HTTPStream Documentation

Release 1.1.0

Nigel Small

November 18, 2013

Contents

HTTPStream is a simple and pragmatic HTTP client library for Python that provides support for incremental JSON document retrieval and RFC 6570 URI Templates.

CHAPTER 1

Resources

A resource is an entity that exists on a distributed system, such as the World Wide Web, and is identified by a URI. Commonly associated with the REST architectural style, web resources are objects upon which HTTP methods like GET and POST can be actioned.

HTTPStream is built around a core `Resource` class that embodies the concept of the web resource and instances can be constructed by simply using the URI by which they are uniquely identified:

```
>>> from httpstream import Resource
>>> resource = Resource("http://example.com/foo/bar")
```

Although the `Resource` class can be used directly, applications may alternatively either inherit from or wrap this class to provide more meaningful naming:

```
from httpstream import Resource

class InheritedMailbox(Resource):

    def __init__(self, uri):
        Resource.__init__(self, uri)

    def deliver(self, message):
        self.post(message)

class WrappedMailbox(object):

    def __init__(self, uri):
        self._resource = Resource(uri)

    def deliver(self, message):
        self._resource.post(message)
```

For simple HTTP access, resources can of course be created and used in an immediate inline context:

```
>>> from httpstream import Resource
>>> results = Resource("https://api.duckduckgo.com/?q=neo4j&format=json").get().content
```

Methods such as `get` return a file-like `Response` object. The response content can be either iterated through or retrieved at once:

```
resource = Resource("http://example.com/")

# print each line of the response in turn
with resource.get() as response:
    for line in response:
        print line

# print the entire response content at once
with resource.get() as response:
    print response.content
```

class `httpstream.`**`Resource`**(*uri*)

A web resource identified by a URI.

`get`(*headers=None*, *redirect_limit=5*, ***kwargs*)

Issue a `GET` request to this resource.

Parameters

- **headers** (*dict*) – headers to be included in the request (optional)
- **redirect_limit** – maximum number of redirects to be handled automatically (optional, default=5)
- **product** – name or (name, version) tuple for the client product to be listed in the `User-Agent` header (optional)
- **chunk_size** – number of bytes to retrieve per chunk (optional, default=4096)

Returns file-like `Response` object from which content can be read

`put`(*body=None*, *headers=None*, ***kwargs*)

Issue a `PUT` request to this resource.

`post`(*body=None*, *headers=None*, ***kwargs*)

Issue a `POST` request to this resource.

`delete`(*headers=None*, ***kwargs*)

Issue a `DELETE` request to this resource.

`head`(*headers=None*, *redirect_limit=5*, ***kwargs*)

Issue a `HEAD` request to this resource.

`resolve`(*reference*, *strict=True*)

Resolve a URI reference against the URI for this resource, returning a new resource represented by the new target URI.

CHAPTER 2

Implicit Resources

A shorthand is also available for implicit resource creation:

```
>>> from httpstream import get
>>> results = get("https://api.duckduckgo.com/?q=neo4j&format=json").content
```

httpstream.**get**(*uri*, *headers=None*, *redirect_limit=5*, ***kwargs*)

httpstream.**put**(*uri*, *body=None*, *headers=None*, ***kwargs*)

httpstream.**post**(*uri*, *body=None*, *headers=None*, ***kwargs*)

httpstream.**delete**(*uri*, *headers=None*, ***kwargs*)

httpstream.**head**(*uri*, *headers=None*, *redirect_limit=5*, ***kwargs*)

CHAPTER 3

Resource Templates

```
>>> from httpstream import ResourceTemplate
>>> searcher = ResourceTemplate("https://api.duckduckgo.com/?q={query}&format=json")
>>> results = searcher.expand(query="neo4j").get().content
```

class `httpstream.`**`ResourceTemplate`**(*uri_template*)

`expand`(***values*)
Expand this template into a full URI using the values provided.

`uri_template`
The URI template string of this resource template.

