Django Cookbook

1. Installation of Django

1. Check the version of Python

```
C:\Users\MRS. LIFNA>python --version
Python 3.6.4
```

2. Install pipenv to handle Application dependencies

```
C:\Users\MRS. LIFNA>pip3 install pipenv
```

3. Install Visual Studio Code from https://code.visualstudio.com/

4. Create a folder for Django Demos (named : DjDemo)

```
C:\Users\MRS. LIFNA\Desktop>mkdir DjDemo

C:\Users\MRS. LIFNA\Desktop>cd DjDemo
```

5. Create a Virtual Environment in this folder (named : env)

```
C:\Users\MRS. LIFNA\Desktop\DjDemo>python -m venv env

C:\Users\MRS. LIFNA\Desktop\DjDemo>dir
 Volume in drive C has no label.
 Volume Serial Number is 76BD-0438

 Directory of C:\Users\MRS. LIFNA\Desktop\DjDemo

03/15/2022  12:49 PM    <DIR>          .
03/15/2022  12:49 PM    <DIR>          ..
03/15/2022  12:49 PM    <DIR>          env
               0 File(s)              0 bytes
               3 Dir(s)  31,570,329,600 bytes free
```

6. Activate this Virtual Environment

```
C:\Users\MRS. LIFNA\Desktop\DjDemo\env\Scripts>activate.bat
(env) C:\Users\MRS. LIFNA\Desktop\DjDemo\env\Scripts>
```

Note: If required upgrade pip

```
(env) C:\Users\MRS. LIFNA\Desktop\DjDemo\env\Scripts>python -m pip install --upgrade pip
```

7. Install Django in the Virtual Environment

```
(env) C:\Users\MRS. LIFNA\Desktop\DjDemo\env\Scripts>python -m pip install django
```

8. Check the installation of Django (via django-admin)

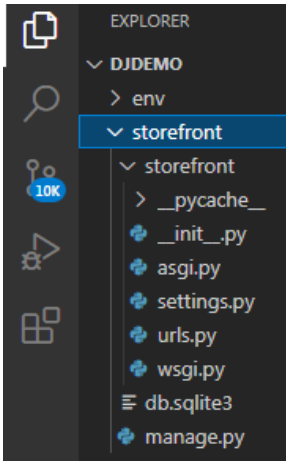
```
(env) C:\Users\MRS. LIFNA\Desktop\DjDemo\env\Scripts>django-admin
Type 'django-admin help <subcommand>' for help on a specific subcommand.
Available subcommands:
[django]
check                                                   runserver
    compilemessages                                     sendtestemail
    createcachetable                                    shell
    dbshell                                             showmigrations
    diffsettings                                        sqlflush
    dumpdata                                            sqlmigrate
    flush                                               sqlsequencereset
    inspectdb                                           squashmigrations
    loaddata                                            startapp
    makemessages                                        startproject
    makemigrations                                      test
    migrate                                             testserver
```

2. Create a Django Project

1. Start the Django Project in DjDemo folder (named : storefront)

```
(env) C:\Users\MRS. LIFNA\Desktop\DjDemo>django-admin startproject storefront
```

Open the project in Visual Studio Code



a. __init__.py : designates folder as a package
b. asgi.py : to manage deployment files
c. settings.py : Project settings
d. urls.py : Maps urls
e. wsgi.py : to manage deployment files
f. manage.py : to execute Django commands

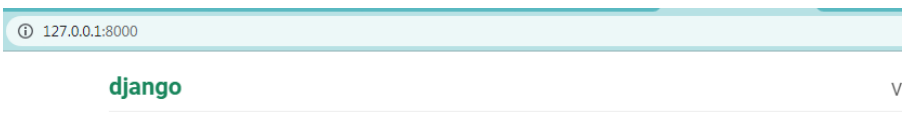
2. Run the Server (Go to the project - storefront)

```
(env) C:\Users\MRS. LIFNA\Desktop\DjDemo>cd storefront
```

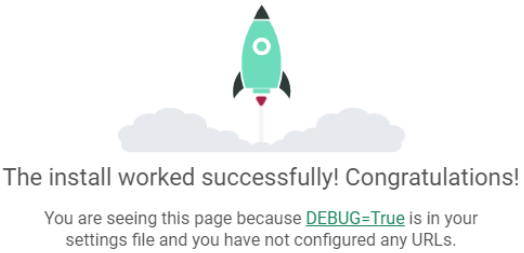
Note : By default, the Server runs in port 8000

```
(env) C:\Users\MRS. LIFNA\Desktop\DjDemo\storefront>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you
 apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
March 15, 2022 - 13:00:22
Django version 3.2.12, using settings 'storefront.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[15/Mar/2022 13:01:20] "GET / HTTP/1.1" 200 10697
[15/Mar/2022 13:01:24] "GET /static/admin/css/fonts.css HTTP/1.1" 200 423
[15/Mar/2022 13:01:24] "GET /static/admin/fonts/Roboto-Regular-webfont.woff HTTP
/1.1" 200 85876
[15/Mar/2022 13:01:24] "GET /static/admin/fonts/Roboto-Bold-webfont.woff HTTP/1.
1" 200 86184
[15/Mar/2022 13:01:24] "GET /static/admin/fonts/Roboto-Light-webfont.woff HTTP/1
.1" 200 85692
```

On the Web Browser:
http://127.0.0.1:8000/



To stop server : Ctrl + C
(On the Command Prompt)



