django-admin

django-admin --version

create project

django-admin startproject demo

python manage.py runserver

python manage.py migrate

django-admin startapp admissions

D:\Sujan\Python\_learning\Environments\schoolApp

python manage.py createsuperuser

testuser

harika123

http://localhost:8000/admin/

files :

manage.py for running the server and migration

\_\_init\_\_.py empty file to represent it as a project

settings.py add application names ,dabase configurations

urls.py map views with urls

views.py develop the view for every action ,develop in the form of functions or class

python manage.py runserver

for running the server

python manage.py runserver 5555

for specifiing the port number

for every action there will be a view , a view will be mapped with an url ,

when user interacted with the url view will recive the request

creating App

django-admin startapp admissions

views.py -- we write a view for every action

models.py -- for database actions we created models for each table we interact

view can be created by function based views and class based views

setting up templtes

1. create templates folder , sub folders for each app

2. setup DIRS in settings.py in projecte folder

import os

Templates --- > DIRS:[] --- > update to 'DIRS': [os.path.join(BASE\_DIR,'templates')]

here Base\_DIR is path for

Sending Data from View to Template

1.create a dictionary

value = {"name":"Sujan","age":36,"adderss":"Kavali"}

return render(request,'admissions/add-admissions.html',values)

2.call that in html file

<h1> Hello {{ name }} welcome to Add admissions page </h1>

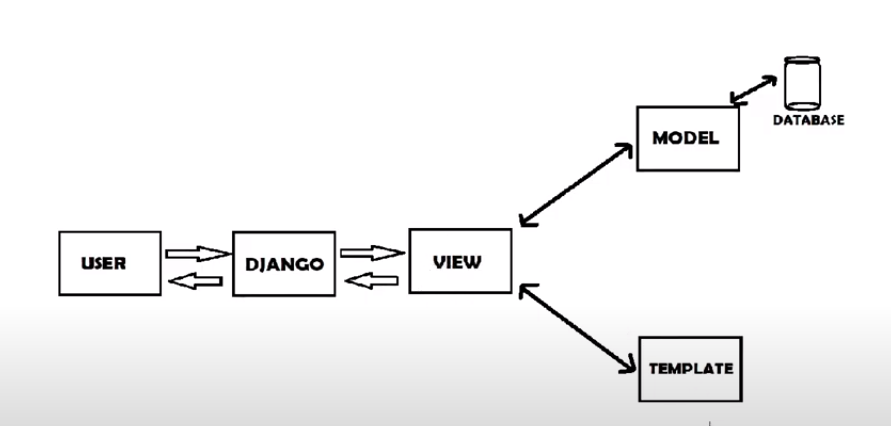
Inserting images file in project

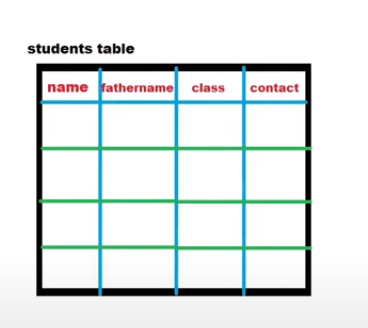
1.create static folder in project folder

2.set the STATICFILES\_DIRS setting in setting.py

3. in template load the static files --- use template tag {% load static %}

Models





Models represents database tables

Tables :

Student

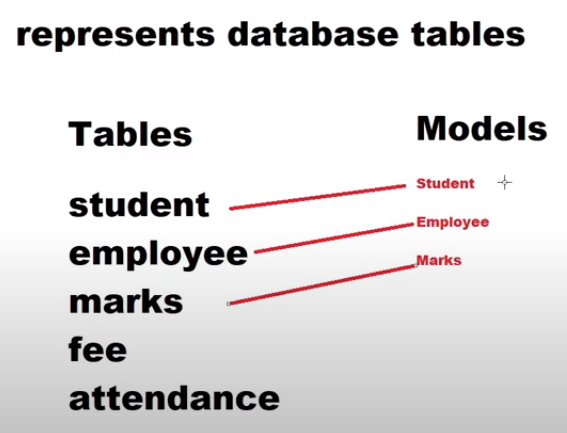
Employee

Marks

Fee

Attendance

We will create models for every table we use in app



Declare call creating models

From Django.db import models

Class Student(models.Model):

Fields

from django.db import models

# Create your models here.

class Student(models.Model):

    fields

class Employee(models.Model):

    fields

after creating models ,run *makemigrations* to create sql queries

>> makemigrations

Once queries are created run *migrate* command,to execute the queries

>> migrate

Execute the queries

Also creates admin,security,sessions tables as well

**Steps to create models**:

1.check database configuration in settings.py

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.sqlite3',

        'NAME': BASE\_DIR / 'db.sqlite3',

    }

}

Open model.py

Define models :create a class for each model that extends Model

Declare attributes with field types



class Student(models.Model):

    name= models.CharField(max\_length=50)

    fathername = models.CharField(max\_length=50)

    classname = models.CharField(max\_length=50)

    contact = models.CharField(max\_length=50)

then run two commands ,makemigrations , migrate

*python manage.py makemigrations*

then a file will be created in migration folder you can check that

Migrations for 'admissions':

admissions\migrations\0001\_initial.py

- Create model Student

>> python manage.py sqlmigrate admissions 0001

>> *python manage.py migrate*

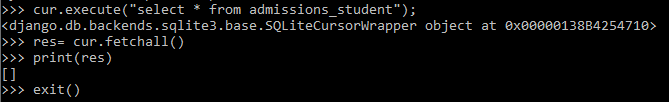
*Will apply all migration , Will create tables in database , by executing the queries*

To Verify if the tables got created or not run

>> python manage.py shell

>> from django.db import connection





Using Models in Views

We are retrieving data from view by calling a model

Data is converted to dictionary

View : Values = {“name” : “Suresh”,”age”:36,”address” : “Visakapatnam”}

Template

{{name}} ---> Suresh

{{age}} 🡪 36

from admissions.models import Student

def admissionReport(request):

