Presents

Django Workshop

Rishabh Rathi

Aditya Sharma

Agenda

DAY ONE

- Introduction to Backend
- Complete Python Bootcamp course
- Introduction to Django
- Installing Django
- Project Intro

DAY TWO

- Client - Server Model
- Django's folder structure
- Django Models
- Django Templates
- Creating home view
- Creating Django Forms

DAY THREE

- Creating Django Models
- Creating the url-shortener app
- Creating analytics page
- Creating Preview Page

Today's Agenda

- Django's Folder Structure
- Django App
- Django Architecture
- Django Templates
- Django Models

Creating a Django Project

- The first step is creating your project by using the

  '**django-admin startproject** project_name'

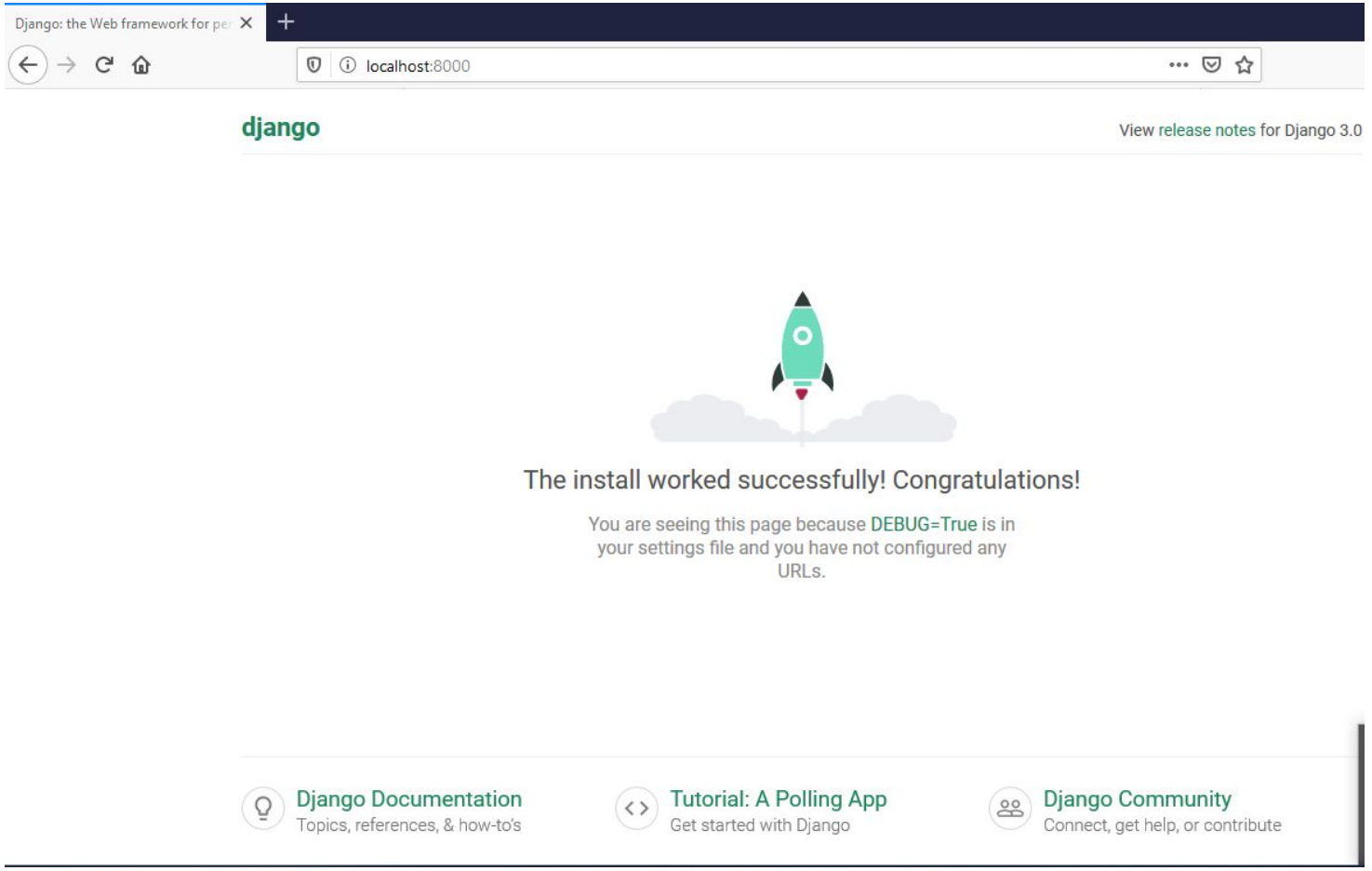
command, where 'project_name' is '**DjangoWorkshop**' in our case. Also, it will generate a lot of files inside our newly created project.

- Change the directory to the newly created project using '**cd**' command

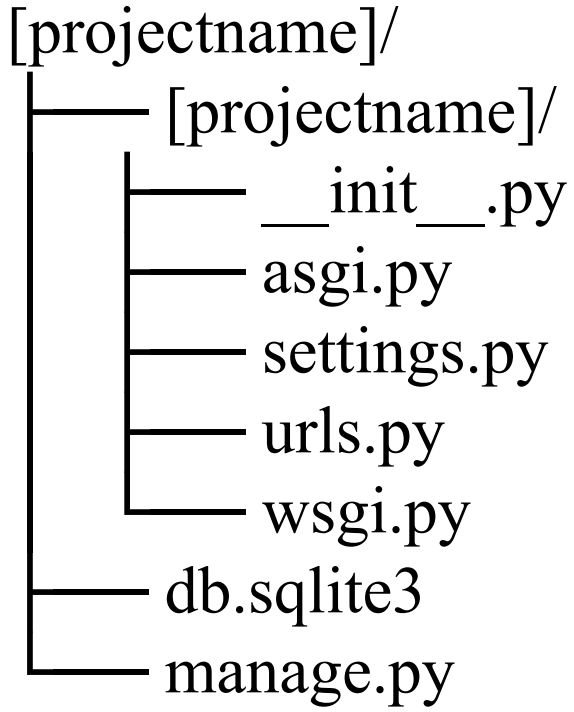
- You can run your project by using '**python manage.py runserver**'

Creating a Django Project (contd.)

