Template

A template is a text file. It can generate any text-based format (HTML, XML, CSV, etc.).

A template contains variables, which get replaced with values when the template is evaluated, and tags, which control the logic of the template.

Template is used by view function to represent the data to user.

User sends request to view then view contact template afterthat view get information from the template and then view gives response to the users.

Create Template Folder and Files

We create **templates** folder inside Project Folder. **templates** folder will contain all html files.

```
geekyshows
    templates      <-- This is templates Folder.
    geekyshows
        __init__.py
        settings.py
        urls.py
        wsgi.py
    manage.py
    course
    fees
```

```
geekyshows
    templates
        courseone.html   }
        coursetwo.html   }  Template files
        feesone.html     }
        feestwo.html     }
geekyshows
        __init__.py
        settings.py
        urls.py
        wsgi.py
    manage.py
    course
    fees
```

Add Templates in settings.py

```
geekyshows
    templates
        courseone.html
        coursetwo.html
        feesone.html
        feestwo.html        geekyshows
        __init__.py
        settings.py
        urls.py
        wsgi.py
    manage.py
    course
    fees
```

settings.py

Old Version Django 3.0:
TEMPLATES_DIR = os.path.join(BASE_DIR, 'templates')

```
TEMPLATES_DIR = BASE_DIR / 'templates'

INSTALLED_APPS = [
    'course',
    'fees'
]

TEMPLATES = [
   {
    'DIRS': [TEMPLATES_DIR],
   }
]
```

Django Version 3.1

Directories where the engine should look for template source files, in search order.

Geeky Steps

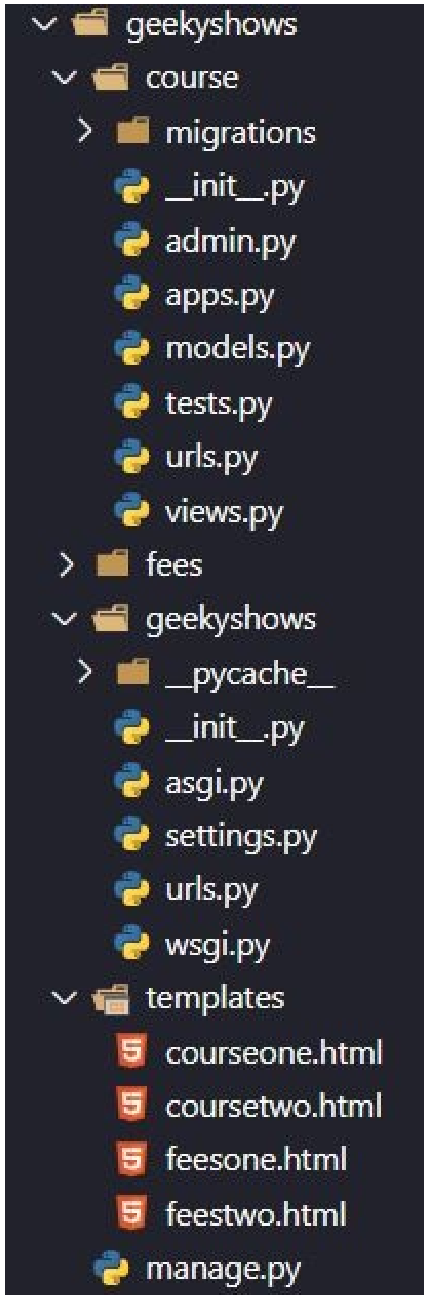
- Create Django Project: *django-admin startproject geekyshows*
- Create Django Application1: *python manage.py startapp course*
- Create Django Application2: *python manage.py startapp fees*
- Add/Install Applications to Django Project (course and fees to geekyshows) using settings.py INSTALLED_APPS
- Create **templates** folder inside Root Project Directory
- Add **templates** directory in settings.py

```
        TEMPLATES_DIR = os.path.join(BASE_DIR, 'templates')
        TEMPLATES_DIR = BASE_DIR / 'templates'
    TEMPLATES = [ {
                'DIRS': [TEMPLATES_DIR], } ]
```

Old Django Version 3.0

Django Version 3.1

- Create template files inside **templates** folder
- Write View Function inside views.py file
- Define url for view function of application using urls.py file



```
geekyshows
  course
    migrations
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    urls.py
    views.py
  fees
  geekyshows
    __pycache__
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py
  templates
    courseone.html
    coursetwo.html
    feesone.html
    feestwo.html
  manage.py
```

Write Templates Files

When we create Template file for application we separate business logic and presentation from the application *views.py* file.

