**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**KHOA CÔNG NGHỆ SỐ**

****

**BÁO CÁO**

**MÔN CHUYÊN ĐỀ PHẦN MỀM**

**ĐỀ TÀI: Cào dữ liệu danh sách khách sạn ở Đà Nẵng từ agoda và đổ vào database**

|  |  |
| --- | --- |
| GIÁO VIÊN HƯỚNG DẪN : | PhD.Phạm Tuấn |
| NHÓM : | 15 |
| SINH VIÊN THỰC HIỆN : | PHẠM XUÂN TOÁN |
|  | NGUYỄN TRỌNG KHANG |
| LỚP : | 221CDPM02 |

**Đà Nẵng, tháng 05 năm 202**

I. Code Crawler dữ liệu từ Agoda

**1.Hotel.py**

class Hotel :

    def \_\_init\_\_(*self*, *ID*, *name*, *star*, *adress*, *price*, *priceSale*,

*url*, *review*, *numOfReview*,*ratingAverage*, *photo*, *status*):

*self*.ID = *ID*

*self*.name = *name*

*self*.star = *star*

*self*.adress = *adress*

*self*.price = *price*

*self*.priceSale = *priceSale*

*self*.url = *url*

*self*.review = *review*

*self*.numOfReview = *numOfReview*

*self*.ratingAverage = *ratingAverage*

*self*.photo = *photo*

*self*.status = *status*

**2.HotelDetail.py**

class HotelDetail:

    def \_\_init\_\_(*self*, *hotel\_id*, *address*, *describe*, *img*, *room\_count*, *select\_count\_sum*):

*self*.hotelId = *hotel\_id*

*self*.address = *address*

*self*.describe = *describe*

*self*.img = *img*

*self*.roomCount = *room\_count*

*self*.selectCountSum = *select\_count\_sum*

**3.HotelRoom.py**

class HotelRoom:

    def \_\_init\_\_(*self*, *hotel\_id*, *master\_room\_id*, *room\_name*, *img*, *select\_count*, *room\_detail*):

*self*.hotelId = *hotel\_id*

*self*.masterRoomId = *master\_room\_id*

*self*.roomName = *room\_name*

*self*.img = *img*

*self*.selectCount = *select\_count*

*self*.roomDetail = *room\_detail*

**4.ListHotel.py**

import time

from selenium import webdriver

from bs4 import BeautifulSoup

from pathlib import Path

from crawler.Hotel import Hotel

import pandas as pd

from selenium.webdriver.chrome.service import Service

class ListHotel:

    hotelList =  []

    def \_\_init\_\_(*self*, *url\_data*):

*self*.url\_data = *url\_data*

    def scrollPage(*self*,*driver*):

        i = 1

        scroll\_pause\_time = 1

        screen\_height = *driver*.execute\_script("return window.screen.height;")   # get the screen height of the web

        while True:

            # scroll one screen height each time

*driver*.execute\_script("window.scrollTo(0, {screen\_height}\*{i});".format(*screen\_height*=screen\_height, *i*=i))

            i += 1

            time.sleep(scroll\_pause\_time)

            # update scroll height each time after scrolled, as the scroll height can change after we scrolled the page

            scroll\_height = *driver*.execute\_script("return document.body.scrollHeight;")

            # Break the loop when the height we need to scroll to is larger than the total scroll height

            if (screen\_height) \* i > scroll\_height:

                time.sleep(2)

                break

    def checkNullData(*self*,*index*):

        if(*index* != None):

            return *index*.text

        else:

            return '0 '

    def get\_Data(*self*,*commandExecutor*):

        s = Service('E:/python/export\_files/chromedriver.exe')

        driver = webdriver.Chrome(*service*=s)

        driver.get(*self*.url\_data)

        time.sleep(2)

        while True:

*self*.scrollPage(driver)

            soup = BeautifulSoup(driver.page\_source, "html.parser")

            li = soup.find\_all("li", {"class": "PropertyCard PropertyCardItem"})

            print('count = ', len(li))

            try:

                #Thêm khach san

                for item in li:

                    if(*self*.getIndex(item) != None):

                        obj = *self*.getIndex(item)