- The project can be viewed in your favorite browser (Google Chrome, Mozilla Firefox, etc.). You can come into your browser and type 'localhost:8000' in the URL.

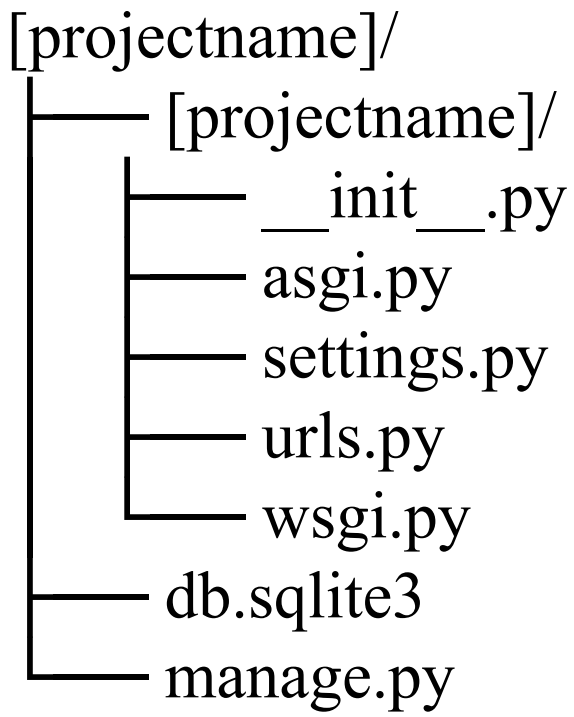


Folder Structure



- **db.sqlite3**: Django uses the SQLite database as default
- **manage.py**: It is the main file to run the project. If this file is missing or not working then the project won't run
- **App folder**: Django creates an App folder with the same name as the project. In this folder, we have many files like settings.py, wsgi.py, asgi.py, and other files.

Folder Structure



- **urls.py**: It is used to keep the URLs and connects to the views file
- **settings.py:** The settings.py file is important because it manages the different components in the project.

Django Admin

- One of the most powerful parts of Django is the automatic admin interface.
- It reads metadata from your models to provide a quick, model-centric interface where trusted users can manage content on your site.

Django MVT Architecture

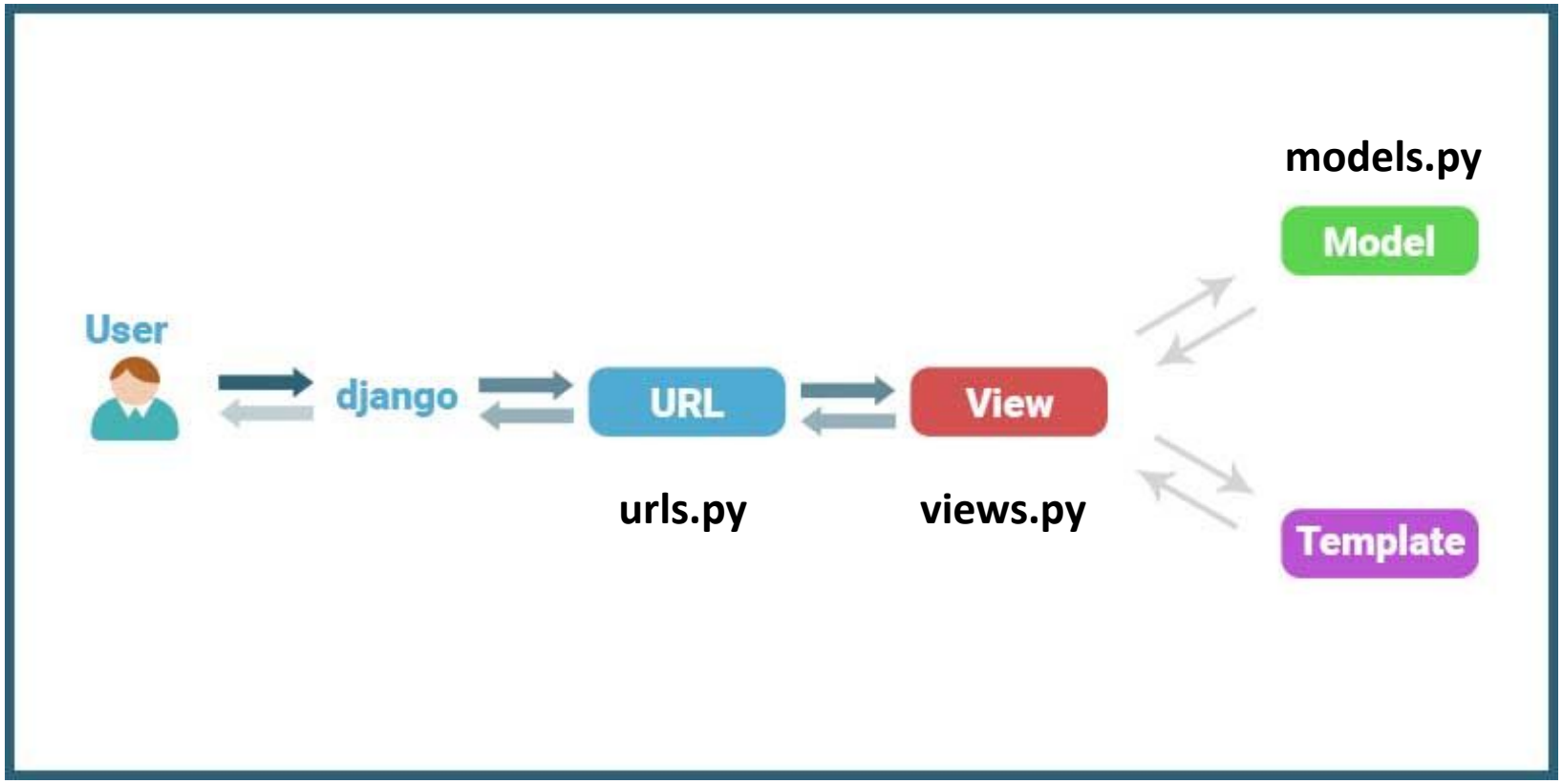
Django is based on **MVT (Model-View-Template)** architecture. MVT is a software design pattern for developing a web application.

