Watson - Http

Release 1.2.4

July 29, 2016

Contents

1 Build Status **3**

2 Installation **5**

3 Testing **7**

4 Contributing **9**

5 Table of Contents **11**

5.1 Usage 11

5.2 Reference Library 13

Python Module Index **23**

Work with HTTP Request/Response objects, sessions, and more.

CHAPTER 1

Build Status

CHAPTER 2

Installation

```
pip install watson-http
```

CHAPTER 3

Testing

Watson can be tested with py.test. Simply activate your virtualenv and run `python setup.py test`.

CHAPTER 4

Contributing

If you would like to contribute to Watson, please feel free to issue a pull request via Github with the associated tests for your code. Your name will be added to the AUTHORS file under contributors.

CHAPTER 5

Table of Contents

5.1 Usage

Tip: watson-http also works particularly well watson-form

5.1.1 Creating a Request

Requests can be instantiated directly from the class, or be created based on environ variables.

Note: Instantiating from the class itself will not populate the Request object with the relevant data from the current server request.

From the environ

```
from watson.http import messages

def application(environ, start_response):
    request = messages.Request.from_environ(environ)
    print(request.method)
```

Tip: watson-http also enables you to deal with other HTTP verbs that may not be accessible by a regular browser. Simply posting HTTP_REQUEST_METHOD and setting it to a valid HTTP verb will convert that request to the specific verb.

From watson.http.messages.Request

```
from watson.http import messages

def application(environ, start_response):
    request = messages.Request('get', get={'get_var': 'somevalue'})
    print(request.method) # get
    print(request.get('get_var')) # somevalue
```

Dealing with Sessions

Tip: You can access many things from the Request, and most work similar to a regular `dict`. These include: headers, server, cookies, get, post, files, url and sessions.

Earlier, we created a request with the *Request.from_environ* method. By default, all requests will be created with the *watson.http.sessions.File* backend for managing sessions. This however can be changed to a different backend by adding the *session_class* argument to the *from_environ* call. *session_class* must inherit from *watson.http.sessions.abc.StorageMixin*. If the class requires any additional configuration (the *http.sessions.file.Storage* class allows you to set the directory sessions are stored in), then you can also pass a dict of options via *session_options*.

```
from watson.http import messages

def application(environ, start_response):
    request = messages.Request.from_environ(environ, session_class=YOUR_SESSION_CLASS, session_optio
```

5.1.2 Creating a Response

While you can simply return a list from a WSGI application, you still need to also call the start_response method. While this maybe sufficient for smaller applications, anything larger requires a more robust approach. A standard WSGI callable may look like below:

```
def application(environ, start_response):
    start_response('200 OK', [('Content-Type', 'text/html')])
    return [b'Hello World']
```

With watson-http this code turns into...

```
from watson.http import messages

def application(environ, start_response):
    response = messages.Response(200, body='Hello World!')
    return response(start_response)
```

The response body by default is interpreted as utf-8, however this can be modified by accessing the response headers.

```
response = messages.Response(200)
response.headers.add('Content-Type', 'text/html; charset=ENCODING')
```

5.1.3 Putting it all together

An example app that outputs get variables may look like:

```
from watson.http import messages

def application(environ, start_response):
    request = messages.create_request_from_environ(environ)

    response = messages.Response(200, body='Hello {name}!'.format(request.get('name', 'World')))
    return response(start_response)
```

When you navigate to `/` you will be presented with 'Hello World!', however if you navigate to `/?name=Simon`, you will be presented with 'Hello Simon!'