*self*.hotelList.append(obj)

            except Exception as e:

                print('error funct getData: ',e)

            button\_next = driver.find\_elements\_by\_class\_name('pagination2\_\_next')

            if len(button\_next) > 0:

                button\_next = button\_next[0]

            else:

                time.sleep(2)

                driver.quit()

                break

            time.sleep(2)

            driver.execute\_script("arguments[0].click();", button\_next)

    def getIndex(*self*,*hotel*):

        for s in *self*.hotelList:

            if(*hotel*.get('data-hotelid') == s.ID ):

                return None

        #Lấy dữ liệu

        try:

            idh = *hotel*.get('data-hotelid')

            name = *hotel*.find("h3",{"class":"PropertyCard\_\_HotelName"}).text

            star = *hotel*.find("i",{"id":"NHAWEB-2124"}).get('title')

            adress = *hotel*.find("span",{"class":"Address\_\_Text"}).text

            price = *self*.checkNullData(*hotel*.find("div",{"class":"PropertyCardPrice"}))

            price\_sale = *self*.checkNullData(*hotel*.find("span",{"class":"PropertyCardPrice\_\_Value"}))

            url =  *hotel*.find("a", {"class":"PropertyCard\_\_Link"})['href']

            review = *self*.checkNullData(*hotel*.find("span",{"class":"Spanstyled\_\_SpanStyled-sc-16tp9kb-0 kkSkZk kite-js-Span Box-sc-kv6pi1-0 eRxXoo"}))

            num\_of\_review = *self*.checkNullData(*hotel*.find("span",{"class":"Spanstyled\_\_SpanStyled-sc-16tp9kb-0 jYmZbG kite-js-Span Box-sc-kv6pi1-0 jjmSNA"}))

            rating\_average = *self*.checkNullData(*hotel*.find("p",{"class":"Typographystyled\_\_TypographyStyled-sc-j18mtu-0 Hkrzy kite-js-Typography"}))

            status = *self*.checkNullData(*hotel*.find("button",{"class":"Buttonstyled\_\_ButtonStyled-sc-5gjk6l-0 evAQLf"}))

            photo = *hotel*.find("div",{"class":"Overlay"}).find('img')['src']

            #Xử lý dữ liệu

            star = float(star[:star.find(" ")] if (star[:star.find(" ")].strip()) else 0)

            price = price[:price.find(" ")].replace('.', '')

            price\_sale = price\_sale[:price\_sale.find(" ")].replace('.', '')

            num\_of\_review = int(num\_of\_review[:num\_of\_review.find(" ")].replace('.', ''))

            rating\_average = rating\_average.replace(',', '.')

            url = 'https://www.agoda.com/' + url

            #Thêm dữ liệu

            item = Hotel(idh,name,star,adress,price,price\_sale,url,

                         rating\_average,num\_of\_review,review,photo,status)

            return item

        except Exception as e:

            print('id-hotel=',idh, ',error getIndex: ',e)

            return None

    def exportCSV(*self*,*path*):

        hotel\_id = []

        hotel\_name = []

        hotel\_star = []

        hotel\_adress = []

        hotel\_price = []

        hotel\_price\_sale = []

        hotel\_url = []

        hotel\_review = []

        hotel\_num\_of\_review = []

        hotel\_rating\_average = []

        hotel\_status = []

        hotel\_photo = []

        for i in *self*.hotelList:

            hotel\_id.append(i.ID)

            hotel\_name.append(i.name)

            hotel\_star.append(i.star)

            hotel\_adress.append(i.adress)

            hotel\_price.append(i.price)

            hotel\_price\_sale.append(i.priceSale)

            hotel\_url.append(i.url)

            hotel\_rating\_average.append(i.ratingAverage)

            hotel\_num\_of\_review.append(i.numOfReview)

            hotel\_review.append(i.review)

            hotel\_status.append(i.status)

            hotel\_photo.append(i.photo)

        d = { 'hotel\_id' : hotel\_id

             ,'hotel\_name' : hotel\_name

             ,'hotel\_start' : hotel\_star

             ,'hotel\_adress' : hotel\_adress

             ,'hotel\_price' : hotel\_price

             ,'hotel\_price\_sale' : hotel\_price\_sale

             ,'hotel\_url' : hotel\_url

             ,'hotel\_rating\_average' : hotel\_rating\_average

             ,'hotel\_review' : hotel\_review

             ,'hotel\_num\_of\_review' : hotel\_num\_of\_review

             ,'hotel\_active' : hotel\_status

             ,'hotel\_image' : hotel\_photo}

        # Ghi vào data

        dFrame = pd.DataFrame(d)

