**WEB PYTHON-Django**

Udje se u programiranje>python>django-admin startproject imeprojekta (website)

Udje se u programiranje>python>website

Pa cmd i ukuca se >python manage.py runserver

Pa se u brauzeru ukuca http://127.0.0.1:8000/ eto ga sajt

**Django Tutorial for Beginners - 3 - Creating Our First App**

Ako hocemo novi direktorijumtj. Novu aplikaciju udje se u cmd

Pa u cmd i ukuca se >python manage.py startapp imeaplikacije

I on napravi novi direktorijum sa sadrzajem.

# 4 - Overview of a Basic App

# 5 - Views

# Ima vec admin <http://127.0.0.1:8000/admin/>

# Ovako se pravi putanja

# Napravimo novi script "C:\Users\ZoricaJelic\Documents\programiranje\python\mata\music\urls.py"

# urls.py

# from django.conf.urls import url

# from . import views

# #music/

# urlpatterns = [

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

# 

# 

# ]

# A u starom "C:\Users\ZoricaJelic\Documents\programiranje\python\mata\mata\urls.py"

# Dodamo ovo

# from django.conf.urls import include, url

# from django.contrib import admin

# urlpatterns = [

# url(r'^admin/', admin.site.urls),

# url(r'^music/', include('music.urls')),

# ]

# Sada odemo u "C:\Users\ZoricaJelic\Documents\programiranje\python\mata\music\views.py"

# Otvorimo ga editorom I ukucamo

# Views.py

# from django.http import HttpResponse

# def index(request):

# return HttpResponse("<h1>Ovo je strana o muzici")

# Pa odemo na <http://127.0.0.1:8000/music/> i dobijemo stranicu.

# 6. baze

# https://www.youtube.com/watch?v=IU3LbtbmXXI&index=6&list=PL6gx4Cwl9DGBlmzzFcLgDhKTTfNLfX1IK

# Udjemo u cmd i ukucamo

# python manage.py migrate

# i on odradi sve migracije

# 7. models

# "C:\Users\ZoricaJelic\Documents\programiranje\python\mata\music\models.py"

# Otvorimo models.py i dodamo

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

# from django.db import models

# # Create your models here.

# class Album(models.Model):

# artist = models.CharField(max\_length=250)

# album\_title = models.CharField(max\_length=500)

# genre = models.CharField(max\_length=100)

# album\_logo = models.CharField(max\_length=1000)

# 

# class Song(models.Model):

# album = models.ForeignKey(Album, on\_delete=models.CASCADE)

# file\_type = models.CharField(max\_length=10)

# song\_title = models.CharField(max\_length=250)

# 8. Aktivacija modela

# "C:\Users\ZoricaJelic\Documents\programiranje\python\mata\mata\settings.py"

# Udjemo u settings.py

# I dodamo

# INSTALLED\_APPS = [

# 'music.apps.MusicConfig',

# 'django.contrib.admin',

# 'django.contrib.auth',

# 'django.contrib.contenttypes',

# 'django.contrib.sessions',

# 'django.contrib.messages',

# 'django.contrib.staticfiles',

# ]

# Udjemo u cmd od glavnog folder i ukucamo

# python manage.py makemigrations music

# I on nam javi sta je sve uradio

# Ako mu ukucamo python manage.py sqlmigrate music 0001

# On nam da izvestaj sta je sve napisao u bazi podataka

# Pa ukucamo python manage.py migrate

# I on ispise sta je sve uradio.

# Nadalje ukoliko se zeli da se napravi neka izmena na modelu i doda neka nova kolona u tabeli tada mora da se uradi prvo u models.py napravimo promenu ( dodamo neku kolonu i tako to) pa makemigration, pa migration, kao gore navedeno, dakle tri koraka tim redom.

