00A0 $\exists$ $\forall$ $\subseteq$ 2713x 27FA$\Longleftrightarrow$ $\surd$ $\sqrt[3]{}$ $\oplus$ $\otimes$

Sanic-For-Pythoneer Documentation

Release 0.1

howie6879

Mar 21, 2021

Contents:

sanic

- **Blog**: sanic-howie6879
- :
- **Source code**: Sanic-For-Pythoneer

CHAPTER 1

:

1.1

Sanic Python

Python3.4 `asyncio` Python IO 3.5 `async/await` IO 3.6 `asyncio` Pyth

1.1.1

Sanic `async/await` `Flask` `Sanic` Flask Sanic Sanic

Sanic Sanic `uvloop` `asyncio` `uvloop` Cython `asyncio`
nodejs gevent Python Go Sanic Sanic

Sanic Python

Windows uvloop Mac Linux

Python

- virtualenv
- pyenv
- anaconda

……

Python

anaconda Python3.6

```
#   python3.6
conda create --name python36 python=3.6
#          python36
source activate python36
```

python3.6 Sanic Python3.5+ Python3.6 asyncio

Sanic

Python pip python3.6

```
#  Sanic    source activate python36
pip install sanic
#    uvloop ujson
SANIC_NO_UVLOOP=true SANIC_NO_UJSON=true pip install sanic
```

python3.6 Sanic Sanic Python Sanic

```
#  Python
python
>>> import sanic
>>>
```

Sanic Sanic Web

1.1.2

Sanic web Sanic Hello World!

run.py :

```
#!/usr/bin/env python
from sanic import Sanic
from sanic.response import text

app = Sanic()


@app.route("/")
async def test(request):
    return text('Hello World!')


if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8000)
```

Sanic 10 Web http://0.0.0.0:8000 c ^_^

Sanic Sanic web Sanic

github examples/demo01/news.py

get_news() :

```
async def get_news(size=10):
    """
    Sanic
              aiohttp
            : pip install aiohttp
       readhub  api
                    examples/demo01/news.py
    """
    async with aiohttp.ClientSession() as client:
        async with client.get(readhub_api, params=params, headers=headers) as response:
            assert response.status == 200
            text = await response.json()
        return text
```

Sanic

Web http://0.0.0.0:8000/ 10 index() http://0.0.0.0:8000/2 10 ……

Sanic http://0.0.0.0:8000/ URL index()

Sanic URL dict key value http://0.0.0.0:8000/ index()

Sanic app.route Sanic app.route

```
@app.route("/")
async def index(request):
    """      /      """
    return text('Hello World!')
```

http://0.0.0.0:8000/ Hello World! 10 get_news()

```
@app.route("/")
async def index(request):
    # html
    html_tem = """
    <div style="width: 80%; margin-left: 10%">
        <p><a href="{href}" target="_blank">{title}</a></p>
        <p>{summary}</p>
         <p>{updated_at}</p>
    </div>
    """
    html_list = []
    #
```

(continues on next page)

(continued from previous page)

```
    all_news = await get_news()
    #       html
    for each_news in all_news:
        html_list.append(html_tem.format(
            href=each_news.get('news_info', [{}])[0].get('url', '#'),
            title=each_news.get('title'),
            summary=each_news.get('summary'),
            updated_at=each_news.get('updated_at'),
        ))

    return html('<hr>'.join(html_list))
```

```
python run news.py
```

http://0.0.0.0:8000/ Sanic

01_0

http://0.0.0.0:8000/2 URL

```
@app.route("/2")
async def page_2(request):
```

3 4 n 2 Sanic

```
@app.route("/<page:int>")
@app.route("/")
async def index(request, page=1):
    """
     / /page
```

(continues on next page)

(continued from previous page)

```
            examples/demo01/news.py
    """
```

```
python run news.py
```

```
http://0.0.0.0:8000/ http://0.0.0.0:8000/2
```

request

```
async def index(request, page=1):
```

http://0.0.0.0:8000/ request

```
<Request: GET />
```

request Request Sanic handle_request Request

URL Sanic handle_request Request URL index request request Request

- json
- token
- form
- files
- args
- raw_args
- cookies
- ip
- port
- socket
- remote_addr
- path
- url

request.py

request GET http://0.0.0.0:8000/json nums nums 10

```
@app.route('/json')
async def index_json(request):
    """

    """
```

(continues on next page)

(continued from previous page)

```
    nums = request.args.get('nums', 1)
    #
    all_news = await get_news()
    try:
        return json(random.sample(all_news, int(nums)))
    except ValueError:
        return json(all_news)
```

```
python run news.py
```

index_json nums http://0.0.0.0:8000/json?nums=2

01_1

Web Sanic sanic.response

```
from sanic.response import html, json
```

body html json Sanic

- json
- text
- raw
- html
- file
- file_stream
- stream

response.py

http://0.0.0.0:8000/html URL

```
Error: Requested URL /html not found
```

html 404

```
@app.exception(NotFound)
def ignore_404s(request, exception):
    return redirect('/')
```

URL http://0.0.0.0:8000/html http://0.0.0.0:8000/

Sanic

1.1.3

Sanic Sanic

- Sanic github https://github.com/channelcat/sanic
- http://sanic.readthedocs.io/en/latest/
- demo demo01

打开“微信 / 发现 / 搜一搜”搜索

1.2

1.2.1

demo2

```
demo02
  config
     __init__.py
     config.py
  run.py
```

run.py

```
#!/usr/bin/env python
from sanic import Sanic
from sanic.response import text

app = Sanic()


@app.route("/")
async def test(request):
    return text('Hello World!')


if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8000, debug=True)
```

debug config.py debug

config.py DEBUG=True run.py

```
#!/usr/bin/env python
from sanic import Sanic
from sanic.response import text
from config import DEBUG

app = Sanic()


@app.route("/")
async def test(request):
    return text('Hello World!')


if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8000, debug=DEBUG)
```

debug False

1.2.2

pro_config dev_config.py

demo2

```
demo02
  config
      __init__.py
      config.py
      dev_config.py
      pro_config.py
  run.py
```

config.py

```
#!/usr/bin/env python
import os


class Config():
    """
    Basic config for demo02
    """
    # Application config
    TIMEZONE = 'Asia/Shanghai'
    BASE_DIR = os.path.dirname(os.path.dirname(__file__))
```

pro_config.py dev_config.py

```
# dev_config
#!/usr/bin/env python
from .config import Config


class DevConfig(Config):
    """
    Dev config for demo02
    """

    # Application config
    DEBUG = True

# pro_config
#!/usr/bin/env python
from .config import Config


class ProConfig(Config):
    """
    Pro config for demo02
    """

    # Application config
    DEBUG = False
```

