gunstar Documentation

Release 0.2.2

Allisson Azevedo

Sep 27, 2017

Contents

Gunstar is a WSGI micro web framework.

Github Repo.

CHAPTER 1

Features

- Based on WebOb.
- Simple url routing.
- Jinja2 templates.
- Session interface with signed cookies.
- Signals with blinker.
- Unit testing support.
- Supports Python 2.6, 2.7, 3.3 and PyPy.

CHAPTER 2

Example

Hello World App:

```
# -*- coding: utf-8 -*-
from gunstar.app import Application
from gunstar.http import RequestHandler

class IndexHandler(RequestHandler):
    def get(self):
        self.response.write('Hello World')


routes = (
    ('/', IndexHandler, 'index_named_url'),
)


app = Application(routes=routes)


if __name__ == '__main__':
    from wsgiref.simple_server import make_server
    server = make_server('127.0.0.1', 8080, app)
    server.serve_forever()
```

CHAPTER 3

Contents

Installation

Install the latest stable release via PyPI:

```
pip install gunstar
```

Gunstar runs with Python 2.6, 2.7, 3.3 and PyPy .

Quickstart

Let's start, create a new project structure for myapp

```
mkdir myapp
cd myapp
mkdir static # for static files
mkdir templates # for jinja2 templates
touch app.py # main app
touch handlers.py # handlers classes
touch tests.py # for testing the app
```

Now, edit the content of app.py file

```
# -*- coding: utf-8 -*-
from gunstar.app import Application


myapp = Application()


if __name__ == '__main__':
    from wsgiref.simple_server import make_server
```

```
    server = make_server('127.0.0.1', 8080, myapp)
    server.serve_forever()
```

Running a development server

```
python app.py
```

Go to your browser and visit http://127.0.0.1:8080, you see a 404 page.

It's time to create your first request handler class, go to handlers.py and edit

```
# -*- coding: utf-8 -*-
from gunstar.http import RequestHandler


class IndexHandler(RequestHandler):

    def get(self):
        self.response.write('Index Handler')
```

Go back to app.py and create routes tuple

```
# -*- coding: utf-8 -*-
from gunstar.app import Application


routes = (
    ('/', 'handlers.IndexHandler', 'index'),
)


myapp = Application(routes=routes)


if __name__ == '__main__':
    from wsgiref.simple_server import make_server
    server = make_server('127.0.0.1', 8080, myapp)
    server.serve_forever()
```

Restart your development server and visit http://127.0.0.1:8080.

The tuple maps a url('/') to a handler('handlers.IndexHandler') and have a name ('index').

Congratulations, your first app is working now!

Working with templates

We need to set up a TEMPLATE_PATH variable in config

```
# -*- coding: utf-8 -*-
from gunstar.app import Application
import os


PROJECT_PATH = os.path.dirname(os.path.abspath(__file__))


class ConfigSettings(object):
```

```
    TEMPLATE_PATH = os.path.join(PROJECT_PATH, 'templates')


routes = (
    ('/', 'handlers.IndexHandler', 'index'),
)


myapp = Application(routes=routes, config=ConfigSettings)


if __name__ == '__main__':
    from wsgiref.simple_server import make_server
    server = make_server('127.0.0.1', 8080, myapp)
    server.serve_forever()
```

Create file templates/index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>MyApp - Index</title>
    <link rel="stylesheet" href="//netdna.bootstrapcdn.com/bootstrap/3.0.0/css/
↪bootstrap.min.css">
  </head>

  <body>

    <div class="container">
      <h1>Hello Stranger!</h1>
    </div>

  </body>
</html>
```

Edit handlers.py to use render_template

```
# -*- coding: utf-8 -*-
from gunstar.http import RequestHandler


class IndexHandler(RequestHandler):

    def get(self):
        self.render_template('index.html')
```

Restart your development server and visit http://127.0.0.1:8080.

Serving static files

Set STATIC_PATH and STATIC_ROOT in config

```
class ConfigSettings(object):
```

```
    TEMPLATE_PATH = os.path.join(PROJECT_PATH, 'templates')
    STATIC_ROOT = os.path.join(PROJECT_PATH, 'static')
    STATIC_PATH = '/static/'
```

Create static/index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>MyApp - Index</title>
    <link rel="stylesheet" href="//netdna.bootstrapcdn.com/bootstrap/3.0.0/css/
↪bootstrap.min.css">
  </head>

  <body>

    <div class="container">
      <h1>Index from static files.</h1>
    </div>

  </body>
</html>
```

Restart your development server and visit http://127.0.0.1:8080/static/index.html.