5.2 Reference Library

5.2.1 watson.http

```python
# -*- coding: utf-8 -*-
__version__ = '1.2.4'

STATUS_CODES = {
    100: 'Continue',
    101: 'Switching Protocols',
    102: 'Processing',
    200: 'OK',
    201: 'Created',
    202: 'Accepted',
    203: 'Non-Authoritative Information',
    204: 'No Content',
    205: 'Reset Content',
    206: 'Partial Content',
    207: 'Multi-Status',
    208: 'Already Reported',
    226: 'IM Used',
    300: 'Multiple Choices',
    301: 'Moved Permanently',
    302: 'Found',
    303: 'See Other',
    304: 'Not Modified',
    305: 'Use Proxy',
    306: 'Switch Proxy',
    307: 'Temporary Redirect',
    308: 'Permanent Redirect',
    400: 'Bad Request',
    401: 'Unauthorized',
    402: 'Payment Required',
    403: 'Forbidden',
    404: 'Not Found',
    405: 'Method Not Allowed',
    406: 'Not Acceptable',
    407: 'Proxy Authentication Reqiured',
    408: 'Request Timeout',
    409: 'Conflict',
    410: 'Gone',
    411: 'Length Required',
    412: 'Precondition Failed',
    413: 'Request Entity Too Large',
    414: 'Request-URI Too Long',
    415: 'Unspported Media Type',
    416: 'Requested Range Not Satisfiable',
    417: 'Exception Failed',
    418: "I'm a teapot",
    420: 'Enhance Your Calm',
    422: 'Unprocessable Entity',
    423: 'Locked',
    424: 'Method Failure',
    425: 'Unordered Collection',
    426: 'Upgrade Required',
    428: 'Precondition Required',
    429: 'Too Many Requested',
```

```
    431: 'Request Header Fields Too Large',
    444: 'No Response',
    449: 'Retry With',
    450: 'Blocked by Windows Parental Controls',
    451: 'Unavailable For Legal Reasons',
    494: 'Request Header Too Large',
    495: 'Cert Error',
    496: 'No Cert',
    497: 'HTTP to HTTPS',
    499: 'Client Closed Request',
    500: 'Internal Server Error',
    501: 'Not Implemented',
    502: 'Bad Gateway',
    503: 'Service Unavailable',
    504: 'Gateway Timeout',
    505: 'HTTP Version Not Supported',
    506: 'Variant Also Negotiates',
    507: 'Insufficient Storage',
    508: 'Loop Detected',
    509: 'Bandwidth Limit Exceeded',
    510: 'Not Extended',
    511: 'Network Authentication Required',
    598: 'Network read timeout error',
    599: 'Network connect timeout error'
}

REQUEST_METHODS = ('OPTIONS',
                   'GET',
                   'HEAD',
                   'POST',
                   'PUT',
                   'DELETE',
                   'TRACE',
                   'CONNECT')

MIME_TYPES = {
    'txt': ('text/plain',),
    'html': ('text/html', 'application/xhtml+xml'),
    'css': ('text/css',),
    'js': ('text/javascript', 'application/javascript'),
    'json': ('application/json',),
    'xml': ('text/xml', 'application/xml')
}
```

5.2.2 watson.http.cookies

class `watson.http.cookies.`**`CookieDict`**(*input=None*)

A dictionary containing cookies.

A basic extension of the SimpleCookie class from the standard library, but designed to work better with wsgi.

Example:

```
cd = CookieDict()
cookie = cd.add('my_cookie', 'some value')
print(cookie)  # my_cookie=some value
print(cd['my_cookie'])  # my_cookie=some value
```

add(*name*, *value=''*, *expires=0*, *path='/'*, *domain=None*, *secure=False*, *httponly=False*, *comment=None*)
Convenience method to add cookies to the dict.

Parameters

- **name** – the name of the cookie
- **value** – the value of the cookie
- **expires** – the expiration date for the cookie in seconds
- **path** – the path in which the cookie is valid
- **domain** – the domain in which the cookie is valid
- **secure** – only send cookies over https
- **httponly** – only send over http requests, not accessible via JavaScript
- **comment** – the associated comment with the cookie

Returns The morsel that was added to the CookieDict

delete(*name*)
Expire a cookie the next time it is sent to the browser.

Parameters **name** – the name of the cookie

expire()
Expire all the cookies in the dictionary.

merge(*cookie_dict*)
Merges an existing cookie dict into another cookie dict.

Parameters **cookie_dict** (CookieDict) – The cookie dict to merge

watson.http.cookies.**cookies_from_environ**(*environ*)
Converts a HTTP_COOKIE from an environ dict into a CookieDict.

5.2.3 watson.http.headers

class watson.http.headers.**HeaderCollection**(*environ=None*)
Retrieves header related variables from an environ.

Allows the use of non-capitalized names.

