Desenvolvimento Rápido em Linux
Trabalho Prático 4 - Banco de Dados
Webcrawler com Python

Bruno Tomé[1]

[1]Instituto Federal de Minas Gerais - Campus Formiga (IFMG)
CEP 35570-000 – Formiga – MG – Brasil

ibrunotome@gmail.com

Resumo. *Desenvolvimento de um Webcrawler com Python e banco de dados MySQL.*

1. Apresentação

O trabalho consiste no desenvolvimento de um *Webcrawler* para salvar nome, endereço, email e telefone das pessoas cadastradas no site http://www9.prefeitura.sp.gov.br/secretarias/smads/estouaqui/pessoas/todos/ utilizando a linguagem Python e banco de dados MySQL.

2. Estudo

Foi utilizado a versão 2.7.10 do Python. As bibliotecas utilizadas para a criação do *webcrawler* foram as 4 seguintes:

- *MySQLdb [1]. MySQLdb is an thread-compatible interface to the popular MySQL database server that provides the Python database API.*
- *urllib2 [2]. The urllib2 module defines functions and classes which help in opening URLs (mostly HTTP) in a complex world — basic and digest authentication, redirections, cookies and more.*
- *BeautifulSoup [3]. Is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.*
- *sys [4]. This module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter. It is always available.*

Os conhecimentos necessários para a realização deste trabalho vieram de um curso chamado “Webcrawler com Python", da *School of Net* [5], e pode ser acessado neste link: https://www.schoolofnet.com/curso-webcrawler-com-python/

3. Caso de uso

O código possui 125 linhas com os comentários. Em uma visão de alto nível, os seguintes passos são realizados:

- 1. Definir link base.
- 2. Ler html.

- 3. Utilizar o soup para ler exatamente as *tags* necessárias para obter informações úteis.
- 4. Criar uma conexão com o banco de dados.
- 5. Inserir os dados enquanto são coletados.
- 6. Repetir o laço até que todas os dados das 10 páginas sejam salvos.

Ao final, temos nome, telefone, email endereço e distrito de 100 pessoas salvos numa tabela “people".

Abaixo está listado o código para a realização deste trabalho.

```
# coding=utf-8

# Module for work with mysql database
import MySQLdb

# The urllib2 module defines functions and classes which help in opening URLs (mostly HTTP)
# in a complex world a basic and digest authentication , redirections , cookies and more.
import urllib2

# Import BeautifulSoup for parse html data
from bs4 import BeautifulSoup

# Import time to know time spent
import time

import sys

reload(sys)

# Link base to make the webcrawler
link_base = 'http :// www9.prefeitura.sp.gov.br'
link_page = link_base + '/ secretarias /smads/estouaqui/pessoas/todos/page:'

# Headers for browser
headers = {'User-Agent': 'GoogleBot'}


def run_request(link):
    """
     Iterate the next 10 pages to get the links and call the function to
    get the data child

    :param link:
    : return :
    """

    global link_page

    if link is None:
        return

    for i in xrange(1, 11):
        soup = get_html(link_page + str(i))

        if soup is None:
```

```
            return

        people_list = soup.find(attrs={'id': 'lista_pessoas'})

        for people in people_list.findAll(attrs={'class': 'pessoa'}):
            if people is None:
                continue

            people_data = people.find(attrs={'class': 'pessoa_dados'})

            if people_data is None:
                return

            link = people_data.h3.a.get('href')

            get_data_child(link_base + link)


def get_data_child(link):
    """
    Get the data of link

    :param link:
    :return:
    """

    soup = get_html(link)

    if soup is None:
        return

    data = soup.find(attrs={'id': 'conteudo'})
    name = data.h2.text
    address = ''
    district = ''
    phone = ''
    email = ''

    count = 1

    # Conection with database
    conn = connect_db()
    conn.autocommit(False)
    cur = conn.cursor()

    # String for insert into database webcrawler_python, table people, the data crawled from the link
    insert = "INSERT INTO people (name, address, district, phone, email) VALUES ('%s', '%s','%s', '%s', '%s')"

    for p in data.find(attrs={'class': 'entidade_dados'}).findAll('p'):

        if count == 1:
            address = p.text.split(':')[1][1:]
        elif count == 2:
            district = p.text.split(':')[1][1:]
```

```
            elif count == 3:
                phone = p.text.split(':')[1][1:]
            elif count == 5:
                email = p.text.split(':')[1][1:]

            count += 1

        # Bind the parameters and make the insert
        insert %= (name, address, district, phone, email)
        cur.execute(insert)
        conn.commit()

        print name


def connect_db():
    """
    Settings for connect to the database
    :return:
    """
    return MySQLdb.connect(host='localhost', user='root', passwd='', db='webcrawler_python')


def get_html(link):
    """
    Make the web request for parsing data
    :param link:
    :return:
    """
    request = urllib2.Request(link, headers=headers)
    return BeautifulSoup(urllib2.urlopen(request), 'html.parser')


if __name__ == '__main__':
    begin = time.time()
    run_request(link_base + '/secretarias/smads/estouaqui/pessoas/todos')
    end = time.time()
    print '\nTotal: ', (end - begin)
```

Listing 1. Código fonte do trabalho

4. Impressões pessoais

Este minicurso tem uma duração de 6 horas e com ele foi possível ver o quão fácil é tratar html e trabalhar com banco de dados em Python.

Para a utilização do bando de dados MySQL foi necessário apenas importar o MySQLdb, no caso de SQLite seria mais fácil ainda. Já com Django daria um pouco mais de trabalho, porém o Django oferece uma interface de administração pronta, o que é bastante útil.

5. Referências

[1] MySQLdb. Disponível em <http://mysql-python.sourceforge.net/MySQLdb.html>

[2] urllib2. Disponível em <https://docs.python.org/2/library/urllib2.html>

[3] BeautifulSoup. Disponível em <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

[4] Sys. Disponível em <https://docs.python.org/2/library/sys.html>

[5] School Of Net, Webcrawler com Python. Disponível em <https://www.schoolofnet.com/curso-webcrawler-com-python/>