MVT Structure has the following three parts –

- **Model:** Model is going to act as the interface of your data. It is responsible for **maintaining data**. It is the logical data structure behind the entire application and is represented by a database (generally relational databases such as MySQL, Postgres).

- **View:** The View is the **user interface** — what you see in your browser when you render a website. It is represented by HTML/CSS/JavaScript and Jinja files.

- **Template:** A template consists of static parts of the desired HTML output as well as some special syntax describing how **dynamic content** will be inserted.



Here, a user requests for a resource to the Django, Django works as a controller and check to the available resource in URL.

If URL maps, a view 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.

Book Store Online : Buy Books O

https://www.amazon.in/Books

Book Store Online : Buy **Books** Online at Best Prices **in In**dia | **Books** Shopping @ **Amazon.in** - **https://www.amazon.in/Books**

https://www.amazon.in/Books - Google Search

Book Store Online : Buy **Books** Online at Best Prices **in In**dia | **Books** Shopping @ **Amazon.in** - **https://www.amazon.in/Books**/b/?ie=UTF8&node=976389031&ref_=nav_cs_books

Microsoft Te

amazo

Orders

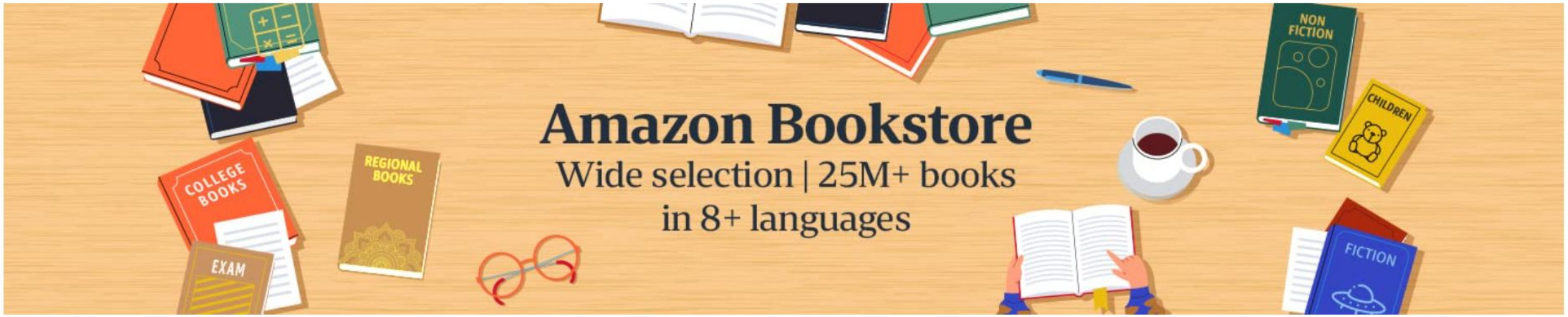
Cart

All Sell

me starting ₹179

Redirects to PrimeVideo.com

Books | Advanced Search | New Releases & Pre-orders | Best Sellers | Browse Genres | Children's & Young Adult | Textbooks | Exam Central | All Indian Languages



With over 25 million books. Browse through variety of genres such as Fiction, Self help, Children's books, School textbooks, Higher education textbooks, and much more.
Explore Editor's corner, Exam Central, Indian Language Books, New Releases & Best sellers .

