#Using powershell as admin n not cmd as admin for smooth movement

Django-admin startproject <project name>

#in the location wherever we need to set up the project

#the defaut project that we create this way comes with a built in small webserver

# we could run it by the command

Python manage.py runserver

# to create a new app, basically every main thing in Django is an app

Python manage.py startapp <name of the app>

# and boom , a new app has been created

#in Django urls.py is basically an index of all the pages, kind of how every activity is mentioned in the #manifest file in android .

#Also in every app we can also create a urls.py file so that all the urls didn’t need to be present on the #main urls.py file, since it will be a big clutter and also a very big and heavy file

#now since how is the main site supposed to know that we have an app , to which we need to be #redirected If pages of it comes or gets fired. For that what we have to do is in the main urls.py file we #need to first in the first line where we imported just url , there we also need to import include package #something like this:-

from django.conf.urls import include , url

#this is the line that needs to be changed from original one

from django.contrib import admin

urlpatterns = [  
 url(r'^admin/', admin.site.urls),  
 url(r'^sample/', include('sample.urls')),

#this is the line that is to be added for every app so that every app gets its #requests redirected properly, the sample is the app here

]

# the urls.py file of the app would look like this:-

from django.conf.urls import url  
from . import views  
#here . means current or same dir  
urlpatterns = [  
 url(r'^$', views.index, name='index'),  
]

# here basically we a redirecting to a view if the first parameter is get, then the view shown on right is #mapped, just like MVC in angularJS

# the view that we named here index is to be created in views.py file , and that is the name of the #method to be created

# if after running the server , if you get something like migrated files something something, then that #means that your website is not linked with the db

#to solve it its simple , in the shell :-

Python manage.py migrate

#now the db is migrated and the preinstalled apps have their tables n all set

#in Django , setting table names in model file, is same like a ORM , you only need to set it once , that is #here only, and it also gets set up in the db and also the same references of python element, we could #use the db data

#then after doing stuff in models file, go to the settings file of the website and mention the app, since it #has to connect to the database as well, earlier it didn’t need to , know it does need to, and enter the #following line in the installed apps section:-

'sample.apps.SampleConfig',

#Where sample is my app there, then apps remain the same , and inside the apps.py file , we have this #class called SampleConfig therefore “.SampleConfig”

#now after that , still if we run then we will get the migration warning / error

# to deal with it what we need to do is :-

Python manage.py makemigrations <app name>

# this will create our tables that we defined in the model .py file as a class

# and then

python manage.py migrate

# for dba api for Django

Python manage.py shell

In [1]: from sample.models import Table1, Table2

In [2]: Table1.objects.all()

Out[2]: <QuerySet []>

In [3]: a = Table1(firstColumn="some wierd data", secondColumn="guessed right again wierd dat

...: a")

In [4]: Table1.objects.all()

Out[4]: <QuerySet []>

In [5]: a

Out[5]: <Table1: Table1 object>

In [6]: a.save()

In [7]: Table1.objects.all()

Out[7]: <QuerySet [<Table1: Table1 object>]>

In [8]: a.firstColumn

Out[8]: 'some wierd data'

In [9]: a.id

Out[9]: 1

In [10]: a.pk

Out[10]: 1

In [11]: b = Table1()

In [12]: b.firstColumn = "wbgioqbf"

In [13]: b.secondColumn = "ivoin"

In [14]:

In [14]:

# the code next is the code that would be there in the models class , method , just like str format #method in java, or similar method in java , what it does is overwrites(c++ terms, don’t know if #overwrite term exist in python or not) the default str method that gots called when ever someone calls #the object of that class

def \_\_str\_\_(self):  
 return self.firstColumn + self.secondColumn

Table1.objects.filter()

#to apply filters, like id=1, its like a where clause

#for admin:-

python manage.py createsuperuser

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

Email address: asd@asdfgh.com

Password:

Password (again):

This password is too short. It must contain at least 8 characters.

This password is too common.

This password is entirely numeric.

Password:

Password (again):

Superuser created successfully.

#then go to the admin.py page / file and register the table in admin so that you can access your table in #and from admin.

#For doing this write the following code in the admin.py

from django.contrib import admin  
from .models import Table1,Table2  
# Register your models here.  
admin.site.register(Table1)  
admin.site.register(Table2)

#for creating view with which we pass something , just like a get method call, we need to create url #something like this:-

# for something like a post or something, sample/123  
url(r'^(?P<table\_id>[0-9]+)/$', views.detail, name='detail'),

#here [0-9] represents int 0 to 9 are valid input, but this is just 1 digit, i.e. we could in total only have 10 #numbers , i.e. 0,1,2,3,4,…9

#therefore we add a plus besides it, so as it will have multiple occurrences of the similar type

#here the “table\_id” could be any valid identifier, user defined , not syntactically related to the table and #to the fact that we have our first table name as table1, it could also be qwe , qwerty…so anything, just #like parameters we pass in a get call

# now in the view where we will define the method something like this:-

def detail(request, table\_id):  
 return HttpResponse("<h1>id:</h1>" + table\_id)

