------------------------Deals with envs----------------------------------------------------

Pycharm creates enviroment and project without any command line if you created a project from it

list of virtual env:conda env list

create env conda create –name myenv django

to activate env: coda activate myenv

to deactivate: deactivate

to remove env : conda remove --name myenv --all

to more options :https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html

------------------------Run server---------------------------------------------------------

run django server : python manage.py runserver

-----------------------Create application--------------------------------------------------

to create application: python manage.py startapp first\_app

and go to django project settings.py and add your app name to INSTALLED\_APPS array as array

-------------------------add a view-------------------------------------------------------

to add a view : go to view file in app folder and

from django.http import HttpResponse

def index(request):

return HttpResponse("Hello World!")

so now you added a view and you need to map it into your urls

---------------------------project urls-------------------------------------------------------

go to urls.py in project folder

from django.conf.urls import url

from yourappname import views

r for regex $ for start string $ for end string thats means the only server url will redirect to index method

and add url(r'^$', views.index, name='index') to urlpatterns list

------------------------------app urls------------------------------------------------------

first we need to go to urls of project and

from django.conf.urls import include

and add url(r'^first\_app/', include('first\_app.urls')) to url patterns

and go to first app folder and add urls.py file

from django.conf.urls import url

from yourappname import views

and create a list

urlpatterns = [

url(r'^$',views.index,name='index'),

]

now if you added new url to here it will be serverurl/first\_app/whateveryouadd

-----------------------Templates----------------------

to create template

//if you have created project with pycharm don’t do until next //

create templates folder in project if not created

go to settings.py in project root and after BASE\_DIR

TEMPLATE\_DIR = os.path.join(BASE\_DIR, "templates")

and search to TEMPLATES dictionary and add TEMPLATE\_DIR to DIR list (usually pycharm adds it)

//

Finally, we are going to add our first actual template create inside templates folder

Index.html

And go to views.py of your app to attach it to your app

We’ll edit our function

def index(request):  
 return HttpResponse("<h1>Hello world!</h1>")

to

def index(request):  
 my\_dict = {'insert\_me': "Hello I am from views.py"}  
 return render(request, 'index.html', context=my\_dict)

and to print insert\_me go to index.html and write your template variable as {{insert\_me}} any where

then run your server

take care don’t forget to runserver from the project root folder

so to make your project more modular create folder inside template folder named with related app

------------------------------------------------static files----------------------------------------

How to insert static media, js or css files

Create new dir inside a project called static

And then will add the dir to settings.py file in STATIC\_URL

Like

STATIC\_URL = '/static/'  
STATICFILES\_DIRS = [  
 STATIC\_DIR,  
]

And you need to write

{% load staticfiles %}

In top of html template

Your new source will be {%static “images/pic.jpg”%}

So now you can access your image by <http://127.0.0.1:8000/static/images/image.png>

And you can do that to your css or js or any files

------------------------------------------------------------------models-------------------------------------------------------

Go to

model.py in application

create your table name class and inherit it from models.Model

class User(models.Model):  
 uid = models.AutoField(primary\_key=True)  
 uname = models.CharField(max\_length=264, blank=False)  
 upassword = models.CharField(max\_length=500, blank=False)  
 ucountry = models.CharField(max\_length=256, blank=False)  
 ucity = models.CharField(max\_length=256, blank=False)  
 ustreetAddress = models.CharField(max\_length=256, blank=False)  
 ufname = models.CharField(max\_length=256, blank=False)  
 ulname = models.CharField(max\_length=256, blank=False)  
 umembership = models.CharField(max\_length=25, blank=False)

after finishing your classes

python manage.py migrate

python manage.py makemigrations appname

python manage.py migrate

from app admin.py register your models

from users.models import User

admin.site.register(User)

to use database admin you need to

-------------------------------------create super user-------------------------------

Python manage.py createsuperuser

Go to

<http://127.0.0.1:8000/admin/>

------------------------------------------filling your database with dummy---------------------------------------------------

