
urlfetch Documentation

Release 1.1.2

Yue Du <ifduyue@gmail.com>

Oct 11, 2019

Contents

1	Getting Started	3
1.1	Install	3
1.2	Usage	3
2	User's Guide	5
2.1	Examples	5
2.2	Reference	10
2.3	Changelog	16
2.4	Contributors	21
3	License	23
	Python Module Index	25
	Index	27

urldata is a simple, lightweight and easy to use HTTP client for [Python](#). It is distributed as a single file module and has no dependencies other than the [Python Standard Library](#).

CHAPTER 1

Getting Started

1.1 Install

```
$ pip install urlfetch
```

OR grab the latest source from github [ifduyue/urlfetch](https://github.com/ifduyue/urlfetch):

```
$ git clone git://github.com/ifduyue/urlfetch.git
$ cd urlfetch
$ python setup.py install
```

1.2 Usage

```
>>> import urlfetch
>>> r = urlfetch.get("http://docs.python.org/")
>>> r.status, r.reason
(200, 'OK')
>>> r.getheader('content-type')
'text/html; charset=UTF-8'
>>> r.getheader('Content-Type')
'text/html; charset=UTF-8'
>>> r.content
...
```


2.1 Examples

2.1.1 urlfetch at a glance

```
>>> import urlfetch
>>> r = urlfetch.get('https://twitter.com/')
>>> r.status, r.reason
(200, 'OK')
>>> r.total_time
0.924283027648926
>>> r.reqheaders
{'Host': 'twitter.com', 'Accept-Encoding': 'gzip, deflate, compress, identity, *', 'Accept': '*/*', 'User-Agent': 'urlfetch/0.5.3'}
>>> len(r.content), type(r.content)
(72560, <type 'str'>)
>>> len(r.text), type(r.text)
(71770, <type 'unicode'>)
>>> r.headers
{'status': '200 OK', 'content-length': '15017', 'strict-transport-security': 'max-age=631138519', 'x-transaction': '4a281c79631ee04e', 'content-encoding': 'gzip', 'set-cookie': 'k=10.36.121.114.1359712350849032; path=/; expires=Fri, 08-Feb-13 09:52:30 GMT; domain=.twitter.com, guest_id=v1%3A135971235085257249; domain=.twitter.com; path=/; expires=Sun, 01-Feb-2015 21:52:30 GMT, _twitter_sess=BAh7CjoPY3JlYXRlZlZhdGwrCIXyK5U8AToMY3NyZl9pZCIlNGIwYjA2NWQ2%250AZGEOMGUzN2Y5Y2Y3NzViYTc5MjdkM2Q6FWluX251d19lc2VyX2Zsb3cwIgp%250AbGFzaElDOidBY3Rpb25Db250cm9sbGVyOjJpGjBGFzaDo6Rmxhc2hIYXNoewAG%250AOgpAdXNlZHSaOgdpZCIlM2Y4MDl1NjVlNzA2M2Q0YTI4NjVmY2UyMWYzZmRh%250AMWY%253D--2869053b52dc7269a8a09ee3608737e0291e4ec1; domain=.twitter.com; path=/; HttpOnly', 'expires': 'Tue, 31 Mar 1981 05:00:00 GMT', 'x-mid': 'eb2ca7a2ae1109f1b2aea10729cdcf1d4821af5', 'server': 'tfe', 'last-modified': 'Fri, 01 Feb 2013 09:52:30 GMT', 'x-runtime': '0.13026', 'etag': '"15f3eb25198930feb6817975576b651b"', 'pragma': 'no-cache', 'cache-control': 'no-cache, no-store, must-revalidate, pre-check=0, post-check=0', 'date': 'Fri, 01 Feb 2013 09:52:30
```

(continues on next page)

