**The Basic CRUD operation of python-Django**

Inspired from this video: https://www.youtube.com/watch?v=Ko5KLX2n334&t=2101s&ab\_channel=ARKPROCODER

1. python -m venv VirtualEnv and activate
2. Create the project:
   1. django-admin startproject NameOfMyProject
3. Go to the project directory and run server:
   1. python manage.py runserver
4. Now we have to create our application:
   1. Python manage.py startapp NameOfOurApps
5. We have to create a TEMPLATES folder under the project folder and create a ‘index.html’ file
6. Now it's time to register our app in the settings file INSTALLEDapp section and DIR section give the ‘templates’.

—------ we are done creating our project, apps and registered those. Now we will focus on our landing page. We are aiming to display our index.html file stuffs on the front end. However, as we know, under the ‘View’ section of the apps folder- whatever function we have python will view that via Url.

Request from the front-end comes to project Urls, and we have to guide from project urls to Views files, and view file stuff will be displayed in the front end. For that, we need to create a couple of stuff.

1. We want to create our own urls for the specific application.
   1. Create ‘urls.py’ file in the apps folder
2. Now we have to redirect our ProjectURLS pages to apps Urls.py file
   1. from django.urls import path, include
      1. Add additionally ‘include’ .. because we want to include/make a bridge between our newly created apps/urls.py and Project/urls.py file
   2. urlpatterns = [

path('admin/', admin.site.urls),

path('', include('my\_app.urls')),

]

* + 1. For the urlpatterns: add a new ‘path’ by saying, please ‘include my apps’s url file). Second line!

Including things are done. Now we have to prepare our newly created apps/urls.py file importing structures. Simply copying the data, but we have to drop couple of unnecessary things. For example , from django.contrib import admin

The above line, we don#t have to ‘include’ anything, so leave behind the ‘include’. We need the ‘urlpatterns but in a different path condition. Now the path condition would be the path of ‘views.py/function’. Things would be looks like:

from django.urls import path

from my\_app import views

urlpatterns = [

path('', views.index, name='index')

]

1. In the above code:
   1. We need to import the views.py file
   2. From the views.py file we need the function we are about the viewing in the front page. In that case ‘index’ function. And exactly the name of the function is ‘index’

Last but not the least, we need to prepare the function in views.py file:

from django.shortcuts import render

def index(request):

return render(request, 'index.html')

1. Explanation of the views.py function:
   1. So we need a function what takes ‘request as it’s first argument.
   2. The function returns a render->
      1. Request as the first argument and the file needed to be rendered as the second argument which is ‘index.html’

Run the server and enjoy the gig show!

\_\_\_\_\_\_\_\_\_\_\_End of our first working application\_\_\_\_\_\_\_\_

**HOW TO PASS THE FONT END DATA TO THE BACK END?**

1. <head> .. <title>Student DB</title> </head>
2. < body> .. <nav class="navbar navbar-dark bg-dark text-center p-3">
3. <div class="container mt-3">
4. <div class="row">
5. <div class="col-md-4">
6. <h2 class="text-white bg-dark p-3 text-center">Insert Student Details</h2>
7. <form action="/insert" method="post">{% csrf\_token %}

After having the heading, in the body, we will deal with the NAV BAR. then

Class, container with mt-3, class=row, class = col-md-4, then a <h2> and finally the form with actin of “/insert”, method = post!!! With csrf token!!

1. take a container

2. make a row

3. divide the row in columns!

Just check the index.html file! At the first place!!

<!doctype html>

<html lang="en">

<head>

<!-- Required meta tags -->

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<!-- Bootstrap CSS -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet"

integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">

<title>Crud Mini Project</title>

</head>

<body>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"

integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p"

crossorigin="anonymous"></script>

<nav class="navbar navbar-expand-lg navbar-dark bg-dark p-3 text-center">

<div class="container-md">

<a class="navbar-brand" href="/">Crud Mini Project</a>

</div>

</nav>

<div class="container mt-5">

<div class="row">

<div class="col-md-4">

<h2 class="text-white bg-dark text-center p-3">Insert Student Details</h2>

<br>

<form action="/insert" method="post">{% csrf\_token %}

<div class="form-group">

<input type="text" class="form-control" placeholder="Enter the Student Name" name="name" required>

