# **Django CRUD operstion.**

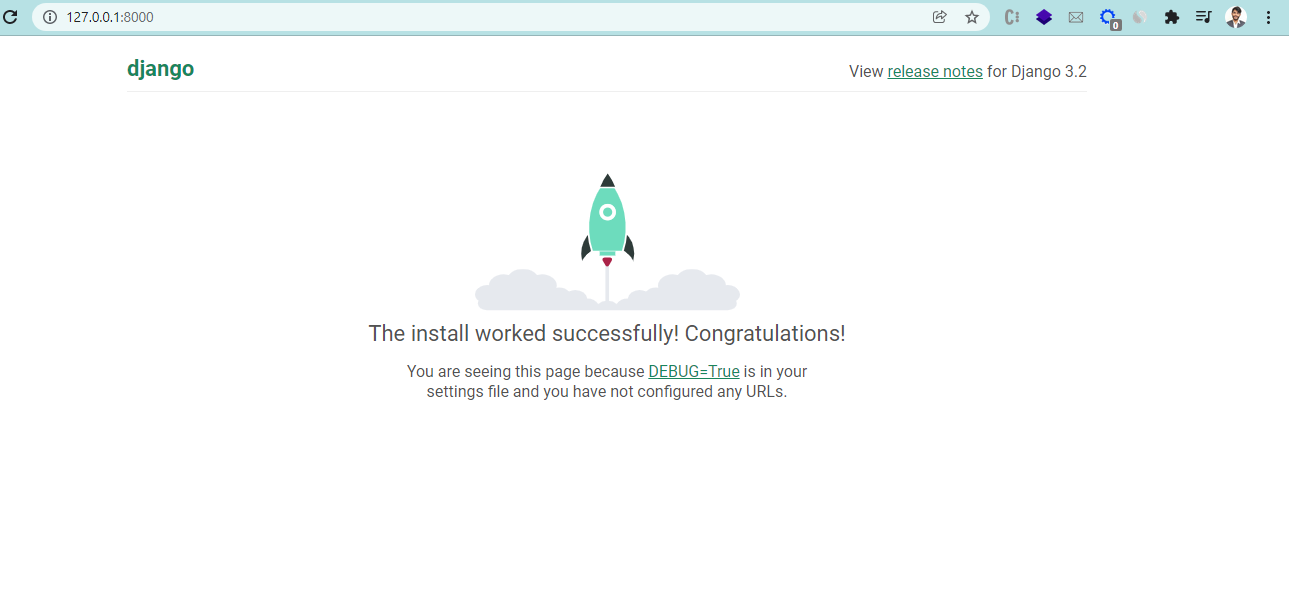
1. Create django project

Creating virtual environment-(cmd)

* python -m venv venv
* ven\Scripts\activate
* pip install django
* django-admin startproject employee\_project(name of my project)

change directory to employee\_project to run project then

* python manage.py runserver



1. Create app in this project

* Python manage.py startapp employee\_register

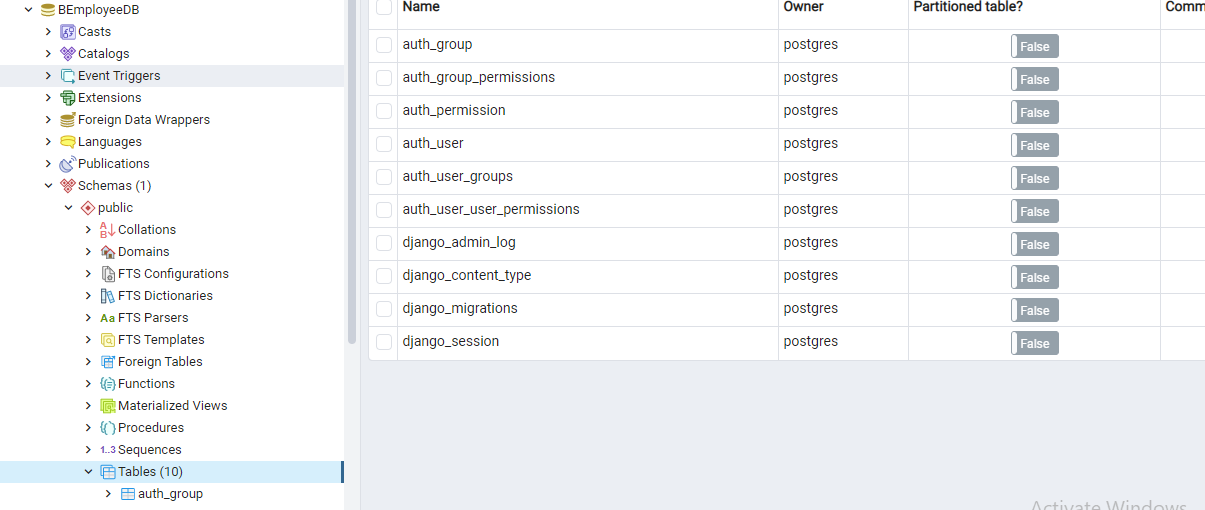
Go to employee\_project setting.py and add employee\_register in installed app

1. Create database with postgress

* Crate data base using name (postgree app)

