

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG

-----o0o-----



ISO 9001:2015

ĐỒ ÁN TỐT NGHIỆP

NGÀNH CÔNG NGHỆ THÔNG TIN

HẢI PHÒNG 2020

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG

-----o0o-----

**TÌM HIỂU LẬP TRÌNH PYTHON VÀ ỨNG DỤNG
PHÁT TRIỂN ỨNG DỤNG WEB VỚI DJANGO**

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành: Công nghệ Thông tin

HẢI PHÒNG 2019

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG

-----o0o-----

**TÌM HIỂU LẬP TRÌNH PYTHON VÀ ỨNG DỤNG
PHÁT TRIỂN ỨNG DỤNG WEB VỚI DJANGO**

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành: Công nghệ Thông tin

Sinh viên thực hiện: Nguyễn Đại Cường

Giáo viên hướng dẫn: TS. Đỗ Văn Chiểu

Mã sinh viên: 1512111007

HẢI PHÒNG 2020

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

-----o0o-----

NHIỆM VỤ THIẾT KẾ TỐT NGHIỆP

Sinh viên: Nguyễn Đại Cường

Mã số: 1512111007

Lớp: CT1901C

Ngành: Công nghệ Thông tin

Tên đề tài: Tìm hiểu lập trình Python và ứng dụng phát triển ứng dụng web với Django

LỜI CẢM ƠN

Lời đầu tiên em xin chân thành cảm ơn các thầy, cô trong khoa Công Nghệ Thông Tin cũng như toàn thể mọi người trong ngôi trường Đại học Dân lập Hải Phòng đã tạo điều kiện thuận lợi cho em trong suốt quá trình học tập tại trường cũng như trong thời gian thực hiện đề án tốt nghiệp.

Đặc biệt, em muốn gửi lời cảm ơn tới Tiến Sĩ - Đỗ Văn Chiêu giảng viên trực tiếp hướng dẫn tận tình chỉ bảo giúp em khắc phục những khó khăn, thiếu sót để có thể hoàn thành các phần trong đề án tốt nghiệp từ lý thuyết cho tới thực hành sử dụng công cụ.

Với hiểu biết tìm tòi của bản thân và sự chỉ bảo hướng dẫn tận tình của giảng viên em đã cố gắng hoàn thành đề án một cách tốt nhất có thể nhưng cũng không thể tránh được thiếu sót. Kính mong nhận được sự đóng góp ý kiến từ thầy cô để em có thể nâng cao cũng như bổ sung thêm kiến thức cho bản thân, hoàn thiện đề án với một kết quả tốt và hoàn chỉnh hơn.

Em xin chân thành cảm ơn!

Hải Phòng, ngày 21 tháng 09 năm 2019.

Sinh viên thực hiện

Nguyễn Đại Cường

MỤC LỤC

LỜI CẢM ƠN

MỤC LỤC

MỞ ĐẦU.....	4
Chương I NGÔN NGỮ LẬP TRÌNH PYTHON.....	7
1. Cài đặt môi trường PyCharm	7
2. Cài đặt python	8
3. Tạo file và viết mã Python trên PyCharm	9
4. Các Khái Niệm Cơ Bản Trong Lập Trình Python	10
5. Cấu trúc dữ liệu là gì, các kiểu cấu trúc dữ liệu trong python.....	11
6. Cấu trúc điều khiển trong python	12
a. Lệnh IF	16
b. Lệnh FOR.....	17
c. Lệnh While.....	18
7. Sử Dụng Hàm Trong Python	19
a. Hàm (Function)	19
b. Các thông số của Hàm (Function Parameters)	20
c. Câu lệnh return trong Python	21
Chương II PHÁT TRIỂN ỨNG DỤNG WEB VỚI DJANGO.....	20
1. Cài đặt django	24
2. Tạo project	26
3. Chạy server	26
4. Tạo Web App.....	27
5. Model	28
6. Hệ thống admin.....	31
7. View và templates.....	33
a. View	33
b. Templates	35
c. Đặt namespace cho URL.....	37

8. Upload file.....	39
a. Tạo form upload.....	39
b. Tạo templates và file template	40
9. Form trong django.....	42
a. Form sử dụng model	42
b. Form không sử dụng model	47
10. Hệ thống user trong Django	51
a. Ví dụ về phân quyền user.....	51
b. Phân quyền view (decorator)	54
11. Custom user model trong Django	57
12. Tùy chỉnh giao diện admin (admin custom admin site django)	59
Chương III XÂY DỰNG KHUNG WEBSITE BÁN HÀNG	58
1. Phân tích cơ sở dữ liệu.....	65
2. Xây dựng khung website bán hàng.....	66
KẾT LUẬN.....	76
DANH MỤC TÀI LIỆU THAM KHẢO.....	77

MỞ ĐẦU

Trong cuộc sống của mỗi chúng ta khi nền kinh tế ngày càng phát triển, ngành công nghệ thông tin trở thành một trợ thủ, điều không thể thiếu với hầu hết các ngành nghề. Có thể ban đầu nhiều người sẽ cảm thấy chưa cần thiết vì chỉ kinh doanh ở quy mô nhỏ, hoặc chưa sẵn sàng sử dụng vì nghĩ rằng phần mềm là một khái niệm gì đó rất mới mẻ, phức tạp, khó sử dụng.

Ngày nay trong cuộc sống 4.0 của chúng ta gần như không thể thiếu được những khái niệm liên quan đến công nghệ thông tin, từ các phương tiện truyền thông, xã hội cho đến lĩnh vực kinh doanh, quản lý..... Mọi thứ đều cần những phần mềm cũng như sản phẩm của công nghệ thông tin để hỗ trợ giúp nâng cao phát triển vững mạnh hơn. Ví dụ trong lĩnh vực kinh doanh buôn bán hàng hóa theo cách truyền thống vẫn còn tồn tại nhiều nhược điểm như thống kê chi tiết chưa chính xác, quy trình thanh toán chưa được chặt chẽ và nhanh chóng .

Cùng với sự phát triển của nhiều ngôn ngữ lập trình web như PHP, Ruby, Scheme thì Python là một cái tên đáng chú ý. Hiện nay ngôn ngữ Python được xếp hạng thứ 1 trong Top 10 các ngôn ngữ lập trình phổ biến nhất đang được thế giới sử dụng. Python là một ngôn ngữ có hình thái rất sáng sủa, cấu trúc rõ ràng, thuận tiện cho người mới bắt đầu học lập trình. Cấu trúc của Python còn cho phép người sử dụng viết mã lệnh với số lần gõ phím tối thiểu nhất. Python là một ngôn ngữ lập trình đơn giản nhưng lại rất hiệu quả. Bên cạnh đó, Python là một ngôn ngữ có tính hướng đối tượng cao. Với ngôn ngữ lập trình python là một ngôn ngữ lập trình đa năng với nhiều ưu điểm vượt trội, và đang đứng ở vị trí số một trong top các ngôn ngữ lập trình hiện nay .

Đồ án gồm có các chương sau: Chương I giới thiệu về ngôn ngữ lập trình python thịnh hành nhất hiện nay cũng như lịch sử hình thành và các phiên bản. Chương II giới thiệu về một Framework được viết bằng ngôn ngữ lập trình python là Django và các ứng dụng. Chương III demo một website bán hàng được xây dựng bởi Django.

Chương I

NGÔN NGỮ LẬP TRÌNH PYTHON

Giới thiệu

Lịch sử hình thành

Python đã được hình thành vào cuối những năm 1980 và được bắt đầu thực hiện vào tháng 12/1989 bởi Guido van Rossum tại CWI tại Hà Lan như là người kế thừa của ngôn ngữ ABC (tự lấy cảm hứng từ SETL) có khả năng xử lý ngoại lệ và giao tiếp với hệ điều hành Amoeba. Van Rossum là tác giả chính của Python, và vai trò trung tâm của ông tiếp tục trong việc quyết định hướng phát triển của Python được phản ánh trong tiêu đề mà cộng đồng Python dành cho ông “Độc tài nhân từ cho cuộc sống” (benevolent dictator for life)(BDFL).

Python 2.0 được phát hành vào ngày 16/10/2000, với nhiều tính năng chính mới bao gồm một bộ dọn rác đầy đủ và hỗ trợ Unicode. Với phiên bản này, quá trình phát triển đã được thay đổi và trở thành minh bạch hơn và được cộng đồng ủng hộ.

Python 3.0 (còn được gọi là Python 3000 hoặc Py3k), một bản phát hành lớn, không tương thích ngược, được phát hành vào ngày 03/12/2008 sau một thời gian dài thử nghiệm. Nhiều trong số các tính năng chính của nó đã được điều chỉnh để tương thích ngược với Python 2.6 và 2.7. Các tính năng và triết lý phát triển Python là 1 ngôn ngữ lập trình đa hình: lập trình hướng đối tượng và hướng cấu trúc được hỗ trợ đầy đủ, và có 1 số tính năng của ngôn ngữ hỗ trợ lập trình theo chức năng và lập trình hướng khía cạnh (Aspect-oriented programming). Nhiều mô hình khác được hỗ trợ bằng việc sử dụng các phần mở rộng, bao gồm thiết kế theo hợp đồng (design by contract) và lập trình luận lý.

Các trang như Mozilla, Reddit, Instagram và PBS đều được viết bằng Python.

Ngôn ngữ lập trình Python được dùng vào các mục đích :

- Phát triển web (trên máy chủ)
- Phát triển phần mềm
- Tính toán một cách khoa học
- Lên kịch bản cho hệ thống

Tại Sao Nên Học Lập Trình Python?

- Python hỗ trợ nhiều nền tảng khác nhau (Windows, Mac, Linux, Raspberry Pi, etc).
- Python có cú pháp đơn giản, dễ đọc hiểu và rất gần gũi với tiếng Anh.
- Cú pháp của Python giúp lập trình viên sử dụng ít dòng code để lập trình cùng một thuật toán hơn so với các ngôn ngữ lập trình khác.
- Python sử dụng trình thông dịch để thực thi các dòng code. Do đó, những dòng code có thể được thực thi ngay lập tức mà không cần biên dịch toàn bộ chương trình. Như vậy giúp chúng ta kiểm tra code nhanh hơn.

Python cũng hỗ trợ hàm, thủ tục, hay kể cả lập trình hướng đối tượng.

Để viết mã nguồn Python, ta có thể sử dụng bất kỳ một trình soạn thảo nào, kể cả những trình soạn thảo đơn giản nhất như NotePad. Tuy nhiên, để phát triển các ứng dụng một cách hiệu quả hơn, ta nên sử dụng một IDE, để có thể tiết kiệm thời gian và công sức viết code. ở đây chúng ta sử dụng một trong những IDE thông dụng nhất để lập trình ứng dụng Python, đó là PyCharm IDE.

1. Cài đặt môi trường PyCharm

Để download Pycharm, ta truy cập vào:

<https://www.jetbrains.com/pycharm/download/#section=windows> và tải về

Hình 1.1 Download pycharm

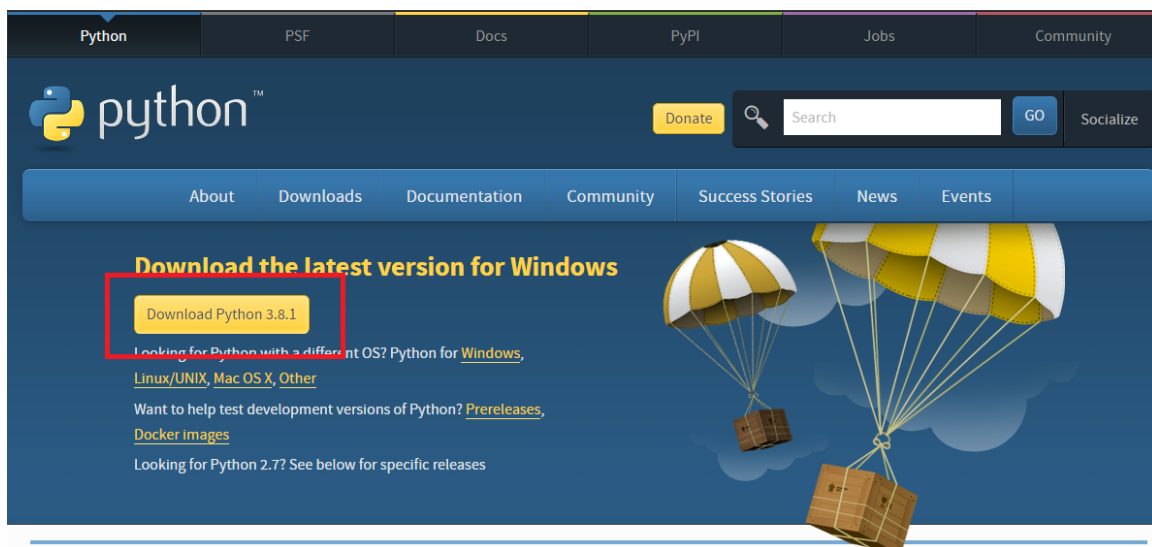
Sau khi đã tải phiên bản pycahrn về máy ta tiến hành cài đặt pycharm theo các bước trong hướng dẫn và những tùy chọn cài đặt.



Hình 1.2 Cài đặt pycharm

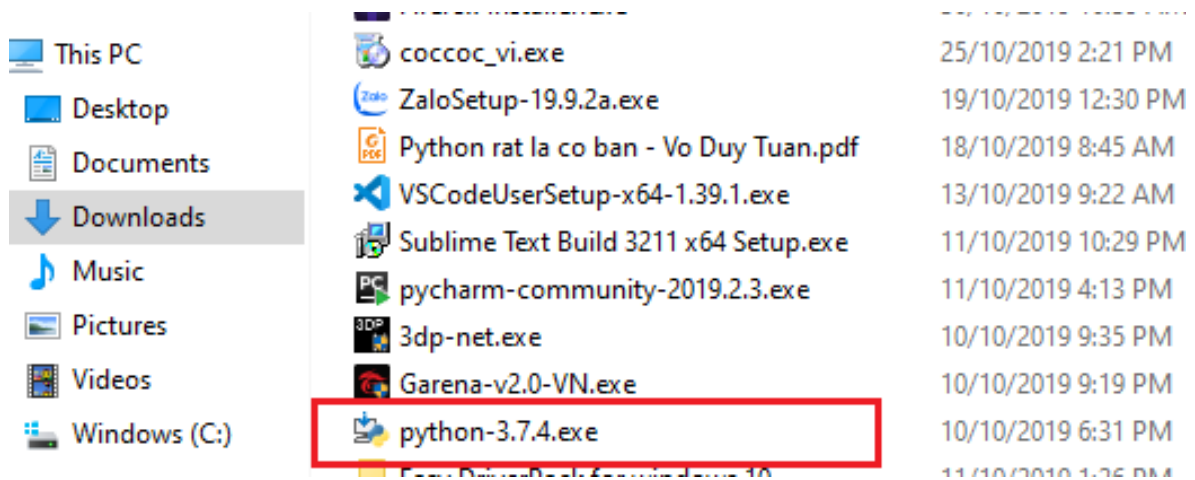
2. Cài đặt python

Để download Python, ta truy cập địa chỉ trang chủ của python qua địa chỉ <https://www.python.org/downloads/> và tải python bản mới nhất.



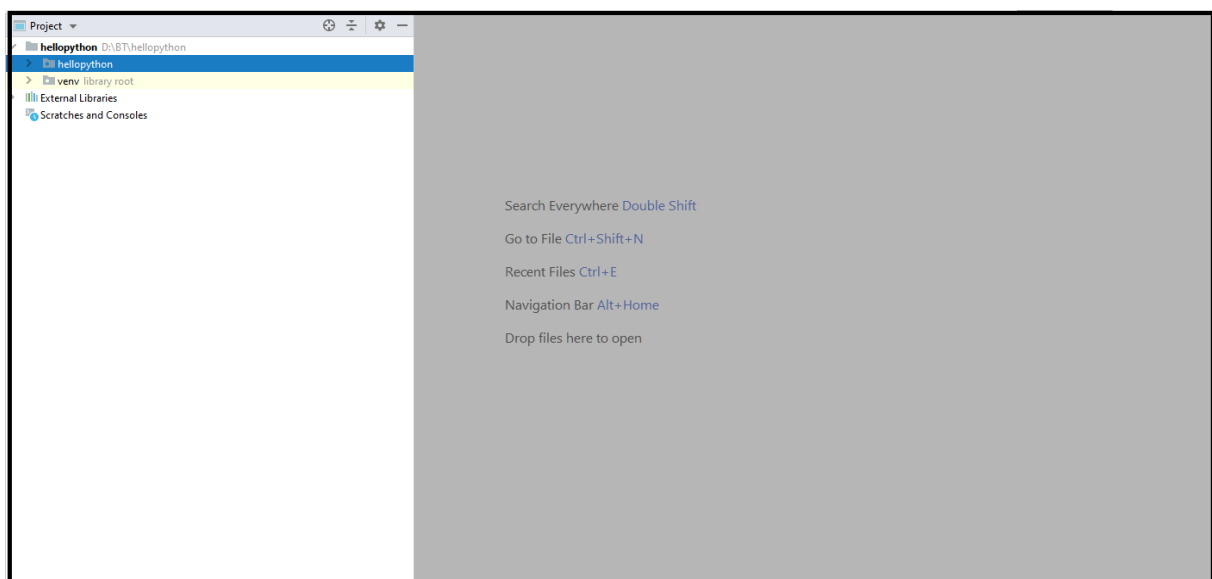
Hình 2.1 Download python

Sau khi download xong nhấn vào file .exe vừa download và kích hoạt để cài đặt. Lúc này chỉ cần thực hiện lần lượt các bước theo hướng dẫn như chọn các thành phần cài đặt, xác định đường dẫn cần thiết và các thông số khác.



Hình 2.2 Cài đặt python

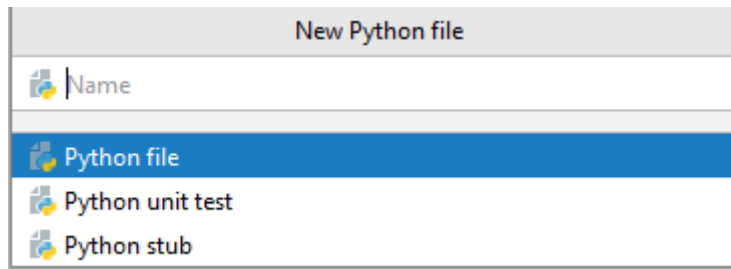
Sau khi quá trình cài đặt được hoàn tất, giao diện của PyCharm sẽ như hình 2.2



Hình 2.2 Tạo project

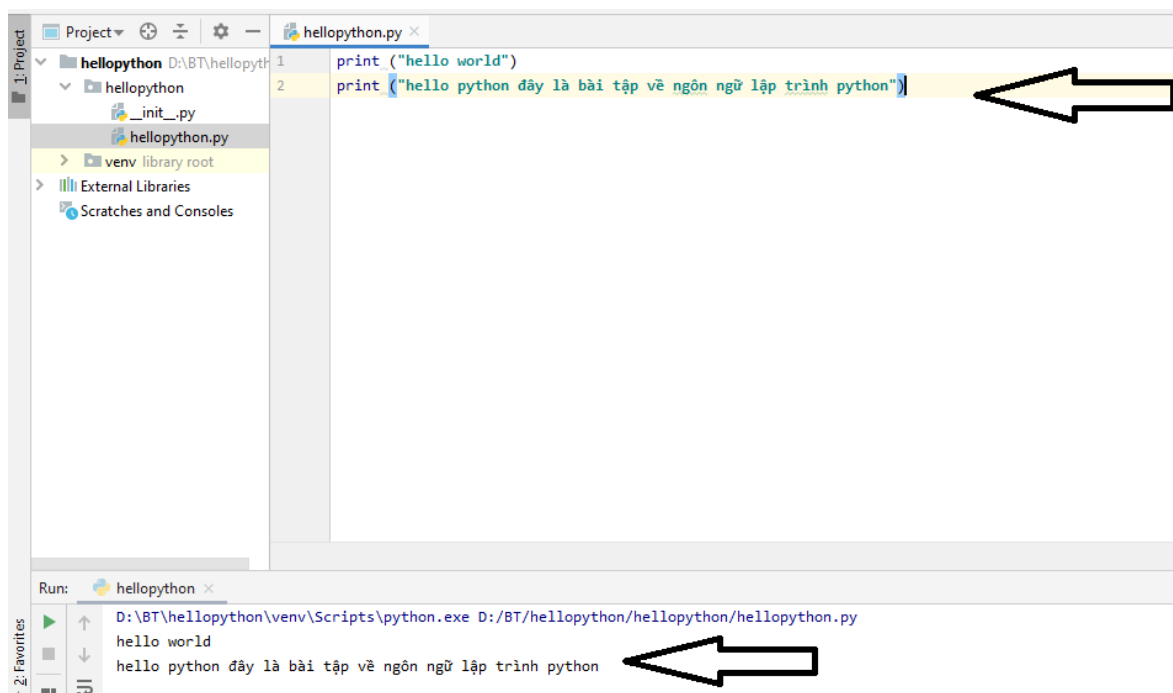
3. Tạo file và viết mã Python trên PyCharm

Sau khi đã tạo xong Project, ta click phải chuột lên Project, rồi tạo mới một Python File, để tạo một file mã nguồn Python. Và đặt tên cho file đó, file được viết bằng ngôn ngữ lập trình python có đuôi “py”



Hình 3.1 Tạo file python

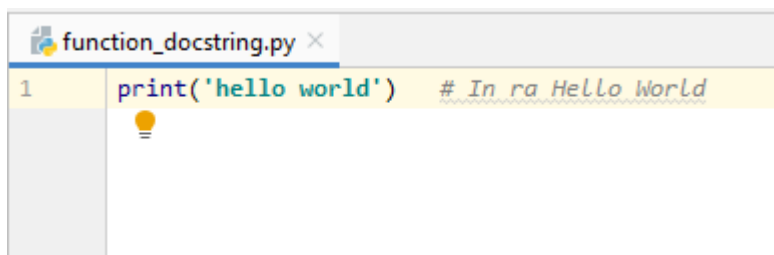
Thử viết một ví dụ in ra **“Hello world”** trên pycharm bằng ngôn ngữ lập trình python.



