FLASK

Guida Introduttiva

A cura di : Salvatore Diprima

Luca Aloi

27/03/2018

Argomenti trattati

- Flask: Storia e Caratteristiche principali
- Flask : Come si usa
- Flask : Esempi di applicazioni

Flask-Storia

FLASK

- Flask è un framework web leggero, sviluppato da Armin Ronacher nel 2010, scritto in Python e basato sullo strumento Werkzeug WSGI(Web Server Gateway Interface) e con il motore template Jinja2. Ha licenza BSD(Berkeley Software Distribution).

- Flask è un microframework perché ha un nucleo semplice ma estensibile. Non c'è uno strato di astrazione per la base di dati o qualsiasi altra componente dove esistono già librerie di terze parti per fornire funzionalità comuni.

Flask-Caratteristiche

- Contiene server e debugger per lo sviluppo
- Supporta richieste RESTful
- Usa Jinja2 per il template
- Supporta cookie di sicurezza (sessioni lato client)
- 100% WSGI 1.0 compatibile
- Basato su Unicode
- Documentazione estensiva
- Compatibilità con Google App Engine
- Estensioni disponibili per migliorare le caratteristiche desiderate

Flask-Quickstart

Applicazione minima

```
from flask import Flask                    (1)
app = Flask(__name__)                      (2)
@app.route('/')                            (3)
def hello_world():                         (4)
    return 'Hello, World!'
```

Flask-Quickstart

(1) Importiamo la classe Flask.

(2) Creiamo un'istanza di questa classe. Il primo argomento è il nome del modulo dell'applicazione. Se usi un singolo modulo sarà **__name__** perché a seconda che sia avviato come applicazione o importato come modulo, il nome sarà diverso **('__main__'** rispetto al nome di importazione effettivo).

(3) Usiamo poi **route()** per dire a Flask quale URL triggererà la nostra funzione.

(4) Alla funzione viene assegnato un nome che viene anche utilizzato per generare la URL per quella particolare funzione e restituisce il messaggio che vogliamo visualizzare nel browser dell'utente.

Flask-Quickstart

- Salviamolo come **hello.py**
- Per eseguire l'applicazione è necessario comunicare al proprio terminale l'applicazione con cui lavorare esportando la variabile di ambiente FLASK_APP:

```
$ export FLASK_APP=hello.py
$ flask run
 * Running on http://127.0.0.1:5000/
```

- Se sei su Windows devi usare **set** invece di export.
- Ora vai su *http://127.0.0.1:5000/* e dovresti vedere il tuo Hello World.

Flask-Quickstart

Routing

- Come visto prima route () viene utilizzato per associare una funzione a un URL. Ecco alcuni esempi di base:

```
@app.route('/')
def index():
    return 'Index Page'
@app.route('/hello')
def hello():
    return 'Hello, World'
```

Flask-Quickstart

Regole delle variabili

- Per aggiungere variabili a un URL, puoi contrassegnare queste sezioni speciali come <nome_variabile>. Tale parte viene quindi passata come argomento di parole chiave alla tua funzione. Opzionalmente è possibile utilizzare un convertitore specificando una regola con <convertitore: nome_variabile>:

```
@app.route('/user/<username>')
def show_user_profile(username):
        # show the user profile for that user
        return 'User %s' % username
@app.route('/post/<int:post_id>')
def show_post(post_id):
        # show the post with the given id, the id is an integer
        return 'Post %d' % post_id
```

Flask-Quickstart

Metodi HTTP

- HTTP conosce diversi metodi per accedere agli URL. Per impostazione predefinita, una route risponde solo alle richieste GET, ma può essere modificata fornendo l'argomento dei metodi al route:

```
from flask import request
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        do_the_login()
    else:
        show_the_login_form()
```

Flask-Quickstart

Static File

- Anche le applicazioni Web dinamiche necessitano di file statici. Di solito è da dove provengono i file CSS e JavaScript. Idealmente il server web è configurato per fornirli, ma si può fare anche creando una cartella chiamata **static** nel tuo package e sarà disponibile in /static sull'applicazione.
- Per generare URL per file statici, si utilizza il nome di endpoint "static":

```
url_for('static', filename='style.css')
```

- Il file deve essere memorizzato sul filesystem come static/style.css.

