muffin Documentation

Release latest

August 29, 2015

Contents

The Muffin – A web framework based on Asyncio stack `(early beta)`

Muffin is a fast, simple and asyncronous web-framework for Python 3.

Example "Hello User" with the Muffin:

```
import muffin


app = muffin.Application('example')


@app.register('/', '/hello/{name}')
def hello(request):
    name = request.match_info.get('name', 'anonymous')
    return 'Hello %s!' % name
```

Save the script as *example.py* and run it:

```
$ muffin example run
```

Open http://fuf.me:5000, http://fuf.me:5000/hello/username in your browser. Enjoy!

Contents

CHAPTER 1

Plugins

The list of Muffin plugins:

- Muffin-Admin – Basic Admin interface
- Muffin-Babel – Localization support
- Muffin-DebugToolbar – Debug Toolbar
- Muffin-Jade – Jade templates
- Muffin-Jinja2 – Jinja2 templates
- Muffin-Metrics – Send metrics to Graphite/Statsd
- Muffin-Mongo – MongoDB (pymongo) support
- Muffin-OAuth – OAuth client
- Muffin-Peewee – Peewee support (SQL, ORM)
- Muffin-REST – Helpers for building REST API
- Muffin-Redis – Redis support
- Muffin-Sentry – Sentry integration
- Muffin-Session – User session (auth)

CHAPTER 2

Requirements

- python >= 3.3

CHAPTER 3

Benchmarks

You could find some tests here: http://klen.github.io/py-frameworks-bench/

CHAPTER 4

Installation

The Muffin should be installed using pip:

```
pip install muffin
```

CHAPTER 5

Usage

See more in the example application sources. The application is deployed on Heroku: https://muffin-py.herokuapp.com

Run example server locally:

```
$ make -C example run
```

And open http://fuf.me:5000 in your browser.

5.1 Configuration

Muffin gets configuration options from python files. By default the package tries to load a configuration from *config* module (config.py).

There are few ways to redifine configuration module:

- Set configuration module in your app initialization:

```
    app = muffin.Application('myapp', CONFIG='config.debug')
```

- Set environment variable *MUFFIN_CONFIG*:

```
    $ MUFFIN_CONFIG=settings_local muffin example run
```

Also you can define any options while initializing your application:

```
app = muffin.Application('myapp', DEBUG=True, ANY_OPTION='Here', ONE_MORE='Yes')
```

5.1.1 Base application options

Base Muffin options and default values:

```
# Configuration module
'CONFIG': 'config'

# Enable debug mode
'DEBUG': False

# Logging options
'LOG_LEVEL': 'WARNING'
'LOG_FORMAT': '%(asctime)s [%(process)d] [%(levelname)s] %(message)s'
'LOG_DATE_FORMAT': '[%Y-%m-%d %H:%M:%S %z]'
```

```
# List of enabled plugins
'PLUGINS': []

# Setup static files in development
'STATIC_PREFIX': '/static'
'STATIC_FOLDERS': ['static']
```

5.1.2 Configuring logging

You can define your logging configurations with Python dictConfig format and place in *LOGGING* conf:

```
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'formatters': {
        'default': {
            'format': '%(asctime)s %(levelname)s %(name)s %(message)s'
        },
    },
    'handlers': {
        'logfile': {
            'level': 'DEBUG',
            'class': 'logging.handlers.RotatingFileHandler',
            'filename': 'my_log.log',
            'maxBytes': 50 * 1024 * 1024,
            'backupCount': 10
        },
    },
    'loggers': {
        '': {
            'handlers': ['logfile'],
            'level': 'ERROR'
        },
        'project': {
            'level': 'INFO',
            'propagate': True,
        },
    }
}
```

To use just get logger with *logging.getLogger()*:

```
import logging
logger = logging.getLogger('project')
```

5.2 CLI integration

Run in your shell:

```
$ muffin path.to.your.module:app_object_name --help
```

5.2.1 Write a custom command

```python
@app.manage.command
def hello(name, upper=False):
    """ Write command help text here.

    :param name:  Write your name
    :param upper: Use uppercase

    """
    greetings = 'Hello %s!' % name
    if upper:
        greetings = greetings.upper()
    print(greetings)
```

```
$ muffin example hello --help

    Write command help text here.

    positional arguments:
    name        Write your name

    optional arguments:
    -h, --help  show this help message and exit
    --upper     Enable use uppercase
    --no-upper  Disable use uppercase

$ muffin example hello mike --upper

    HELLO MIKE!
```

CHAPTER 6

Testing

6.1 Setup tests

Set module path to your Muffin Application in pytest configuration file or use command line option `--muffin-app`.

Example:

```
$ py.test -xs --muffin-app example
```

6.2 Testing application

See examples:

```
import pytest

@pytest.mark.async
def test_async_code():
    from aiohttp import request
    response = yield from request('GET', 'http://google.com')
    text = yield from response.text()
    assert 'html' in text

def test_app(app):
    """ Get your app in your tests as fixture. """
    assert app.name == 'my app name'
    assert app.cfg.MYOPTION == 'develop'

def test_view(client):
    """ Make HTTP request to your application. """
    response = client.get('/my-handler')
    assert 'mydata' in response.text
```

CHAPTER 7

Deployment

Use `muffin` command. By example:

```
$ muffin example run --workers=4
```

See `muffin {APP} run --help` for more info.

CHAPTER 8

Bug tracker

If you have any suggestions, bug reports or annoyances please report them to the issue tracker at https://github.com/klen/muffin/issues

CHAPTER 9

Contributing

Development of The Muffin happens at: https://github.com/klen/muffin

CHAPTER 10

Contributors

- **‘Kirill Klenov <https://github.com/klen‘_**
- Diego Garcia

CHAPTER 11

License

Licensed under a MIT license (See LICENSE)