    #get sll the records from the table

    #store it in dictionary students

    results = Student.objects.all() # Select \* from students

    students={'allstudends': results}

    return render(request,'admissions/admission-report.html',students)

here results is an set of objects

results --- > {student1,student2, student3, student4, student5}

we need to run a for loop to display values in each objects

{% for s in students %}

{{s.name}}

{{s.fathername}}

{% endfor %}

cur.execute("insert into admissions\_student values(1,'sujan','subbarao', '12','13232412')")

cur.execute("insert into admissions\_student values(2,'Likith','sujan', '3','131232412')")

cur.execute("insert into admissions\_student values(3,'Harika','Munish', '11','131232122')")

cur.execute("insert into admissions\_student values(4,ravi,'Munish', '11','131232122')")

**Connecting to MYSQL**

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.mysql',

        'NAME': 'school\_app',

        'USER':'root',

        'PASSWORD':'harika123',

        'HOST':'localhost',

        'PORT':'3306',

    }

}

(project1\_env) D:\Sujan\Python\_learning\Environments\schoolApp>python manage.py shell

Python 3.11.1 (tags/v3.11.1:a7a450f, Dec 6 2022, 19:58:39) [MSC v.1934 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license" for more information.

(InteractiveConsole)

>>> from django.db import connection

>>> cur = connection.cursor()

**SQL commands used**

show databases;

create user 'schoolappuser'@'localhost' identified with mysql\_native\_password by 'harika123';

grant all on school\_app.\* to 'schoolappuser'@'localhost';

select \* from school\_app.admissions\_student

once connection created

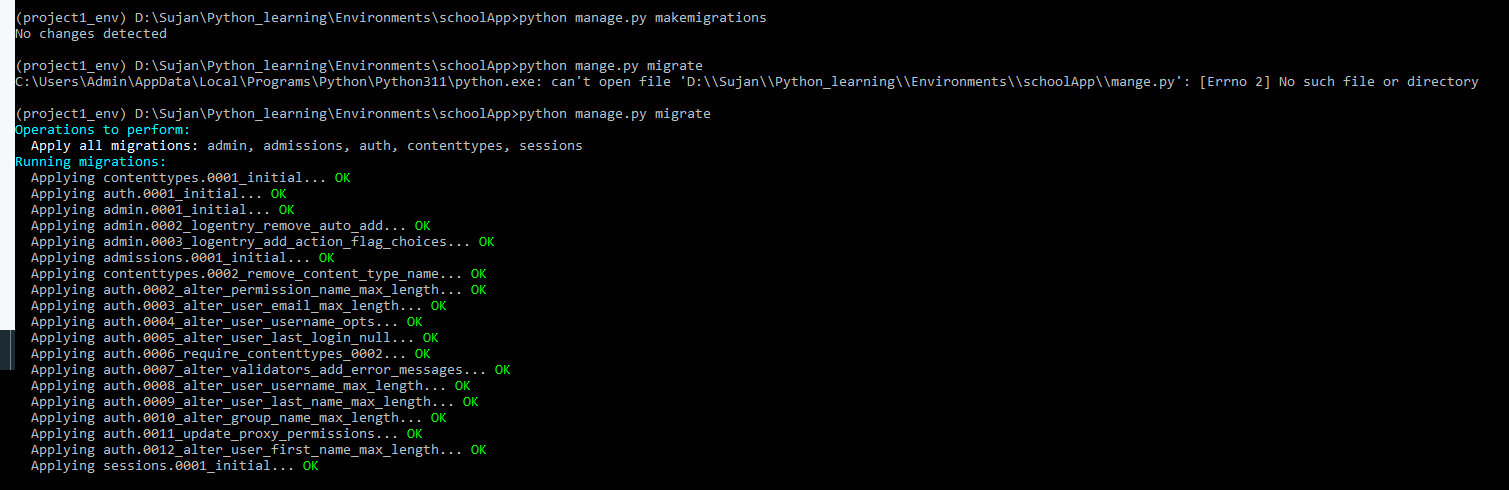
run on commend promt

>> python manage.py makemigrations

If changes made then it will show No changes detected

then run >> python mange.py migrate

this will create all required tables

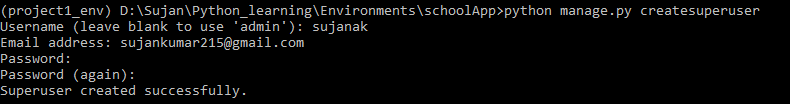


Admin UI in Django

*Python manage.py createsuperuser provide username and password*

*sujanak*

*harika123*



Register the model in abmin.py

from django.contrib import admin

from admissions.models import Student

# Register your models here.

admin.site.register(Student)

to display the table on admin page

from django.contrib import admin

from admissions.models import Student

# Register your models here.

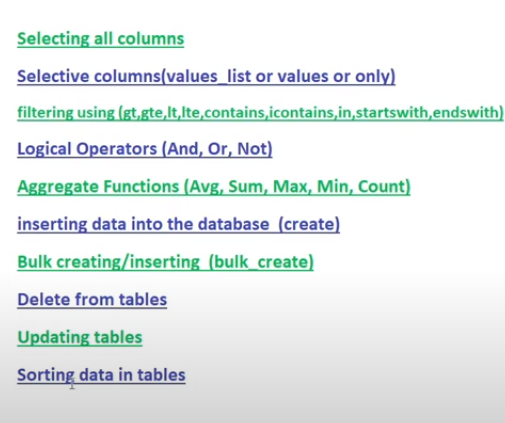
class StudentAdmin(admin.ModelAdmin):

    list\_display=['id','name','fathername','classname','contact']

admin.site.register(Student,StudentAdmin)