Flask-Quickstart

Rendering Templates

- Generare una pagina HTML da Python non è immediato, per questo Flask configura in automatico il motore di template Jinja2. Per eseguire il rendering puoi utilizzare il metodo **render_template()**.
- Tutto quello che devi fare è fornire il nome del template e le variabili che vuoi passare al motore di template come argomenti di parole chiave:

Flask-Quickstart

```
from flask import render_template


@app.route('/hello/')
@app.route('/hello/<name>')
def hello(name=None):
    return render_template('hello.html', name=name)
```

Flask cercherà i templates nella ‘templates’ folder.

Flask-Quickstart

Ecco un esempio di template:

```
<!doctype html>
<title>Hello from Flask</title>
{% if name %}
    <h1>Hello {{ name }}!</h1>
{% else %}
    <h1>Hello, World!</h1>
{% endif %}
```

Flask-Quickstart

Accessing Request Data

- Per le applicazioni Web è fondamentale interagire con i dati che un client invia al server. In Flask questa informazione è fornita dal global request object.

Flask-Quickstart

The Request Object

- Il request object è documentato nella sezione delle API.
- Ecco una panoramica generale di alcune delle operazioni più comuni. Prima di tutto occorre importarlo dal modulo flask:

```
from flask import request
```

- Per accedere ai dati del modulo (dati trasmessi in una richiesta POST o PUT) è possibile utilizzare l'attributo del modulo:

Flask-Quickstart

```
@app.route('/login', methods=['POST', 'GET'])
def login():
        error = None
        if request.method == 'POST':
                if valid_login(request.form['username'],
                request.form['password']):
                    return log_the_user_in(request.form['username'])
        else:
                error = 'Invalid username/password'
# the code below is executed if the request method # was GET or the credentials were invalid
return render_template('login.html', error=err
```

Flask-Quickstart

Sessions

- Oltre all'oggetto request esiste anche un secondo oggetto chiamato session che consente di memorizzare informazioni specifiche per un utente da una richiesta all'altra. Questo è implementato tramite dei cookie.
- Per usare le sessioni occorre settare una a secret key.

Flask-Quickstart

```
from flask import Flask, session, redirect, url_for, escape, request
app = Flask(__name__)
@app.route('/')
def index():
        if 'username' in session:
                return 'Logged in as %s' % escape(session['username'])      return
'You are not logged in'
@app.route('/login', methods=['GET', 'POST'])
def login():
        if request.method == 'POST':
                session['username'] = request.form['username']
                return redirect(url_for('index'))
        return ''' <form method="post"> <p><input type=text name=username>
<p><input type=submit value=Login> </form> '''
@app.route('/logout')
        def logout(): # remove the username from the session if it's there
        session.pop('username', None)
        return redirect(url_for('index')) # set the secret key. keep this really secret:
app.secret_key = 'A0Zr98j/3yX R~XHH!jmN]LWX/,?RT'
```

Flask-Applicazione1

Creazione di una Web App da zero utilizzando Python Flask e MySQL

- Utilizzeremo Python, Flask e MySQL per realizzare una semplice applicazione web per una lista dei desideri, dove gli utenti potranno registrarsi, accedere e creare le proprie liste dei desideri.
- Verranno Create due pagine html (index e signup), il file signup.css e jQuery AJAX per scrivere i dati di registrazione nel metodo signUp.

