Django Workflow

Django is a open source , web framework (A framework is just a collection of modules) built entirely in python and it follows Model View Tempelate Architecture . Django is fast and powerful web framework . Django is preferred because of it's rich library which contains Django REST framework , being responsible for building APIs and Django CMS, which helps in managing the website content .

1. Create a new project and a new app called mainapp.

2. Edit settings.py. Fix the following:
    1. Database settings
    2. Installed apps
    3. Media and media admin paths
    4. Most recently, to add CSS and JS paths

3. Edit urls.py in the project folder and
    1. Uncomment the admin paths
    2. Include mainapp.urls.py

4. Create a new urls.py in the mainapp folder
    1. Create a path that display a view called home

5. Create a new templates folder

6. Create a base.html file in the templates folder
    1. Create a place-holder variable within base.html

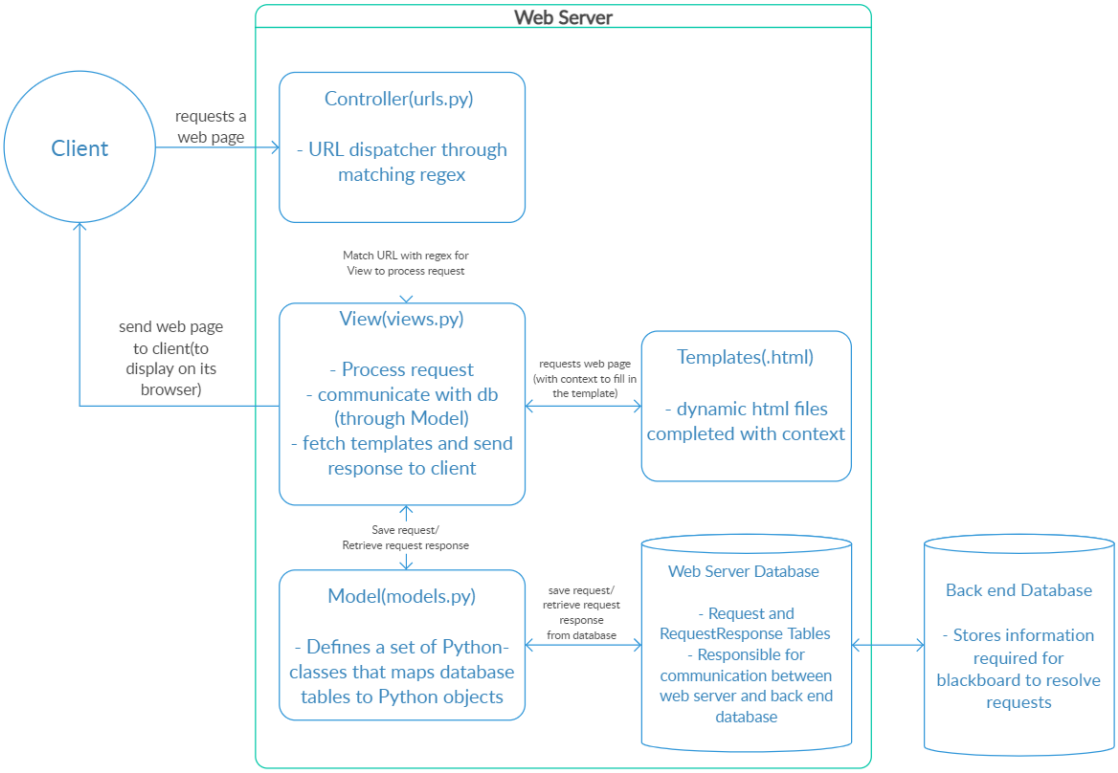
7. Edit the views.py file. Create a new view called home and return 'hello world' using the base.html template and a place-holder variable