# 9. Database API

# Udje se u cmd i ukuca

# python manage.py shell

# i pozeleni i pokaze [1]

# i tu kucamo

# from music.models import Album, Song

# Album.objects.all()

# I izadje nesto sto zapravo znaci da nema nista u bazi

# a=Album(artist=”Taylor Swift”,album\_title=”ti ne licis”, genre=”narodna”, album\_logo=”www.matkos.in.rs”)

# ne prima znake navoda

# a.save()

# a.artist I izadje Taylor Swift

# a.id

# I izadje 1

# a.pk

# ovo pk je primarni ki

# b=Album()

# b.artist=”Ljuba” ne prima navode

# b.album\_title=”Pare”

# b.genre=”punk”

# b.album\_logo=”nekilink”

# b.save()

# a.artist i izadje ono sto je bilo pod a

# b.artist i izadje drugi

# 10. Filtriranje rezultata iz baze podataka

# Hocemo da nam izda nazive svih albuma, zato dodamo u models.py

# class Album(models.Model):

# artist = models.CharField(max\_length=250)

# album\_title = models.CharField(max\_length=500)

# genre = models.CharField(max\_length=100)

# album\_logo = models.CharField(max\_length=1000)

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

# return self.album\_title +' \_' + self.artist

# Ovde pisemo sta hocemo I kako hocemo da nam pokaze spisak albuma.

# Pa ukucamo exit pa ponovo python manage.py shell pa

# from music.models import Album, Song

# Sada udjemo u cmd i ukucamo Pa Album.objects.all()

# I da nam spisak svih albuma kao i artistu.

# Ukucamo Album.objects.filter(id=1)

# I izadje album sa id=1, i tako dalje.

# A ako ukucamo Album.objects.filter(id=3) izadje nam prazno jer toga nema.

# Sada malo drugacije filtriranje

# Album.objects.filter(artist\_\_startswith=’Ljuba’)

# Ne prima navode. I ispise nam, pocevsi od dela imena Ljuba. Na primer ako smo u bazu uneli Ljuba Alicic, ne moramo da sada u komandi unesemo celo ime i prezime vec samo deo imena sa kojim pocinje celo ime i prezime.

# A moze i samo ime bez prezimena izvodjaca

# Album.objects.filter(artist\_\_startswith=’Taylor’)

# 11. Admin interfejs

# Ukucamo exit

# Udjemo u cmd I ukucamo

python manage.py createsuperuser

I on vam pokaze da moze da vam da user name kao vase ime, ali vi mozete da odaberete admin, a sve ovo ima i u zvanicnom tutorijalu na adresi

<https://docs.djangoproject.com/en/1.10/intro/tutorial02/>

Sada popunite vas email ova ko [admin@example.com](mailto:admin@example.com)

Pa password stonitenis

I tako dva puta I to je to. python manage.py runserver

Sada se startuje server pa se ode u brauzer i na <http://127.0.0.1:8000/admin/> i to je to. Tamo udjemo u admin panel sa pasvordom I voila.

Sada udjemo u Music I tamo editujemo script admin.py i dodamo

from django.contrib import admin

from .models import Album

admin.site.register(Album)

I sada se setamo po sadrzajima u bazi music gde mozemo da menjamo nazive albuma, dodajemo i svasta radimo.

**12. Pisanje nekog drugog vjua**

**Hocemo** da nam sada da malo vise na nasem sajtu jer u jedinom viewu koji imamo samo pise da je to strana sa muzikom a nigde nista nema da se vidi zato cemo da gradimo nas sajt tako da se vidi spisak albuma koji imaju ispred sebe broj – id i kad se klikne da se ode na taj broj, dakle da u brauzeru se vidi nesto ovako music/712 , dakle aplikacija je muzika a album je 712-esti. Zato idemo u "C:\Users\ZoricaJelic\Documents\programiranje\python\gaga\music\urls.py" i otvorimo ga editorom i dodamo

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

#/music/712

url(r'^(?P<album\_id>[0-9]+)$', ),

]

Ovo cemo kasnije da dopunimo

Sada da doteramo view. Udjemo u nas "C:\Users\ZoricaJelic\Documents\programiranje\python\gaga\music\views.py"

I editujemo ga pa mu dodamo sledece:

def index(request):

return HttpResponse("<h1>Ovo je strana o muzici")

def detail(request, album\_id):

return HttpResponse("<h2>Detalji za album ciji je id:" + str(album\_id) + "</h2>")

Sto znaci da kad se iz urls pozove music/1 on ce da pozove ovu funkciju I da izda ovaj odgovor , pa zato da doteramo nas urls

url(r'^(?P<album\_id>[0-9]+)/$',views.detail, name='detail' ),

]

## I sada odemo I u brauzer ukucamo <http://127.0.0.1:8000/music/2/> ion nam da sta treba (Detalji za album ciji je id:2).