CHAPTER 4

Requests & Responses

HTTPStream defines four types of response objects. A standard `Response` is generated on receipt of a 2xx status code and a `ClientError` and `ServerError` may be raised on receipt of 4xx and 5xx statuses respectively. The fourth response type is `Redirection` which is generally consumed internally but may also be returned under certain circumstances.

Response objects are file-like and as such may be `read` or iterated through The `iter_lines` and `iter_json` methods may be used to step through known types of content. The response object itself may also be iterated directly and an appropriate type of iterator is selected depending on the type of content available. The example below shows how to print each line of textual content as it is received:

```
>>> for line in res.get():
...     print line
```

class `httpstream.`**`Request`**(*method*, *uri*, *body=None*, *headers=None*)

`body`
Content of the request.

`headers`
Dictionary of headers attached to the request.

`method` = **None**
HTTP method of this request

`submit`(*redirect_limit=0*, *product=None*, ***response_kwargs*)
Submit this request and return a `Response` object.

`uri`
URI of the request.

class `httpstream.`**`Response`**(*http*, *uri*, *request*, *response*, ***kwargs*)
File-like object allowing consumption of an HTTP response.

`chunk_size` = **None**
Default chunk size for this response

`close`()
Close the response, discarding all remaining content and releasing the underlying connection object.

`closed`
Indicates whether or not the response is closed.

content
Fetch all content, returning a value appropriate for the content type.

content_length
The length of content as provided by the *Content-Length* header field. If the content is chunked, this returns `None`.

content_type
The type of content as provided by the *Content-Type* header field.

encoding
The content character set encoding.

headers
The response headers.

is_chunked
Indicates whether or not the content is chunked.

is_json
Indicates whether or not the content is JSON.

is_text
Indicates whether or not the content is text.

is_tsj
Indicates whether or not the content is tab-separated JSON.

iter_chunks(*chunk_size=None*)
Iterate through the content as chunks of text. Chunk sizes may vary slightly from that specified due to multi-byte characters. If no chunk size is specified, a default of 4096 is used.

iter_json()
Iterate through the content as individual JSON values.

iter_lines(*keep_ends=False*)
Iterate through the content as lines of text.

iter_tsj()
Iterate through the content as lines of tab-separated JSON.

json
Fetch all content, decoding from JSON and returning the decoded value.

read(*size=None*)
Fetch some or all of the response content, returning as a bytearray.

reason
The reason phrase attached to this response.

request
The `Request` object which preceded this response.

status_code
The status code of the response

text
Fetches all content as a string.

tsj
Fetches all content, decoding from tab-separated JSON and returning the decoded values.

uri
The URI from which the response came.

class `httpstream.`**`Redirection`**(*http*, *uri*, *request*, *response*, ***kwargs*)
Bases: `httpstream.http.Response`

class `httpstream.`**`ClientError`**(*http*, *uri*, *request*, *response*, ***kwargs*)
Bases: `exceptions.Exception`, `httpstream.http.Response`

class `httpstream.`**`ServerError`**(*http*, *uri*, *request*, *response*, ***kwargs*)
Bases: `exceptions.Exception`, `httpstream.http.Response`

CHAPTER 5

Incremental JSON Parsing

class httpstream.**JSONStream**(*source*)

Streaming JSON decoder. This class both expects Unicode input and will produce Unicode output.

httpstream.**assembled**(*iterable*)

Returns a JSON-derived value from a set of key-value pairs as produced by the JSONStream process. This operates in a similar way to the built-in *dict* function. Internally, this uses the *merged* function on each pair to build the return value.

```
>>> data = [
...     (("drink",), "lemonade"),
...     (("cutlery", 0), "knife"),
...     (("cutlery", 1), "fork"),
...     (("cutlery", 2), "spoon"),
... ]
>>> assembled(data)
{'cutlery': ['knife', 'fork', 'spoon'], 'drink': 'lemonade'}
```

Parameters **iterable** – key-value pairs to be merged into assembled value

httpstream.**grouped**(*iterable*, *level=1*)

CHAPTER 6

URIs

class `httpstream.`**`URI`**(*value*)

Uniform Resource Identifier.