</div>

<div class="form-group">

<input type="email" class="form-control mt-2" placeholder="Enter the Student Email" name="email"

required>

</div>

<div class="form-group">

<input type="numer" class="form-control mt-2" placeholder="Enter the Student Age" name="age"

required>

<div class="form-group">

<select id="gender" class="form-control mt-2" name="gender" required>

<option selected>Select Gender</option>

<option value="Male">Male</option>

<option value="Female">Female</option>

<option value="Others">Others</option>

</select>

</div>

<br>

<div class="d-grid gap-2">

<button class="btn btn-success" type="submit">Submit</button>

</div>

</div>

</form>

</div>

<div class="col-md-8">

</div>

</div>

</div>

</body>

</html>

Once you are done with the html template settings **now to receive the data from font-end to backend:**

1. **take the /insert action and pass the csrf toke**n
2. And it will search for /insert actions to the urls.py file!
3. We will have to create an insert path ..

When we are ready with the forms [name, email, age, zender] and submit field, we will start thinking about actions.

1. Whenever we submit something it is always look for actions. We will now modify our form/action field. There should be a ‘/insert’ and {% csrf\_token %}

<form action="/insert" method="post">{% csrf\_token %}

1. Since we have set an action called ‘/insert’ to our html font end about to display content? We have to create a path to our **apps/urls** for the insert. Just like the about page or homepage etc. code should be following: under the urlpatters segment:

path('insert', views.insertData, name="insertData"),

1. From the code, we can see we would now need to create a function called ‘indertData’ in our apps.views section! And in the function we will set a simple if condition to collect the student data from form: name, email, age and gender by it’s given name from the **name**  code as follow:

def insertData(request):

if request.method=="POST":

name = request.POST.get('name')

email = request.POST.get('email')

age = request.POST.get('age')

gender = request.POST.get('gender')

print(name, email, age, gender)

return render(request, 'index.html')

N:B: we have set the print option to check in the server wether we are getting the front end data to back end or not?

N:B2: please don’t forget to migrate before you given the data in the font end and check in the backend!

SO THE DATA IS PERFECTLY PASSING TO THE BACK END1

**HOW TO STORE YOUR DATA TO THE DATABASE**

1. To store the newly created student data to our DB we have to create a table [python class in the apps/models.py file]. In the function create variables fields to receive the data from frontend! With exact name. Then create a python string constructor def \_\_str\_\_(self): and return self.OneVariable. The code looks like bellow:

from django.db import models

class StudentDataBase(models.Model):

name = models.CharField(max\_length=25, blank=False, null=False)

email = models.EmailField()

age = models.IntegerField()

gender = models.CharField(max\_length=25, blank=False, null=False)

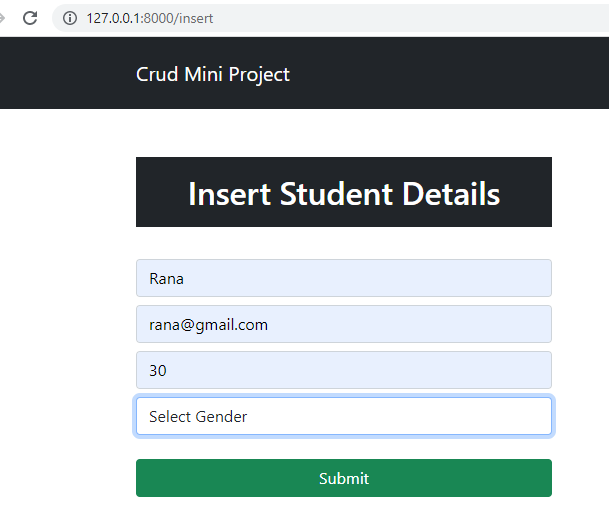
def \_\_str\_\_(self):

return self.name

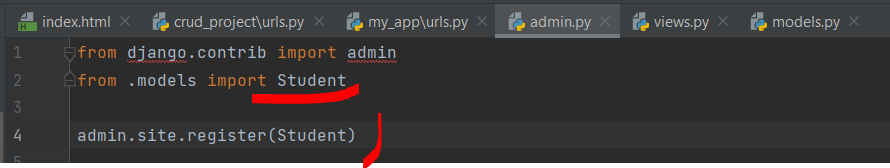
NB: When ever we change something on DB [models.py] we have to make a migration and migrate. Python manage.py makemigrations, python manage.py migrate

1. Our BD is created. But how to save the data in the DB? Whatever the data we received from our ‘POST request form’ [in views.insertData function] we have to store that in our newly created models/Student table.
   1. Import the DataTable [studnent class] in our apps/views.py

Let’s try to give some request from font-end to store in backend! We will submit couple of student details from font-end and try to store in our DB.



