

PRESENTED BY GROUP 4

DJANGO FRAMEWORK

THE PERFECTIONISTS WITH
DEADLINES

WWW.AMTHUCLANGDAIHOC.COM



PRESENTED BY GROUP 4

1. Trần Vỹ Khang - 22520628

2. Nguyễn Đăng Nguyên Khang - 22520617

3. Hồ Hoàng Diệp - 22520249

WWW.AMTHUCLANGDAIHOC.COM

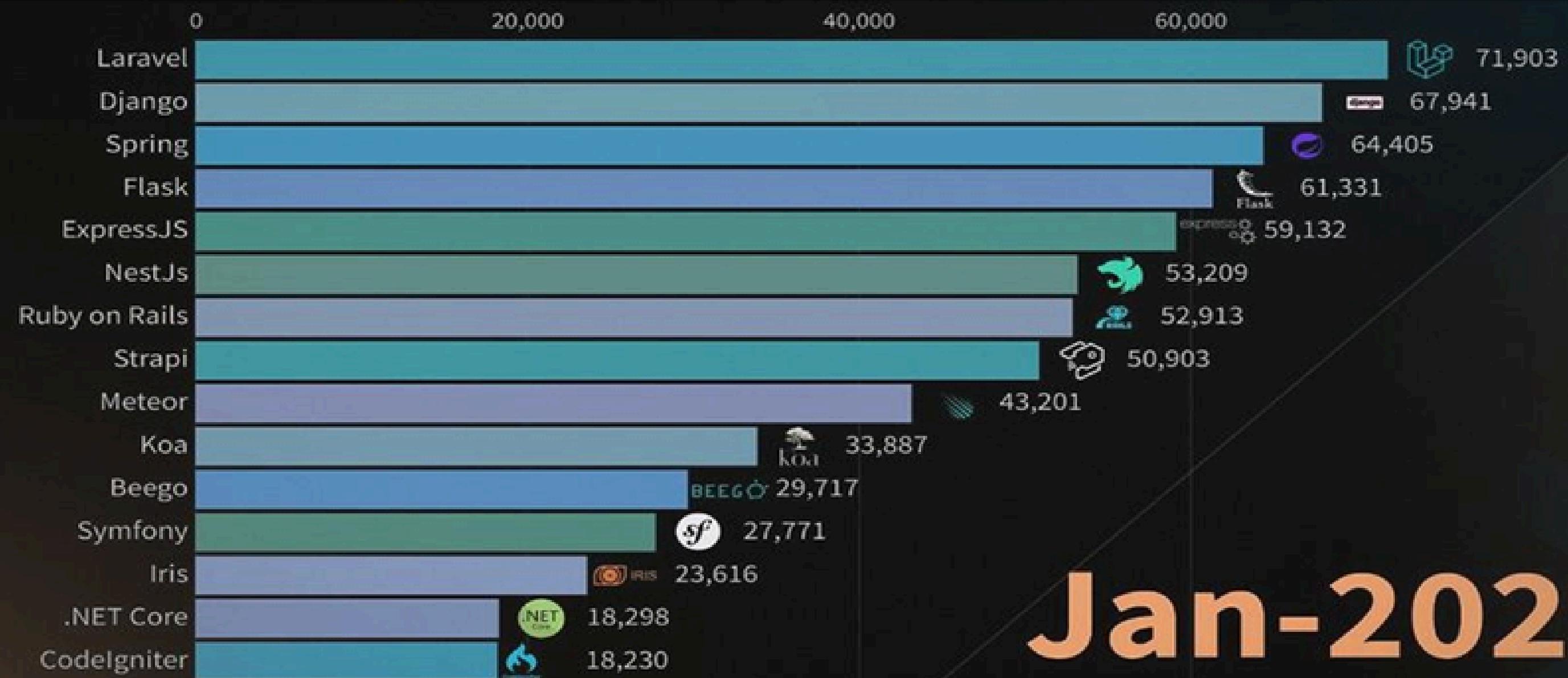


OVERVIEW DJANGO

Django là một trong những Open-source Framework về Back-end cho lập trình viên khi muốn xây dựng Web-server bằng Python

VÌ SAO NÊN SỬ DỤNG DJANGO ?

Most Popular Backend Frameworks



Jan-2023

-
- DỄ DÀNG TIẾP CẬN VỚI NGƯỜI MỚI BẮT ĐẦU
 - DỄ DÀNG LÀM QUEN VỚI CÁC FUNCTION , SYNTAX
 - HỖ TRỢ ĐA NỀN TẢNG
 - CỘNG ĐỒNG LỚN MẠNH
-

ƯU ĐIỂM CỦA DJANGO SO VỚI CÁC FRAMEWORK KHÁC

- Giúp xây dựng trang web nhanh hơn (vd Youtube, Instagram ...).
- Các dòng code thực hiện ít hơn. (python)
- Cung cấp độ bảo mật cao (Admin site)
- Cung cấp các chức năng có sẵn để tái sử dụng ,

THE STRUCTURE OF A DJANGO PROJECT

- I. Create New Project .
- II. The Developer Server.
- III. Creating the Polls app.

