

# DJANGO WEB ARAYÜZÜ

Merhaba arkadaşlar, uzun süre sonra tekrar Django web arayüzü ( framework ) hakkında yeni ve güncel bilgiler paylaşmak için tekrar beraberiz. Zero2Hero şeklinde ilerleyip, herkesin anlayabileceği şekilde konuşma diliyle yazacağım. Bu sebeple noktalama işaretlerinde hatalar yapılmış olabilir.

Django nedir önce kısa bir bilgi verelim. Django; Python kodlarıyla hazırlanmış bir web arayüzüdür. Yani Php ile hazırlanan Wordpress, OpenCart, NukePHP gibi Django'da Python kodlarıyla geliştirilmiş bir araçtır diyebiliriz. Tam olarak Türkçe karşılığı terimi bulamıyoruz ancak kısa sürede web siteleri oluşturmaınızı sağlar. Örnek olarak bir kaç web sitesi verebiliriz.

<http://disqus.com/>  
<http://pinterest.com/>  
<https://www.youtube.com/>  
<https://www.dropbox.com/>  
<http://instagram.com/>  
<http://support.mozilla.org/en-US/home>  
<http://dpaste.com/>  
<http://www.spormarket.com.tr>  
<http://www.grafson.com>  
<http://www.izmirteknikservis.tk>

Daha fazlası: <https://www.djangosites.org/>

Bir çok siteyi zaten tanıyorsunuz.

Bunlar size Django ile neler yapılabileceği hakkında fikirler verebilir. Python tüm işletim sistemlerinde rahatlıkla kullanıldığı için Django'yu da tüm işletim sistemlerinde kullanabilirsiniz. Symbian, ios, android, windows, raspberry pi gibi küçük bilgisayarlar üzerinde linux ve windows kurulumu yapılarakta kullanılabilir.

Her zaman söylediğim gibi **tüm yazılım dilleri aynıdır sadece syntax (yazım şekli) değişir**. Buradan yola çıkarak Django'yu gözünüzde zor diye büyütmeğe herşeyi yapabilirim diye büyütmeğe daha iyidir.

Peki Django neden bu kadar çok firma tarafından kullanılırken Türkiye'de pek fazla destek görmüyor?

Aslında görüyor ama klasik yazılım dilleri gibi yükle-kullan olarak kullanılamıyor. Klasik Asp, Php, Asp.NET gibi hosting alıp dosyalarınızı yükleyip hosting firmanız ile görüşerek hemen kullanmaya başlayamıyorsunuz. Django kullanmaya başladığınızda server (backend diyebiliriz) tarafını da sizin kontrol etmeniz gerekir. Bir çok firma artık size vps desteği

vererek ssh ile bağlanıp kendi serverınızı kullanarak Django ile çalışmanıza yardımcı oluyor. Hatta hazır Django kurulumu yaptırılabilir. Zamanında DjangoTürkiye.com'da bunu yapmıştık ama o zaman kimse Django'yu bilmiyordu, bu sebeple beklediğim gibi bir sıçrama olmamıştı.

Konuyu anlatmaya başlarken en azından bir kaç yazılım diliyle çalıştığınızı kabul ederek devam edeceğim. Hiçbir yazılım dili ile bir çalışma yapmadınız yada henüz karar vermediyseniz biraz zorlanacaksınız ama yine de fikriniz oluşacak.

Şu an Python'ı biliyor ve Django harici Python web arayüzlerini denediniz ise Django için hazırsınız diye düşünerek kurulum ve kullanıma başlayabiliriz.

Django projesi <https://www.djangoproject.com/> adresinde tanıtılmakta ve anlatılmaktadır. Sürekli gereksinimlere karşı da güncellenmektedir. İleri de sizde gruba dahil olabilirsiniz.

**Bu arada projeye bağış yapmayı unutmayın.** <https://www.djangoproject.com/fundraising/>

İlk başlayanlar veya geçiş yapanlar için gördüğüm en zor adım kurulum aşaması. Bu sebeple en uzun duracağım konu burası olacak.

