

**TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

🙜🙢🙠🙞



**ĐỒ ÁN NGÀNH**

**ĐỀ TÀI : NGHIÊN CỨU PYTHON**

* **DJANGO.**

**GVHD: Dương Hữu Thành**

**SV - MSSV:**

**Phan Nguyễn Mạnh Huy - 1451010071**

**Nguyễn Mạnh Khang – 1451010085**

***TP.HỒ CHÍ MINH***

*Tháng 10 Năm 2017*

NGHIÊN CỨU PYTHON - DJANGO



[Date]

LỜI CẢM ƠN

MỤC LỤC:

PHẦN 1: NHỮNG NGHIÊN CỨU SƠ LƯỢC VỀ PYTHON VÀ DJANGO.

Chương I: Giới thiệu về Python.

1. Giới thiệu về ngôn ngữ lập trình Python:

* Python là ngôn ngữ lập trình cấp cao;
* **Python** là một [ngôn ngữ lập trình](https://vi.wikipedia.org/wiki/Ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh) [thông dịch](https://vi.wikipedia.org/wiki/Tr%C3%ACnh_th%C3%B4ng_d%E1%BB%8Bch) do [Guido van Rossum](https://vi.wikipedia.org/w/index.php?title=Guido_van_Rossum&action=edit&redlink=1) tạo ra năm [1990](https://vi.wikipedia.org/wiki/1990). Python hoàn toàn [tạo kiểu động](https://vi.wikipedia.org/w/index.php?title=T%E1%BA%A1o_ki%E1%BB%83u_%C4%91%E1%BB%99ng&action=edit&redlink=1) và dùng cơ chế [cấp phát bộ nhớ tự động](https://vi.wikipedia.org/wiki/Qu%E1%BA%A3n_l%C3%BD_b%E1%BB%99_nh%E1%BB%9B).
* Python mới nhất hiện nay là **Python 3**, còn gọi là **Python 3000** hoặc **Py3K**: Dòng 3.x sẽ không hoàn toàn tương thích với dòng 2.x, tuy vậy có công cụ hỗ trợ chuyển đổi từ các phiên bản 2.x sang 3.x. Nguyên tắc chủ đạo để phát triển Python 3.x là "bỏ cách làm việc cũ nhằm hạn chế trùng lặp về mặt chức năng của Python".
* Python là ngôn ngữ thông dịch;
* Python được thiết kế để trở thành một ngôn ngữ dễ học, mã nguồn dễ đọc, bố cục trực quan, dễ hiểu.

1. Cú pháp trong Python:
   1. Từ khóa:
   * Python tăng cường sử dụng từ khóa tiếng Anh, hạn chế các ký hiệu và cấu trúc cú pháp so với các ngôn ngữ khác.
   * Python là một ngôn ngữ phân biệt kiểu chữ HOA, chữ thường.
   * Như C/C++, các từ khóa của Python đều ở dạng chữ thường:

Ví dụ: and, for, assert, if, else, print, while, try catch except, def,....

* 1. Khối lệnh:

Trong các ngôn ngữ khác như Java hay C/C++, các khối lệnh thường được đặt trong cặp dấu {}. Tuy nhiên, Python lại có một cách riêng biệt để tạo khối lệnh, đó là thụt đầu dòng; các lệnh con thuộc một câu hoặc khối lệnh cha sẽ được thụt đầu dòng vào sâu hơn khối lệnh cha chứa nó.

Ví dụ: Một khối lệnh trong C++ được biểu diễn như sau:

*#include <math.h>*

2 //...