(continued from previous page)

```

GMT', 'x-frame-options': 'SAMEORIGIN', 'content-type': 'text/html; charset=utf-8',
'x-xss-protection': '1; mode=block', 'vary': 'Accept-Encoding'})
>>> r.getheaders()
[('status', '200 OK'), ('content-length', '15017'), ('expires', 'Tue, 31 Mar 198
1 05:00:00 GMT'), ('x-transaction', '4a281c79631ee04e'), ('content-encoding', 'g
zip'), ('set-cookie', 'k=10.36.121.114.1359712350849032; path=/; expires=Fri, 08
-Feb-13 09:52:30 GMT; domain=twitter.com, guest_id=v1%3A135971235085257249; dom
ain=twitter.com; path=/; expires=Sun, 01-Feb-2015 21:52:30 GMT, _twitter_sess=B
Ah7CjoPY3JlYXRlZF9hdGwrCIXyK5U8AToMY3NyZl9pZC1lNGIwYjA2NWQ2%250AZGE0MGUzN2Y5Y2Y3
NzViYTc5MjdkM2Q6FWluX25ld19lc2VyX2Zsb3cwIgp%250AbGFzaElDOidBY3Rpb25Db250cm9sbGV
yOjpGbGFzaDo6Rmxhc2hIYXNoewAG%250AOGpAdXNlZHSaOgdpZC1lM2Y4MD1lNjVlNzA2M2Q0YTI4Nj
VmY2UyMWYzZmRh%250AMWY%253D--2869053b52dc7269a8a09ee3608737e0291e4ec1; domain=.t
witter.com; path=/; HttpOnly'), ('strict-transport-security', 'max-age=631138519
'), ('x-mid', 'eb2ca7a2ae1109f1b2ae10729cdcfld4821af5'), ('server', 'tfe'), ('
last-modified', 'Fri, 01 Feb 2013 09:52:30 GMT'), ('x-runtime', '0.13026'), ('et
ag', '"15f3eb25198930feb6817975576b651b"'), ('pragma', 'no-cache'), ('cache-cont
rol', 'no-cache, no-store, must-revalidate, pre-check=0, post-check=0'), ('date'
, 'Fri, 01 Feb 2013 09:52:30 GMT'), ('x-frame-options', 'SAMEORIGIN'), ('content
-type', 'text/html; charset=utf-8'), ('x-xss-protection', '1; mode=block'), ('va
ry', 'Accept-Encoding')]
>>> # getheader doesn't care whether you write 'content-length' or 'Content-Leng
th'
>>> # It's case insensitive
>>> r.getheader('content-length')
'15017'
>>> r.getheader('Content-Length')
'15017'
>>> r.cookies
{'guest_id': 'v1%3A135971235085257249', '_twitter_sess': 'BAh7CjoPY3JlYXRlZF9hdG
wrCIXyK5U8AToMY3NyZl9pZC1lNGIwYjA2NWQ2%250AZGE0MGUzN2Y5Y2Y3NzViYTc5MjdkM2Q6FWluX
25ld19lc2VyX2Zsb3cwIgp%250AbGFzaElDOidBY3Rpb25Db250cm9sbGVyOjpGbGFzaDo6Rmxhc2hI
YXNoewAG%250AOGpAdXNlZHSaOgdpZC1lM2Y4MD1lNjVlNzA2M2Q0YTI4NjVmY2UyMWYzZmRh%250AMW
Y%253D--2869053b52dc7269a8a09ee3608737e0291e4ec1', 'k': '10.36.121.114.135971235
0849032'}
>>> r.cookiestring
'guest_id=v1%3A135971235085257249; _twitter_sess=BAh7CjoPY3JlYXRlZF9hdGwrCIXyK5U
8AToMY3NyZl9pZC1lNGIwYjA2NWQ2%250AZGE0MGUzN2Y5Y2Y3NzViYTc5MjdkM2Q6FWluX25ld19lc2
VyX2Zsb3cwIgp%250AbGFzaElDOidBY3Rpb25Db250cm9sbGVyOjpGbGFzaDo6Rmxhc2hIYXNoewAG%
250AOGpAdXNlZHSaOgdpZC1lM2Y4MD1lNjVlNzA2M2Q0YTI4NjVmY2UyMWYzZmRh%250AMWY%253D--2
869053b52dc7269a8a09ee3608737e0291e4ec1; k=10.36.121.114.1359712350849032'

```

2.1.2 urllib3.fetch

`urllib3.fetch()` will determine the HTTP method (GET or POST) for you.

```

>>> import urllib3
>>> # It's HTTP GET
>>> r = urllib3.fetch("http://python.org/")
>>> r.status
200
>>> # Now it's HTTP POST
>>> r = urllib3.fetch("http://python.org/", data="foobar")
>>> r.status
200

```

2.1.3 Add HTTP headers

```
>>> from urllib3 import fetch
>>> r = fetch("http://python.org/", headers={"User-Agent": "urllib3"})
>>> r.status
200
>>> r.reqheaders
{'Host': u'python.org', 'Accept': '*/*', 'User-Agent': 'urllib3'}
>>> # alternatively, you can turn randua on
>>> # randua means generate a random user-agent
>>> r = fetch("http://python.org/", randua=True)
>>> r.status
200
>>> r.reqheaders
{'Host': u'python.org', 'Accept': '*/*', 'User-Agent': 'Mozilla/5.0 (Windows NT
6.1; WOW64) AppleWebKit/535.1 (KHTML, like Gecko) Chrome/14.0.835.8 Safari/535.1
'}
>>> r = fetch("http://python.org/", randua=True)
>>> r.status
200
>>> r.reqheaders
{'Host': u'python.org', 'Accept': '*/*', 'User-Agent': 'Mozilla/5.0 (Windows; U;
Windows NT 6.0; en-US; rv:1.9.2) Gecko/20100115 Firefox/3.6 (.NET CLR 3.5.30729
)'}
>>>
```

2.1.4 POST data

```
>>> from urllib3 import post
>>> r = post("http://python.org", data={'foo': 'bar'})
>>> r.status
200
>>> # data can be bytes
>>> r = post("http://python.org", data="foo=bar")
>>> r.status
200
>>>
```