Now we will write business logic in *views.py* file and presentation code in template file.

templates

courseone.html

```
<html>
      <head>
          <style> ………. </style>
     </head>
     <body>
           <h1>Hello Django</h1>
     </body>
     <script>…………</script>
</html>
```

templates
coursetwo.html

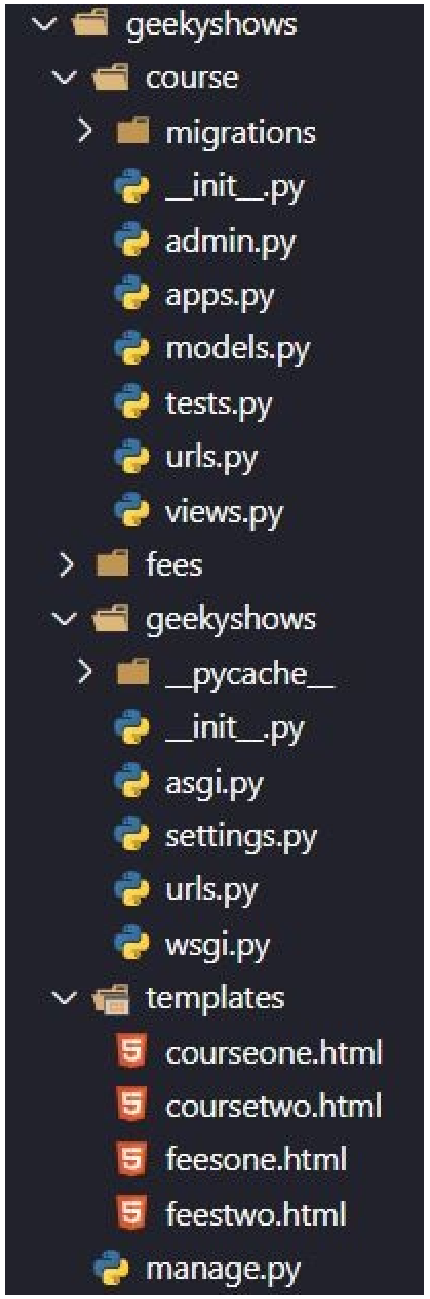
```
<html>
     <body>
          <h1>Hello Python</h1>
     </body>
</html>
```

templates
feesone.html

```
<html>
     <body>
          <h1>300</h1>
     </body>
</html>
```

templates
feestwo.html

```
<html>
     <body>
          <h1>200</h1>
     </body>
</html>
```



```
geekyshows
  course
    migrations
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    urls.py
    views.py
  fees
  geekyshows
    __pycache__
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py
  templates
    courseone.html
    coursetwo.html
    feesone.html
    feestwo.html
  manage.py
```

