The project folder has certain files

Manage.py, it is used to run management commands

Settings.py, contains project configuration

\_\_init\_\_.py, this tells that this folder is python package

Urls.py, used to map the routes and paths in your project

Wsgi.py, gateway interface for deployment

**Django philosophy**

App, web app does something; it is usually composed of models, views, templates and tests

Project, it is a collection of apps

You can’t run a Django app without a project

**Stuff inside the app directory**

Migrations, keep track of the changes done to models such that it can keep the database and models synchronized

Admin.py, configuration for built in Django admin app

Apps.py, config file for the app

Models.py, the models are translated into Django tables by default, here you can define those entities

Test.py, used to write unit tests for the app

Views.py, this is the file where we handle the request/response cycle of the web application

Django app already comes with default apps, to add additional app you will have to add that to the settings page

**Views**

Views are python functions that receive the http request and returns the http response

**URLS**

After you have created a view, you need to tell Django when to serve it

You do that by modifying the urls page

Regex to match an empty string ^$, this matches the empty string which means the home page

If you wanted to match the home page you could have written ^homepage$

**Model**

Each class in the model is represented as a database table

Create relationship between models using the foreign key that will take care of the database relationship as well

Related\_name will create a reverse relationship

To ignore the reverse relationship you can make it related\_name='+'

Django will automatically create the primary key for us

Once the model is created, next step is to move that to the database

You will have to add this to the all the related models

on\_delete=models.CASCADE

To inspect the sql of migrations

**python manage.py sqlmigrate boards 0001**

All the work here is done using Django ORM which is a layer of abstraction

That communicates with the database

Stopped at experimenting with the Models API

Every model comes with a special attribute called the **model manager**

It is mainly used to execute queries in the database

It can be accesses via the objects

List all the objects, Board.objects.all()

\_\_str\_\_ can be defined in each model to get the string representation of an object

To list all the model objects

boards\_list = Board.objects.all()

To list specific model object

django\_board **=** Board**.**objects**.**get(id**=**1)

Board**.**objects**.**get(name**=**'Django')

Get method usage

|  |  |
| --- | --- |
| Create an object without saving | board = Board() |
| Save an object (create or update) | board.save() |
| Create and save an object in the database | Board.objects.create(name='...', description='...') |

Stopped at Django Template Engine Setup

Django test, automate the **test suite**

Success code is 200 and error code is 500

Add more tests, cases and check if you are getting the right view for each of the url route

To see more details of the test case, you can set the verbosity

**python manage.py test --verbosity=2**

**Boot Strap** is a toolkit that is used to develop HTML, CSS and Javascript

Stopped at refreshing the page **127.0.0.1:8000** we can see it worked:

Django **Admin** tool is there to manage the site

Before using it, you will have to create an admin account

|  |
| --- |
| When a request is made, Django will check the url file to find the appropriate view to deliver  If it does not find a match it throws 404 exception |

View function is used to process the request

Stopped at basic URL

The order in which you look for the URL is important, because the URL mentioned below may never get executed, so be careful with that

Basically the URL that catches a lot has to go later

You can capture the data from the url, like so (?P<pk>\d+)

Stopped at Using the URLs API