



# Introduction to Django

## Django Session-1



# Table of Contents

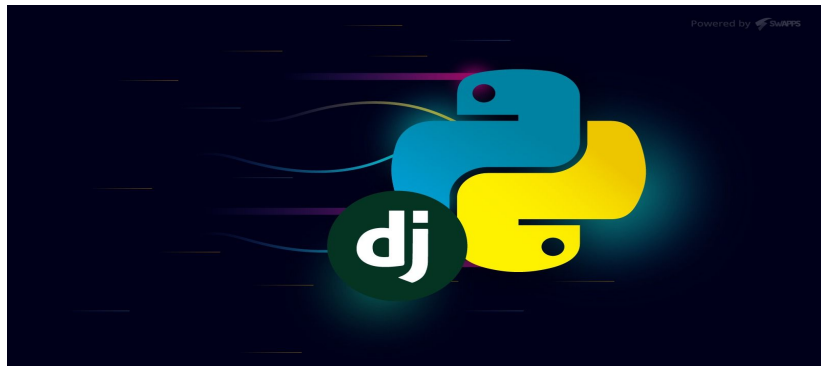


- ▶ What is Django?
- ▶ How Django Works
- ▶ Installation
- ▶ Create Project
- ▶ Project Structure
- ▶ Login to Admin Site



# What is Django?

1





# What is Django?



MAX BURSTEIN

```
3. from django.shortcuts import render
4. from django.http import HttpResponseRedirect, HttpResponseServerError
5. from django.views.decorators.csrf import csrf_exempt
6. from django.contrib.sessions.models import Session
7. from django.contrib.auth.decorators import login_required
8.
9. import redis
10.
11. #login_required
12. def home(request):
13.     comments = Comments.objects.select_related().all()[0:100]
14.     return render(request, 'index.html', locals())
15.
16. #csrf_exempt
17. def node_api(request):
18.     try:
19.         #Get User from sessionId
20.         session = Session.objects.get(session_key=request.POST.get('sessionId'))
21.         user_id = session.get_decoded().get('_auth_user_id')
22.         user = User.objects.get(id=user_id)
23.
24.         #Create comment
25.         Comments.objects.create(user=user, text=request.POST.get('comment'))
26.
27.         #Once comment has been created post it to the chat channel
28.         r = redis.StrictRedis(host='localhost', port=6379, db=0)
29.         r.publish('chat', user.username + ' : ' + request.POST.get('comment'))
30.
31.         return HttpResponseRedirect("Everything worked :)")
32.     except Exception, e:
33.         return HttpResponseServerError(str(e))
```



Students, write your response!

Pear Deck Interactive Slide

Do not remove this bar



# What is Django?

Django is a free and open source web framework.

It is based on python.

Allows you to build web applications easily without all the installation or dependency issues that you usually encounter with other frameworks.





# Who uses Django?

YouTube

Dropbox

reddit

Instagram

Quora

Spotify

DISQUS



# Why Django?

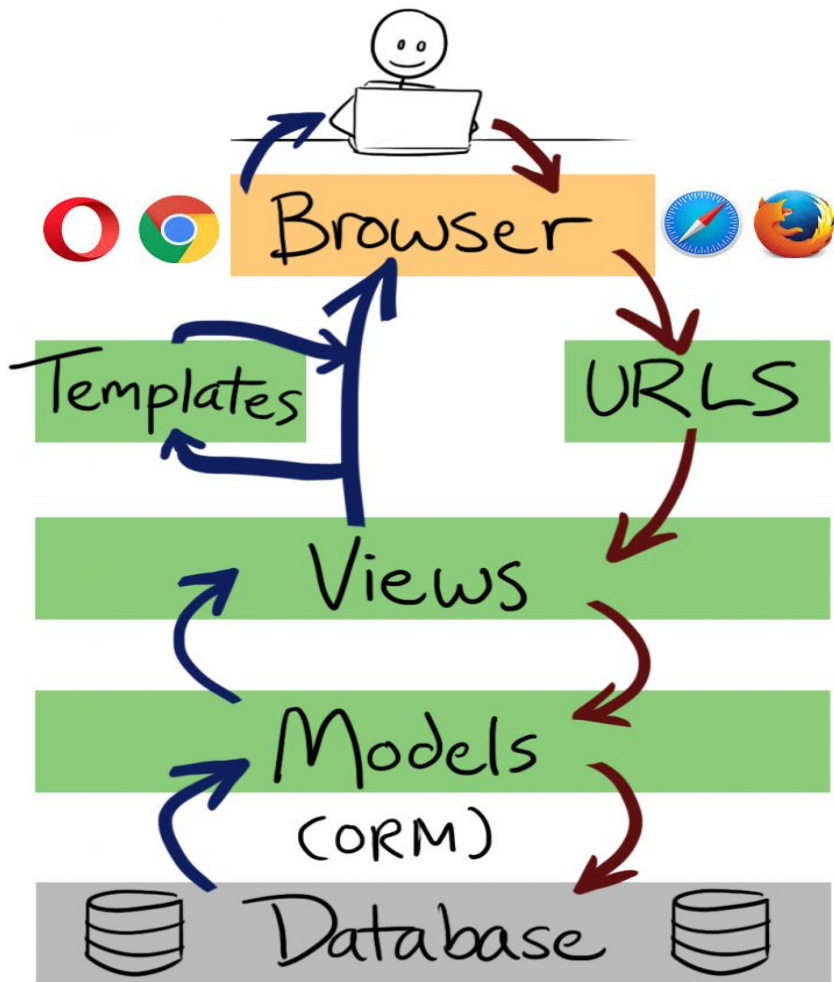
- Fully functional framework that requires nothing else
- Built in admin interface
- Very Scalable
- Security
- Thousands of additional packages