8. Create a media folder the project folder (1 level up from the mainapp folder)

9. Edit the models.py file

10. Run syncdb

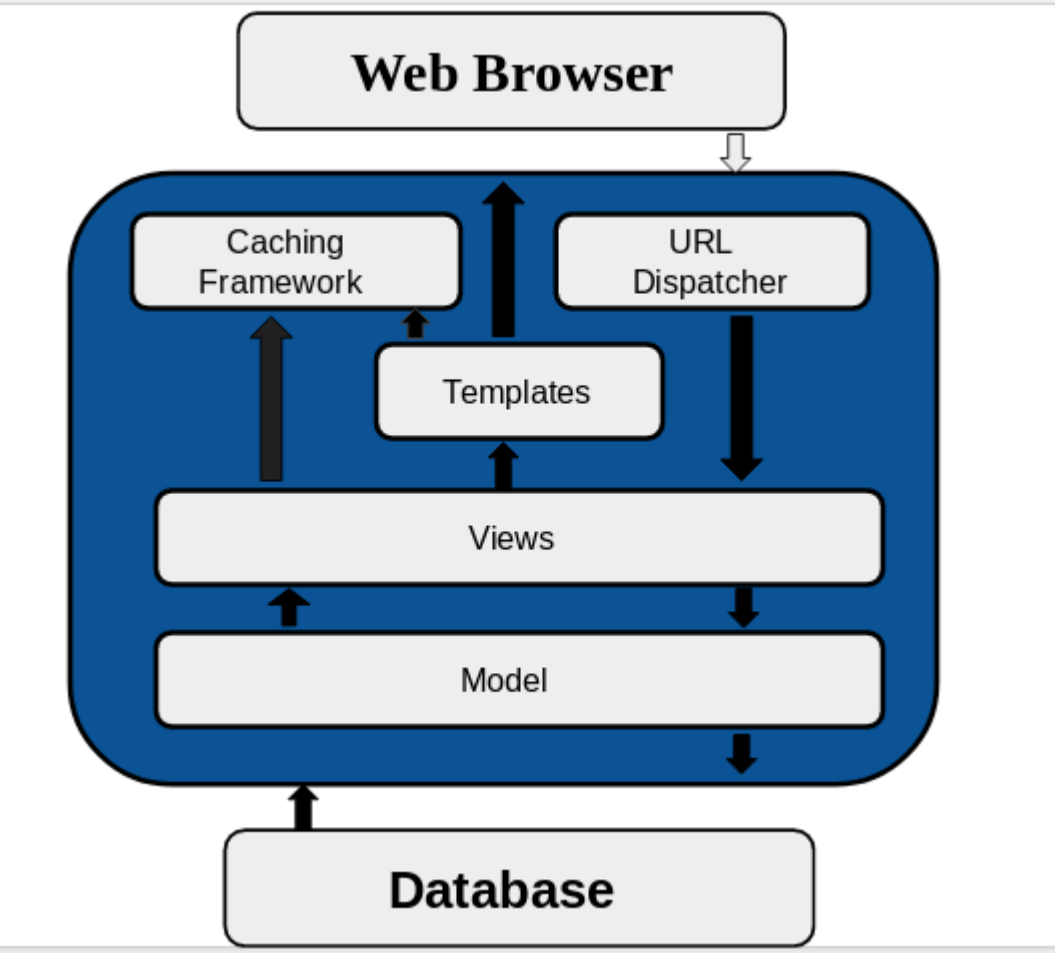
11. At this point, you should be able to access both your admin page and your homepage that outputs 'hello world' by running the development server.



Lessons Learnt

Django's strength is also it's weakness. It's easy to get a new project going but changing the database structure is a pain. You can't add new columns to an existing table in your models.py file and expect syncdb to update your database schema.

Let's Break down in Step by Step:



1. URL Dispatcher

Django gets user requests by URL locator and responds back to it. django.urls module is used by it to manage the URL's requests. For any application in Django we create Python Module named as URLconf(URL configuration) as per Django Documentation. This module is responsible for mapping between URL path expressions to some specified view.

Syntax of the module is path(route,view,kwargs,name)

//urls.py

```
from django.contrib import admin
from django.urls import path
urlpatterns = [
      path('admin/', admin.site.urls),
           ]
```

2. Views

A view is a part of your web application , which is a Python function responsible for accepting a web requests and returning corresponding web response to it. The response can be of HTML type containing HTML contents , or an image , or an XML document , or sometimes even errors like 404 error.

It is a part of our application where main logic of our application is written there ,remember that you have to link view to URL to see it as web page. For example

//view.py

```
from django.http import HttpResponse
def index(request):
return HttpResponse('<h1>Hello World</h1>')
```

//urls.py

```
from django.conf.urls import url
from app_name import views
urlpatterns = [
                url(r'^$', views.index ,name = 'index')
              ]
```

3. Model

Django model is a class which contains certain fields and methods and every attribute of class is field of database table. Django Model is a subclass of django.db.models.Model . Models are described in your respective applications as app_name/models.py where each model class is mapped to single table of your database . Whatever field you create in your model it will added to your database automatically , thereby reducing extra line of codes for creating table in database.

