Django Framework

- Django is a web application framework written in Python programming language.
- It is based on MVT (Model View Template) design pattern.
- The Django is very demanding due to its rapid development feature.
- It takes less time to build application after collecting client requirement.

Features of Django

- Rapid Development
- Secure
- Scalable
- Fully loaded
- Versatile
- Open Source
- Vast and Supported Community

Creating your First Project in Django

1. Create a directory in which you wish to maintain all your Django projects
2. Go to terminal. Command for creating a project

   `django-admin startproject projectname`

   eg: `django-admin startproject FirstProject`

   Change the directory to your Project: `cd FirstProject`

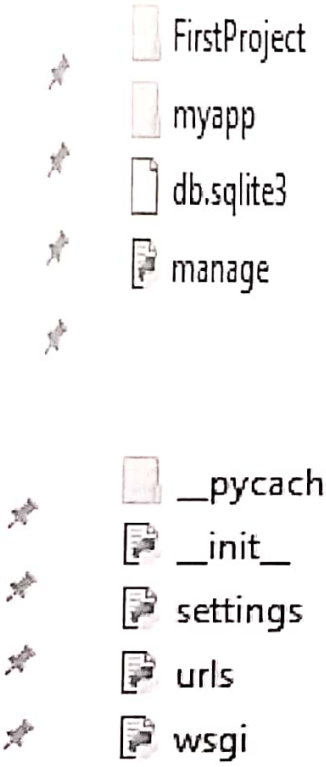
3. Command for creating an application under the Project

   `python3 manage.py startapp appname`

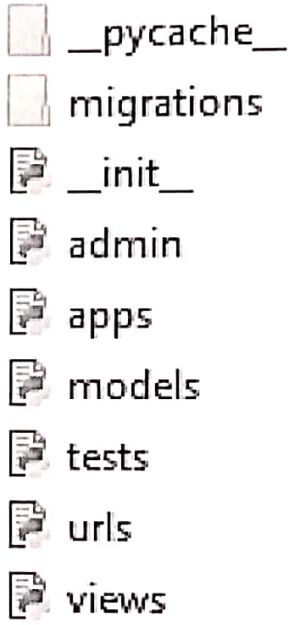
   eg: `python3 manage.py startapp myapp`

Directory structure of FirstProject

```
B:.
├───FirstProject
│   └───__pycache__
└───myapp
    ├───migrations
    └───__pycache__
```



Directory structure of First Project



Directory Structure of myapp

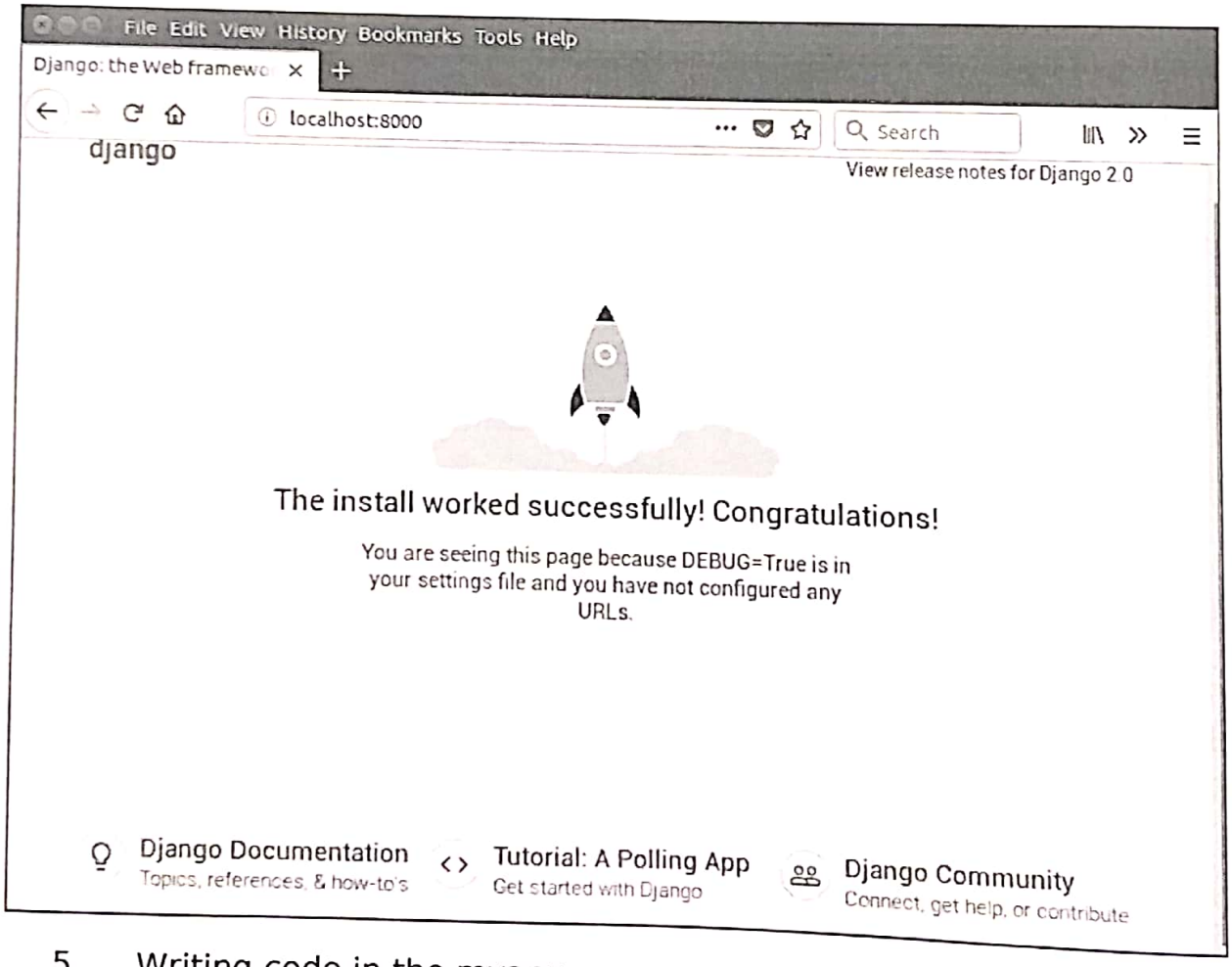
- **manage.py:** It is a command-line utility which allows us to interact with the project in various ways and also used to manage an application that we will see later on in this tutorial.
- A directory (djangpapp) located inside, is the actual application package name. Its name is the Python package name which we'll need to use to import module inside the application.