Example:

```
headers = HeaderCollection.from_environ(environ)
print(headers.get('Content-Type'))
```

__init__(*environ=None*)

add(*field*, *value*, *replace=False*, ***options*)
Add a header to the collection.

Parameters

- **field** (*string*) – The field name
- **value** (*mixed*) – The value of the field
- **replace** (*boolean*) – Whether or not to replace an existing field
- **options** (*kwargs*) – Any additional options for the header

Example:

```
headers = ...
headers.add('Content-Type', 'text/html', charset='utf-8')
```

classmethod from_environ(*environ*)
Instantiate the collection from an existing environ.

get(*field*, *option=None*, *default=None*)
Retrieve an individual header or it's option.

Example:

```
# Content-Type: text/html; charset=utf-8
headers = HeaderCollection()
headers.add('Content-Type', 'text/html', charset='utf-8')
option = headers.get('Content-Type', 'charset') # utf-8
```

Parameters

- **field** – the header field
- **option** – the option to retrieve from the field
- **default** – the default value if the option does not exist

Returns The default value or the value from the option

get_option(*field*, *option=None*, *default=None*)
Retrieve an individual header or it's option.

Example:

```
# Content-Type: text/html; charset=utf-8
headers = HeaderCollection()
headers.add('Content-Type', 'text/html', charset='utf-8')
option = headers.get('Content-Type', 'charset') # utf-8
```

Parameters

- **field** – the header field
- **option** – the option to retrieve from the field
- **default** – the default value if the option does not exist

Returns The default value or the value from the option

items()
Returns tuple pairs of environ vars and their values.

set(*field*, *value*, ***options*)
Add a header to the collection.

Any existing headers with the same name will be removed.

Parameters

- **field** (*string*) – The field name
- **value** (*mixed*) – The value of the field
- **options** (*kwargs*) – Any additional options for the header

Example:

```
headers = ...
headers.add('Content-Type', 'text/html', charset='utf-8')
```

class `watson.http.headers.`**`ServerCollection`**(*environ=None*)

Retrieves server related variables from an environ.

Example:

```
server = ServerCollection(environ)
print(server['SCRIPT_NAME'])
```

`__init__`(*environ=None*)

`items`()

Returns tuple pairs of environ vars and their values.

`watson.http.headers.`**`convert_to_http_field`**(*field*)

Convert a field from Title-Case to HTTP_UPPER_CASE.

`watson.http.headers.`**`convert_to_wsgi`**(*field*)

Convert a field name from UPPER_CASE to Title-Case.

`watson.http.headers.`**`fix_http_headers`**(*environ*, *remove=False*)

Add **HTTP_** to the relevant headers that its not included with.

5.2.4 watson.http.messages

class `watson.http.messages.`**`MessageMixin`**

Base mixin for all Http Message objects.

class `watson.http.messages.`**`Request`**(*environ*)

Provides a simple and usable interface for dealing with Http Requests. Requests are designed to be immutable and not altered after they are created, as such you should only set get/post/cookie etc attributes via the __init__. By default the session storage method is MemoryStorage which will store session in ram.

See:

- http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html
- http://ken.coar.org/cgi/draft-coar-cgi-v11-03.txt

Example:

```
request = Request.from_environ(environ)
print(request.method)
print(request.post('my_post_var'))

request = Request.from_dicts(server={'HTTP_METHOD': 'GET'}, get={'get_var': 'somevalue'})
print(request.method) # get
print(request.get('get_var')) # somevalue
```

`__init__`(*environ*)

`body`

Return the body of the request as a string.

If unable to decode, return empty body.

`host`()

Determine the real host of a request.

Returns X_FORWARDED_FOR header variable if set, otherwise a watson.http.uri.Url hostname attribute.

is_method(**methods*)

Determine whether or not a request was made via a specific method.

Example:

```
request = ... # request made via GET
request.is_method('get') # True
```

Parameters **method** (*string|list|tuple*) – the method or list of methods to check

Returns Boolean

is_secure()

Determine whether or not the request was made via Https.

Returns Boolean

is_xml_http_request()

Determine whether or not a request has originated via an XmlHttpRequest, assuming the relevant header has been set by the request.

Returns Boolean

json_body

Returns the body encoded as JSON.

post

A dict of all POST variables associated with the request.