MODE gunicorn worker

__init__.py

```
#!/usr/bin/env python
import os


def load_config():
    """
    Load a config class
    """

    mode = os.environ.get('MODE', 'DEV')
    try:
        if mode == 'PRO':
            from .pro_config import ProConfig
            return ProConfig
        elif mode == 'DEV':
            from .dev_config import DevConfig
            return DevConfig
        else:
            from .dev_config import DevConfig
            return DevConfig
    except ImportError:
        from .config import Config
        return Config


CONFIG = load_config()
```

MODE DEV run.py

```
#!/usr/bin/env python
from sanic import Sanic
from sanic.response import text
from config import CONFIG

app = Sanic()
app.config.from_object(CONFIG)

@app.route("/")
async def test(request):
    return text('Hello World!')


if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8000, debug=app.config['DEBUG'])
```

```
#   MODE
export MODE=PRO
```

supervisor environment = MODE="PRO"

1.2.3

ZooKeeper

打开“微信 / 发现 / 搜一搜”搜索

1.3

Sanic

github Python :

```
pro_name
  docs            #
  src or pro_name/#
  tests           #
  README.md       #
 requirements.txt #
```

src pro_name

rss

1.3.1

-
-
-
-

demo01 run.py app.py

```
sample01
  docs
     demo.md
  src
     run.py
  tests
  .gitignore
 requirements.txt
```

rss json run.py

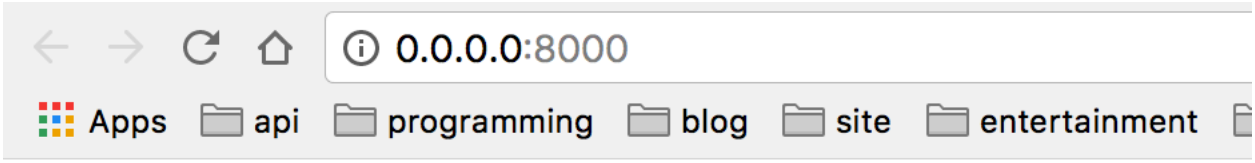
```python
#!/usr/bin/env python
from sanic import Sanic
from sanic.response import json
from feedparser import parse

app = Sanic()


@app.route("/")
async def index(request):
    url = "http://blog.howie6879.cn/atom.xml"
    feed = parse(url)
    articles = feed['entries']
    data = []
    for article in articles:
        data.append({"title": article["title_detail"]["value"], "link": article["link"]})
    return json(data)


if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8000)
```

http://0.0.0.0:8000/ json




```
[
  - {
        title: "2.Docker - 实例演示 - owllook",
        link: "http://blog.howie6879.cn/2017/08/22/27/"
    },
  - {
        title: "1.Docker - 初使用",
        link: "http://blog.howie6879.cn/2017/08/15/26/"
    },
  - {
        title: "gRPC使用初试",
        link: "http://blog.howie6879.cn/2017/08/03/25/"
    },
  - {
        title: "talonspider - 简单的爬虫框架",
        link: "http://blog.howie6879.cn/2017/06/07/24/"
    },
  - {
        title: "你的浏览器可好|Chrome插件篇",
        link: "http://blog.howie6879.cn/2017/05/11/23/"
    },
  - {
        title: "owllook -- 一个简洁的网络小说搜索引擎",
        link: "http://blog.howie6879.cn/2017/03/10/22/"
    },
  - {
        title: "sanic使用记录",
        link: "http://blog.howie6879.cn/2017/02/28/21/"
    },
```


rss-json

json

json

jinja2 template

Sanic jinja2

```
#!/usr/bin/env python
from sanic import Sanic
from sanic.response import json, text, html
from feedparser import parse
from jinja2 import Template

app = Sanic()

#
template = Template(
    """
    <!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>rss </title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body>
<article class="markdown-body">
    {% for article in articles %}
    <b><a href="{{article.link}}">{{article.title}}</a></b><br/>
    <i>{{article.published}}</i><br/>
    <hr/>
    {% endfor %}
</article>
</body>
</html>
    """
)


@app.route("/")
async def index(request):
    url = "http://blog.howie6879.cn/atom.xml"
    feed = parse(url)
    articles = feed['entries']
    data = []
    for article in articles:
        data.append({"title": article["title_detail"]["value"], "link": article["link"]})
    return json(data)


@app.route("/html")
async def rss_html(request):
    url = "http://blog.howie6879.cn/atom.xml"
    feed = parse(url)
    articles = feed['entries']
    data = []
    for article in articles:
        data.append(
            {"title": article["title_detail"]["value"], "link": article["link"],
↪"published": article["published"]})
```

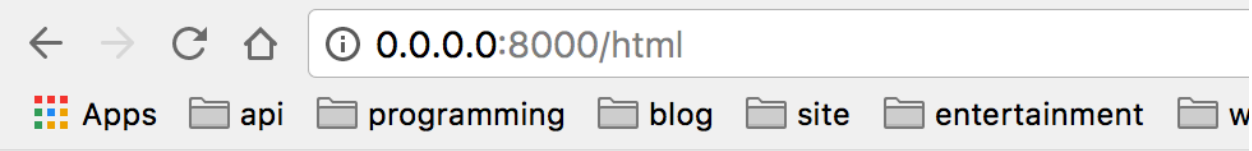
(continues on next page)