1. Create django superuser
2. From font-end go to the admin panel to control administration
3. We can’t find the student table there since the Table isn’t register yet!
4. We have to go to my\_apps/admin.py and register the Student table and refresh the admin page!



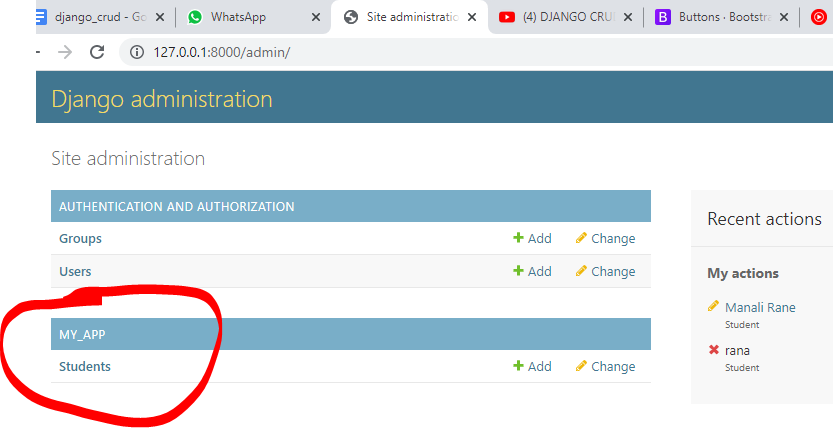
1. Now we have to save the font end data [what comes to our views insertStudentData method] to our newly created table in DB.
   1. apps/views:
      1. from .models import StudentDataBase
      2. In the insertStudentData
         1. Add extra field likewise:
         2. query = StudentDataBase(name=name, email=email, age=age, gender=gender)

query.save()

* + 1. Go to the font page and give multiple inputs
    2. Now login to the administration page using superuser and check the StudentDataBase and check the given inputs!

N:B: Make sure to migrate DB

1. Now refresh to the admin page in font end and you will have the Student DB Table there!



***HOW TO DISPLAY THE DB DATA IN FRONT END?***

1. To Display the DB data in font-end(ex. homepage) first we have to create a table in our index.html file.
2. We’ve created two div at the beginning. We will work on the second div looks like
   1. <div class="col-md-8">
3. Get the table we like to work with from getboostrap website. And display the dummy data from the font end.
4. To show the exact data first we have to work in our view/index function. ***Remember, we shouldn't do that in our insertData function!***
5. first we will fetch the data from Student object as data = Student.ojects.all()
6. Convert the data into a dictionary-> context = {"data":data} and return the context in return- index.html, context

view/index function:

from django.shortcuts import render

from .models import StudentDatabase # import the models.py Student Table to store the font-end data

def index(request):

# get the data from DB and show that in font-end

data = StudentDataBase.object.all()

context = {'data': data}

return render(request, 'index.html', context)

1. We have to get the table section in html file and for tableHead we will edit our desire fields
2. And tableBoday we have to run a for loop. Over the dictionary, we received the newly edited index function from the view file!
   1. N:B: we have to get the data from the data variable, not the returned variable!
3. And the HTML looks like:

<div class="col-md-8">

<!-- show the DB Student Data in the font end -->

<h2 class="text-white bg-dark p-3 text-center"> Student Details</h2>

<table class="table table-bordered">

<thead>

<tr>

<th scope="col">Id</th>

<th scope="col">Name</th>

<th scope="col">Email</th>

<th scope="col">Age</th>

<th scope="col">Gender</th>

<th scope="col">Edit</th>

<th scope="col">Delete</th>

</tr>

</thead>

<tbody>

{% for d in data %}

<tr>

<th scope="row">{{d.id}}</th>

<td>{{d.name}}</td>

<td>{{d.email}}</td>

<td>{{d.age}}</td>

<td>{{d.gender}}</td>

<td><button type="button" class="btn btn-primary">Edit</button></td>

<td><button type="button" class="btn btn-danger">Delete</button></td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

AND THIS IS THE WAY WE HAVE TO SHOW ANY DATABASE DATA TO THE FONTEND!