I. CREATE NEW PROJECT .

- Đầu tiên, nếu muốn sử dụng Django , phải có sẵn Python

Dùng terminal, tạo ra dự án bằng câu lệnh :

“django-admin startproject mysite”

‘mysite’ là tên thư mục dự án
, có thể đổi thành tên khác

```
mysite/
  manage.py
  mysite/
    __init__.py
    settings.py
    urls.py
    asgi.py
    wsgi.py
```

I. CREATE NEW PROJECT .

Ngang cấp với thư mục mysite là file 'manage.py' , nó cho phép dự án chúng ta được thực thi thông qua câu lệnh trong terminal , thực hiện quản lý các cơ sở dữ liệu và các tương tác khác nhau .

__init__.py : là file quản lý các packet của dự án , cho phép bạn import các module từ đó.

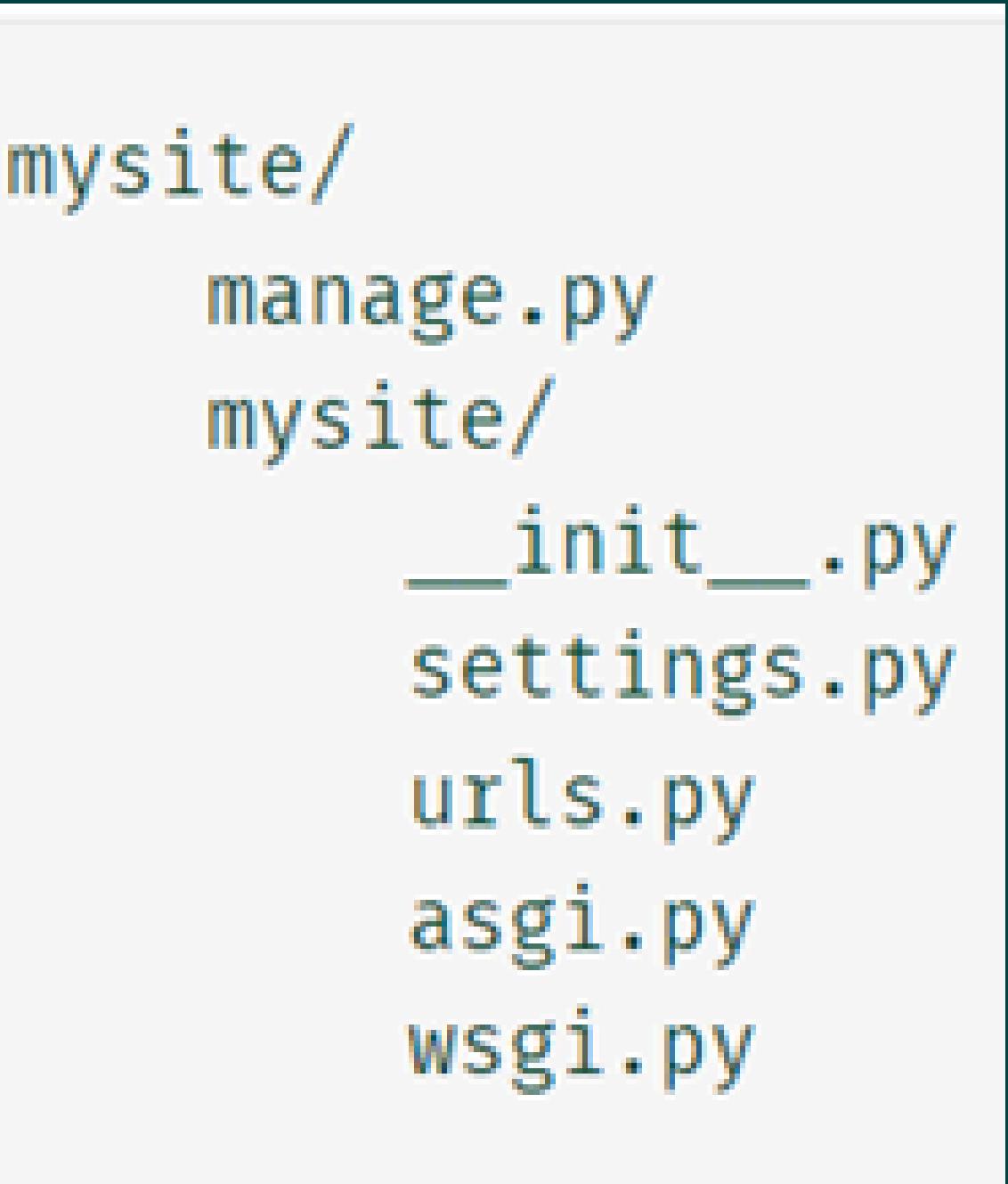
The screenshot shows a file tree structure within a white rectangular area. At the top level, there is a folder named 'mysite'. Inside 'mysite', there are several files: 'manage.py', '_init_.py', 'settings.py', 'urls.py', 'asgi.py', and 'wsgi.py'. All file names are displayed in a monospaced font, with the exception of the folder name 'mysite' which is in a standard sans-serif font.

```
mysite/
  manage.py
  __init__.py
  settings.py
  urls.py
  asgi.py
  wsgi.py
```

