index
/usr/lib/python2.7/urllib.py
Module Docs

urllib (version 1.17)

Open an arbitrary URL.

See the following document for more info on URLs:
"Names and Addresses, URIs, URLs, URNs, URCs", at
http://www.w3.org/pub/WWW/Addressing/Overview.html

See also the HTTP spec (from which the error codes are derived):
"HTTP - Hypertext Transfer Protocol", at
http://www.w3.org/pub/WWW/Protocols/

Related standards and specs:
- RFC1808: the "relative URL" spec. (authoritative status)
- RFC1738 - the "URL standard". (authoritative status)
- RFC1630 - the "URI spec". (informational status)

The object returned by URLopener().open(file) will differ per
protocol. All you know is that is has methods read(), readline(),
readlines(), fileno(), close() and info(). The read*(), fileno()
and close() methods work like those of open files.
The info() method returns a mimetools.Message object which can be
used to query various info about the object, if available.
(mimetools.Message objects are queried with the getheader() method.)

Modules

base64	re	ssl	sys
os	socket	string	time

Classes

URLopener
FancyURLopener

class **FancyURLopener**(URLopener)
Derived class with handlers for errors we can handle (perhaps).

Methods defined here:

__init__(self, *args, **kwargs)

get_user_passwd(self, host, realm, clear_cache=0)

http_error_301(self, url, fp, errcode, errmsg, headers, data=None)
Error 301 -- also relocated (permanently).

http_error_302(self, url, fp, errcode, errmsg, headers, data=None)
Error 302 -- relocated (temporarily).

http_error_303(self, url, fp, errcode, errmsg, headers, data=None)
Error 303 -- also relocated (essentially identical to 302).

http_error_307(self, url, fp, errcode, errmsg, headers, data=None)
Error 307 -- relocated, but turn POST into error.

http_error_401(self, url, fp, errcode, errmsg, headers, data=None)
Error 401 -- authentication required.
This function supports Basic authentication only.

http_error_407(self, url, fp, errcode, errmsg, headers, data=None)
Error 407 -- proxy authentication required.
This function supports Basic authentication only.

http_error_default(self, url, fp, errcode, errmsg, headers)
Default error handling -- don't raise an exception.

prompt_user_passwd(self, host, realm)
Override this in a GUI environment!

redirect_internal(self, url, fp, errcode, errmsg, headers, data)

retry_http_basic_auth(self, url, realm, data=None)

retry_https_basic_auth(self, url, realm, data=None)

retry_proxy_http_basic_auth(self, url, realm, data=None)

retry_proxy_https_basic_auth(self, url, realm, data=None)

Methods inherited from URLopener:

__del__(self)

addheader(self, *args)
Add a header to be used by the HTTP interface only
e.g. u.addheader('Accept', 'sound/basic')

cleanup(self)

close(self)

http_error(self, url, fp, errcode, errmsg, headers, data=None)
Handle http errors.
Derived class can override this, or provide specific handlers
named http_error_DDD where DDD is the 3-digit error code.

open(self, fullurl, data=None)
Use URLopener().open(file) instead of open(file, 'r').

open_data(self, url, data=None)
Use "data" URL.

open_file(self, url)
Use local file or FTP depending on form of URL.

open_ftp(self, url)
Use FTP protocol.

open_http(self, url, data=None)
Use HTTP protocol.

open_https(self, url, data=None)
Use HTTPS protocol.

open_local_file(self, url)
Use local file.

open_unknown(self, fullurl, data=None)
Overridable interface to open unknown URL type.

open_unknown_proxy(self, proxy, fullurl, data=None)
Overridable interface to open unknown URL type.

retrieve(self, url, filename=None, reporthook=None, data=None)
retrieve(url) returns (filename, headers) for a local object
or (tempfilename, headers) for a remote object.

Data and other attributes inherited from URLopener:

version = 'Python-urllib/1.17'

class **URLopener**

Class to open URLs.
This is a class rather than just a subroutine because we may need
more than one set of global protocol-specific options.
Note -- this is a base class for those who don't want the
automatic handling of errors type 302 (relocated) and 401
(authorization needed).