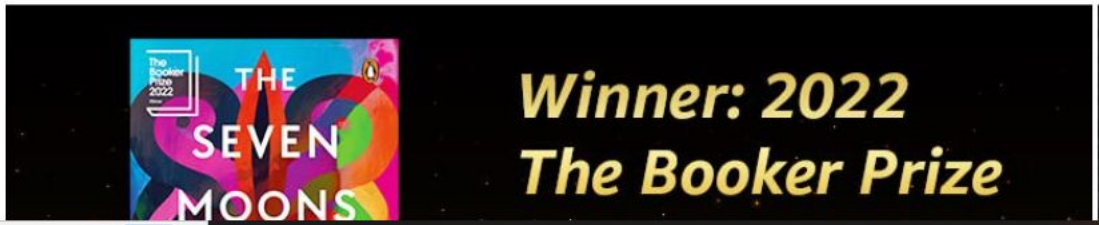
New Arrivals

Last 30 days
Last 90 days
Next 90 days

English & Indian Languages

- [ ] English
- [ ] Hindi
- [ ] Marathi

Top picks for you this week



Type here to search

ENG 19:10 29-10-2022

Book Store Online : Buy Books O

amazon.in/b?ie=UTF8&node=976389031

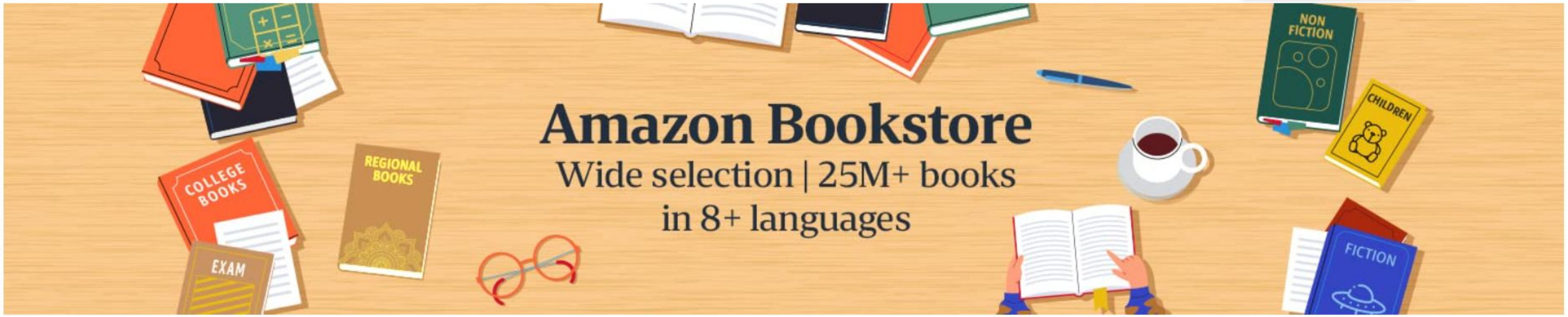
Microsoft Teams - I... | sharmaar1 Geeksfo... | RKNEC - Calendar -... | CPT Online Classes... | Made With ML - M... | Must Do Coding Q... | Problemset - Codef... | Placements - Geeks... | Extensions of the Pr...

amazon.in | Hello Select your address | Books | Hello, sign in Account & Lists | Returns & Orders | 0 Cart

All | Sell | Best Sellers | Today's Deals | Mobiles | Customer Service | Electronics | Home & Kitchen | Fashion | Books | Prime | New Releases | Computers

Sign in
New customer? Start here.

Books | Advanced Search | New Releases & Pre-orders | Best Sellers | Browse Genres | Children's & Young Adult | Textbooks | Exam Central | All Indian Languages



With over 25 million books. Browse through variety of genres such as Fiction, Self help, Children's books, School textbooks, Higher education textbooks, and much more.
Explore Editor's corner, Exam Central, Indian Language Books, New Releases & Best sellers .

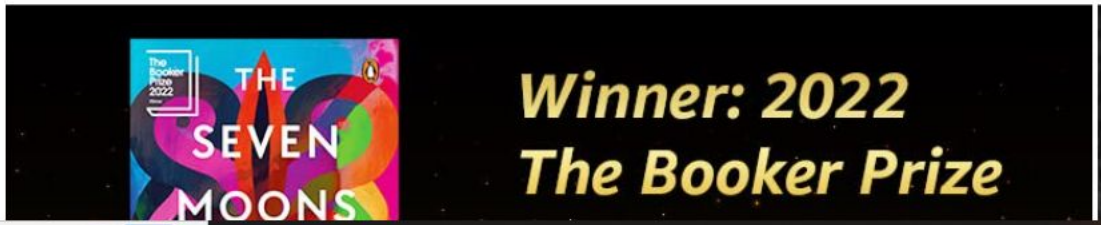
New Arrivals

Last 30 days
Last 90 days
Next 90 days

English & Indian Languages

- [ ] English
- [ ] Hindi
- [ ] Marathi

Top picks for you this week



Type here to search | ENG | 19:10 | 29-10-2022

Online Shopping site in India: Sh

https://www.amazon.in/electronics

Electronics Store: Buy **Electronics** products Online at Best Prices **in** India at **Amazon.in** - **https://www.amazon.in/electronics**

https://www.amazon.in/electronics - Google Search

Electronics Store: Buy **Electronics** products Online at Best Prices **in** India at **Amazon.in** - **https://www.amazon.in/electronics**/b/?ie=UTF8&node=976419031&ref_=nav_cs_electronics

Microsoft Te

amazo

All Sell

Orders Cart

me starting ₹179

PrimeVideo.com

New customer? Start here.

