Crwy Documentation

1.0.5

Yue.Wu

9 22, 2017

Contents

1 **3**
- 1.1 . . . 3
- 1.2 . . . 3
- 1.3 . . . 3

2 **5**
- 2.1 . . . 5
- 2.2 startproject . . . 5
- 2.3 createspider . . . 6
- 2.4 list . . . 6
- 2.5 runspider . . . 6
- 2.6 version . . . 7

3 **9**
- 3.1 basic . . . 9
- 3.2 sqlite . . . 10
- 3.3 queue . . . 11
- 3.4 redis_queue . . . 12

4 Utils **15**
- 4.1 Html . . . 15
  - 4.1.1 html_downloader . . . 15
  - 4.1.2 html_parser . . . 15
- 4.2 Sql . . . 16
  - 4.2.1 db . . . 16

5 **17**
- 5.1 Redis . . . 17
- 5.2 SSDB . . . 18
- 5.3 . . . 19

6 FAQ **21**

7 **23**

:

CHAPTER 1

- Python2.7
- Works on Linux, Mac OSX

```
pip install crwy
```

1. : https://codeload.github.com/wuyue92tree/crwy/zip/master
2. ,crwy,

```
python setup.py install
```

,

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

crwy

CHAPTER 2

: crwy,

```
Crwy - no active project found!!!

Usage:
  crwy <commands> [option] [args]

Avaliable Commands:
  list              list all spider in your project
  runspider         run a spider
  startproject      create a new project
  createspider      create a new spider
  version           show version

Use "crwy <command> -h" to see more info about a command
```

crwylist, runspider, startproject, createspider,?

startproject

```
crwy startproject spidertest
```

:

```
Project start......enjoy^.^
```

?

```
spidertest
- crwy.cfg            settings
- data                (sqlite)
```

```
|   - __init__.py
|   - __init__.pyc
- spidertest
|   - settings.py
- src
    - __init__.py
    - __init__.pyc
```

: spidertest, ()

createspider

”-h”,:

```
crwy createspider -h
```

:

```
Usage:  crwy createspider [option] [args]

Options:
  -h, --help            show this help message and exit
  -l, --list            list available spider template name
  -p PREVIEW, --preview=PREVIEW
                        preview spider template
  -t TEMPLATE, --tmpl=TEMPLATE
                        spider template
  -n NAME, --name=NAME  new spider name
```

- -l:
- -p:
- -t:
- -n:

:

```
crwy createspider -t basic -n basictest
```

src : (:crwy.cfg),

list

```
crwy list
```

runspider

”-h”,:

```
crwy createspider -h
```

```
Usage:  crwy runspider [option] [args]

Options:
  -h, --help            show this help message and exit
  -n NAME, --name=NAME  spider name
  -p PROCESS, --process=PROCESS
                        crawler by multi process
```

- -n:
- -p: -p

version

crwy

```
crwy version
```

CHAPTER 3

basic

basic

- : html_downloader
- : html_parser

:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from __future__ import print_function

from crwy.spider import Spider


class ${class_name}Spider(Spider):
    def __init__(self):
        Spider.__init__(self)
        self.spider_name = '${spider_name}'

    def crawler_${spider_name}(self):
        try:
            url = 'http://example.com'
            response = self.html_downloader.download(url)
            soups = self.html_parser.parser(response.content)
            print(url)
            print(soups)
            self.logger.info('%s[%s] --> crawler success !!!' % (
                self.spider_name, self.worker, self.func_name()))

        except Exception as e:
            self.logger.exception('%s[%s] --> %s' % (
                self.spider_name, self.worker, e))
```