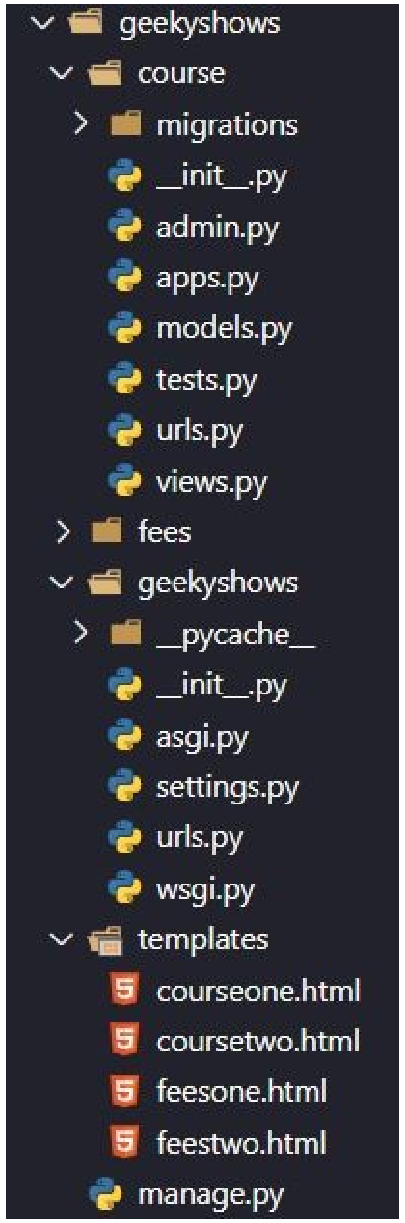
Rendering Templates Files

By Creating Template file for application we separate business logic and presentation from the application *views.py* file. Now we will write business logic in views.py file and presentation code in html file.

Still *views.py* will be responsible to process the template files for this we will use *render()* function in *views.py* file.

views.py

```
from django.shortcuts import render

def function_name(request):

    Dynamic Data, if else, any python code logic

    return render(request, template_name, context=dict_name,
content_type=MIME_type, status=None, using=None)


def learn_django(request):

    return render(request, ‘courseone.html’)
```



```
v geekyshows
  v course
    > migrations
      __init__.py
      admin.py
      apps.py
      models.py
      tests.py
      urls.py
      views.py
  > fees
  v geekyshows
    > __pycache__
      __init__.py
      asgi.py
      settings.py
      urls.py
      wsgi.py
  v templates
      courseone.html
      coursetwo.html
      feesone.html
      feestwo.html
    manage.py
```

render ( )

render ( ) Function - It combines a given template with a given context dictionary and returns an HttpResponse object with that rendered text.

Syntax:-

render(request, template_name, context=dict_name, content_type=MIME_type, status=None, using=None)

Where,

request – The request object used to generate this response.*

template_name – The full name of a template to use or sequence of template names. If a sequence is given, the first template that exists will be used. *

context – A dictionary of values to add to the template context. By default, this is an empty dictionary. If a value in the dictionary is callable, the view will call it just before rendering the template.

content_type – The MIME type to use for the resulting document. Defaults to 'text/html'.

status – The status code for the response. Defaults to 200.

using – The NAME of a template engine to use for loading the template.

render ( )

Syntax:-

render(request, template_name, context=dict_name, content_type=MIME_type, status=None, using=None)

Example:-

render(request, ‘courseone.html’, context=cname, content_type=‘application/xhtml+xml’)