# 13 - Connecting to the Database

Cilj nam je da na prvoj stranici bude spisak svih albuma. Zato nam trebaju podaci iz baze podataka. Zato u views dodamo

from django.http import HttpResponse

from .models import Album

I malo cemo da promenimo funkciju index prvo ovako:

def index(request):

all\_albums = Album.objects.all()

html = ''

return HttpResponse(html)

Dakle all\_albums = Album.objects.all() ovo je kao u terminal sto smo probali, a html = '' cemo tek da napisemo I to je argument funkcije koja je httpResponse.

Pa jos doteramo ovako:

def index(request):

all\_albums = Album.objects.all()

html = ''

for album in all\_albums:

url = '/music/' + str(album.id) + '/'

html += '<a href="' + url + '">' + album.album\_title +'</a><br>'

return HttpResponse(html)

I odemo na adresu <http://127.0.0.1:8000/music/> i to radi, izdaje sve albume i linkovi vode na detalje, za sada prikazuju samo id tog albuma, ali bice i jos drugih detalja.

**14 - Templates**

**H**ocemo da nasu naslovnu stranu malo doteramo i da joj dodamo neki templejt, zato dodamo nesto i zbrisemo nesto u views.py sledece:

from django.shortcuts import render

from django.http import HttpResponse

from django.template import loader

from .models import Album

# Create your views here.

def index(request):

all\_albums = Album.objects.all()

template = loader.get\_template('')

return HttpResponse('')

def detail(request, album\_id):

return HttpResponse("<h2>Detalji za album ciji je id:" + str(album\_id) + "</h2>")

Sada se udje u folder music pa se napravi ponovo u njemu folder koji se idto zove kao aplikacija music pa u njemu se sacuva novonapravljeni fajl index.html.

Dakle ovo je putanja za index.html:

"C:\Users\ZoricaJelic\Documents\programiranje\python\gaga\music\templates\music\index.html"

I u njemu samo nekoliko redova

<ul>

<li><a href="#">Album title</a></li>

</ul>

A u views –u dopunimo naredbu za templejte ovako:

template = loader.get\_template('music/index.html')

A django ce to automatski da nadje, dobar mali radnik.

U views-u jos malo promena

template = loader.get\_template('music/index.html')

context = {

'all\_albums': all\_albums,

}

return HttpResponse(template.render(context,request ))

Dakle dodat je ovaj context.

A index.html doteramo ovako:

<ul>

{% for album in all\_albums%}

<li><a href="/music/{{album.id}}">{{album.album\_title}}</a></li>

{% endfor %}

</ul>

Kao sto moze da se primeti, znaci {% python kod %} se koriste za ubacivanje Python koda, a {{}} kada se vadi nesto iz baze podataka.

Ponovo se refresuje brauzer i dobijamo skoro isto ali sada ima templejt kao sto smo i hteli.

**Sada je stanje ovako**

Views.py

from django.shortcuts import render

from django.http import HttpResponse

from django.template import loader

from .models import Album

# Create your views here.

def index(request):

all\_albums = Album.objects.all()

template = loader.get\_template('music/index.html')

context = {

'all\_albums': all\_albums,

}

return HttpResponse(template.render(context,request ))

def detail(request, album\_id):

return HttpResponse("<h2>Detalji za album ciji je id:" + str(album\_id) + "</h2>")

U folderu novi\music\templates\music\index.html

Index.html

{% if all\_albums %}

<ul>

{% for album in all\_albums %}

<li><a href="/music/{{album.id}}">{{album.album\_title}}</a></li>

{% endfor %}

</ul>

{% else %}

<h1> Nemate ni jedan album u bazi</h1>

{% endif %}

Urls.py

from django.conf.urls import url

from . import views

#music/

urlpatterns = [

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

#music/712/

url(r'^(?P<album\_id>[0-9]+)/$', views.detail,name='detail'),

Tako se namesta templejt I za sada sve radi.