Flask-Applicazione1

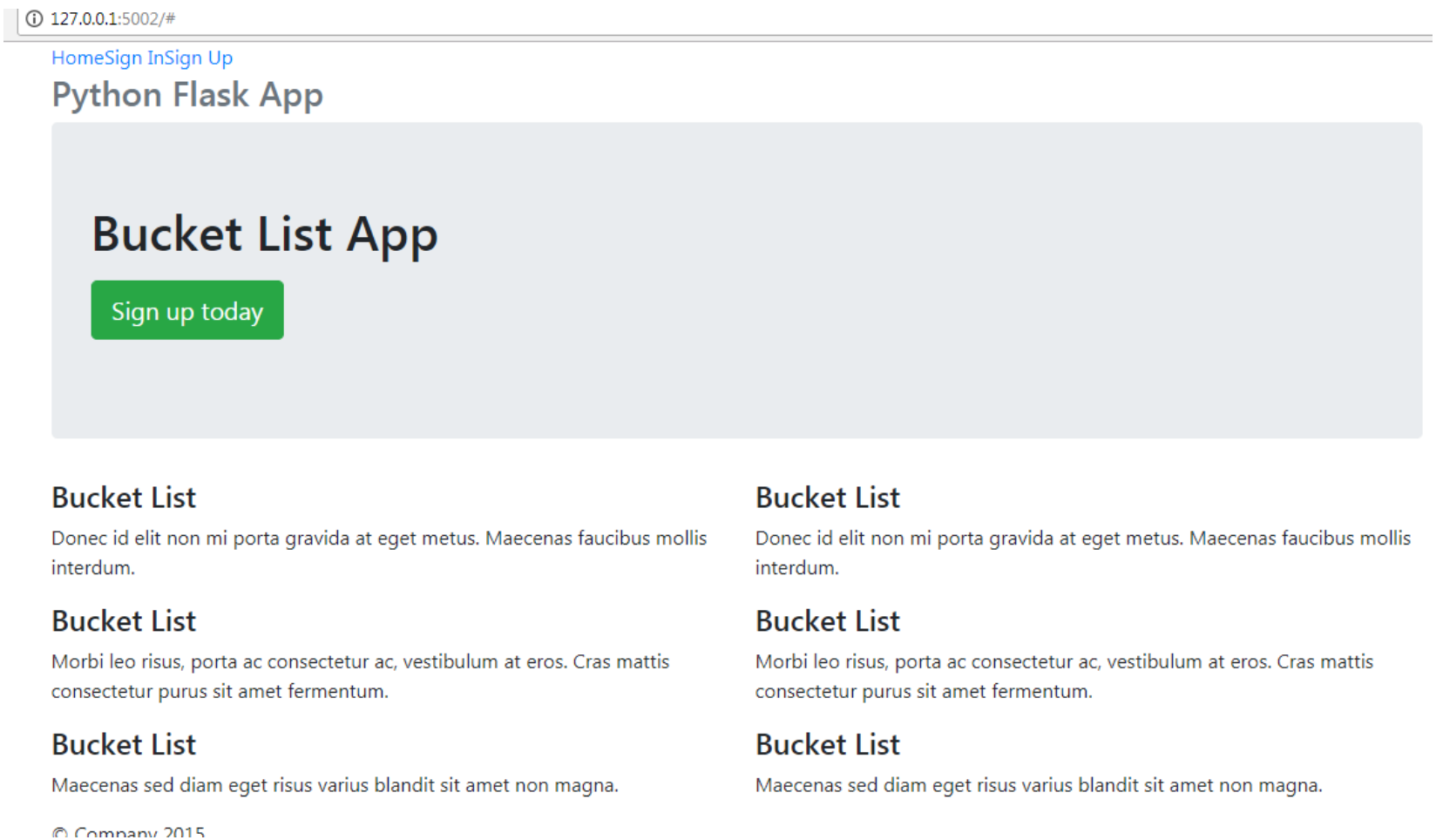
In [*]:

```
from flask import Flask, render_template, json, request
from flask.ext.mysql import MySQL
from werkzeug import generate_password_hash, check_password_hash

mysql = MySQL()
app = Flask(__name__)

# MySQL configurations
app.config['MYSQL_DATABASE_USER'] = 'root'
app.config['MYSQL_DATABASE_PASSWORD'] = ''
app.config['MYSQL_DATABASE_DB'] = 'BucketList'
app.config['MYSQL_DATABASE_HOST'] = 'localhost'
mysql.init_app(app)


@app.route('/')
def main():
    return render_template('index.html')

@app.route('/showSignUp')
def showSignUp():
    return render_template('signup.html')
```