**HOW TO PERFORM EDIT OPERATION**

1. To perform edit operation to a button: we have to change the button to an **anchor tag**. And pass one **href** to the send the request to the urls.py urlpatterns. In HTML:

<td><a type="button" href= "/update/{{d.id}}" class="btn btn-primary">Edit</a></td>

<td><a type="button" href= "/delete/{{d.id}}" class="btn btn-danger">Delete</a></td>

1. We have to pass the primary key to track the changes of particular field of the DB table as well as the fontEnd request query!

2. Now we will create particular urls for the anchor tags actions in urls.py

path('update/<id>', views.updateData, name='updateData'), # url path for the href action of the edit button

path('Delete/<id>', views.deleteData, name='deleteData'), # url path for the href action of the delete button

3. Now we will create the updateData and deleteData functions in our views.py file!

* We will copy the index function[the first function]
* We will add one extra input argument: id
* We will perform the operation form newly created ‘edit.html’
  + We have to create the edit.html in our templates folder!
* Copy all of index.html page and delete the second column thing from:
  + <div class="col-md-8">
* The button name ‘Submit’ change to Update
* Form action changes from ‘/insert’ to ‘/update/{{d.id}}
* Added **value field** to name, email, age, and gender! In gender check for the different types of way to add values!
* Change the crud mini project to update
* Change the Update student details field and its bg color to info!
* The code looks like bellow and don’t forget to check the comments

<!doctype html>

<html lang="en">

<head>

<!-- Required meta tags -->

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<!-- Bootstrap CSS -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet"

integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">

<title>Crud Mini Project</title>

</head>

<body>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"

integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p"

crossorigin="anonymous"></script>

<nav class="navbar navbar-expand-lg navbar-dark bg-dark p-3 text-center">

<div class="container-md">

<a class="navbar-brand" href="/">Crud Mini Project Update</a>

</div>

</nav>

<div class="container mt-5">

<div class="row">

<div class="col-md-4">

<h2 class="text-white bg-info text-center p-3">Update Student Details</h2>

<br>

<form action="/update/{{d.id}}" method="post">{% csrf\_token %} [!--new action update been set along with id! --]

<div class="form-group">

<input type="text" class="form-control" placeholder="Enter the Student Name" name="name" value={{d.name}} required>

</div> [!--value field added--]

<div class="form-group">

<input type="email" class="form-control mt-2" placeholder="Enter the Student Email" name="email" value={{d.email}}

required> [!--value field added--]

</div>

<div class="form-group">

<input type="number" class="form-control mt-2" placeholder="Enter the Student Age" name="age" value={{d.age}}

required> [!--value field added--]

<div class="form-group">

<select id="gender" class="form-control mt-2" name="gender" required>

<option selected>{{d.gender}}</option> [!--value field added--]

<option value="Male">Male</option>

<option value="Female">Female</option>

<option value="Others">Others</option>

</select>

</div>

<br>

<div class="d-grid gap-2">

<button class="btn btn-success" type="submit">Update</button>

</div>

</div>

</form>

</div>

</div>

</div>

</body>

</html>

Now we have to make some changes to our views updateData function:

1. So we have to take the index function conditions at the first place.
   1. Edit = Student.objects.get(id=id) – change the index condition for the Class/Student
2. We need to filter the requests using the ID!
3. We need to save the edited name, email, age etc
4. We have to save that!
5. We need to redirect that to the home page after performing the edit part!
6. The entire function looks like bellow:

def updateData(request, id):

# we will take the index function conditions to perform our tasks.

if request.method == "POST":

name = request.POST['name']

email = request.POST['email']

age = request.POST['age']

gender = request.POST['gender']

edit = Student.objects.get(id=id) # we will filter the particular data with the id, so get id!

edit.name = name # whatever we done in the edit part has to to follow and save there

edit.email = email

edit.age = age

edit.gender = gender

edit.save() # save the entire edit thing

return redirect("/") # and redirect to the home!

d = Student.objects.get(id=id) # to filer the particular data the query would be the line and filtered by id!

context = {'d': d} # we must have to change the student dictionary and its variable names

return render(request, 'edit.html', context) # refer to newly created edit.html page

**HOW TO DELETE DATA**

1. Just run the following function

**def deleteData(request, id):**

**# filtered the request by ID**

**d = Student.objects.get(id=id)**

**d.delete() #delete**

**return redirect("/") #redirect the homepage**