wsgi_intercept Documentation

Titus Brown, Kumar McMillan, Chris Dent

Jul 06, 2018

Contents

Installs a WSGI application in place of a real host for testing.

CHAPTER 1

Introduction

Testing a WSGI application sometimes involves starting a server at a local host and port, then pointing your test code to that address. Instead, this library lets you intercept calls to any specific host/port combination and redirect them into a WSGI application importable by your test program. Thus, you can avoid spawning multiple processes or threads to test your Web app.

CHAPTER 2

Supported Libaries

`wsgi_intercept` works with a variety of HTTP clients in Python 2.7, 3.4 and beyond, and in pypy.

- urllib2
- urllib.request
- httplib
- http.client
- httplib2
- requests
- urllib3

CHAPTER 3

How Does It Work?

`wsgi_intercept` works by replacing `httplib.HTTPConnection` with a subclass, `wsgi_intercept.WSGI_HTTPConnection`. This class then redirects specific server/port combinations into a WSGI application by emulating a socket. If no intercept is registered for the host and port requested, those requests are passed on to the standard handler.

The easiest way to use an intercept is to import an appropriate subclass of `~wsgi_intercept.interceptor.Interceptor` and use that as a context manager over web requests that use the library associated with the subclass. For example:

```
import httplib2
from wsgi_intercept.interceptor import Httplib2Interceptor
from mywsgiapp import app

def load_app():
    return app

http = httplib2.Http()
with Httplib2Interceptor(load_app, host='example.com', port=80) as url:
    response, content = http.request('%s%s' % (url, '/path'))
    assert response.status == 200
```

The interceptor class may aslo be used directly to install intercepts. See the module documentation for more information.

Older versions required that the functions `add_wsgi_intercept(host, port, app_create_fn, script_name='')` and `remove_wsgi_intercept(host,port)` be used to specify which URLs should be redirected into what applications. These methods are still available, but the `Interceptor` classes are likely easier to use for most use cases.

Note: `app_create_fn` is a *function object* returning a WSGI application; `script_name` becomes `SCRIPT_NAME` in the WSGI app's environment, if set.

Note: If `http_proxy` or `https_proxy` is set in the environment this can cause difficulties with some of the intercepted libraries. If requests or urllib is being used, these will raise an exception if one of those variables is set.

Note: If `wsgi_intercept.STRICT_RESPONSE_HEADERS` is set to `True` then response headers sent by an application will be checked to make sure they are of the type `str` native to the version of Python, as required by pep 3333. The default is `False` (to preserve backwards compatibility)

CHAPTER 4

Install

```
pip install -U wsgi_intercept
```

CHAPTER 5

Packages Intercepted

Unfortunately each of the HTTP client libraries use their own specific mechanism for making HTTP call-outs, so individual implementations are needed. At this time there are implementations for `httplib2`, `urllib3` and `requests` in both Python 2 and 3, `urllib2` and `httplib` in Python 2 and `urllib.request` and `http.client` in Python 3.

If you are using Python 2 and need support for a different HTTP client, require a version of `wsgi_intercept<0.6`. Earlier versions include support for `webtest`, `webunit` and `zope.testbrowser`.

The best way to figure out how to use interception is to inspect the tests. More comprehensive documentation available upon request.

CHAPTER 6

History

Pursuant to Ian Bicking's "best Web testing framework" post, Titus Brown put together an in-process HTTP-to-WSGI interception mechanism for his own Web testing system, twill. Because the mechanism is pretty generic – it works at the httplib level – Titus decided to try adding it into all of the *other* Python Web testing frameworks.

The Python 2 version of wsgi-intercept was the result. Kumar McMillan later took over maintenance.

The current version is tested with Python 2.7, 3.4, 3.5, 3.6, and pypy and was assembled by Chris Dent. Testing and documentation improvements from Sasha Hart.

CHAPTER 7

Project Home

This project lives on GitHub. Please submit all bugs, patches, failing tests, et cetera using the Issue Tracker.

Additional documentation is available on Read The Docs.

CHAPTER 8

Examples

8.1 Interceptor

Context manager based WSGI interception.

class wsgi_intercept.interceptor.**HttpClientInterceptor**(*app*, *host=None*, *port=80*, *prefix=None*, *url=None*)

Interceptor for httplib and http.client.

class wsgi_intercept.interceptor.**Httplib2Interceptor**(*app*, *host=None*, *port=80*, *prefix=None*, *url=None*)

Interceptor for httplib2.

class wsgi_intercept.interceptor.**Interceptor**(*app*, *host=None*, *port=80*, *prefix=None*, *url=None*)

A convenience class over the guts of wsgi_intercept.

An Interceptor subclass provides a clean entry point to the wsgi_intercept functionality in two ways: by encapsulating the interception addition and removal in methods and by providing a context manager that automates the process of addition and removal.

Each Interceptor subclass is associated with a specific http library.

