Admin Application

It is a built-in application provided by Django.

This application provides admin interface for CRUD operations without writing sql statements.

It reads metadata from your models to provide a quick, model-centric interface where trusted users can manage content on your site.

Admin Application can be accessed using http://127.0.0.1:8000/admin

Super User is required to login into Admin Application

Create Super User

We need super user to login into admin interface of the admin application.

createsuperuser command is used to create super user.

Syntax:- python manage.py createsuperuser

How to Register Model

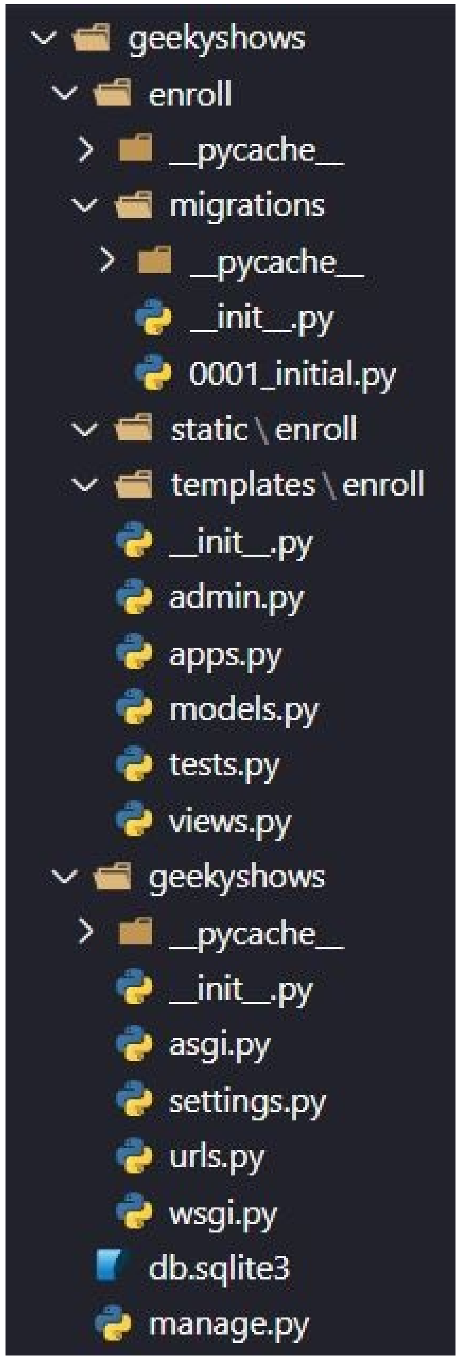
We are registering our table which we has created using model class, to default admin interface.

To Register Follow:-

- Open *admin.py* file which is inside Application Folder
- Import your own Model Class created inside Application's models.py
- admin.site.register(ModelClassName)

Example:-

- Open *admin.py*
- from enroll.models import Student
- admin.site.register(Student)



```
geekyshows
  enroll
    __pycache__
    migrations
      __pycache__
      __init__.py
      0001_initial.py
    static \ enroll
    templates \ enroll
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    views.py
  geekyshows
    __pycache__
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py
  db.sqlite3
  manage.py
```

__str__( ) Method

The __str__() method is called whenever you call str() on an object. To display an object in the Django admin site and as the value inserted into a template when it displays an object. Thus, you should always return a nice, human-readable representation of the model from the __str__() method.

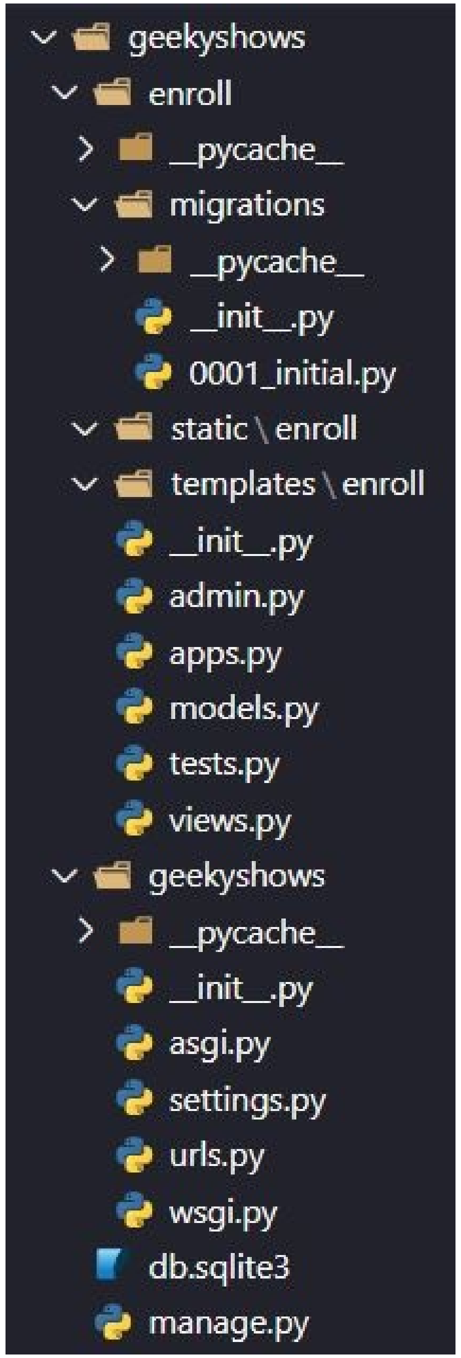
Write this Method in your own model class which is inside models.py file.