2

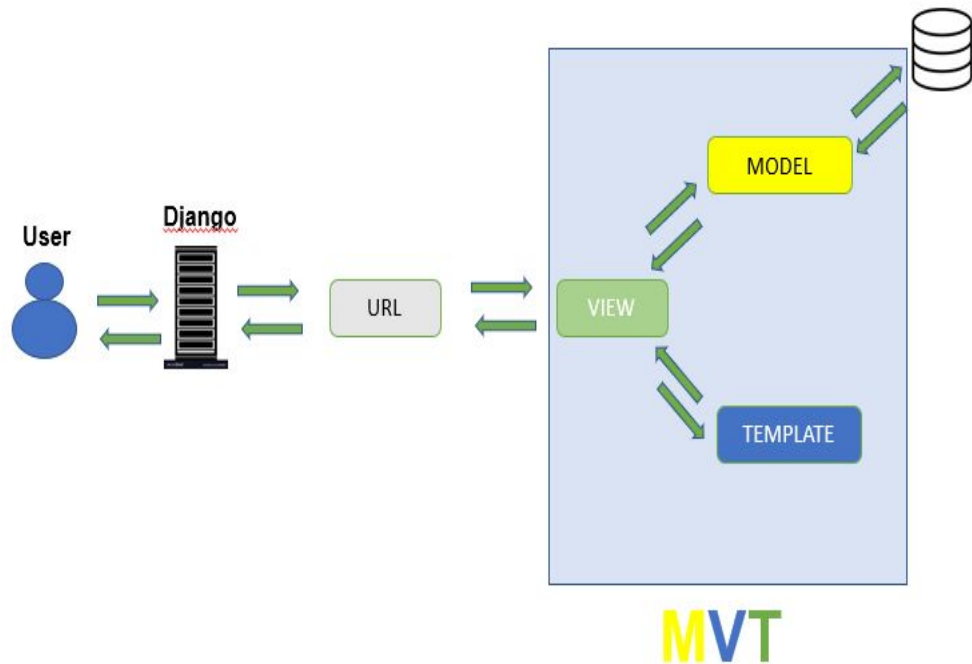
# How Django Works







# How Django Works



A user requests for a resource to the Django, Django works as a controller and check to the available resource in URL (urls.py file). If URL maps, a view (function) is called that interact with model and template, it renders a template. Django responds back to the user and sends a template as a response.



# 3 Installation





# Installation



Create virtualenv

Activate virtualenv

installation

- `pip install virtualenv`
- `virtualenv nameofyourvirtualenv`
- `nameofyourvirtualenv/Scripts/activate`  
(Windows)
- `source`  
`nameofyourvirtualenv/bin/activate`  
(Linux and mac)
- `pip install django`