2.1.5 Upload files

```
>>> from urllib3 import post
>>> r = post(
...     'http://127.0.0.1:8888/',
...     headers = {'Referer': 'http://127.0.0.1:8888/'},
...     data = {'foo': 'bar'},
...     files = {
...         'formname1': open('/tmp/path/to/file1', 'rb'),
...         'formname2': ('filename2', open('/tmp/path/to/file2', 'rb')),
...         'formname3': ('filename3', 'binary data of /tmp/path/to/file3'),
...     },
... )
>>> r.status
200
>>>
```

2.1.6 Basic auth and call github API

```
>>> from urlfetch import get
>>> import pprint
>>> r = get('https://api.github.com/gists', auth=('username', 'password'))
>>> pprint.pprint(r.json)
[{'comments': 0,
  'created_at': u'2012-03-21T15:22:13Z',
  'description': u'2_urlfetch.py',
  'files': {'2_urlfetch.py': {'filename': u'2_urlfetch.py',
                              'language': u'Python',
                              'raw_url': u'https://gist.github.com/raw/2148359/58c9062e0fc7bf6b9c43d2cf345ec4e6df2fef3e/2_urlfetch.py',
                              'size': 218,
                              'type': u'application/python'}}},
  'git_pull_url': u'git://gist.github.com/2148359.git',
  'git_push_url': u'git@gist.github.com:2148359.git',
  'html_url': u'https://gist.github.com/2148359',
  'id': u'2148359',
  'public': True,
  'updated_at': u'2012-03-21T15:22:13Z',
  'url': u'https://api.github.com/gists/2148359',
  'user': {'avatar_url': u'https://secure.gravatar.com/avatar/68b703a082b87cce010blaf5836711b3?d=https://a248.e.akamai.net/assets.github.com%2Fimages%2Fgravatar-140.png',
           'gravatar_id': u'68b703a082b87cce010blaf5836711b3',
           'id': 568900,
           'login': u'ifduyue',
           'url': u'https://api.github.com/users/ifduyue'}},
  ...]
```

2.1.7 urlfetch.Session

`urlfetch.Session` can hold common headers and cookies. Every request issued by a `urlfetch.Session` object will bring up these headers and cookies. `urlfetch.Session` plays a role in handling cookies, just like a `cookiejar`.

```
>>> from urlfetch import Session
>>> s = Session(headers={"User-Agent": "urlfetch session"}, cookies={"foo": "bar"})
>>> r = s.get("https://twitter.com/")
>>> r.status
200
>>> r.reqheaders
{'Host': u'twitter.com', 'Cookie': 'foo=bar', 'Accept': '*/*', 'User-Agent': 'urlfetch session'}
>>> r.cookies
{'guest_id': 'v1%3A134136902538582791', '_twitter_sess': 'BAh7CDoPY3JlYXRlZF9hdGwrCGoD0084ASIKZmxhc2hJQzonQWN0aW9uQ29u%250AdHJvbGxlcjo6Rmxhc2g6OkZsYXNoSGFzaHsABjoKQHVzZWRR7ADoHaWQiJWM2%250AMDAyMTY2YjFhY2YzNjk3NzU3ZmEwYTZjMTc2ZWl0--81b8c092d264beladb8b52eef177ab4466520f65', 'k': '10.35.53.118.1341369025382790'}
>>> r.cookiesstring
'guest_id=v1%3A134136902538582791; _twitter_sess=BAh7CDoPY3JlYXRlZF9hdGwrCGoD0084ASIKZmxhc2hJQzonQWN0aW9uQ29u%250AdHJvbGxlcjo6Rmxhc2g6OkZsYXNoSGFzaHsABjoKQHVzZWRR7ADoHaWQiJWM2%250AMDAyMTY2YjFhY2YzNjk3NzU3ZmEwYTZjMTc2ZWl0--81b8c092d264beladb8b52eef177ab4466520f65; k=10.35.53.118.1341369025382790'
```

(continues on next page)

(continued from previous page)

```

>>> s.putheader("what", "a nice day")
>>> s.putcookie("yah", "let's dance")
>>> r = s.get("https://twitter.com/")
>>> r.status
200
>>> r.reqheaders
{'Host': u'twitter.com', 'Cookie': "guest_id=v1%3A134136902538582791; _twitter_s
ess=BAh7CD0PY3JlYXRlZF9hdGwrCGoD0084ASIKZmxhc2hJQzonQWN0aW9uQ29u%250AdHJvbGxlcjo
6Rmxhc2g6OkZsYXNoSGFzaHsABjoKQHVzZWR7ADoHaWQiJWM2%250AMDYMTY2YjFhY2YzNjk3NzU3Zm
EwYTZjMTc2ZWl0--81b8c092d264beladb8b52eef177ab4466520f65; k=10.35.53.118.1341369
025382790; foo=bar; yah=let's dance", 'What': 'a nice day', 'Accept': '*/*', 'Us
er-Agent': 'urllib3 session'}
>>> # session cookiestring is also assignable
>>> s.cookiestring = 'foo=bar; l=2'
>>> s.cookies
{'l': '2', 'foo': 'bar'}