I. CREATE NEW PROJECT .

'settings.py': chứa tất cả cấu hình cài đặt Django (host, database , URL, Security ...)

'urls.py' : file này sẽ chứa các đường dẫn sẽ được xử lý kèm theo với các function liên quan tới JS hoặc FE trong file 'view.py' .



```
mysite/
manage.py
mysite/
__init__.py
settings.py
urls.py
asgi.py
wsgi.py
```

I. CREATE NEW PROJECT .

‘asgi.py’(Asynchronous Server Gateway Interface): dùng để hỗ trợ các tiêu chuẩn mới như ASGI, các tính năng không đồng bộ như WebSockets, Server-Sent Events (SSE), và các ứng dụng real-time.

‘wsgi.py’: thường được sử dụng để cấu hình ứng dụng Django để chạy trên các máy chủ web như Gunicorn hoặc uWSGI.

```
mysite/
    manage.py
    mysite/
        __init__.py
        settings.py
        urls.py
        asgi.py
        wsgi.py
```

II. THE DEVELOPER SERVER.

Sau khi khởi tạo một project, để khởi động một máy chủ ta sử dụng lệnh sau sẽ xuất hiện link (mặc định sẽ sử dụng IP và port 127.0.0.1 và 8000)

... \> py manage.py runserver

```
Starting development server at http://127.0.0.1:8000/
```

Trong file **setting.py** , truy cập trang web trên
nhiều thiết bị với IP mạng và port cụ thể

```
ALLOWED_HOSTS = ['127.0.0.1', 'localhost', '10.0.22.255']
```

python manage.py runserver 10.0.22.255:5500

III. CREATING THE APP.

Để tạo ra app, hãy đảm bảo chúng ta đang
ở đường dẫn ngang cấp với file
`manage.py`

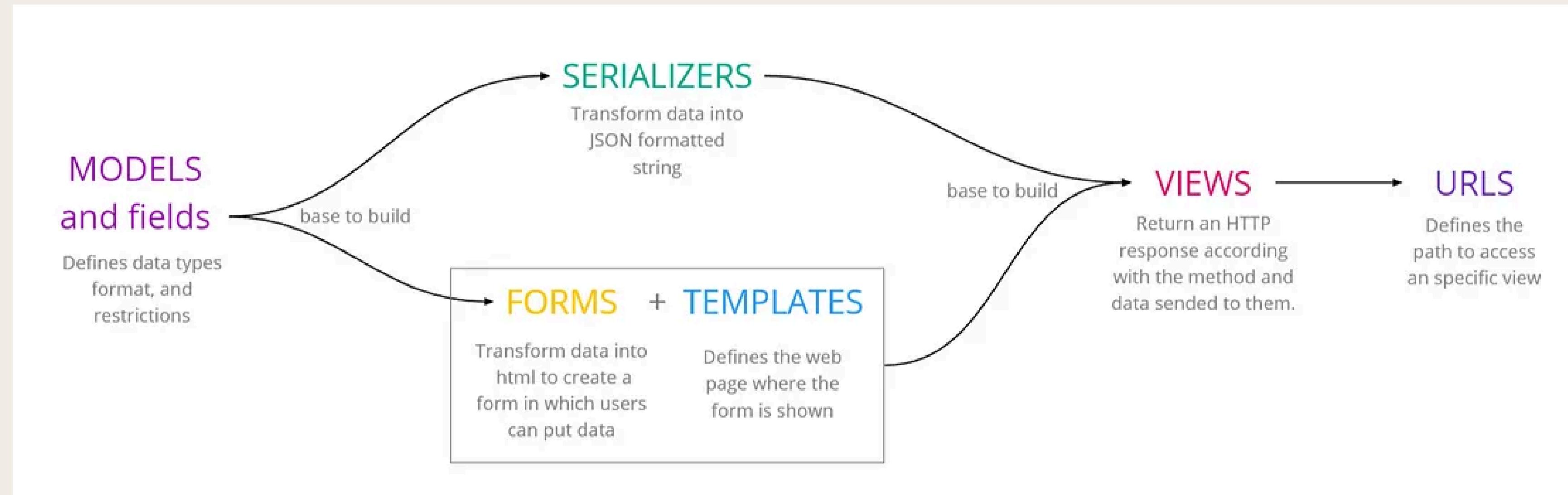
```
... \> py manage.py startapp polls
```