**15 - Render Template Shortcut**

Nesto se izbrise a nesto se promeni tako da views.py izgleda ovako

from django.http import HttpResponse

from django.shortcuts import render

from .models import Album

def index(request):

all\_albums = Album.objects.all()

context = {

'all\_albums': all\_albums,

}

return render(request,'music/index.html',context )

def detail(request,album\_id):

return HttpResponse("<h2>detalji za albume"+str(album\_id)+"</h2>")

a index.html

{% if all\_albums %}

<h1>Ovo su svi moji albumi</h1>

<ul>

{% for album in all\_albums %}

<li><a href="/music/{{ album.id}}/">{{album.album\_title}}</a></li>

{% endfor %}

</ul>

{% else %}

<h3>Nemate ni jednog albuma</h3>

{% endif %}

**16 - Raising a 404 HTTP Error**

Ako gore u adresnoj liniji ukucamo <http://127.0.0.1:8000/music/54/> videcemo da aplikacija prikazuje stranu gde pominje detalje tog albuma koji uopste ne postoji. Zato cemo da se organizujemo da onemogucimo nesto tako glupo, vec ako postoji alum da prikaze njegove detalje, a ako ne postoji da javi porukom da ne postoji i da izda gresku 404.

Evo kako:

views.py

from django.http import Http404

from django.http import HttpResponse

from django.shortcuts import render

from .models import Album

def index(request):

all\_albums = Album.objects.all()

context = {'all\_albums': all\_albums,

}

return render(request,'music/index.html',{'all\_albums': all\_albums} )

def detail(request,album\_id):

try:

album = Album.objects.get(pk=album\_id)

except Album.DoesNotExist:

raise Http404("Album ne postoji")

return render(request,'music/detail.html',{'album': album})

index.html

{% if all\_albums %}

<h1>Ovo su svi moji albumi</h1>

<ul>

{% for album in all\_albums %}

<li><a href="/music/{{ album.id}}/">{{album.album\_title}}</a></li>

{% endfor %}

</ul>

{% else %}

<h3>Nemate ni jednog albuma</h3>

{% endif %}

I novi fajl pored indexa je detail.html "C:\Users\ZoricaJelic\Documents\programiranje\python\gaga\music\templates\music\detail.html"

detail.html

{{album}}

Samo ovo. I sve radi. Perfektno.

**17 – Dodavanje pesama u nasu bazu podataka**

U nasoj bazi se nalaze samo trenutno samo albumi, zato cemo sada da dodamo i pesme. Kao sto smo imali dodato za albume jednu funkciju za prezentaciju, tako cemo i ovde da je dodamo u models.py:

class Song(models.Model):

album = models.ForeignKey(Album, on\_delete=models.CASCADE)

file\_type = models.CharField(max\_length=10)

song\_title = models.CharField(max\_length=250)

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

return self.song\_title

Hocemo takodje da se i u nasem admin panelu pojavi opcija za pesme kao i za albume zato cemo u nasem admin.py da doteramo to:

from django.contrib import admin

from .models import Album, Song

admin.site.register(Album)

admin.site.register(Song)

I restartujemo server

python manage.py runserver I rifresujemo stranicu I sve radi.

Sada otvorimo shell da ubacimo preko terminal neke pesme komandom python manage.py shell pa kad se pojavi shell ukucamo from music.models import Album, Song. Posto je u modelu zapisano da svaka pesma je kaskadne strukture sto znaci da ni jedna pesma nije samostalna vec pripada nekom albumu , shodno tome mi prvo moramo da napravimo hijerarhiju u kojoj prvo pozivamo iz baze album koji ima pk=1 sto znaci primary key=1, naredbom:

album1 = Album.objects.get(pk=1), dakle iz baze podataka uzmi album koji ima pk=1. Pa malo da se igramo a da proverimo da li je sve ok ukucamo album1.artist i lupimo enter i izadje nam ljuba alicic, dakle artista i to kakav .

Sada ukucamo song=Song() i promenljivoj song dodeljujemo objekat Song(). Sdaa cemo da joj upisujemo vrednosti ali tako da prati kolone koje su definisane u modelu tj. Kako izgledaju u nasoj bazi podataka.

Dakle prvo:

song.album = album1 dakle nasa pesma pripada albumu1 tj. Ljubi alicicu . I nastavljamo sa komandama da popunimo sve kolone u tabeli song, dakle

song.file\_type = ‘mp3’ ,

song.song\_title = ‘Vencajte me sa njenom lepotom’ i dalje ne prima znake navoda.

I na kraju song.save() i tako snimimo sve to u bazu.

Sada ako rifresujemo nasu stranicu vidimo da ima 1 song i to ovaj koji smo upravo ubacili.

A ako hocemo da promenimo naslov onda ovako:

song.song\_title = ‘Vencajte me sa njom’

song.save() i ponovo rifres i promenjen naslov pesme. Tako je bre.

**18 - Related Objects Set**

# 