Amazon Fashion

Starting ₹199

Deals on clothing, footwear, beauty & more

PAY ON DELIVERY | LATEST TRENDS

Get extra up to 5% back with Amazon Pay ICICI Bank

On 1st fashion order | *T&C apply

Shop & Pay | Earn rewards daily

Claim your scratch cards

Redeem your collected rewards

Unlock more rewards when you pay or shop

Explore all offers in one place

Revamp your home in style

Bedsheets, curtains & more

Home decoration

Home storage

Lighting solutions

Car & bike essentials | Up to 60% off

Cleaning accessories

Tyre & rim care

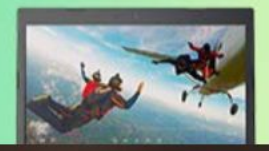
Helmets

Vacuum cleaner

Sign in for your best experience

Sign in securely

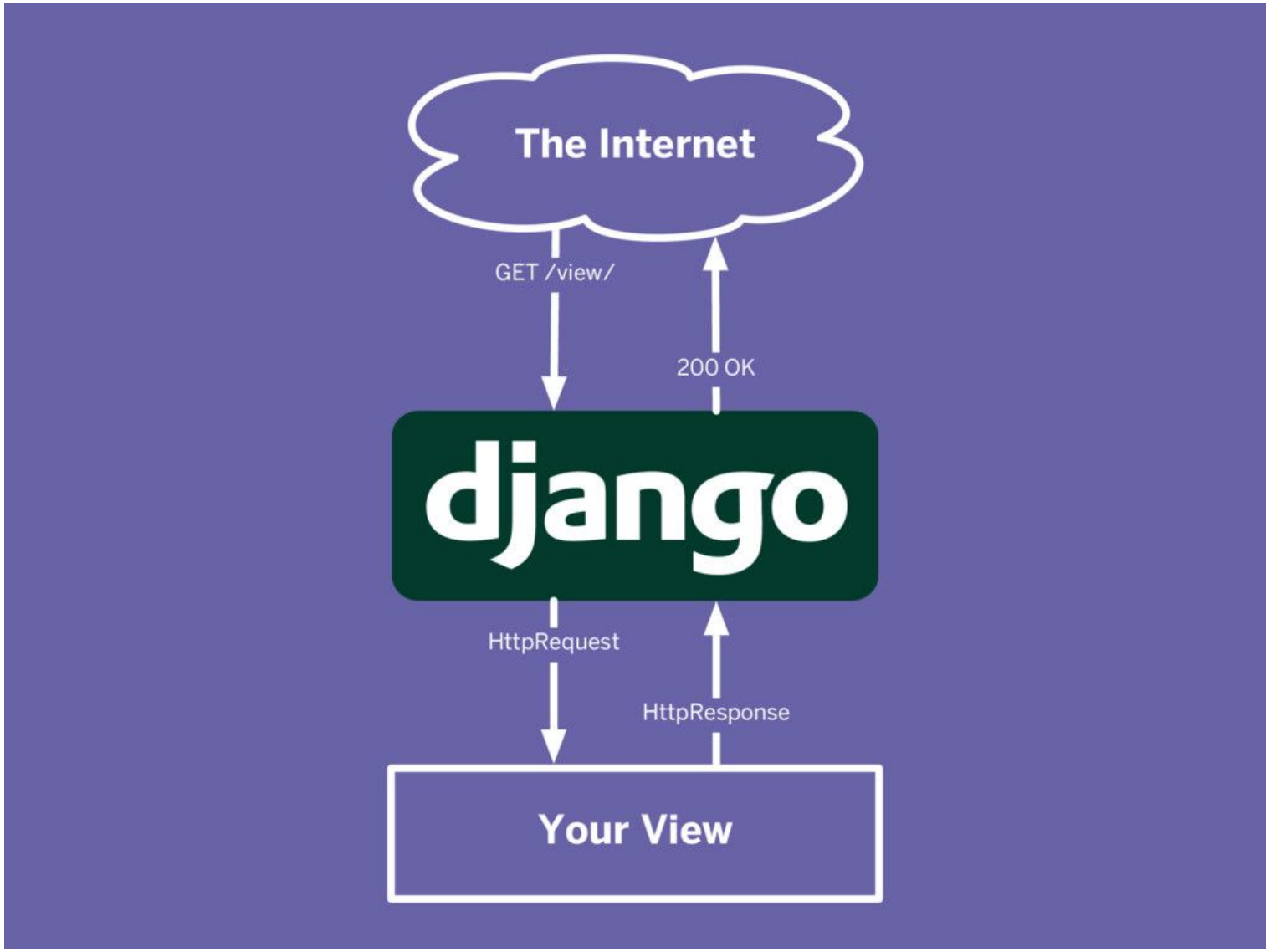
LAPTOPS FROM TOP BRANDS



Waiting for www.amazon.in...