Methods defined here:

__del__(self)

__init__(self, proxies=None, context=None, **x509)
```
# Constructor
```

addheader(self, *args)
```
Add a header to be used by the HTTP interface only
e.g. u.addheader('Accept', 'sound/basic')
```

cleanup(self)

close(self)

http_error(self, url, fp, errcode, errmsg, headers, data=None)
```
Handle http errors.
Derived class can override this, or provide specific handlers
named http_error_DDD where DDD is the 3-digit error code.
```

http_error_default(self, url, fp, errcode, errmsg, headers)
```
Default error handler: close the connection and raise IOError.
```

open(self, fullurl, data=None)
```
Use URLopener().open(file) instead of open(file, 'r').
```

open_data(self, url, data=None)
```
Use "data" URL.
```

open_file(self, url)
```
Use local file or FTP depending on form of URL.
```

open_ftp(self, url)
```
Use FTP protocol.
```

open_http(self, url, data=None)
```
Use HTTP protocol.
```

open_https(self, url, data=None)
```
Use HTTPS protocol.
```

open_local_file(self, url)
```
Use local file.
```

open_unknown(self, fullurl, data=None)
```
Overridable interface to open unknown URL type.
```

open_unknown_proxy(self, proxy, fullurl, data=None)
```
Overridable interface to open unknown URL type.
```

retrieve(self, url, filename=None, reporthook=None, data=None)
```
retrieve(url) returns (filename, headers) for a local object
or (tempfilename, headers) for a remote object.
```

Data and other attributes defined here:

version = 'Python-urllib/1.17'

Functions

basejoin = urljoin(base, url, allow_fragments=True)

Join a base URL and a possibly relative URL to form an absolute interpretation of the latter.

ftperrors()

Return the set of errors raised by the FTP class.

getproxies = getproxies_environment()

Return a dictionary of scheme -> proxy server URL mappings.

Scan the environment for variables named <scheme>_proxy;
this seems to be the standard convention. In order to prefer lowercase
variables, we process the environment in two passes, first matches any
and second matches only lower case proxies.

If you need a different way, you can pass a proxies dictionary to the
[Fancy]URLopener constructor.

localhost()

Return the IP address of the magic hostname 'localhost'.

pathname2url(pathname)

OS-specific conversion from a file system path to a relative URL
of the 'file' scheme; not recommended for general use.

quote(s, safe='/')

quote('abc def') -> 'abc%20def'

Each part of a URL, e.g. the path info, the query, etc., has a
different set of reserved characters that must be quoted.

RFC 2396 Uniform Resource Identifiers (URI): Generic Syntax lists
the following reserved characters.

```
reserved    = ";" | "/" | "?" | ":" | "@" | "&" | "=" | "+" |
              "$" | ","
```

Each of these characters is reserved in some component of a URL,
but not necessarily in all of them.

By default, the quote function is intended for quoting the path
section of a URL. Thus, it will not encode '/'. This character
is reserved, but in typical usage the quote function is being
called on a path where the existing slash characters are used as
reserved characters.

quote_plus(s, safe='')
Quote the query fragment of a URL; replacing ' ' with '+'

splitattr(url)
splitattr('/path;attr1=value1;attr2=value2;...') ->
'/path', ['attr1=value1', 'attr2=value2', ...].

splithost(url)
splithost('//host[:port]/path') --> 'host[:port]', '/path'.

splitnport(host, defport=-1)
Split host and port, returning numeric port.
Return given default port if no ':' found; defaults to -1.
Return numerical port if a valid number are found after ':'.
Return None if ':' but not a valid number.

splitpasswd(user)
splitpasswd('user:passwd') -> 'user', 'passwd'.

splitport(host)
splitport('host:port') --> 'host', 'port'.

splitquery(url)
splitquery('/path?query') --> '/path', 'query'.

splittag(url)
splittag('/path#tag') --> '/path', 'tag'.

splittype(url)
splittype('type:opaquestring') --> 'type', 'opaquestring'.

splituser(host)
splituser('user[:passwd]@host[:port]') --> 'user[:passwd]', 'host[:port]'.

splitvalue(attr)
splitvalue('attr=value') --> 'attr', 'value'.

thishost()
Return the IP address of the current host.

unquote(s)
unquote('abc%20def') -> 'abc def'.

unquote_plus(s)
unquote('%7e/abc+def') -> '~/abc def'

unwrap(url)
unwrap('<URL:type://host/path>') --> 'type://host/path'.

url2pathname(pathname)

```
OS-specific conversion from a relative URL of the 'file' scheme
to a file system path; not recommended for general use.
```

urlcleanup()

urlencode(query, doseq=0)

```
Encode a sequence of two-
element tuples or dictionary into a URL query string.

If any values in the query arg are sequences and doseq is true, each
sequence element is converted to a separate parameter.

If the query arg is a sequence of two-element tuples, the order of the
parameters in the output will match the order of parameters in the
input.
```

urlopen(url, data=None, proxies=None, context=None)

```
Create a file-like object for the specified URL to read from.
```

urlretrieve(url, filename=None, reporthook=None, data=None, context=None)

Data

__all__ = ['urlopen', 'URLopener', 'FancyURLopener', 'urlretrieve', 'urlcleanup', 'quote', 'quote_plus', 'unquote', 'unquote_plus', 'urlencode', 'url2pathname', 'pathname2url', 'splittag', 'localhost', 'thishost', 'ftperrors', 'basejoin', 'unwrap', 'splittype', 'splithost', ...]
__version__ = '1.17'