Hình 3.2 Ví dụ Hello world.

4. Các Khái Niệm Cơ Bản Trong Lập Trình Python

Comments : Trong Python bất kỳ văn bản nào ở bên phải biểu tượng # thì sẽ được trình biên dịch hiểu là một comment và không biên dịch phần đó.



Sử dụng càng nhiều comment hữu ích trong chương trình của bạn sẽ làm cho công việc lập trình của bạn dễ dàng hơn:

- Giải thích các giả định
 - Giải thích các quyết định quan trọng
 - Giải thích chi tiết quan trọng
 - Giải thích vấn đề bạn đang cố gắng giải quyết
 - Giải thích các vấn đề đang cố gắng khắc phục trong chương trình của mình, v.v.
-
- **Hằng số** (Literal Constants): Ví dụ về một hằng số theo nghĩa đen là một số như 5, 1.23 hoặc một chuỗi như 'python' hay "It's a string!". Nó được gọi là nghĩa đen bởi vì sử dụng giá trị của nó theo nghĩa đen. Số 2 luôn luôn đại diện cho chính nó và không có gì khác và nó là một hằng số vì giá trị của nó không thể thay đổi. Do đó, tất cả những giá trị này được gọi là hằng số.
 - **Số** (Numbers) : Số chủ yếu có hai loại – số nguyên (integer) và số thực (float).
 - **Chuỗi** (String): Một chuỗi là một dãy các ký tự .Chuỗi về cơ bản chỉ là một loạt các từ.
 - **Biến**: Biến chính xác như tên gọi của nó, tức là giá trị của nó có thể thay đổi. Các biến có thể giúp lưu trữ bất cứ cái gì nếu có thể định nghĩa được nó. Các biến chỉ là một phần của bộ nhớ máy tính nơi lưu trữ một số thông tin.
 - **Đối tượng** (Object): Python đề cập đến bất cứ điều gì được sử dụng trong một chương trình như là một đối tượng.

5. Cấu trúc dữ liệu là gì, các kiểu cấu trúc dữ liệu trong python.

Việc tổ chức, quản lý và lưu trữ dữ liệu rất quan trọng vì nó cho phép truy cập dễ dàng hơn và sửa đổi hiệu quả. Cấu trúc dữ liệu (Data Structure) cho phép bạn sắp xếp dữ liệu của mình theo cách cho phép bạn lưu trữ các bộ dữ liệu được thu thập, liên quan đến chúng và theo đó mà thực hiện các thao tác trên chúng.

Python có hỗ trợ ngầm cho Cấu trúc dữ liệu cho phép lưu trữ và truy cập dữ liệu. Các cấu trúc này được gọi là List, Dictionary, Tuple và Set.

Python cho phép người dùng tạo Cấu trúc dữ liệu của riêng họ, cho phép toàn quyền kiểm soát chức năng. Các cấu trúc dữ liệu nổi bật nhất là Stack, Queue, Tree, Linked List, v.v. đồng thời cũng có sẵn trong các ngôn ngữ lập trình khác.

Cấu trúc dữ liệu tích hợp (Built-in Data Structures)

Về cấu trúc dữ liệu trong Python, các Cấu trúc dữ liệu này được tích hợp sẵn với Python giúp lập trình dễ dàng hơn và giúp các lập trình viên sử dụng chúng để có được các giải pháp nhanh hơn. Và có các kiểu cấu trúc dữ liệu là :

List : Được sử dụng để lưu trữ dữ liệu của các loại dữ liệu khác nhau một cách tuần tự. Có các địa chỉ được gán cho mọi thành phần của danh sách, được gọi là Index. Giá trị chỉ mục bắt đầu từ 0 và tiếp tục cho đến khi phần tử cuối cùng được gọi là chỉ số dương.

- **Dictionary**: Được sử dụng để lưu trữ các cặp key-value. Để hiểu rõ hơn, hãy nghĩ đến một thư mục điện thoại nơi hàng trăm và hàng ngàn tên và số tương ứng của chúng đã được thêm vào. Bây giờ các giá trị không đổi ở đây là Tên và Số điện thoại được gọi là các phím. Và các tên và số điện thoại khác nhau là các giá trị đã được đưa vào các phím. Nếu truy cập các giá trị của các phím, sẽ nhận được tất cả tên và số điện thoại. Vì vậy, đó là những gì một cặp key-value. Và trong Python, cấu trúc này được lưu trữ bằng Dictionary
- **Set**: Là một tập hợp các yếu tố không có thứ tự là duy nhất. Có nghĩa là ngay cả khi dữ liệu được lặp lại nhiều lần, nó sẽ chỉ được nhập vào tập hợp một lần.
- **Tuple (các bộ dữ liệu)** : Tuples giống như các list với ngoại lệ là dữ liệu một khi được nhập vào bộ dữ liệu không thể thay đổi bất kể điều gì. Ngoại lệ duy nhất là khi dữ liệu bên trong Tuple có thể thay đổi, chỉ sau đó dữ liệu Tuple có thể được thay đổi.

LIST	TUPLE
Được sử dụng cho các loại dữ liệu đồng nhất	Thường được sử dụng cho các loại dữ liệu không đồng nhất
Có thể thay đổi trong môi trường	Bất biến trong môi trường giúp lặp lại nhanh hơn
Không có yếu tố bất biến	Các yếu tố bất biến có thể được sử dụng làm key cho từ điển
Không đảm bảo rằng dữ liệu được bảo vệ chống ghi	Việc thực hiện một bộ dữ liệu không thay đổi đảm bảo rằng nó được bảo vệ chống ghi

Hình 5.1 So sánh Data List và Data Tuple

6. Cấu trúc điều khiển trong python

Python luôn chạy một loạt các câu lệnh theo thứ tự từ trên xuống một cách chính xác. Câu lệnh điều khiển là loại câu lệnh được dùng để điều khiển luồng chạy của các câu lệnh khác trong chương trình.

Điều này đạt được bằng cách sử dụng các câu lệnh điều khiển. Có 3 lệnh kiểm soát cấu trúc điều khiển của Python là: **if**, **for** và **while**.

a. Lệnh IF

Được sử dụng để kiểm tra một điều kiện: **nếu** điều kiện là đúng sẽ chạy một khối các câu lệnh (được gọi là **if-block**), **nếu sai** chương trình sẽ xử lý một khối các câu lệnh khác (được gọi là **else-block**).

Ví dụ :

Cho một số nguyên cho trước, nhập vào một số nguyên khác và báo về các kết quả. Nếu thỏa mãn các điều kiện thì in ra các kết quả khác nhau

```
if.py x
1   number = 22
2   guess = int(input('Mời nhập vào số nguyên: '))
3   if guess == number:
4       # Bat dau Block moi o day
5       print('Chính xác!')
6       # Ket thuc Block
7   elif guess < number:
8       # Bat dau Block khac
9       print('Không, Số đã nhập bé hơn!')
10      # Ban co the them vao bat cu gi vao day...
11  else:
12      print('Không, số đã nhập lớn hơn')
13
14  print('Kết thúc!')
```

Kết quả cho thấy khi chúng ta nhập một số nguyên bất kì. Nếu thỏa mãn điều kiện bằng với số cho trước thì in ra “chính xác”. Và kết quả khác khi điền vào số lớn hoặc nhỏ hơn số cho trước.

Kết quả:


```
D:\BT\hellopython\venv\Scripts\python.exe
Nhap vao mot so nguyen: 25
Khong, So da nhap l hon
Ket thuc!

Process finished with exit code 0
```

b. Lệnh FOR

Câu lệnh `for..in` là một câu lệnh lặp khác, nó lặp đi **lặp lại** qua một chuỗi (sequences) các đối tượng tức là đi qua từng mục trong một chuỗi. Chúng ta sẽ tìm hiểu nhiều hơn về sequences trong bài về cấu trúc dữ liệu trong Python sau. Những gì bạn cần biết ngay bây giờ là một chuỗi trình từ chỉ là một tập hợp các items.

Ví dụ

```
for.py x
1   for i in range(1,20):
2       print(i)
3   else:
4       print('KẾT THÚC GIÁ TRỊ!')
```

Kết quả Vòng lặp for trong Python hoàn toàn khác với vòng lặp trong C / C ++. vòng lặp for trong Python tương tự như vòng lặp foreach trong C#. Trong C / C ++, nếu bạn muốn viết `for(int i = 0; i < 5; i ++)`, thì trong Python bạn chỉ cần viết `for i in range(0,5)`. Vòng lặp for đơn giản hơn, biểu cảm hơn và ít bị lỗi hơn trong Python.

```
D:\BT\hellopython\venv\Scripts\python.exe D:/BT/hellopython/hellopython/for.py
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
Ket thuc in gia tri!

Process finished with exit code 0
```

c. Lệnh While

Trong Python: Câu lệnh while cho phép liên tục thực thi một khối các câu lệnh miễn là điều kiện là đúng. Một câu lệnh while là một ví dụ về cái được gọi là câu lệnh lặp. Một câu lệnh while có thể có một mệnh đề khác tùy chọn.

Ví dụ nhập vào một số nguyên sao cho thỏa mãn các điều kiện cho trước, sau khi thỏa mãn một khối các lệnh thì in ra kết quả “Hoàn thành”.

```
while.py x
1   number = 23
2   running = True
3
4   while running:
5       guess = int(input('Nhập vào một số nguyên: '))
6       if guess == number:
7           print('Chính xác ')
8           running = False
9       elif guess < number:
10          print('Số đã nhập nhỏ hơn số đã chọn.')
11      else:
12          print('Số đã nhập lớn hơn số đã chọn')
13
14  print('Hoàn thành')
```

Kết quả :

```
D:\BT\hellopython\venv\Scripts\python.exe
Nhập vào một số nguyên: 20
Số đã nhập nhỏ hơn số đã chọn.
Nhập vào một số nguyên: 24
Số đã nhập lớn hơn số đã chọn
Nhập vào một số nguyên: 23
Chính xác
Hoàn thành
```

7. Sử Dụng Hàm Trong Python

a. Hàm (Function)

Là những phần tái sử dụng của chương trình. Chúng cho phép đặt tên cho một khối các câu lệnh, cho phép chạy khối đó bằng cách sử dụng tên được chỉ định ở bất kỳ đâu trong chương trình với số lần không hạn chế. Điều này được gọi là gọi hàm. Các Hàm được xác định bằng cách sử dụng từ khóa “def“. Sau khi từ khóa này xuất hiện một tên định danh cho hàm, theo sau là một cặp dấu ngoặc đơn có thể kèm theo một số tên của các biến và bởi dấu hai chấm cuối cùng kết thúc dòng. Tiếp theo sau là khối các câu lệnh của Hàm.

Một ví dụ về hàm đơn giản:

```
function.py x
1 def say_hello():
2     # Block này thuộc vào hàm
3     print('hello world!')
4
5
6     # Kết thúc hàm
7
8     say_hello() # Gọi hàm lần thứ nhất
9     say_hello() # Gọi hàm lần thứ hai
```

Kết quả :

```
D:\BT\hellopython\venv\Scripts\python.exe
hello world!
hello world!
Process finished with exit code 0
```

b. Các thông số của Hàm (Function Parameters)

Một hàm có thể lấy tham số, là các giá trị bạn cung cấp cho hàm để hàm có thể sử dụng các giá trị đó cho một mục đích cụ thể. Các tham số này giống như các biến ngoại, trừ các giá trị của các biến này được xác định khi chúng ta gọi hàm và đã được gán các giá trị khi hàm chạy.

Các tham số được chỉ định trong cặp dấu ngoặc trong khai báo hàm, được phân tách bằng dấu phẩy. Khi chúng ta gọi hàm, chúng ta cung cấp các giá trị theo cùng một cách. Lưu ý thuật ngữ được sử dụng – các tên được đưa ra trong định nghĩa hàm được gọi là **tham số** trong khi các giá trị bạn cung cấp trong lệnh gọi hàm được gọi là **đối số**.

Ví dụ :

```
function_param.py x
1 def print_max(a, b):
2     if a > b:
3         print(a, ' la so lon nhat')
4     elif a == b:
5         print(a, ' bang ', b)
6     else:
7         print(b, ' la so lon nhat')
8
9
10 print_max(5, 6)
11
12 x = 7
13 y = 8
14
15 print_max(x, y)
```

Kết quả:

```
↑ D:\BT\hellopython\venv\Scripts\python.exe
↓ 6 la so lon nhat
  8 la so lon nhat
⏏ Process finished with exit code 0
```

c. Câu lệnh return trong Python

Lệnh return thường được dùng để thoát hàm và trở về nơi mà tại đó hàm được gọi. Lệnh này có thể chứa biểu thức được tính toán và giá trị trả về. Nếu không có biểu thức nào trong câu lệnh hoặc không có lệnh return trong hàm thì hàm sẽ trả về None. Lệnh return dùng để trả về một giá trị (hoặc một biểu thức), hoặc đơn giản là trả về "không gì cả". Khi lệnh return được thực thi, hàm sẽ kết thúc. return là lệnh không bắt buộc phải có trong thân hàm.

Ví dụ	Mô tả
return 3	Hàm trả về một giá trị, và kết thúc
return	Hàm trả về không gì cả, và kết thúc

Cú pháp : **return [danh_sach_bieu_thuc]**

Ví dụ về lệnh return so sánh hai số x,y và tìm số lớn nhất. nếu x=y thì trả về kết quả hai số bằng nhau. Nếu $x > y$ thì in ra x và ngược lại.

```
function_return.py x
1 def maximum(x, y):
2     if x > y:
3         return x
4     elif x == y:
5         return 'The numbers are equal'
6     else:
7         return y
8
9
10 print(maximum(2, 3))
```

Kết quả đầu ra: in ra giá trị lớn nhất trong hai số cho trước

```
D:\BT\hellopython\venv\Scripts\python.exe
3
Process finished with exit code 0
```

d. DocStrings trong Python

Chuỗi đầu tiên ngay sau tiêu đề hàm được gọi là docstring (documentation string), nó được dùng để giải thích chức năng cho hàm. Mặc dù docstring là không bắt buộc, nhưng việc giải thích ngắn gọn về chức năng của hàm sẽ giúp người dùng sau khi gọi hàm có thể hiểu ngay hàm sẽ làm gì mà không cần phải tìm lại định nghĩa hàm để xem xét.

Việc thêm tài liệu cho code giúp sau khi quay trở lại có thể nhớ được chi tiết, rõ ràng đoạn code đã viết trước đó mà không có sai sót gì.

Python có một tính năng tiện lợi gọi là **chuỗi tài liệu** , thường được gọi bằng tên ngắn hơn của nó **docstrings** . **DocStrings** là một công cụ quan trọng mà bạn nên sử dụng vì nó giúp ghi lại chương trình tốt hơn và dễ hiểu hơn. Chúng ta thậm chí có thể in hoặc sử dụng các chuỗi này trong các hàm.

Ví dụ về hàm DocStrings trong Python

Cho hai số nguyên x,y tìm số lớn hơn . nếu : x lớn hơn y thì in ra “x is maxium”. và ngược lại nếu y lớn hơn y thì in ra “y is maxium” và kết thúc.

```
function_docstring.py x
1 def print_max(x, y):
2     '''Prints the maximum of two numbers.
3
4     The two values must be integers.'''
5     # convert to integers, if possible
6     x = int(x)
7     y = int(y)
8
9     if x > y:
10        print(x, 'is maximum')
11    else:
12        print(y, 'is maximum')
13
14
15    print_max(3, 5)
16    print(print_max.__doc__)
```

Kết quả: in ra giá trị lớn nhất (y=5 , y >3) suy ra **y is maxium**

```
D:\BT\hellopython\venv\Scripts\python.exe
5 is maximum
Prints the maximum of two numbers.

The two values must be integers.

Process finished with exit code 0
```

CHƯƠNG II

PHÁT TRIỂN ỨNG DỤNG WEB VỚI DJANGO

Giới thiệu về web framework django

Django là một web framework khá nổi tiếng được viết hoàn toàn bằng ngôn ngữ Python. Nó là một framework với đầy đủ các thư viện, module hỗ trợ các web-developer. Mục tiêu chính của Django là đơn giản hóa việc tạo các website phức tạp có sử dụng cơ sở dữ liệu. Django tập trung vào tính năng “có thể tái sử dụng” và “có thể tự chạy”, tính năng phát triển nhanh, không làm lại những gì đã làm.

Django tập trung vào tính năng “có thể tái sử dụng” và “có thể tự chạy”, tính năng phát triển nhanh, không làm lại những gì đã làm. 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. Với sự kết hợp hoàn hảo đó chúng ta hoàn toàn có thể xây dựng một website bán hàng hay quản lý hàng hóa với độ chi tiết và chính xác cao.

Một số ưu điểm khi dùng Django là :

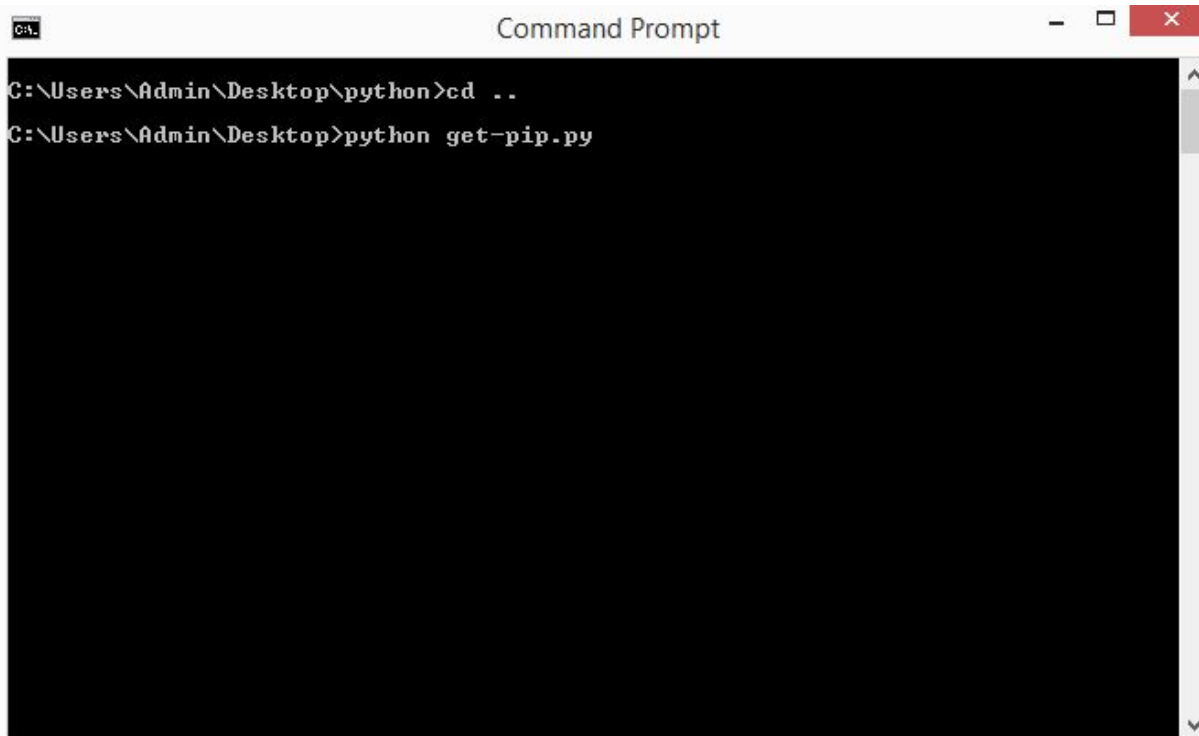
- Nhanh: Django được thiết kế với triết lý làm sao để các lập trình viên đưa các ý tưởng trở thành một sản phẩm nhanh nhất có thể.
- Có đầy đủ các thư viện/module cần thiết: Django có sẵn các thư viện về user authentication, content admin, site maps...
- Đảm bảo về tính bảo mật: Không còn các nỗi lo về các lỗi bảo mật thông thường. Django cũng cung cấp cả phương pháp để lưu mật khẩu an toàn .
- Khả năng mở rộng tốt: Django có thể đáp ứng lượng traffic lớn.