Type here to search

ENG 19:11 29-10-2022


The Internet
GET /view/
200 OK
django
HttpRequest
HttpResponse
Your View

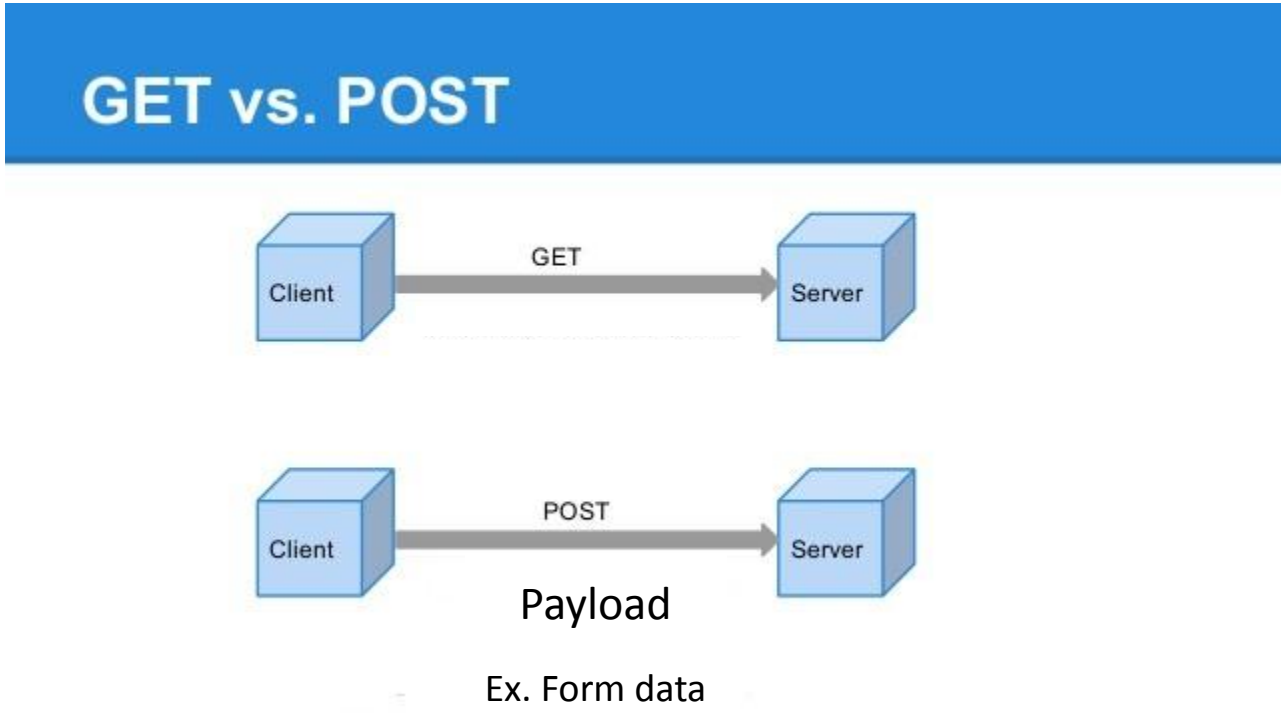
GET vs POST

The **GET** Method

GET is used to **request data** from a specified resource.

The **POST** Method

POST is used to **send data** to a server to create/update a resource.



Django App

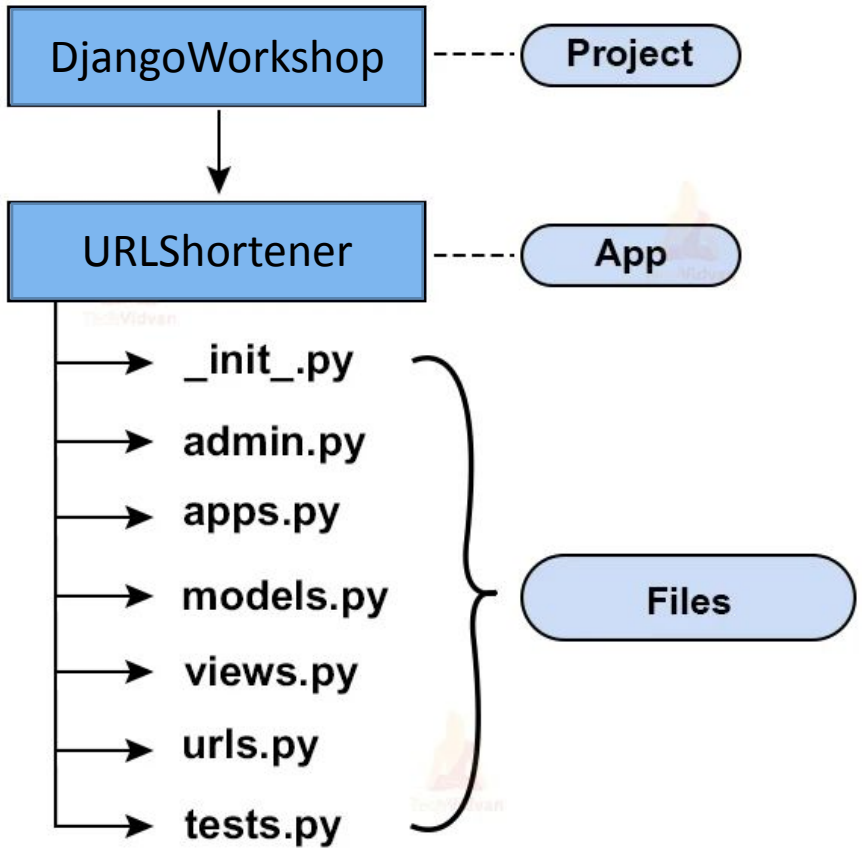
- Django uses the concept of Projects and apps for managing the codes and presents them in a readable format.
- A Django **project contains one or more apps** within it
- For example - a real-world Django e-commerce site will have one app for user authentication, another app for payments, and a third app for item listing details
- Each app will focus on a **single functionality**