Back to employee\_project setting.py

* import psycopg2
* DATABASES = {
* 'default': {
* 'ENGINE': 'django.db.backends.postgresql',
* 'NAME': name,
* 'USER': 'postgres',
* 'PASSWORD': 'emon',
* 'HOST': 'localhost',
* 'PORT':'5432',
* }
* }
* Python manage.py migrate



Done database connection

1. Create model on employee\_register models.py file

from turtle import position

from django.db import models

# Create your models here.

class Position(models.Model):

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

class Employee(models.Model):

    fullname = models.CharField(max\_length=100)

    emp\_code = models.CharField(max\_length=3)

    mobile = models.CharField(max\_length=15)

    position = models.ForeignKey(Position,on\_delete=models.CASCADE)

Make migrations

* python manage.py makemigrations employee\_register
* python manage.py sqlmigrate employee\_register 0001 (modification)
* python manage.py migrate (to apply migration)

1. employee\_register -> views.py

Create some function to view

def employee\_list(request):

    return

def employee\_form(request):

    return

def employee\_delete(request):

    return

1. employee\_project -> urls.py

from django.urls import path, include

urlpatterns = [

    path('admin/', admin.site.urls),

    path('employee/', include('employee\_register.urls'))

]

1. employee\_register -> create urls.py

from django.urls import path, include

from . import views

urlpatterns = [

    path('', views.employee\_form),

    path('list/', views.employee\_list),

]

#Template Section:

Create a folder name template in employee\_register folder. Add another folder to template name will be same as employee\_register.

Then create html file in employee\_register folder

* base.html
* {% block content %}
* {% endblock content %}

Use this in body section

* Employee\_list
* {% extends "employee\_register/base.html" %}
* {% block content %}
* <p>We will show employee list</p>
* {% endblock content %}
* Employee\_form

{% extends "employee\_register/base.html" %}

{% block content %}

<p>We will show employee form</p>

{% endblock content %}

Now design part of base html.

1. Form

* Create from.py in employee\_register

from django import forms

from .models import Employee

class EmployeeForm(forms.ModelForm):

    class Meta:

        model = Employee

        fields = '\_\_all\_\_'

* Add this code on views.py

from .forms import EmployeeForm

def employee\_form(request):

    form = EmployeeForm()

    return render(request, "employee\_register/employee\_form.html", {'form':form})

* Use this in form html page

{{form}}

Output like this



* In postgress add multiple position
* Import crispy form and set up this in setting and html file(pip install django-crispy-forms)
* Modify form html page and add submit button
* Label name modify if need and change position dropdown(forms.py)

**Form.py**

Need to modify this

Add logic to save data in form

* **Save information to database**

In views.py file employee\_form class-

def employee\_form(request):

    if request.method == "GET":

        form = EmployeeForm()

        return render(request, "employee\_register/employee\_form.html", {'form':form})

    else:

        form = EmployeeForm(request.POST)

        if form.is\_valid():

            form.save()

        return redirect('/employee/list')

* Show data in list page

Employee\_register>views.py

def employee\_list(request):

    context = {'employee\_list': Employee.objects.all()}

    return render(request, "employee\_register/employee\_list.html", context)

Design list.html

<table class="table table-border">

    <thead class="border-bottom font-weight-bold">

        <tr>

            <td>Full Name</td>

            <td>Mobile</td>

            <!-- <td>Employee Code</td> -->

            <td>Position</td>

            <td>Action</td>

        </tr>

    </thead>

    <tbody>

        {%  for employee in employee\_list %}

            <tr>

                <td>{{employee.fullname}}</td>

                <td>{{employee.mobile}}</td>

                <td>{{employee.position}}</td>

                <td>

                    <a href="" class="btn text-secondary px-0">

                        <i class="fa fa-edit fa-lg"></i>

                    </a>

                </td>

            </tr>

        {% endfor %}

    </tbody>

</table>

* **Edit data**
* Add action column in employee\_list.html table head
* Add icon on td

Add urls in employee\_register>urls.py

    path('<int:id>/', views.employee\_form, name='employee\_update'), #edit information

* Modify employee\_register>views.py employee\_form function get and post method to update data

def employee\_form(request, id=0):

    if request.method == "GET":

        if id == 0:

            form = EmployeeForm()

        else:

            employee = Employee.objects.get(pk=id)

            form = EmployeeForm(instance=employee)

        return render(request, "employee\_register/employee\_form.html", {'form':form})

    else:

        if id ==0:

            form = EmployeeForm(request.POST)

        else:

            employee = Employee.objects.get(pk=id)

            form = EmployeeForm(request.POST, instance=employee)

        if form.is\_valid():

            form.save()

        return redirect('/employee/list')

employee\_list.html>

<a href="{% url 'employee\_update' employee.id %}" class="btn text-secondary px-0">

                        <i class="fa fa-edit fa-lg"></i>

                    </a>

* **Delete data**

Employee\_list.html>

<form action="{% url 'employee\_delete' employee.id%}" method="post" class="d-inline">

                        {% csrf\_token %}

                        <button type="submit" class="btn">

                            <i class="fa fa-trash-o fa-lg text-denger float-right"></i>

                        </button>

                    </form>

Goto the urls employee\_register file>

    path('delete/<int:id>/', views.employee\_delete, name='employee\_delete'), #delete

Modify employee\_delete function in views.py

def employee\_delete(request,id):

    employee = Employee.objects.get(pk=id)

    employee.delete()

    return redirect('/employee/list')

**Done CRUD Project**