(continued from previous page)

```
    html_content = template.render(articles=data)
    return html(html_content)


if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8000)
```

sample01 http://0.0.0.0:8000/html ^_^



2.Docker - 实例演示 - owllook
2017-08-22T12:18:06.000Z

1.Docker - 初使用
2017-08-15T12:13:30.000Z

---

gRPC使用初试
2017-08-03T12:57:28.000Z

---

talonspider - 简单的爬虫框架
2017-06-07T15:56:08.000Z

你的浏览器可好!Chrome插件篇
2017-05-11T15:47:40.000Z

---

owllook -- 一个简洁的网络小说搜索引擎
2017-03-10T11:09:50.000Z

---

sanic使用记录
2017-02-28T08:33:44.000Z

--1.vscode搭建haskell环境
2017-02-11T06:58:20.000Z

---

rss-html

statics templates

1.3.2

sample02 __init__.py

```
from src.views import app
```

src

```
sample02
  docs
     demo.md
  src
     config #
     statics # css js img
     templates # Jinja2
     views #
     __init__.py
     run.py #
  tests
  requirements.txt
```

sample02

/views/rss.py sample02 :

```
enable_async = sys.version_info >= (3, 6)

app = Sanic()

# jinjia2 config
env = Environment(
    loader=PackageLoader('views.rss', '../templates'),
    autoescape=select_autoescape(['html', 'xml', 'tpl']),
    enable_async=enable_async)


async def template(tpl, **kwargs):
    template = env.get_template(tpl)
    rendered_template = await template.render_async(**kwargs)
    return html(rendered_template)


@app.route("/html")
async def rss_html(request):
    url = "http://blog.howie6879.cn/atom.xml"
    feed = parse(url)
    articles = feed['entries']
    data = []
    for article in articles:
        data.append(
```

(continues on next page)

(continued from previous page)

```
            {"title": article["title_detail"]["value"], "link": article["link"],
↪"published": article["published"]})
    return await template('rss.html', articles=articles)
```

jinja2 python 3.6+ jinja2

run.py /views/rss.py app

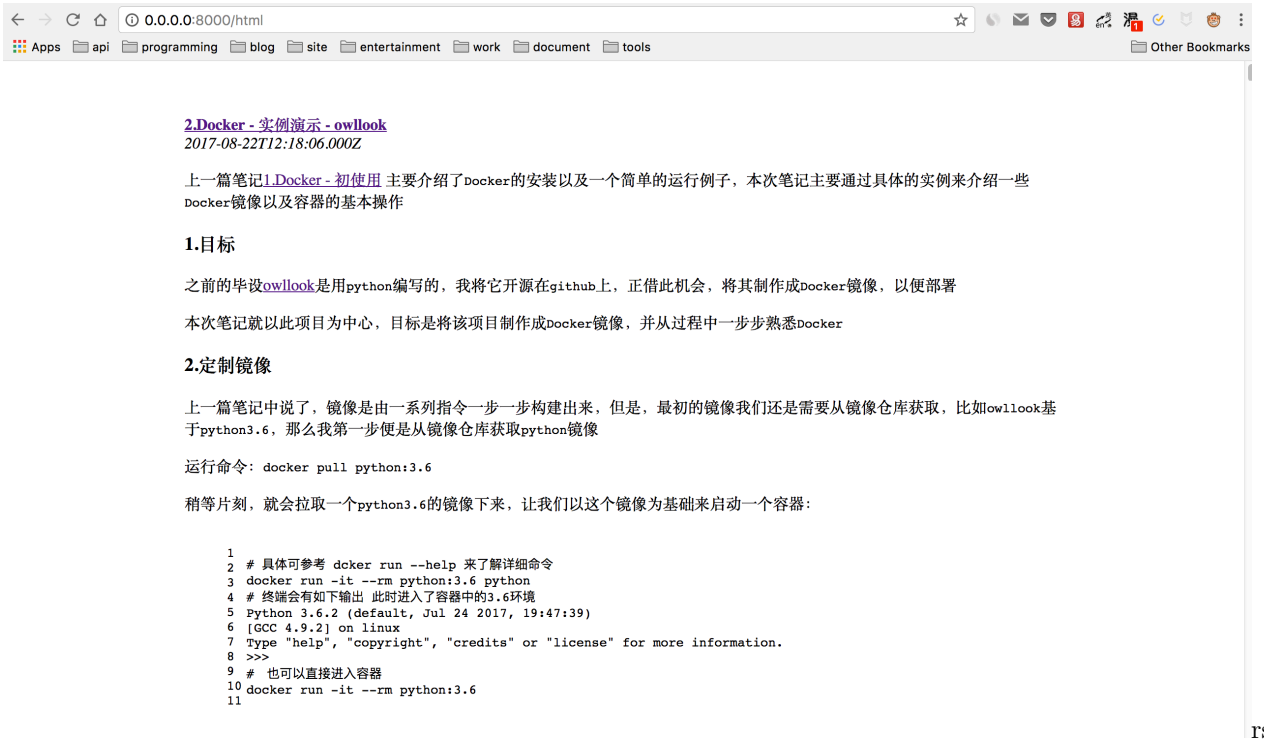
```
# !/usr/bin/env python
import sys
import os

sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
from src.views import app
from src.config import CONFIG

app.statics('/statics', CONFIG.BASE_DIR + '/statics')

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8000)
```

css sample02 http://0.0.0.0:8000/html



rss-html

1.3.3

views templates statics Blueprint

demo03

打开“微信 / 发现 / 搜一搜”搜索

1.4

rss

sanic

-
-
- Jinja2

1.4.1

python

```
#!/usr/bin/env python
from sanic import Sanic
from sanic.response import text

app = Sanic()

#     /     test
@app.route("/")
async def test(request):
    return text('Hello World!')


if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8000)
```

, test

0.0.0.0:8000/ test

(sanic0.1.2)

```
@app.route("/")
async def test(request):
    return text('Hello World!')
```

/ uri test dict 0.0.0.0:8000/ get / test

sanic.py :

```
# Decorator
def route(self, uri, methods=None):

    def response(handler):
        #   add     handler  uri
        #          namedtuple
        # Route(handler=handler, methods=methods_dict, pattern=pattern,␣
↪parameters=parameters)
        self.router.add(uri=uri, methods=methods, handler=handler)
        return handler

    return response
```

uri 103 handle_request

```
async def handle_request(self, request, response_callback):
    """
    Takes a request from the HTTP Server and returns a response object to be sent back
    The HTTP Server only expects a response object, so exception handling must be done␣
↪here
    :param request: HTTP Request object
    :param response_callback: Response function to be called with the response as the␣
↪only argument
    :return: Nothing
    """
```

handle_request request.url

Jinja2

1.4.2

sanic (html) (css)

- css js statics
- html templates
- ( ) views

http

-

- css html

- ...