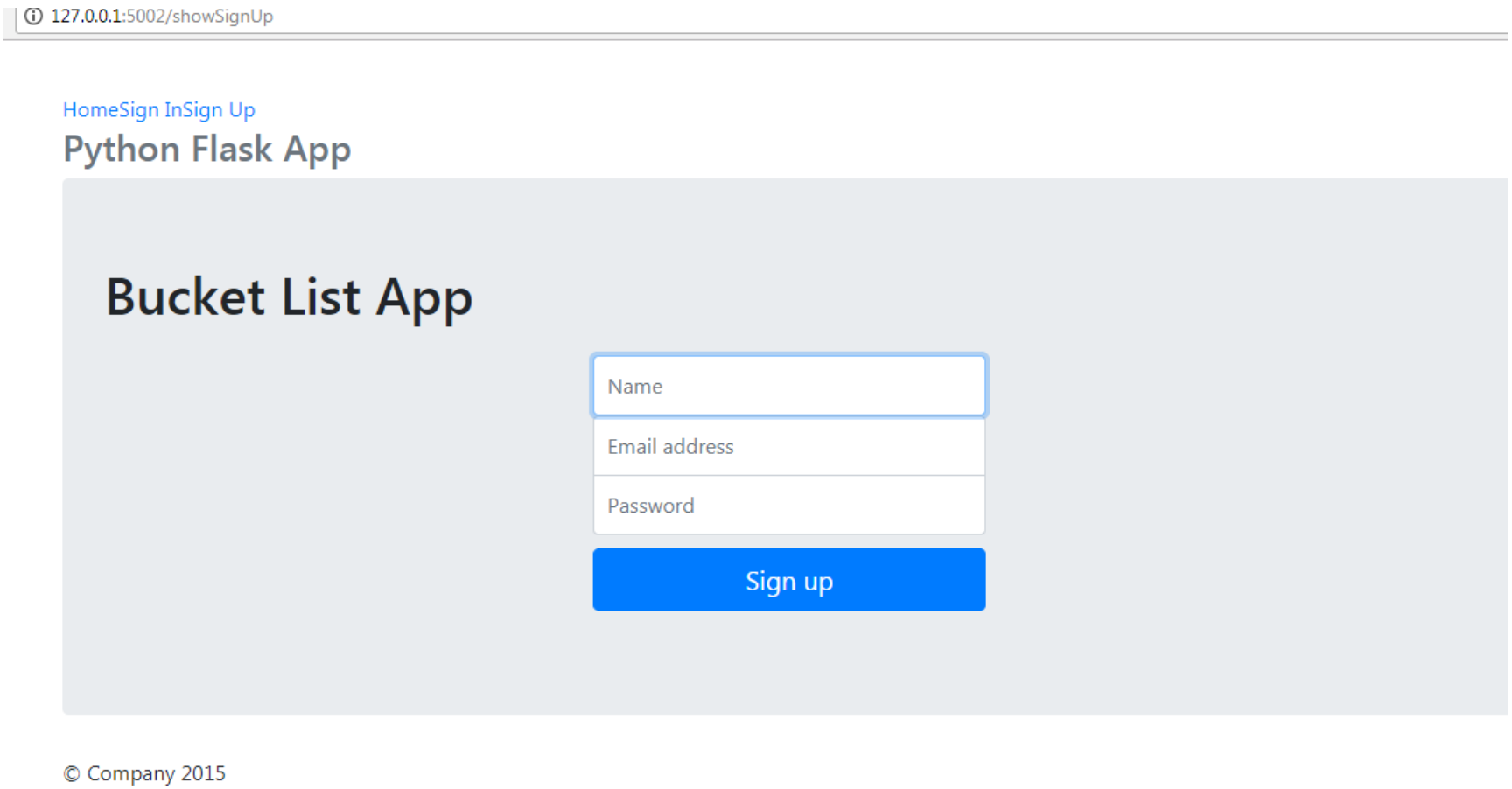
Flask-Applicazione1

```
24
25 @app.route('/signUp',methods=['POST','GET'])
26 def signUp():
27     try:
28         _name = request.form['inputName']
29         _email = request.form['inputEmail']
30         _password = request.form['inputPassword']
31
32         # validate the received values
33         if _name and _email and _password:
34
35             # All Good, let's call MySQL
36
37             conn = mysql.connect()
38             cursor = conn.cursor()
39             _hashed_password = generate_password_hash(_password)
40             cursor.callproc('sp_createUser',(_name,_email,_hashed_password))
41             data = cursor.fetchall()
42
43             if len(data) is 0:
44                 conn.commit()
45                 return json.dumps({'message':'User created successfully !'})
46             else:
47                 return json.dumps({'error':str(data[0])})
48         else:
49             return json.dumps({'html':'<span>Enter the required fields</span>'})
50
51     except Exception as e:
52         return json.dumps({'error':str(e)})
53     finally:
54         cursor.close()
55         conn.close()
56
57 if __name__ == "__main__":
58     app.run(port=5002)
```