- **__init__.py:** It is an empty file that tells to the Python that this directory should be considered as a Python package.
- **settings.py:** This file is used to configure application settings such as database connection, static files linking etc.
- **urls.py:** This file contains the listed URLs of the application. In this file, we can mention the URLs and corresponding actions to perform the task and display the view.
- **wsgi.py:** It is an entry-point for WSGI-compatible web servers to serve Django project

4. Running the Django Project. Use command

```
python3 manage.py runserver
```

The server has started and can be accessed at localhost with port 8000. Let's access it using the browser.



5. Writing code in the myapp

Code of myapp/view.py

```
from django.http import HttpResponse
def index(request):
    return HttpResponse("Welcome to the Django Project")
```

Create an urls.py file in myapp directory

Myapp/urls.py

```
from django.urls import path

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

Now access the urls.py file under the FirstProject

FirsProject/urls.py

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

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

Run the server : `python3 manage.py runserver`

Now run the Application in the Browser: http://localhost:8000/myapp/

Hello, Welcome to Django!

Database Application with Django Admin and Django Templates

Step 1: To Interact with Django Admin you need to migrate. It will perform migrations for admin, auth, super user.

Command: python3 manage.py migrate

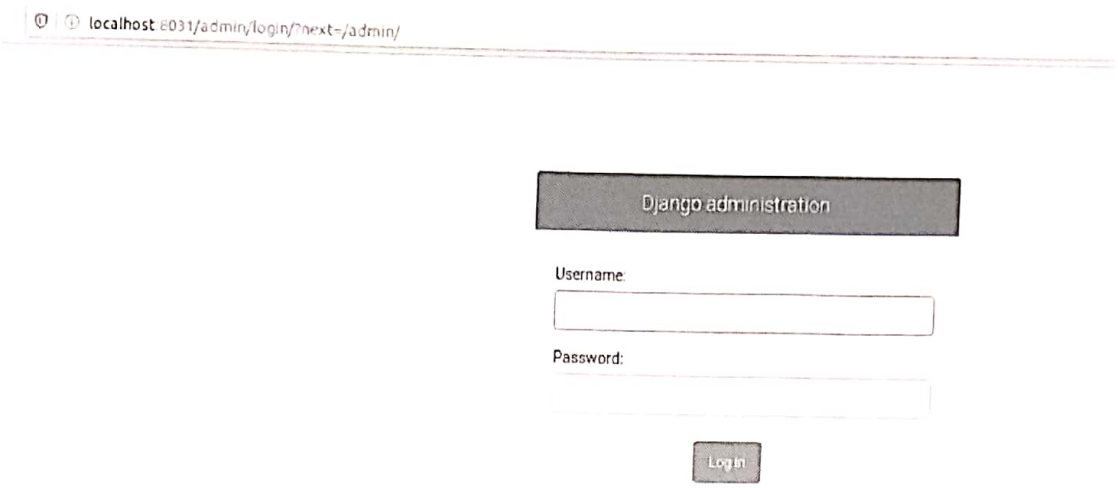
Then you need to create a super user for admin.

Command: python3 manage.py createsuperuser

Then Run the project using:

python3 manage.py runserver portnumber

Run the project on brower : http://localhost:8031/admin/



Step 2: Configuring templates and using Filters in Django

Create a folder name 'templates' in application directory

dboperations/templates/

After creating the directory you need to add the directory in settings.py file. Look for **'TEMPLATES'** tag.

database_project/settings.py file