Polls app giống như mô hình xây nên
dùng để tương tác với người dùng, thu
thập ý kiến và xử lý với `models.py` và
`views.py`

một project thì có thể có nhiều app và
một app có thể nằm trong nhiều project.

```
polls/  
    __init__.py  
    admin.py  
    apps.py  
    migrations/  
        __init__.py  
    models.py  
    tests.py  
    views.py
```

DJANGO ĐƯỢC XÂY DỰNG THEO MÔ HÌNH MVT



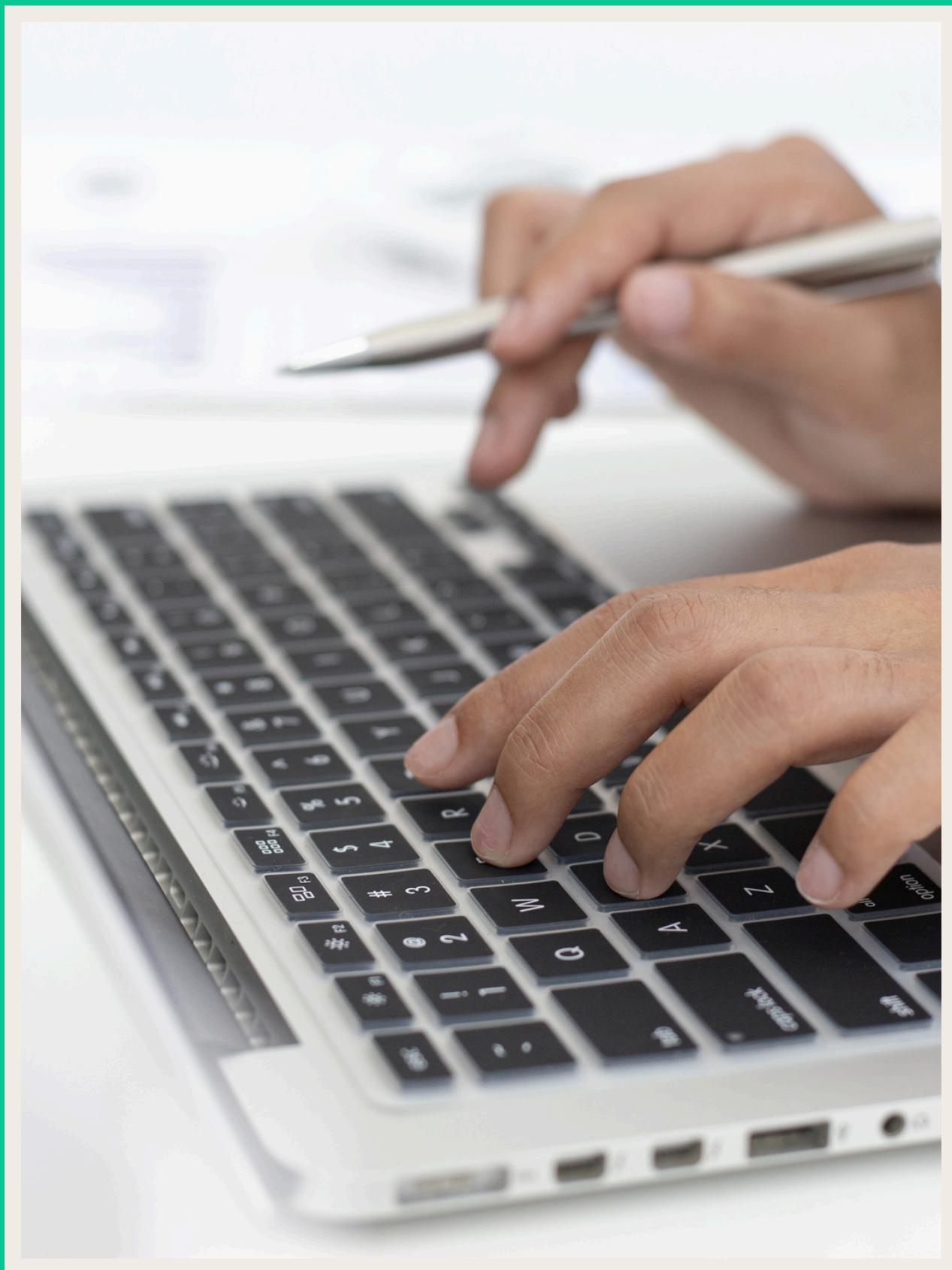
TEMPLATE AND VIEWS

- Template là một file văn bản có thể ở bất kỳ định dạng nào(html,xml,csv).

```
TEMPLATES = [
    {
        "BACKEND": "django.template.backends.django.DjangoTemplates",
        "DIRS": [
            # Thêm đường dẫn các file template vào đây.
        ],
        "APP_DIRS": True,
        "OPTIONS": { },
    },
]
```

TEMPLATE AND VIEWS

- Template chứa các biến(sẽ được thay thế bằng các giá trị khi được đánh giá), và tags được sử dụng để kiểm soát logic trong template.
- Biến:
 - Các biến bắt đầu bằng “{{” và kết thúc bằng “}}” Ví dụ: {{ variable }}
 - Tên biến có thể chứa bất kì kí tự nào nhưng không được bắt đầu bằng dấu gạch dưới hoặc số.
 - Dấu chấm “.” có một ý nghĩa đặc biệt trong một số trường hợp. Thường thì dấu chấm được sử dụng để truy cập các thuộc tính hoặc phương thức của một đối tượng. Ví dụ: user.name có thể đề cập đến thuộc tính "name" của đối tượng "user".



TEMPLATE AND VIEWS

- Filter:

- Có thể chỉnh sửa hiển thị của các biến bằng cách sử dụng filter. Một số filter thông dụng:
 - default: Sử dụng giá trị mặc định nếu biến không tồn tại hoặc rỗng.
Ví dụ: {{ value|default:"nothing" }}
 - length: Trả về độ dài của giá trị, có thể là chuỗi hoặc danh sách. Ví dụ: {{ value|length }}
 - {{ name|lower }}: hiển thị giá trị của {{ name }} sau khi chuyển đổi văn bản thành chữ thường

- Tags:

- Phức tạp hơn biến. Một số tạo văn bản ở đầu ra, một số tạo luồng điều khiển bằng cách thực hiện các vòng lặp hoặc logic và một số tải thông tin bên ngoài vào mẫu để các biến sau này sử dụng.
- Có cấu trúc { % tag % }
- Có thể sử dụng filter và các toán tử khác nhau trong thẻ if.
- Thẻ {{ csrf_token }} trong Django được sử dụng để bảo vệ các biểu mẫu tránh khỏi tấn công CSRF (Cross-Site Request Forgery), một loại tấn công mạng phổ biến.

