Μ.Δ.Ε. ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ

ΑΝΑΠΤΥΞΗ ΛΟΓΙΣΜΙΚΟΥ

ΤΟΥΦΕΞΗΣ ΜΙΧΑΛΗΣ ΑΕΜ 66
ΚΩΝΣΤΑΝΤΙΝΙΔΗΣ ΜΗΝΑΣ

1/2/2017

ΔΗΜΙΟΥΡΓΙΑ ΣΕΛΙΔΑΣ ΕΝΟΣ ΜΟΥΣΙΚΟΥ ΣΥΓΚΡΟΤΗΜΑΤΟΣ ΜΕ ΧΡΗΣΗ DJANGO, ΚΑΙ ΑΝΑΠΤΥΞΗ ΣΕ ΠΕΡΙΒΆΛΛΟΝ ΑΝΑΠΤΥΞΗΣ PYCHARM

Ξεκινάμε ένα νέο project στο pycharm με την ονομασία myproject. Ρυθμίζουμε την σωστή ώρα από το TIME_ZONE = 'Europe/Berlin', στο αρχείο settings.py το οποίο περιέχει τις σχετικές παραμέτρους για το site μας.

Στο ίδιο αρχείο ορίζουμε τα στατικά αρχεία, media(όπου αποθηκεύονται οι εικόνες του site) και url του site μας.

```
STATIC_URL = '/static/'
STATIC_ROOT = os.path.join(BASE_DIR, 'static')
MEDIA_URL = '/media/'
MEDIA_ROOT= os.path.join(BASE_DIR, "media")
```

.

Για να φτιαχθεί η (κενή μέχρι στιγμής) β.δ. μας τρέχουμε
manage.py migrate.

Φτιάχνουμε (για λόγους οργάνωσης) ένα νέο application μέσα στο Project μας
manage.py startapp shituation.

Έπειτα πρέπει να πούμε στο Django να χρησιμοποιήσει το νέο αυτό application
Πάμε στο mysite/settings.py

```
INSTALLED_APPS = (
'django.contrib.admin',
'django.contrib.auth',
'django.contrib.contenttypes',
'django.contrib.sessions',
'django.contrib.messages',
'django.contrib.staticfiles',
'shituation',
).
```

models.py

Στο shituation/models.py ορίζουμε τα μοντέλα τα οποία θα αποτελούν τα αντικείμενα του site και ταυτόχρονα της βάσης δεδομένων μας. Φροντίζουμε να κάνουμε τα κατάλληλα imports, και να σχεδιάσουμε σωστά τις σχέσεις με τα ξένα κλειδιά. Όλα τα μοντέλα συνδέονται με λογικό και απλό τρόπο, εκτός από το

μοντέλο Post, στο οποίο ορίζονται ώς ξένα κλειδιά όλα τα προτεύοντα κλειδιά των υπολοίπων μοντέλων, έτσι ώστε το μοντέλο Post να μπορεί να έχει πρόσβαση σε όλα τα πεδία, όλων των μοντέλων.

```
from django.db import models
from  django.utils import timezone


class Demo(models.Model):
    name = models.CharField(max_length=50)
    date = models.DateField()
    cover = models.ImageField()

    def __str__(self):
        return self.name


class Song(models.Model):
    title = models.CharField(max_length=100)
    lyrics = models.TextField()
    youtube = models.URLField()
    demo = models.ForeignKey('Demo', null=True, related_name='songs')

    def __str__(self):
        return self.title


class Live(models.Model):
    live_date = models.DateField()
    location = models.CharField(max_length=200)
    city = models.CharField(max_length=50, null=True)
    poster = models.ImageField()

    def __str__(self):
        return self.location


class Photos(models.Model):
    live = models.ForeignKey('Live', blank=True, null=True, related_name='photo')
    photo = models.ImageField()

    def __str__(self):
        return str(self.photo)


class LiveVideo(models.Model):
    live = models.ForeignKey('Live', blank=True, null=True, related_name='liveVideo')
    video = models.URLField()

    def __str__(self):
        return self.video


class Post(models.Model):
    author = models.ForeignKey('auth.User')
    title = models.CharField(max_length=200)
    text = models.TextField()
    created_date = models.DateTimeField(default=timezone.now)
    published_date = models.DateTimeField(blank=True, null=True)
    demo = models.ForeignKey('Demo', blank=True, null=True, related_name='post_demo')
    live = models.ForeignKey('Live', blank=True, null=True, related_name='post_live')
    photo = models.ForeignKey('Photos', blank=True, null=True)
    video = models.ForeignKey('LiveVideo', blank=True, null=True)
    song = models.ForeignKey('Song', blank=True, null=True)
    band = models.ForeignKey('Bands', blank=True, null=True)

    def publish(self):
        self.published_date = timezone.now()
        self.save()
```

```
    def __str__(self) -> object:
         return self.title


class Bands(models.Model):
    name = models.CharField(max_length=50)
    live = models.ForeignKey('Live', null=True, related_name='band')

    def __str__(self):
        return self.name
```