1. Cài đặt django

Django là framework giúp cho việc xây dựng các website và phát triển ứng dụng web một cách dễ dàng nhanh chóng hơn, và ít code hơn. Đầu tiên chúng ta truy cập vào trang chủ của django <https://www.djangoproject.com/download/> để

tìm hiểu và download. Đầu tiên cần tải pip về để cài đặt Django. Chúng ta tải pips tại <https://pip.pypa.io>. Tiến hành download file get-pip.py về.

Tiếp theo chúng ta bật cửa sổ CMD lên và chạy file get-pip.py..

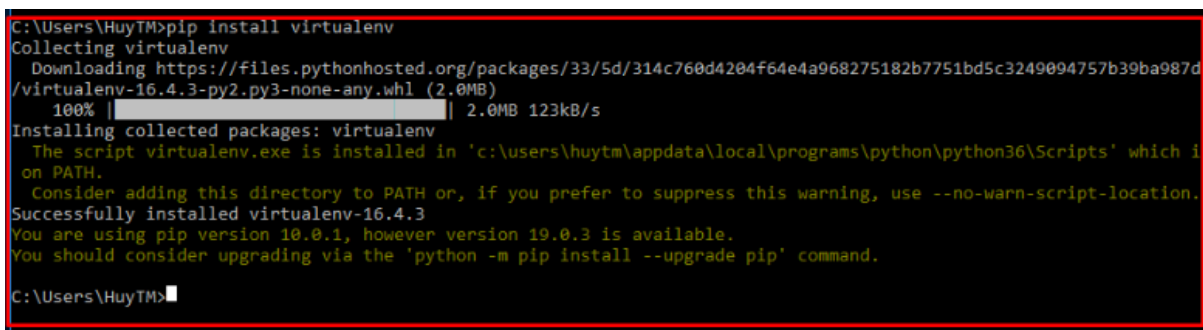


```
Command Prompt
C:\Users\Admin\Desktop\python>cd ..
C:\Users\Admin\Desktop>python get-pip.py
```

Hình 1.1 Cài đặt pip

Sau khi cài xong pips ta tiến hành download Django bằng cách gõ câu lệnh:

“*pip install Django*” để cài đặt django.



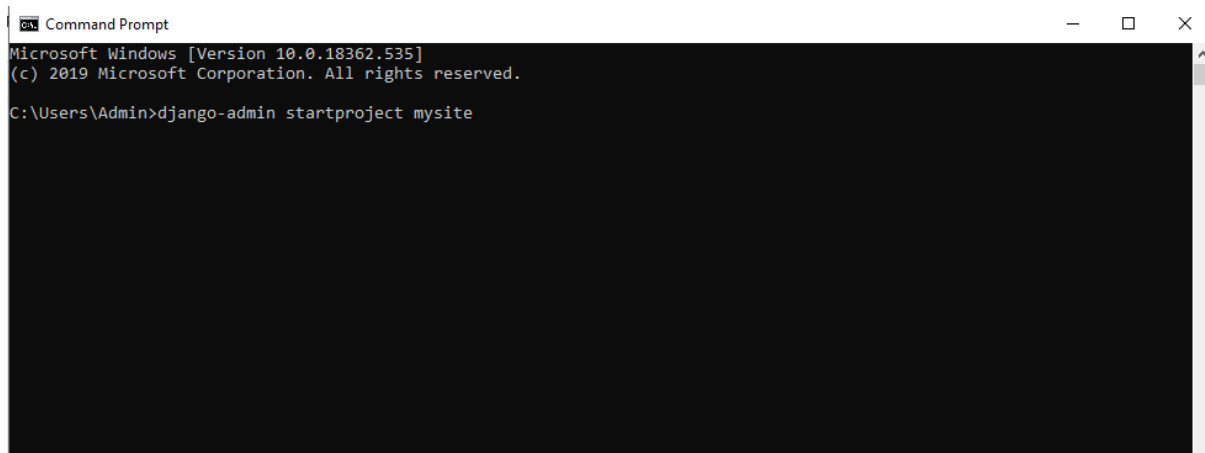
```
C:\Users\HuyTM>pip install virtualenv
Collecting virtualenv
  Downloading https://files.pythonhosted.org/packages/33/5d/314c760d4204f64e4a968275182b7751bd5c3249094757b39ba987d/virtualenv-16.4.3-py2.py3-none-any.whl (2.0MB)
    100% |#####| 2.0MB 123kB/s
Installing collected packages: virtualenv
  The script virtualenv.exe is installed in 'c:\users\huytm\appdata\local\programs\python\python36\Scripts' which is on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed virtualenv-16.4.3
You are using pip version 10.0.1, however version 19.0.3 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Users\HuyTM>
```

2. Tạo project

Bật cửa sổ CMD trên máy tính và gõ lệnh **django-admin startproject mysite**
Trong đó **django-admin startproject** là lệnh để tạo project , mysite là tên của project

Chúng ta cũng có thể sử dụng môi trường lập trình PyCharm để làm việc .Sau khi đã cài đặt được django trên máy tính, trong cửa sổ terminal của PyCharm chúng ta sẽ tiến hành tạo project bằng lệnh “**django-admin startproject mysite**”.



Hình 2.1 Tạo project

Lệnh startproject sẽ tạo một thư mục có tên là mysite, cấu trúc bên trong thư mục sẽ gồm các file , mỗi file sẽ để làm một số công việc khác nhau như chạy server hay cài đặt cấu hình server.

```
mysite/  
1  manage.py  
2  mysite/  
3  __init__.py  
4  settings.py  
5  urls.py  
6  wsgi.py
```

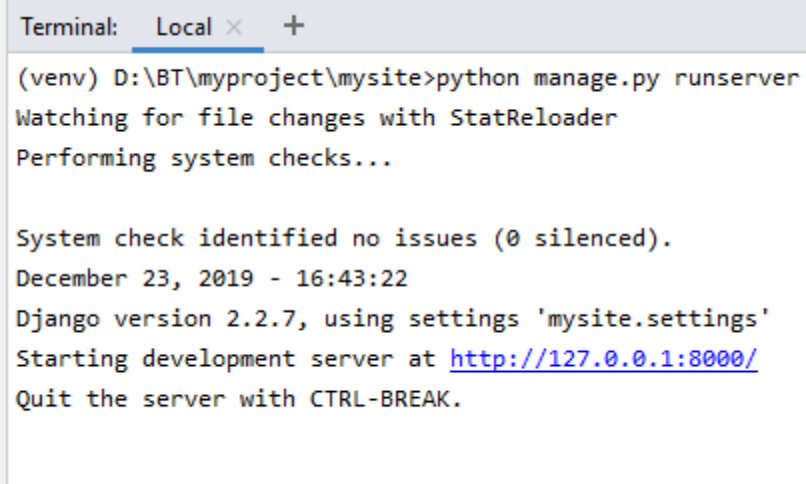
Hình 2.2 Cấu trúc project

3. Chạy server

Sử dụng môi trường lập trình pycharm. Trong cửa sổ terminal chúng ta chạy sever bằng file manage.py với tham số là runserver.

Chúng ta chuyển đến thư mục có chứa file manage.py (hoặc dùng lệnh cd [địa chỉ thư mục] và gõ lệnh:

“python manage.py runserver” để chạy server.

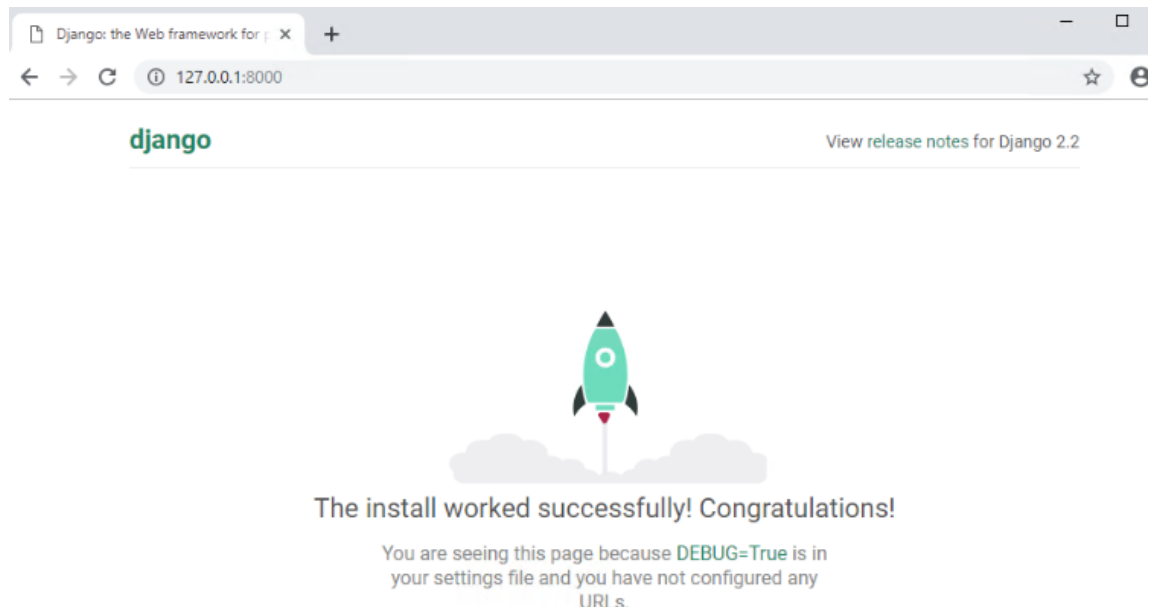


```
Terminal: Local x +
(venv) D:\BT\myproject\mysite>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
December 23, 2019 - 16:43:22
Django version 2.2.7, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

A blue arrow points to the command `python manage.py runserver` in the terminal.

Sau đó chúng ta truy cập vào đường dẫn <http://127.0.0.1:8000/> hoặc <http://localhost:8000/> để xem chúng ta đã chạy server thành công chưa.



Hình 3.1 Chạy server

4. Tạo Web App

Web App là một project bao gồm nhiều app, trong đó mỗi app thực hiện một công việc riêng biệt và thư mục này chứa các file chuẩn của một ứng dụng web Django.

Tạo web app bằng lệnh “*python manage.py startapp polls*”

Trong đó “*python manage.py startapp*” là lệnh để tạo web app. “**polls**” là tên app. Một thư mục với tên polls sẽ được tạo ra và có cấu trúc như sau:

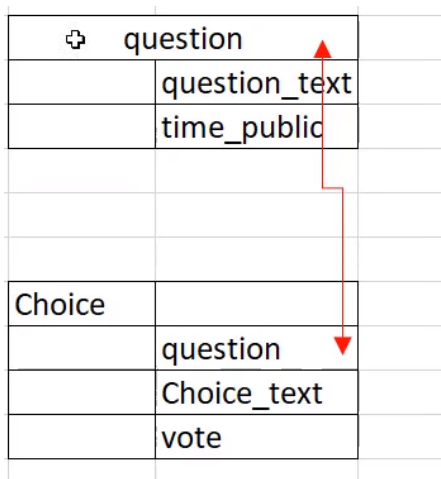
myproject\mysite\polls

```
1 | polls/
2 |     __init__.py
3 |     admin.py
4 |     apps.py
5 |     migrations/
6 |     __init__.py
7 |     models.py
8 |     tests.py
9 |     views.py
```

Hình 4.1 Cấu trúc App

5. Model

Model là phần ánh xạ giữa CSDL và python để python lấy được CSDL thông qua model.



Hình 5.1 Ví dụ CSDL

Chúng ta tiến hành xây dựng một mô hình thăm dò ý kiến đơn giản có 2 trường là Question(câu hỏi) và Choice (lựa chọn) dựa trên mô hình CSDL hình 5.1.

Trong file models.py ta viết lệnh để xây dựng một mô hình thăm dò ý kiến đơn giản với 2 class question và choice

Với trường Question có kiểu mà text với 200 kí tự và thêm định dạng thời gian

Choice câu trả lời có kiểu text và có thể tích vào câu trả lời để bình chọn (vote).


myproject\mysite\models.py

```
models.py x
1  from django.db import models
2
3  # Create your models here.
4  class Question(models.Model) :
5      question_text = models.CharField(max_length=200)
6      time_pub = models.DateTimeField()
7
8  class Choice(models.Model):
9      question = models.ForeignKey(Question, on_delete=models.CASCADE)
10     choice_text =models.CharField(max_length=100)
11     vote = models.IntegerField(default=0)
12 |
```

Sau đó ta cấu hình trong file url.py của app polls để hiển thị view thăm dò ý kiến (list) chạy với đường dẫn là <http://localhost:8000/polls/list>.

Polls/urls.py

```
urls.py x
1  from django.urls import path
2  from django.views.generic import TemplateView
3  from . import views
4
5  app_name = 'polls'
6  urlpatterns = [
7      path('detail/<int:question_id>/', views.detailView, name='detail'),
8      path('list/', views.viewlist, name='view_list'),
9      path('',views.index, name="index"),
10     path('<int:question_id>', views.vote, name='vote'..)
11 ]
12
```



Truy cập đường dẫn <http://localhost:8000/polls/list> và bình chọn cho các câu hỏi. kết quả bình chọn sẽ được gửi lên và lưu lại.



Bạn thích màu gì ?

- Đỏ
- Xanh

Đây là một ví dụ đơn giản về 1 cuộc thăm dò ý kiến với câu hỏi là “bạn thích màu gì” nếu thích màu đỏ thì tích vào ô màu đỏ và gửi, nếu thích màu xanh thì tích vào màu xanh và gửi, kết quả sẽ được gửi lên trên server và sẽ hiển thị bình chọn (VOTE) xem kết quả mà chúng ta vừa chọn.

Sau mỗi lần ấn gửi chúng ta có thể xem được số lượng bình chọn của kết quả là bao nhiêu và thực hiện vòng lặp vote tiếp hoặc kết thúc.



Bạn thích màu gì ?

Kết quả của cuộc bình chọn là:

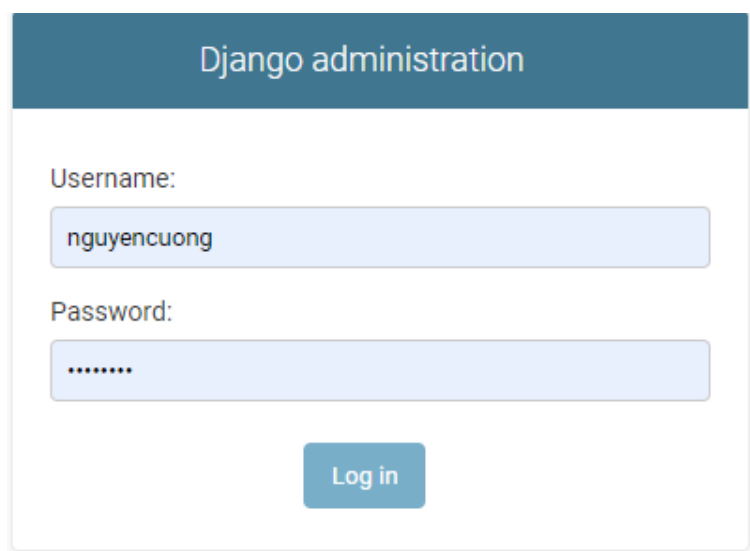
Đỏ :29

Xanh :17

[Vote tiếp](#)

6. Hệ thống admin

Khi viết một ứng dụng nào đó, chẳng hạn như website bán hàng, blog, web tin tức, diễn đàn...v.v. ngoài các trang hiển thị thông tin thì chúng ta còn phải xây dựng một trang nữa là trang admin để quản lý mọi thứ, trong đó lại bao gồm nhiều trang nhỏ hơn như thêm, sửa, xóa bài viết, cài đặt trang web. Và Django cung cấp sẵn một trang admin cho riêng chúng ta bằng cách truy cập vào đường dẫn <http://localhost:8000/admin>



Hình 6.1 Giao diện admin

Django administration

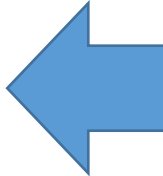
Site administration

AUTHENTICATION AND AUTHORIZATION	
Groups	+ Add Change
Users	+ Add Change

Tại trang admin chúng ta có thể thao tác với 2 bảng là User và Group. Ta sẽ thêm hai bảng Question và Choice (đã làm ở phần 5) mà chúng ta đã tạo. lí do hai bảng đó chưa được hiển thị ở đây là vì chúng ta chưa đăng ký các bảng đó với trang admin của django nên chỉ hiển thị mặc định có hai bảng user và group.

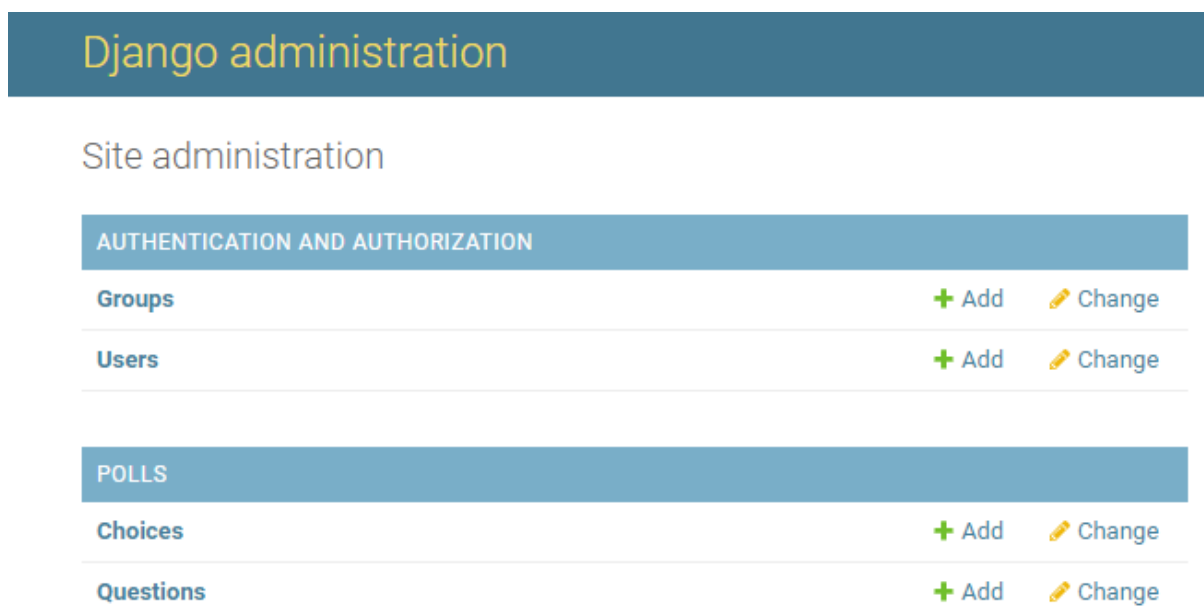
Để đăng kí các bảng (hay các mô hình) với trang admin thì chúng ta cần dùng phương thức **admin.site.register()** trong file admin.py mà Django đã tạo cho chúng ta