url `/admin/***` `/post/***` url url

Blueprint sanic rss Blueprint

- `/json/index` json
- `/html/index` html

Blueprint demo04

```
.
  __init__.py
  config
      __init__.py
      config.py
      dev_config.py
      pro_config.py
  run.py
  statics
      rss_html # rss_html   css js
          css
              main.css
          js
              main.js
      rss_json # rss_json   css js
          css
              main.css
          js
              main.js
  templates
      rss_html # rss_html   html
          index.html
          rss.html
      rss_json # rss_json   html
           index.html
  views
       __init__.py
      rss_html.py # rss_html
      rss_json.py # rss_json
```

run.py sample01 `rss_html` `rss_json`

```
#!/usr/bin/env python
#
# rss_html.py
import sys

from sanic import Blueprint
from sanic.response import html

from src.config import CONFIG
```

(continues on next page)

(continued from previous page)

```
html_bp = Blueprint('rss_html', url_prefix='html')
html_bp.static('/statics/rss_html', CONFIG.BASE_DIR + '/statics/rss_html')

# jinjia2 config
env = Environment(
    loader=PackageLoader('views.rss_html', '../templates/rss_html'),
    autoescape=select_autoescape(['html', 'xml', 'tpl']),
    enable_async=enable_async)

@html_bp.route("/")
async def index(request):
    return await template('index.html')

#!/usr/bin/env python
#
# rss_json.py
import sys

from sanic import Blueprint
from sanic.response import html

from src.config import CONFIG


json_bp = Blueprint('rss_json', url_prefix='json')
json_bp.static('/statics/rss_json', CONFIG.BASE_DIR + '/statics/rss_json')

# jinjia2 config
env = Environment(
    loader=PackageLoader('views.rss_json', '../templates/rss_json'),
    autoescape=select_autoescape(['html', 'xml', 'tpl']),
    enable_async=enable_async)

@json_bp.route("/")
async def index(request):
    return await template('index.html')
```

- /html/ /json/
- route
- html css
-
- ...

```
cd /Sanic-For-Pythoneer/examples/demo04/sample01/src
python run.py
```

- http://0.0.0.0:8000/html/
- http://0.0.0.0:8000/json/
- http://0.0.0.0:8000/html/index
- http://0.0.0.0:8000/json/index
  ^_^

1.4.3

demo04

打开“微信 / 发现 / 搜一搜”搜索

1.5

Sanic async/await http aihttp requests Sanic

1.5.1 Mysql

mysql aiomysql sqlalchemy ORM aiomysql

```
from aiomysql.sa import create_engine
#
```

```
aio_mysql
  demo.py
  model.py
  requirements.txt
```

```
create database test_mysql;

CREATE TABLE user
(
  id        INT AUTO_INCREMENT
    PRIMARY KEY,
  user_name VARCHAR(16) NOT NULL,
  pwd       VARCHAR(32) NOT NULL,
  real_name VARCHAR(6)  NOT NULL
);
```

```
# script: model.py
import sqlalchemy as sa

metadata = sa.MetaData()

user = sa.Table(
    'user',
    metadata,
    sa.Column('id', sa.Integer, autoincrement=True, primary_key=True),
    sa.Column('user_name', sa.String(16), nullable=False),
    sa.Column('pwd', sa.String(32), nullable=False),
    sa.Column('real_name', sa.String(6), nullable=False),
)

# script: demo.py
import asyncio

from aiomysql.sa import create_engine

from model import user,metadata


async def go(loop):
    """
    aiomysql   https://github.com/aio-libs/aiomysql
    :param loop:
    :return:
    """
    engine = await create_engine(user='root', db='test_mysql',
                                 host='127.0.0.1', password='123456', loop=loop)
    async with engine.acquire() as conn:
        await conn.execute(user.insert().values(user_name='user_name01', pwd='123456',␣
↪real_name='real_name01'))
        await conn.execute('commit')

        async for row in conn.execute(user.select()):
            print(row.user_name, row.pwd)

    engine.close()
    await engine.wait_closed()
```

(continues on next page)

(continued from previous page)

```
loop = asyncio.get_event_loop()
loop.run_until_complete(go(loop))
```

python demo.py

```
user_name01 123456
```

aio_mysql SQLAlchemy ORM gino

1.5.2 MongoDB

MongoDB MongoDB Python motor

```
aio_mongo
  demo.py
  requirements.txt
```

MongoDB demo.py

```
#!/usr/bin/env python
import os

from functools import wraps

from motor.motor_asyncio import AsyncIOMotorClient

MONGODB = dict(
    MONGO_HOST=os.getenv('MONGO_HOST', ""),
    MONGO_PORT=os.getenv('MONGO_PORT', 27017),
    MONGO_USERNAME=os.getenv('MONGO_USERNAME', ""),
    MONGO_PASSWORD=os.getenv('MONGO_PASSWORD', ""),
    DATABASE='test_mongodb',
)

class MotorBaseOld:
    """
        db
    """
    _db = None
    MONGODB = MONGODB

    def client(self, db):
        # motor
        self.motor_uri = 'mongodb://{account}{host}:{port}/{database}'.format(
            account='{username}:{password}@'.format(
                username=self.MONGODB['MONGO_USERNAME'],
                password=self.MONGODB['MONGO_PASSWORD']) if self.MONGODB['MONGO_USERNAME
↪'] else '',
            host=self.MONGODB['MONGO_HOST'] if self.MONGODB['MONGO_HOST'] else 'localhost
↪',
```

(continues on next page)

(continued from previous page)

```
            port=self.MONGODB['MONGO_PORT'] if self.MONGODB['MONGO_PORT'] else 27017,
            database=db)
        return AsyncIOMotorClient(self.motor_uri)

    @property
    def db(self):
        if self._db is None:
            self._db = self.client(self.MONGODB['DATABASE'])[self.MONGODB['DATABASE']]

        return self._db
```