4

# Create Project

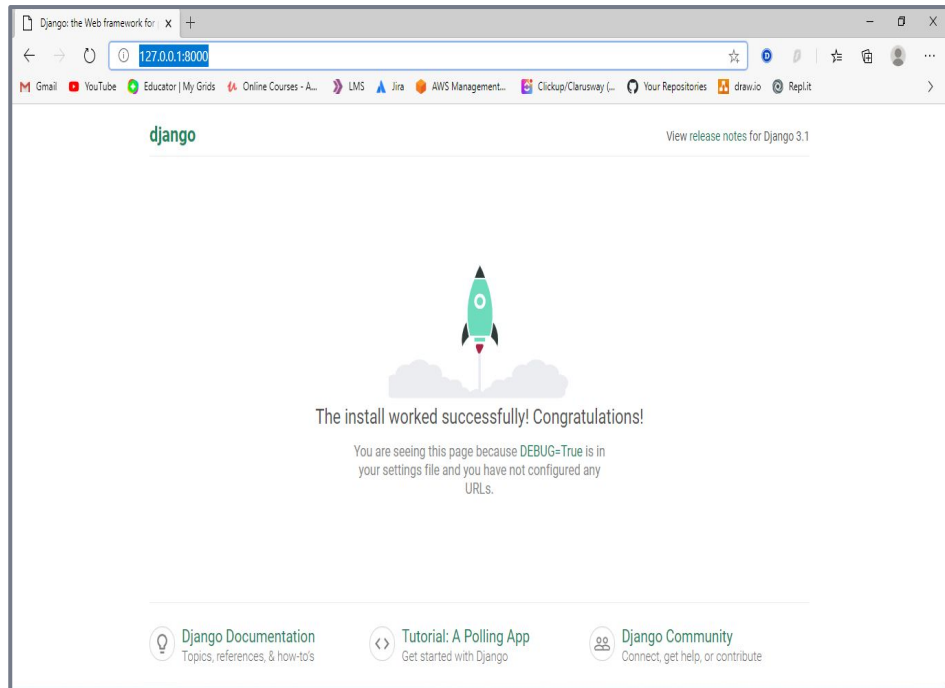


# Create Project



`django-admin startproject`

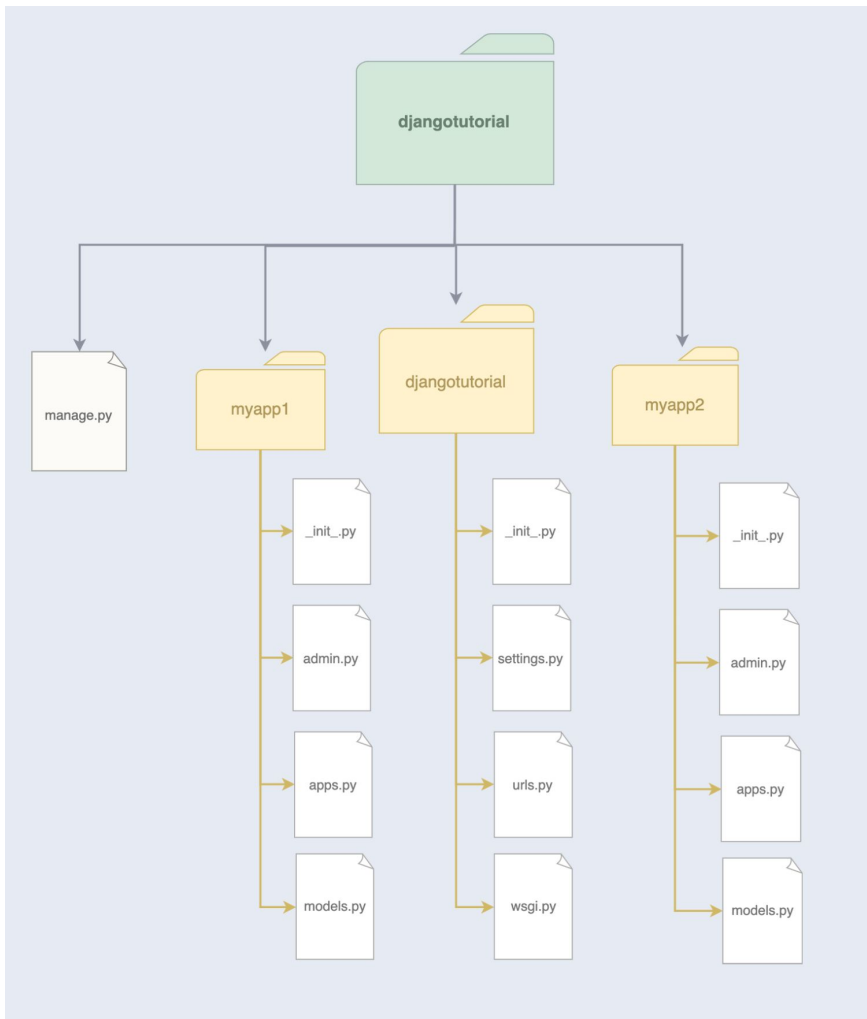
`python manage.py runserver`





4

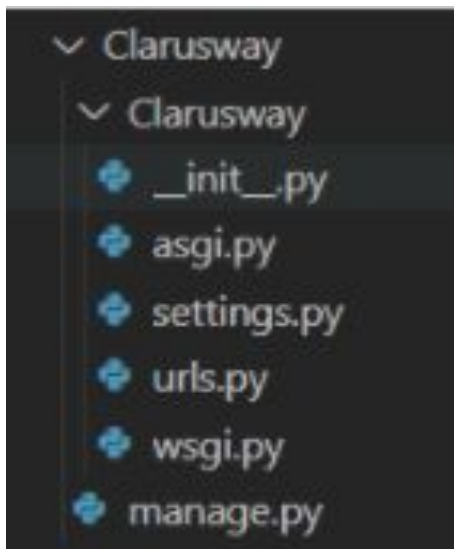
# Project Structure





# Project Structure

## django-admin startproject



- **\_\_init\_\_.py** : An empty file that tells Python that this directory should be considered a Python package.
- **setting.py** : Contains all the website settings.
- **urls.py** : In this file we store all links of the project and functions to call.
- **wsgi.py** : An entry-point for WSGI-compatible web servers to serve your project.
- **asgi.py** : An entry-point for ASGI-compatible web servers to serve your project.
- **manage.py** : A command-line utility that lets you interact with this Django project in various ways.



5

# First App







# Project vs App



## What's the difference between a project and an app?

An app is a Web application that does something – e.g., a Weblog system, a database of public records or a small poll app. A project is a collection of configuration and apps for a particular website. A project can contain multiple apps. An app can be in multiple projects



# ▶ Start an App

`python manage.py startapp appname`

add your app to INSTALLED\_APP list

```
INSTALLED_APPS = [  
    .....  
    'appname'  
]
```



# Create View



appname/views.py

```
1 # HttpResponse is used to
2 # pass the information
3 # back to view
4 from django.http import HttpResponse
5
6 # Defining a function which
7 # will receive request and