```
admin.py x
1 from django.contrib import admin
2 from .models import Choice, Question
3 # Register your models here.
4
5 admin.site.register(Question)
6 admin.site.register(Choice)
```

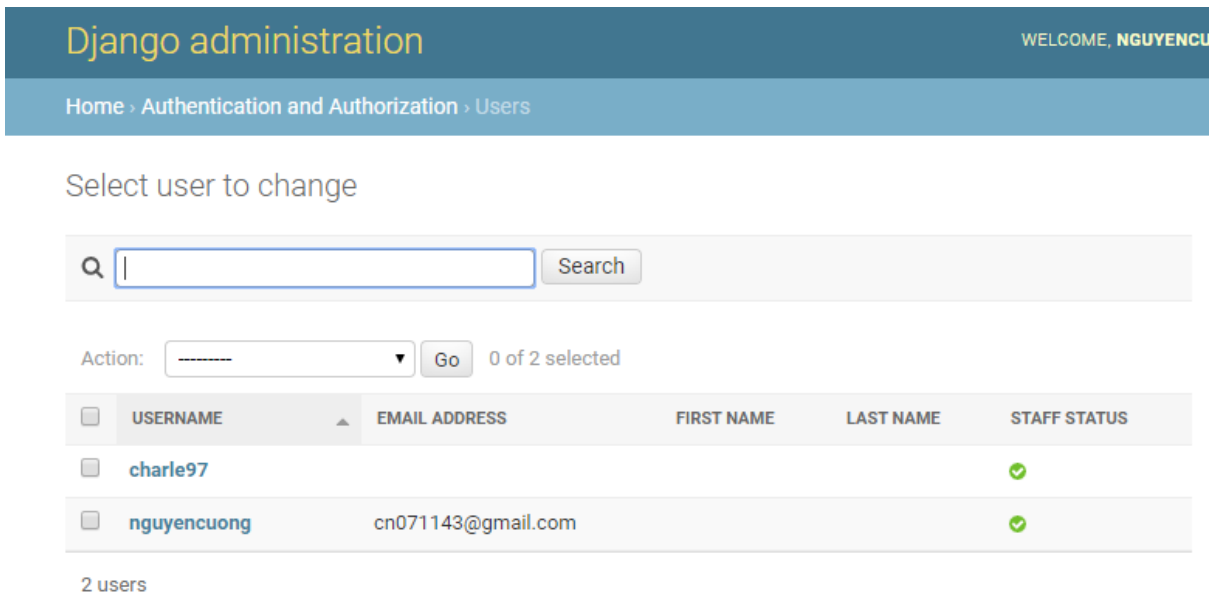


Sau khi đã đăng kí xong thì 2 bảng Question và Choice sẽ hiện ra trong giao diện admin.

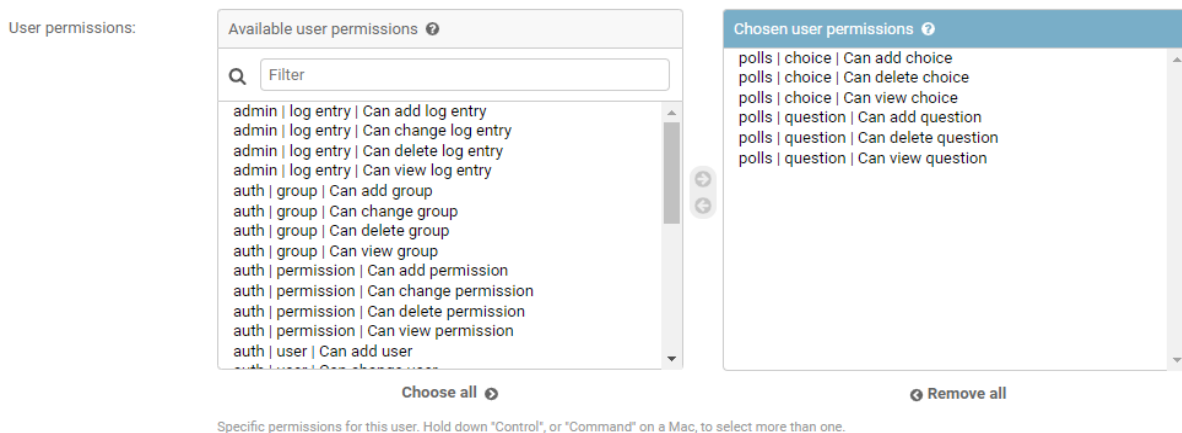
Giao diện admin mặc định của Django rất đơn giản, bạn có thể thực hiện thêm, sửa, xóa hoặc phân quyền hạn cho các user một cách dễ dàng.



Nếu chúng ta có tài khoản admin của django thì chúng ta có thể được quyền thêm, xóa người dùng, phân quyền người dùng cho các tài khoản.



Hình 6.2 Giao diện người dùng của Django



Hình 6.3 Bảng chia quyền cho user của Django

Với quyền hạn admin chúng ta có thể cấp phép, phân quyền cho những user trong quyền hạn nhất định. Như ở đây chúng ta chỉ cho người dùng có quyền được xem câu hỏi và câu trả lời của cuộc thăm dò ý kiến, tức là người dùng này không được quyền bình chọn cũng như thêm sửa xóa .

7. View và templates

a. View

Trong Django thì một View là một hàm / phương thức làm một công việc cụ thể nào đó, một view thường đi kèm với một Template.

Trong ứng dụng polls mà chúng ta đã tạo ở trên, chúng ta sẽ xây dựng các view sau:

- Index – Hiển thị các câu hỏi mới.
- Detail – Hiển thị một câu hỏi nhất định nào đó và đưa các câu trả lời để người dùng chọn.
- Result – Hiển thị kết quả bầu chọn của người dùng.
- Vote – Xử lý việc trả lời của người dùng.

Trong Django, một trang web được tạo ra bởi các hàm View, Django sẽ chọn View nào tùy thuộc vào URL mà chúng ta đã thiết lập. Có thể bạn đã từng thấy những đường dẫn URL nhìn rất “không đẹp mắt” như :

“ME2/Sites/dirmod.asp?sid=&type=gen&mod=Core+Pages&gid=A6CD4967199A42D9B65B1B” do website tự tạo ra, Django cho phép chúng ta tạo những đường dẫn dễ nhìn hơn. Để từ một đường dẫn URL đến một View thì Django sử dụng khái niệm URLConf, đây là một module Python của Django làm nhiệm vụ phân tích đường dẫn và chuyển đến một hàm View nhất định.

Chúng ta sẽ viết thêm một số hàm View trong file views.py

“polls/views.py”

```
def viewlist(request):
    list_question = Question.objects.all()
    context = {"dsquest": list_question}
    return render(request, "polls/question_list.html",
context)

def detailView(request, question_id):
    q = Question.objects.get(pk=question_id)
    return render(request, "polls/detail_question.html",
{"qs": q})


def vote(request, question_id):
    q= Question.objects.get(pk=question_id)
```

Tiếp theo chúng ta tạo thêm các url dẫn đến từng view này ở trong file urls.py

“polls/urls.py”

```
from django.urls import path
from django.views.generic import TemplateView
from . import views

app_name = 'polls'
urlpatterns = [
    path('detail/<int:question_id>/', views.detailView,
name='detail'),
    path('list/', views.viewlist, name='view_list'),
    path('', views.index, name="index"),
    path('<int:question_id>', views.vote, name='vote' )
]
```



Khi gõ địa chỉ lên thanh URL của trình duyệt, Django sẽ đọc biến `urlpatterns` trong file `mysite.urls` vì mặc định file này được trỏ tới trong biến `ROOT_URLCONF` trong file `mysite/settings.py`, các đối tượng url sẽ được đọc dần dần từ trên xuống dưới cho đến khi có một đường dẫn vừa khít với URL mà bạn nhập vào.

b. Templates

Templates là một layout được thiết kế các khung web có sẵn, ta chỉ cần thêm nội dung chính của nó vào, và nhờ các template ta mới tiết kiệm thời gian trong việc phát triển website.

Trình duyệt chỉ hiểu code HTML chứ không hiểu code Python, để có thể sử dụng code Python thì Django cung cấp cho chúng ta các thẻ template, thẻ template bắt đầu và kết thúc bằng cặp kí tự `{% %}` hoặc `{{ }}`,

Các câu lệnh Python nằm trong cặp dấu `{% %}`, còn các biến thì nằm trong cặp `{{ }}`.

Ngôn ngữ Template của Django được thiết kế với mục đích chính là hỗ trợ những người đã từng làm việc với HTML, do đó nếu đã từng học HTML thì sẽ không quá khó khăn để làm quen với Template.

Ở đây trong file `base.html` (Hình 7.1) được dùng để hiển thị những phần 'tĩnh'. Một templates đơn giản để hiển thị dòng chữ “chào mừng đến với website”....

“Polls/base.html”

```
base.html x
1  {% load static%}
2  <!DOCTYPE html>
3  <html lang="en">
4  <head>
5      <meta charset="UTF-8">
6      <title>Title</title>
7      <link rel="stylesheet" href="{% static 'polls/style.css'%}">
8  </head>
9  <body>
10     <div class="header">
11         <h1>Chào mừng đến với website</h1>
12     </div>
13     <div class="menu">
14         <ul>
15             <li>Link 1</li>
16             <li>Link 2</li>
17             <li>Link 3</li>
18             <li>Link 4</li>
19         </ul>
20     </div>
21     <div class="noidung">
22         {% block content123 %}
23         {% endblock %}
24     </div>
25
26 </body>
27 </html>
```

Hình 7.1 cấu trúc template

File index.html được kế thừa (extends) từ base.html để hiển thị các nội dung (phần động) mà chúng ta muốn hiển thị lên cũng như tương tác với nó.

```
1  {% extends 'polls/base.html' %}
2
3  {% block content123 %}
4      <h1>đây là template</h1>
5      <p> Chào các bạn mình là <b>{{ name }}</b></p>
6      <h2>Tài sản của tôi gồm: </h2>
7      <ul>
8          {% for item in taisan %}
9              <li>{{ item }}</li>
10         {% endfor %}
11     </ul>
12
13     {% endblock %}
14
```

Hình 7.2 Lệnh kế thừa

Sau đó chúng ta truy cập vào đường dẫn <http://localhost:8000/polls/> để xem kết quả cho templates chúng ta vừa viết.

Chào mừng đến với website

[Link 1](#) [Link 2](#) [Link 3](#) [Link 4](#)

đây là template

Chào các bạn mình là **nguyễn cường**

Tài sản của tôi gồm:

- Điện thoại
- Máy tính
- Xe máy
- Tiền

Nếu đã từng làm việc với các ngôn ngữ như Javascript, PHP, JSP... hay các ngôn ngữ có thể trộn chung với code HTML thì cũng nên phân biệt là Template của Django không giống các ngôn ngữ đó. Các ngôn ngữ như Javascript, PHP... là ngôn ngữ lập trình, dùng để thực hiện các công việc mang tính logic, còn HTML chỉ là ngôn ngữ đánh dấu, tức là chỉ dùng để hiển thị giao diện chứ không mang nặng phần tính toán, Template cũng vậy, đây chỉ là ngôn ngữ hỗ trợ hiển thị giao diện.

c. Đặt namespace cho URL

Khi dùng đến URL động thì lại phát sinh một vấn đề nữa, mặc định thì Django tự động tìm các file template bên trong thư mục template, vậy thì giả sử khi chúng ta có thêm nhiều ứng dụng khác ngoài polls, chẳng hạn như một ứng dụng blog, trong đó cũng có hàm view detail(), và hàm view này cũng sử dụng một template tên là detail.html, vậy thì khi đó Django sẽ gắn template của ứng dụng polls vào view detail() của ứng dụng blog, như thế sẽ báo lỗi vì ứng dụng blog sẽ không có các biến giống như polls.

Để giải quyết vấn đề này, chúng ta sẽ đặt namespace cho các biến url, chúng cũng giống như một cách gộp nhóm những thứ giống nhau lại với nhau. Để đặt tên namespace cho các đối tượng url thì chúng ta chỉ cần đặt giá trị cho biến `app_name` trong file `urls.py` là được .

“`polls/urls.py`”

```
urls.py x
1 from django.urls import path
2   from django.views.generic import TemplateView
3 from . import views
4
5 app_name = 'polls'
6 urlpatterns = [
7     path('detail/<int:question_id>/', views.detailView, name='detail'),
8     path('list/', views.viewlist, name='view_list'),
9     path('', views.index, name="index"),
10    path('<int:question_id>', views.vote, name='vote' ..)
11 ]
12
```

Trong file template, ví dụ như `detail_question.html` thì chúng ta chỉ cần viết đủ tên `<namespace>:<tên biến url>` là được, ở đây url có tên là `vote` thì chỉ cần viết đầy đủ là `polls:vote`..

“`polls/templates/polls/detail_question.html`”

```
urls.py x
1 from django.urls import path
2   from django.views.generic import TemplateView
3 from . import views
4
5 app_name = 'polls'
6 urlpatterns = [
7     path('detail/<int:question_id>/', views.detailView, name='detail'),
8     path('list/', views.viewlist, name='view_list'),
9     path('', views.index, name="index"),
10    path('<int:question_id>', views.vote, name='vote' ..)
11 ]
12
```

8. Upload file

Upload file là thực hiện chức năng đăng tải lên thư mục gốc của server.

Các hàm view mà chúng ta đã viết đều nhận một tham số đầu vào là một đối tượng `HttpRequest`, đối tượng này lưu trữ những thông tin về dữ liệu được gửi từ người dùng lên server, ví dụ như khi bạn nhập `localhost:8000/` thì trình duyệt sẽ tạo một đối tượng lưu trữ những thông tin chẳng hạn như phương thức gửi lên, địa chỉ ip, url, ngày giờ gửi... rồi nén tất cả những thông tin đó lại và gửi đến server của chúng ta qua giao thức HTTP. Khi dữ liệu được gửi đến, server Django sẽ tạo một đối tượng thuộc lớp `HttpRequest` và đưa những thông tin được gửi đến vào đối tượng này rồi truyền vào làm tham số cho hàm view tương ứng, trong hàm view đó chúng ta có thể lấy các thông tin từ đối tượng `HttpRequest` này để xử lý. Bên trong lớp `HttpRequest` có một thuộc tính tên là `FILES`, thuộc tính này lưu trữ những thông tin về file được gửi lên để chúng ta xử lý.

Khi submit file lên server, data file sẽ nằm trong `request.FILES`. Điều bắt buộc cho HTML form phải có thuộc tính `enctype="multipart/form-data"` nếu không `request.FILES` sẽ trống.

Form cần được submit với **POST** method.

Django có các loại mẫu thích hợp để xử lý upload files: `FileField` và `ImageField`. Các loại file được upload `FileField` or `ImageField` sẽ không được lưu trữ trong database nhưng sẽ được lưu trữ ở `fileSystem`.

`FileField` và `ImageField` được tạo như String field trong database(thông thường là `varchar`), mục đích để ánh xạ tới file thực tế.

a. Tạo form upload

Đầu tiên ta tạo một form để upload file ,chúng ta tạo một app mới với tên là `file_uploader` bằng lệnh : **`python manage.py startapp file_uploader`**

App này nhằm mục đích giúp chúng ta tải file lên server và lưu trữ ở `fileSystem`

Sau đó khai báo app **`file_uploader`** trong phần **`install app`** ở file **`setting.py`**.

```
INSTALLED_APPS = [  
    'polls.apps.PollsConfig',  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'file_uploader',  
    'pagination',  
]
```

Hình 8.1 Khai báo App

b. Tạo templates và file template

Tạo một file html và đặt tên là **fileUploadrTemplate** sau đó tạo thư mục templates và file template: Để gửi file lên thì trong thẻ <form> chúng ta phải khai báo thuộc tính **enctype="multipart/form-data"**.

“file_uploader/templates/fileUploaderTemplate.html”

```
fileUploaderTemplate.html x  
1 <form action="" method="POST" enctype="multipart/form-data">  
2     {% csrf_token %}  
3     <table>  
4         {{ form.as_table }}  
5     </table>  
6     <input type="submit" value="Submit" />  
7 </form>
```

Hình 8.2 Khai báo thuộc tính

Sau khi đã có template và form để hiển thị thì chúng ta tiến hành tạo view để kết nối chúng với nhau. Chúng ta viết hàm **fileUploaderView()** và hàm xử lý việc ghi file là **upload()**.

If form.is_valid(): Phương thức `is_valid()` có trong lớp `django.forms.Form` sẽ kiểm tra tính hợp lệ của dữ liệu được gửi lên. Mục đích chính của lớp Form trong Django chính là hỗ trợ chúng ta kiểm tra sự đúng đắn của dữ liệu được gửi lên. Ở đây phương thức `is_valid()` chỉ có chức năng đơn giản là kiểm tra xem ô text title có dữ liệu hay không vì mặc định các field của Form trong Django bắt buộc phải có dữ liệu mới được nhận.

“file_uploader/views.py”

```
views.py x
1  from django.shortcuts import render
2  from django.db import models
3
4  # Create your models here.
5  from django.http import HttpResponse
6  from .forms import UploadFileForm
7
8
9  def fileUploaderView(request):
10     if request.method == 'POST':
11         form = UploadFileForm(request.POST, request.FILES)
12         if form.is_valid():
13             upload(request.FILES['file'])
14             return HttpResponse("<h2>File uploaded successful!</h2>")
15         else:
16             return HttpResponse("<h2>File uploaded not successful!</h2>")
17
18     form = UploadFileForm()
19     return render(request, 'fileUploaderTemplate.html', {'form': form})
20
21
22 def upload(f):
23     file = open(f.name, 'wb+')
24     for chunk in f.chunks():
25         file.write(chunk)
26 # Create your views here.
```

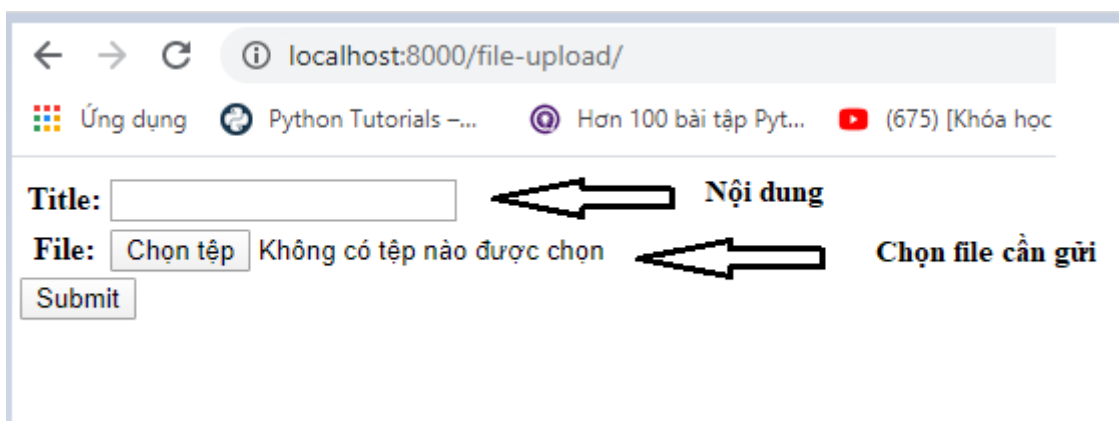
Hình 8.3 Kiểm tra tính hợp lệ

Cuối cùng chúng ta định nghĩa các url, tạo url trở tới hàm view trong app.

“file_uploader/urls.py”

```
urls.py x
1  from django.conf.urls import url
2
3  from . import views
4
5  urlpatterns = [
6     url(r'^$', views.fileUploaderView),
7 ]
```

Bây giờ chúng ta có thể chạy server và truy cập url “<http://localhost:8000/file-upload>” để thực hiện upload file.



Hình 8.4 Giao diện upload file.

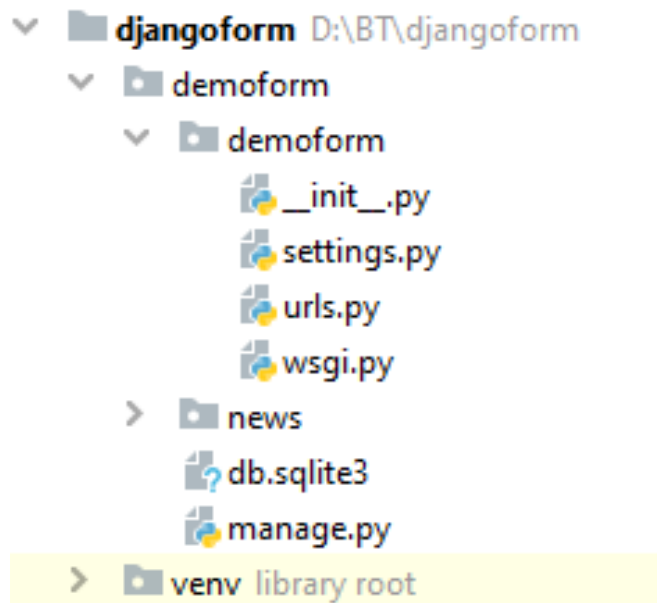
9. Form trong django

a. Form sử dụng model

Trong HTML thì form là một tập các thẻ element nằm giữa cặp thẻ `<form>...</form>` cho phép người dùng thực hiện một công việc nào đó, sau đó gửi dữ liệu lên server rồi trả về câu trả lời. Hầu hết HTML chỉ hỗ trợ các thẻ element cơ bản như text, checkbox... muốn hiển thị các element cao cấp hơn như DateTimePicker (chọn ngày tháng), Slider (thanh trượt)... phải dùng đến các ngôn ngữ hỗ trợ thêm khác như Javascript, CSS... Hoặc có thể dùng lớp Form có sẵn của Django, Django cung cấp lớp Form hỗ trợ tạo form một cách nhanh chóng và dễ dàng.

Dùng form để nhập dữ liệu sau đó đổ dữ liệu vào trong model thế nên dùng model form sẽ tiết kiệm thời gian hơn. ở đây chúng ta sẽ làm riêng 1 project để làm về form, cụ thể ở đây chúng ta sẽ thiết kế form để nhập dữ liệu sau đó gửi lên server và lưu lại.

Ở đây chúng ta tạo mới một project có tên **djangoform** để ví dụ về phần model form, với chức năng đơn giản nhập dữ liệu (đăng bài viết hoặc tin tức) gửi lên server và lưu lại.. Trong project chúng ta tạo thêm một app có tên là “**news**” bằng lệnh **python manage.py startapp news**



Hình 9.1 Tạo project

Sau đó chúng ta thêm app “news” vừa tạo vào phần install app ở trong file settings.py.

“demoform/settings.py”

```
INSTALLED_APPS = [  
    'news',  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
]
```

Trong mỗi app có những phần gọi là router, nó sẽ định tuyến từng đường link ở trong từng app nên ta sẽ tạo một file.py trong app news có tên là “urls” và import vào file views.

“new/urls.py”

```
1 from django.urls import path
2 from . import views
3 urlpatterns = [
4     path('', views.index, name='index' ),
5 ]
6
```

Hình 9.2 Import app định tuyến.

Trước khi làm form thì trong **models.py** ta tạo một class tên là “Post” để định nghĩa bài viết gồm có:

title : là tiêu đề của bài viết với tối đa là 255 kí tự và nó không được rỗng.

content : chính là phần nội dung của bài viết với 1000 kí tự và cũng không được trống.

time_create : thời gian khởi tạo với định dạng chính là giờ hiện tại mà chúng ta thêm bài viên vào(**default=timezone.datetime.now()**).

“news/models.py”

```
models.py x
1 from django.db import models
2 from django.utils import timezone
3 # Create your models here.
4
5
6 class Post(models.Model):
7     title = models.CharField(max_length=255, blank=False, null=False)
8     content = models.TextField(max_length=1000, blank=False, null=False)
9     time_create = models.DateTimeField(default=timezone.datetime.now())
10
11 def __str__(self):
12     return self.title
```

Hình 9.3 Tạo class.

Sau đó chúng ta tạo một file có tên là **forms.py** để nhập dữ liệu cho class vừa tạo và post nó lên và lưu vào trong database. Thêm các trường để hiển thị ra bên ngoài forms (tiêu đề bài viết, nội dung, thời gian khởi tạo).