Returns A dict of POST variables

class watson.http.messages.**Response**(*status_code=None*, *headers=None*, *body=None*, *version=None*)

Provides a simple and usable interface for dealing with Http Responses.

See:

- http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html

Example:

```
def app(environ, start_response):
    response = Response(200, body='<h1>Hello World!</h1>')
    response.headers.add('Content-Type', 'text/html', charset='utf-8')
    response.cookies.add('something', 'test')
    start_response(*response.start())
    return [response()]
```

__init__(*status_code=None*, *headers=None*, *body=None*, *version=None*)

Parameters

- **status_code** (*int*) – The status code for the Response
- **headers** (watson.http.headers.HeaderCollection) – Valid response headers.
- **body** (*string*) – The content for the response
- **version** (*string*) – The Http version for the response

body
Returns the decoded body based on the response encoding type.

cookies
Returns the cookies associated with the Response.

encoding
Retrieve the encoding for the response from the headers, defaults to UTF-8.

raw()
Return the raw encoded output for the response.

start()
Return the status_line and headers of the response for use in a WSGI application.

Returns The status line and headers of the response.

status_code
The status code for the Response.

status_line
The formatted status line including the status code and message.

5.2.5 watson.http.sessions.abc

class watson.http.sessions.abc.**StorageMixin**(*id=None*, *timeout=None*, *autosave=True*)
The base mixin for all session storage adapters.

By default, if no id is specified when the session is created a new session id will be generated. When a user is logged in, it is good practice to regenerate the id of the session id to prevent session hijacking.

If autosave is set to True, then when data is added to the session the save() method will be called. If set to False, the developer will be required to manually call the save() method themselves.

To function correctly sessions require that cookies are enabled in the users browser.

Example:

```
session = SessionStorageMethod()
# where SessionStorageMethod is a valid storage class
session['key'] = 'some value'
session['key'] # 'some value'
```

__init__(*id=None*, *timeout=None*, *autosave=True*)

Parameters

- **id** – the id of the session
- **timeout** – the expiry time from the current time in seconds
- **key** – the key used to reference the session id in a cookie
- **autosave** – save the contents on __setitem__

cookie_params
The cookie params used when saving the session id as a cookie.

data
The data associated with the session.

destroy()
Destroy the session data from storage, but leave the actual session intact.

exists()
Determine whether or not the session exists in storage.

Returns Boolean whether or not the session id exists.

generate_id()

Returns A new session id based on a random 24 char string

id
The id of the session.

load()
Loads the data from storage into the session. If the session data was set to expire before the current time, destroy the session.

regenerate_id()
Regenerate a new id for the session.

save()
Save the contents of the session into storage.

watson.http.sessions.abc.**random**() → x in the interval [0, 1).

5.2.6 watson.http.sessions.file

class watson.http.sessions.file.**Storage**(*id=None*, *timeout=None*, *autosave=True*, *storage=None*)
A file based storage adapter for session data.

By default it will store data in the systems temp directory, however this can be overriden in the __init__.

__init__(*id=None*, *timeout=None*, *autosave=True*, *storage=None*)
Initialize the File Storage object.

Parameters **storage** – where the files should be stored

5.2.7 watson.http.sessions.memcache

class watson.http.sessions.memcache.**Storage**(*id=None*, *timeout=None*, *autosave=True*, *config=None*)
A memcache based storage adapter for session data.

__init__(*id=None*, *timeout=None*, *autosave=True*, *config=None*)

5.2.8 watson.http.sessions.memory

class watson.http.sessions.memory.**Storage**(*id=None*, *timeout=None*, *autosave=True*)
A ram based storage adapter for session data.

5.2.9 watson.http.uri

class watson.http.uri.**Url**(*url*)
An object based representation of a Url.

__init__(*url*)

Initialize the url object.

Create a new Url object from either a well formed url string, a dict of key/values, or a ParseResult.

Parameters **url** (*mixed*) – The value to generate the url from.

subdomain

Returns the subdomain for the URL. With thanks: http://stackoverflow.com/questions/1189128/regex-to-extract-subdomain-from-url

5.2.10 watson.http.wsgi

watson.http.wsgi.**get_form_vars**(*environ*, *dict_type*)

Convert environ vars into GET/POST/FILES objects.