Syntax:-

```
def __str__(self):
    return self.fieldName
```

Example:-

```
def __str__(self):
    return self.stuname
```



```
geekyshows
  enroll
    __pycache__
    migrations
      __pycache__
      __init__.py
      0001_initial.py
    static\enroll
    templates\enroll
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    views.py
  geekyshows
    __pycache__
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py
  db.sqlite3
  manage.py
```

ModelAdmin

The ModelAdmin class is the representation of a model in the admin interface.

To show table's all data in admin interface we have to create an ModelAdmin class in admin.py file of Application folder.

Syntax:-

Creating Class

Class ModelAdminClassName(admin.ModelAdmin):

    list_display=('fieldname1', 'fieldname2', …….)

ModelAdmin Options

Register Above Created Class

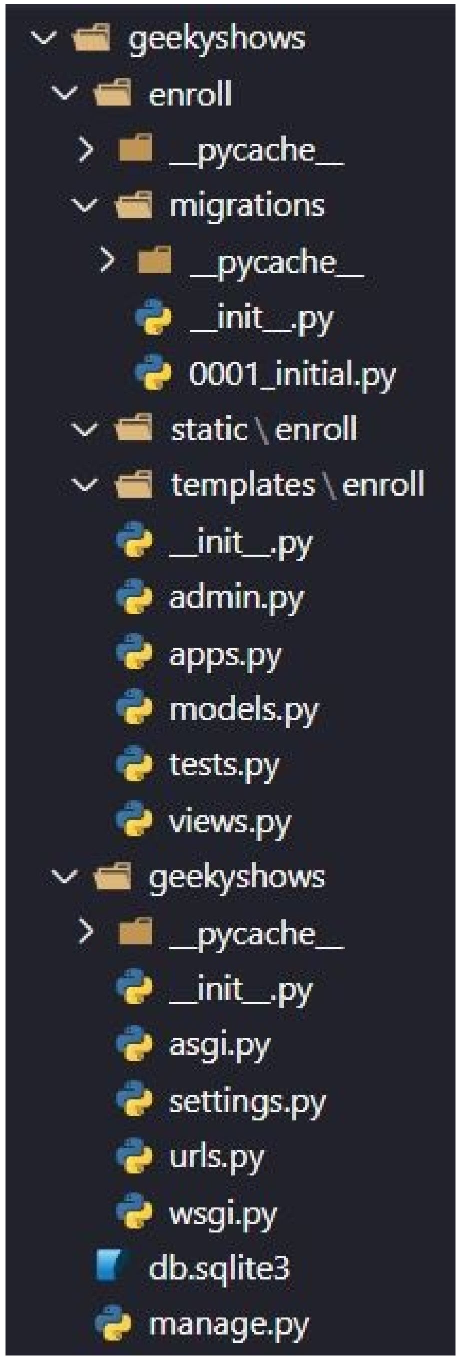
admin.site.register(ModelClassName, ModelAdminClassName)

Example: -

class StudentAdmin(admin.ModelAdmin):

    list_display=('id', 'stuid', 'stuname')

admin.site.register(Student, StudentAdmin)



```
geekyshows
  enroll
    __pycache__
    migrations
      __pycache__
      __init__.py
      0001_initial.py
    static \ enroll
    templates \ enroll
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    views.py
  geekyshows
    __pycache__
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py
  db.sqlite3
  manage.py
```

list_display

Set list_display to control which fields are displayed on the change list page of the admin. If you don't set list_display, the admin site will display a single column that displays the __str__() representation of each object

There are four types of values that can be used in list_display:

- The name of a model field.
- A callable that accepts one argument, the model instance.
- A string representing a ModelAdmin method that accepts one argument, the model instance.
- A string representing a model attribute or method (without any required arguments).

Register Model by Decorator

A decorator can be used to register ModelAdmin Classes.

Syntax:- @admin.register(ModelClassName1, ModelClassName2,… ,site=custom_admin_site)

Register Model Classes

@admin.register(ModelClassName)

Creating Class

```
Class ModelAdminClassName(admin.ModelAdmin):
        list_display=(‘fieldname1’, ‘fieldname2’, …….)
```

Example: -

```
@admin.register(Student)
class StudentAdmin(admin.ModelAdmin):
        list_display=(‘id’, ‘stuid’, ‘stuname’)
```