Create new app

The next step is creating your app by using the command

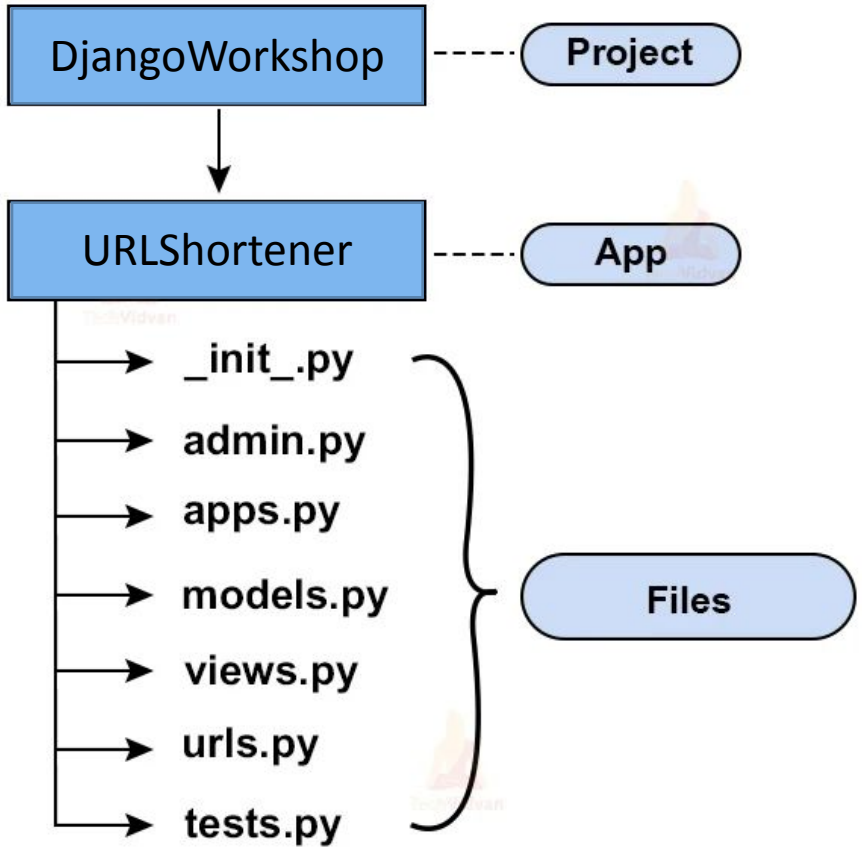
django-admin startapp url_shortener

App folder structure



- **admin.py**: Used for registering the Django models into the Django administration.
- **apps.py:** Used to help the user include the application configuration for their app.
- **models.py:** Define the structure of the database. It tells about the actual design, relationships between the data sets, and their attribute constraints