```

2.1.8 Streaming

```

>>> import urllib3
>>> with urllib3.get('http://some.very.large/file') as r:
>>>     with open('some.very.large.file', 'wb') as f:
>>>         for chunk in r:
>>>             f.write(chunk)

```

2.1.9 Proxies

```

>>> from urllib3 import get
>>> r = get('http://docs.python.org/', proxies={'http': '127.0.0.1:8888'})
>>> r.status, r.reason
(200, 'OK')
>>> r.headers
{'content-length': '8719', 'via': '1.1 tinyproxy (tinyproxy/1.8.2)', 'accept-ran
ges': 'bytes', 'vary': 'Accept-Encoding', 'server': 'Apache/2.2.16 (Debian)', 'l
ast-modified': 'Mon, 30 Jul 2012 19:22:48 GMT', 'etag': '"13cc5e4-220f-4c610fcaf
d200"', 'date': 'Tue, 31 Jul 2012 04:18:26 GMT', 'content-type': 'text/html'}

```

2.1.10 Redirects

```

>>> from urllib3 import get
>>> r = get('http://tinyurl.com/urllib3', max_redirects=10)
>>> r.history
[<urllib3.Response object at 0x274b8d0>]
>>> r.history[-1].headers
{'content-length': '0', 'set-cookie': 'tinyUUID=036051f7dc296a033f0608cf; expire
s=Fri, 23-Aug-2013 10:25:30 GMT; path=/; domain=.tinyurl.com', 'x-tiny': 'cache
0.0016100406646729', 'server': 'TinyURL/1.6', 'connection': 'close', 'location':
'https://github.com/iftuyue/urllib3', 'date': 'Thu, 23 Aug 2012 10:25:30 GMT',
'content-type': 'text/html'}
>>> r.headers

```

(continues on next page)

(continued from previous page)

```
{'status': '200 OK', 'content-encoding': 'gzip', 'transfer-encoding': 'chunked',
 'set-cookie': '_gh_sess=BAh7BzoPc2Vzc2lvbl9pZCI1N2VjNWm3NjMzOTJhY2YyMGYyNTJlYzU4NmZjMmRlY2U6EF9jc3JmX3Rva2VuIjFlc1VzYnpYlU1NLV0ZqeXg4S1NRQUx3VllmM3VEa2ZaZml
 iRHBzSGRzPQ%3D%3D--cbe63e27e8e6bf07edf0447772cf512d2fbdf2e2; path=/; expires=Sat
 , 01-Jan-2022 00:00:00 GMT; secure; HttpOnly', 'strict-transport-security': 'max
 -age=2592000', 'connection': 'keep-alive', 'server': 'nginx/1.0.13', 'x-runtime'
 : '104', 'etag': '"4137339e0195583b4f034c33202df9e8"', 'cache-control': 'private
 , max-age=0, must-revalidate', 'date': 'Thu, 23 Aug 2012 10:25:31 GMT', 'x-frame
 -options': 'deny', 'content-type': 'text/html; charset=utf-8'}
>>>
>>> # If max_redirects exceeded, an exception will be raised
>>> r = get('http://google.com/', max_redirects=1)
Traceback (most recent call last):
  File "<input>", line 1, in <module>
  File "urlfetch.py", line 627, in request
    raise UrlfetchException('max_redirects exceeded')
UrlfetchException: max_redirects exceeded
```

2.2 Reference

class urlfetch.**Response** (*r*, ****kwargs**)

A Response object.

```
>>> import urlfetch
>>> response = urlfetch.get("http://docs.python.org/")
>>> response.total_time
0.033042049407959
>>> response.status, response.reason, response.version
(200, 'OK', 10)
>>> type(response.body), len(response.body)
(<type 'str'>, 8719)
>>> type(response.text), len(response.text)
(<type 'unicode'>, 8719)
>>> response.getheader('server')
'Apache/2.2.16 (Debian)'
>>> response.getheaders()
[
  ('content-length', '8719'),
  ('x-cache', 'MISS from localhost'),
  ('accept-ranges', 'bytes'),
  ('vary', 'Accept-Encoding'),
  ('server', 'Apache/2.2.16 (Debian)'),
  ('last-modified', 'Tue, 26 Jun 2012 19:23:18 GMT'),
  ('connection', 'close'),
  ('etag', '"13cc5e4-220f-4c36507ded580"'),
  ('date', 'Wed, 27 Jun 2012 06:50:30 GMT'),
  ('content-type', 'text/html'),
  ('x-cache-lookup', 'MISS from localhost:8080')
]
>>> response.headers
{
  'content-length': '8719',
  'x-cache': 'MISS from localhost',
  'accept-ranges': 'bytes',
```

(continues on next page)

(continued from previous page)

```

'vary': 'Accept-Encoding',
'server': 'Apache/2.2.16 (Debian)',
'last-modified': 'Tue, 26 Jun 2012 19:23:18 GMT',
'connection': 'close',
'etag': '"13cc5e4-220f-4c36507ded580"',
'date': 'Wed, 27 Jun 2012 06:50:30 GMT',
'content-type': 'text/html',
'x-cache-lookup': 'MISS from localhost:8080'
}

```

Raises *ContentLimitExceeded*

body

Response body.