MongoDB db _db db

```
def singleton(cls):
    """
                https://github.com/howie6879/Sanic-For-Pythoneer/blob/master/docs/
↪part2/%E9%99%84%E5%BD%95%EF%BC%9A%E5%85%B3%E4%BA%8E%E8%A3%85%E9%A5%B0%E5%99%A8.md
    :param cls: cls
    :return: instance
    """
    _instances = {}

    @wraps(cls)
    def instance(*args, **kw):
        if cls not in _instances:
            _instances[cls] = cls(*args, **kw)
        return _instances[cls]

    return instance

@singleton
class MotorBase:
    """
     mongodb
    About motor's doc: https://github.com/mongodb/motor
    """
    _db = {}
    _collection = {}
    MONGODB = MONGODB

    def __init__(self):
        self.motor_uri = ''

    def client(self, db):
        # motor
        self.motor_uri = 'mongodb://{account}{host}:{port}/{database}'.format(
            account='{username}:{password}@'.format(
                username=self.MONGODB['MONGO_USERNAME'],
                password=self.MONGODB['MONGO_PASSWORD']) if self.MONGODB['MONGO_USERNAME
↪'] else '',
            host=self.MONGODB['MONGO_HOST'] if self.MONGODB['MONGO_HOST'] else 'localhost
↪',
```

(continues on next page)

(continued from previous page)

```
            port=self.MONGODB['MONGO_PORT'] if self.MONGODB['MONGO_PORT'] else 27017,
            database=db)
        return AsyncIOMotorClient(self.motor_uri)

    def get_db(self, db=MONGODB['DATABASE']):
        """
          db
        :param db: database name
        :return: the motor db instance
        """
        if db not in self._db:
            self._db[db] = self.client(db)[db]

        return self._db[db]

    def get_collection(self, db_name, collection):
        """

        :param db_name: database name
        :param collection: collection name
        :return: the motor collection instance
        """
        collection_key = db_name + collection
        if collection_key not in self._collection:
            self._collection[collection_key] = self.get_db(db_name)[collection]

        return self._collection[collection_key]
```

MotorBase demo aio_mongo

1.5.3 Redis

Redis asyncio_redis

```
aio_redis
  demo.py
  requirements.txt
```

redis

```
#!/usr/bin/env python
import os
import asyncio_redis

REDIS_DICT = dict(
    IS_CACHE=True,
    REDIS_ENDPOINT=os.getenv('REDIS_ENDPOINT', "localhost"),
    REDIS_PORT=os.getenv('REDIS_PORT', 6379),
    REDIS_PASSWORD=os.getenv('REDIS_PASSWORD', None),
    DB=0,
    POOLSIZE=10,
```

(continues on next page)

(continued from previous page)

```
)


class RedisSession:
    """
     redis
    """
    _pool = None

    async def get_redis_pool(self):
        if not self._pool:
            self._pool = await asyncio_redis.Pool.create(
                host=str(REDIS_DICT.get('REDIS_ENDPOINT', "localhost")), port=int(REDIS_
↪DICT.get('REDIS_PORT', 6379)),
                poolsize=int(REDIS_DICT.get('POOLSIZE', 10)), password=REDIS_DICT.get(
↪'REDIS_PASSWORD', None),
                db=REDIS_DICT.get('DB', None)
            )

        return self._pool
```

aio_redis

1.5.4

demo05

打开“微信 / 发现 / 搜一搜”搜索

1.6

Sanic pro issue

Async Python 3.5+ web server that's written to go fast

Sanic session cache reload authorized

- api json api
- gRPC
- Blueprint
- html&templates
- cache
-
- session

demo06 rss rss issue

```
src
   config
       __init__.py
       config.py
       dev_config.py
       pro_config.py
   database
       __init__.py
       redis_base
   grpc_service
       __init__.py
       grpc_asyncio_client.py
       grpc_client.py
       grpc_server.py
       hello_grpc.py
       hello_pb2.py
       hello_pb2_grpc.py
       proto
   statics
       rss_html
           css
           js
       rss_json
            css
            js
   templates
       rss_html
       rss_json
   tools
       __init__.py
       mid_decorator.py
   views
       __init__.py
       rss_api.py
       rss_html.py
       rss_json.py
   run.py
```

database gRPC grpc_service gRPC

rss_json.py json sample

1.6.1

api blog rss rss_json.py

```
#!/usr/bin/env python
from feedparser import parse
from sanic import Blueprint
from sanic.response import json

api_bp = Blueprint('rss_api', url_prefix='v1')


@api_bp.route("/get/rss/<param>")
async def get_rss_json(request, param):
    if param == 'howie6879':
        url = "http://blog.howie6879.cn/atom.xml"
        feed = parse(url)
        articles = feed['entries']
        data = []
        for article in articles:
            data.append({"title": article["title_detail"]["value"], "link": article["link
↪"]})
        return json(data)
    else:
        return json({'info': '   http://0.0.0.0:8000/v1/get/rss/howie6879'})
```

GET http://0.0.0.0:8000/v1/get/rss/howie6879 json post

```
{
    "name": "howie6879"
}
```

rss_json.py

```
@api_bp.route("/post/rss/", methods=['POST'])
async def post_rss_json(request, **kwargs):
    post_data = json_loads(str(request.body, encoding='utf-8'))
    name = post_data.get('name')
    if name == 'howie6879':
        url = "http://blog.howie6879.cn/atom.xml"
        feed = parse(url)
        articles = feed['entries']
        data = []
        for article in articles:
            data.append({"title": article["title_detail"]["value"], "link": article["link
↪"]})
        return json(data)
    else:
        return json({'info': '   '})
```

post http://0.0.0.0:8000/v1/post/rss/ name post data

mid_decorator.py

```python
def auth_params(*keys):
    """
    api
    :param keys: params
    :return:
    """

    def wrapper(func):
        @wraps(func)
        async def auth_param(request=None, rpc_data=None, *args, **kwargs):
            request_params, params = {}, []
            if isinstance(request, Request):
                # sanic request
                if request.method == 'POST':
                    try:
                        post_data = json_loads(str(request.body, encoding='utf-8'))
                    except Exception as e:
                        return response_handle(request, {'info': 'error'})
                    else:
                        request_params.update(post_data)
                        params = [key for key, value in post_data.items() if value]
                elif request.method == 'GET':
                    request_params.update(request.args)
                    params = [key for key, value in request.args.items() if value]
                else:
                    return response_handle(request, {'info': 'error'})
            else:
                pass

            if set(keys).issubset(set(params)):
                kwargs['request_params'] = request_params
                return await dec_func(func, request, *args, **kwargs)
            else:
                return response_handle(request, {'info': 'error'})

        return auth_param

    return wrapper


async def dec_func(func, request, *args, **kwargs):
    try:
        response = await func(request, *args, **kwargs)
        return response
    except Exception as e:
        return response_handle(request, {'info': 'error'})
```

```python
@api_bp.route("/post/rss/", methods=['POST'])
@auth_params('name')
async def post_rss_json(request, **kwargs):
```

name

Sanic demo

1.6.2 gRPC

http gRPC Sanic RPC grpclib src/grpc_service

1.6.3 Blueprint

Blueprint sanic

```
# main.py
from sanic import Sanic
from sanic.response import json

app = Sanic()

@app.route("/")
async def test(request):
    return json({"hello": "world"})

# http://0.0.0.0:8000/
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8000)
```