```
    def run(self):
        self.crawler_${spider_name}()
```

,Spider,html_downloaderhtml_parser,UtilsHtml

sqlite

sqlitesqlite

- : html_downloader
- : html_parser
- : sqlite

:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from __future__ import print_function

from sqlalchemy import Integer, Column, String
from crwy.spider import Spider
from crwy.utils.sql.db import Database, Base


class Test(Base):
    __tablename__ = "test"
    id = Column(Integer, primary_key=True, unique=True)
    title = Column(String(20))
    url = Column(String(20))


class ${class_name}Spider(Spider):
    def __init__(self):
        Spider.__init__(self)
        self.spider_name = '${spider_name}'
        self.sql = Database('sqlite:///./data/test.db')
        self.sql.init_table()

    def crawler_${spider_name}(self):
        try:
            url = 'http://example.com'
            response = self.html_downloader.download(url)
            soups = self.html_parser.parser(response.content)
            title = soups.find('title').text
            item = Test(title=title.decode('utf-8'), url=url.decode('utf-8'))
            self.sql.session.merge(item)
            self.sql.session.commit()
            print(url)
            print(soups)
            self.logger.info('%s[%s] --> crawler success !!!' % (
                self.spider_name, self.worker))

        except Exception as e:
            self.logger.exception('%s[%s] --> %s' % (
```

```
                self.spider_name, self.worker, e))

    def run(self):
        self.crawler_${spider_name}()
```

:

1. classBase(sqlalchemydeclarative_base)table
2. Databasesqlite,init_table(), Sqlite Click
3. session.merge(),session.commit()

queue

queue,

- ,URL
- URL
- : html_downloader
- : html_parser

:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from __future__ import print_function

import sys
import Queue
from crwy.spider import Spider

queue = Queue.Queue()


class ${class_name}Spider(Spider):
    def __init__(self):
        Spider.__init__(self)
        self.spider_name = '${spider_name}'

    def crawler_${spider_name}(self):
        while True:
            try:
                if not queue.empty():
                    url = 'http://example.com/%d' % queue.get()
                    response = self.html_downloader.download(url)
                    soups = self.html_parser.parser(response.content)
                    print(url)
                    print(soups)
                    print('Length of queue : %d' % queue.qsize())
                else:
                    self.logger.info('%s[%s] --> crawler success !!!' % (
                        self.spider_name, self.worker))
                    sys.exit()
```

```
            except Exception as e:
                self.logger.exception('%s[%s] --> %s' % (
                    self.spider_name, self.worker, e))
                continue

    def run(self):
        for i in range(1, 10):
            queue.put(i)

        self.crawler_${spider_name}()
```

redis_queue

redis_queueredis,

- redis: RedisQueue,
- ,URL
- URL
- : html_downloader
- : html_parser

:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from __future__ import print_function

import sys
from crwy.spider import Spider
from crwy.utils.queue.RedisQueue import RedisQueue
from crwy.utils.filter.RedisSet import RedisSet


queue = RedisQueue('foo')
s_filter = RedisSet('foo')


class ${class_name}Spider(Spider):
    def __init__(self):
        Spider.__init__(self)
        self.spider_name = '${spider_name}'

    def crawler_${spider_name}(self):
        while True:
            try:
                if not queue.empty():
                    url = 'http://example.com/%s' % queue.get()
                    if s_filter.sadd(url) is False:
                        print('You got a crawled url. %s' % url)
                        continue
                    response = self.html_downloader.download(url)
                    soups = self.html_parser.parser(response.content)
                    print(url)
```

```
                    print(soups)
                    print('Length of queue : %s' % queue.qsize())
                else:
                    self.logger.info('%s[%s] --> crawler success !!!' % (
                        self.spider_name, self.worker))
                    sys.exit()

            except Exception as e:
                self.logger.exception('%s[%s] --> %s' % (
                    self.spider_name, self.worker, e))
                continue

    def add_queue(self):
        for i in range(100):
            queue.put(i)
        print(queue.qsize())

    def run(self):
        try:
            worker = sys.argv[4]
        except :
            print('No worker found!!!\n')
            sys.exit()

        if worker == 'crawler':
            self.crawler_${spider_name}()
        elif worker == 'add_queue':
            self.add_queue()
        elif worker == 'clean':
            queue.clean()
            s_filter.clean()
        else:
            print('Invalid worker <%s>!!!\n' % worker)
```

add_queue(),,

CHAPTER 4

Utils

Html

html_downloader

requests

2.12.0

- download(url, method='GET', timeout=60)

  url: URL
  method: GET
  timeout: (60)
  **kwargs: requests

- downloadFile(url, save_path='./data/')

  url: URL
  save_path:

requests: http://www.python-requests.org/en/master/

html_parser

BeautifulSoup4

- parser(response)

  UTF-8

- gbk_parser(response)