for::

```
{% for athlete in athlete_list %}  
  <li>{{ athlete.name }}</li>  
{% endfor %}
```

If, elif, else:

```
{% if athlete_list %}  
  Number of athletes: {{ athlete_list|length }}  
{% elif athlete_in_locker_room_list %}  
  Athletes should be out of the locker room soon!  
{% else %}  
  No athletes.  
{% endif %}
```

TEMPLATE AND VIEWS

Tương tự, comment được đặt
giữa { % comment %} và
{% endcomment %} - thông tin
trong comment không hiển thị
lên giao diện html

Ngoài ra còn có include tag
cho phép kế thừa nội dung của
file html khác

templates/footer.html :

```
<p>You have reached the bottom of this page, thank you for
```

templates/template.html :

```
<h1>Hello</h1>
```

```
<p>This page contains a footer in a template.</p>
```

```
{% include 'footer.html' %}
```

TEMPLATE AND VIEWS

- Views.py là nơi định nghĩa các hàm để kiểm soát logic của trang Web:
 - Xử lý các request của người dùng khi người dùng thao tác trên giao diện.
 - Hiển thị các form có sẵn của django hoặc các form được định nghĩa trong forms.py ra template.
 - Lấy dữ liệu từ database để hiển thị ra template.

```
def index(request):  
    if request.method == 'POST':  
        if 'button-login' in request.POST:  
            email = request.POST.get('input-login-account')  
            password = request.POST.get('input-login-password')  
            user = authenticate(request, username=email, password=password)  
            if user is not None:  
                login(request, user)  
                return redirect('whilelogin')  
            else:  
                return render(request, "index.html", {'error_message_login': 'Invalid email or password'})
```

```
<form class="input_text_login" method="POST">  
    {% csrf_token %}  
    <input type="text" name="input-login-account" class="box_input_text_account" placeholder="Email address" />  
    <input type="password" name="input-login-password" class="box_input_text_password" placeholder="Password" />  
    {% if error_message_login %}  
        <p class="error">{{ error_message_login }}</p>  
    {% endif %}
```