**Django ORM(object relation model)**

We can create different queries using ORM , such as where ,group by, joins,etc.,



|  |  |  |
| --- | --- | --- |
| SQL query | Django | Practice |
| Select \* from Admissions\_student | Student.object.all() | (project1\_env) D:\Sujan\Python\_learning\Environments\schoolApp>python manage.py shell  >>> from admissions.model import Student  >>> qs = Student.objects.all()  >>> print(qs.query)  SELECT `admissions\_student`.`id`, `admissions\_student`.`name`, `admissions\_student`.`fathername`, `admissions\_student`.`classname`, `admissions\_student`.`contact` FROM `admissions\_student`  >>> for s in qs:  ... print(s.name,"::",s.fathername)  ...  sujan :: subbarao  Likith :: sujan  Harika :: Munish  ravi :: ragu  xyz :: abc  krishna :: srina |
| Select \* from Admissions\_student where id=3; | Student.object.get(id=2)  Get used only if we get single row it is not for multiple records preferred for id | >>> qs1= Student.objects.get(id=3)  >>> print(qs1.name,qs1.fathername)  Harika Munish  >>> qs1= Student.objects.get(name='sujan') # not preferred  >>> print(qs1.name,qs1.fathername)  sujan subbarao |
| Select id,name,fathername from Admissions\_student | Students,objects.all().values\_list(‘id’,’name’,’contact’)  Students,objects.all().values(‘id’,’name’,’contact’)  Students,objects.all().only(’name’,’contact’) # returns id also | >>> qs2 = Student.objects.all().values\_list('name','contact') # rerurns in tuple format  <QuerySet [('sujan', '13232412'), ('Likith', '131232412'), ('Harika', '131232122'), ('ravi', '131232122'), ('xyz', '12345678'), ('krishna', '19283192')]>  >>> qs3= Student.objects.all().values('id','name','contact') # rerurns in disctionary format  <QuerySet [{'id': 1, 'name': 'sujan', 'contact': '13232412'}, {'id': 2, 'name': 'Likith', 'contact': '131232412'}, {'id': 3, 'name': 'Harika', 'contact': '131232122'}, {'id': 4, 'name': 'ravi', 'contact': '131232122'}, {'id': 5, 'name': 'xyz', 'contact': '12345678'}, {'id': 6, 'name': 'krishna', 'contact': '19283192'}]>  >>> qs4 = Student.objects.all().only('name','contact')  # returns in object format adds id field  >>> for s in qs4:  ... print(s.name,s.contact,s.id)  ...  sujan 13232412 1  Likith 131232412 2  Harika 131232122 3  ravi 131232122 4  xyz 12345678 5  krishna 19283192 6 |
| Select \* from admission\_student where id>3 | Student.objects.filter(id\_\_gt=3)  gt – greater than  gte – greater than or equal to  lt  lte  contains  icontains | >>> qs5 = Student.objects.filter(id\_\_gt=3)  >>> for e in qs5:  ... print(e.name)  ...  ravi  xyz  Krishna |
| Select \* from admissions\_student where name like ‘%sh%’ | Student.objects.filter(name\_\_contain=’sh’)  ## contains is case sensitive  Student.objects.filter(name\_\_icontain=’sh’)  ## icontains is case in sensitive | >>> qs6= Student.objects.filter(name\_\_contains='su')  >>> print(qs6)  <QuerySet [<Student: Student object (1)>]>  >>> for e in qs6:  ... print(e.name)  ...  Sujan |
| Select \* from admissions\_student where name like ‘s%’ | Student.objects.filter(name\_\_startswith =’s’)  Student.objects.filter(name\_\_endswith =’th’) | >>> qs= Student.objects.filter(name\_\_startswith='s')  >>> for e in qs:  ... print(e.name)  >>> qs= Student.objects.filter(name\_\_endswith ='th')  >>> for e in qs:  ... print(e.name)  ...  Likith |
| Select \* from admissions\_student where id in (1,3,5) | Student.objects.filter(id\_\_in =[1,3,5]) | >>> qs = Student.objects.filter(id\_\_in =[1,3,5])  >>> for e in qs:  ... print(e.name)  ...  sujan  Harika  xyz |
| Select \* from admissions\_student where id >2 and classname >5 | Student.objects.filter(id\_\_gt=2) & Student.objects.filter (classname\_gt=5) | >>> qs = Student.objects.filter(id\_\_gt=2) & Student.objects.filter (classname\_\_gt=5)  >>> for e in qs:  ... print(e.name) |
| Select \* from admissions\_student where NOT id >2 | Student.objects.exclude(id\_\_gt=2) | >>> qs =Student.objects.exclude(id\_\_gt=2)  >>> for e in qs:  ... print(e.name)  ...  sujan  Likith |
| Aggregation | From Django.db.model import Avg,Sum,count,max,min  Student.objects.all.aggregate(Avg(‘classname’)) | >>> from admissions.models import Student  >>> from django.db.models import Avg,Sum,Max,Min,Count  >>> Student.object.all().aggregate(Avg('classname'))  >>> Student.objects.all().aggregate(Avg('classname'))  {'classname\_\_avg': 7.5}  >>> s1 = Student.objects.all().aggregate(Avg('classname'))  >>> print(s1)  {'classname\_\_avg': 7.5}  >>> s1 = Student.objects.all().aggregate(Sum('id'))  >>> print(s1)  {'id\_\_sum': Decimal('21')}  >>> s1 = Student.objects.all().aggregate(Count('id'))  >>> print(s1)  {'id\_\_count': 6}  >>> Student.objects.filter(id\_\_gt=3).aggregate(Max('id'))  {'id\_\_max': 6} |
| Insert into admission\_student | **Method 1:**  Create an object and save  S= Student(name=’’,fathername=’’,classname=,contact=’’)  S.save()  **Method 2:**  Student.objects.create(name=’’,fathername=’’,classname=,contact=’’)  **Bulk insertion**  Student.objects.bulk\_create(  [  Student(name=’’,fathername=’’,classname=,contact=’’),  Student(name=’’,fathername=’’,classname=,contact=’’)  Student(name=’’,fathername=’’,classname=,contact=’’)  ]  ) | **Method 1:**  >>> s= Student(name='ramesh',fathername='rangarao',classname=10,contact='12312121')  >>> s.save()  **Method 2:**  >>> Student.objects.create(name='boby',fathername='verayya',classname=9,contact='2412312232')  <Student: Student object (8)>  **Bulk insertion**  >>> Student.objects.bulk\_create(  ... [  ... Student(name='yuva',fathername='reventh',classname=1,contact='47458115987'),  ... Student(name='iyan',fathername='shafi',classname=2,contact='1547621453'),  ... Student(name='Afan',fathername='aslan',classname=9,contact='1547621453')  ... ]  ... )  [<Student: Student object (None)>, <Student: Student object (None)>, <Student: Student object (None)>] |
| Deletion from admission\_student where id=3 | Student = Student.objects.get(id=7).delete()  **Multiple row deletion**  Student.objects.filter(id\_\_in=[10,11]).delete() | >>> Student.objects.get(id=7).delete()  (1, {'admissions.Student': 1})  **Multiple row deletion**  >>> Student.objects.filter(id\_\_in=[10,11]).delete()  (2, {'admissions.Student': 2}) |
| Update | S= Sudent.object.get(id=6)  S.classname=8  s.contact= | >>> S= Student.objects.get(id=6)  >>> s.classname=8  >>> s.contact='123487910'  >>> s.save() |
| Sort | Student = Student.objects.all().order\_by( | >>> stu = Student.objects.all().order\_by('name')  >>> for s in stu:  ... print(s.name,'',s.classname)  Descending order  >>> stu = Student.objects.all().order\_by('-name')  >>> for s in stu:  ... print(s.name,'',s.classname) |