```
forms.py x
1 from django import forms
2   from .models import Post
3   from django.template.defaulttags import widthratio
4
5   from .models import Post
6
7   class PostForm(forms.ModelForm):
8       class Meta:
9           model = Post
10          fields = ('title', 'content', 'time_create',)
11          widgets = {
12              'title': forms.TextInput(attrs={'class': 'tieude123'}),
13              'content': forms.Textarea(attrs={'class': 'noidungbt'})
14          }
```

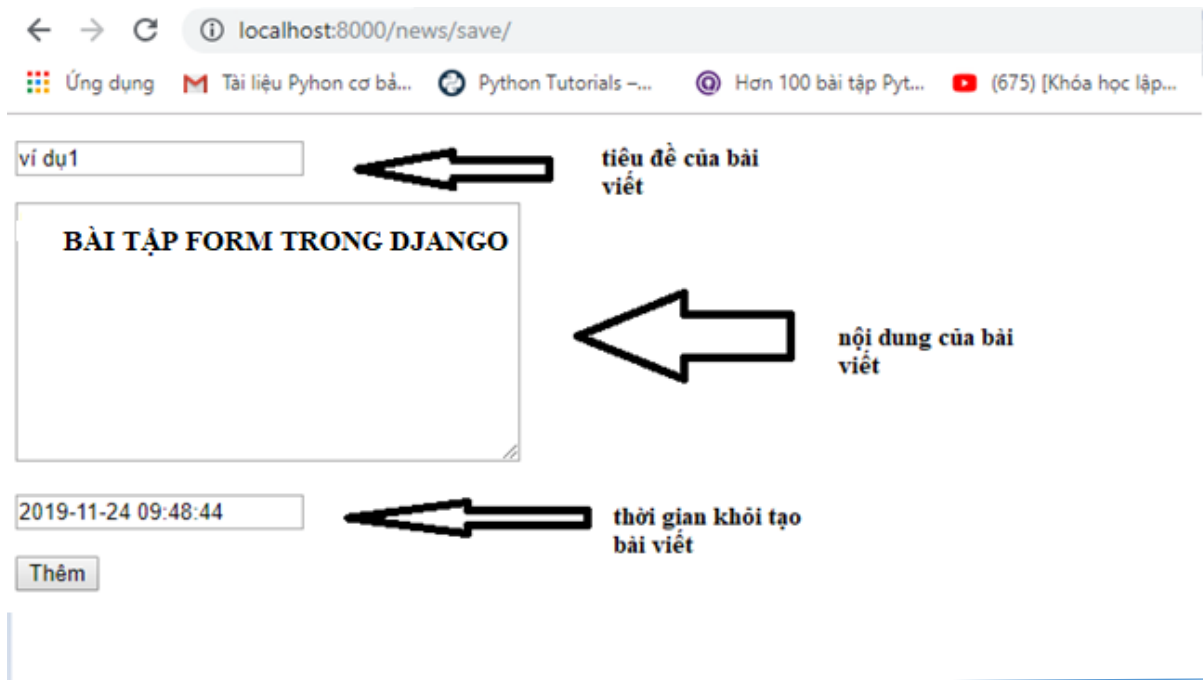
Hình 9.4 Tạo form nhập dữ liệu.

Sau đó chúng ta tạo một templates có tên “**news**” với một file **add_news.html** để in ra một số thứ như nội dung, tiêu đề, thời gian và một nút submit.

“*news/templates/news/add_news.html*”

```
add_news.html x
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Title</title>
6 </head>
7 <body>
8 <form action="{% url 'news:save' %}" method="post">
9     <p> {{ f.title }}</p>
10    <p> {{ f.content }}</p>
11    <p>{{ f.time_create }}</p>
12    {% csrf_token %}
13    <input type="submit" value="Thêm">
14 </form>
15 </body>
16 </html>
```

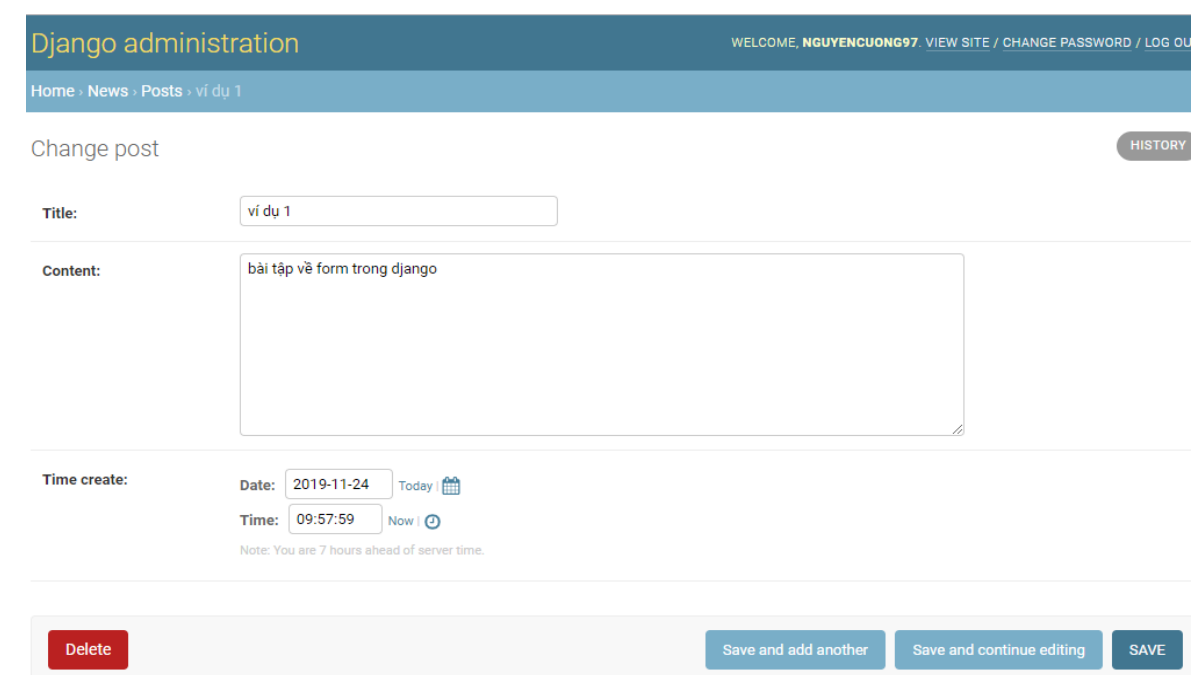
Kết quả sau khi truy cập vào đường dẫn <http://localhost:8000/news/add/> đã khai báo ở trong file **url.py**



Đây chính là một form đơn giản để nhập và gửi dữ liệu lên trên server. Và để xem bài viết đã được thêm vào hay chưa, chúng ta hãy truy cập tài khoản admin (chúng ta sẽ tạo một tài khoản admin bằng lệnh :**python manage.py createsuperuser** [tên tài khoản,mật khẩu,emai]]

Createsuperuser là lệnh để tạo một tài khoản admin (supperuser) với đầy đủ các quyền hạn.

Sau khi đăng nhập vào trang admin thì ở đây bài viết đã được hiển thị trong server và với tài khoản admin của django chúng ta có thể thêm, sửa, xóa bài viết ngay tại trang admin của django.



Và bài viết đã được hiển thị trong phần admin của django. Như vậy chỉ cần chúng ta thêm bài viết hay nội dung ở trên đường dẫn dẫn <http://localhost:8000/news/add/> thì sẽ tự động được thêm trên trang admin.

b. Form không sử dụng model

Ví dụ : tạo một form đơn giản để gửi email (sendemail) bao gồm :

- Tiêu đề
- Nội dung
- Tên email
- Một nút submit

Tạo một class “**SendEmail**” với các trường tiêu đề, tên email, nội dung email sử dụng hàm **EmailField**.

“**EmailField**” để phân biệt xem đây có phải là định dạng mặc định của một email hay không.



```
1 from django import forms
2 from .models import Post
3 from django.template.defaulttags import widthratio
4
5
6
7 class SendEmail(forms.Form):
8     title = forms.CharField(max_length=100, widget=forms.TextInput(attrs={'class': 'tieude'}))
9     email = forms.EmailField()
10    content = forms.CharField(widget=forms.Textarea(attrs={'value': 'cuongnguyen', 'id': 'noidung'}))
11    cc = forms.BooleanField(required=False)
```

Hình 9.5 Tạo form gửi email.

Sau đó tạo một views có tên là “email_view”

```
def email_view(request):
    b = SendEmail()
    return render(request, 'news/email.html', {'f': b})
```

Và một templates có tên là “**email.html**”

```
email.html x
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <title>Title</title>
6 </head>
7 <body>
8 <style>
9 .cuongnguyen{
10 padding: 1px;
11 }
12 </style>
13 <form action="{% url 'news:pro' %}" method="post">
14 {{ f }}
15 {% csrf_token %}
16 <input type="submit" value="Gửi mail">
17 </form>
18 </body>
19 </html>
```

Sau đó thêm vào đường dẫn trong file **urls.py** với tên truy cập là: **name =email**.

“Django/demoform/news/urls.py”

```
urls.py x
1 from django.urls import path
2 from . import views
3 app_name = "news"
4 urlpatterns = [
5     path('add/', views.add_post, name="add"),
6     path('', views.IndexClass.as_view(), name='index'),
7     path('save/', views.ClassSaveNews.as_view(), name="save"),
8     path('email/', views.email_view, name='email'),
9     path('process/', views.process, name='pro'), #process xử lý
10 ]
11 |
```

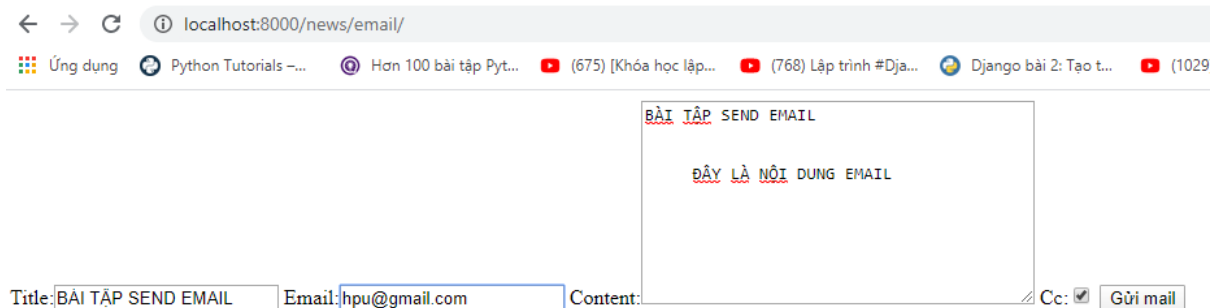
Cuối cùng runserver bằng lệnh “**python manage.py runserver**” và xem kết quả:

Truy cập vào: “<http://localhost:8000/news/email/>”

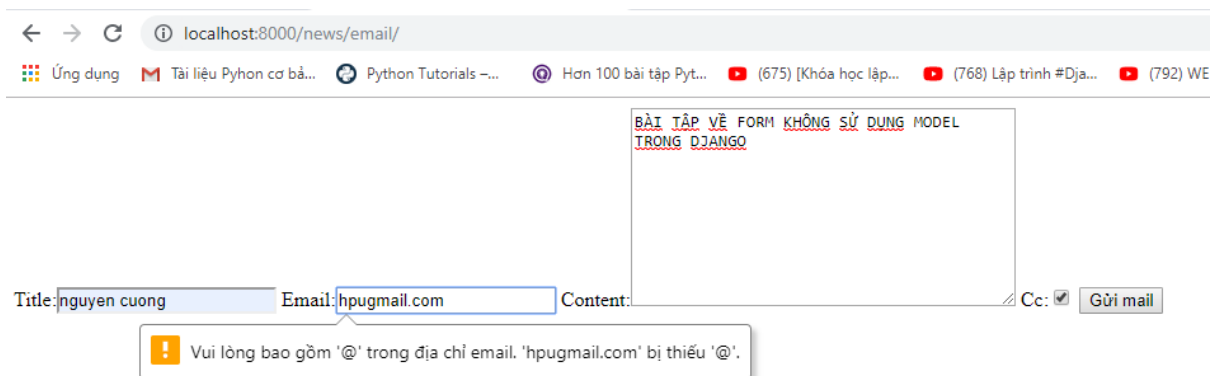
Với trường “**title**” và “**content**” là dạng input nhập vào dữ liệu nhưng trường “**Email**” ở đây chúng ta đã cho nó với định dạng “EmailField” tức là chúng

ta phải nhập đúng định dạng của một email (ví dụ : abc@gmail.com) nếu không nó sẽ báo lỗi. dưới đây là 2 trường hợp hiển thị đúng và sai định dnagj của một email

Hiển thị đúng định dạng.



Lỗi sai định dạng email :ở đây hệ thống thông báo lỗi là bị thiếu “@”, không giống định dạng của một email.



Ở trong file **urls.py** chúng ta sẽ tạo thêm một view hiển thị nội dung chúng ta vừa gửi lên với tên là process (xử lí).



Và tạo luôn một templates để hiển thị views chúng ta vừa tạo và đặt tên là **print_email.html**. và chúng ta thiết kế để hiển thị dưới dạng bảng (table).
news/print_email.html”.

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Title</title>
6      <style>
7          td{
8              border: 1px solid #000;
9          }
10     </style>
11 </head>
12 <body>
13     <table>
14         <tr>
15             <td>Tiêu đề</td>
16             <td>Nội dung</td>
17             <td>Email</td>
18             <td>CC</td>
19         </tr>
20         <tr>
21             <td>{{ email_data.title }}</td>
22             <td>{{ email_data.email }}</td>
23             <td>{{ email_data.content }}</td>
24             <td>{{ email_data.cc }}</td>
25         </tr>
26     </table>
27 </body>
28 </html>

```

Và truy cập lại vào <http://localhost:8000/news/email/> Kết quả là sẽ hiển thị những nội dung chúng ta vừa nhập và in ra..

Tiêu đề	Nội dung	Email	CC
		BÀI TẬP VỀ FORM KHÔNG SỬ DỤNG MODEL TRONG DJANGO	
nguyen cuong	hpu@gmail.com		<input checked="" type="checkbox"/>

10. Hệ thống user trong Django

User là gì?

User ở trong hệ thống mặc định của django là một bảng gồm các user chung với nhau. Admin thì cũng là một user và những tài khoản khác cũng là một user và sự khác nhau ở đây chính là sự phân quyền.

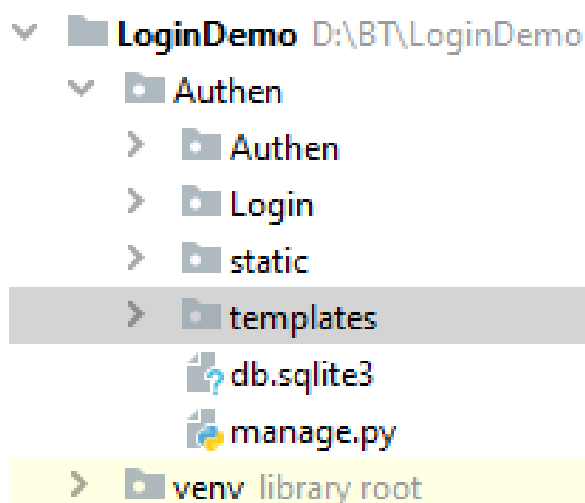
Chẳng hạn user1 là quản lý thì có thể quản lý các nhân viên (user2, user3, user4..) và phân cho những user khác những quyền hạn nhất định có thể giống hoặc khác nhau và cũng có thể thu hồi lại, và ở đây chúng ta sẽ làm rõ sự phân quyền ấy là như thế nào.

a. Ví dụ về phân quyền user

Chúng ta tạo một project để ví dụ **user model** và đặt tên project là **“LoginDemo”**.

Sau đó sẽ tạo thêm project tên là **“Authen”** bằng lệnh **“django-admin startproject Authen”**.

Trong **“Authen”** chúng ta tạo thêm một app có tên là **“Login”** bằng lệnh **“python manage.py startapp Login”**.



Trong file **views.py** cấu hình đơn giản để hiển thị view với dòng chữ xin chào.

Tiếp theo cấu hình đường link với tên truy cập là login (**name=login**)

“LoginDemo/Login/views.py”

```
views.py x
1 from django.shortcuts import render, HttpResponseRedirect
2 from django.views import View
3 # Create your views here.
4
5
6 class IndexClass(View):
7     def get(self, request):
8         return HttpResponseRedirect('<h1>Xin chao</h1>')
9
```

“LoginDemo/Login/urls”

```
LoginDemo D:\BT\LoginDemo
├── Authen
│   ├── Authen
│   │   ├── __init__.py
│   │   ├── settings.py
│   │   ├── urls.py
│   │   └── wsgi.py
│   └── Login
│       └── migrations
└── ...
1 from django.urls import path
2 from .views import IndexClass
3
4
5
6 app_name='Login'
7 urlpatterns = [
8     path('', IndexClass.as_view(), name='index_'),
9     path('login/', LoginClass.as_view(), name='login'),
10 ]
```

Hình 10.1 Cấu hình tên truy cập

Tiếp theo chúng ta import thêm một hàm để xác thực người dùng là hàm **authenticate**.

“**from django.contrib.auth import authenticate**”.

Tạo một tài khoản admin bằng lệnh “**python manage.py createsuperuser [...]**”

Ở đây chúng a sẽ thiết kế một form đăng nhập đơn giản để trả về kết quả nếu đã là user thì báo về **đăng nhập thành công**, ngược lại thì trả về kết quả **đăng nhập thất bại**.