ImageField. Για να χρησιμοποιηθεί πρέπει να έχετε εγκατεστημένη την βιβλιοθήκη Pillow.

Αν η Pillow δεν είναι εγκατεστημένη τότε σε console με administrator δικαιώματα γράφουμε **pip install Pillow**

Φτιάχνουμε ένα φάκελο media στο root φάκελο του project μας (στο ίδιο επίπεδο με το blog). Εκεί θα αποθηκεύονται οι εικόνες μας.

Στο urls.py κάνουμε

```
from django.contrib.staticfiles.urls import staticfiles_urlpatterns
from django.conf.urls.static import static

urlpatterns += staticfiles_urlpatterns()
urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

manage.py makemigrations blog
Φτιάχνει ένα αρχείο που περιγράφει τις αλλαγές που πρέπει να γίνουν στη βάση μας

manage.py migrate shituation
Δημιουργεί την βάση δεδομένων, και αποθηκεύει ώς οντότητες τα μοντέλα.

Παραμετροποίηση admin.py

Κάνουμε τα κατάλληλα import.
Ορίζουμε στις κλάσεις μοντέλα τύπου admin, τα οποία προσθέτουν modules(λειτουργίες) εμφάνισης , φιλτραρίσματος και αναζήτησης στο περιβάλλον διαχείρησης.

```
from django.contrib import admin
from .models import *
from django.contrib.admin import AdminSite


class Demo_Admin(admin.ModelAdmin):
    list_display = ('name', 'date', 'cover')
    list_filter = ('name', 'date')
    search_fields = ('name', 'date')
```

```
class Song_Admin(admin.ModelAdmin):
    list_display = ('title', 'youtube', 'demo')
    list_filter = ('title', 'demo')
    search_fields = ('title', 'demo__name')


class Live_Admin(admin.ModelAdmin):
    list_display = ('location', 'city', 'live_date', 'poster')
    list_editable = ('poster', 'poster')
    list_filter = ('location', 'city', 'live_date')
    search_fields = ('location', 'city', 'live_date')


class Photos_Admin(admin.ModelAdmin):
    list_display = ('live', 'photo')
    list_editable = ('photo', 'photo')
    list_filter = ('live', 'live__city', 'live__live_date')
    search_fields = ('live__location', 'live__city', 'live__live_date')


class LiveVideo_Admin(admin.ModelAdmin):
    list_display = ('live', 'video')
    list_editable = ('video', 'video')
    list_filter = ('live', 'live__city', 'live__live_date')
    search_fields = ('live__location', 'live__city', 'live__live_date')


class Post_Admin(admin.ModelAdmin):
    list_display = ('author',
'title','published_date','song','live','photo','video')
    list_display_links = ('title','title')
    list_filter = ('author', 'published_date','live','live__city')
    search_fields = ('title', 'published_date', 'live__city',
'live__location','live__live_date', 'photo__live__live_date',
'video__live__live_date')


class My_Admin(AdminSite):                    #override Admin site, και αλλαγή
    site_header = 'Shituation Site Administration' #τίτλου και επικεφαλίδας.
    site_title = 'Shituation'




admin.site= My_Admin(name='my_admin') #ορισμός του My_Admin ώς κεντρική σελίδα.
admin.site.register(Demo, Demo_Admin) #εγγραφή των μοντέλων ξεχωριστά, μαζί με
admin.site.register(Song, Song_Admin)   #τις 'επεκτάσεις' admin.
admin.site.register(Live, Live_Admin)
admin.site.register(LiveVideo, LiveVideo_Admin)
admin.site.register(Photos, Photos_Admin)
admin.site.register(Post, Post_Admin)
admin.site.register(Bands)
```

url.py