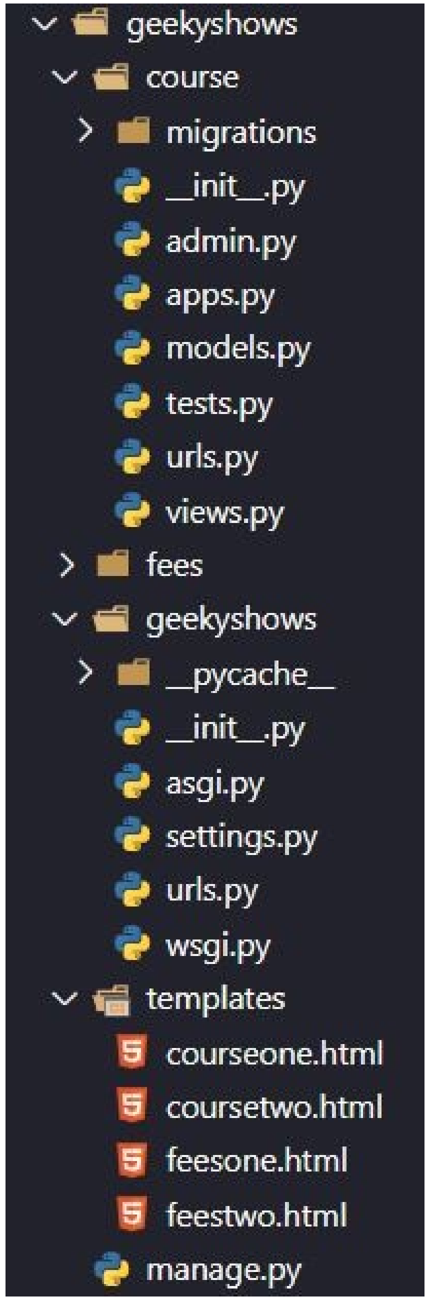
templates/

courseone.html

```
<html>
     <head>
          <style> ………. </style>
     </head>
     <body>
          <h1>Hello Django</h1>
     </body>
     <script>…………</script>
</html>
```

<u>views.py</u>

```
from django.shortcuts import render
def learn_django(request):
       return render(request, 'courseone.html')
```



```
geekyshows
  course
    migrations
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    urls.py
    views.py
  fees
  geekyshows
    __pycache__
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py
  templates
    courseone.html
    coursetwo.html
    feesone.html
    feestwo.html
  manage.py
```

Geeky Steps

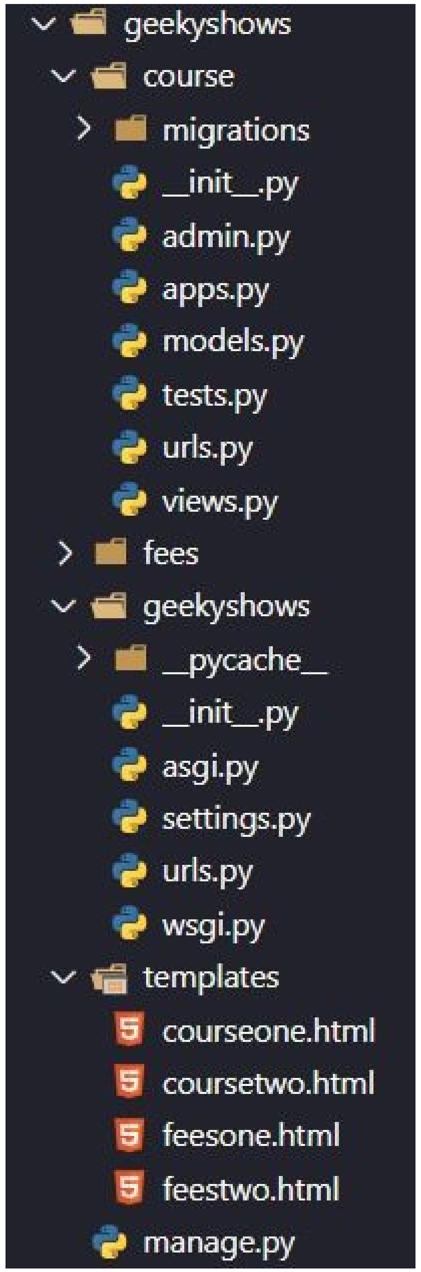
- Create Django Project: *django-admin startproject geekyshows*
- Create Django Application1: *python manage.py startapp course*
- Create Django Application2: *python manage.py startapp fees*
- Add/Install Applications to Django Project (course and fees to geekyshows) using settings.py INSTALLED_APPS
- Create **templates** folder inside Root Project Directory
- Add **templates** directory in settings.py

```
    TEMPLATES_DIR = os.path.join(BASE_DIR, 'templates')
    TEMPLATES_DIR = BASE_DIR / 'templates'
  TEMPLATES = [ {
          'DIRS': [TEMPLATES_DIR],      } ]
```

Old Django Version 3.0

Django Version 3.1

- Create template files inside templates folder
- Write View Function inside views.py file
- Define url for view function of application using urls.py file
- Write Template files code



```
geekyshows
  course
    migrations
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    urls.py
    views.py
  fees
  geekyshows
    __pycache__
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py
  templates
    courseone.html
    coursetwo.html
    feesone.html
    feestwo.html
  manage.py
```