One important point to notice here is ‘id’ field is created automatically which changes with each record input by the user.

For example

```
from django.db import models
class College(models.Model):
first_name = models.CharField(max_length=50)
last_name = models.CharField(max_length=50)
```

Here first_name and last_name are respective class attributes and they are mapped to database as their attributes with ‘id’ as additional one.

```
CREATE TABLE app_college ("id" INT NOT NULL PRIMARY
KEY,"first_name" varchar(50) NOT NULL,"last_name" varchar(50)
NOT NULL);
```

After completion of your Model don’t forget to register in INSTALLED_APPS which is present in settings.py.

4. Template

Django provides template system which helps to distinguish between python and html contents. Dynamic HTML pages are generated by using template system. Here I have quoted how to write HTML content and thereby returns HttpResponse but before that you have to understand what does Render function do and what are their arguments?

Render function basically links your python code to HTML with following as arguments

Request

Path to your template

Dictionary of parameters

//view.py

```
from django.http import HttpResponse
from django.template import loader
def index(request):
template = loader.get_template('index.html')
context = {}
return HttpResponse(template.render(context , request))
```

//index.html

```
<!DOCTYPE html>
<html>
<head>
<title>My first Program</title>
```

```
</head>
<body>
<h1>Hello World</h1>
</body>
</html>
```

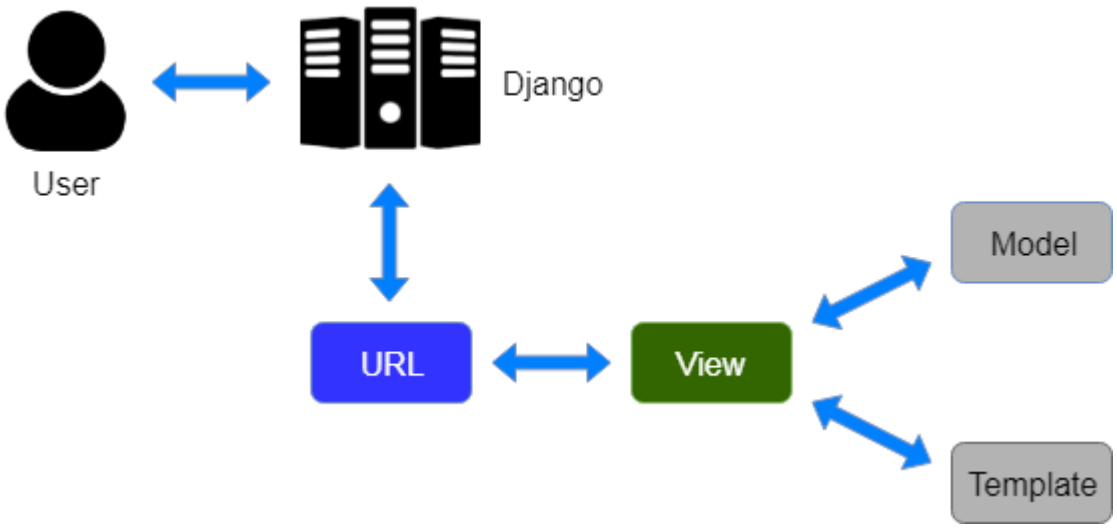
After writing this run following command in your terminal

python3 manage.py runserver

Then run http://127.0.0.1:8000 in browser

Output — Hello World

So this is how our Model View Template Architecture works



5. Caching Framework

For dynamic webpages , when user requests a particular page , then all following sort of queries takes place like from database queries to template rendering to business logic to make the requested page visible to user. Sometimes it is expensive. To resolve this task caching plays an important role.

To cache something means to save the result of an expensive calculation so that you don’t have to perform the calculation next time. Here is a pseudo code as per Django Documentation

```
given a URL, try finding that page in the cache
if the page is in the cache:
 return the cached page
else:  generate the page  save the generated page in the cache
(for next time)
return the generated page.
```

So this was the workflow of Django , i.e how these components interact with each other and makes Django a powerful Web Development Framework.