```
from django.conf.urls import url
from . import views
urlpatterns = [
    url(r'^$', views.home_page, name='home_page'),
    url(r'^demos/$', views.demos_albums, name='demos_albums'),
```

```
    url(r'^demos/demo/(?P<pk>[0-9]+)/$', views.demo_details,
name='demo_details'),
    url(r'^demos/demo/lyrics/(?P<pk>[0-9]+)/$', views.lyrics, name='lyrics'),
    url(r'^live/$', views.live, name='live'),
    url(r'^live/details/(?P<pk>[0-9]+)/$', views.live_details,
name='live_details'),
    url(r'^live/details/videos/(?P<pk>[0-9]+)/$', views.live_videos,
name='live_videos'),
    url(r'^live/details/photos/(?P<pk>[0-9]+)/$', views.live_photos,
name='live_photos'),
    url(r'^about/$', views.about, name='about')
]
```

views.py

Ο controller του site μας. Εδώ γίνεται η αντιστοίχιση των παραμέτρων-μοντέλων της σελίδας. Ενδιαφέρον προκαλεί το home_page, που δέχεται ώς όρισμα το search από φόρμα, και να εκτελέι προχωρημένη αναζήτηση μεταξύ των αντικειμένων εξόδου.

```
from django.shortcuts import render
from django.shortcuts import get_object_or_404
from .models import *
from django.db.models import Q


def song_list(request):
    songs = Song.objects.all()
    demo = Demo.objects.get(name='Hell')
    return render(request, 'shituation/song_list.html', {'songs': songs,
'demo':demo})


def home_page(request):
    search = request.GET.get('search','')
    date = request.GET.get('date','')
    photo =request.GET.get('photo','')
    if(date==''):
        if(search==''):
            posts =
Post.objects.filter(published_date__lte=timezone.now()).order_by('-
published_date')
        else:
            posts = Post.objects.filter((Q(title__contains=search) |
Q(text__contains=search) | Q(author__username__contains=search)
                                        |Q(song__title__contains=search)|
Q(song__demo__name__contains=search)
                                        |
Q(photo__live__location__contains=search)|Q(photo__live__city__contains=search)
                                        |
Q(video__live__location__contains=search)|Q(video__live__city__contains=search)
                                        |Q(live__city__contains=search)|
Q(live__location__contains=search)
                                        &
Q(published_date__lte=timezone.now()))).order_by('-published_date')
    else:
            posts = Post.objects.filter(published_date__lte=date).order_by('-
published_date')
    return render(request, 'shituation/home_page.html', {'posts': posts})


def demos_albums(request):
    demos = Demo.objects.all()
```

```
    return  render(request, 'shituation/demos_albums.html', {'demos': demos})


def demo_details(request, pk):
    demo = get_object_or_404(Demo, pk=pk)
    songs = Song.objects.filter(demo__pk=pk)
    return render(request, 'shituation/demo_details.html',{'demo':demo,
'songs':songs})


def lyrics(request, pk):
    song = get_object_or_404(Song, pk=pk)
    return render(request, 'shituation/lyrics.html',{'song':song})


def live(request):
    lives = Live.objects.all()
    return render(request, 'shituation/live.html', {'lives':lives})


def live_details(request, pk):
    live_ = get_object_or_404(Live, pk=pk)
    return render(request, 'shituation/live_details.html', {'live_':live_})


def live_videos(request, pk):
    live_=get_object_or_404(Live, pk=pk)
    live_videos = LiveVideo.objects.filter(live__pk=pk)
    return render(request, 'shituation/live_videos.html',
{'live_videos':live_videos, 'live_':live_})


def live_photos(request, pk):
    live_=get_object_or_404(Live, pk=pk)
    live_photos = Photos.objects.filter(live__pk=pk)
    return render(request, 'shituation/live_photos.html',
{'live_photos':live_photos, 'live_':live_})