**Model Forms**

Create a file forms.py in your application folder

Create a class that is inherited form forms.ModelForm

from django import forms

from admissions.models import Student

class StudentModelForm(forms.ModelForm):

    class Meta:

        model = Student

        fields = '\_\_all\_\_'

in view.py

def addAdmission(request):

    form = StudentModelForm

    studentform = {'Form':form}

    return render(request,'admissions/add-admissions.html',studentform)

in Add\_admissions.html

As table

 <form method = "POST">

                <table>

                {{Form.as\_table}}

                </table>

                <br/>

                <input type="submit" value = "Add Student">

            </form>

As paragraph

<form method = "POST">

                <table>

                {{Form.as\_p}}

                </table>

                <br/>

                <input type="submit" value = "Add Student">

            </form>

Read form input and save in database

1. Get form object with inputs
2. Save

If request.method==’POST’:

form = StudentModelForm(request.POST)

if form.is\_valid():

form.save()

def addAdmission(request):

    form = StudentModelForm

    studentform = {'Form':form}

    #values = {"name":"Sujan","age":36,"adderss":"Kavali"}

    if request.method=='POST':

        # check if the method we get the data is Post or not

        form = StudentModelForm(request.POST) # check if the data is valid or not

        if form.is\_valid():

            form.save()

        return homePage(request) # return to home page after submission

    return render(request,'admissions/add-admissions.html',studentform)

(CSRF token missing.)

Cross site request forgery : hacker attack

Add {% csrf\_token %} in Add admission.html page

    <form method = "POST">

                <table>

                {{Form.as\_ul}}

                </table>

                <br/>

                <input type="submit" value = "Add Student">

                {% csrf\_token %}

            </form>

To create forms

Just create form independent of Database object ,

1. Create a file forms.py in your application folder
2. Create a class that is inherited from forms.Form
3. Define fields
4. Use Form in view
5. Display the form in template

Vendors

Name

Address

contact

item

class VendorForm(Forms.form):

name=forms.CharField()

address= forms.CharField()

contact= forms.CharField()

item= forms.CharField()

CRUD (create Read UPDATE DELETE) operations

Delete

Localhost:8000/ad/delete

View

Def updateStudent(request,id):

s= Student.objects.get(id=id)

s.delete()

def deleteStudent(request,id):

    s= Student.objects.get(id=id) # select \* from admission\_student where id=id

    s.delete() # Delete Selected Student

    return admissionReport(request)

url

path(‘delete/**<int:id>**’,deleteStudent),

path('delete/<int:id>',deleteStudent),

admission report .html

 <table>

            <tr>

                <th> Student name</th>

                <th> Father name</th>

                <th> Class name</th>

                <th> Contact</th>

                <th>Actions</th>

            </tr>

            {% for s in allstudents %}

            <tr>

                <td> {{s.name}} </td>

                <td> {{s.fathername}} </td>

                <td> {{s.classname}} </td>

                <td> {{s.contact}} </td>

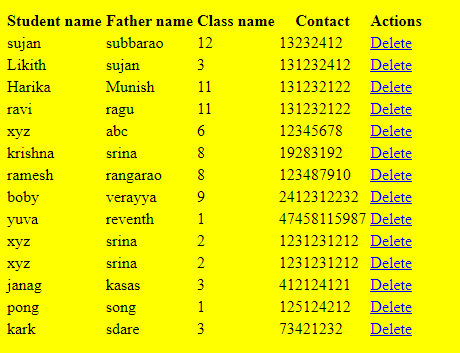
                <td><a href="/adm/delete/{{s.id}}">Delete</td>

            <br/>

            {% endfor %}

            </tr>

        </table>



Update the row

Urls.py

 path('update/<int:id>',updateStudent),

views.py

def updateStudent(request,id):

    s= Student.objects.get(id=id)

    form = StudentModelForm(instance=s)

    dict = {'form':form}

    if request.method=='POST':

        # check if the method we get the data is Post or not

        form = StudentModelForm(request.POST,instance=s) # check if the data is valid or not  ,do changes on existing student

        if form.is\_valid():

            form.save()

        return admissionReport(request)  # return redirect('adm/admreport')

    # return to home page after submission

    return render(request,'admissions/update-admission.html',dict)

update-admission.html

{% load static %}

<html>

    <head>

        <title>

            This is update admissions page

        </title>

        <link rel="stylesheet" href="{% static "css/style.css" %}"/>

        <body >

            <h1 >

               welcome to update students page

            </h1>

            <img src="{% static "images/logo-03.png" %}" width="100" height="100" alt="Image Not Found">