STATIC FILES

- Khi xây dựng web ta cần thêm một số tệp bổ sung như: css, hình ảnh,... Ở Django, các tệp này được gọi là “static files”. Django cung cấp django.contrib.staticfiles để giúp quản lý chúng.
- Lưu trữ các tệp tĩnh trong thư mục có tên static trong project của bạn. Ví dụ:
 - my_app/static/my_app/example.jpg
- Ngoài việc sử dụng thư mục static bên trong ứng dụng, ta có thể thu thập các static files từ danh sách các thư mục định nghĩa trong STATICFILES_DIRS.

STATIC_URL = "static/"

```
{% load static %}

![My image]({% static 'my_app/example.jpg' %})
```

```
STATICFILES_DIRS = [
    'D:\\WEB_WORK\\UI_Web_Page_Offical\\User_Picture'
]
```

ADMIN

- Một trong những phần mạnh mẽ nhất của Django là automatic admin interface. Nó đọc metadata từ các model của bạn để cung cấp giao diện nhanh chóng, model-centric interface là nơi trusted users có thể quản lý nội dung của trang web.
- Để bật trang admin:
 - Trong urls.py:

```
urlpatterns = [
    path('admin/', admin.site.urls),
]
```

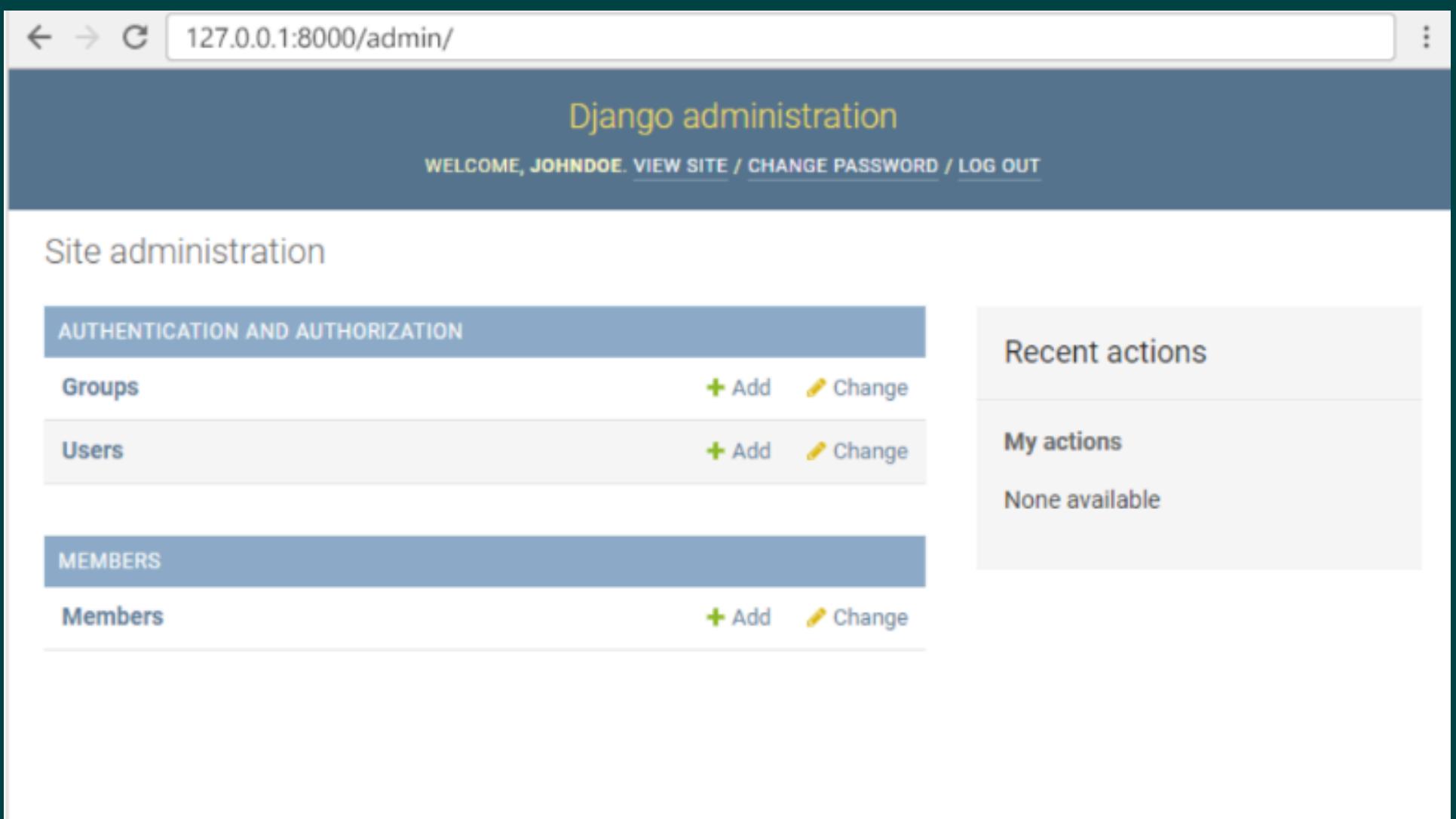
- Sau đó truy cập vào trang admin bằng cách dùng URL trang web thêm “/admin/” vào.
- Nếu bạn cần tạo người dùng để đăng nhập vào trang admin, hãy sử dụng lệnh createsuperuser. Theo mặc định, việc đăng nhập vào quản trị viên yêu cầu người dùng phải có thuộc tính is_staff là True.