def about(request):
    return render(request, 'shituation/about.html')
```

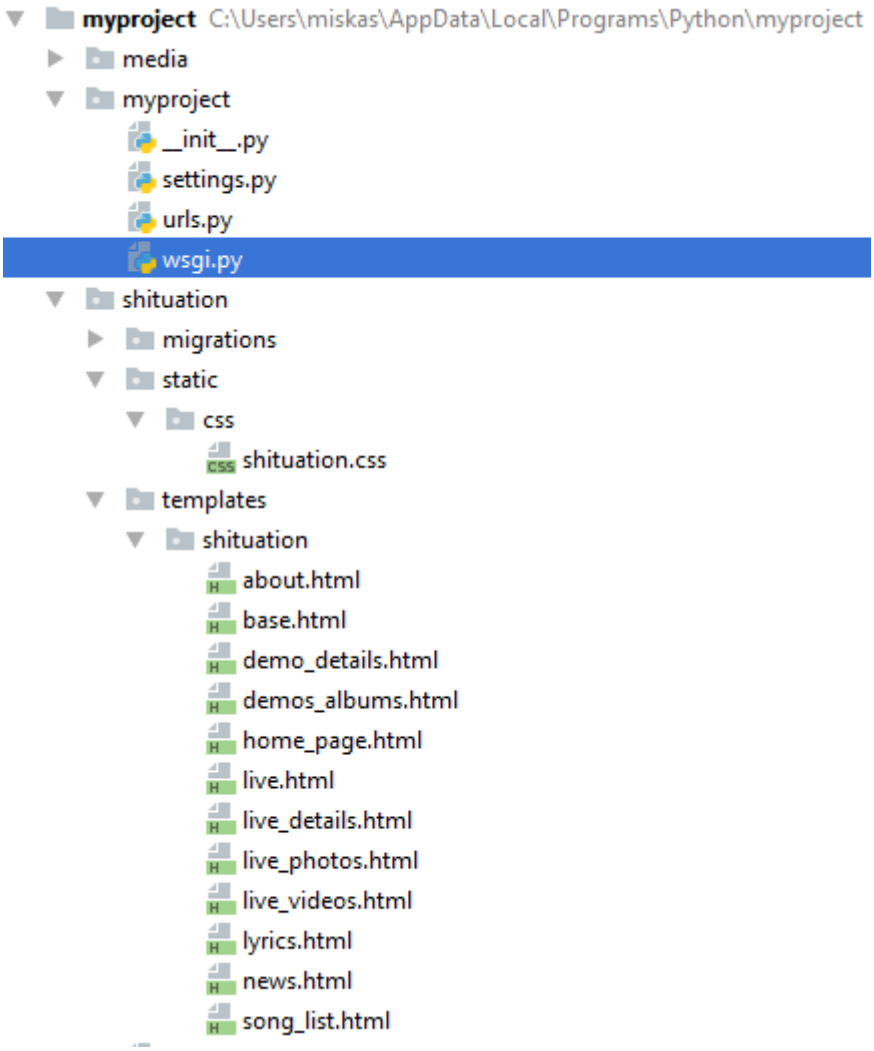
templates

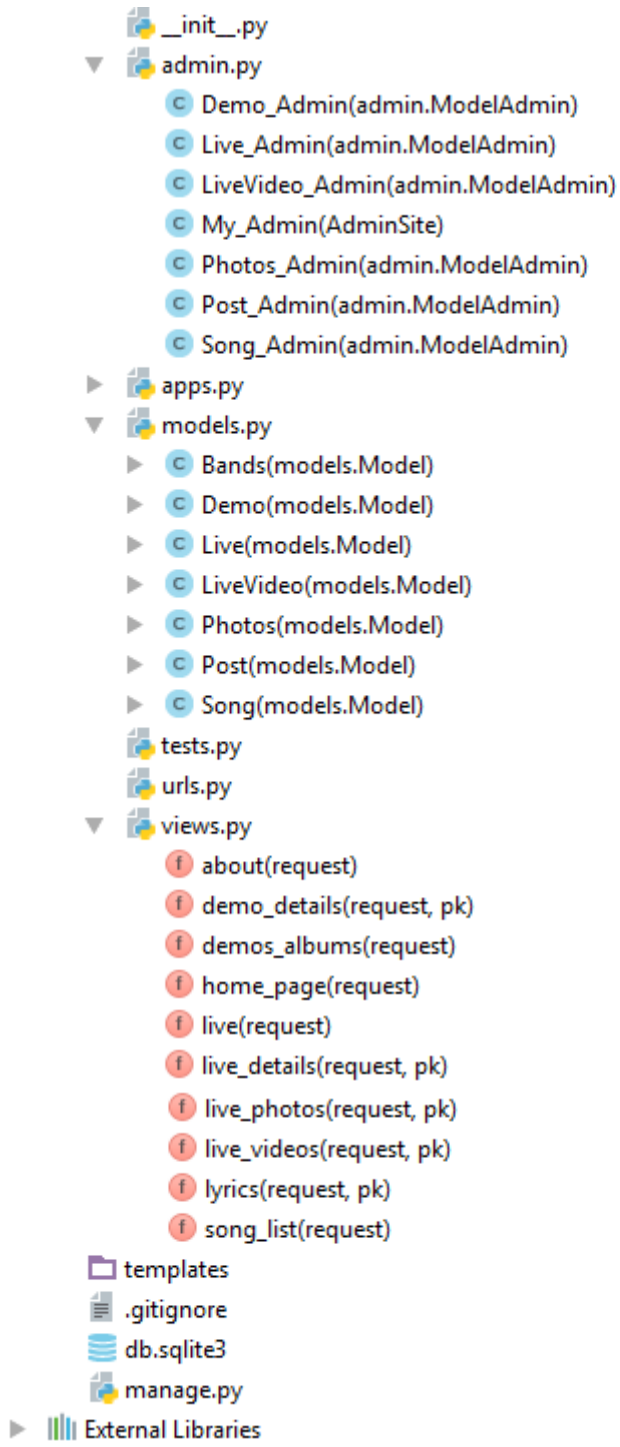
Περιέχει όλα τα html αρχεία της σελίδας μας, τα οποία περιέχουν links για πλοήγηση μεταξύ τους. Δημειουργείται πρώτα το base.html που περιέχει το μερος της σελίδας που παραμένει πάντα σταθερό. Όλα τα άλλα αρχεία το επεκτείνουν.

Το home_page.html αποτελεί την βασική σελίδα του site. Δημοσιέυει αντικείμενα Post, τα οποία μπορει να περιέχουν πεδία από τα αντικείμενα όλων των μοντέλων.

Το αρχείο shituation.css περιέχει όλες τις κλάσεις και id για την μορφολογία των html αρχείων, καθώς επίσης και javascript για την παρουσίαση μπλόκ φωτογραφιών με την μορφή carousel.

Ωργάνωση αρχείων





deployment

Κάνουμε εγκατάσταση του git.

command-line