Raises *ContentLimitExceeded*, *ContentDecodingError*

close()

Close the connection.

content

cookies

Cookies in dict

cookiestring

Cookie string

classmethod from_httplib(connection, **kwargs)

Make an *Response* object from a httplib response object.

headers

Response headers.

Response headers is a dict with all keys in lower case.

```

>>> import urlfetch
>>> response = urlfetch.get("http://docs.python.org/")
>>> response.headers
{
  'content-length': '8719',
  'x-cache': 'MISS from localhost',
  'accept-ranges': 'bytes',
  'vary': 'Accept-Encoding',
  'server': 'Apache/2.2.16 (Debian)',
  'last-modified': 'Tue, 26 Jun 2012 19:23:18 GMT',
  'connection': 'close',
  'etag': '"13cc5e4-220f-4c36507ded580"',
  'date': 'Wed, 27 Jun 2012 06:50:30 GMT',
  'content-type': 'text/html',
  'x-cache-lookup': 'MISS from localhost:8080'
}

```

json

Load response body as json.

Raises *ContentDecodingError*

links

Links parsed from HTTP Link header

next ()
read (chunk_size=65536)

Read content (for streaming and large files)

Parameters **chunk_size** (*int*) – size of chunk, default is 8192.

reason = None

Reason phrase returned by server.

status = None

Status code returned by server.

status_code = None

An alias of *status*.

text

Response body in unicode.

total_time = None

total time

version = None

HTTP protocol version used by server. 10 for HTTP/1.0, 11 for HTTP/1.1.

class urlfetch.Session (headers={}, cookies={}, auth=None)

A session object.

urlfetch.Session can hold common headers and cookies. Every request issued by a *urlfetch.Session* object will bring u these headers and cookies.

urlfetch.Session plays a role in handling cookies, just like a cookiejar.

Parameters

- **headers** (*dict*) – Init headers.
- **cookies** (*dict*) – Init cookies.
- **auth** (*tuple*) – (username, password) for basic authentication.

cookies = None

cookies

cookiestring

Cookie string.

It's assignalbe, and will change *cookies* correspondingly.

```
>>> s = Session()
>>> s.cookiestring = 'foo=bar; l=2'
>>> s.cookies
{'l': '2', 'foo': 'bar'}
```

delete (*args, **kwargs)

Issue a delete request.

fetch (*args, **kwargs)

Fetch an URL

get (*args, **kwargs)

Issue a get request.

head (**args, **kwargs*)
Issue a head request.

headers = **None**
headers

options (**args, **kwargs*)
Issue a options request.

patch (**args, **kwargs*)
Issue a patch request.

popcookie (*key*)
Remove an cookie from default cookies.

popheader (*header*)
Remove an header from default headers.

post (**args, **kwargs*)
Issue a post request.

put (**args, **kwargs*)
Issue a put request.

putcookie (*key, value=""*)
Add an cookie to default cookies.

putheader (*header, value*)
Add an header to default headers.

request (**args, **kwargs*)
Issue a request.

snapshot ()

trace (**args, **kwargs*)
Issue a trace request.

`urlfetch.request` (*url, method='GET', params=None, data=None, headers={}, timeout=None, files={}, randua=False, auth=None, length_limit=None, proxies=None, trust_env=True, max_redirects=0, source_address=None, **kwargs*)
request an URL

Parameters

- **url** (*string*) – URL to be fetched.
- **method** (*string*) – (optional) HTTP method, one of GET, DELETE, HEAD, OPTIONS, PUT, POST, TRACE, PATCH. GET is the default.
- **params** (*dict/string*) – (optional) Dict or string to attach to url as querystring.
- **headers** (*dict*) – (optional) HTTP request headers.
- **timeout** (*float*) – (optional) Timeout in seconds
- **files** – (optional) Files to be sended
- **randua** – (optional) If True or path string, use a random user-agent in headers, instead of 'urlfetch/' + `__version__`
- **auth** (*tuple*) – (optional) (username, password) for basic authentication
- **length_limit** (*int*) – (optional) If None, no limits on content length, if the limit reached raised exception 'Content length is more than ...'

- **proxies** (*dict*) – (optional) HTTP proxy, like {'http': '127.0.0.1:8888', 'https': '127.0.0.1:563'}
- **trust_env** (*bool*) – (optional) If True, urllib3 will get informations from env, such as HTTP_PROXY, HTTPS_PROXY
- **max_redirects** (*int*) – (integer, optional) Max redirects allowed within a request. Default is 0, which means redirects are not allowed.
- **source_address** (*tuple*) – (optional) A tuple of (host, port) to specify the source_address to bind to. This argument is ignored if you're using Python prior to 2.7/3.2.