“**Logindemo/Login/views.py**”

```

views.py x
1 from django.shortcuts import render, HttpResponseRedirect
2 from django.views import View
3 from django.contrib.auth import authenticate, login, decorators
4 from django.contrib.auth.mixins import LoginRequiredMixin # mixins dùng để kết nối cá đối tượng 1 cách mềm dẻo
5 from .forms import PostForm
6 # Create your views here.
7
8
9 class IndexClass(View):
10     def get(self,request):
11         return HttpResponseRedirect('<h1>Xin Chào, đây là bài tập User model trong django</h1>')
12
13
14 class LoginClass(View):
15     def get(self, request):
16         return render(request, template_name='Login/login.html')
17
18     def post(self, request):
19         user_name = request.POST.get('tendangnhap')
20         matkhau = request.POST.get('password')
21         my_user = authenticate(username=user_name, password=matkhau)
22         if my_user is None:
23             return HttpResponseRedirect('Đăng nhập thất bại User không tồn tại')# nếu tk k tồn tại thì trả về user k tồn tại
24         login(request, my_user)
25         return render(request, 'Login/thanhcong.html')
26

```

Hình 10.2 Form trả kết quả đăng nhập

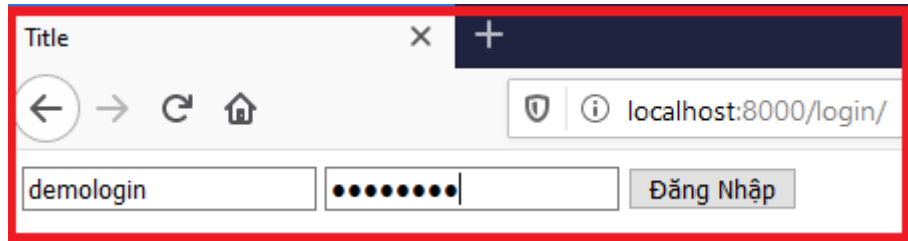
Tiếp theo tạo một file template tên là “**login.html**” để hiển thị thông tin đăng nhập

```

login.html x
1 <!DOCTYPE html>
2 <html lang="en" xmlns="http://www.w3.org/1999/html">
3 <head>
4     <meta charset="UTF-8">
5     <title>Title</title>
6 </head>
7 <body>
8     <div class="Login">
9         <form action="" method="post">
10             {% csrf_token %}
11             <input type="text" name="tendangnhap">
12             <input type="password" name="password">
13             <input type="submit" value="Đăng Nhập">
14         </form>
15     </div>
16 </body>
17 </html>

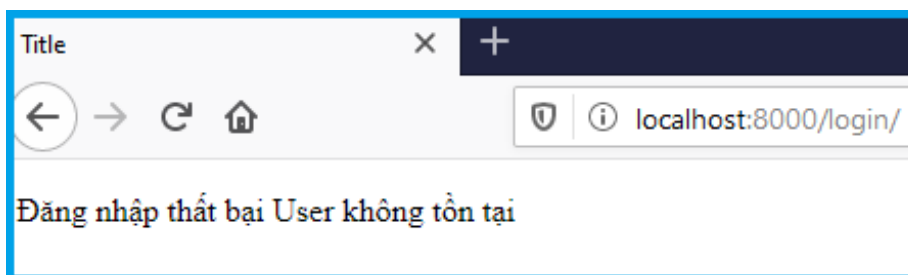
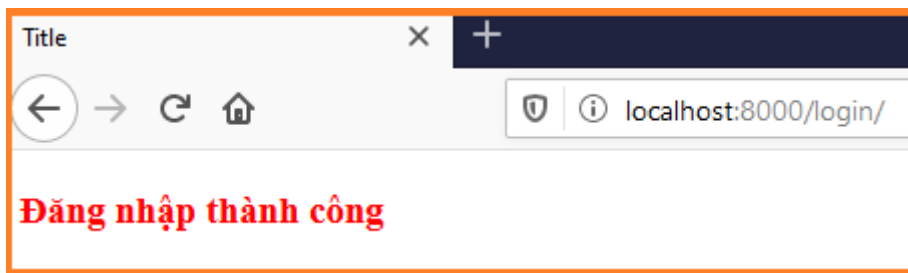
```

Cuối cùng truy cập vào “<http://localhost:8000/login/>” bằng tài khoản admin chúng ta đã tạo sẵn.



Hình 10.3 Đăng nhập

Nếu đăng nhập đúng với tk admin thì sẽ hiển thị đăng nhập thành công và ngược lại sẽ báo đăng nhập thất bại (không phải là một user trong hệ thống).



b. Phân quyền view (decorator)

Ví dụ ta có một vài sản phẩm mới ra mắt mà không muốn người ngoài biết được thì bắt buộc ta phải có một “hàng rào”, và ở đây ta sẽ làm một ví dụ bắt buộc phải đăng nhập thì mới được xem thông tin của một view đã thiết lập “hàng rào”.

Trong file **views.py** tạo một view có tên là “**view_product**” và sẽ thiết lập “hàng rào” sao cho chỉ khi đăng nhập mới có thể xem được view này bằng hàm **decorator**.

Login/views.py”

```
def view_product(request):
    return HttpResponse('ĐT Việt Nam đã thắng 6-0 trước U22 Brunei trong trận mở màn SEA Games 31
        'Bên cạnh đó họ cũng sẽ quan tâm tới kết quả trận U22 Thái Lan - U22 Indi
        'Hai đội tuyển này được đánh giá là những đối thủ chính của U22 Việt Nam
        'Thực lực của U22 Việt Nam lẫn U22 Thái Lan và U22 Indonesia là không qu
        'Giả định rằng U22 Việt Nam toàn thắng trước U22 Brunei, U22 Lào, U22 Si
```

Trước tiên trong file **views.py** ta import vào một “**decorators**”.

Hàm “**@decorators.login_required(login_url='/login/')**” giúp ta thực hiện hành động bắt buộc phải đăng nhập(login) thì mới được xem view.

“**from django.contrib.auth import decorators**”

```
@decorators.login_required(login_url='/login/') #bắt buộc phải đăng nhập trước khi xem
def view_product(request):
    return HttpResponse('ĐT Việt Nam đã thắng 6-0 trước U22 Brunei trong trận mở màn
        'Bên cạnh đó họ cũng sẽ quan tâm tới kết quả trận U22 Thái Lan
        'Hai đội tuyển này được đánh giá là những đối thủ chính của U
        'Thực lực của U22 Việt Nam lẫn U22 Thái Lan và U22 Indonesia
        'Giả định rằng U22 Việt Nam toàn thắng trước U22 Brunei, U22
```

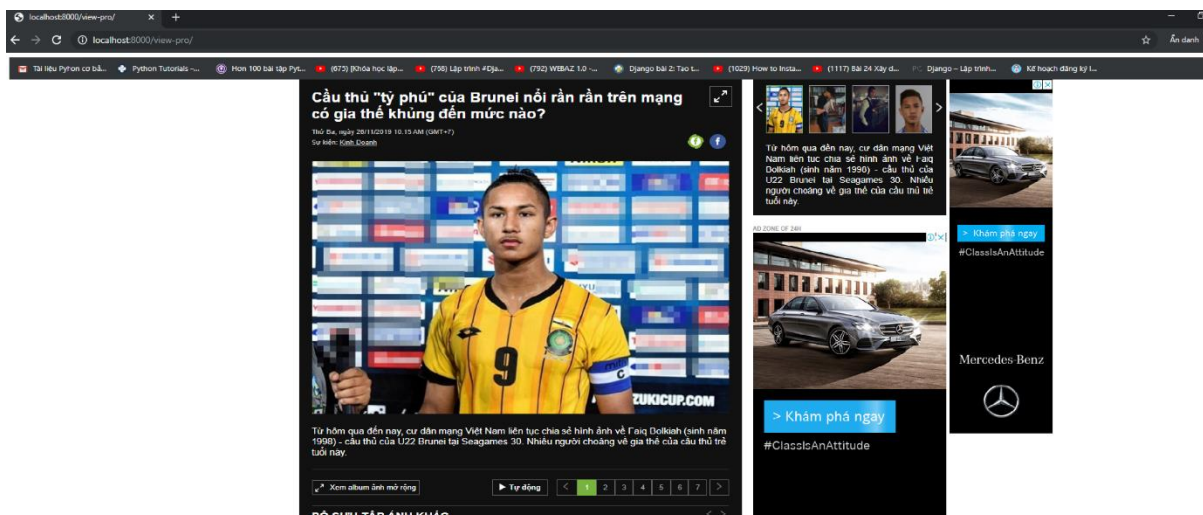
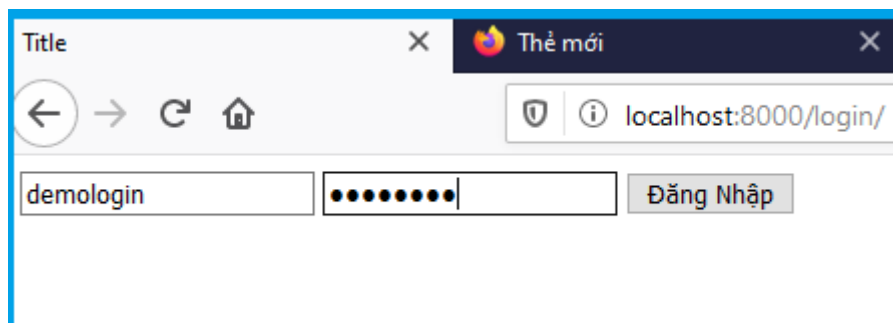
Cuối cùng trong file **url.py** chỉ cần khai báo nó với tên “**view-pro**” để chạy với đường dẫn “<http://localhost:8000/view-pro/>”.

```
urls.py x
1 from django.urls import path
2 from .views import IndexClass, LoginClass, ViewUser, view_product, AddPost
3
4
5
6 app_name='Login'
7 urlpatterns = [
8     path('', IndexClass.as_view(), name='index'),
9     path('login/', LoginClass.as_view(), name='login'),
10    path('user-view/', ViewUser.as_view(), name='user_view'),
11    path('view-pro/', view_product, name='view-product'),
12    path('add-post/', AddPost.as_view(), name='add_post'),
13 ]
14
```

Hình 10.4 Khai báo đường dẫn

Sau khi truy cập <http://localhost:8000/view-pro/> đã hiển thị giao diện đăng nhập.

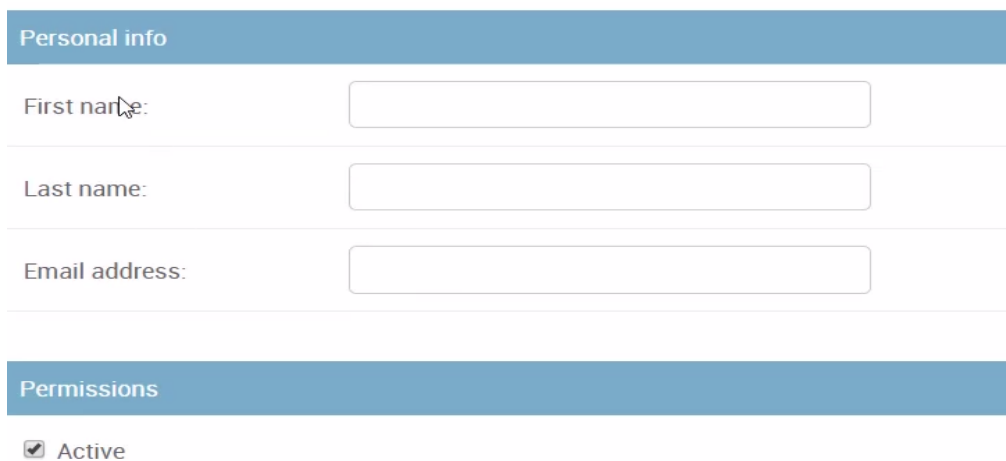
Bởi vì chưa đăng nhập nên ở đây chưa hiển thị thông tin của view_product. Và chúng ta hãy đăng nhập bằng tài khoản của django admin.



Sau khi đăng nhập đã hiện thông tin của “view_product”. Vậy là chúng ta đã thực hiện xong việc phân quyền với quyền chỉ được xem khi đăng nhập bằng “@decorators.login_required(login_url='/login/')”.

11. Custom user model trong Django

Phần user của django khá là đơn giản và nó chỉ có một vài thuộc tính cơ bản như first name, last name, địa chỉ email... mà trong thực tế chúng ta có thể cần yêu cầu cao hơn một chút như tuổi, giới tính và địa chỉ...



The image shows a Django user profile form. It has two main sections: 'Personal info' and 'Permissions'. The 'Personal info' section contains three input fields: 'First name', 'Last name', and 'Email address'. The 'Permissions' section contains a checkbox labeled 'Active' which is checked.

Để thêm các trường, thuộc tính mới trong user thì đầu tiên trong file **model.py** chúng ta cần phải import vào **AbstractUser**

“**AbstractUser**” là một class cơ sở để chúng ta có thể kế thừa và định nghĩa thêm một số các thuộc tính của user.

Tiếp theo chúng ta tạo thêm một class và đặt tên là “**Myuser**” kế thừa từ “**AbstractUser**”. Sau đó chúng ta thêm một số thuộc tính như là tuổi(age), giới tính(sex) và địa chỉ(address). Trong file **models.py** giới tính sẽ được để theo kiểu **choice** (lựa chọn) để có thể chọn giới tính là nam hoặc nữ hay là không xác định.

models.py”

```
models.py x
1 from django.db import models
2 from django.contrib.auth.models import AbstractUser
3 # Create your models here.
4
5
6
7
8
9 class MyUser(AbstractUser): #thêm lựa chọn giới tính cho user, địa chỉ
10     sex_choice = ((0, 'Nữ'), (1, 'Nam'), (2, 'không xác định'))
11     age = models.IntegerField(default=0)
12     sex = models.IntegerField(choices=sex_choice, default=0)
13     address = models.CharField(default='', max_length=255)
```

Tiếp theo trong file **admin.py** đăng kí view **Myuser** để hiển thị trong giao diện quản trị ,sau đó Import “**MyUser**” và dùng “**admin.site.register(MyUser)**” để đăng kí user.

admin.py”.

```
admin.py x
1 from django.contrib import admin
2 from .models import Post, MyUser
3 # Register your models here.
4 admin.site.register(Post)
5 admin.site.register(MyUser)
6
```

Sau đó chỉ việc chạy server (**python manage.py runserver**) , truy cập vào địa trang admin và vào phần user để kiểm tra thuộc tính đã được thêm hay chưa.

Và ở trong phần user đã có thêm các thuộc tính như tuổi, giới tính và địa chỉ rồi.

Username:
Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

First name:

Last name:

Email address:

Staff status
Designates whether the user can log into this admin site.

Active
Designates whether this user should be treated as active. Unselect this instead of deleting accounts.

Date joined: **Date:** Today | 📅
Time: Now | 🕒
Note: You are 7 hours ahead of server time.

Age: ▾

Sex: ▾

Address:

12. Tùy chỉnh giao diện admin (admin custom admin site django)

Là thay đổi giao diện của trang admin ví dụ như thay đổi màu sắc hay từ tiếng anh thành tiếng việt cũng như bố cục của trang admin nhằm giúp trở nên đẹp mắt và thuận tiện hơn.

Đầu tiên chúng ta thêm dòng lệnh

```
“STATICFILES_DIRS=[  
    os.path.join(BASE_DIR, "static"), ]”
```

vào trong file “setting.py”.

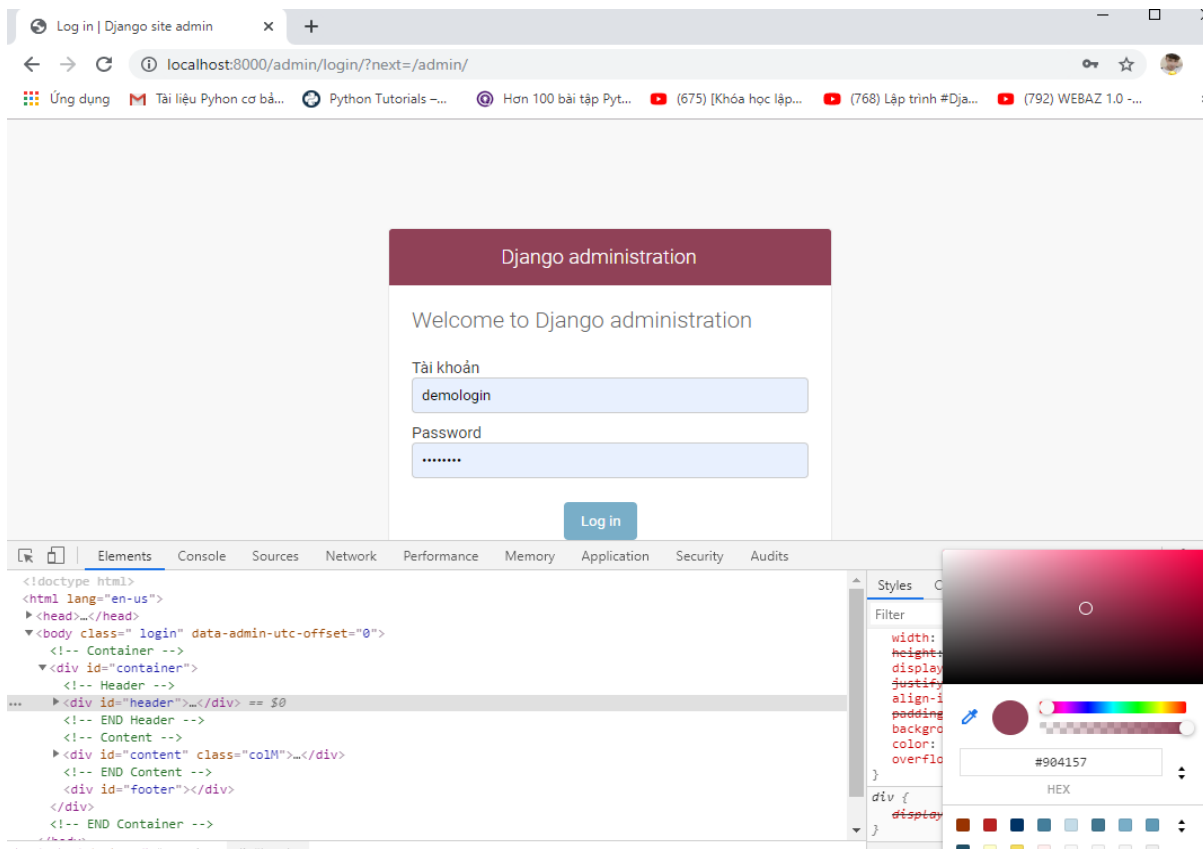
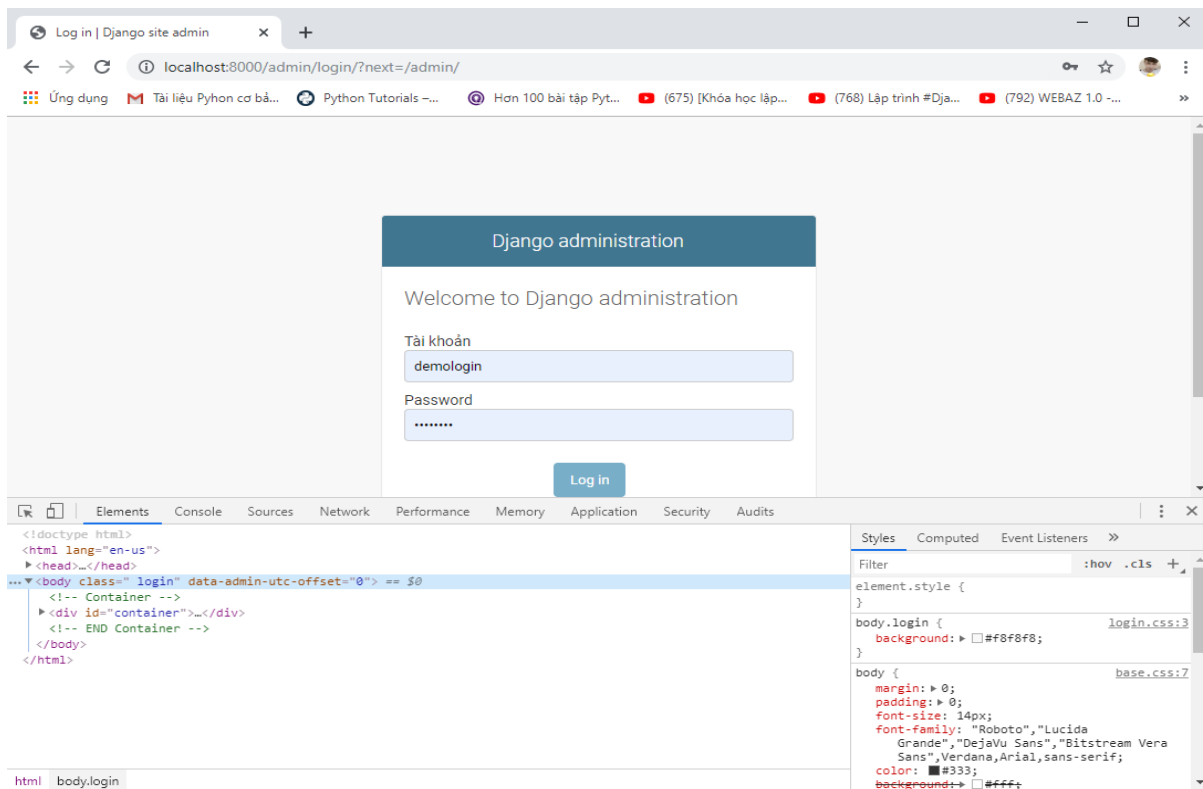
```
STATIC_URL = '/static/'  
  
STATICFILES_DIRS = [  
    os.path.join(BASE_DIR, "static"),  
]  
|
```

Hình 12.1 Thêm static

sau đó chúng ta chạy câu lệnh “**python manage.py collectstatic**” để lấy 119 file để có thể thay đổi giao diện admin

```
119 static files copied to  
'D:\BT>LoginDemo\Authen\static'
```

tiếp theo chúng ta truy cập vào trang admin và dùng cách ấn “**F12**” có thể chỉnh sửa màu sắc đơn giản.



Hình 12.2 Thay đổi màu giao diện

Ngoài ra chúng ta cũng có thể chỉnh sửa giao diện login admin bằng cách vào file “**login.html**” để thay đổi cũng như viết hóa một số thứ trong giao diện admin

login.html”.