ADMIN

- Ta có thể đăng ký mô hình trong admin.py. VD:

```
from django.contrib import admin  
  
from .models import Member  
  
# Register your models here.  
  
admin.site.register(Member)
```



The screenshot shows the Django administration interface at the URL 127.0.0.1:8000/admin/. The top navigation bar displays "Django administration", "WELCOME, JOHNDOE.", and links for "VIEW SITE", "CHANGE PASSWORD", and "LOG OUT". The main content area is titled "Site administration". It contains two main sections: "AUTHENTICATION AND AUTHORIZATION" and "MEMBERS". The "AUTHENTICATION AND AUTHORIZATION" section includes links for "Groups" and "Users", each with "Add" and "Change" buttons. The "MEMBERS" section includes a link for "Members", also with "Add" and "Change" buttons. To the right, there is a sidebar with "Recent actions" (empty) and "My actions" (also empty).



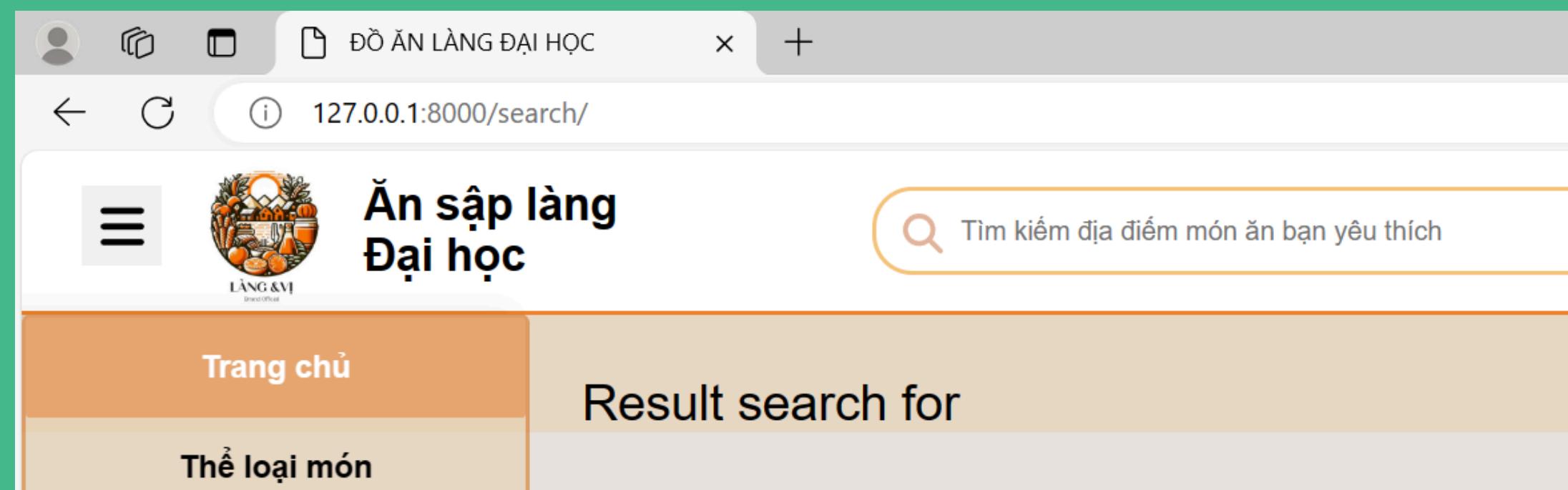
DJANGO URLs

File url.py được dùng để định tuyến các đường dẫn cho website, đưa dữ liệu vào file views.py để phản hồi về nội dung mong muốn