Returns A *Response* object

Raises *URLError*, *Urllib3Exception*, *TooManyRedirects*,

`urllib3.fetch(*args, **kwargs)`
fetch an URL.

fetch() is a wrapper of *request()*. It calls *get()* by default. If one of parameter data or parameter files is supplied, *post()* is called.

`urllib3.get(url, params=None, data=None, headers={}, timeout=None, files={}, randua=False, auth=None, length_limit=None, proxies=None, trust_env=True, max_redirects=0, source_address=None, **kwargs)`
Issue a get request

`urllib3.post(url, params=None, data=None, headers={}, timeout=None, files={}, randua=False, auth=None, length_limit=None, proxies=None, trust_env=True, max_redirects=0, source_address=None, **kwargs)`
Issue a post request

`urllib3.head(url, params=None, data=None, headers={}, timeout=None, files={}, randua=False, auth=None, length_limit=None, proxies=None, trust_env=True, max_redirects=0, source_address=None, **kwargs)`
Issue a head request

`urllib3.put(url, params=None, data=None, headers={}, timeout=None, files={}, randua=False, auth=None, length_limit=None, proxies=None, trust_env=True, max_redirects=0, source_address=None, **kwargs)`
Issue a put request

`urllib3.delete(url, params=None, data=None, headers={}, timeout=None, files={}, randua=False, auth=None, length_limit=None, proxies=None, trust_env=True, max_redirects=0, source_address=None, **kwargs)`
Issue a delete request

`urllib3.options(url, params=None, data=None, headers={}, timeout=None, files={}, randua=False, auth=None, length_limit=None, proxies=None, trust_env=True, max_redirects=0, source_address=None, **kwargs)`
Issue a options request

`urllib3.trace(url, params=None, data=None, headers={}, timeout=None, files={}, randua=False, auth=None, length_limit=None, proxies=None, trust_env=True, max_redirects=0, source_address=None, **kwargs)`
Issue a trace request

`urllib3.patch(url, params=None, data=None, headers={}, timeout=None, files={}, randua=False, auth=None, length_limit=None, proxies=None, trust_env=True, max_redirects=0, source_address=None, **kwargs)`
Issue a patch request

2.2.1 Exceptions

```
class urlfetch.UrlfetchException
    Base exception. All exceptions and errors will subclass from this.

class urlfetch.ContentLimitExceeded
    Content length is beyond the limit.

class urlfetch.URLError
    Error parsing or handling the URL.

class urlfetch.ContentDecodingError
    Failed to decode the content.

class urlfetch.TooManyRedirects
    Too many redirects.

class urlfetch.Timeout
    Request timed out.
```

2.2.2 helpers

```
urlfetch.parse_url(url)
    Return a dictionary of parsed url

    Including scheme, netloc, path, params, query, fragment, uri, username, password, host, port and http_host

urlfetch.get_proxies_from_environ()
    Get proxies from os.environ.

urlfetch.mb_code(s, coding=None, errors='replace')
    encoding/decoding helper.

urlfetch.random_useragent(filename=True)
    Returns a User-Agent string randomly from file.

    Parameters filename (string) – (Optional) Path to the file from which a random useragent is
    generated. By default it's True, a file shipped with this module will be used.

    Returns An user-agent string.

urlfetch.url_concat(url, args, keep_existing=True)
    Concatenate url and argument dictionary
```

```
>>> url_concat("http://example.com/foo?a=b", dict(c="d"))
'http://example.com/foo?a=b&c=d'
```

Parameters

- **url** (*string*) – URL being concat to.
- **args** (*dict*) – Args being concat.
- **keep_existing** (*bool*) – (Optional) Whether to keep the args which are already in url, default is True.

```
urlfetch.choose_boundary()
    Generate a multipart boundry.
```

Returns A boundary string

`urllib3.encode_multipart(data, files)`

Encode multipart.

Parameters

- **data** (*dict*) – Data to be encoded
- **files** (*dict*) – Files to be encoded

Returns Encoded binary string

Raises `Urllib3Exception`

2.3 Changelog

Time flies!!

2.3.1 1.1.2 (2019-03-27)

Small optimizations:

- Larger chunk when reading response
- Read chunks into list and then join them to bytes
- Close response when exception occurs

2.3.2 1.1.1 (2018-12-20)

- Updated user-agent list.

2.3.3 1.1.0 (2018-11-16)

New features:

- Support `source_address`
- Support `no_proxy` environment variable

2.3.4 1.0.3 (2018-01-03)

Improvements:

- Run tests against Python 3.5 3.6 3.7 and PyPy.
- Try to deal with `data_files` paths.
- Some minor changes regarding coding style.

2.3.5 1.0.2 (2015-04-29)

Fixes:

- `python setup.py test` causes `SandboxViolation`.

