

---

# **Watson - Http**

***Release 1.1.2***

September 30, 2014



|          |                             |           |
|----------|-----------------------------|-----------|
| <b>1</b> | <b>Build Status</b>         | <b>3</b>  |
| <b>2</b> | <b>Installation</b>         | <b>5</b>  |
| <b>3</b> | <b>Testing</b>              | <b>7</b>  |
| <b>4</b> | <b>Contributing</b>         | <b>9</b>  |
| <b>5</b> | <b>Table of Contents</b>    | <b>11</b> |
| 5.1      | Usage . . . . .             | 11        |
| 5.2      | Reference Library . . . . . | 13        |
|          | <b>Python Module Index</b>  | <b>19</b> |



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



---

**Build Status**

---





---

## Installation

---

```
pip install watson-http
```



---

### Testing

---

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



---

## 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.



---

## Table of Contents

---

### 5.1 Usage

---

**Tip:** watson-http also works particularly well with 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.create_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 `create_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 `create_request_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.create_request_from_environ(environ, session_class=YOUR_SESSION_CLASS, session_options=YOUR_SESSION_OPTIONS)
```

### 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

```

1  # -*- coding: utf-8 -*-
2  __version__ = '1.1.2'
3
4  STATUS_CODES = {
5      100: 'Continue',
6      101: 'Switching Protocols',
7      102: 'Processing',
8      200: 'OK',
9      201: 'Created',
10     202: 'Accepted',
11     203: 'Non-Authoritative Information',
12     204: 'No Content',
13     205: 'Reset Content',
14     206: 'Partial Content',
15     207: 'Multi-Status',
16     208: 'Already Reported',
17     226: 'IM Used',
18     300: 'Multiple Choices',
19     301: 'Moved Permanently',
20     302: 'Found',
21     303: 'See Other',
22     304: 'Not Modified',
23     305: 'Use Proxy',
24     306: 'Switch Proxy',
25     307: 'Temporary Redirect',
26     308: 'Permanent Redirect',
27     400: 'Bad Request',
28     401: 'Unauthorized',
29     402: 'Payment Required',
30     403: 'Forbidden',
31     404: 'Not Found',
32     405: 'Method Not Allowed',
33     406: 'Not Acceptable',
34     407: 'Proxy Authentication Required',
35     408: 'Request Timeout',
36     409: 'Conflict',
37     410: 'Gone',
38     411: 'Length Required',
39     412: 'Precondition Failed',
40     413: 'Request Entity Too Large',
41     414: 'Request-URI Too Long',
42     415: 'Unsupported Media Type',
43     416: 'Requested Range Not Satisfiable',
44     417: 'Exception Failed',
45     418: "I'm a teapot",
46     420: 'Enhance Your Calm',
47     422: 'Unprocessable Entity',
48     423: 'Locked',
49     424: 'Method Failure',
50     425: 'Unordered Collection',
51     426: 'Upgrade Required',
52     428: 'Precondition Required',
53     429: 'Too Many Requested',

```

```
54     431: 'Request Header Fields Too Large',
55     444: 'No Response',
56     449: 'Retry With',
57     450: 'Blocked by Windows Parental Controls',
58     451: 'Unavailable For Legal Reasons',
59     494: 'Request Header Too Large',
60     495: 'Cert Error',
61     496: 'No Cert',
62     497: 'HTTP to HTTPS',
63     499: 'Client Closed Request',
64     500: 'Internal Server Error',
65     501: 'Not Implemented',
66     502: 'Bad Gateway',
67     503: 'Service Unavailable',
68     504: 'Gateway Timeout',
69     505: 'HTTP Version Not Supported',
70     506: 'Variant Also Negotiates',
71     507: 'Insufficient Storage',
72     508: 'Loop Detected',
73     509: 'Bandwidth Limit Exceeded',
74     510: 'Not Extended',
75     511: 'Network Authentication Required',
76     598: 'Network read timeout error',
77     599: 'Network connect timeout error'
78 }
79
80 REQUEST_METHODS = ('OPTIONS',
81                    'GET',
82                    'HEAD',
83                    'POST',
84                    'PUT',
85                    'DELETE',
86                    'TRACE',
87                    'CONNECT')
88
89 MIME_TYPES = {
90     'txt': ('text/plain',),
91     'html': ('text/html', 'application/xhtml+xml'),
92     'css': ('text/css',),
93     'js': ('text/javascript', 'application/javascript'),
94     'json': ('application/json',),
95     'xml': ('text/xml', 'application/xml')
96 }
```

## 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.

**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`

## 5.2.5 `watson.http.sessions.abc`

## 5.2.6 `watson.http.sessions.file`

## 5.2.7 `watson.http.sessions.memcache`

## 5.2.8 `watson.http.sessions.memory`

## 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`



## W

`watson.http.cookies`, [14](#)

`watson.http.headers`, [15](#)

`watson.http.uri`, [17](#)