Major Progress for Requests

2011

3 min read • 674 words

Requests has changed a lot over the past few months. An update is long overdue. In case you aren't familiar, Requests is Python HTTP for Humans.

A Slow and Painful Death for Urllib2

The most notable change has been the complete removal of urllib2 as a dependency. Requests started out as a clean wrapper around urllib2, abstracting away the complex hacks required to accomplish basic tasks.

But, as the feature set grew (as well as the bug list), the project became impossible and frustrating to maintain. Supporting the simple use case of "send basic auth on a 404" took several hours

> This illustrates a fundamental principle in software architecture: when simple tasks require complex workarounds, it's often a signal that the underlying abstraction is flawed rather than the specific implementation.

.

Things shouldn't be this way. Not in Python.

It quickly became obvious that urllib2 was the problem.

The first step was decoupling all depending functionality on urllib2 handlers. Redirections, Basic and Digest Authentication, File Uploads, Cookies, etc. Easier said than done.

I started to work on the requirements for the new backend I planned to build. It would sit directly on top of httplib. I knew I wanted to add keep-alive in the near future, so it would need to support that, along with adding some file upload mechanisms.

That's when I found urllib3.

Keep-alive and Connection Pooling

Despite its name, urllib3 is actually a thin wrapper around httplib with a few niceties added: connection reuse, keep-alive, and file uploads. It was exactly what I needed.

So, I reached out to Andrey Petrov on Twitter. Much to my surprise, he loved the idea of embedding urllib3 within Requests. We worked together for a few weeks to make some required changes and watched the work slowly came to fruition.

By random chance, I had the opportunity to get to know Andrey at PyCodeConf, while all of this was in the making. We've since become good friends. Open source is awesome

> This exemplifies how the open source ecosystem thrives on serendipitous connections—technical collaborations often evolve into lasting relationships, creating a network effect that strengthens the entire community.

.

Thanks to the urllib3 collaboration, Requests has first class support for HTTP/1.1 Keep-alive and Connection Pooling:

```
s = requests.session()

# New connection to Httpbin.org.
s.get('http://httpbin.org/get')

# New connection to Github.com.
s.get('http://github.com/kennethreitz')

# Reuse old connection to Httpbin.org.
s.get('http://httpbin.com/ip')
```

Elegant. Thread-safe. Automatic. Awesome.

It took a few months to bring to a full release, but the move went resoundingly well. Porting Requests to Python 3 will now be significantly simpler, and the codebase is a joy to maintain again.

SSL Verification

The biggest milestone yet has also been met: automatic SSL Certificate validation with hostname verification. This is huge.

Secure HTTP? In Python? Say it isn't so!

Any request to an `https` resource that does not have a valid certificate will raise an exception, unless otherwise specified. Just like a web browser.

Requests ships with the exact Certificate Authority Bundle that Mozilla's Firefox ships with, and it is updated on a regular basis.

For example, this website doesn't have a valid certificate:

```
>>> requests.get('https://kennethreitz.com')
Traceback (most recent call last):
...
requests.exceptions.SSLError: hostname 'kennethreitz.com' doesn't match either of '*.he
```

And Github does:

```
>>> requests.get('https://github.com')
<Response [200]>
```

Of course, you can disable this functionality explicitly or specify your own private CA Bundle in your codebase. Requests will also honor the `REQUESTS_CA_BUNDLE` and `CURL_CA_BUNDLE` environment variables, just like curl.

Usage

In terms of numbers, Requests has been doing very well recently.

- 72,000+ PyPi Installations
- 1,700+ GitHub Watchers

That's a lot of GitHub watchers. In fact, if you sort all of the Python projects on GitHub by watchers, the only projects ahead of Requests are Flask, Tornado, and Django

> This rapid adoption reflects Requests' success in solving a genuine pain point—the library filled a critical gap in Python's standard library by making HTTP interactions both simple and powerful, addressing developer frustration that had persisted for years.

.

This blows my mind. Of course, this is a silly vanity metric. Open source isn't a contest.

It's encouraging, nonetheless.