3 delta = b \* b – 4 \* a \* c;

4 **if** (delta > 0)

5 {

6 // Khối lệnh mới bắt đầu từ ký tự { đến }

7 x1 = (- b + sqrt(delta)) / (2 \* a);

8 x2 = (- b - sqrt(delta)) / (2 \* a);

9 printf("Phuong trinh co hai nghiem phan biet:**\n**");

10 printf("x1 = **%f**; x2 = **%f**", x1, x2);

11 }

// KỂ CẢ TRONG JAVA VÀ C/C++, CÁC KHỐI LỆNH ĐỀU ĐƯỢC CHỨA TRONG CẶP DẤU {}

Đoạn mã trên được viết lại bằng python như sau:

**import** **math**

2 *#...*

3 delta = b \* b – 4 \* a \* c

4 **if** delta > 0:

5 *# Khối lệnh mới, thụt vào đầu dòng, không cần tới {}*

6 x1 = (- b + math.sqrt(delta)) / (2 \* a)

7 x2 = (- b – math.sqrt(delta)) / (2 \* a)

8 **print** "Phuong trinh co hai nghiem phan biet:"

9 **print** "x1 = ", x1, "; ", "x2 = ", x2

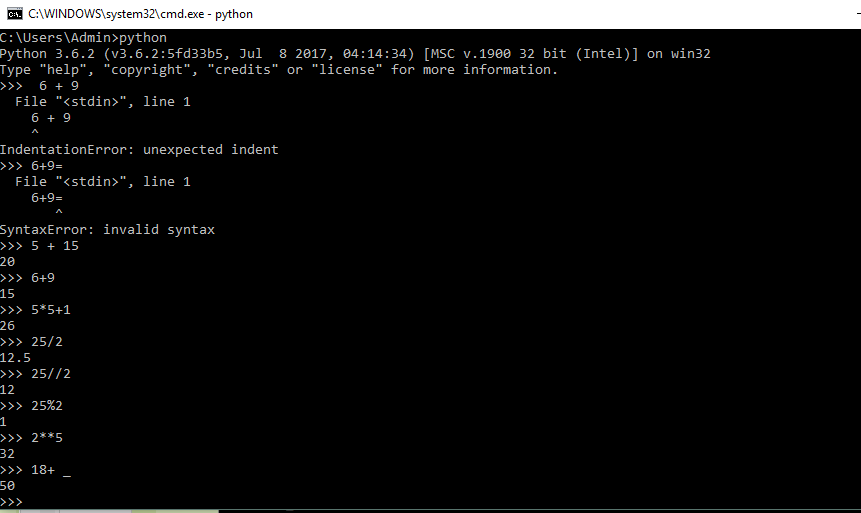
* Ta có thể dùng Tab hoặc Spacebar để thụt các câu lệnh vào.
* Để biểu thị một khối lệnh rỗng, Python có từ khóa **pass**
  1. Định danh:
* Python là ngôn ngữ phân biệt hoa, thường (case sensitive), tức là hai định danh **my\_var** và **my\_VAR** là khác nhau.
* Một số quy ước trong đặt tên định danh
  + Tên định danh trong python phải bắt đầu bằng các ký tự a -> z, hoặc A -> Z, hoặc các ký tự gạch chân \_
  + Ký tự số không được bắt đầu cho tên định danh
  + Không sử dụng ký tự đặc biệt trong tên định danh
  + Các từ khóa không được dùng làm tên định danh
  + Tên class phải bắt đầu bằng ký tự hoa, còn tên các định danh khác (biến, hàm, đối tượng, module) bắt đầu bằng ký tự thường
  1. Các cú pháp khác:
* Viết lệnh trên nhiều dòng
  + - Các lệnh trong python kết thúc bằng newline. Tuy nhiên, ta có thể sử dụng \ thể hiện sự nối tiếp để viết lệnh trên nhiều dòng.
    - Các lệnh bên trong dấu [], {}, () thì không cần sử dụng \ khi viết trên nhiều dòng
* Ta cũng có thể viết nhiều lệnh trung cùng dòng và sử dụng dấu chấm phẩy (;) để cách các lệnh
* Comment trong python sử dụng # comment trên một dòng và trong cặp dấu nháy ba """ """ để viết comment trên nhiều dòng.