```
TEMPLATES = [
  {
    'BACKEND': 'django.template.backends.django.DjangoTemplates',
    'DIRS': [os.path.join(BASE_DIR,'templates')],
    'APP_DIRS': True,
    'OPTIONS': {
      'context_processors': [
        'django.template.context_processors.debug',
```

```
            'django.template.context_processors.request',
            'django.contrib.auth.context_processors.auth',
            'django.contrib.messages.context_processors.messages',
        ],
```

Filters

They help you modify variables at display time. Filters structure looks like the following: {{var|filters}}.

Some examples –

- **{{string|truncatewords:80}}** – This filter will truncate the string, so you will see only the first 80 words.
- **{{string|lower}}** – Converts the string to lowercase.
- **{{string|escape|linebreaks}}** – Escapes string contents, then converts line breaks to tags.

You can also set the default for a variable.

Step 3: Now create a Project and create application by using commands

```
django-admin startproject database_project
python3 manage.py startapp dboperations
```

Step 4: Creating Models

For Creating tables in database Sqlite you need to create classes in models

dboperations/models.py file

```
from django.db import models

# Create your models here.

class Student(models.Model):
            first_name = models.CharField(max_length=20)
            last_name = models.CharField(max_length=30)
            contact = models.IntegerField()
```

```
        email = models.EmailField(max_length=50)

class Course(models.Model):

    student = models.ForeignKey(Student,on_delete=models.CASCADE)

    name = models.CharField(max_length=30)
```

Step 5: Register the Models

You need to register the classes made in models.py file to Admin.

dboperations/admin.py file

```
from django.contrib import admin

# Register your models here.

from dboperations.models import Student

from dboperations.models import Course

admin.site.register(Student) # Student is registered

admin.site.register(Course)  # Course is registered
```

Step 6: Activating Models

For activating the models you need to do changes in db_project/settings.py file

To include the app in our project, we need to add a reference to its configuration class in the **INSTALLED_APPS setting.** The DboperationsConfig class is in the dboperations/apps.py file, so its dotted path is 'dboperations.apps.DboperationsConfig'. Edit the mysite/settings.py file and add that dotted path to the INSTALLED_APPS setting. It'll look like this:

db_project/settings.py file

```
INSTALLED_APPS = [

'dboperations.apps.DboperationsConfig',

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',
```

```
    'django.contrib.staticfiles',
]
```

Step 7: To make the changes in the sqlite database you need to perform migrations

```
python3 manage.py makemigrations dboperations

python3 manage.py migrate
```

Step 8: Create View, Template, Urls files

dboperations/views.py file

```
from django.shortcuts import render

#importing loading from django template

from django.template import loader

from dboperations.models import Student, Course


# Create your views here.

from django.http import HttpResponse

def index(request):

  # template = loader.get_template('index.html') # getting our template

  # return HttpResponse(template.render())       # rendering the
template in HttpResponse

        students = Student.objects.all()

        course = Course.objects.all()

return render(request,"index.html",
{'students':students,'course':course})
```

Create a template index.html in templates folder

dboperations/templates/index.html file

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Student Records</title>
```

```
</head>
<body>
        <h2>Student Records</h2>
<table border = 1>
       <tr>
        <th>Student ID</th>
        <th>Student First Name</th>
        <th>Student Last Name</th>
        <th>Student Contact</th>
        <th>Student Email</th>

    </tr>
                {% for student in students %}
    <tr>
        <td>{{ student.id }}</td>
        <td>{{ student.first_name }}</td>
        <td>{{ student.last_name }}</td>
        <td>{{ student.contact }}</td>
        <td>{{ student.email }}</td>
   {% endfor %}
</tr>
</table>
<h2> Course Details </h2>
<table border = 1>
 <tr>
       <th>Student ID</th>
       <th>Course Name</th>
       </tr>
<tr>
    {% for c in course %}
```

```
      <td> {{c.student}}</td>
      <td> {{c.name}} </td>

  </tr>
   {% endfor %}
</table>
  <br>
<br>
</body>
</html>
```

dboperations/urls.py file

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

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

db_project/urls.py file

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

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

Step 9: Run the Server and Run Application on Browser

Run the server:python3 manage.py runrver 8050

Now you can perform CRUD operations directly using Django Admin.

localhost:8050/admin/

Now you can run the applicaiton on the browser by using templates and view the results from the database.

http://localhost:8050/dboperations/

localhost:8050/dboperations/

Student Records

Student ID	Student First Name	Student Last Name	Student Contact	Student Email
3	Naman	Gupta	1254785125	naman@gmail.com
4	Bansri	Shukla	1258741266	bansri@gmail.com

Course Details

Student ID	Course Name
Student object (3)	MCA
Student object (4)	BCA