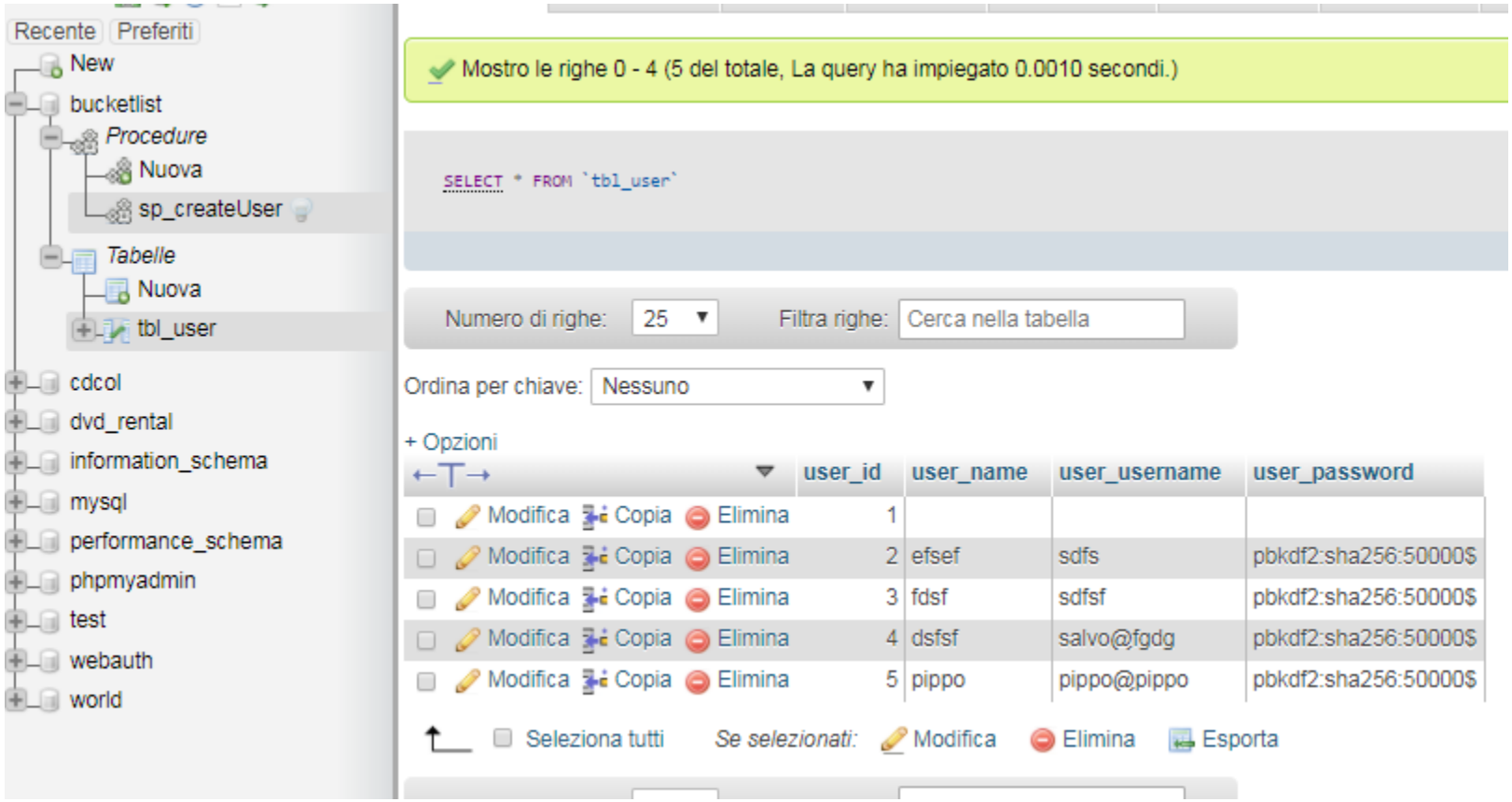
Flask-Applicazione1



Flask-Applicazione1



Flask-Applicazione1



Flask-Applicazione2

Tweet Sentiment Labeller

- Applicazione che cerca i Tweets relativi ad una parola chiave e scarica i dati etichettati con la loro polarità in formato CSV.