1. Hệ thống kiểu dữ liệu:

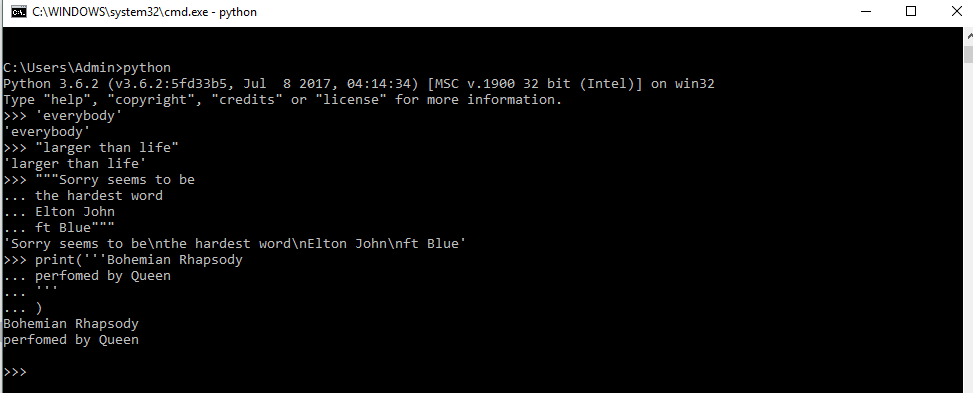
* Python sử dụng hệ thống kiểu latent typing ( tự động xác định kiểu) –Kiểm tra các ràng buộc về kiểu dữ liệu tại thời điểm thực thi.  Khi thực thi, nếu một thao tác trên một đối tượng bị thất bại, thì có nghĩa là đối tượng đó không sử dụng một kiểu thích hợp.
* Sử dụng Python, ta không cần phải khai báo biến. Biến được xem là đã khai báo nếu nó được gán một giá trị lần đầu tiên. Căn cứ vào mỗi lần gán, Python sẽ tự động xác định kiểu dữ liệu của biến. Python có một số kiểu dữ liệu thông dụng sau:
  + int, long: số nguyên (trong phiên bản 3.x long được nhập vào trong kiểu int). Độ dài của kiểu số nguyên là tùy ý, chỉ bị giới hạn bởi bộ nhớ máy tính.
  + float: số thực;
  + list: dãy trong đó các phần tử của nó có thể được thay đổi, chẳng hạn [8, 2, 'b', -1.5]. Kiểu dãy khác với kiểu mảng (array) thường gặp trong các ngôn ngữ lập trình ở chỗ các phần tử của dãy không nhất thiết có kiểu giống nhau. Ngoài ra phần tử của dãy còn có thể là một dãy khác.
  + str: chuỗi ký tự. Từng ký tự trong chuỗi không thể thay đổi. Chuỗi ký tự được đặt trong dấu nháy đơn, hoặc nháy kép; ….

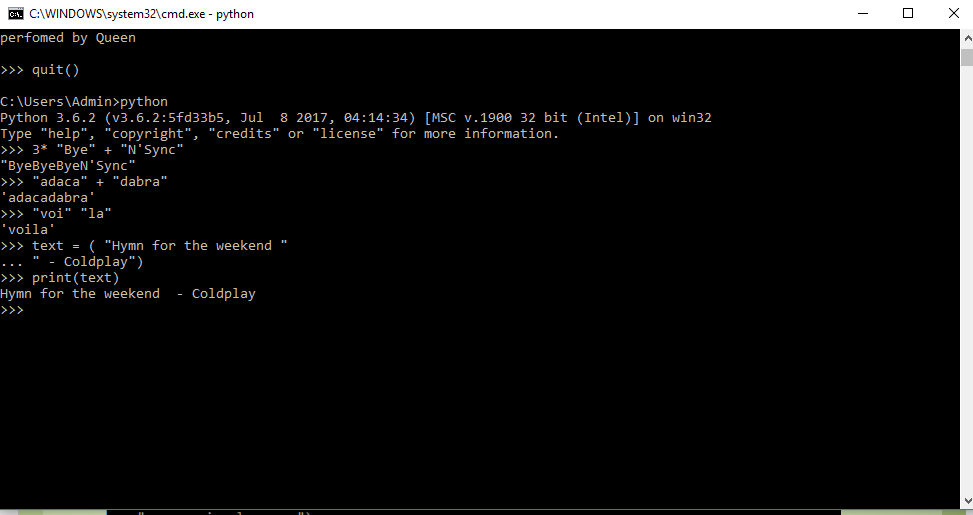
1. Sơ lược sử dụng python:

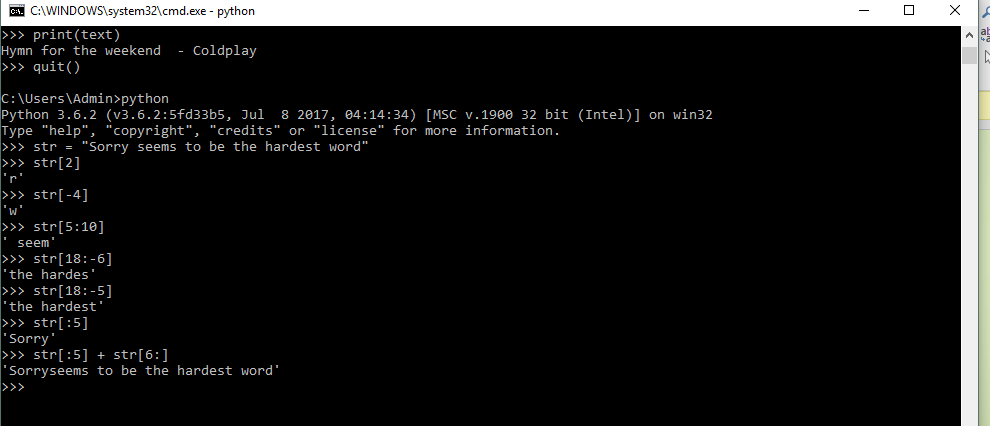
* Toán tử:
  + + Cộng - Trừ \* Nhân / Chia
  + // (chia làm tròn)
  + % (phần dư)
  + \*\* (lũy thừa)
  + & (and) | (or) ~ (not) ^ (xor)
  + << (left shift) >> (right shift)
  + == (bằng) <= >= != (khác)



* Chuỗi:
  + Chuỗi có thể đặt giữ cặp dấu ***nháy đơn***(‘<chuỗi>’), hoặc ***nháy ké***p (“<chuỗi>”) hoặc ***nháy ba*** ("""<chuỗi>""" hoặc ''' <chuỗi> ''')
  + Dùng phép toán **+** để cộng hai chuỗi
  + Dùng phép toán **\*** để lặp lại một chuỗi
  + Dùng số chỉ mục để lấy ký tự trong chuỗi, nếu ***chỉ mục âm*** thì tính từ bên phải chuỗi
  + Lấy một chuỗi con từ vị trí i đến j trong chuỗi str: **str[i:j]**
  + Lấy một chuỗi con từ vị trí i đến cuối chuỗi: **str[i:]**
  + Lấy chuỗi con đầu đến vị trí thứ i trong chuỗi: **str[:i]**
  + In giá trị: print <biểu thức>







* Cấu trúc rẽ nhánh:
  + Dạng 1:

**if** biểu\_thức\_điều\_kiện:

*# lệnh...*

* + Dạng 2:

**if** biểu\_thức\_điều\_kiện:

*# lệnh...*

**else**:

*# lệnh...*

* + Dạng 3:

**if** biểu\_thức\_điều\_kiện\_1:

*# lệnh... (được thực hiện nếu biểu\_thức\_điều\_kiện\_1 là đúng/true)*

**elif** biểu\_thức\_điều\_kiện\_2:

*# lệnh... (được thực hiện nếu biểu\_thức\_điều\_kiện\_1 là sai/false, nhưng biểu\_thức\_điều\_kiện\_2 là đúng/true)*

**else**:

*# lệnh... (được thực hiện nếu tất cả các biểu thức điều kiện đi kèm if và elif đều sai)*

Ví dụ:

x = int(input(**"Nhap x = "**))  
**if** (x > 0):  
 print(**'%s la so duong'** % x)  
**elif** (x < 0):  
 print(**'%s la so am'** % x)  
**else**:  
 print(**'la so 0'**);

Cấu trúc lặp:

**while** biểu\_thức\_đúng:

*# lệnh...*

**for** phần\_tử **in** dãy:

*# lệnh...*

*# In các phần tử trong danh sách songs*songs = [**"Hello"**, **"Lady"**, **"Goodbye"**]  
**for** song **in** songs:  
 print(song)  
  
*# In 10 số từ 0 -> 9***for** i **in** range(10):  
 print(i)  
  
*# In 5 số từ 5 -> 9***for** i **in** range(5, 10):  
 print(i)  
  
*# In các số chẳn 2 -> 10***for** i **in** range(2, 11, 2):  
 print(i)

* Hàm:

**def** tên\_hàm (tham\_biến\_1, tham\_biến\_2, tham\_biến\_n):

*# lệnh...*

**return** giá\_trị\_hàm

Ví dụ:

**def** logout\_view(request):  
 logout(request)  
 **return** redirect(**"/"**)

* Lớp:

**class** **T**ên\_Lớp\_1:

*#...*

**class** **T**ên\_Lớp\_2(Tên\_Lớp\_1):

*"""Lớp 2 kế thừa lớp 1"""*

x = 3 *# biến thành viên của lớp*

*#*

**def** phương\_thức(self, tham\_biến):

*#...*

*# khởi tạo*

a = Tên\_Lớp\_2()

**print** a.x

**print** a.phương\_thức(m) *# m là giá trị gán cho tham biến*

Chương II: Giới thiệu về Django.

Tổng quan:

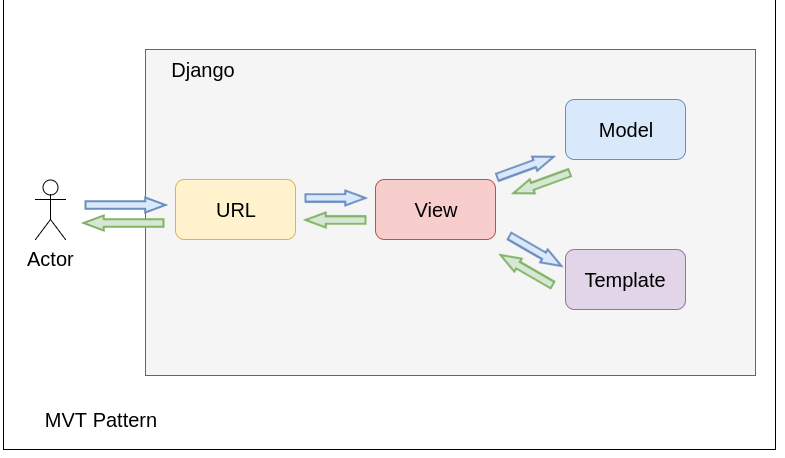
* Django là một framework để phát triển ứng dụng web cho python
* Django được thiết kế để phát triển các ứng dụng web một cách nhanh chóng và dễ dàng.

**Tại sao nên dùng Django**

* **Nhanh**: Django được thiết kế với triết lý làm sao để các lập trình viên đưa các ý tưởng trở thành một sản phẩm nhanh nhất có thể (trong một vài giờ đồng hồ chẳng hạn)
* **Có đầy đủ các thư viện/module cần thiết**: Django có sẵn các thư viện về user authentication, content admin, site maps, RSS feed v.v…
* **Đảm bảo về tính bảo mật**: Không còn các nỗi lo về các lỗi bảo mật thông thường như SQL Injection, cross-site scripting, cross-site request forgery hay clickjacking nữa. Django cũng cung cấp cả phương pháp để lưu mật khẩu an toàn nữa!
* **Khả năng mở rộng tốt**: Django có thể đáp ứng lượng traffic lớn, nghĩa là bạn không cần phải lo lắng về khả năng scale sản phẩm của mình nữa.
* **Tính linh hoạt**: Xây dựng CMS, hoặc Ecommerce Website, hay kể cả Social Network, Scientific Computing Platforms, tất cả đều có thể với Django.