```
$ git init
Initialized empty Git repository in ~/myproject/.git/
$ git config --global user.name "Your Name"
$ git config --global user.email you@example.com
```

.gitignore

```
*.pyc
*~
__pycache__
myvenv
db.sqlite3
/static
.DS_Store
```

command-line

```
$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .gitignore
        shituation/
        manage.py
        myproject/

nothing added to commit but untracked files present (use "git add" to track)
```

command-line

```
$ git add --all .
$ git commit -m "My Django Girls app, first commit"
 [...]
 13 files changed, 200 insertions(+)
 create mode 100644 .gitignore
 [...]
 create mode 100644 mysite/wsgi.py
```

Στο github

command-line

```
$ git remote add origin https://github.com/miskas138/my-first-site.git
$ git push -u origin master
```

command-line

```
Username for 'https://github.com': miskas138
Password for 'https://miskas138@github.com':
Counting objects: 6, done.
Writing objects: 100% (6/6), 200 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/miskas138/my-first-site.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

PythonAnywhere command-line

```
$ git clone https://github.com/miskas138/my-first-site.git
```

PythonAnywhere command-line

```
$ cd my-first-site

$ virtualenv --python=python3.5 myvenv
Running virtualenv with interpreter /usr/bin/python3.5
[...]
Installing setuptools, pip...done.

$ source myvenv/bin/activate

(myvenv) $  pip install django~=1.10.0
Collecting django
[...]
Successfully installed django-1.10.4
```

PythonAnywhere command-line

```
(mvenv) $ python manage.py migrate
Operations to perform:
[...]
  Applying sessions.0001_initial... OK
(mvenv) $ python manage.py createsuperuser
```

shituation_pythonanywhere_com_wsgi.py

```
import os
import sys

path = '/home/shituation/my-first-blog'  # use your own PythonAnywhere username
here
if path not in sys.path:
    sys.path.append(path)

os.environ['DJANGO_SETTINGS_MODULE'] = 'myproject.settings'

from django.core.wsgi import get_wsgi_application
from django.contrib.staticfiles.handlers import StaticFilesHandler
application = StaticFilesHandler(get_wsgi_application())
```

Πρέπει να προσέξουμε επίσης το πεδίο

```
ALLOWED_HOSTS = ['shituation.pythonanywhere.com']
```

το οποίο είναι κενό σε local λειτουργία.