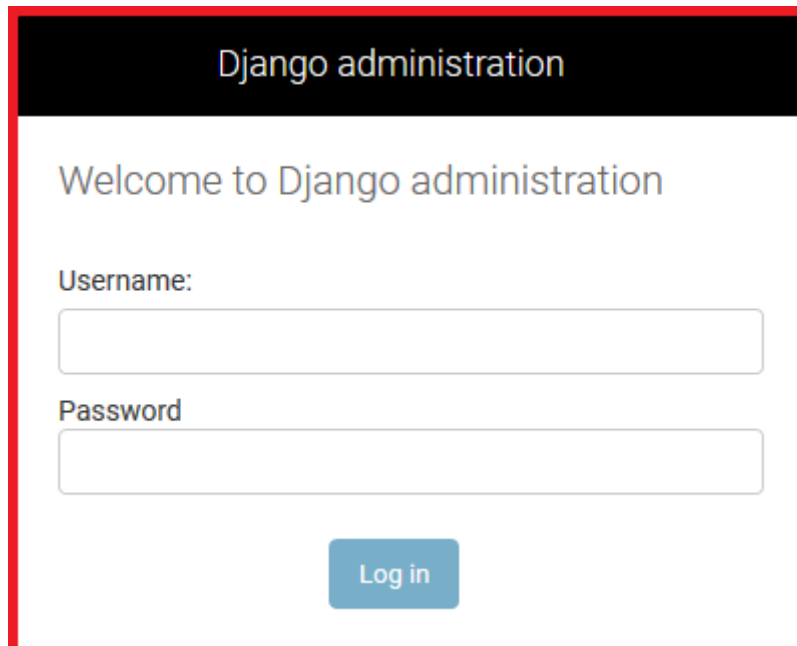
```
<form action="{{ app_path }}" method="post" id="login-form">{% csrf_token %}
  <div class="form-row">
    {{ form.username.errors }}
    {{form.username.label_tag}} {{ form.username }}
  </div>
  <div class="form-row">
    {{ form.password.errors }}
    Password {{ form.password }}
    <input type="hidden" name="next" value="{{ next }}">
  </div>
  {% url 'admin_password_reset' as password_reset_url %}
  {% if password_reset_url %}
  <div class="password-reset-link">
    <a href="{{ password_reset_url }}">{% trans 'Forgotten your password or username?' %}</a>
  </div>
  {% endif %}
```

Thường thì giao diện của django sẽ sử dụng ngôn ngữ là tiếng anh. Chúng ta sẽ viết hóa bằng cách thay đổi một số nội dung ở trong file **login.html**.

Trong file **login.html** , thay đổi nội dung của thẻ user name và thẻ pass word để hiện thị tiếng việt.

```
<form action="{{ app_path }}" method="post" id="login-form">{% csrf_token %}
  <div class="form-row">
    {{ form.username.errors }}
    Tài khoản đăng nhập {{ form.username }}
  </div>
  <div class="form-row">
    {{ form.password.errors }}
    Mời bạn nhập mật khẩu {{ form.password }}
    <input type="hidden" name="next" value="{{ next }}">
  </div>
  {% url 'admin_password_reset' as password_reset_url %}
  {% if password_reset_url %}
  <div class="password-reset-link">
    <a href="{{ password_reset_url }}">{% trans 'Forgotten your password or username?' %}</a>
  </div>
  {% endif %}
```

Truy cập <http://localhost:8000/admin> username và password đã được thay đổi, ta cũng có thể làm điều tương tự với các thành phần khác.



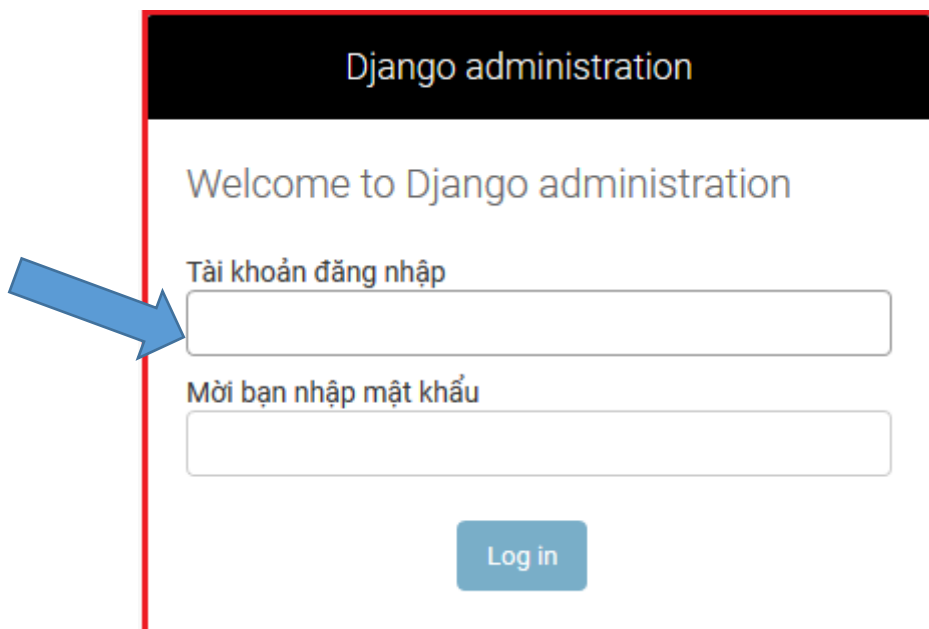
Django administration

Welcome to Django administration

Username:

Password

Log in



Django administration

Welcome to Django administration

Tài khoản đăng nhập

Mời bạn nhập mật khẩu

Log in

Chương III

XÂY DỰNG KHUNG WEB SITE BÁN HÀNG

Giới thiệu

Đối với mỗi doanh nghiệp, tổ chức, cửa hàng hay cá nhân, dù đã và đang kinh doanh online, hay bắt đầu kinh doanh online thì đều hiểu rằng một trang website bán hàng chuyên nghiệp có tầm quan trọng như thế nào trong việc quyết định đến hiệu quả kinh doanh online.

Những lưu ý khi thiết kế website bán hàng :

- Giao diện website phải phù hợp với mặt hàng kinh doanh, đây là điều hết sức quan trọng, vì mỗi một mặt hàng kinh doanh sẽ có giao diện riêng, tuy rằng cùng là website bán hàng, nhưng không phải mặt hàng nào cũng sử dụng giao diện lung tung được, ví dụ như bạn bán mỹ phẩm, không thể nào bạn lại đi sử dụng giao diện của bán hàng thiết bị điện tử được. Giao diện website rất quan trọng, vì khi khách hàng của bạn vừa truy cập vào trang website bán hàng của bạn, họ sẽ không đi sâu vào bên trong ngay, hay sử dụng những tính năng trên đó ngay, mà sẽ nhìn ngay ngoài trang chủ trước. Đối với những trang website bán hàng, bạn nên thiết kế website với giao diện 1100px, 1200px, hoặc full màn hình, vì đây là những kích thước giúp website của bạn có không gian trở nên rộng rãi hơn cho việc trưng bày giới thiệu sản phẩm, cũng như bố trí được các banner sự kiện, chương trình khuyến mãi, nhưng vẫn làm nổi bật lên thứ quan trọng nhất đó là sản phẩm mà bạn đang kinh doanh.

- Một trang website bán hàng muốn để lại ấn tượng đối với khách hàng tốt nhất bạn nên sử dụng loại hình thiết kế website chuyên nghiệp, vì lúc đó giao diện website của bạn sẽ được thiết kế mới hoàn toàn dựa theo thương hiệu hiện có của bạn như logo, dải màu thương hiệu, lĩnh vực kinh doanh... khi đó thương hiệu của bạn sẽ đồng nhất cả ngoài thực tế và trên website, giúp website của bạn để lại ấn tượng đối với khách hàng truy cập.

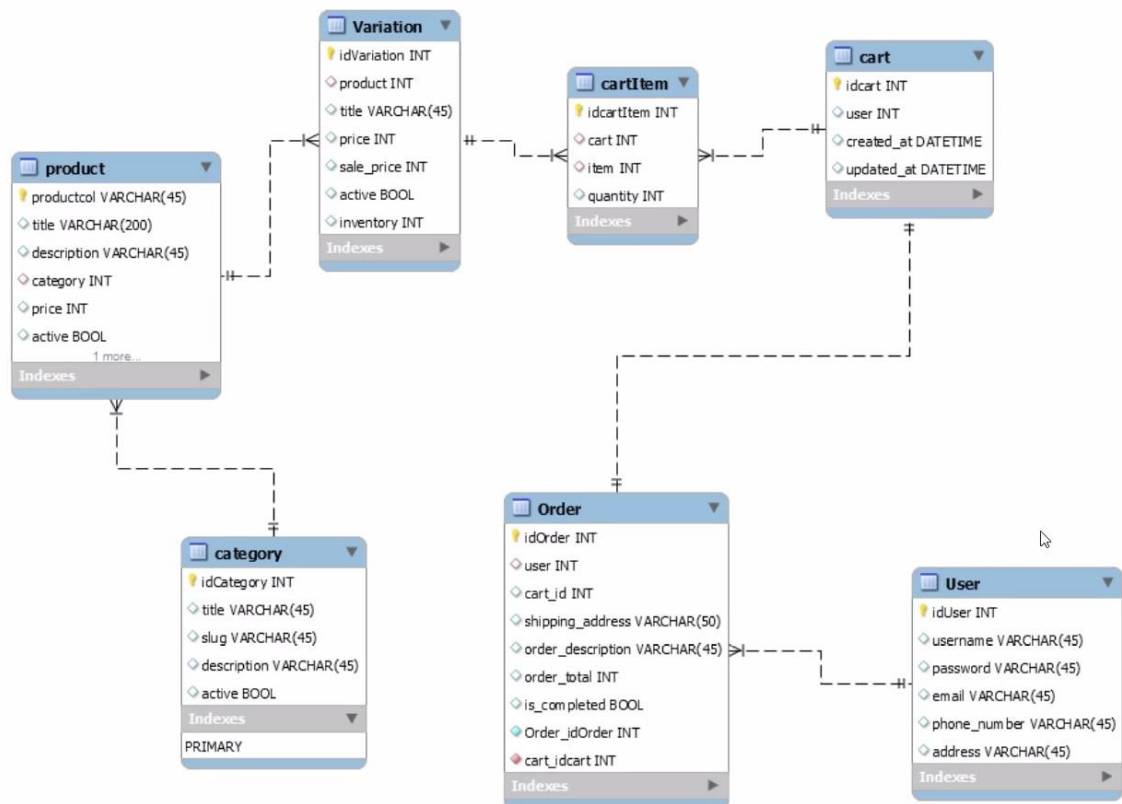
- Giao diện, bố cục website phải phù hợp với đối tượng sử dụng đây là điều mà gần như những người lần đầu thiết kế website thường hay mắc phải nhất, ví dụ như nếu mặt hàng của bạn đa phần là người lớn tuổi thì bạn không nên thiết kế bố cục quá phức tạp sẽ khiến người dùng khó tiếp cận được sản phẩm mà họ mong muốn, hay để đặt mua được sản phẩm.

- Chức năng website phù hợp với nhu cầu sử dụng, đối với những website thiết kế theo module có sẵn thông thường bạn sẽ nhận được một trang website với đầy đủ tính năng của một trang website bán hàng, tuy nhiên đối với website thiết kế chuyên nghiệp thì không như vậy, các đơn vị thiết kế website chuyên nghiệp sẽ xây dựng từng tính năng riêng phù hợp với nhu cầu sử dụng thực sự của bạn

- Kết nối các công cụ mạng xã hội Mạng xã hội hiện nay rất phát triển, đây cũng là một kênh giúp bán hàng rất hiệu quả, điều này bất cứ ai trong kinh doanh online đều biết đến, trước khi thiết kế website chắc hẳn bạn đã sở hữu cho mình fanpage bán hàng hay zalo bán hàng... vì vậy, khi thiết kế website bán hàng hãy lưu ý rằng website của bạn phải được kết nối với những kênh mạng xã hội này, vì đây là cổng để chuyển khách hàng của bạn từ mạng xã hội dần về website.

Kết hợp những kiến thức đã làm được ở phần I và II ta sẽ tiến hành xây dựng một trang bán hàng đơn giản

1. Phân tích cơ sở dữ liệu



Hình 1.1 Cơ sở dữ liệu bán hàng

Trong một website bán hàng thì ngoài việc có một giao diện thân thiện dễ sử dụng và tiếp cận tốt với mục đích của khách hàng thì ngoài ra chúng ta phải chú tâm đến sản phẩm mà chúng ta bán.

Product trong bảng product gồm có tiêu đề sản phẩm, mô tả sản phẩm, loại sản phẩm, giá tiền và trạng thái.

Variation để quản lí những sự thay đổi của sản phẩm ví dụ như giá của sản phẩm theo từng ngày hoặc vào những ngày giảm giá.

Cartitem là bảng giỏ hàng, một giỏ hàng có thể có nhiều hàng hóa, và một sản phẩm có thể được đặt bởi nhiều giỏ hàng khác nhau.

Order lấy thông tin của khách hàng cũng như để giao hàng, trạng thái của đơn hàng như thế nào.

User hệ thống khách hàng.

2. Xây dựng khung website bán hàng

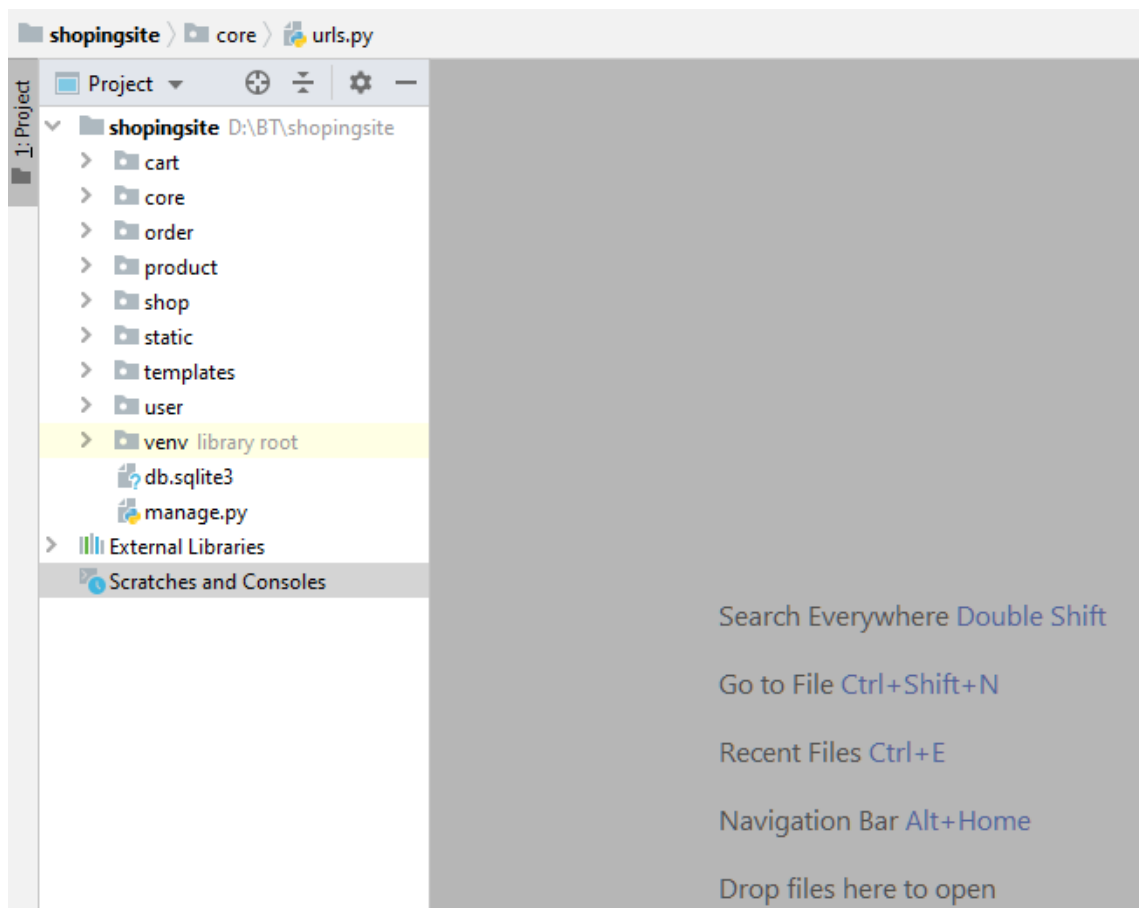
Chúng ta sẽ tạo một project tên là **Shopingsite** để làm ví dụ về một web bán hàng đơn giản dựa theo mô hình trên

Đầu tiên chúng ta tạo các app tương ứng là những bảng trong Hình 1.1 bằng lệnh :

```
django-admin startapp [tên app]
```

ở đây chúng ta có 6 app :

- Cart
- Core
- Order
- Product
- Shop
- User



Hình 2.1 Tạo project bán hàng

Và thêm những app vừa tạo vào phần **install app** trong file “**setting.py**”

```
settings.py x
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'cart',
41     'product',
42     'order',
43     'user',
44     'core',
45 ]
```

Hình 2.2 Install App

Tiếp theo chúng ta phải xây dựng model cho từng app mà chúng ta vừa tạo ra.

Trong app **product** khai báo những class có các bảng có liên quan đến sản phẩm và những thành phần bên trong bảng **product** ở trong file **model.py**

Ở đây chúng ta sẽ khai báo 3 class liên quan đến sản phẩm là:

- **category** (loại sản phẩm)
- **product** (sản phẩm)
- **variation** (quản lý những sự biến đổi của sản phẩm)

và định nghĩa kiểu từng thành phần, từng thuộc tính trong từng bảng trong file **model.py**.

“product/model.py”

```
models.py x
1  from django.db import models
2
3  # Create your models here.
4
5
6  class Category(models.Model):
7      title = models.CharField(default='', max_length=100)
8      slug = models.CharField(max_length=100, default='')
9      description = models.TextField(default='')
10     active = models.BooleanField(default=True)
11
12
13     class Product(models.Model):
14         title = models.CharField(max_length=255, default='')
15         description = models.TextField(default='')
16         category = models.ForeignKey(Category, on_delete=models.CASCADE)
17         product_img = models.CharField(max_length=255, default='')
18         price = models.IntegerField(default=0)
19         active = models.BooleanField(default=True)
20
21
22     class Variation(models.Model):
23         product = models.ForeignKey(Product, on_delete=models.CASCADE)
24         title = models.CharField(max_length=255, default='')
25         price = models.IntegerField(default=0) #Giá
26         sale_price = models.IntegerField(default=0) #giá sale
27         inventory = models.IntegerField(default=0) #hàng tồn kho
28         active = models.BooleanField(default=True)
29
```

Hình 2.3 Khai báo class

Tiếp theo xây dựng model trong app **cart** (giỏ hàng) khai báo 2 class liên quan đến giỏ hàng là:

- **Cart**
- **CartItem**

Có trường **auto_now** và **auto_now_add** là dùng để xác thực thời gian mà chúng ta thêm vào giỏ hàng chính là thời điểm hiện tại và thời gian chúng ta update thêm giỏ hàng.

cart/model.py

```
models.py x
1  from django.db import models
2  from product.models import Variation
3  from user.models import CustomerUser
4  # Create your models here.
5
6  class Cart(models.Model):
7      user = models.ForeignKey(CustomerUser, on_delete=models.CASCADE)
8      created_at = models.DateTimeField(auto_now_add=True)
9      updated_at = models.DateTimeField(auto_now=True)
10
11
12  class CartItem(models.Model):
13      cart = models.ForeignKey(Cart, on_delete=models.CASCADE)
14      item = models.ForeignKey(Variation, on_delete=models.CASCADE)
15      quantity = models.IntegerField(default=0)
16
```

Hình 2.4 Xây dựng model

Tiếp theo xây dựng model người dùng (khách hàng) trong bảng **user** khai báo class **CustomerUser** và thêm trường địa chỉ và số điện thoại khách hàng.

```
models.py x
1 from django.db import models
2 from django.contrib.auth.models import AbstractUser
3 # Create your models here.
4
5
6 class CustomerUser(AbstractUser):
7     phone_number = models.CharField(default='', max_length=15)
8     address = models.CharField(default='', max_length=255)
9
10
```

Cuối cùng chúng ta cũng xây dựng model trong bảng **order** với những trường:

- **user**(khách hàng)
- **cart**(giỏ hàng)
- **shipping_address** (địa chỉ giao hàng)
- **is_complete** (đơn hàng đã được giao hay chưa).

