wsgi-statsd Documentation

Release latest

January 22, 2015

Contents

wsgi_statsd is a WSGI middleware that provides an easy way to time all requests. Integration is as easy as just wrapping your existing wsgi application.

Contents

CHAPTER 1

Usage

In your wsgi.py file wrap your WSGI application as follows:

```
import statsd
from wsgi_statsd import StatsdTimingMiddleware


def application(environ, start_response):
    response_body = 'The request method was %s' % environ['REQUEST_METHOD']
    status = '200 OK'
    response_headers = [('Content-Type', 'text/plain'),
                        ('Content-Length', str(len(response_body)))]

    start_response(status, response_headers)

    return [response_body]

client = statsd.StatsClient(
    prefix='your_prefix',
    host='your_host',
    port=8125
)

application = StatsdTimingMiddleware(application, client)
```

Note: If an unhandled exception happens, it will not be timed by default. This is a design decision to separate error reporting and actual statistical measurements. To enable exception timing, pass *time_exception=True* to the middleware constructor:

```
application = StatsdTimingMiddleware(application, client, time_exceptions=True)
```

CHAPTER 2

What it does

wsgi-statsd uses the statsd timer function, generates a key and sets the amount of time the request took as the value. The key is constructed as follows:

```
<statsd-client-prefix>.<separator-joined-path>.<request-method>.<response-code>.<exception-
```

Using the `foo` prefix, in your statsd client, and calling the `www.spam.com/bar/test/` page will result in `foo.bar_test.GET.200` having a value equal to the time it took to handle the request.

If you passed *time_exceptions=True* and any exception occurs during the response, then the key name will be postfixed with the exception class name: `foo.bar_test.GET.500.ValueError`.

CHAPTER 3

Customizing for your needs

It's possible to customize the way `wsgi_statsd` generates the key and/or time. `StatsdTimingMiddleware` has `send_stats` and `get_key_name` which you can override:

```
class CustomStatsdMiddleware(StatsdTimingMiddleware):

    def get_key_name(self, environ, response_interception):
        return super(self, CustomStatsdMiddleware).get_key_name(environ, response_interception) + '.'


    def send_stats(self, start, environ, response_interception):
        super(self, CustomStatsdMiddleware).send_stats(start + 10, environ, response_interception)
```

wsgi-statsd uses underscores as a separator for the path part of the key that is sent to statsd as that makes it easy to retrieve the data from graphite. You can override this default by passing a `separator` value to the middleware constructor:

```
StatsdTimingMiddleware(application, client, separator='.')
```

CHAPTER 4

Contact

If you have questions, bug reports, suggestions, etc. please create an issue on the GitHub project page.

CHAPTER 5

License

This software is licensed under the MIT license.

Please refer to the license file.