How to run python server

On CMD: python manage.py runserver

Proceed to ‘localhost:8000’ in google chrome.

step 0: call virtual envt

call djangoEnv/scripts/activate

step 1: Navigate into PythonDjango cd .. and start project

django-admin startproject ~\*Project\_name\*~

step 2: in command prompt, make apps folder

cd ~\*Project\_name\*~

>mkdir apps

>cd apps

step 3: make init file in cmd prompt

copy nul> \_\_init\_\_.py

step 4: create a new app within app folder in cmd prompt.

>python ../manage.py startapp ~\*app\_name~\*

Step 4.1: SETTINGS

apps.~\*appname" in settings installed apps. COMA

step 5:\*~Project name\*~/urls.py (MACRO level)

Then enter in Visual Studio:

from django.conf.urls import url, include

urlpatterns=[

url(r'^', include ('apps.~\*app\_name\*~.urls'))

]

step 6: make micro urls file under ~\*app\_name\*~

Use cmd prompt

>cd apps/\*~app\_name\*~

>copy nul> urls.py

Then, enter in Visual Studio:

from django.conf.urls import url, include

from . import views

urlpatterns=[

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

]

step 7: in Visual Studio: fix up and set up VIEWS

apps/~\*app\_name\*~/views.py

In Visual Studio:

from django.shortcuts import render, HttpResponse, redirect

def index(request):

response="Hi, I'm a bunny, and this is a request!"

return HttpResponse(response)

step 8: create the html

create folder "templates" under your folder \*~app\_name\*~

include index.html under that templates folder

Folder structure

\*~app\_name\*~ > templates > index.html

step 9: sessions

cmd

python manage.py makemigrations

python manage.py migrate

step 10: migrations/models

from \_\_future\_\_ import unicode\_literals

from django.db import models

# Create your models here.

class User(models.Model):

first\_name = models.CharField(max\_length=255)

last\_name = models.CharField(max\_length=255)

email = models.EmailField(unique=True)

age = models.IntegerField()

created\_at = models.DateTimeField(auto\_now\_add=True)

updated\_at = models.DateTimeField(auto\_now=True)

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

return self.email

How to create relationships:

1. one to many relationships:

author = models.ForeignKey(Author, related\_name="books")

2. many to many

books = models.ManyToManyField(Book, related\_name="publishers")

step 11: python shell

python manage.py shell

from apps.~\*app\_name\*~.models import ~\*User\*~

step 12 : creating new records

Blog.objects.create(name="Star Wars Blog", desc="Everything about Star Wars")

delete

Dojo.objects.get(id=1).delete()

Setting Relationship

Comment.objects.create(blog=Blog.objects.get(id=1), comment="test") - create a new comment where the comment's blog points to Blog.objects.get(id=1).

various methods of retrieving

Blog.objects.get(id=1) - retrieves where id is 1; get only retrieves 1 record

Blog.object.filter(name="mike") - retrieves records where name is "mike"; returns multiple records

Blog.objects.exclude(name="mike") - opposite of filter; returns multiple records

Blog.objects.order\_by("created\_at") - orders by created\_date field

Blog.objects.order\_by("-created\_at") - reverses the order

Blog.objects.raw("SELECT \* FROM {{app\_name}}\_{{class/table name}}") - performs a raw SQL query

Blog.objects.first().comments.all() - grabs all comments from the first Blog

Blog.objects.get(id=15).comments.first() - grabs the first comment from Blog id = 15

Comment.objects.get(id=15).blog.name - returns the name of the blog where Comment id = 15 belongs to

Step 13. Validation

models.py

class BlogManager(models.Manager):

class Blog(models.Model):

objects=BlogManager()