            <form method = "POST">

                <table>

                {{form.as\_table }}

                </table>

                <br/>

                <input type="submit" value = "update Student">

                {% csrf\_token %}

            </form>

        </body>

</html>

Admissions report

<table>

            <tr>

                <th> Student name</th>

                <th> Father name</th>

                <th> Class name</th>

                <th> Contact</th>

                <th>Actions</th>

            </tr>

            {% for s in allstudents %}

            <tr>

                <td> {{s.name}} </td>

                <td> {{s.fathername}} </td>

                <td> {{s.classname}} </td>

                <td> {{s.contact}} </td>

                <td>

                    <a href="/adm/delete/{{s.id}}">Delete</a> &nbsp;&nbsp;&nbsp;

                    <a href="/adm/update/{{s.id}}">Update</a>

                </td>

            <br/>

            {% endfor %}

            </tr>

        </table>

Create Class based view :

Create subclass of view - Django.views.generic

Implement separate methods for each http method

In urls.py

Use classname.as\_view()

Views.py

from django.views.generic import View

class firstClassBasedView(View):

    def get(self,request):

        return HttpResponse("<h1> Hello ... this my first base view</h1>")

        pass

    def post():

        pass

urls.py

 path('firstcbv/',firstClassBasedView.as\_view()),

performing CRUD operations :

from django.views.generic import View

-View

-ListView

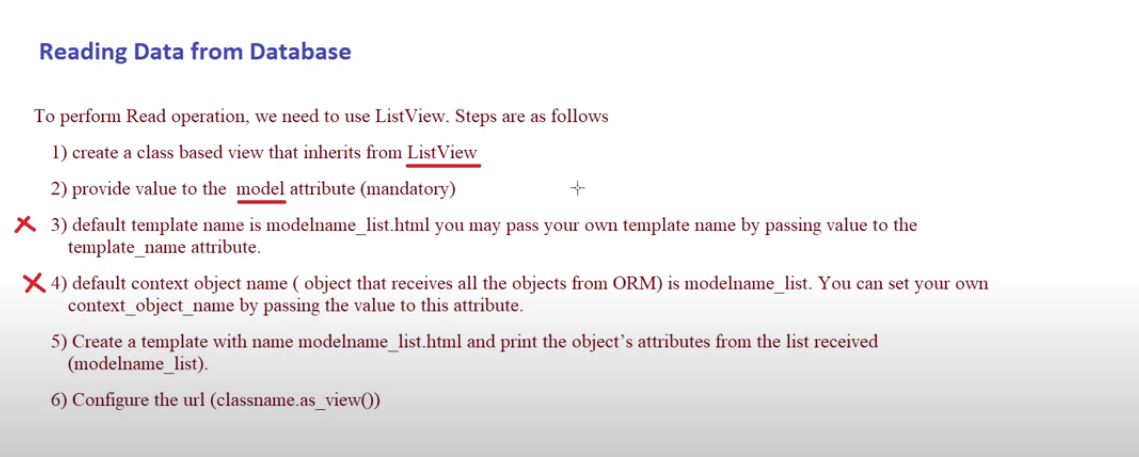
-DetailView

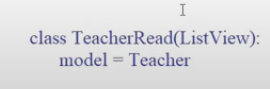
-CreateView

-UpdateView

-DeleteView

Read data from database





**teacher\_list** is the context object name will be automatically created

If we want to define our own list name

context\_object\_name = ‘result’

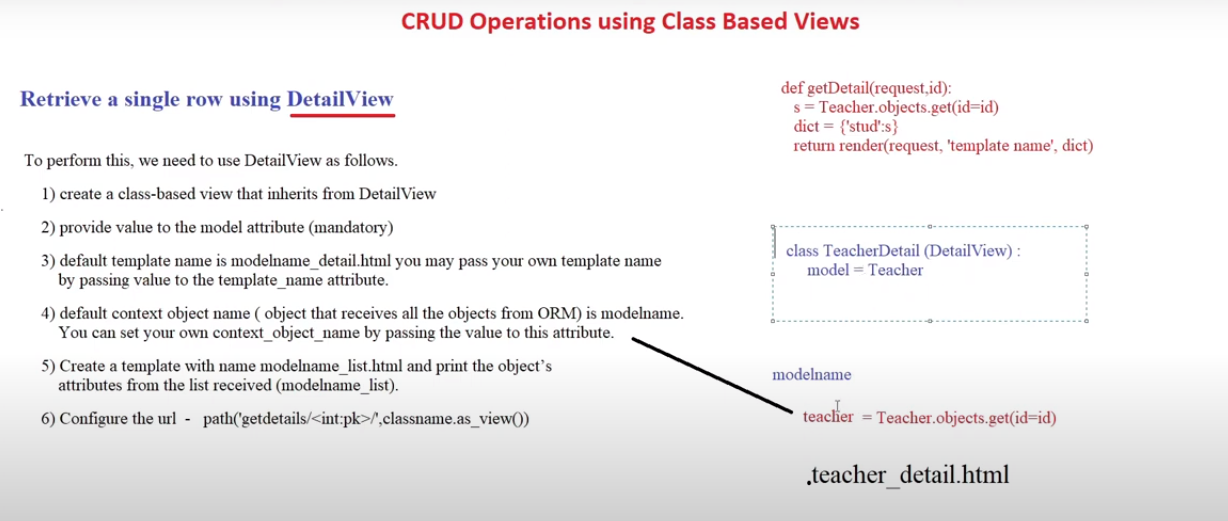
default template name **teacher\_list.html** will be created

teacher\_list will be sent to this teacher\_list.html template

if we want to define our own template name

template\_name = tlist.html

**Retrieve a single row using DetailView**



View.py

class GetTeacher(DetailView):