Install Faker library

Pip install Faker

import os  
import django  
  
os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'Creatly.settings')  
  
django.setup()  
  
from faker import Faker  
from users.models import User  
  
fakegen = Faker()  
  
  
def add\_user(N=5):  
 for entry in range(N):  
 fake\_uname = fakegen.name()  
 fake\_upassword = fakegen.name()  
 fake\_ucountry = fakegen.name()  
 fake\_ucity = fakegen.name()  
 fake\_ustreetAddress = fakegen.name()  
 fake\_fname = fakegen.name()  
 fake\_lname = fakegen.name()  
 fake\_umembership = fakegen.name()  
  
 usr = User.objects.get\_or\_create(uname=fake\_uname, upassword=fake\_upassword, ucountry=fake\_ucountry,  
 ucity=fake\_ucity, ustreetAddress=fake\_ustreetAddress, ufname=fake\_fname,  
 ulname=fake\_lname, umembership=fake\_umembership)[0]  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 print('populating script')  
 add\_user(20)  
 print("Populating Complete!")

-----------------------------------------------------------inject database to front end----------------------------------------

In views.py

from users.models import Tablename

and inside a view function

def home(request):  
 users\_list = Tablename.objects.order\_by('uid')  
 users\_dict = {'access\_records': users\_list}  
 return render(request, 'home.html', context=users\_dict)

------------------------------------------------forms------------------------------------------

Create forms.py file in application

from django import forms  
  
  
class FormName(forms.Form):  
 name = forms.CharField()  
 email = forms.EmailField()  
 text = forms.CharField(widget=forms.Textarea)

we need to show it using a view

// . means current dir

so inside view.py file

from . import forms

//after that we’ll create a view

def registration(request):  
 form = forms.FormName()  
 return render(request, 'registration.html', {'form': form})

and we’ll add the view to url.py

url(r'^reg/', views.registration, name='reg')

form validation

inside view

if request.method == ‘POST’:

form = forms.FormName(request.POST)

if form.is\_valid():

#Do nothing

print(“form validation success. Prints in console”)

print(“Name”+form.cleaned\_data[‘name’])

print(“Name”+form.cleaned\_data[‘email’])

print(“Name”+form.cleaned\_data[‘text’])

return render(request, 'registration.html', {'form': form})

------------------------check for bots---------------------

Add thins to form class

botcatcher = forms.CharField(required=False,  
 widget=forms.HiddenInput)

def clean\_botcatcher(self):  
 botcatcher = self.cleaned\_data['botcatcher']  
 if len(botcatcher)>0:  
 raise forms.ValidationError("Bte3mel eh hena yabnel teeeeeeeeet")

or you can

from django.core import validators

botcatcher = forms.CharField(required=False,  
 widget=forms.HiddenInput,  
 validators=[validators.MaxLengthValidator(0)])

and you can add your own validation function

def check\_for\_z(value):

if value[0].lower() != ‘z’:

raise forms.ValidationError(“NAME NEEDS TO START WITH Z”)

IN FORM CLASS

name = forms.CharField(validators=[check\_for\_z])

custom validators

class FormName(forms.Form):  
 username = forms.CharField()  
 useremail = forms.EmailField()  
 verimail = forms.EmailField(label="Enter your email again:")  
 text = forms.CharField(widget=forms.Textarea)  
 botcatcher = forms.CharField(required=False,  
 widget=forms.HiddenInput,  
 validators=[validators.MaxLengthValidator(0)])

def clean(self):  
 all\_clean\_data = super(FormName, self).clean()  
 useremail = all\_clean\_data['useremail']  
 verifymail = all\_clean\_data['verimail']  
  
 if useremail != verifymail:  
 raise forms.ValidationError("MAKE SURE EMAILS MATCH!")  
 return

------------------------model to form---------------------------

In forms

from django import forms  
from users.models import User  
from django.core import validators  
  
  
class NewUser(forms.ModelForm):  
 class Meta:  
 model = User  
 fields = '\_\_all\_\_'

in views

def registration(request):  
  
 form = NewUserForm()  
 if request.method == 'POST':  
 form = NewUserForm(request.POST)  
 if form.is\_valid():  
 form.save(commit=True)  
 return home(request)  
 else:  
 print("error form invalid")  
  
 return render(request, 'registration.html', {'form': form})