Blueprint

```
  server.py
  static
     novels
         css
            result.css
         img
            read_content.png
         js
             main.js
  template
     novels
         index.html
  views
      novels_blueprint.py
```

templates blueprint templats static blueprint blueprint-owllook Sanic

1.6.4 html&templates

web html sanic html jinja2

```
#  python3.5+
#
from sanic import Blueprint
from jinja2 import Environment, PackageLoader, select_autoescape

#  blueprint
bp = Blueprint('novels_blueprint')
bp.static('/static', './static/novels')

# jinjia2 config
env = Environment(
    loader=PackageLoader('views.novels_blueprint', '../templates/novels'),
    autoescape=select_autoescape(['html', 'xml', 'tpl']))

def template(tpl, **kwargs):
    template = env.get_template(tpl)
    return html(template.render(kwargs))

@bp.route("/")
async def index(request):
    return template('index.html', title='index')
```

python3.6

```
#  python3.5+
#
#!/usr/bin/env python
import sys

from feedparser import parse
from jinja2 import Environment, PackageLoader, select_autoescape
from sanic import Blueprint
from sanic.response import html

from src.config import CONFIG

# https://github.com/channelcat/sanic/blob/5bb640ca1706a42a012109dc3d811925d7453217/
↪examples/jinja_example/jinja_example.py
#       3.6+
enable_async = sys.version_info >= (3, 6)

html_bp = Blueprint('rss_html', url_prefix='html')
html_bp.static('/statics/rss_html', CONFIG.BASE_DIR + '/statics/rss_html')

# jinjia2 config
env = Environment(
    loader=PackageLoader('views.rss_html', '../templates/rss_html'),
    autoescape=select_autoescape(['html', 'xml', 'tpl']),
    enable_async=enable_async)


async def template(tpl, **kwargs):
    template = env.get_template(tpl)
```

(continues on next page)

(continued from previous page)

```
    rendered_template = await template.render_async(**kwargs)
    return html(rendered_template)
```

1.6.5 cache

redis aiocache aioredis

http://0.0.0.0:8000/v1/get/rss/howie6879 rss

```
@cached(ttl=1000, cache=RedisCache, key="rss", serializer=PickleSerializer(), port=6379,␣
↪namespace="main")
async def get_rss():
    print("    3 ...")
    await asyncio.sleep(3)
    url = "http://blog.howie6879.cn/atom.xml"
    feed = parse(url)
    articles = feed['entries']
    data = []
    for article in articles:
        data.append({"title": article["title_detail"]["value"], "link": article["link"]})
    return data


@api_bp.route("/get/rss/<name>")
async def get_rss_json(request, name):
    if name == 'howie6879':
        data = await get_rss()
        return json(data)
    else:
        return json({'info': '   http://0.0.0.0:8000/v1/get/rss/howie6879'})
```

3 redis json redis

1.6.6

pr

1.6.7 session

sanic sanic_session

打开“微信 / 发现 / 搜一搜”搜索

1.7

Sanic api Session...

extensions

打开“微信 / 发现 / 搜一搜”搜索

1.8

```
pro_name
   docs            #
   src or pro_name/#
   tests           #
```

(continues on next page)

(continued from previous page)

```
  README.md       #
 requirements.txt #
```

demo06 test

1.8.1

Sanic pytest Sanic Sanic pytest pytest-sanic

pytest-sanic demo06 rss api

```
tests
   setting.py
   test_rss.py
```

setting.py

```
# setting.py
def rss_data():
    return {
        "name": "howie6879"
    }
```

/v1/post/rss/ POST

```
# test_rss.py
async def test_http_rss(test_cli):
    data = setting.rss_data()
    response = await test_cli.post('/v1/post/rss/', data=ujson.dumps(data))
    resp_json = await response.json()
    assert resp_json['status'] == 1

#    pytest tests/test_rss.py
"""
============================================== test session starts␣
↪==============================================
platform darwin -- Python 3.6.0, pytest-3.2.3, py-1.4.34, pluggy-0.4.0
rootdir: /Users/howie/Documents/programming/python/git/Sanic-For-Pythoneer/examples/
↪demo06/sample, inifile:
plugins: celery-4.0.2, sanic-0.1.5
collected 2 items

tests/test_rss.py .s

======================================== 1 passed, 1 skipped in 2.13 seconds␣
↪=========================================
"""
```

clone 1 passed, 1 skipped in 2.13 seconds

HTTP gRPC gRPC DIS_GRPC_TEST = True gRPC

Sanic ip

locust tests

```
locust_rss
    __init__.py
    action.py
    locust_rss_http.py
    locustfile.py
    utils.py
setting.py
test_rss.py
```

locust_rss action.py

```
HTTP_URL = "http://0.0.0.0:8000/v1/post/rss/"
GRPC_URL = "0.0.0.0:8990"


def json_requests(client, data, url):
    func_name = inspect.stack()[1][3]
    headers = {'content-type': 'application/json'}
    return post_request(client, data=json.dumps(data), url=url, func_name=func_name,␣
↪headers=headers)


def action_rss(client):
    data = {
        "name": "howie6879"
    }
    json_requests(client, data, HTTP_URL)
```

locust_rss_http.py

```
class RssBehavior(TaskSet):
    @task(1)
    def interface_rss(self):
        action.action_rss(self.client)
```

utils.py post_request

```
def post_request(client, data, url, func_name=None, **kw):
    """
     post
    """
    func_name = func_name if func_name else inspect.stack()[1][3]
    with client.post(url, data=data, name=func_name, catch_response=True, timeout=2,␣
↪**kw) as response:
        result = response.content
```

(continues on next page)