        # lọc các dòng trùng nhau

        dFrame = dFrame.drop\_duplicates(*subset*=None, *keep*='first', *inplace*=False, *ignore\_index*=False)

        #Ghi vào CSV

        filepath = Path(*path* +'/hotel\_agoda.csv')

        filepath.parent.mkdir(*parents*=True, *exist\_ok*=True)

        dFrame.to\_csv(filepath,*encoding*='utf-8-sig',*index*=False)

**5. ListHotelDetail.py**

import time

from selenium import webdriver

from bs4 import BeautifulSoup

from crawler.HotelRoom import HotelRoom

from crawler.HotelDetail import HotelDetail

import os

import pandas as pd

from selenium.webdriver.chrome.service import Service

class ListHotelDetail:

    def \_\_init\_\_(*self*, *url\_data*):

*self*.url\_data = *url\_data*

    def scrollPage(*self*,*driver*):

        i = 5

        scroll\_pause\_time = 1

        screen\_height = *driver*.execute\_script("return window.screen.height;")   # get the screen height of the web

        while True:

            # scroll one screen height each time

*driver*.execute\_script("window.scrollTo(0, {screen\_height}\*{i});".format(*screen\_height*=screen\_height, *i*=i))

            i += 1

            time.sleep(scroll\_pause\_time)

            button\_add = *driver*.find\_elements\_by\_class\_name('MasterRoom-showMoreLessButton')

            for s in button\_add:

                if(s.get\_attribute('data-element-name') == 'room-grid-show-more'):

*driver*.execute\_script("arguments[0].click();",s)

            # update scroll height each time after scrolled, as the scroll height can change after we scrolled the page

            scroll\_height = *driver*.execute\_script("return document.body.scrollHeight;")

            # Break the loop when the height we need to scroll to is larger than the total scroll height

            if(screen\_height) \* i > scroll\_height:

                time.sleep(2)

                break

    def checkNullData(*self*,*index*):

        if(*index* != None):

            return *index*.text

        else:

            return '0 '

    def get\_Data(*self*,*commandExecutor*,*detailHotel*,*listRoom*):

        s = Service('E:/python/export\_files/chromedriver.exe')

        driver = webdriver.Chrome(*service*=s)

        driver.get(*self*.url\_data)

        time.sleep(2)

        while True:

*self*.scrollPage(driver)

            soup = BeautifulSoup(driver.page\_source, "html.parser")

            idh= soup.find("div",{"class":"MapCompact"})

            hotel\_id = idh.get('data-provider-id')

            describe = soup.find("p",{"class":"Typographystyled\_\_TypographyStyled-sc-j18mtu-0 fHvoAu kite-js-Typography"}).text

            address = soup.find("span",{"class":"Spanstyled\_\_SpanStyled-sc-16tp9kb-0 gwICfd kite-js-Span HeaderCerebrum\_\_Address"}).text

            images = soup.find\_all("img",{"class":"SquareImage"})

            url\_img = ''

            for i in images:

                url\_img = url\_img +","+i['src']

            url\_img = url\_img + ","

            room = soup.find\_all("div",{"class":"MasterRoom MasterRoom--withMoreLess"})

            master\_room\_id = []

            room\_names = []

            child\_room\_price = []

            child\_room\_capacity = []

            room\_image = []

            child\_detail\_room = []

            hotel\_room = []

            select\_count = []

            for i in room:

                hotel\_room.append(hotel\_id)

                master\_room\_id.append(i.get('id'))

                id\_room = i.get('id')

                child\_room = i.find\_all("div",{"class":"ChildRoomsList-room-contents"})

                room\_name = i.find("span",{"class":"MasterRoom\_\_HotelName"}).text

                room\_names.append(room\_name)

                images = i.find\_all("img")

                url\_img = ''

                for i in images:

                    url\_img = url\_img +","+i['src']

                url\_img = url\_img + ","

                room\_image.append(url\_img)

                detail = '~'

                for y in child\_room:

                    price\_child = y.find("h1").text[:-2]

                    child\_room\_price.append(price\_child)

                    capacity = y.find("span",{"class":"Capacity-iconGroup"})

                    count = capacity.find\_all("i")

                    child\_room\_capacity.append(len(count))

                    detail = detail + "price = "+price\_child + "-capacity = " + str(len(count)) + '~'

                select\_count.append(len(child\_room))

                child\_detail\_room.append(detail)

                child = HotelRoom(hotel\_id, id\_room, room\_name, url\_img, len(child\_room), detail)