8 # perform task depending
9 # upon function definition
10 def index(request):
11
12     # This will return Hello, world. This is fscohort index page.
13     # string as HttpResponse
14     return HttpResponse("Hello, world. This is fscohort index page.")
15
```

# URL Conf.



appname/urls.py

root urls.py

```
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('', views.index, name='index'),
6 ]
7
```

```
1 from django.contrib import admin
2 from django.urls import include, path
3
4 urlpatterns = [
5     path('fscohort/', include('fscohort.urls')),
6     path('admin/', admin.site.urls),
7 ]
8
```

**include()** : Referencing other URLconfs. Whenever Django encounters include(), it chops off whatever part of the URL matched up to that point and sends the remaining string to the included URLconf for further processing.



6

# Login to Admin Site

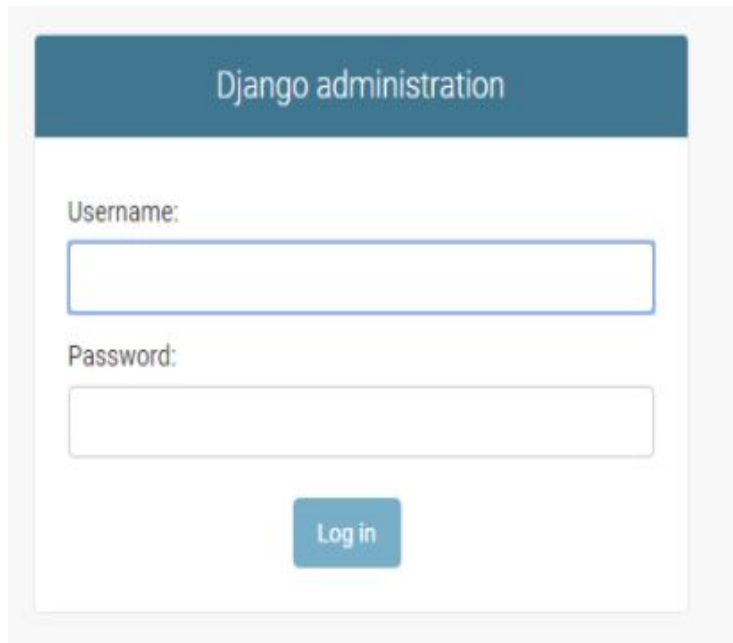




# ► Login to Admin Site

- `python manage.py migrate`
- `python manage.py createsuperuser`
- `python manage.py runserver`

go to `127.0.0.1:8000/admin` and enter your username and password.



The image shows a screenshot of the Django administration login interface. At the top, there is a dark blue header bar with the text "Django administration" in white. Below the header, the page has a white background. There are two input fields: the first is labeled "Username:" and the second is labeled "Password:". Both fields are empty and have a light blue border. Below the password field, there is a blue button with the text "Log in" in white.



# THANKS!

## Any questions?

