an *IndexPage* instance with a `projects: List[str]` attribute plus other attributes for repository metadata
  - *get_project_page()* — Returns a *ProjectPage* instance with a `packages: List[DistributionPackage]` attribute plus other attributes for repository metadata
  - *stream_project_names()* — Retrieves project names from a repository using a streaming request
- New utility functions:
  - `parse_repo_links()` — Parses an HTML page and returns a pair of repository metadata and a list of *Link* objects
  - `parse_repo_project_page()` — Parses a project page and returns a *ProjectPage* instance
  - `parse_repo_project_response()` — Parses a `requests.Response` object containing a project page and returns a *ProjectPage* instance
  - *parse_links_stream()* — Parses an HTML page as stream of `bytes` or `str` and returns a generator of *Link* objects
  - *parse_links_stream_response()* — Parses a streaming `requests.Response` object containing an HTML page and returns a generator of *Link* objects
  - `parse_repo_index_page()` — Parses a simple repository index/root page and returns an *IndexPage* instance
  - `parse_repo_index_response()` — Parses a `requests.Response` object containing an index page and returns an *IndexPage* instance
- The following functions & methods are now deprecated and will be removed in a future version:
  - `PyPISimple.get_projects()`
  - `PyPISimple.get_project_files()`
  - `parse_simple_index()`

  - `parse_project_page()`
  - `parse_links()`

- Support Warehouse's *`X-PyPI-Last-Serial`* header by attaching the value to the objects returned by *`get_index_page()`* and *`get_project_page()`*
- Support **PEP 629** by attaching the repository version to the objects returned by *`get_index_page()`* and *`get_project_page()`* and by raising an *`UnsupportedRepoVersionError`* when a repository with an unsupported version is encountered

3.13 v0.6.0 (2020-03-01)

- Support Python 3.8
- *`DistributionPackage.sig_url`* is now always non-`None`, as Warehouse does not report proper values for *`has_sig`*

3.14 v0.5.0 (2019-05-12)

- The *`PyPISimple`* constructor now takes an optional `session` argument which can be used to specify a `requests.Session` object with more complicated configuration than just authentication
- Support for PEP 592; *`DistributionPackage`* now has a `yanked` attribute

3.15 v0.4.0 (2018-09-06)

- Publicly (i.e., in the README) document the utility functions
- Gave *`PyPISimple`* an `auth` parameter for specifying login/authentication details

3.16 v0.3.0 (2018-09-03)

- When fetching the list of files for a project, the project name is now used to resolve ambiguous filenames.
- The filename parser now requires all filenames to be all-ASCII (except for wheels).

3.17 v0.2.0 (2018-09-01)

- The filename parser now rejects invalid project names, blatantly invalid versions, and non-ASCII digits.
- RPM packages are now recognized.

3.18 v0.1.0 (2018-08-31)

Initial release

`pypi-simple` is a client library for the Python Simple Repository API as specified in **PEP 503** and updated by **PEP 592**, **PEP 629**, **PEP 658**, **PEP 691**, **PEP 700**, **PEP 708**, and **PEP 714**. With it, you can query the Python Package Index (PyPI) and other pip-compatible repositories for a list of their available projects and lists of each project's available package files. The library also allows you to download package files and query them for their project version, package type, file digests, `requires_python` string, PGP signature URL, and metadata URL.

CHAPTER

FOUR

INSTALLATION

`pypi-simple` requires Python 3.8 or higher. Just use pip for Python 3 (You have pip, right?) to install it:

```
python3 -m pip install pypi-simple
```

`pypi-simple` can optionally make use of tqdm. To install it alongside `pypi-simple`, specify the `tqdm` extra:

```
python3 -m pip install "pypi-simple[tqdm]"
```

CHAPTER

FIVE

EXAMPLE

Get information about a package:

```
>>> from pypi_simple import PyPISimple
>>> with PyPISimple() as client:
...     requests_page = client.get_project_page('requests')
>>> pkg = requests_page.packages[0]
>>> pkg.filename
'requests-0.2.0.tar.gz'
>>> pkg.url
'https://files.pythonhosted.org/packages/ba/bb/
↪dfa0141a32d773c47e4dede1a617c59a23b74dd302e449cf85413fc96bc4/requests-0.2.0.tar.gz'
>>> pkg.project
'requests'
>>> pkg.version
'0.2.0'
>>> pkg.package_type
'sdist'
>>> pkg.digests
{'sha256': '813202ace4d9301a3c00740c700e012fb9f3f8c73ddcfe02ab558a8df6f175fd'}
```

Download a package with a tqdm progress bar:

```
from pypi_simple import PyPISimple, tqdm_progress_factory

with PyPISimple() as client:
    page = client.get_project_page("pypi-simple")
    pkg = page.packages[-1]
    client.download_package(
        pkg, path=pkg.filename, progress=tqdm_progress_factory(),
    )
```

CHAPTER

SIX

INDICES AND TABLES

- genindex
- search

PYTHON MODULE INDEX

p

INDEX

Symbols

A

C

D

E

F

G

H

I

L