(continued from previous page)

```
        res = to_json(result)
        if res['status'] == 1:
            response.success()
        else:
            response.failure("%s-> %s" % ('error', result))
        return result
```

locustfile.py

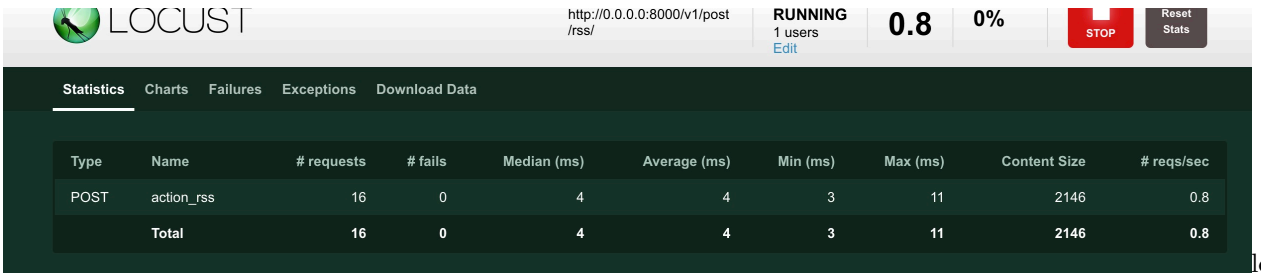
```
cd Sanic-For-Pythoneer/examples/demo06/sample/tests/locust_rss

#
locust -f locustfile.py --no-web -c 1 -n 1

# Output:
[2018-01-14 14:54:30,119] 192.168.2.100/INFO/locust.main: Shutting down (exit code 0),␣
↪bye.
 Name                                                          # reqs      # fails    ␣
↪Avg     Min     Max  |  Median   req/s
--------------------------------------------------------------------------------------------
↪--------------------------------------------------
 POST action_rss                                                 1     0(0.00%)    ␣
↪1756    1756    1756  |    1800    0.00
--------------------------------------------------------------------------------------------
↪--------------------------------------------------
 Total                                                           1     0(0.00%)      ␣
↪                                0.00

Percentage of the requests completed within given times
 Name                                                          # reqs    50%    66%    ␣
↪75%    80%    90%    95%    98%    99%   100%
--------------------------------------------------------------------------------------------
↪--------------------------------------------------
 POST action_rss                                                 1   1800   1800   ␣
↪1800   1800   1800   1800   1800   1800   1756
--------------------------------------------------------------------------------------------
↪--------------------------------------------------
```

```
locust -f locustfile.py   http://0.0.0.0:8089/
```



locust

1.8.2

Deploying

Sanic

- python -m sanic server.app –host=0.0.0.0 –port=8000 –workers=4
- gunicorn myapp:app –bind 0.0.0.0:8000 –worker-class sanic.worker.GunicornWorker

- Gunicorn + Supervisor + Caddy
- Docker

Gunicorn config/gunicorn.py

```
# gunicorn.py
bind = '127.0.0.1:8001'
backlog = 2048

workers = 2
worker_connections = 1000
timeout = 30
keepalive = 2

spew = False
daemon = False
umask = 0
```

gunicorn -c config/gunicorn.py --worker-class sanic.worker.GunicornWorker server:app

Supervisor

```
[program:demo]
command      = gunicorn -c config/gunicorn.py --worker-class sanic.worker.GunicornWorker␣
↪server:app
directory    = /your/path/
user         = root
process_name = %(program_name)s
autostart    = true
autorestart  = true
startsecs    = 3
redirect_stderr         = true
stdout_logfile_maxbytes = 500MB
stdout_logfile_backups  = 10
stdout_logfile          = ~/supervisor/demo.log
environment             = MODE="PRO"
```

( ) ” ” Caddy Caddy Go Web HTTPS Caddyfile

```
www.your.domain.com {
    proxy / 127.0.0.1:8001
    timeouts none
```

(continues on next page)

(continued from previous page)

```
    gzip
}

your.domain.com {
    redir http://www.your.domain.com
}
```

Supervisor Caddy

Docker Docker

Dockerfile

```
docker build -t demo:0.1 .
docker run -d -p 8001:8001 demo:0.1
```

daocloud

1.9

demo06

打开“微信 / 发现 / 搜一搜”搜索

1.10 :

CHAPTER 2

:

2.1 :

2.2 Sanic 0.1.2

Sanic async/await Flask uvloop libuv Sanic
Sanic Router Blueprint

- Sanic
-
-

Sanic-0.1.2

-
-

Sanic-0.1.2

```
.
  __init__.py
  blueprints.py
  config.py
  exceptions.py
  log.py
  request.py
  response.py
  router.py
  sanic.py
```

(continues on next page)

(continued from previous page)

```
  server.py
  utils.py
```

Sanic github sanic_annotation

2.2.1 simple_server.py

simple_server

```
from sanic_0_1_2.src import Sanic
from sanic_0_1_2.src.response import json

app = Sanic(__name__)


@app.route("/")
async def test(request):
    return json({"test": True})


app.run(host="0.0.0.0", port=8000)
```

sanic_annotation clone +

```
git clone https://github.com/howie6879/sanic_annotation
cd sanic_annotation/sanic_0_1_2/examples/
```

- Sanic Sanic
- json json HTTPResponse content_type
  - text content_type="text/plain; charset=utf-8"
  - html content_type="text/html; charset=utf-8"

Sanic app = Sanic(__name__) sanic.py Sanic

- route() uri Router().add()
- exception() Handler
- middleware()
- register_blueprint() blueprint register route exception middleware app.route app.exception app.exception
- handle_request() on_message_complete handle_request handle_request write_response write_response uri demo '/' write_res True})
- run() Sanic server.serve
- stop()

Sanic

```
#     /     test
@app.route("/")
async def test(request):
    return json({"test": True})
```

app.route Sanic uri, methods

url path Sanic.router Router.routes = []

Route namedtuple

```
[Route(handler=<function test at 0x10a0f6488>, methods=None, pattern=re.compile('^/$'),␣
↪parameters=[])]
```

uri '/' test '/' ,handle_request request.url test

write_response test json({"test": True})

