Date: 27/02/20

Joincfe.com/from-zero // github.com/codingforenterpreneurs/Try-Django (without number)

Djangoproject.com // stackoverflow.com

py -m pip install --user virtualenv

set path – to the virutualenv C:\>setx /m PATH "C:\myfolder;%PATH%"

to create the virtualenvironmnt, run virtualenv . (dot) // .\Scripts\activate \\deactivate

Django 2.0.7 // pip freeze

>Django-admin // >Django-admin startproject <projectname> // >python manage.py runserver

Fn + ctrl + p to stop the server from cmd

#create a component/app

>python manage.py startapp products

Python manage.py migrate// python manage.py createsuperuser

>python manage.py createsuperuser

#create component / app under the project > python managepy startapp <appname>

When we make changes to the apps under models, always run the below command:

>python manage.py makemigrations //>python manage.py migrate

Models.py

from django.db import models

# Create your models here.

class Product(models.Model):

"My Product class"

title = models.TextField()

description = models.TextField()

price = models.TextField()

summary = models.TextField(default='My default')

admin.py

from django.contrib import admin

from .models import Product

# Register your models here.

admin.site.register(Product)

Add component ‘products’ below under settings.py

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'products',

]

Running project inside python shell:

>python manage.py shell

From products.modes import Product

Product.objects.all()

#create new rows in the table

Product.objects.create(title = ‘Odessye’, description=’new book’, price=20, summary=’good book’)

think ‘views’ as a place where x the webpages are managed

create new app ‘pages’ > views

rom django.shortcuts import render

# Create your views here.

from django.http import HttpResponse

def hom\_view(\*args, \*\*kwargs):

return HttpResponse("<h1> Hello World </h1>")

--

Under settings>urls.py

from django.contrib import admin

from django.urls import path

from pages import views

urlpatterns = [

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

path('', views.hom\_view, name='home')

#path('sona/', views.hom\_view, name='home') # this also works when we put /sona/ in the url

]

Def home\_view():

Return render(request, “home.html”, {}) # curly braces are for the context

> create a folder under ‘src’ as templates.

> create ‘home.html’ and add some html text inside

> in settings.py under TEMPLATES, DIR give the absolute path to ‘home.html’.

- create any number of html pages inside templates and write function inside ‘views.py’ to render the pages.

Django templaing engines: inheritance tag is {% <id> %} {% end id %}

Base.html #master template for the html pages

<!doctype html>

<html>

<head> <title> Code for the beginners </title> </head>

<h1> Hi Guys </h1>

<body>

<p> puneeth kumar </p>

{% block content %}

<p> hhh </p>

replace text # this text will not print

{% endblock %}

</body>

</html>

---

Home.html

{% extends 'base.html' %} # extends the master template

<h1> Home Page </h1>

{% include navbar.html %} # include a template into this page (\*\* navbar.html to be created)

{% block content %}

<h2> Hi Men </h2>

<p> User: {{request.user}} </p> # Django template tag ‘{{}}’ used to put context variables

</p> Authenticity:

{{request.user.is\_authenticated}} </p>

<p> My world </p>

{% endblock content %}

<p> end </p>p>

Conditions can be written inside single curly braces – {% if %}

Django template tag filter {{ variable | add:100 }} # adding 100 to variable in an item of a forloop. Filter stacking also possible with additional functions.

Home.html

{% extends 'base.html' %}

<h1> Home Page </h1>

{% block content %}

{% include 'nav.html' %}

<h2> Hi Men </h2>

<h3> {{my\_list}} </h3>

<ol>

{% for sub\_item in my\_list %}

<li> {{ florloop.counter}} {{sub\_item}} </li>

{% endfor %}

</ol>

<p> User: {{request.user}} </p>

</p> Authenticity:

{{request.user.is\_authenticated}} </p>

<p> My world {{name}}, {{age}}</p>

{% endblock content %}

<p> end </p>p>

---

Nav.html

<nav>

<ul>

<li>Orange

<li>Banana

<li>Apple

<li>Pineapple

</ul>

</nav>

--

Views.y

from django.shortcuts import render

# Create your views here.

from django.http import HttpResponse

def hom\_view(req,\*args, \*\*kwargs):

print(req.user)

print(args, kwargs)

#return HttpResponse("<h1> Hello World </h1>")

person = {'name': 'Jaison', 'age':29, 'my\_list' : ['Ginger', 'salt', 'pepper', 'cardamom', 'tamarind']}

return render(req, "home.html", person)

--

Base.html

<!doctype html>

<html>

<head> <title> Code for the beginners </title> </head>

<h1> Hi Guys </h1>

<body>

<p> puneeth kumar </p>

{% block content %}

<p> hhh </p>

replace text

{% endblock %}

</body>

</html>

--

In shell:

From models import Product

Obj = Product.objects.get(id=1)

Dir(obj)

Obj.title

Context = {‘object’: obj}

Object.price

Exit() # exit out of python shell

==

App: Products, pages. In Products, created models for data entry to database. In view, created functions rendering html pages. Under templates, created inheritable html templates. In Pages app, created the actual html pages rendered from views and extended from templates.

Django Form models:

‘GET’ method is the default for ‘form. When we got url, ‘get’ gets the information. ‘POST’ saves the information.

Difference between GET and POST method in HTTP. Both GET and POST method is used to transfer data from client to server in HTTP protocol but Main difference between POST and GET method is that GET carries request parameter appended in URL string while POST carries request parameter in message body which makes it more secure way of transferring data from client to server in http protocol.

By default, form field values are required. Default is true.

#uninstall Django. Activate the virtualenvironment. Then pip uninstall Django