# now to actually use html file we use templates pretty much like angularJS:-

from django.template import loader

#this file need to be included in the views file where we are including template

#now where to place template is that , in the app folder , create a directory or folder named “templates” #, although it can be anything but it is a convention so we should follow so that we don’t get confused in #future or someone else don’t gets confused in future if someone sees the code. Then inside that #template folder or dir, create a sub folder of the same name as the app, in our case it is sample, and #inside it create our html file:-

#inside the method , i.e. url redirected view:-

template = loader.get\_template('sample/index.html')  
# by default django is set to look for it in under templates dir  
all\_tables = Table1.objects.all()  
context = {  
 'all\_tables': all\_tables,  
}  
return HttpResponse(template.render(context, request))

#here the context dictionary that we are passing can be of any name variable but again it’s a convention

#key of the dictionaries would be the ones that would be used in the template, and the values are the #ones that are the values that we assign from our python or Django code

#remember never ever forget to return request object here

#Now inside the html template , to use Django , or in actual words embed python or Django in html we #need to use opening “{%” and closing as “%}”.

# also for places where we need to pass values, there directly {{ }} can be used , again like anguarJS

<body>  
 {% for table\_row in all\_tables%}  
 <h1>the values of the data are: {{ table\_row.id}}</h1>  
 {% endfor%}  
</body>

#here interesting thing to note is that , we cannot place : in the end of the loop , if, etc.

# also due to that we need to explicitly mention the endfor and endif statements, this is there mainly #probably since in html indenting could change and therefore it cant be relied on it completely

#also there is shortcut for rendering by 2 lines lesser code:-

from django.shortcuts import render

#importing this, then , inside the method or url directed views:-

# to clean up the code a little bit , using django.shortcuts  
all\_tables = Table1.objects.all()  
context = {  
 'all\_tables': all\_tables,  
}  
return render(request, 'sample/index.html', context)

#here render method is having that base-derived kinda relationship with httprequest , since we need to #return Httprequest object, but this works perfectly well due to their internal derivations

#for having dyanamic url, and not hard coded, what we can do is:-

<a href="sample/designTemp">this and</a>  
<br>  
<a href="{% url 'dtemp' %}">both means the same this is dynamic path</a>

#commenting there with code had some issues , since dango does not know that <!-- --> is a comment #block tag for html, since it gives its data and/or the error after parsing and replacing its contents, then #it is the html to decide what to display what not to.

<!-- if in the above something after it was present like Table1's firstColumn then  
we would write it as {% url 'dtemp' all\_tables.firstColumn %}  
since all\_tables is the context passed in context from view  
-->

#In the template, here ‘dtemp’ is the name attribute that we give in the urls.py page, so we only need to #change the regular expression that is there in that url method call and the rest is changed everywhere

#for running server,

PS E:\Work\django\sampleProject> python .\manage.py runserver

#for accessing the site in localhost,

Localhost:8000

#default

#giving global namespace so that the same sub url can be used elsewhere too where it tends to exist

#in the url one ,

app\_name = 'sample'

#in the template:

<a href="{% url 'sample:dtemp' %}">both means the same this is dynamic path</a>

#for 404 shortcut:-

from django.shortcuts import get\_object\_or\_404

#in views

def fourShort(request):  
 table = get\_object\_or\_404(Table1, pk=2)  
 return render(request, 'sample/page200.html', {'table': table})

#rather than:-

def four(request):  
 # that means we cannot customize the 404 page...at least not yet  
 # raise Http404(render(request, 'sample/four.html', {'not\_required': "actually"}))  
 try:  
 table = Table1.objects.filter(pk=2)  
 except:  
 raise Http404("Why are you looking here man!")  
 return render(request, 'sample/page200.html', {'table': table})

#basically for the forms , say we specified method as post then one the function where that data will go #, at that place we can get that data by:-

request.POST['<name>']

# where <name> is the name given to say the text input html field, just like usual

#similarly for get it would mostly be:-

request.GET['<name>']

#for REST api we need to install rest framework of Django via pip or something, after it

#create a new app:-

#in the settings.py > INSTALLED\_APPS :-

#add the following:-

'rest\_framework',  
'stocks.apps.StocksConfig',

#where second line is for the new app so that Django recognizes it as the new app

#create a new model and register it in admin so as to manage it directly if you want though:-

admin.site.register(stock)

#then migrate the tables , n before that do makemigrations

Python manage.py makemigrations

Python manage.py migrate

Python manage.py createsuperuser

#now we need to create a serializer to convert our model data or our db data to JSON

#for that we first create a py file serialiser.py

from rest\_framework import serializers  
from .models import stock  
  
class StockSerializer(serializers.ModelSerializer):  
  
 class Meta:  
 model = stock  
 # fields = ('ticker', 'volume')  
 # if we wanna return some particular values  
 fields = '\_\_all\_\_'

#then in views:-

<http://www.django-rest-framework.org/tutorial/1-serialization/>

A perfect explanation step by step for Django-rest framework… very nice and easy to understand