    model=Teacher

urls.py

path('getteacherdetail/<int:pk>/',GetTeacher.as\_view()),

Teacher\_detail.html

<html>

    <head>

        <title>

            This is Teacher detail page

        </title>

        <body  bgcolor="white">

            <h1>

                Teacher report

            </h1>

            Name : {{teacher.name}}<br/>

            Subject Name : {{teacher.subject}}<br/>

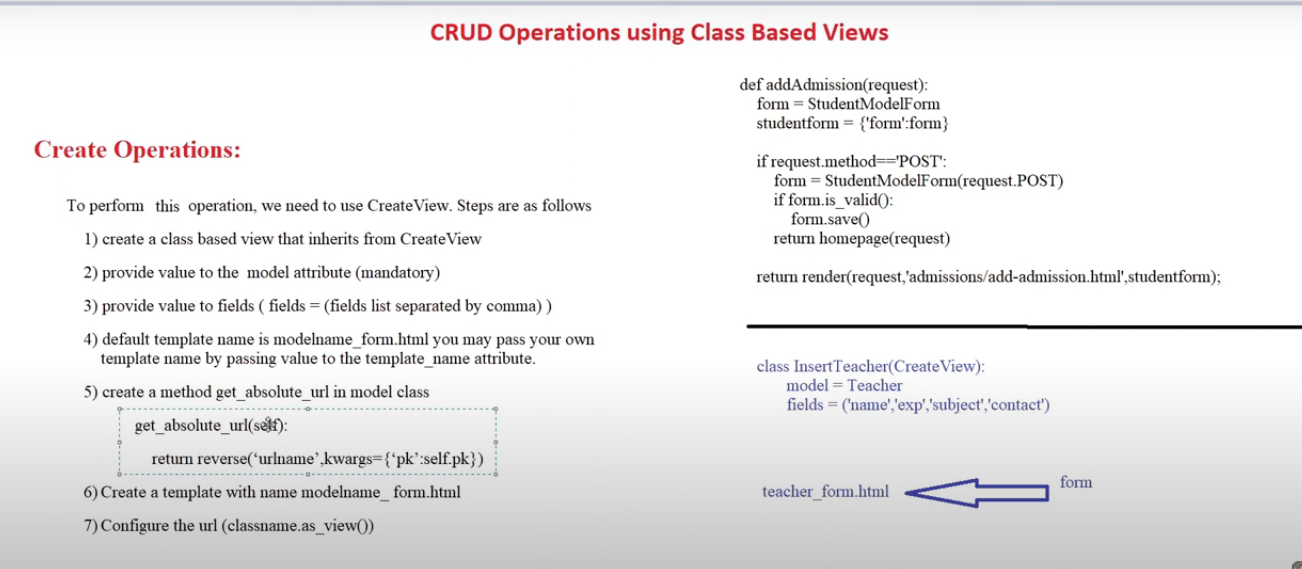
            Experiance : {{teacher.exp}}<br/>

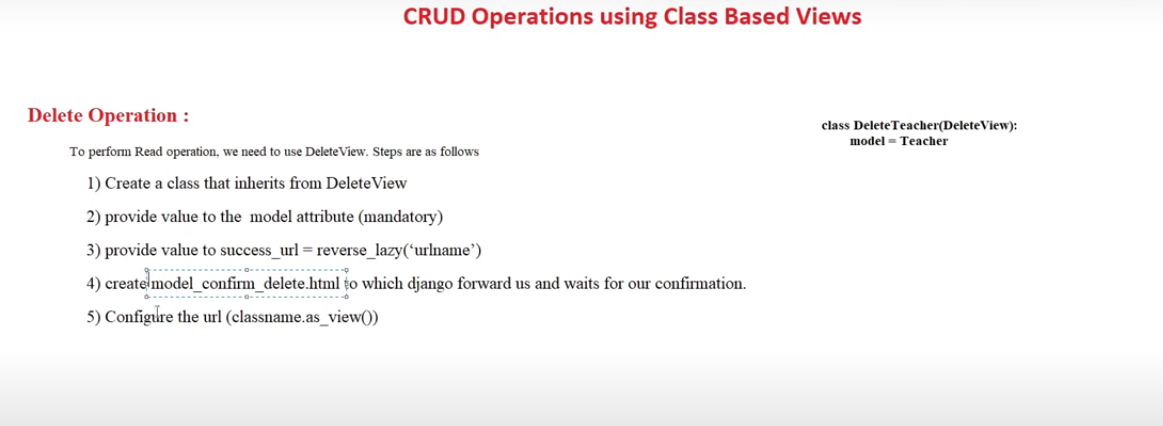
            Contact : {{teacher.contact}}<br/>

        </body>

</html>

Add Teacher





View.py

class UpdateTeacher(UpdateView):

    model=Teacher

    fields= ('name','contact')

class DeleteTeacher(DeleteView):

    model=Teacher

    success\_url =reverse\_lazy('listteachers')

url.py

    path('updateteacher/<int:pk>/',UpdateTeacher.as\_view()),

    path('deleteteacher/<int:pk>/',DeleteTeacher.as\_view()),

teacher\_list.html

  <table>

            <tr>

                <th> ID</th>

                <th> Teacher name</th>

                <th> Experiance </th>

                <th> Subject name</th>

                <th> Contact</th>

                <th>Actions</th>

            </tr>

            {% for s in teacher\_list %}

            <tr>

                <td><a href="/adm/getteacherdetail/{{s.id}}">{{s.id}} </a></td>

                <td> {{s.name}} </td>

                <td> {{s.exp}} </td>

                <td> {{s.subject}} </td>

                <td> {{s.contact}} </td>

                <td>

                    <a href="/adm/deleteteacher/{{s.id}}">Delete</a> &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

                    <a href="/adm/updateteacher/{{s.id}}">Update</a>

                </td>

            <br/>

            {% endfor %}

            </tr>

        </table>

Teacher\_confirm\_delete.html

<html>

    <head>

        <title>

            Delete form

        </title>

    </head>

        <body >

            <form method="post">

                Are you sure to Delete this Teacher ?<br/>

                <button type="submit" value = "confirm">Confirm</button>

                {% csrf\_token %}

            </form>

        <a href="/adm/teacherlist">      <button type="button" value = "Cancel">Cancel</button> </a>

        </body>

</html>

Security in Django (authentication & authorization)

1. Add the auth urls
2. Create the login form
3. Secure the views
4. Create the users
5. Test it

Step1:

In Schoolapp/urls add

path('accounts/', include('django.contrib.auth.urls')),

step2 : create registration folder in template folder then create login.html page

<!DOCTYPE html>

<html>

    <head>

        <title> </title>

    </head>

    <body>

        <form mothod = "POST">

            {{form.as\_table}}

            <button type= "submit" name="button">LOGIN </button>

            {% csrf\_token %}

        </form>

    </body>

</html>

Step3 :

Use decorator @login\_required in every methods and views we have created in view.py

from django.contrib.auth.decorators import login\_required

@login\_required

def feeCollectionReport(request):

    return render(request,'finance/feeCollectReport.html')

    #return HttpResponse("<h1>I will view collect report from this view</h1>")

For class based view

 path('teacherslist/',login\_required(TeacherRead.as\_view()),name='listteachers'),

to redirect the log out page in settings.py

LOGOUT\_REDIRECT\_URL= '/userlogout'

Step 4 :

Create users

User1

**school\_admis**

harika123

admin

sujanak

harika123

school\_user1 ---- harika123 Teachers group

group level permissions

autherizations

view.py

from django.contrib.auth.decorators import login\_required,permission\_required

@permission\_required('admissions.delete\_student')

Syntax:

@permission\_required(‘appName.operation\_modelname’)

Operations names

add –insert

delete – delete

change – update

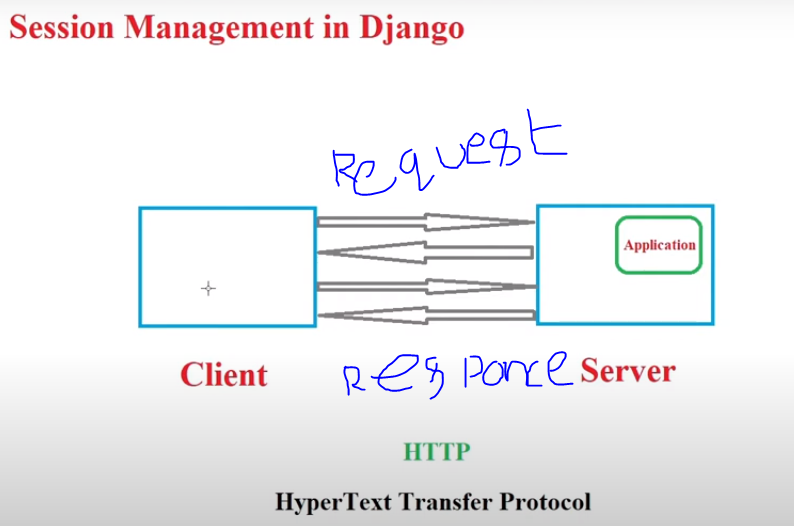
view – select

@permission\_required(‘appName.change\_student’) -🡪 for updation

@permission\_required(‘appName.add\_student’) --- > for insertion

**Sessions Management in DJANGO**

**Client server model**



HTTP is a stateless protocol

Hidden variables

Cookies

Sessions

**REST API**

(project1\_env) D:\Sujan\Python\_learning\Environments>django-admin startproject restappdj

(project1\_env) D:\Sujan\Python\_learning\Environments>django-admin startapp app

(project1\_env) D:\Sujan\Python\_learning\Environments>cd restappdj

(project1\_env) D:\Sujan\Python\_learning\Environments\restappdj>django-admin startapp app

(project1\_env) D:\Sujan\Python\_learning\Environments\restappdj>code .

(project1\_env) D:\Sujan\Python\_learning\Environments\restappdj>python manage.py runserver

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'app'

]

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.mysql',

        'NAME': 'rest\_api\_app',

        'USER':'schoolappuser',

        'PASSWORD':'harika123',

        'HOST':'localhost',

        'PORT':'3306',

    }

}

(project1\_env) D:\Sujan\Python\_learning\Environments\restappdj>python manage.py migrate

Operations to perform:

Apply all migrations: admin, auth, contenttypes, sessions

Running migrations:

Applying contenttypes.0001\_initial... OK

Applying auth.0001\_initial... OK

Applying admin.0001\_initial... OK

Applying admin.0002\_logentry\_remove\_auto\_add... OK

Applying admin.0003\_logentry\_add\_action\_flag\_choices... OK

Applying contenttypes.0002\_remove\_content\_type\_name... OK

Applying auth.0002\_alter\_permission\_name\_max\_length... OK

Applying auth.0003\_alter\_user\_email\_max\_length... OK

Applying auth.0004\_alter\_user\_username\_opts... OK

Applying auth.0005\_alter\_user\_last\_login\_null... OK

Applying auth.0006\_require\_contenttypes\_0002... OK

Applying auth.0007\_alter\_validators\_add\_error\_messages... OK

Applying auth.0008\_alter\_user\_username\_max\_length... OK

Applying auth.0009\_alter\_user\_last\_name\_max\_length... OK

Applying auth.0010\_alter\_group\_name\_max\_length... OK

Applying auth.0011\_update\_proxy\_permissions... OK

Applying auth.0012\_alter\_user\_first\_name\_max\_length... OK

Applying sessions.0001\_initial... OK

(project1\_env) D:\Sujan\Python\_learning\Environments\restappdj>python manage.py createsuperuser

Username (leave blank to use 'admin'): restuser

Email address: sujankumar215@gmail.com

Password:

Password (again):

Superuser created successfully.