Kiến trúc tổng quan của Django:

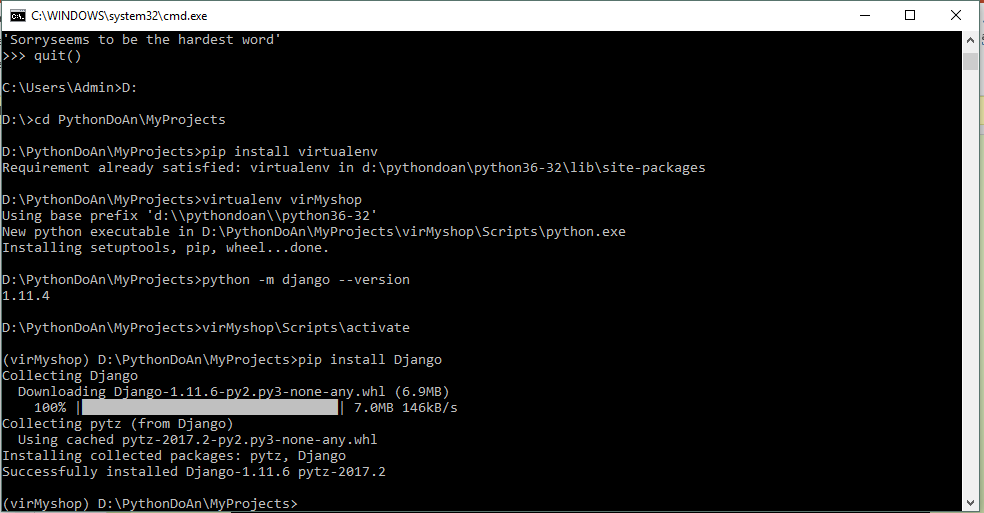
Django là mô hình MVT ( Model – View – Template)

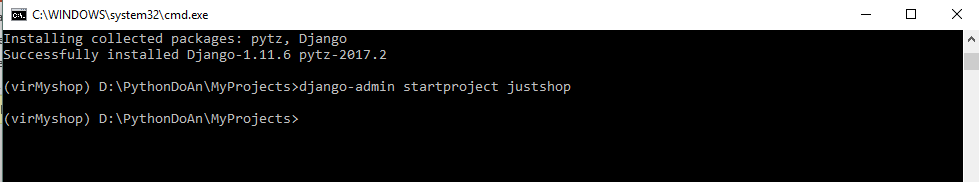


* **Chu trình của một request**: Khi django nhận được một một request, dựa vào **URL** sẽ quyết định view nào sẽ được sử dụng.
* **View** sẽ thao tác với **Model** lấy dữ liệu từ database sau đó gửi các dữ liệu cần thiết tới **Template** để **Template** tạo ra file html và gửi về **View** sau đó trả về cho người dùng.

**Cài đặt Django**: Sử dụng phiên bản Python v3.5.2

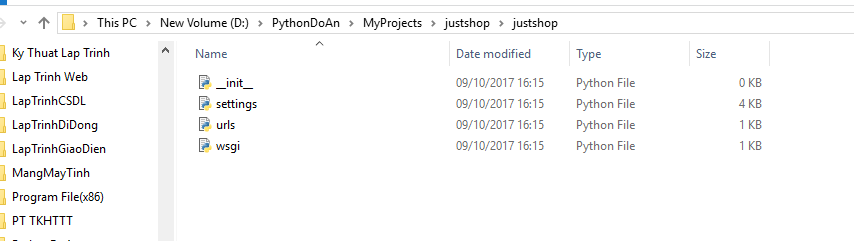
* Cài đặt database: Chỉ cần thiết nếu ta dùng các engine database lớn như mysql, postgres, oracle, ... Mặc định python đã có sẵn SQLite
* Cài đặt môi trường ảo virtualenv
  + Mở command line và gõ lệnh: **pip install virtualenv**
  + Tạo virtualenv: **virtualenv virMyshop**
  + Kích hoạt virtualenv: **virMyshop \Script\activate**
* Cài đặt Django
  + Mở command line và gõ lệnh: **pip install Django**
  + Kiểm tra đã cài đặt django: **python –m django --version**





Sau khi startproject ta có được thư mục justshop với cấu trúc sau:

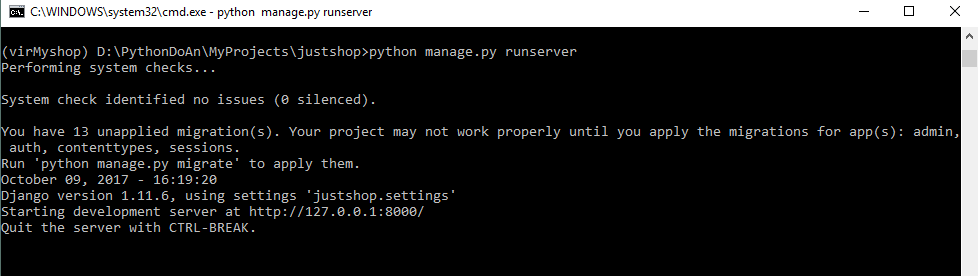
justshop/  
 manage.py  
 justshop/  
 \_\_init\_\_.py  
 settings.py  
 urls.py  
 wsgi.py



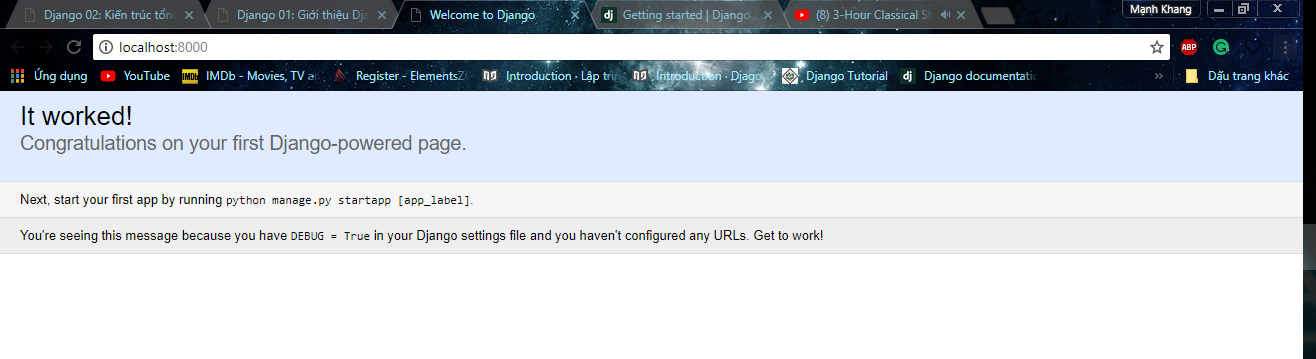
* **Thư mục gốc just/ chứa project, có thể thay đổi tên tùy ý.**
* **manage.py:sử dụng command line cho phép tương tác project django theo nhiều cách khác nhau**
* **Thư mục con justshop/ là gói python cho project**
* **\_\_init\_\_.py là file rổng để nói python biết đây là thư mục python package**
* **settings.py chứa thông tin cấu hình cho project**
* **urls.py khai báo các url cho project**
* **wsgi.py:** Dùng để deploy project của bạn lên Server.

Để kiểm tra project django hoạt động ta làm như sau:

* + Chuyển đến thư mục project;
  + Chạy lệnh **python manage.py runserver** sẽ có kết quả như bên dưới



Sau đó truy cập vào <http://127.0.0.1:8000/> Nếu hiện ra như sau là thành công:



### **Tạo một application**

Tại thư mục gốc của project (chứu file manage.py), để tạo một application với tên là “accounts” ta chạy câu lệnh:

python manage.py startapp accounts

Screen Clipping

Django sẽ tự động tạo cho chúng ta thư mục “book” với cấu trúc như sau:

accounts/

   \_\_init\_\_.py

   admin.py

   models.py

   tests.py

   views.py

Trong đó:

**\_\_init\_\_.py:**   Để chỉ cho python rằng thư mục này là một package.

**admin.py**: File giúp ta tạo các có thể thay đổi được trong trang quản trị

**models.py**: File chứa mã code của model

**tests.py**: File chứa mã code của unit tests

**views.py**: Nới chứa các mã code của view.

Và để project sử dụng được application ta phải khai báo nó trong file settings.py:

INSTALLED\_APPS = (

   'django.contrib.admin',

   'django.contrib.auth',

   'django.contrib.contenttypes',

   'django.contrib.sessions',

   'django.contrib.messages',

   'django.contrib.staticfiles',

   'accounts',

)

* Mỗi app trong **INSTALLED\_APPS** sẽ có các table riêng cần tạo trong database trước khi sử dụng chúng. Để làm điều này thì tại thư mục project, ta thực thi lệnh: **python manage.py migrate**

PHẦN 2: TRÌNH BÀY ĐỒ ÁN.

* Thiết lập môi trường
* Tạo models danh mục sản phẩm
* Tạo model tài khoản trong admin site
* Hiển thị danh mục sản phẩm
* Tạo danh mục template

Tạo môi trường để lập trình django: (CMD)

C:>pip install virtualenv

C:>pip install django

Tạo virtualenv :

C:> virtualenv myapp

Kích hoạt máy ảo :

C:> cd myapp\Scripts\activate

Sau khi kích hoạt ta cài django cho máy ảo này:

C:>pip install django

Kiểm tra các module đang sử dụng:

C:>pip freeze

(Django ==1.11.4,

Pytz == 2017.2)

Tạo 1 dự án mới:

C:>django-admin startproject shop

Tạo 1 app để chạy chương trình :

Di chuyển tới project : C:>cd myshop

C:>django-admin startapp shop

Di chuyển tới thư mục app mà chúng ta làm để tiến hành tạo

C:> cd myshop

Muốn django hiểu app này thì ta phải thiết lập trong setting.py của project

INSTALL\_APP = (

….

‘shop’,

)

* Tạo models danh mục sản phẩm

App shop sẽ lưu các model này:

Để sử dụng được FIELD IMAGE ta phải cài module Pillow

C:>pip install Pillow==3.0.0

Chạy lệnh lưu dữ liệu:

C:>python manage.py makemigration

C:>python manage.py migrate

* Tạo model tài khoản trong admin site

Tạo tài khoản admin :

C:>python manage.py createsuperuser

Chạy server để kiểm tra thử :

C:>python manage.py runserver

* Hiển thị danh mục sản phẩm

Them đường dẫn trong file urls.py trong shop:

urlpatterns = [

url(r'^$', views.product\_list, name='product\_list'),

url(r'^(?P<category\_slug>)[-\w]+/$', views.product\_list, name='product\_list\_by\_category'),

url(r'^(?P<id>\d+)/(?P<slug>[-\w]+)/$', views.product\_detail, name='product\_detail'),

]

Sửa urls.py của myshop:

Tạo thư mục Template trong app:

Tạo shopping cart:

C:>python manage.py startapp cart

Thêm cart vào INSTALL\_APP

Django không nhận tập tin static:

B1 : Xóa bộ nhớ cache trình duyệt

B2: python manage.py collectstatic --noinput --clear

**Launching asynchronous tasks with Celery - Khởi tạo những tác vụ không đồng bộ với Celery**

C:>pip install celery

myshop/myshop/\_\_init\_\_.py

# import celery

from .celery import app as celery\_app

Tạo mới file : celery.py

import os

from celery import Celery

from django.conf import settings

# set the default Django settings module for the 'celery' program.

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'myshop.settings')

app = Celery('myshop')

app.config\_from\_object('django.conf:settings')

app.autodiscover\_tasks(lambda: settings.INSTALLED\_APPS)

Cấu hình cho order:

from celery import task

from django.core.mail import send\_mail

from .models import Order

@task

def order\_created(order\_id):

"""

Task to send an e-mail notification when an order is successfully created.

"""

order = Order.objects.get(id=order\_id)

subject = 'Order nr. {}'.format(order.id)

message = 'Dear {},\n\nYou have successfully placed an order. Your order id is {}.'.format(order.first\_name,

order.id)

mail\_sent = send\_mail(subject, message, 'admin@myshop.com', [order.email])

return mail\_sent

orders/views.py

from .tasks import order\_created