3. Creating an App

Note : Project is a collection of Apps, each with a specific functionality

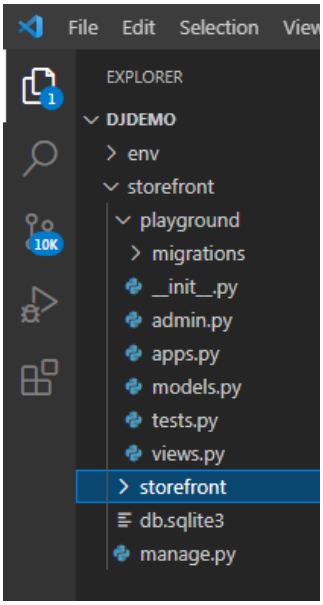
1. Check the Settings of the Project

a. Go to project, storefront in Visual Studio Code
b. Open settings.py
c. From the list of Installed Apps, delete sessions as it is deprecated.
d. Save settings.py

```
# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

2. Create an App (named : playground)

```
(env) C:\Users\MRS. LIFNA\Desktop\DjDemo\storefront>python manage.py startapp playground
```



A separate folder is create for the App - playground
Contents of the folder are :

a. migrations (folder) : for generating db tables
b. __init__.py : designates folder as package
c. admin.py : deines admin interfaces for the App
d. apps.py : to configure the Apps
e. models.py :
   i. to define model classes for the App.
   ii. model classe, pulls data from db and presents to user
f. tests.py : to write unit tests.
g. views.py : request Handler

3. Register the App created

```
# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'playground'
]
```

a. Go to project, storefront in Visual Studio Code
b. Open settings.py
c. In the list of Installed Apps, add - ‘playground’
d. Save settings.py

4. Creating Views for the App

Note : In Django,
Views (Request Handler) : accepts requests from users and return responses
Templates : something what user sees

1. Create a View

1. Go to, App folder - playground in Visual Studio Code
2. Open views.py
3. Add the following code to create a Request Handler for HttpResponse

```
from django.shortcuts import render
from django.http import HttpResponse

def say_hello(request):
    return HttpResponse('Hello World')
```

4. Save views.py

2. Map URLs to Views

Note : When the user types the urls, the corresponding view function needs to be invoked.

1. Go to, App folder - playground in Visual Studio Code
2. Create urls.py file
3. Add the following code. (Add the urlpatterns. Mapping url (route) to the views defined.)

```
from django.urls import path
from . import views                    // From current directory import views

# URLConfig
urlpatterns = [
    path('hello/',views.say_hello),
]
```

4. Save urls.py
5. Go to the Project folder - storefront in Visual Studio Code
6. Open urls.py
7. Add the following code. (Import App URLConfig into Project URLConfig)

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

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

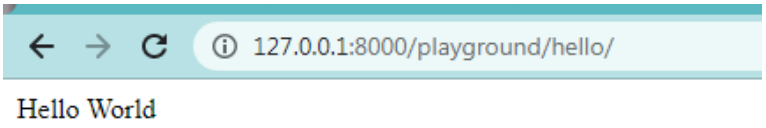
Note : Always end the route with a forward slash. Eg : playground/

8. Save urls.py

5. Checking the Created View

1. Run the Server to check the views

```
(env) C:\Users\MRS. LIFNA\Desktop\DjDemo\storefront>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 17 unapplied migration(s). Your project may not work properly until you
 apply the migrations for app(s): admin, auth, contenttypes.
Run 'python manage.py migrate' to apply them.
March 15, 2022 - 15:05:55
Django version 3.2.12, using settings 'storefront.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```



6. Creating Templates for Users in Django

1. Create a Template

1. Go to, App folder - playground in Visual Studio Code
2. Create a folder, templates
3. Create a file, hello.html
4. Add the following code

```
<h1>Hello World!!</h1>
```

5. Save hello.html

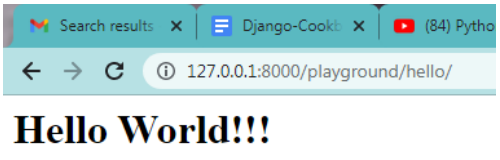
2. Update the view function to render the created Template, hello.html

1. Go to views.py in the App folder - playground
2. Edit the view function - say_hello to render, hello.html file on User Request

```
from django.shortcuts import render
from django.http import HttpResponse

def say_hello(request):
    return render(request,'hello.html')
```

3. Rerun the server to see the created Template, hello.html (rendered as per user request)

```
(env) C:\Users\MRS. LIFNA\Desktop\DjDemo\storefront>python manage.py runserver
```



7. Making the Templates dynamic

1. Go to views.py in the App folder - playground
2. Edit the view function - say_hello to render hello.html file on User Request by mapping objects that pass a string which is mapped to another object.

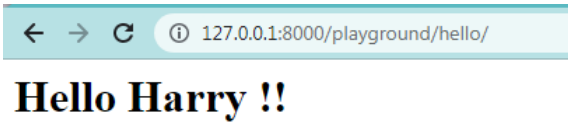
```
from django.shortcuts import render
from django.http import HttpResponse

def say_hello(request):
    return render(request,'hello.html',{ 'name' : 'Harry'})
```

3. Go to, App folder - playground in Visual Studio Code
4. Edit hello.html file
5. Add the following code

```
<h1>Hello {{ name }} !!</h1>
```

6. Rerun the server to see the created Template, hello.html (rendered as per user request)

   (env) C:\Users\MRS. LIFNA\Desktop\DjDemo\storefront>python manage.py runserver



8. Let's write some Logic in the Template

1. Go to, App folder - playground in Visual Studio Code
2. Edit hello.html file
3. Add the following code. (Here, if-else logic is written

```
{% if name %}
<h1>Hello {{ name }} !! </h1>
{% else %}
<h1>Hello World!!</h1>
{% endif %}
```

References:

1. https://docs.djangoproject.com/en/4.0/howto/windows/
2. Python Django Tutorial for Beginners
3. Django Web Framework
4. Your First Steps With Django: Set Up a Django Project
5. PPTs - Django