  GBK

- jsonp_parser(response)

json(key),dict

beautifulsoup4: https://www.crummy.com/software/BeautifulSoup/

Sql

db

sqlalchemy http://docs.sqlalchemy.org/en/latest/core/engines.html

- __init__(db_url, **kwargs)

  db_url

- init_table()

- drop_table()

sqlalchemy: http://www.sqlalchemy.org/

CHAPTER 5

Redis

redis

- __init__(name, namespace='queue', **redis_kwargs)

  name:
  namespace: (queue)
  **redis_kwargs: redis

- **qsize()**
- **empty()** True
- **put()**
- **get()**
- **get_nowait()**
- **clean()**

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# author: wuyue92tree@163.com

import redis


class RedisQueue(object):
    """Simple Queue with Redis Backend"""

    def __init__(self, name, namespace='queue', **redis_kwargs):
        """The default connection parameters are: host='localhost', port=6379, db=0"""
        self.__db = redis.Redis(**redis_kwargs)
        self.key = '%s:%s' % (namespace, name)

    def qsize(self):
```

```
        """Return the approximate size of the queue."""
        return self.__db.llen(self.key)

    def empty(self):
        """Return True if the queue is empty, False otherwise."""
        return self.qsize() == 0

    def put(self, item):
        """Put item into the queue."""
        self.__db.rpush(self.key, item)

    def get(self, block=True, timeout=None):
        """Remove and return an item from the queue.

        If optional args block is true and timeout is None (the default), block
        if necessary until an item is available."""
        if block:
            item = self.__db.blpop(self.key, timeout=timeout)
        else:
            item = self.__db.lpop(self.key)

        if item:
            item = item[1]
        return item

    def get_nowait(self):
        """Equivalent to get(False)."""
        return self.get(False)

    def clean(self):
        """Empty key"""
        return self.__db.delete(self.key)
```

SSDB

SSDB

- __init__(name, **ssdb_kwargs)

  name:
  **ssdb_kwargs: ssdb

- **qsize()**
- **empty()** True
- **put()**
- **get()**
- **clean()**

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# author: wuyue92tree@163.com

import pyssdb
```

```
class SsdbQueue(object):
    """Simple Queue with SSDB Backend"""

    def __init__(self, name, **ssdb_kwargs):
        """The default connection parameters are: host='localhost', port=8888"""
        self.__db = pyssdb.Client(**ssdb_kwargs)
        self.key = name

    def qsize(self):
        """Return the approximate size of the queue."""
        return self.__db.qsize(self.key)

    def empty(self):
        """Return True if the queue is empty, False otherwise."""
        return self.qsize() == 0

    def put(self, item):
        """Put item into the queue."""
        self.__db.qpush(self.key, item)

    def get(self):
        """Remove and return an item from the queue.

        If optional args block is true and timeout is None (the default), block
        if necessary until an item is available."""

        item = self.__db.qpop(self.key)

        return item

    def clean(self):
        """Empty key"""
        return self.__db.qclear(self.key)
```

eg: default_logger.conf

```
#logger.conf
###############################################
[loggers]
keys=root,fileLogger,rtLogger

[logger_root]
level=INFO
handlers=consoleHandler

[logger_fileLogger]
handlers=consoleHandler,fileHandler
qualname=fileLogger
propagate=0

[logger_rtLogger]
handlers=consoleHandler,rtHandler
```

```
qualname=rtLogger
propagate=0

##############################################
[handlers]
keys=consoleHandler,fileHandler,rtHandler

[handler_consoleHandler]
class=StreamHandler
level=INFO
formatter=simpleFmt
args=(sys.stderr,)

[handler_fileHandler]
class=FileHandler
level=DEBUG
formatter=defaultFmt
args=('./log/default.log', 'a')

[handler_rtHandler]
class=handlers.RotatingFileHandler
level=DEBUG
formatter=defaultFmt
args=('./log/default.log', 'a', 10*1024*1024, 5)

##############################################

[formatters]
keys=defaultFmt,simpleFmt

[formatter_defaultFmt]
format=%(asctime)s %(filename)s %(funcName)s [line:%(lineno)d] %(levelname)s
↪%(message)s
datefmt=%Y-%m-%d %H:%M:%S

[formatter_simpleFmt]
format=%(name)-12s: %(levelname)-8s %(message)s
datefmt=
```

CHAPTER 6

FAQ

.....

CHAPTER 7

- genindex
- modindex
- search