App folder structure

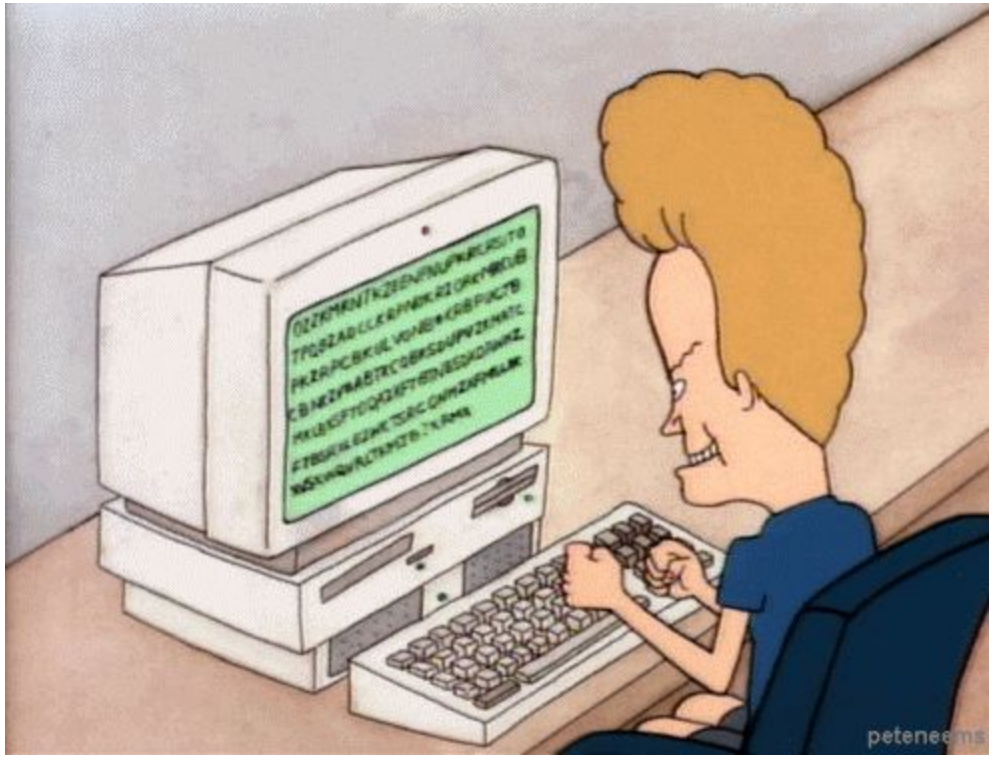


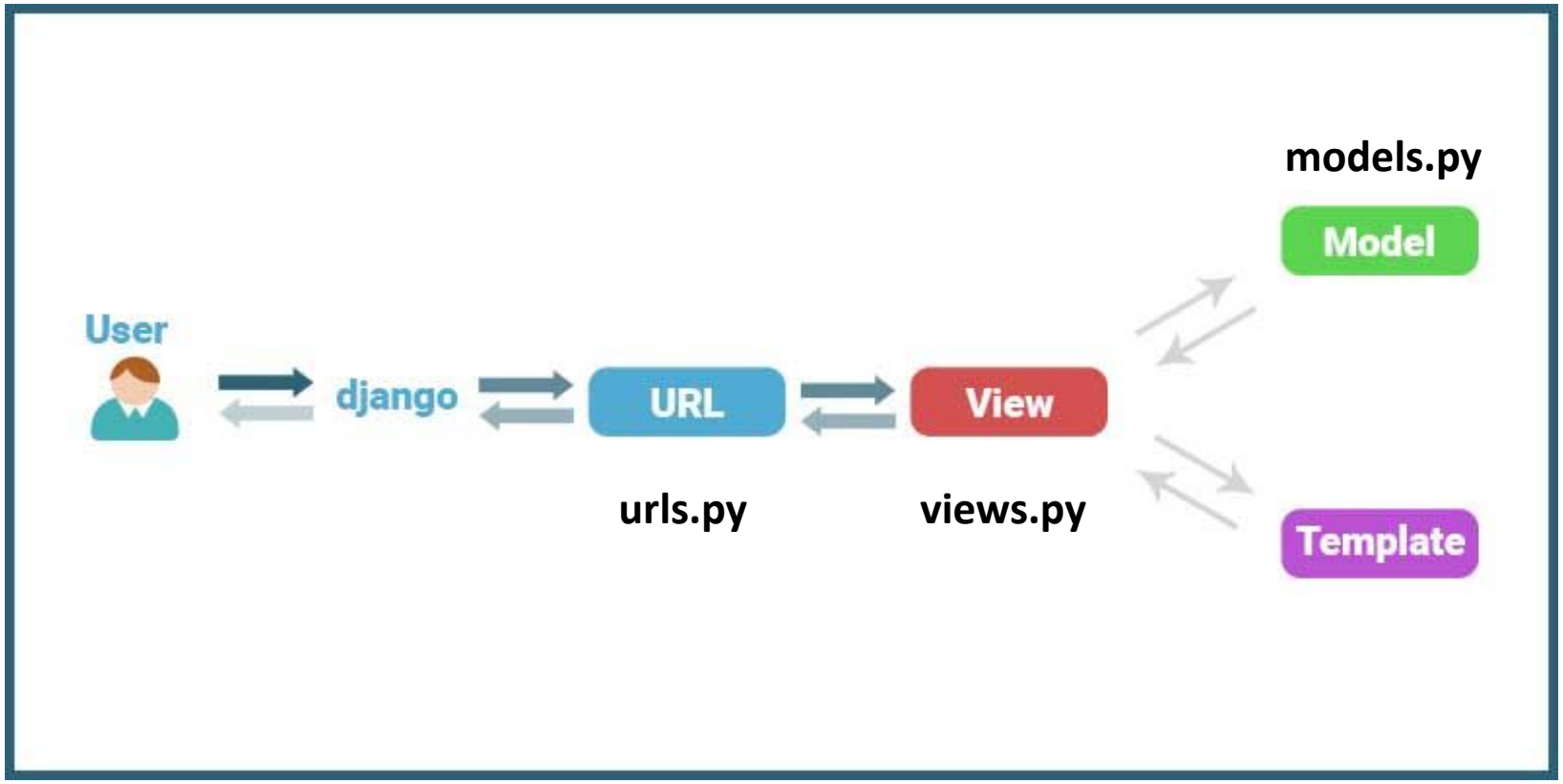
- **views.py**: Views provide an interface through which a user interacts with a Django web application

- **urls.py:** Works the same as that of the urls.py in the project file structure. The primary aim being, linking the user's URL request to the corresponding pages it is pointing to

- **tests.py:** Allows the user to write test code for their web applications

Let's get coding





Here, a user requests for a resource to the Django, Django works as a controller and check to the available resource in URL.

If URL maps, a view 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.

Write our first View

```
from django.http import HttpResponse


def test(request):
    return HttpResponse("Hello, world.")
```

Django Template

- Templates are the third and most important part of Django's MVT Structure.
- A template in Django is basically written in HTML, CSS and JavaScript in an **.html file**.
- Django framework efficiently handles and generates **dynamic HTML web pages** that are visible to end-user.
- Django mainly functions with a backend so, in order to provide frontend and provide a layout to our website, we use templates.

Handle Templates

Templates folder is used to put all the HTML files related to the project.

The default creation of the project doesn’t give this folder, so we have to create this folder in the **app directory**.

The Django template language

Variables

A variable outputs a value from the context, which is a dict-like object mapping keys to values.

Variables are surrounded by **{{** and **}}** like this:

```
My first name is {{ first_name }}. My last name is {{ last_name }}.
```

With a context of **{'first_name': 'John', 'last_name': 'Doe'}**, this template renders to:

```
My first name is John. My last name is Doe.
```

The Django template language

For Loop

```
{% for i in list %}
    // statements
{% endfor %}
```

If else

```
{% if variable %}
    // statements
{% else %}
    // statements
{% endif %}
```

Beautify Site (Static Files)

- **static** folder is used to put all the css, js, images, etc files related to the project.
- The default creation of the project doesn’t give this folder, so we have to create this folder in the **app directory**.

How to Load and use Static Files?

```
{% load static %}
<img src="{% static 'my_app/example.jpg' %}" alt="My image">
```

Small Task

- Step 1 : Create a new html page named “task.html”
- Step 2 : Create a basic css file named “style.css”
- Step 3 : We need a new url localhost:8000/url/task.
- Step 4 : We need a view which will respond with task.html
- Step 5 : We need dynamic data on our html page

Slug in Django

- A slug is a short label for something, containing only letters, numbers, underscores or hyphens.

- They’re generally used in URLs.

```
https://www.djangoproject.com/weblog/2008/apr/12/spring/
```

the last bit (**spring**) is the slug.

Defining URL Path For Slug Capture

```
urlpatterns = [
    path('<slug:name>', views.show_name)
]
```

Using The Captured Slug

```
def show_name(request, name):
    return HttpResponse("My name is", name)
```

Django Forms

- In HTML, a form is a collection of elements inside **<form>...</form>** that allow a visitor to send information to the server

- We then collect this information in our **View**
- Process it according to our logic
- Perform database operations (if needed)
- Send **response** back to frontend

QUESTIONS?