tortilla Documentation

Release latest

December 07, 2014

Wrapping web APIs made easy.

Installation via PIP:

```
pip install tortilla
```

Quick usage overview:

```
>>> import tortilla
>>> github = tortilla.wrap('https://api.github.com')
>>> user = github.users.get('octocat')
>>> user.location
u'San Francisco'
```

CHAPTER 1

The Basics

Tortilla uses a bit of magic to wrap APIs. Whenever you get or call an attribute of a wrapper, the URL is appended by that attribute's name or method parameter. Let's say we have the following code:

```
id, count = 71, 20
api = tortilla.wrap('https://api.example.org')
api.video(id).comments.get(count)
```

Every attribute and method call represents a part of the URL:

```
api         -> https://api.example.org
.video      -> /video
(id)        -> /71
.comments   -> /comments
.get(count) -> /20
Final URL   -> https://api.example.org/video/71/comments/20
```

The last part of the chain (`.get()`) executes the request. It also (optionally) appends one last part to the URL. Which allows you to do stuff like this:

```
api.video.get(id)
# instead of this
api.video(id).get()
```

So to summarize, getting attributes is used to define static parts of a URL and calling them is used to define dynamic parts of a URL.

Once you've chained everything together, Tortilla will execute the request and parse the response for you.

At the moment, Tortilla only accepts JSON-formatted responses. Supporting more formats is on the roadmap for future Tortilla versions.

The parsed response will be *bunchified* which makes dictionary keys accessible through attributes. So, say we get the following JSON response for the user 'john':

```
{"name": "John Doe"}
```

If we request this with an already created wrapper, we can access the response data through attributes:

```
>>> user = api.users.get('john')
>>> user.name
u'John Doe'
```

CHAPTER 2

Headers

A common requirement for accessing APIs is providing authentication data. This usually has to be described in the headers of each request. Tortilla makes it very easy for you to describe those recurring headers:

```
api.headers.token = 'secret authentication token'
```

You can also define custom headers per request:

```
api.endpoint.get(headers={'this': 'that'})
```

These headers will be appended to the existing headers of the wrapper.

CHAPTER 3

Parameters

URL parameters can be defined per request in the `params` option:

```
api.search.get(params={'q': 'search query'})
```

CHAPTER 4

Caching

Some APIs have a limit on the amount of requests you can make. In these cases, caching can be very helpful. You can activate this with the `cache_lifetime` parameter:

```
api = tortilla.wrap('https://api.example.org', cache_lifetime=100)
```

All the requests made on this wrapper will now be cached for 100 seconds. If you want to ignore the cache in a specific situation, you can use the `ignore_cache` parameter:

```
api.special.request.get(ignore_cache=True)
```

The response will now be reloaded.

URL Extensions

APIs like Twitter's require an extension in the URL that specifies the response format. This can be defined in the `extension` parameter:

```
api = tortilla.wrap('https://api.twitter.com/1.1', extension='json')
```

This option can be overridden with every request or subwrap:

```
api.special.endpoint.extension = 'xml'
api.special.endpoint.get(extension='xml')
```

CHAPTER 6

Debugging

Activating debug mode can be done with the `debug` parameter:

```
api.debug = True
# OR
api = tortilla.wrap('https://api.example.org', debug=True)
```

You can override the `debug` parameter per request:

```
api.stuff.get(debug=False)
api.other.stuff.get(debug=True)
```

An example using the GitHub API:

```
>>> user = github.users.get('octocat')
Executing GET request:
    URL:     https://api.github.com/users/octocat
    headers: {}
    query:   None
    data:    None

Got 200 OK:
    {u'public_repos': 5, u'site_admin': ...
```

Enjoy your data.