Process all get and post vars from a <form> and return MultiDict of each.

W

watson.http.cookies, 14
watson.http.headers, 15
watson.http.messages, 17
watson.http.sessions.abc, 19
watson.http.sessions.file, 20
watson.http.sessions.memcache, 20
watson.http.sessions.memory, 20
watson.http.uri, 20
watson.http.wsgi, 21

Symbols

__init__() (watson.http.headers.HeaderCollection method), 15
__init__() (watson.http.headers.ServerCollection method), 17
__init__() (watson.http.messages.Request method), 17
__init__() (watson.http.messages.Response method), 18
__init__() (watson.http.sessions.abc.StorageMixin method), 19
__init__() (watson.http.sessions.file.Storage method), 20
__init__() (watson.http.sessions.memcache.Storage method), 20
__init__() (watson.http.uri.Url method), 20

A

add() (watson.http.cookies.CookieDict method), 14
add() (watson.http.headers.HeaderCollection method), 15

B

body (watson.http.messages.Request attribute), 17
body (watson.http.messages.Response attribute), 18

C

convert_to_http_field() (in module watson.http.headers), 17
convert_to_wsgi() (in module watson.http.headers), 17
cookie_params (watson.http.sessions.abc.StorageMixin attribute), 19
CookieDict (class in watson.http.cookies), 14
cookies (watson.http.messages.Response attribute), 19
cookies_from_environ() (in module watson.http.cookies), 15

D

data (watson.http.sessions.abc.StorageMixin attribute), 19
delete() (watson.http.cookies.CookieDict method), 15
destroy() (watson.http.sessions.abc.StorageMixin method), 19

E

encoding (watson.http.messages.Response attribute), 19
exists() (watson.http.sessions.abc.StorageMixin method), 19
expire() (watson.http.cookies.CookieDict method), 15

F

fix_http_headers() (in module watson.http.headers), 17
from_environ() (watson.http.headers.HeaderCollection class method), 16

G

generate_id() (watson.http.sessions.abc.StorageMixin method), 20
get() (watson.http.headers.HeaderCollection method), 16
get_form_vars() (in module watson.http.wsgi), 21
get_option() (watson.http.headers.HeaderCollection method), 16

H

HeaderCollection (class in watson.http.headers), 15
host() (watson.http.messages.Request method), 17

I

id (watson.http.sessions.abc.StorageMixin attribute), 20
is_method() (watson.http.messages.Request method), 18
is_secure() (watson.http.messages.Request method), 18
is_xml_http_request() (watson.http.messages.Request method), 18
items() (watson.http.headers.HeaderCollection method), 16
items() (watson.http.headers.ServerCollection method), 17

J

json_body (watson.http.messages.Request attribute), 18

L

load() (watson.http.sessions.abc.StorageMixin method), 20

M

merge() (watson.http.cookies.CookieDict method), 15
MessageMixin (class in watson.http.messages), 17

P

post (watson.http.messages.Request attribute), 18

R

random() (in module watson.http.sessions.abc), 20
raw() (watson.http.messages.Response method), 19
regenerate_id() (watson.http.sessions.abc.StorageMixin method), 20
Request (class in watson.http.messages), 17
Response (class in watson.http.messages), 18

S

save() (watson.http.sessions.abc.StorageMixin method), 20
ServerCollection (class in watson.http.headers), 17
set() (watson.http.headers.HeaderCollection method), 16
start() (watson.http.messages.Response method), 19
status_code (watson.http.messages.Response attribute), 19
status_line (watson.http.messages.Response attribute), 19
Storage (class in watson.http.sessions.file), 20
Storage (class in watson.http.sessions.memcache), 20
Storage (class in watson.http.sessions.memory), 20
StorageMixin (class in watson.http.sessions.abc), 19
subdomain (watson.http.uri.Url attribute), 21

U

Url (class in watson.http.uri), 20

W

watson.http.cookies (module), 14
watson.http.headers (module), 15
watson.http.messages (module), 17
watson.http.sessions.abc (module), 19
watson.http.sessions.file (module), 20
watson.http.sessions.memcache (module), 20
watson.http.sessions.memory (module), 20
watson.http.uri (module), 20
watson.http.wsgi (module), 21