Flask-Applicazione2

In [*]:

```
# author = rhnvrm <hello@rohanverma.net>
import os
from flask import Flask, request, render_template, jsonify
#from twitter import TwitterClient



import re
import tweepy
from tweepy import OAuthHandler
from textblob import TextBlob



class TwitterClient(object):
    '''
    Generic Twitter Class for the App
    '''
    def __init__(self, query, retweets_only=False, with_sentiment=False):
        # keys and tokens from the Twitter Dev Console
        consumer_key = 'GCoJ7r7LFCkq2gFJJg0NgrxX1'
        consumer_secret = '8IAkXM9hIvXXfoXv1JaYwYB7LRfRJ70UaoOfIFyFUwJCAbOSTP'
        access_token = '959029213658525696-zbMpfddnsf0gdBY3rc66sk7m6IKrEW1'
        access_token_secret = 'hOLkuy4P6P8qmZ1Pg4ujhG4vEWVus9HDLR9kMGOOTKgzH'
        # Attempt authentication
        try:
            self.auth = OAuthHandler(consumer_key, consumer_secret)
            self.auth.set_access_token(access_token, access_token_secret)
            self.query = query
            self.retweets_only = retweets_only
            self.with_sentiment = with_sentiment
            self.api = tweepy.API(self.auth)
            self.tweet_count_max = 100  # To prevent Rate Limiting
        except:
            print("Error: Authentication Failed")
```

Flask-Applicazione2

```
36
37      def set_query(self, query=''):
38          self.query = query
39
40      def set_retweet_checking(self, retweets_only='false'):
41          self.retweets_only = retweets_only
42
43      def set_with_sentiment(self, with_sentiment='false'):
44          self.with_sentiment = with_sentiment
45
46      def clean_tweet(self, tweet):
47          return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)", " ", tweet).split())
48
49      def get_tweet_sentiment(self, tweet):
50          analysis = TextBlob(self.clean_tweet(tweet))
51          if analysis.sentiment.polarity > 0:
52              return 'positive'
53          elif analysis.sentiment.polarity == 0:
54              return 'neutral'
55          else:
56              return 'negative'
57
```