Improvements:

- `python setup.py test` handles dependencies automatically.
- `random_useragent()`: check if `urlfetch.useragents.list` exists at the import time.

2.3.6 1.0.1 (2015-01-31)

Fixes:

- `urlfetch.Response.history` of a redirected response and its precedent responses should be different.

Improvements:

- Simplified some code.
- Added some tests.

2.3.7 1.0 (2014-03-22)

New features:

- Support `idna`.
- Assignable `Session.cookiestring`.

Backwards-incompatible changes:

- Remove `raw_header` and `raw_response`.
- `random_useragent()` now takes a single filename as parameter. It used to be a list of filenames.
- No more `.title()` on request headers' keys.
- Exceptions are re-designed. `socket.timeout` now is `Timeout`, ..., see section *Exceptions* in *Reference* for more details.

Fixes:

- Parsing links: If `Link` header is empty, `[]` should be returned, not `[{'url': ''}]`.
- Http request's `Host` header should include the port. Using `netloc` as the http host header is wrong, it could include `user:pass`.
- Redirects: `Host` in `reqheaders` should be `host:port`.
- Streaming decompress not working.

2.3.8 0.6.2 (2014-03-22)

Fix:

- Http request's `host` header should include the port. Using `netloc` as the http host header is wrong, it could include `user:pass`.

2.3.9 0.6.1 (2014-03-15)

Fix:

- Parsing links: If `Link` header is empty, `[]` should be returned, not `[{'url': ''}]`.

2.3.10 0.6 (2013-08-26)

Change:

- Remove lazy response introduced in 0.5.6
- Remove the `dump`, `dumps`, `load` and `loads` methods of `urldata.Response`

2.3.11 0.5.7 (2013-07-08)

Fix:

- Host header field should include host and port

2.3.12 0.5.6 (2013-07-04)

Feature:

- Lay response. Read response when you need it.

2.3.13 0.5.5 (2013-06-07)

Fix:

- fix docstring.
- `parse_url` raise exception for `http://foo.com/`

2.3.14 0.5.4.2 (2013-03-31)

Feature:

- `urldata.Response.link`, links parsed from HTTP Link header.

Fix:

- Scheme doesn't correspond to the new location when following redirects.

2.3.15 0.5.4.1 (2013-03-05)

Fix:

- `urldata.random_useragent()` raises exception `[Errno 2] No such file or directory`.
- `urldata.encode_multipart()` doesn't use `isinstance: (object, class-or-type-or-tuple)` correctly.

2.3.16 0.5.4 (2013-02-28)

Feature:

- HTTP Proxy-Authorization.

Fix:

- Fix docstring typos.
- `urlfetch.encode_multipart()` should behave the same as `urllib.urlencode(query, doseq=1)`.
- `urlfetch.parse_url()` should parse urls like they are HTTP urls.

2.3.17 0.5.3.1 (2013-02-01)

Fix:

- `urlfetch.Response.content` becomes empty after the first access.

2.3.18 0.5.3 (2013-02-01)

Feature:

- NEW `urlfetch.Response.status_code`, alias of `urlfetch.Response.status`.
- NEW `urlfetch.Response.total_time`, `urlfetch.Response.raw_header` and `urlfetch.Response.raw_response`.
- Several properties of `urlfetch.Response` are cached to avoid unnecessary calls, including `urlfetch.Response.text`, `urlfetch.Response.json`, `urlfetch.Response.headers`, `urlfetch.Response.cookies`, `urlfetch.Response.cookiesstring`, `urlfetch.Response.raw_header` and `urlfetch.Response.raw_response`.

Fix:

- `urlfetch.mb_code()` may silently return incorrect result, since the encode errors are replaced, it should be decode properly and then encode without replace.

2.3.19 0.5.2 (2012-12-24)

Feature:

- `random_useragent()` can accept list/tuple/set params and can accept more than one params which specify the paths to check and read from. Below are some examples:

```
>>> ua = random_useragent('file1')
>>> ua = random_useragent('file1', 'file2')
>>> ua = random_useragent(['file1', 'file2'])
>>> ua = random_useragent(['file1', 'file2'], 'file3')
```

Fix:

- Possible infinite loop in `random_useragent()`.

2.3.20 0.5.1 (2012-12-05)

Fix:

- In some platforms `urlfetch.useragents.list` located in wrong place.
- `random_useragent()` will never return the first line.
- Typo in the description of `urlfetch.useragents.list` (the first line).

2.3.21 0.5.0 (2012-08-23)

- Redirects support. Parameter `max_redirects` specify the max redirects allowed within a request. Default is 0, which means redirects are not allowed.
- Code cleanups

2.3.22 0.4.3 (2012-08-17)

- Add `params` parameter, `params` is dict or string to attach to request url as querysting.
- Gzip and deflate support.

2.3.23 0.4.2 (2012-07-31)

- HTTP(S) proxies support.

2.3.24 0.4.1 (2012-07-04)

- Streaming support.