*listRoom*.append(child)

            break

        d = HotelDetail(hotel\_id, address, describe, url\_img, len(room), len(child\_room\_price))

*detailHotel*.append(d)

    def exportCSV(*self*,*path*,*detailHotel*,*listRoom*):

        check\_detail = os.path.exists(*path* + '/hotel\_detail.csv')

        check\_room = os.path.exists(*path* + '/hotel\_room.csv')

        d = *detailHotel*[0]

        d = {'hotel\_id': d.hotelId,

             'address' : d.address,

             'describe': d.describe,

             'img' : d.img,

             'room\_count' : d.roomCount,

             'select\_count\_sum': d.selectCountSum}

        df = pd.DataFrame(d,*index*=[0])

        if(check\_detail == False):

            df.to\_csv(*path* + '/hotel\_detail.csv',*mode* = 'ab',*encoding*='utf-8-sig',*index*=False)

        else:

            df.to\_csv(*path* + '/hotel\_detail.csv',*mode* = 'ab', *header*=False,*encoding*='utf-8-sig',*index*=False)

        hotel\_room = []

        master\_room\_id = []

        room\_names = []

        room\_image = []

        select\_count = []

        child\_detail = []

        for i in *listRoom*:

            hotel\_room.append(i.hotelId)

            master\_room\_id.append(i.masterRoomId)

            room\_names.append(i.roomName)

            room\_image.append(i.img)

            select\_count.append(i.selectCount)

            child\_detail.append(i.roomDetail)

        d2 = {'hotel\_id': hotel\_room,

             'master\_room\_id' : master\_room\_id,

             'room\_names': room\_names,

             'img' : room\_image,

             'select\_count' : select\_count,

             'room\_detail' : child\_detail}

        df2 = pd.DataFrame(d2)

        if(check\_room == False):

            df2.to\_csv(*path* + '/hotel\_room.csv', *mode*='ab',*encoding*='utf-8-sig',*index*=False)

        else:

            df2.to\_csv(*path* + '/hotel\_room.csv', *mode*='ab', *header*=False,*encoding*='utf-8-sig',*index*=False)

**6.Crawl.py**

from crawler.ListHotel import ListHotel

from crawler.ListHotelDetail import ListHotelDetail

from crawler.HotelDetail import HotelDetail

command\_executor = 'http://172.17.0.2:4444'

def getAllListHotel(*url*):

    obj = ListHotel(*url*,command\_executor)

    obj.get\_Data()

    obj.exportCSV('./')

    print(len(obj.hotelList))

def getDetailHotel(*url*):

    detail = []

    listRoom = []

    obj = ListHotelDetail(*url*)

    obj.get\_Data(command\_executor,detail , listRoom)

    obj.exportCSV('./',detail , listRoom)

**7.main.py**

from crawler.Crawl import \*

import pandas as pd

import numpy as np

getAllListHotel('https://www.agoda.com/vi-vn/search?city=16440')

II. Code xây dựng API

1. main.py

from fastapi import FastAPI

import pymysql

con = pymysql.connect(db='DB\_Angoda', user='root', passwd='', host='localhost', port=3306)

app=FastAPI()

# best way to make api

@app.get('/api/hotel')

async def index():

with con.cursor() as cursor:

# Read a single record

sql = "SELECT \* FROM Hotel"

cursor.execute(sql)

result = cursor.fetchall()

return {

"success": True,

"data":result

}

# search data

@app.get('/api/search\_price/{search\_price}')

async def search(price1, price2):

with con.cursor() as cursor:

# Read a single record

sql = "SELECT \* FROM Hotel where hotel\_price>= "+price1+" and hotel\_price<=" +price2

cursor.execute(sql)

result = cursor.fetchall()

return {

"success": True,

"data":result

}

@app.get('/api/search\_start/{search\_start}')

async def search(start1, start2):

with con.cursor() as cursor:

# Read a single record

sql = "SELECT \* FROM Hotel where hotel\_start>= "+start1+" and hotel\_start<=" +start2

cursor.execute(sql)

result = cursor.fetchall()

return {

"success": True,

"data":result

}