See Also:

RFC 3986

`absolute_path_reference`

The path, query and fragment parts of this URI or `None` if undefined.

Component Definition:

```
https://bob@example.com:8080/data/report.html?date=2000-12-25#summary
                           \_____________________________________________/
                                                 |
                                      absolute_path_reference
```

Returns combined string values of path, query and fragment parts or `None`

Return type percent-encoded string or `None`

`authority`

The authority part of this URI or `None` if undefined.

Component Definition:

```
https://bob@example.com:8080/data/report.html?date=2000-12-25#summary
        \_____________________/
                   |
              authority
```

Return type `Authority` instance or `None`

`fragment`

The *fragment* part of this URI or `None` if undefined.

Component Definition:

```
https://bob@example.com:8080/data/report.html?date=2000-12-25#summary
                                                              \_____/
```

```
                                                                  |
                                                              fragment
```

Returns

Return type unencoded string or `None`

hierarchical_part

The authority and path parts of this URI or `None` if undefined.

Component Definition:

```
https://bob@example.com:8080/data/report.html?date=2000-12-25#summary
        \_____________________________________/
                          |
                  hierarchical_part
```

Returns combined string values of authority and path parts or `None`

Return type percent-encoded string or `None`

host

The *host* part of this URI or `None` if undefined.

```
>>> URI(None).host
None
>>> URI("").host
None
>>> URI("http://example.com").host
'example.com'
>>> URI("http://example.com:8080/data").host
'example.com'
```

Component Definition:

```
https://bob@example.com:8080/data/report.html?date=2000-12-25#summary
            \_________/
                 |
               host
```

Returns

Return type unencoded string or `None`

host_port

The *host* and *port* parts of this URI separated by a colon or `None` if both are undefined.

```
>>> URI(None).host_port
None
>>> URI("").host_port
None
>>> URI("http://example.com").host_port
'example.com'
>>> URI("http://example.com:8080/data").host_port
'example.com:8080'
>>> URI("http://bob@example.com:8080/data").host_port
'example.com:8080'
```

Component Definition:

```
https://bob@example.com:8080/data/report.html?date=2000-12-25#summary
            \_______________/
                    |
               host_port
```

Returns

Return type percent-encoded string or `None`

path

The *path* part of this URI or `None` if undefined.

Component Definition:

```
https://bob@example.com:8080/data/report.html?date=2000-12-25#summary
                              \_________________/
                                       |
                                     path
```

Returns

Return type `Path` instance or `None`

port

The *port* part of this URI or `None` if undefined.

```
>>> URI(None).port
None
>>> URI("").port
None
>>> URI("http://example.com").port
None
>>> URI("http://example.com:8080/data").port
8080
```

Component Definition:

```
https://bob@example.com:8080/data/report.html?date=2000-12-25#summary
                         \__/
                          |
                        port
```

Returns

Return type integer or `None`

query

The *query* part of this URI or `None` if undefined.

Component Definition:

```
https://bob@example.com:8080/data/report.html?date=2000-12-25#summary
                                               \_____________/
                                                      |
                                                    query
```

Return type `Query` instance or `None`

resolve(*reference*, *strict=True*)
Transform a reference relative to this URI to produce a full target URI.

See Also:

RFC 3986 § 5.2.2

scheme
The scheme part of this URI or `None` if undefined.

Component Definition:

```
https://bob@example.com:8080/data/report.html?date=2000-12-25#summary
\___/
  |
scheme
```

Return type unencoded string or `None`

string
The full percent-encoded string value of this URI or `None` if undefined.

```
>>> URI(None).string
None
>>> URI("").string
''
>>> URI("http://example.com").string
'example.com'
>>> URI("foo/bar").string
'foo/bar'
>>> URI("http://bob@example.com:8080/data/report.html?date=2000-12-25#summary").string
'http://bob@example.com:8080/data/report.html?date=2000-12-25#summary'
```

Component Definition:

```
https://bob@example.com:8080/data/report.html?date=2000-12-25#summary
\_____________________________________________________________________/
                                  |
                               string
```

Return type percent-encoded string or `None`

Note: Unlike `string`, the `__str__` method will always return a string, even when the URI is undefined; in this case, an empty string is returned instead of `None`.