Install Restapi framework

(project1\_env) D:\Sujan\Python\_learning\Environments\restappdj>pip install djangorestframework

In settings.py

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'app',

    'rest\_framework'

]

App/models.py

class Employee(models.Model):

    name = models.CharField(max\_length=30)

    email= models.CharField(max\_length=30)

    password = models.CharField(max\_length=30)

    phone= models.CharField(max\_length=20)

**migrate commands**

(project1\_env) D:\Sujan\Python\_learning\Environments\restappdj>python manage.py makemigrations

Migrations for 'app':

app\migrations\0001\_initial.py

- Create model Employee

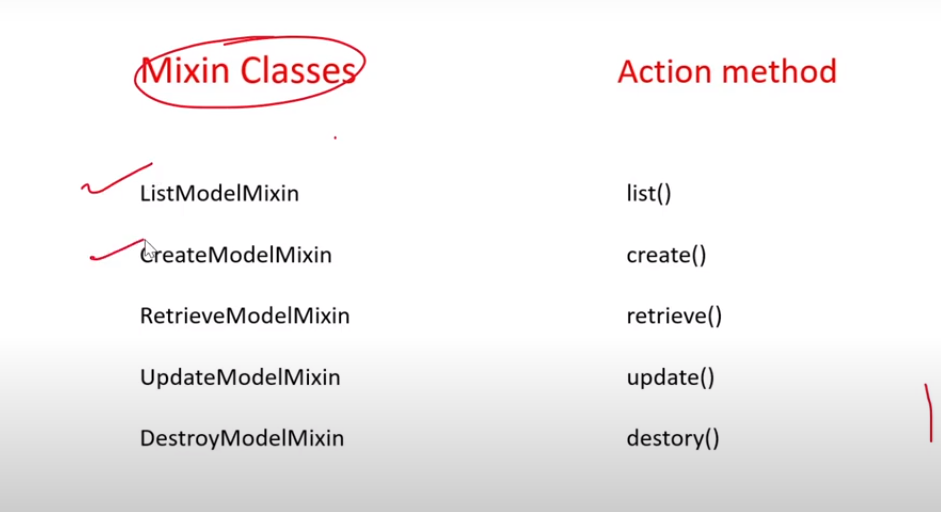
(project1\_env) D:\Sujan\Python\_learning\Environments\restappdj>python manage.py migrate

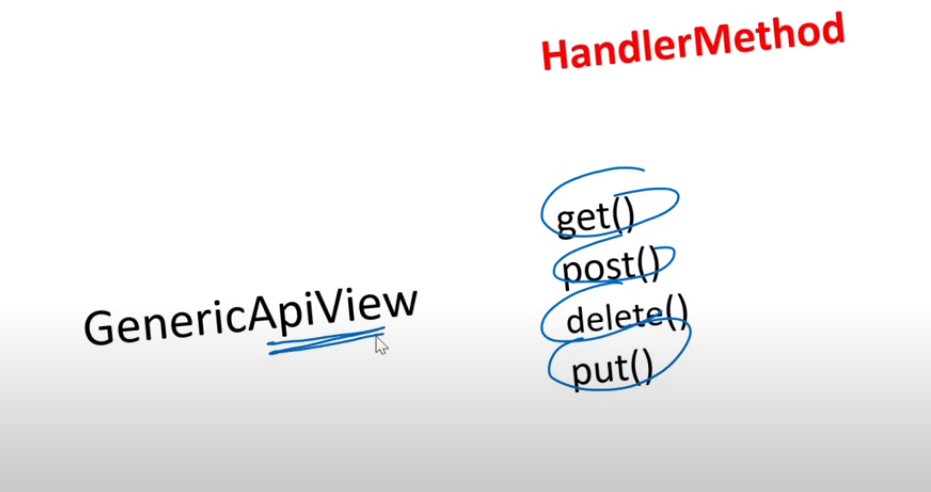
Operations to perform:

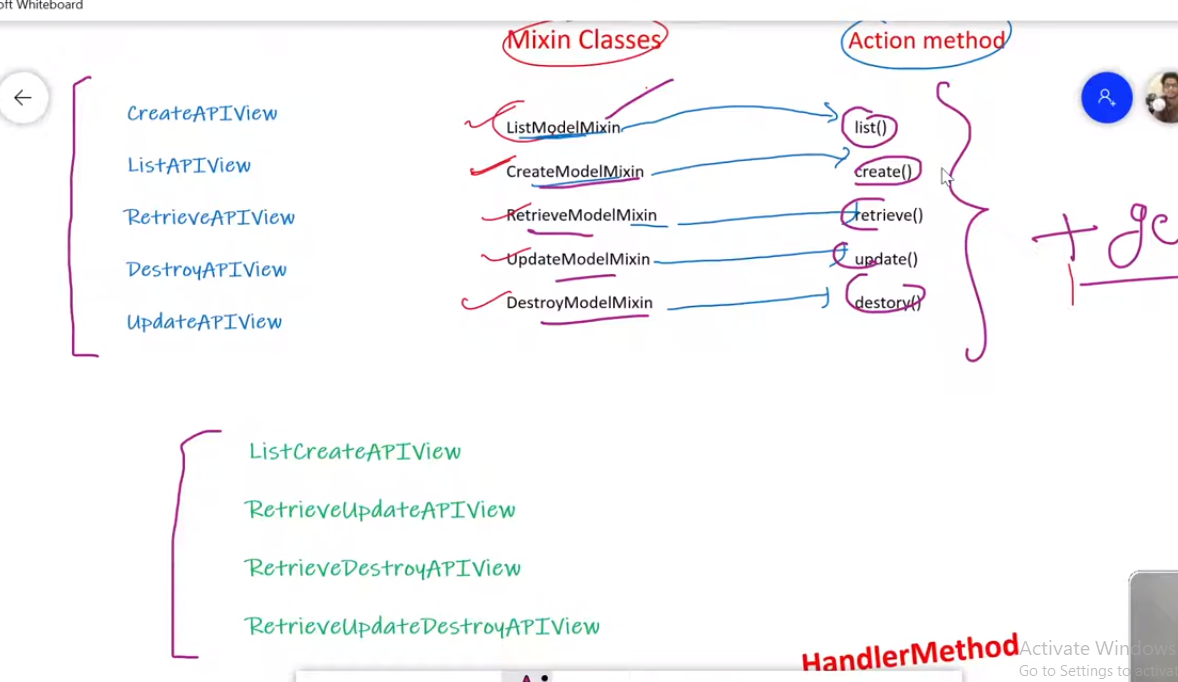
Apply all migrations: admin, app, auth, contenttypes, sessions

Running migrations:

Applying app.0001\_initial... OK







View Set