2.3.25 0.4.0 (2012-07-01)

- NEW `urlfetch.Session` to manipulate cookies automatically, share common request headers and cookies.
- NEW `urlfetch.Response.cookies` and `urlfetch.Response.cookiestring` to get response cookie dict and cookie string.

2.3.26 0.3.6 (2012-06-08)

- Simplify code
- Trace method without data and files, according to RFC2612
- `urlencode(data, 1)` so that `urlencode({'param': [1, 2, 3]})` => `'param=1¶m=2¶m=3'`

2.3.27 0.3.5 (2012-04-24)

- Support specifying an IP for the request host, useful for testing API.

2.3.28 0.3.0 (2012-02-28)

- Python 3 compatible

2.3.29 0.2.2 (2012-02-22)

- Fix bug: file upload: file should always have a filename

2.3.30 0.2.1 (2012-02-22)

- More flexible file upload
- Rename fetch2 to request
- Add auth parameter, instead of put basic authentication info in url

2.3.31 0.1.2 (2011-12-07)

- Support basic auth

2.3.32 0.1 (2011-12-02)

- First release

2.4 Contributors

- Andrey Usov <<https://github.com/ownport>>
- Liu Qishuai <<https://github.com/lqs>>
- wangking <<https://github.com/wangking>>

CHAPTER 3

License

Code and documentation are available according to the BSD 2-clause License:

```
Copyright (c) 2012-2019, Yue Du  
All rights reserved.
```

```
Redistribution and use in source and binary forms, with or without modification,  
are permitted provided that the following conditions are met:
```

- * Redistributions of source code must retain the above copyright notice,
this list of conditions and the following disclaimer.
- * Redistributions in binary form must reproduce the above copyright notice,
this list of conditions and the following disclaimer in the documentation
and/or other materials provided with the distribution.

```
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND  
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED  
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE  
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE  
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL  
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR  
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER  
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,  
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE  
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```


u

`urlfetch` (*Unix, Windows*), [10](#)

B

body (*urlfetch.Response* attribute), 11

C

choose_boundary() (*in module urlfetch*), 15
close() (*urlfetch.Response* method), 11
content (*urlfetch.Response* attribute), 11
ContentDecodingError (*class in urlfetch*), 15
ContentLimitExceeded (*class in urlfetch*), 15
cookies (*urlfetch.Response* attribute), 11
cookies (*urlfetch.Session* attribute), 12
cookiestring (*urlfetch.Response* attribute), 11
cookiestring (*urlfetch.Session* attribute), 12

D

delete() (*in module urlfetch*), 14
delete() (*urlfetch.Session* method), 12

E

encode_multipart() (*in module urlfetch*), 15

F

fetch() (*in module urlfetch*), 14
fetch() (*urlfetch.Session* method), 12
from_httplib() (*urlfetch.Response* class method), 11

G

get() (*in module urlfetch*), 14
get() (*urlfetch.Session* method), 12
get_proxies_from_env() (*in module urlfetch*), 15

H

head() (*in module urlfetch*), 14
head() (*urlfetch.Session* method), 12
headers (*urlfetch.Response* attribute), 11
headers (*urlfetch.Session* attribute), 13

J

json (*urlfetch.Response* attribute), 11

L

links (*urlfetch.Response* attribute), 11

M

mb_code() (*in module urlfetch*), 15

N

next() (*urlfetch.Response* method), 12

O

options() (*in module urlfetch*), 14
options() (*urlfetch.Session* method), 13

P

parse_url() (*in module urlfetch*), 15
patch() (*in module urlfetch*), 14
patch() (*urlfetch.Session* method), 13
popcookie() (*urlfetch.Session* method), 13
popheader() (*urlfetch.Session* method), 13
post() (*in module urlfetch*), 14
post() (*urlfetch.Session* method), 13
put() (*in module urlfetch*), 14
put() (*urlfetch.Session* method), 13
putcookie() (*urlfetch.Session* method), 13
putheader() (*urlfetch.Session* method), 13

R

random_useragent() (*in module urlfetch*), 15
read() (*urlfetch.Response* method), 12
reason (*urlfetch.Response* attribute), 12
request() (*in module urlfetch*), 13
request() (*urlfetch.Session* method), 13
Response (*class in urlfetch*), 10

S

Session (*class in urlfetch*), 12

`snapshot()` (*urllib3.Session method*), 13
`status` (*urllib3.Response attribute*), 12
`status_code` (*urllib3.Response attribute*), 12

T

`text` (*urllib3.Response attribute*), 12
`Timeout` (*class in urllib3*), 15
`TooManyRedirects` (*class in urllib3*), 15
`total_time` (*urllib3.Response attribute*), 12
`trace()` (*in module urllib3*), 14
`trace()` (*urllib3.Session method*), 13

U

`url_concat()` (*in module urllib3*), 15
`URLError` (*class in urllib3*), 15
`urllib3` (*module*), 10
`Urllib3Exception` (*class in urllib3*), 15

V

`version` (*urllib3.Response attribute*), 12