Gimlet

Release 0.5.1

November 30, 2014

Contents

Gimlet is a Python infrastructure block to provide versatile key-value 'session' storage for WSGI applications. The design philosophy is 'as fast as possible with slow components', which is to say, I/O load will be minimized, but it will work with fairly simple and mature storage backends. It provides:

- Easy setup and configuration
- Key-value dict-like session access
- Multiple backend options, including redis and SQL
- Efficient - absolute minimal I/O load
- Optional client-side storage for a whitelist of keys

It is also:

- 2 oz gin
- 1/2 oz lime juice
- 1/4 oz simple syrup
- lime garnish

Get the code at GitHub.

CHAPTER 1

Quick Start

Gimlet provides a WSGI Middleware which populates a Session object in the WSGI environ. The most simple setup looks like:

```
from gimlet.middleware import SessionMiddleware
from gimlet.backends.pyredis import RedisBackend

backend = RedisBackend()

app = SuperAwesomeApp()
app = SessionMiddleware(app, 's3krit', backend)
```

Inside your app code, you can access the session like:

```
session = environ['gimlet.session']
session['user'] = 'Frodo'
```

The session data will automatically be persisted at the end of the request.

A unique identifier for the session (also visible to the client) is available as `session.id`.

CHAPTER 2

Key Options

Typical web applications tend to have a concentration of session access on a relatively small set of keys, with small values. For example, a common session variable may be a key referencing the logged-in user's account. Gimlet provides fuctionality to store these commonly-accessed small keys on the clientside, in the session cookie. This substantially reduces the I/O load of the application as awhole, without limiting the session flexibility (particularly important for adapting legacy apps).

To specify that a key should be stored on the client, pass the `clientside=True` argument:

```
session.set('cart_id', 12345, clientside=True)
```

Warning: Keys that are stored on the client side are not encrypted by default, so it will be possible for eavesdroppers or end users to view their contents. They are signed, however, so they cannot be modified without detection. To enable encryption of cookies, supply a random 64-char hex string as the `encryption_key` argument to `SessionMiddleware`.

Keys can also be set as permanent or not. For example:

```
session.set('account_id', 777, permanent=False)
```

Or, combined:

```
session.set('cart_id', 12345, clientside=True, permanent=True)
```

CHAPTER 3

Contents

3.1 API Reference

3.2 Contributing

Patches and suggestions are strongly encouraged! Github pull requests are preferred, but other mechanisms of feedback are welcome.

Gimlet has a comprehensive test suite with 100% line and branch coverage, as reported by the excellent `coverage` module. To run the tests, simply run in the top level of the repo:

```
$ nosetests
```

There are no PEP8 or Pyflakes warnings in the codebase. To verify that:

```
$ pip install pep8 pyflakes
$ pep8 -r .
$ pyflakes .
```

Any pull requests must maintain the sanctity of these three pillars.

You can test these three things on all supported platforms with Tox:

```
$ tox
```

3.3 Gimlet Changelog

3.3.1 Version 0.5

- Remove support for hybrid HTTP/HTTPS environments. The use cases for this are essentially gone with the advent of OCSP, SPDY, etc, and removing support simplifies things and streamlines the cookie payload.

3.3.2 Version 0.4

- ?

3.3.3 Version 0.3

- ?

3.3.4 Version 0.2

- ?

3.3.5 Version 0.1

- Initial release.

CHAPTER 4

License

Gimlet is licensed under an MIT license. Please see the LICENSE file for more information.

g

G