Create your Django app in which after running the server, you should see on the browser, the text “Hello! I am learning Django”, which you defined in the index view

Create a Django Project

On the command prompt run the following command to create a new Django project.

django-admin startproject <myproject>

Create a Django App

Navigate to your project directory and create a new Django app. Replace "myapp" with your desired app name:

cd <myproject>

python manage.py startapp <myapp>

Define a View

Open the views.py file inside your app directory (e.g., myapp/views.py) and define a view that will return the "Hello! I am learning Django" text:

```
from django.http import HttpResponse

def index(request):
   return HttpResponse("Hello! I am learning Django")
```

Create a URL Mapping

In the same app directory, create a urls.py file (e.g., myapp/urls.py) if it doesn't exist, and define a URL mapping for your view:

```
from django.urls import path

from . import views


urlpatterns = [
   path('', views.index, name='index'),
]
```

Configure the Project URL Routing

Open the project's urls.py file (e.g., myproject/urls.py) and include the URL patterns of your app:

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
   path('admin/', admin.site.urls),                    //already existing in the urls.py file
   path('', include('myapp.urls')),                    //add line for url routing
]
```

Run the Development Server

```
python manage.py runserver
```

Open your web browser and go to **http://127.0.0.1:8000/.**

Design a Django application that adds web pages with views and templates.

Create a New Django Project and App

Create a new Django project

```
django-admin startproject myproject
```

Create a new Django app

```
cd myproject
python manage.py startapp myapp
```

Define Views for Each Page

Inside your app directory (e.g., myapp), open the views.py file. Define a view function for each web page you want to create. For example, let's create two pages: "home" and "about":

```
from django.shortcuts import render
def home(request):
   return render(request, 'myapp/home.html')
def about(request):
   return render(request, 'myapp/about.html')
```

Create Templates

Create HTML templates for each page in your app's template directory. By default, Django expects templates to be located in myapp/templates/myapp/. Create two HTML files named home.html and about.html inside this directory:

myapp/templates/myapp/home.html:

```
<!DOCTYPE html>
<html>
<head>
    <title>Home Page</title>
</head>
<body>
    <h1>Welcome to the Home Page</h1>
    <p>This is the home page of our website.</p>
</body>
</html>
```

myapp/templates/myapp/about.html:

```
<!DOCTYPE html>
<html>
<head>
    <title>About Us</title>
</head>
<body>
    <h1>About Us</h1>
    <p>We are a team of developers learning Django.</p>
</body>
</html>
```

Create URL Mappings

In your app directory, create a myapp/urls.py file (if it doesn't exist) and define URL patterns for the views:

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

```
urlpatterns = [
    path('', views.home, name='home'),
    path('about/', views.about, name='about'),
]
```

Configure Project URL Routing

In your project's urls.py file (e.g., myproject/urls.py), include the URL patterns from your app:

```
from django.contrib import admin
from django.urls import path, include

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

Run the Development Server

Start the development server with the following command:

```
python manage.py runserver
```

Access the Web Pages

Home Page: http://127.0.0.1:8000/

About Page: http://127.0.0.1:8000/about/

Write and run Django code to add data to your site using relational databases with Django's Object Relational Mapper. - We will create a simple Django app to manage a list of books in a library.

Create a New Django Project and App

Create a new Django project

```
django-admin startproject library_project
```

Create a new Django app

```
cd library_project
python manage.py startapp library
```

Define a Model

Open the models.py file inside your app directory (library/models.py) and define a model for the book:

```
from django.db import models

class Book(models.Model):
    title = models.CharField(max_length=100)
    author = models.CharField(max_length=100)
    published_date = models.DateField()

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

This model represents a book with title, author, and published date.

Create Migrations

Generate migrations for your model using the following commands:

```
python manage.py makemigrations library
python manage.py migrate
```

This will create the necessary database tables based on your model.

Create an Admin Interface (Optional)

If you want to manage your data through the Django admin interface, you can register your model in the admin.py file (library/admin.py):

```
from django.contrib import admin
from .models import Book

admin.site.register(Book)
```

Add Data

You can add data to your site using the Django shell. Start the shell with the following command:

```
python manage.py shell
```

Then, you can create and save instances of the Book model:

```
from library.models import Book
```

```
from datetime import date

# Create a new book

book1 = Book(title='Django for Beginners', author='William S. Vincent',
published_date=date(2020, 1, 1))

book1.save()


# Create another book

book2 = Book(title='Python Crash Course', author='Eric Matthes',
published_date=date(2015, 11, 1))

book2.save()
```

Retrieve Data

You can retrieve data from your database using Django's ORM. For example, to retrieve all books, you can do the following:

```
all_books = Book.objects.all()

for book in all_books:

   print(book.title, book.author, book.published_date)
```

Run the Development Server

```
python manage.py runserver
```

Access the Admin Interface (Optional)

If you registered your model in the admin interface, you can access it by going to **http://127.0.0.1:8000/admin/** in your browser and log in using the admin credentials.

Creating a Django app for a public site where users can pick their favorite programming language and vote involves several steps. Here are detailed instructions to create such an app:

Create a New Django Project and App

```
# Create a new Django project

        django-admin startproject programming_language_poll

# Create a new Django app

        cd programming_language_poll

        python manage.py startapp poll
```

Define a Model for Programming Languages

Open the models.py file inside your app directory (poll/models.py) and define a model to represent programming languages and their vote counts:

```
from django.db import models

class ProgrammingLanguage(models.Model):
    name = models.CharField(max_length=100)
    votes = models.PositiveIntegerField(default=0)

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

Create Migrations

Generate migrations for your model using the following commands:

```
python manage.py makemigrations poll
python manage.py migrate
```

This will create the necessary database tables based on your model.

Create Views and Templates

In your app directory, create views for displaying the list of programming languages and handling the voting process in views.py (poll/views.py):

```
from django.shortcuts import render, get_object_or_404, redirect
from .models import ProgrammingLanguage

def index(request):
    languages = ProgrammingLanguage.objects.all()
    return render(request, 'poll/index.html', {'languages': languages})

def vote(request, language_id):
    language = get_object_or_404(ProgrammingLanguage, pk=language_id)
    language.votes += 1
    language.save()
```

```
    return redirect('poll:index')
```

Create HTML templates for displaying the list of languages and voting buttons in your app's template directory (poll/templates/poll/). Here's a basic template for index.html:

```
<!DOCTYPE html>
<html>
<head>
    <title>Favorite Programming Language Poll</title>
</head>
<body>
    <h1>Vote for Your Favorite Programming Language</h1>
    <ul>
        {% for language in languages %}
            <li>{{ language.name }} - Votes: {{ language.votes }} <a href="{% url 'poll:vote' language.id %}">Vote</a></li>
        {% endfor %}
    </ul>
</body>
</html>
```

Create URL Mappings

Define URL patterns for your views in urls.py (poll/urls.py):

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

app_name = 'poll'
urlpatterns = [
    path('', views.index, name='index'),
    path('<int:language_id>/vote/', views.vote, name='vote'),
]
```

Configure Project URL Routing

In your project's urls.py file (programming_language_poll/urls.py), include the URL patterns from your app:

```
from django.contrib import admin
from django.urls import path, include


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

Run the Development Server

```
python manage.py runserver
```

Access the Site: http://127.0.0.1:8000/poll/