Router dict

- add(self, uri, methods, handler) self.routes
- get(self, request) request.url

app.run(host="0.0.0.0", port=8000) Sanic run http server run serve

```
try:
    serve(
        host=host,
        port=port,
        debug=debug,
        #
        after_start=after_start,
        #
        before_stop=before_stop,
        # Sanic(__name__).handle_request()
        request_handler=self.handle_request,
        #   Config
        request_timeout=self.config.REQUEST_TIMEOUT,
        request_max_size=self.config.REQUEST_MAX_SIZE,
    )
except:
    pass
```

server.py Sanic

- serve() TCP loop.run_forever() Protocol HttpProtocol
- HttpProtocol asyncio.Protocol server.py

Sanic

demo

- sanic.py
- server.py
- router.py

- request.py
- response.py
- exceptions.py
- config.py
- log.py

__init__.py Sanic 10 demo 8 demo Sanic

2.2.2 blueprints.py

blueprints blueprints

```
from sanic_0_1_2.src import Sanic
#  Blueprint
from sanic_0_1_2.src import Blueprint
from sanic_0_1_2.src.response import json, text

app = Sanic(__name__)
blueprint = Blueprint('name', url_prefix='/my_blueprint')
blueprint2 = Blueprint('name2', url_prefix='/my_blueprint2')


@blueprint.route('/foo')
async def foo(request):
    return json({'msg': 'hi from blueprint'})


@blueprint2.route('/foo')
async def foo2(request):
    return json({'msg': 'hi from blueprint2'})


app.register_blueprint(blueprint)
app.register_blueprint(blueprint2)

app.run(host="0.0.0.0", port=8000, debug=True)
```

```
blueprint = Blueprint('name', url_prefix='/my_blueprint')
blueprint2 = Blueprint('name2', url_prefix='/my_blueprint2')
```

blueprint blueprint2 Blueprint blueprints.py:

- BlueprintSetup
  - add_route app
  - add_exception app
  - add_middleware app

- Blueprint name( ) url_prefix url
  - route self.deferred_functions app
  - middleware
  - exception
  - record self.deferred_functions
  - make_setup_state BlueprintSetup
  - register route middleware exception app make_setup_state BlueprintSetup add_*** Sanic().

route register

```
#      self.deferred_functions       handler(foo), uri, methods
@blueprint.route('/foo')
async def foo(request):
    return json({'msg': 'hi from blueprint'})


@blueprint2.route('/foo')
async def foo2(request):
    return json({'msg': 'hi from blueprint2'})

#       Sanic().register_blueprint()
app.register_blueprint(blueprint)
app.register_blueprint(blueprint2)
```

app.run(host="0.0.0.0", port=8000, debug=True)

2.2.3

Sanic middleware&exception route route

Debug Sanic Sanic Sanic

.

- sanic_annotation

2.3

2.3.1

python Python

```
#!/usr/bin/env
# -*-coding:utf-8-*-
# script: 01.py
__author__ = 'howie'
from functools import wraps
def decorator(func):
    @wraps(func)
```

(continues on next page)

(continued from previous page)

```
    def wrapper(*args, **kwargs):
        print("%s was called" % func.__name__)
        func(*args, **kwargs)
    return wrapper
@decorator
def hello(name="howie"):
    print("Hello %s!" % name)
hello()
```

```
outputs:
hello was called
Hello howie!
```

2.3.2

python

```
#!/usr/bin/env
# -*-coding:utf-8-*-
# script: 02-1.py
__author__ = 'howie'
def decorator(func):
    print("%s was called" % func.__name__)
    func()
def hello(name="howie"):
    print("Hello %s!" % name)
decorator(hello)
```

```
#!/usr/bin/env
# -*-coding:utf-8-*-
# script: 02-2.py
__author__ = 'howie'
def decorator(func):
    print("%s was called" % func.__name__)
    func()
@decorator
def hello(name="howie"):
    print("Hello %s!" % name)
hello
```

```
outputs: shell
hello was called
Hello howie!
```

02-2.py hello hello() TypeError: 'NoneType' object is not callable decorator func()

```
#!/usr/bin/env
# -*-coding:utf-8-*-
# script: 02-3.py
__author__ = 'howie'
def decorator(func):
    print("%s was called" % func.__name__)
    return func
@decorator
def hello(name="howie"):
    print("Hello %s!" % name)
hello()
```

```
#!/usr/bin/env
# -*-coding:utf-8-*-
# script: 02-4.py
__author__ = 'howie'
def decorator(func):
    print("%s was called" % func.__name__)
    func()
    return bye
def bye():
    print("bye~")
@decorator
def hello(name="howie"):
    print("Hello %s!" % name)
hello()
```

decorator :

```
#!/usr/bin/env
# -*-coding:utf-8-*-
# script: 02-5.py
__author__ = 'howie'
def decorator(func):
    def wrapper():
      print("%s was called" % func.__name__)
      func()
      print("bye~")
    return wrapper
@decorator
def hello(name="howie"):
    print("Hello %s!" % name)
hello()
```

```
outputs: shell
hello was called
Hello howie!
bye~
```

hello()==decorator(hello)()==wrapper() wrapper()

```
#!/usr/bin/env
# -*-coding:utf-8-*-
# script: 02-6.py
__author__ = 'howie'
def decorator(func):
    def wrapper():
      print("%s was called" % func.__name__)
      func()
      print("bye~")
    return wrapper
@decorator
def hello(name="howie"):
    print("Hello %s!" % name)
hello()
print(hello.__name__)
```

```
outputs: shell
hello was called
Hello howie!
bye~
wrapper
```

wrapper hello wrapper functions.wraps

```
#!/usr/bin/env
# -*-coding:utf-8-*-
# script: 02-7.py
__author__ = 'howie'
from functools import wraps
def decorator(func):
    @wraps(func)
    def wrapper():
      print("%s was called" % func.__name__)
      func()
      print("bye~")
    return wrapper
@decorator
def hello(name="howie"):
    print("Hello %s!" % name)
hello()
print(hello.__name__)
```

```
outputs: shell
hello was called
Hello howie!
bye~
hello
```

functions.wraps ~ 01.py ~

```
#!/usr/bin/env
# -*-coding:utf-8-*-
# script: 01.py
__author__ = 'howie'
from functools import wraps
def decorator(func):
    @wraps(func)
    def wrapper(*args, **kwargs):
        print("%s was called" % func.__name__)
        func(*args, **kwargs)
    return wrapper
@decorator
def hello(name="howie"):
    print("Hello %s!" % name)
hello('world')
```

2.3.3