---

user_info
The user information part of this URI or `None` if undefined.

Component Definition:

```
https://bob@example.com:8080/data/report.html?date=2000-12-25#summary
        \_/
         |
     user_info
```

Returns string value of user information part or `None`

Return type unencoded string or `None`

class `httpstream.`**`Authority`**(*string*)

A host name plus optional port and user information detail.

Syntax `authority := [ user_info "@" ] host [ ":" port ]`

See Also:

RFC 3986 § 3.2

`host`

The host part of this authority component, an empty string if host is empty or `None` if undefined.

```
>>> Authority(None).host
None
>>> Authority("").host
''
>>> Authority("example.com").host
'example.com'
>>> Authority("example.com:8080").host
'example.com'
>>> Authority("bob@example.com").host
'example.com'
>>> Authority("bob@example.com:8080").host
'example.com'
```

Returns

`host_port`

The host and port parts of this authority component or `None` if undefined.

```
>>> Authority(None).host_port
None
>>> Authority("").host_port
''
>>> Authority("example.com").host_port
'example.com'
>>> Authority("example.com:8080").host_port
'example.com:8080'
>>> Authority("bob@example.com").host_port
'example.com'
>>> Authority("bob@example.com:8080").host_port
'example.com:8080'
```

Returns

`port`

The port part of this authority component or `None` if undefined.

```
>>> Authority(None).port
None
>>> Authority("").port
None
>>> Authority("example.com").port
None
>>> Authority("example.com:8080").port
8080
>>> Authority("bob@example.com").port
None
```

```
>>> Authority("bob@example.com:8080").port
8080
```

Returns

string
The full string value of this authority component or :*py:const:'None* if undefined.

```
>>> Authority(None).string
None
>>> Authority("").string
''
>>> Authority("example.com").string
'example.com'
>>> Authority("example.com:8080").string
'example.com:8080'
>>> Authority("bob@example.com").string
'bob@example.com'
>>> Authority("bob@example.com:8080").string
'bob@example.com:8080'
```

Returns

user_info
The user information part of this authority component or `None` if undefined.

```
>>> Authority(None).user_info
None
>>> Authority("").user_info
None
>>> Authority("example.com").user_info
None
>>> Authority("example.com:8080").user_info
None
>>> Authority("bob@example.com").user_info
'bob'
>>> Authority("bob@example.com:8080").user_info
'bob'
```

Returns

class `httpstream.`**Path**(*string*)

remove_dot_segments()
Implementation of RFC3986, section 5.2.4

segments

string

with_trailing_slash()

without_trailing_slash()

class `httpstream.`**Query**(*string*)

classmethod decode(*string*)

classmethod `encode` (*iterable*)

`string`

CHAPTER 7

URI Templates

class `httpstream.`**`URITemplate`**(*template*)

A URI Template is a compact sequence of characters for describing a range of Uniform Resource Identifiers through variable expansion.

This class exposes a full implementation of RFC6570.

`expand`(***values*)

Expand into a URI using the values supplied

`string`

See Also:

RFC 6570

CHAPTER 8

Percent Encoding

Percent encoding is used within URI components to allow inclusion of certain characters which are not within a permitted set.

`httpstream.`**`percent_encode`**(*data*, *safe=None*)
Percent encode a string of data, optionally keeping certain characters unencoded.

`httpstream.`**`percent_decode`**(*data*)
Percent decode a string of data.

See Also:

RFC 3986 § 2.1

CHAPTER 9

Errors

exception httpstream.**NetworkAddressError**(*message*, *host_port=None*)

host_port

exception httpstream.**SocketError**(*code*, *host_port=None*)

code

host_port

exception httpstream.**RedirectionError**(**args*, ***kwargs*)