django-web-profiler Documentation

Release 0.0.1

django-web-profiler

Jun 07, 2017

Contents

CHAPTER 1

Introduction:

django-web-profiler is a django profiling tool which logs, stores debug toolbar statistics and also a set of URL's statistics using a management command. It logs request values such as device, ip address, user cpu time, system cpu time, No of queries, sql time, no of cache calls, missing, setting data cache calls for a particular url.

It provides a basic UI, which will differentiate development url statistics, production level statistics which generates using a management command.

Source Code is available in Micropyramid Repository(https://github.com/MicroPyramid/django-web-profiler).

Modules used:

- Python >= 2.6 (or Python 3.4)
- Django = 1.11.2
- Django Compressor = 2.1.1
- Django Debug Toolbar = 1.8
- requests = 2.17.3
- JQuery >= 1.7

CHAPTER 2

Installation Procedure

1. Install django-web-packer using the following command:

```
pip install django-web-profiler

        (or)

git clone git://github.com/micropyramid/django-web-profiler.git

cd django-web-profiler

python setup.py install
```

2. Add app name in settings.py:

```
INSTALLED_APPS = [
   '..................',
   'compressor',
   'debug_toolbar',
   'django_web_profiler',
   '..................'
]
```

3. Add 'django_web_profiler.middleware.DebugLoggingMiddleware' to your project middlewares:

    MIDDLEWARE = ['..................', 'django_web_profiler.middleware.DebugLoggingMiddleware'
        '..................'

    ]

    Disable 'debug_toolbar.middleware.DebugToolbarMiddleware' if you've already using it.

4. Make sure that 'debug-toolbar' has enabled for your application. After installing debug toolbar, add the following details to settings.py:

    INTERNAL_IPS = ('127.0.0.1',)

5. After installing/cloning, add the following details in settings file about urls, logger names:

```
URLS = ['http://stage.testsite.com/', 'http://stage.testsite.com/testing/']
```

6. Add the following logger to your existing loggers and create a folder called 'logs' where all profiler log files are stored:

```
    'request-logging': {
        'level': 'DEBUG',
        'handlers': ['console', 'file_log'],
        'propagate': False,
    },

Here file_log is a handler which contains a path where log files are stored.
```

CHAPTER 3

Sample Application

1. Install application requirements using the following command:

```
pip install -r requirements.txt
```

2. Load the application load using the following command:

```
python sandbox/manage.py loaddata sandbox/fixtures/users.json
```

3. Using the following command, we can generate url statistics in production environment i.e debug=False:

   python sandbox/manage.py logging_urls

We are always looking to help you customize the whole or part of the code as you like.

Visit our Django Development page Here

We welcome your feedback and support, raise github ticket if you want to report a bug. Need new features? Contact us here

or

mailto:: "hello@micropyramid.com"