Flask-Applicazione2

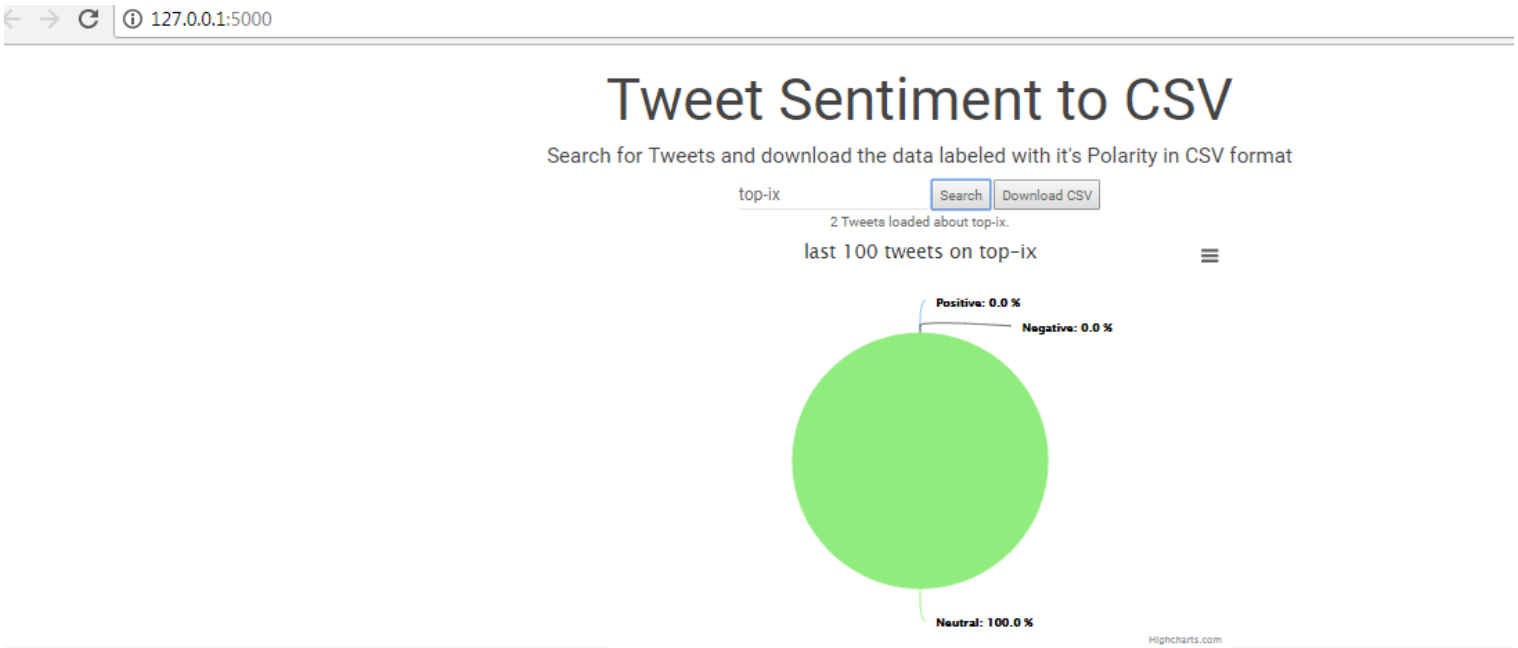
```
    def get_tweets(self):
        tweets = []

        try:
            recd_tweets = self.api.search(q=self.query,
                                          count=self.tweet_count_max)
            if not recd_tweets:
                pass
            for tweet in recd_tweets:
                parsed_tweet = {}

                parsed_tweet['text'] = tweet.text
                parsed_tweet['user'] = tweet.user.screen_name

                if self.with_sentiment == 1:
                    parsed_tweet['sentiment'] = self.get_tweet_sentiment(tweet.text)
                else:
                    parsed_tweet['sentiment'] = 'unavailable'

                if tweet.retweet_count > 0 and self.retweets_only == 1:
                    if parsed_tweet not in tweets:
                        tweets.append(parsed_tweet)
                elif not self.retweets_only:
                    if parsed_tweet not in tweets:
                        tweets.append(parsed_tweet)

            return tweets

        except tweepy.TweepError as e:
            print("Error : " + str(e))
```

Flask-Applicazione2

```
89  app = Flask(__name__)
90  # Setup the client <query string, retweets_only bool, with_sentiment bool>
91  api = TwitterClient('@ilsalvador83')
92
93
94  def strtobool(v):
95      return v.lower() in ["yes", "true", "t", "1"]
96
97
98  @app.route('/')
99  def index():
100
101     return render_template('index.html')
102
103
104 @app.route('/tweets')
105 def tweets():
106         retweets_only = request.args.get('retweets_only')
107         api.set_retweet_checking(strtobool(retweets_only.lower()))
108         with_sentiment = request.args.get('with_sentiment')
109         api.set_with_sentiment(strtobool(with_sentiment.lower()))
110         query = request.args.get('query')
111         api.set_query(query)
112
113         tweets = api.get_tweets()
114         return jsonify({'data': tweets, 'count': len(tweets)})
115
116
117 #port = int(os.environ.get('PORT', 5000))
118 #app.run(host="0.0.0.0", port=port, debug=True)
119
120 if __name__ == "__main__":
121     app.run()
```

```
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Flask-Applicazione2



Contatti e Fonti

GitHUB

- Ilsalvador83
- cryptoluca

- http://flask.pocoo.org/docs/0.12/quickstart/#quickstart

- https://github.com/jay3dec/PythonFlaskMySQLApp---Part-1
- https://github.com/rhnvrm/labeled-tweet-generator