Thêm path() function vào danh sách urlpatterns trong file urls.py,

```
from django.contrib import admin
from django.urls import path,include
from . import views
from .views import CustomPasswordResetView,CustomPasswordResetDoneView,CustomP

urlpatterns = [
    path('', views.index, name="index"),
    path('index/', views.index, name='index'),
    path('search/', views.search, name="search"),
    path('admin/', admin.site.urls),
```



views.py

```
def search(request):
    searched = ""
    keys = []
    if request.method == "POST":
        searched = request.POST["searched"]
        keys = Post.objects.filter(title__conta
return render(request, 'Search/search.html'
```



DJANGO URLs

Một cách thông dụng để xài path là để validate cho một biến có trong url, ví dụ như id. Sử dụng syntax <data_type:variable>

```
# urls.py
from django.urls import path
from .views import RetrieveUserView

urlpatterns = [
    path('users/<int:id>', RetrieveUserView.as_view(), name='users')
]
```

Sử dụng để lấy đường dẫn cho đối tượng user cụ thể, có thể áp dụng làm đường dẫn cho trang profile của từng user



DJANGO URLs

RE_PATH()

Regular expression trong python có dạng (?P<name>expression)

Re_path được sử dụng để xác định các URL patterns, kiểm tra tính hợp lệ của một đường dẫn

```
urlpatterns = [
    re_path(r"^index/(:page-(?P<page_number>[0-9]+))/)?$", views.index)
```

Giả sử page_number = 2 thì mẫu sẽ cho kết quả khớp với index/page-2/



DJANGO MODELS

Model.py là nơi khai báo các đối tượng, dữ liệu cần được lưu trữ. Nó chứa các trường và thao tác trên dữ liệu lưu trữ. Mỗi đối tượng trong model ánh xạ tới một bảng cơ sở dữ liệu.

Ví dụ: định nghĩa class Person gồm first_name và last_name:

```
from django.db import models

class Person(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
```

Sử dụng lệnh migrate và makemigrations để đưa các trường vào database.

Person được lưu trữ ở database sẽ có dạng

```
CREATE TABLE myapp_person (
    "id" bigint NOT NULL PRIMARY KEY GENERATED BY DEFAULT AS
IDENTITY,
    "first_name" varchar(30) NOT NULL,
    "last_name" varchar(30) NOT NULL
);
```

DJANGO MODELS

Cơ sở dữ liệu được tạo mặc định là SQLite3, Django hỗ trợ cho nhiều CSDL khác như PostgreSQL, MySQL, Oracle, MongoDB...

Để sử dụng được các đối tượng trong models.py, trong INSTALLED_APPS của file setting.py cần thêm vào thư mục chứa models.py.

Model method:

Có thể xây dựng các method cho các đối tượng ngay bên trong Model

```
# Create your models here.

class Post(models.Model):
    title = models.CharField(max_length=100)
    content = models.TextField
    date = models.DateTimeField(auto_now_add=True)
    star = models.FloatField(default=0)
    address = models.CharField(max_length=100)
    image = models.ImageField()

    def __str__(self):
        return self.title
```



INSERT, UPDATE, DELETE DATA

Member.objects.all() trả về tất cả các dữ liệu có trong database dưới dạng QuerySet object

Để đưa dữ liệu vào thì chỉ cần gán dữ liệu cần đưa vào cho một biến và dùng lệnh save()

```
member1 = Member(firstname='Tobias', lastname='Refsnes')
member2 = Member(firstname='Linus', lastname='Refsnes')
member3 = Member(firstname='Lene', lastname='Refsnes')
member4 = Member(firstname='Stale', lastname='Refsnes')
member5 = Member(firstname='Jane', lastname='Doe')
members_list = [member1, member2, member3, member4, member5]
for x in members_list:
    x.save()
```

Tương tự với update data, ta chỉ cần dùng lệnh save với đối tượng muốn thay đổi giá trị.

Để xóa một dữ liệu, ta sử dụng lệnh delete với đối tượng muốn xóa.

```
x = Member.objects.all()[5]
x.delete()
```



DJANGO QUERYSET

01. GET DATA

value() method trả về từng đối tượng được truy xuất, với tên và giá trị tạo thành một cặp key/value, tương tự với việc SELECT trong SQL

02. FILTER DATA

filter() method trả về từng đối tượng được truy xuất theo một điều kiện nhất định, tương tự với mệnh đề WHERE trong SQL

03. ORDER BY DATA

order_by() method trả về từng đối tượng được truy xuất và sắp xếp theo một điều kiện nhất định, tương tự với mệnh đề ORDER BY trong SQL

DJANGO DOCUMENT

<https://docs.djangoproject.com/en/5.0/>



PRESENTED BY GROUP 4

**THANK
YOU VERY
MUCH!**