Working with session

The session is available in RequestHandler.session if you set SECRET_KEY in config:

```
class ConfigSettings(object):

    TEMPLATE_PATH = os.path.join(PROJECT_PATH, 'templates')
    STATIC_ROOT = os.path.join(PROJECT_PATH, 'static')
    STATIC_PATH = '/static/'
    SECRET_KEY = 'my-secret-key'
```

Edit handlers.py:

```
# -*- coding: utf-8 -*-
from gunstar.http import RequestHandler


class IndexHandler(RequestHandler):

    def get(self):
        view_count = self.session.get('view_count', 0)
        view_count += 1
        self.session.set('view_count', view_count)
        self.session.save()
        self.render_template('index.html', view_count=view_count)
```

Edit templates/index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>MyApp - Index</title>
    <link rel="stylesheet" href="//netdna.bootstrapcdn.com/bootstrap/3.0.0/css/
↪bootstrap.min.css">
  </head>

  <body>

    <div class="container">
      <h1>Hello Stranger!</h1>
      <h2>It's your {{ view_count }} visit to this page</h2>
    </div>

  </body>
</html>
```

Restart your development server and reload page to see view_count increment.

Testing

Do a favor to yourself and use nose to run the tests:

```
pip install nose
```

Gunstar has a TestCase with a nice test client. You have to override get_app method and return your app, that's it.

Edit tests.py:

```
# -*- coding: utf-8 -*-
from gunstar.testing import TestCase
from app import myapp


class AppTestCase(TestCase):

    def get_app(self):
        return myapp

    def test_index_handler(self):
        resp = self.client.get('/')
        self.assertEqual(resp.status_code, 200)
        self.assertTrue('<h1>Hello Stranger!</h1>' in resp.text)
        self.assertTrue('1 visit to this page' in resp.text)
        self.assertEqual(resp.context['view_count'], 1)

        resp = self.client.get('/')
        self.assertEqual(resp.status_code, 200)
        self.assertEqual(resp.context['view_count'], 2)

        resp = self.client.get('/')
        self.assertEqual(resp.status_code, 200)
        self.assertEqual(resp.context['view_count'], 3)
```

```
    def test_static_file(self):
        resp = self.client.get('/static/index.html')
        self.assertEqual(resp.status_code, 200)
        self.assertTrue('<h1>Index from static files.</h1>' in resp.text)
```

And run nose to call the tests:

```
nosetests
..
----------------------------------------------------------------------
Ran 2 tests in 0.166s

OK
```

Configuration

Gunstar configuration is handled by gunstar.config.Config class.

Use uppercase letters for your config keys, see the example below.

```
>>> from gunstar.config import Config
>>> class ConfigObject(object):
...     KEY1 = 'key1'
...     Key2 = 'key2'
...     key3 = 'key3'
...
>>> config = Config()
>>> config.load_from_object(ConfigObject)
>>> 'KEY1' in config
True
>>> 'Key2' in config
False
>>> 'key3' in config
False
>>>
```

How to load config

Load from object

```
from gunstar.config import Config

class ConfigObject(object):

    KEY1 = 'key1'
    key2 = 'key2'

config = Config()
config.load_from_object(ConfigObject)
```

Load from object in python file

```
# file settings.py
class Settings(object):
```

```
    KEY1 = 'key1'
    key2 = 'key2'
```

```
from gunstar.config import Config

config = Config()
config.load_from_object('settings.Settings')
```

Load from python file

```
# file settings.py

KEY1 = 'key1'
key2 = 'key2'
```

```
from gunstar.config import Config

config = Config()
config.load_from_object('settings')
```

Routing

Just define a tuple with three elements: url, handler and url name.

```
routes = (
    ('/', 'handlers.IndexHandler', 'index'),
)
```

Working with tokens

You can set a token in url definition, for example:

```
# file handlers.py
# -*- coding: utf-8 -*-
from gunstar.http import RequestHandler


class IndexHandler(RequestHandler):

    def get(self, name):
        self.render_template('index.html', name=name)
```

```
routes = (
    ('/{name}/', 'handlers.IndexHandler', 'index'),
    # convert to regex r'^/([^/]+)/$'
    # match '/allisson/'
    # match '/@allisson/'
)
```

For convinience, you can add a filter to token.

Example with int filter:

```
# file handlers.py
# -*- coding: utf-8 -*-
from gunstar.http import RequestHandler


class PostHandler(RequestHandler):

    def get(self, id):
        self.render_template('index.html', id=id)
```

```
routes = (
    ('/posts/{id:int}/', 'handlers.PostHandler', 'post_index'),
    # convert to regex r'^/posts/([\d]+)/$'
    # match '/posts/1/'
    # match '/posts/2/'
)
```

Example with string filter:

```
# file handlers.py
# -*- coding: utf-8 -*-
from gunstar.http import RequestHandler


class SearchHandler(RequestHandler):

    def get(self, query):
        self.render_template('index.html', query=query)
```

```
routes = (
    ('/search/{query:string}/', 'handlers.SearchHandler', 'search'),
    # convert to regex r'^/search/([\w]+)/$'
    # match '/search/mysearch/'
    # match '/search/my_search/'
)
```

Example with slug filter:

```
# file handlers.py
# -*- coding: utf-8 -*-
from gunstar.http import RequestHandler


class PostHandler(RequestHandler):

    def get(self, title):
        self.render_template('index.html', title=title)
```

```
routes = (
    ('/post/{title:slug}/', 'handlers.PostHandler', 'post'),
    # convert to regex r'^/post/([\w-]+)/$'
    # match '/post/my_post/'
    # match '/post/my-post/'
)
```

Example with path filter:

```
# file handlers.py
# -*- coding: utf-8 -*-
from gunstar.http import RequestHandler


class WikiHandler(RequestHandler):

    def get(self, title):
        self.render_template('index.html', wiki=wiki)
```

```
routes = (
    ('/wiki/{name:path}/', 'handlers.WikiHandler', 'wiki'),
    # convert to regex r'^/wiki/([^/].*?)/$'
    # match '/wiki/Allisson/Detail/'
    # match '/wiki/Allisson/Detail/Age/'
)
```

Example with re filter:

```
# file handlers.py
# -*- coding: utf-8 -*-
from gunstar.http import RequestHandler


class PostHandler(RequestHandler):

    def get(self, title):
        self.render_template('index.html', name=name)
```

```
routes = (
    ('/post/{name:re:([\w-@]+)}/', 'handlers.PostHandler', 'post'),
    # convert to regex r'^/post/([\w-@]+)/$'
    # match '/post/@allisson-azevedo/'
    # match '/post/@allisson_azevedo/'
)
```

Working with handler import

You can import the handler directly or inform the location

```
# file handlers.py
# -*- coding: utf-8 -*-
from gunstar.http import RequestHandler


class PostHandler(RequestHandler):

    def get(self, title):
        self.render_template('index.html')
```

```
from handlers import PostHandler

# inform location or import directly.
routes = (
    ('/post1/{title}/', 'handlers.PostHandler', 'post1'),
```

```
    ('/post2/{title}/', PostHandler, 'post2'),
)
```

Application

The heart of Gunstar, a wsgi app.

Example:

```
# file app.py
from gunstar.app import Application

myapp = Application()
```

In this case, myapp can run with any wsgi server.

Example with wsgiref (included in python):

```
# file run.py
from app import myapp

from wsgiref.simple_server import make_server
server = make_server('127.0.0.1', 8080, myapp)
server.serve_forever()
```

Example with gunicorn (pip install gunicorn):

```
gunicorn app:myapp -b 127.0.0.1:8080
```

Initialize routes

Method one: pass routes tuple directly to Application class

```
from gunstar.app import Application

routes = (
    ('/', 'handlers.IndexHandler', 'index'),
)


myapp = Application(routes=routes)
```

Method two: calling Application.add_route()

```
from gunstar.app import Application

myapp = Application()
myapp.add_route('/', 'handlers.IndexHandler', 'index')
```

Initialize config

Method one: pass the config directly to Application class

```
from gunstar.app import Application

class Settings(object):
    KEY1 = 'key1'

myapp = Application(config=Settings)
```

Method two: calling Application.load_config()

```
from gunstar.app import Application

class Settings(object):
    KEY1 = 'key1'

myapp = Application()
myapp.load_config(Settings)
```

Handlers

All handlers must be a subclass of gunstar.http.RequestHandler

```
# -*- coding: utf-8 -*-
from gunstar.http import RequestHandler


class Handler(RequestHandler):

    def get(self):
        self.response.write('respond GET method')

    def post(self):
        self.response.write('respond POST method')
```

Subclasses of RequestHandler have:

- RequestHandler.request: Instance of webob.Request
- RequestHandler.response: Instance of webob.Response
- RequestHandler.session: Instance of gunstar.session.Session

Example:

```
# -*- coding: utf-8 -*-
from gunstar.http import RequestHandler


class IndexHandler(RequestHandler):

    def get(self):
        name = self.request.GET.get('name', 'Stranger')
        self.response.write('Hello, {0}'.format(name))
```

Using templates

You need to set TEMPLATE_PATH in your config:

```
# -*- coding: utf-8 -*-
from gunstar.app import Application
from gunstar.http import RequestHandler
import os


PROJECT_PATH = os.path.dirname(os.path.abspath(__file__))


class ConfigSettings(object):

    TEMPLATE_PATH = os.path.join(PROJECT_PATH, 'templates')


class IndexHandler(RequestHandler):

    def get(self):
        self.render_template('index.html', var1='var1', var2='var2')


routes = (
    ('/', IndexHandler, 'index'),
)


app = Application(routes=routes, config=ConfigSettings)
```

You can add filters and globals overriding methods:

```
# -*- coding: utf-8 -*-
from gunstar.http import RequestHandler


# filter
def datetimeformat(value, format='%H:%M / %d-%m-%Y'):
    return value.strftime(format)


# global
def hello(name):
    return 'Hello {0}'.format(name)


class BaseHandler(RequestHandler):

    def get_template_globals(self):
        template_globals = super(BaseHandler, self).get_template_globals()
        template_globals['hello'] = hello
        return template_globals

    def get_template_filters(self):
        template_filters = super(BaseHandler, self).get_template_filters()
        template_filters['datetimeformat'] = datetimeformat
        return template_filters
```

Using abort

The method abort() is used to send a http code to client:

```
# -*- coding: utf-8 -*-
from gunstar.http import RequestHandler


class IndexHandler(RequestHandler):

    def get(self):
        self.abort(404, message='Not found page.')
        # shortcut for:
        # self.response.status_code = 404
        # self.seponse.write('Not found page.')
```

Using redirect

The method redirect() is used to redirect client to another location:

```
# -*- coding: utf-8 -*-
from gunstar.http import RequestHandler


class IndexHandler(RequestHandler):

    def get(self):
        self.redirect('http://gunstar.readthedocs.org')
        # Move to http://gunstar.readthedocs.org with http code = 302


class IndexHandler2(RequestHandler):

    def get(self):
        self.redirect('http://gunstar.readthedocs.org', permanent=True)
        # Move to http://gunstar.readthedocs.org with http code = 301


class IndexHandler3(RequestHandler):

    def get(self):
        self.redirect('http://gunstar.readthedocs.org', status_code=307)
        # Move to http://gunstar.readthedocs.org with http code = 307
```

Using reverse_route

The method reverse_route() is used to generates a url to the given route name:

```
# -*- coding: utf-8 -*-
from gunstar.app import Application
from gunstar.http import RequestHandler


class IndexHandler(RequestHandler):
```

```
    def get(self):
        post_url = self.reverse_route('post_detail', 'my-post-slug')
        self.redirect(post_url)


class PostHandler(RequestHandler):

    def get(self, post_slug):
        self.response.write('This is the post {0}'.format(post_slug))


routes = (
    ('/', IndexHandler, 'index'),
    ('/posts/{post_slug:slug}/', PostHandler, 'post_detail'),
)


app = Application(routes=routes)
```

Session

You need to set SECRET_KEY in your config to use the session:

```
# -*- coding: utf-8 -*-
from gunstar.app import Application


class ConfigSettings(object):

    SECRET_KEY = 'my-secret-key'


routes = (
    ('/', 'handlers.IndexHandler', 'index'),
)


myapp = Application(routes=routes, config=ConfigSettings)
```

If you want to create a good secret key, follow this snippet:

```
# snippet from http://flask.pocoo.org/docs/quickstart/#sessions
>>> import os
>>> os.urandom(24)
'\xfd{H\xe5<\x95\xf9\xe3\x96.5\xd1\x01O<!\xd5\xa2\xa0\x9fR"\xa1\xa8'
```

Login Example