“order/model.py”

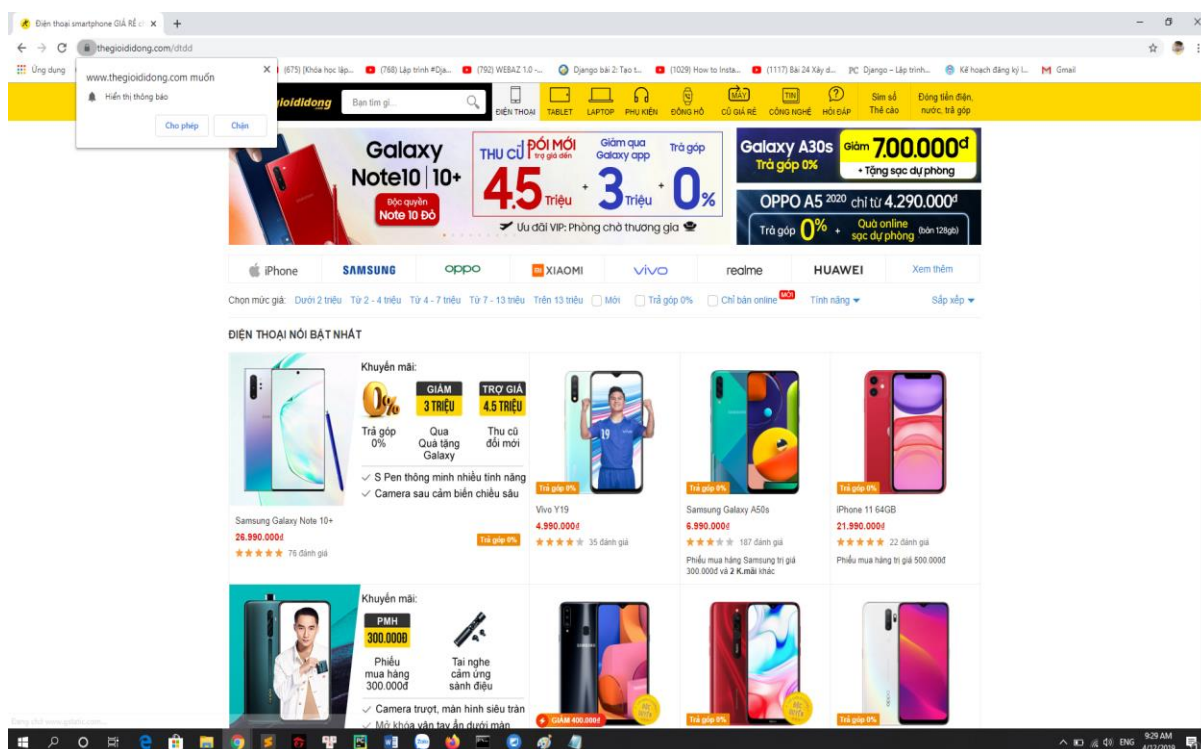
```
models.py x
1 from django.db import models
2 from user.models import CustomerUser
3 from cart.models import Cart
4 # Create your models here.
5
6
7 class Order(models.Model):
8     user = models.ForeignKey(CustomerUser, on_delete=models.CASCADE)
9     cart = models.ForeignKey(Cart, on_delete=models.CASCADE)
10    shipping_address = models.CharField(max_length=255, default='')
11    order_description = models.TextField(default='')
12    is_completed = models.BooleanField(default=False)
13
14
```

Vậy là đã xong phần xây dựng model cho một website bán hàng với những gì cần thiết cho một web bán hàng.

Tiếp theo thì chúng ta cần trang trí lại trang web sao cho đẹp mắt làm hấp dẫn khách hàng với những thay đổi trang trí cũng như các khuyến mại hay tiện ích thì chúng ta sẽ thêm những template vào.

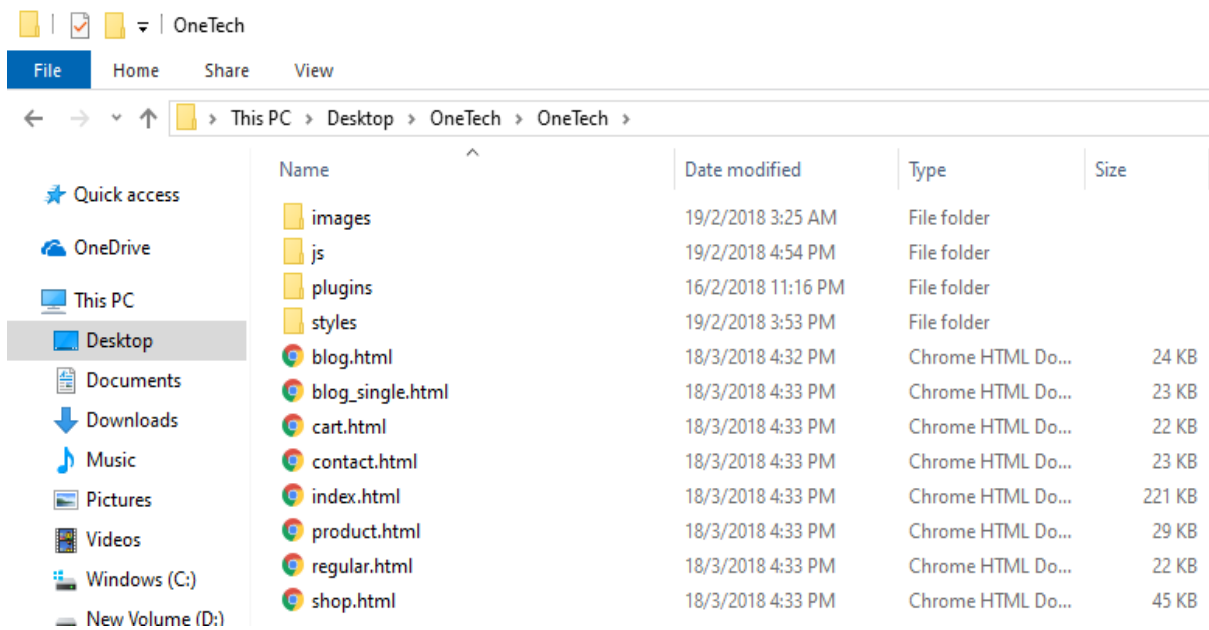
Ta có một ví dụ với website của thế giới di động với đầy đủ những thông tin sản phẩm, giá, loại sản phẩm và trang trí vô cùng bắt mắt, giúp người dùng có thể tiếp cận nhanh chóng đến sản phẩm mà họ cần tìm chỉ qua một click chuột.

Chúng ta cũng sẽ dùng template có sẵn để làm trang web trở lên hoàn thiện hơn.



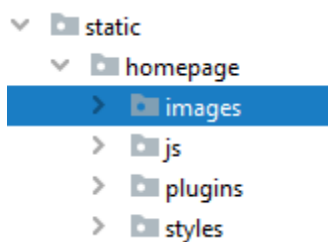
Hình 2.5 Ví dụ về website của thế giới di động

Đầu tiên chúng ta chuẩn bị một template có sẵn.
(nguồn : <https://colorlib.com/wp/template/onetech/>)

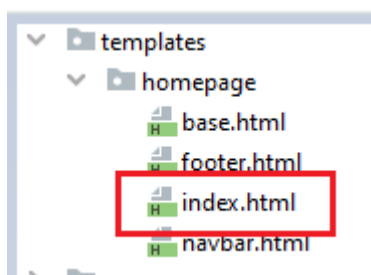


Hình 2.6 Download template

Tiếp theo trong project ta tạo thêm hai folder có tên là **“static”** và **“templates”** để chứa những template. Và copy toàn bộ file trong template đã chuẩn bị sẵn vào file **static**.



Trong template mà chúng ta chuẩn bị trước có một file tên là **index.html** ta sẽ copy file đó vào trong folder **templates** của project.



Sau đó ta tiến hành load toàn bộ static ở trong file **“index.html”**.


```
1  {% extends 'homepage/base.html' %}
2  {% load static %}
3
4  {% block cssblock %}
5  <link rel="stylesheet" type="text/css" href="{% static 'homepage/styles/bootstrap4/bootstrap.min.css' %}">
6  <link href="{% static 'homepage/plugins/fontawesome-free-5.0.1/css/fontawesome-all.css' %}" rel="stylesheet" type="text/css">
7  <link rel="stylesheet" type="text/css" href="{% static 'homepage/plugins/OwlCarousel2-2.2.1/owl.carousel.css' %}">
8  <link rel="stylesheet" type="text/css" href="{% static 'homepage/plugins/OwlCarousel2-2.2.1/owl.theme.default.css' %}">
9  <link rel="stylesheet" type="text/css" href="{% static 'homepage/plugins/OwlCarousel2-2.2.1/animate.css' %}">
10 <link rel="stylesheet" type="text/css" href="{% static 'homepage/plugins/slick-1.8.0/slick.css' %}">
11 <link rel="stylesheet" type="text/css" href="{% static 'homepage/styles/main_styles.css' %}">
12 <link rel="stylesheet" type="text/css" href="{% static 'homepage/styles/responsive.css' %}">
13 {% endblock %}
14
15 {% block content %}
16 <div class="super_container">
17     <!-- head der-->
18
19     <div class="banner">
20         <div class="banner_background" style="..."></div>
21         <div class="container fill_height">
22             <div class="row fill_height">
23                 <div class="banner_product_image"></div>
24                 <div class="col-lg-5 offset-lg-4 fill_height">
25                     <div class="banner_content">
26                         <h1 class="banner_text">new era of smartphones</h1>
27                         <div class="banner_price"><span>$530</span><span>$460</span></div>
28                         <div class="banner_product_name">Apple Iphone 6s</div>
29                         <div class="button_banner_button"><a href="#">Shop Now</a></div>
```

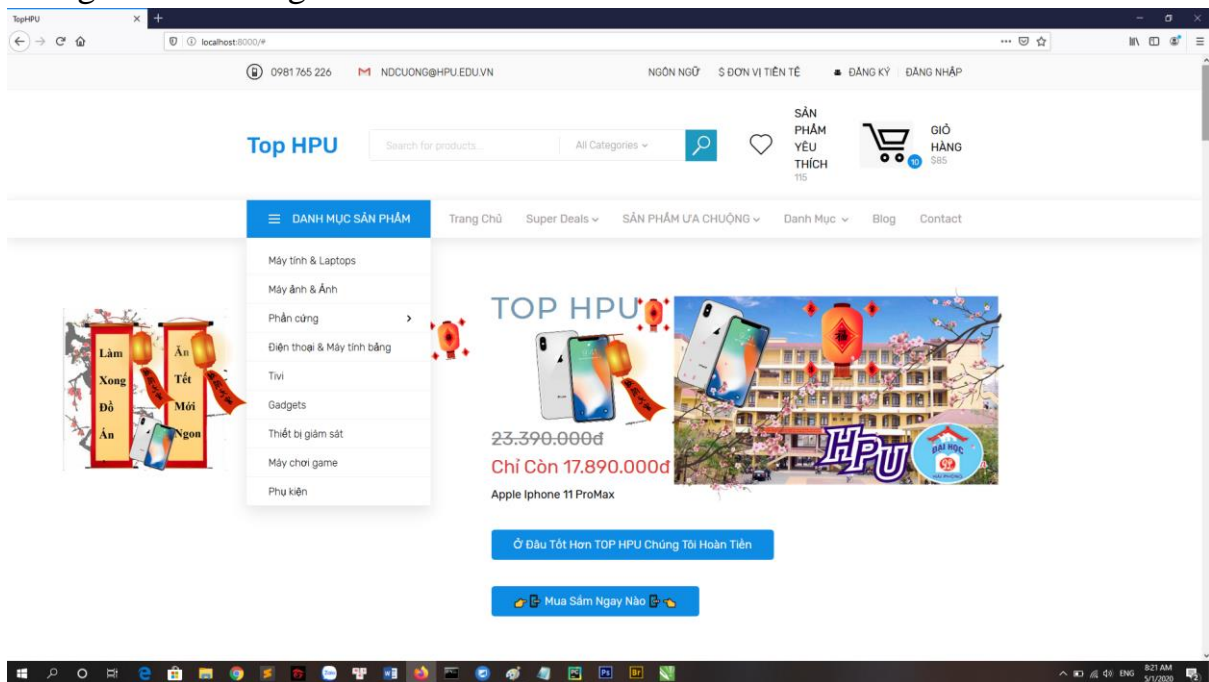
Hình 2.7 Load static

Sau khi đã hoàn thành load static xong thì chúng ta truy cập vào địa chỉ “<http://localhost:8000/>” để xem kết quả.

Web site đã hiển thị được với một giao diện đẹp mắt hơn. Với những thông tin của các loại sản phẩm, giá cả và hình ảnh hiển thị tượng trưng cho sản phẩm.

Ta đặt tên web site là **Top HPU**

Trang chủ của trang web.



trong trang web có nhiều hạng mục. (sản phẩm hot, bán chạy trong tuần, sản phẩm đang giảm giá, các chương trình khuyến mại).

Sản Phẩm Hot Trong Tuần



Tai Nghe Bluetooth
Tai Nghe Lightning
620.000đ
180.000đ

Còn Hàng: 22 Đã Bán: 8





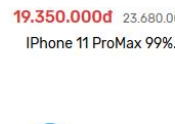



Hãy Nhanh Tay
Khuyến Nại Kết Thúc Sau:

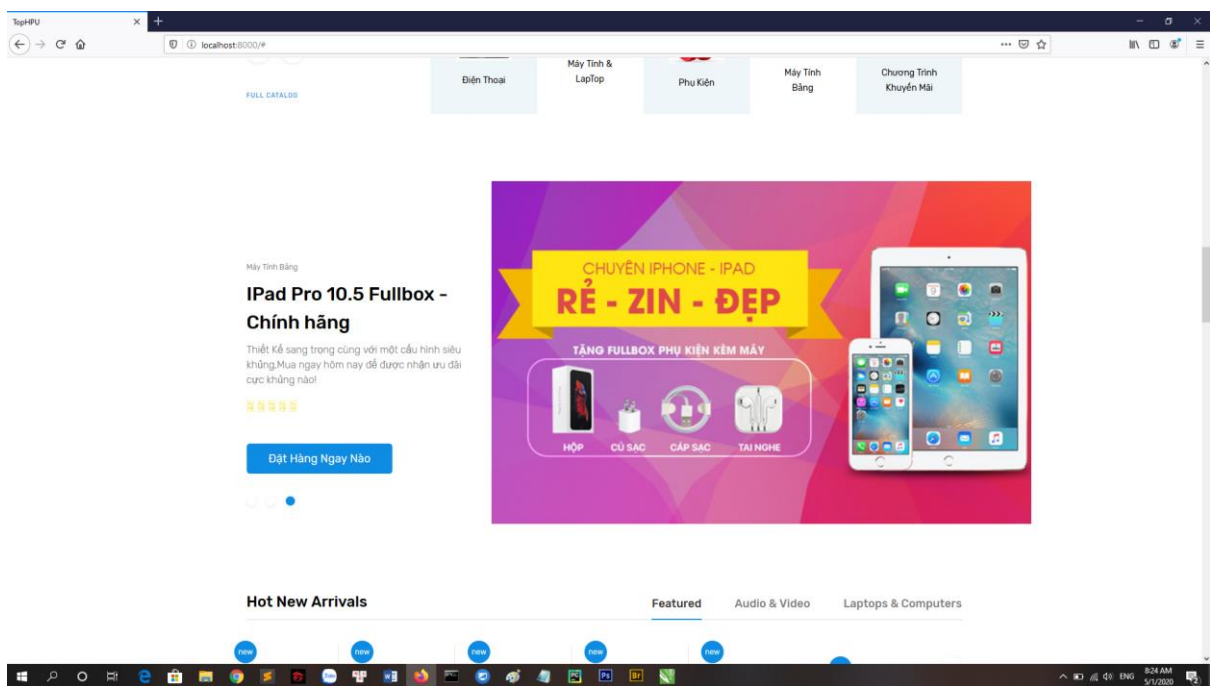
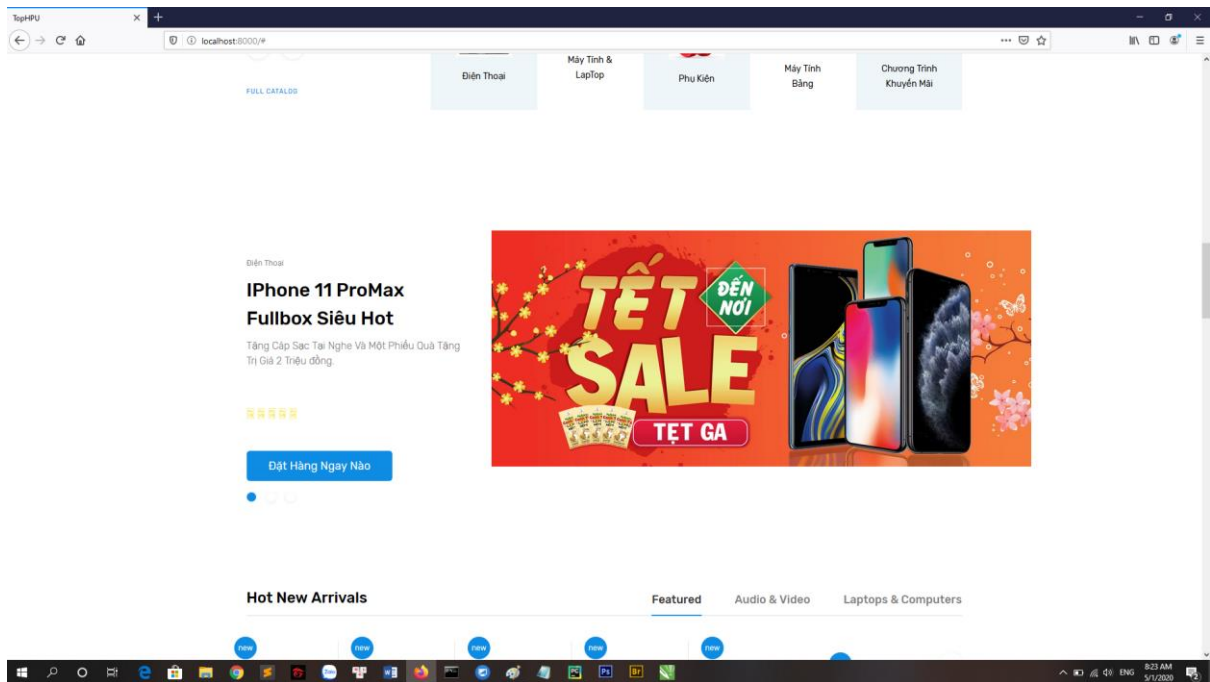
47	56	31
HOURS	MINS	SECS

Sản Phẩm Hot Bán Chạy

Sản Phẩm Khuyến Mãi

Top Bình Chọn

 <p>19.350.000đ 23.680.000 iPhone 11 ProMax 99%...</p>	 <p>8.650.000đ iPhone 7Plus (Hết Hàng)..</p>	 <p>830.000đ Tai Nghe Bluetooth...</p>	 <p>8.200.000đ iPad Air 2 Wifi Cellular 64GB</p>
 <p>850.000đ Apple iPod shuffle</p>	 <p>21.900.090đ 19.900.090 Samsung Note 10 Plus</p>	 <p>8.200.000đ Samsung J330F...</p>	 <p>120.000đ Cáp Sạc...</p>



KẾT LUẬN

Sau thời gian tham khảo tìm tòi và dưới sự chỉ bảo của thầy hướng dẫn về bài tập đồ án Tìm hiểu lập trình python và ứng dụng phát triển ứng dụng web với Django trong khoảng thời gian nhất định dành cho việc thực hiện đề tài, nên một số vấn đề vẫn chưa được hoàn chỉnh. Tuy nhiên, đồ án đã đạt được một số kết quả:

Những kết quả đạt được trong đồ án :

- Tìm hiểu về ngôn ngữ python và phát triển ứng dụng web django.
- Tìm hiểu môi trường lập trình PyCharm.
- Tìm hiểu về cách quản lý một website bán hàng.
- Lập trình python, khung phát triển web Django.
- Demo một khung website bán hàng viết bằng ngôn ngữ python.

Tuy nhiên trong quá trình làm bài vẫn còn một số điều cần bổ sung như:

- Giao diện vẫn chưa thực sự đẹp mắt, còn thiếu sót
- Chưa thao tác thành thực về lập trình python
- Website bán hàng vẫn còn thiếu sót
- Trình bày thiếu logic, cách diễn đạt còn kém.

Em sẽ cố gắng để ngày càng hoàn thiện, trau dồi kỹ năng lập trình cũng như thiết kế trở nên tốt hơn.

Em xin chân thành cảm ơn!

DANH MỤC TÀI LIỆU THAM KHẢO

Tài liệu tham khảo :

- 1 Sách tự học lập trình Python căn bản-NXB Đại Học Quốc Gia HCM
- 2 Sách điện tử Learn python the hard way.
- 3 <https://binhthanh dang.files.wordpress.com/2015/12/python-rat-la-co-ban-vo-duy-tuan.pdf>
- 4 <https://www.w3schools.com/>
- 5 <https://docs.djangoproject.com/en/3.0/>
- 6 <http://phocode.com/python/django/django-host-website-voi-pythonanywhere/>

Chú thích : những từ được in đậm dùng để chỉ các thư mục, từ khóa hay các lệnh quan trọng

In đậm nằm trong dấu “” là những đường dẫn của thư mục.