HTTPretty Documentation

Release 0.8.14

Gabriel Falcão

February 16, 2017

Contents

Contents:

CHAPTER 1

What is HTTPretty ?

Once upon a time a python developer wanted to use a RESTful api, everything was fine but until the day he needed to test the code that hits the RESTful API: what if the API server is down? What if its content has changed ?

Don't worry, HTTPretty is here for you:

```
import requests
from sure import expect
import httpretty


@httpretty.activate
def test_yipit_api_returning_deals():
    httpretty.register_uri(httpretty.GET, "http://api.yipit.com/v1/deals/",
                           body='[{"title": "Test Deal"}]',
                           content_type="application/json")

    response = requests.get('http://api.yipit.com/v1/deals/')

    expect(response.json()).to.equal([{"title": "Test Deal"}])
```

A more technical description

HTTPretty is a HTTP client mock library for Python 100% inspired on ruby's [FakeWeb](http://fakeweb.rubyforge.org/). If you come from ruby this would probably sound familiar :smiley:

Installing

Installing httpretty is as easy as:

```
pip install HTTPretty
```

CHAPTER 2

Demo

expecting a simple response body

```
import requests
import httpretty

def test_one():
    httpretty.enable()  # enable HTTPretty so that it will monkey patch the socket module
    httpretty.register_uri(httpretty.GET, "http://yipit.com/",
                           body="Find the best daily deals")

    response = requests.get('http://yipit.com')

    assert response.text == "Find the best daily deals"

    httpretty.disable()  # disable afterwards, so that you will have no problems in code that uses t
    httpretty.reset()    # reset HTTPretty state (clean up registered urls and request history)
```

CHAPTER 3

Motivation

When building systems that access external resources such as RESTful webservices, XMLRPC or even simple HTTP requests, we stumble in the problem:

> *"I'm gonna need to mock all those requests"*

It brings a lot of hassle, you will need to use a generic mocking tool, mess with scope and so on.

The idea behind HTTPretty (how it works)

HTTPretty monkey patches Python's socket core module, reimplementing the HTTP protocol, by mocking requests and responses.

As for how it works this way, you don't need to worry what http library you're gonna use.

HTTPretty will mock the response for you :) *(and also give you the latest requests so that you can check them)*

CHAPTER 4

Acknowledgements

caveats

`forcing_headers` + `Content-Length`

if you use the `forcing_headers` options make sure to add the header `Content-Length` otherwise the [requests](http://docs.python-requests.org/en/latest/) will try to load the response endlessly

supported libraries

Because HTTPretty works in the socket level it should work with any HTTP client libraries, although it is battle tested against:

- requests
- httplib2
- urllib2

CHAPTER 5

API Reference

CHAPTER 6

Indices and tables

- genindex
- modindex
- search