Each class may be passed a url or a host and a port. If no args are passed a hostname will be automatically generated and the resulting url will be returned by the context manager.

class wsgi_intercept.interceptor.**RequestsInterceptor**(*app*, *host=None*, *port=80*, *prefix=None*, *url=None*)

Interceptor for requests.

class wsgi_intercept.interceptor.**Urllib3Interceptor**(*app*, *host=None*, *port=80*, *prefix=None*, *url=None*)

Interceptor for requests.

class wsgi_intercept.interceptor.**UrllibInterceptor**(*app*, *host=None*, *port=80*, *prefix=None*, *url=None*)

Interceptor for urllib2 and urllib.request.

Example using *httplib2*, others are much the same:

```
import httplib2
from wsgi_intercept.interceptor import Httplib2Interceptor


def app(environ, start_response):
    start_response('200 OK', [('Content-Type', 'text/plain')])
    return [b'Whee']


def make_app():
    return app


http = httplib2.Http()
with Httplib2Interceptor(make_app, host='localhost', port=80) as url:
    resp, content = http.request(url)
    assert content == b'Whee'
```

8.2 http_client_intercept

Intercept HTTP connections that use httplib (Py2) or http.client (Py3).

Warning: This intercept will fail to install if you access access HTTPConnection or HTTPSConnection before the intercept is installed. For example, do not use "from http.client import HTTPConnection". Instead, "import http.client" and reference http.client.HTTPConnection after the intercept is installed.

Example:

```
try:
    import http.client as http_lib
except ImportError:
    import httplib as http_lib
from wsgi_intercept import (
    http_client_intercept, add_wsgi_intercept, remove_wsgi_intercept
)


def app(environ, start_response):
    start_response('200 OK', [('Content-Type', 'text/plain')])
    return [b'Whee']


def make_app():
    return app


host, port = 'localhost', 80
http_client_intercept.install()
add_wsgi_intercept(host, port, make_app)
HTTPConnection = http_lib.HTTPConnection
client = HTTPConnection(host)
```

(continues on next page)

(continued from previous page)

```
client.request('GET', '/')
response = client.getresponse()
content = response.read()
assert content == b'Whee'
remove_wsgi_intercept(host, port)
http_client_intercept.uninstall()
```

8.3 httplib2_intercept

Note: No effort is made to pass SSL certificate or version information to the the underlying `HTTPSConnection`. The assumption is that wsgi-intercept is testing the behavior of the application, not the connection.

Example:

```
import httplib2
from wsgi_intercept import httplib2_intercept, add_wsgi_intercept


def app(environ, start_response):
    start_response('200 OK', [('Content-Type', 'text/plain')])
    return [b'Whee']


def make_app():
    return app


host, port = 'localhost', 80
url = 'http://{0}:{1}/'.format(host, port)
httplib2_intercept.install()
add_wsgi_intercept(host, port, make_app)
http = httplib2.Http()
resp, content = http.request(url)
assert content == b'Whee'
httplib2_intercept.uninstall()
```

8.4 requests_intercept

Intercept HTTP connections that use requests.

Example:

```
import requests
from wsgi_intercept import requests_intercept, add_wsgi_intercept


def app(environ, start_response):
    start_response('200 OK', [('Content-Type', 'text/plain')])
    return [b'Whee']
```

(continues on next page)

(continued from previous page)

```
def make_app():
    return app


host, port = 'localhost', 80
url = 'http://{0}:{1}/'.format(host, port)
requests_intercept.install()
add_wsgi_intercept(host, port, make_app)
resp = requests.get(url)
assert resp.content == b'Whee'
requests_intercept.uninstall()
```

8.5 urllib3_intercept

Intercept HTTP connections that use urllib3.

Example:

```
import urllib3
from wsgi_intercept import urllib3_intercept, add_wsgi_intercept

pool = urllib3.PoolManager()


def app(environ, start_response):
    start_response('200 OK', [('Content-Type', 'text/plain')])
    return [b'Whee']


def make_app():
    return app


host, port = 'localhost', 80
url = 'http://{0}:{1}/'.format(host, port)
urllib3_intercept.install()
add_wsgi_intercept(host, port, make_app)
resp = pool.request('GET', url)
assert resp.data == b'Whee'
urllib3_intercept.uninstall()
```

8.6 urllib_intercept

Intercept HTTP connections that use urllib.request (Python 3) aka urllib2 (Python 2).

Example:

```
try:
    from urllib.request import urlopen
except ImportError:
    from urllib2 import urlopen
```

(continues on next page)

(continued from previous page)

```
from wsgi_intercept import (
    urllib_intercept, add_wsgi_intercept, remove_wsgi_intercept
)


def app(environ, start_response):
    start_response('200 OK', [('Content-Type', 'text/plain')])
    return [b'Whee']


def make_app():
    return app


host, port = 'localhost', 80
url = 'http://{0}:{1}/'.format(host, port)
urllib_intercept.install_opener()
add_wsgi_intercept(host, port, make_app)
stream = urlopen(url)
content = stream.read()
assert content == b'Whee'
remove_wsgi_intercept()
```

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Python Module Index

W

Index