```
# -*- coding: utf-8 -*-
from gunstar.http import RequestHandler


class LoginHandler(RequestHandler):
```

```
    def get(self):
        self.render_template('login.html')

    def post(self):
        username = self.request.POST.get('username', None)
        password = self.request.POST.get('password', None)
        user = your_code_to_get_user_by_username_and_password(username, password)
        if user:
            self.session.set('user_id', user.id)
            self.session.save()
            self.redirect('/')
        else:
            error = 'Invalid login'
            self.render_template('login.html', error=error)


class LogoutHandler(RequestHandler):

    def get(self):
        # if you want to remove all keys in the session:
        # self.session.clear()
        self.session.delete('user_id')
        self.session.save()
        self.redirect('/login/')


class IndexHandler(RequestHandler):

    def get(self):
        user_id = self.session.get('user_id', None)
        if not user_id:
            self.redirect('/login/')
        user = your_code_to_get_user_by_user_id(user_id)
        self.render_template('index.html', user=user)
```

Signals

Gunstar signal support is provided by the excellent Blinker library.

request_started_signal

This signal is sent when request started.

Example:

```
from gunstar.signals import request_started_signal

def receive_request_started_signal(app, request):
    print(app)
    print(request)

request_started_signal.connect(receive_request_started_signal)
```

request_finished_signal

This signal is sent when response is sent to the client.

Example:

```
from gunstar.signals import request_finished_signal

def receive_request_finished_signal(app, response):
    print(app)
    print(response)

request_finished_signal.connect(receive_request_finished_signal)
```

request_exception_signal

This signal is sent when an exception happens during request processing.

Example:

```
from gunstar.signals import request_exception_signal

def receive_request_exception_signal(app, request, exc_info):
    print(app)
    print(request)
    print(exc_info)

request_exception_signal.connect(receive_request_exception_signal)
```

template_rendered_signal

This signal is sent when a template was successfully rendered.

Example:

```
from gunstar.signals import template_rendered_signal

def receive_template_rendered_signal(app, handler, template, context):
    print(app)
    print(handler)
    print(template)
    print(context)

template_rendered_signal.connect(receive_template_rendered_signal)
```

Testing

Gunstar has a TestCase, Subclass of unittest.TestCase with additional support for testing.

You have to implement the method get_app() in your TestCase.

Example:

```
# file app.py
# -*- coding: utf-8 -*-
from gunstar.app import Application
from gunstar.http import RequestHandler
import os


PROJECT_PATH = os.path.dirname(os.path.abspath(__file__))


class ConfigSettings(object):

    TEMPLATE_PATH = os.path.join(PROJECT_PATH, 'templates')


class IndexHandler(RequestHandler):

    def get(self):
        self.render_template('index.html', title='index')


routes = (
    ('/', IndexHandler, 'index'),
)

myapp = Application(routes=routes, config=ConfigSettings)
```

```
# file tests.py
# -*- coding: utf-8 -*-
from gunstar.testing import TestCase
from app import myapp


class IndexHandlerTest(TestCase):

    def get_app(self):
        return myapp

    def test_get(self):
        # test status code
        resp = self.client.get('/')
        self.assertEqual(resp.status_code, 200)

        # resp.request_started is the request that originated the response
        self.assertEqual(resp.request_started.method, 'GET')
        self.assertEqual(resp.request_started.path_qs, '/')

        # resp.template has the string rendered by template
        # resp.context has the context passed to render_template
        self.assertEqual(resp.text, resp.template)
        self.assertEqual(resp.context['title'], 'index')

        # request with parameters
        resp = self.client.get('/', data={'name':'allisson'})
        self.assertEqual(resp.status_code, 200)
        self.assertEqual(resp.request_started.path_qs, '/?name=allisson')

        # request with headers
```

```
        resp = self.client.get('/', headers={'NAME':'allisson'})
        self.assertEqual(resp.status_code, 200)
        self.assertEqual(resp.request_started.headers['NAME'], 'allisson')

    def test_post(self):
        # test status code
        resp = self.client.post('/')
        self.assertEqual(resp.status_code, 200)

        # test form
        resp = self.client.post('/', data={'name': 'allisson', 'age': 30})
        self.assertEqual(resp.request_started.POST['name'], 'allisson')
        self.assertEqual(resp.request_started.POST['age'], '30')

        # request with headers
        resp = self.client.post('/', headers={'NAME':'allisson'})
        self.assertEqual(resp.status_code, 200)
        self.assertEqual(resp.request_started.headers['NAME'], 'allisson')

    def test_put_delete_options_head(self):
        resp = self.client.put('/')
        self.assertEqual(resp.status_code, 405)

        resp = self.client.delete('/')
        self.assertEqual(resp.status_code, 405)

        resp = self.client.options('/')
        self.assertEqual(resp.status_code, 405)

        resp = self.client.head('/')
        self.assertEqual(resp.status_code, 405)
```

CHAPTER 4

Indices and tables

- genindex
- modindex
- search