Şu an son sürüm olarak **1.9 ( 1.9.1 )** hazırlandı. Önceki sürüm **1.8 ( 1.8.8 )** LTS yani uzun süre destek verilecek sürümdür. **Nisan 2018'e** kadar da destek verilecek.

Bu konu hakkında da sorular geliyor.

## Neden güncelleme yapılıyor? Neden önceden yapmadılar?

Kısa bir örnek vereyim hemen, HTML5 ile gelen yeni özellikler için bile eklemeler yapıldı. JSONField

Diğer birkaç konuya örnek gerekirse; sık kullanılan bir çok komutların tek fonksiyonda toplanması, güvenlik açıkları yamaları yada veritabanlarında yapılan değişikliklere eklenti sağlanması

Daha fazla merak edenler için:<https://docs.djangoproject.com/en/1.9/releases/>

Release Series	Release Date	End of mainstream support <sup>1</sup>	End of extended support <sup>2</sup>
1.10	August 2016	April 2017	December 2017
1.11 LTS <sup>3</sup>	April 2017	December 2017	Until at least April 2020
2.0	December 2017	August 2018	April 2019
2.1	August 2018	April 2019	December 2019
2.2 LTS	April 2019	December 2019	Until at least April 2022
3.0	December 2019	August 2020	April 2021

[1] Güvenlik düzeltmeleri ve veri kaybı hata, çökme hataları, yeni tanıtılan özellikleri önemli fonksiyonellik hataları, Django eski sürümlerinden gerilemeleri

[2] Güvenlik düzeltmeleri ve veri kaybı hata

[3] Python 2.7 ile desteklenen son sürüm

\*Daha iyi bir çeviri gerekir.

Muslu YÜKSEKTEPE – 2016

[www.muslu.org](http://www.muslu.org) | [www.djangoturkiye.com](http://www.djangoturkiye.com)

Bağış Hesabı: TR03 0006 2000 7500 0006 6675 10

# KURULUM

Kurulum için farklı işletim sistemlerinde bir çok yol mevcut. Ben Ubuntu üzerinde çalışıyorum ama diğer işletim sistemlerine de değineceğim. Zaten kurulumdan sonra yazım aşamasında farklılık yok.

1. olarak indir-kur yönetimine değinelim.

- <https://www.djangoproject.com/download/1.9.1/tarball/>

Yukarıdaki linkten Django-1.9.1.tar.gz (7.1MB) adında sıkıştırılmış bir dosya indirilecek.

\* Windows kullananlar için 7zip veya Winrar bu dosyayı açacaktır.

Sıkıştırılmış dosyası açtıktan sonra aşağıdaki gibi bir görüntü oluşacak.

## **Linux:**

```
muslu@muslu-MS-7641:~/İndirilenler/Django-1.9.1$ ls
AUTHORS CONTRIBUTING.rst django Django.egg-info docs extras Gruntfile.js INSTALL js_tests LICENSE MANIFEST.in
package.json PKG-INFO README.rst scripts setup.cfg setup.py tests
```

## **sudo python setup.py install**

komutu ile kurulumu başlatabiliriz.

## **django-admin -version**

komutu ile kurulumun doğru tamamlandığı kontrol edilir.

Tek satırda yapmak isterseniz:

```
cd İndirilenler/
```

```
tar -xzf Django-1.9.1.tar.gz && cd Django-1.9.1/ && sudo python setup.py install
```

```
django-admin -version
```

**&& kullanarak sırayla komut ekleyebilirsiniz.**

Alınabilecek hatalar:

**[Errno 13] Permission denied: '/usr/local/lib/python2.7/dist-packages/.....'**

komutun başına **sudo** eklemeyi unuttunuz ve yetkiniz yok.

**sudo python setup.py install**

olmalı

### ***Windows:***

Python kurulmuş ve PATH alanına eklenmiş olduğu yani komut satırında **python** komutuna izin verilmiş olması gerekir.

### **python setup.py install**

kurulum tamamlandıktan sonra komut satırına **python** yazarak python idesi açılabilir.

```
import django  
django.VERSION  
exit()
```

komutları djangonun kurulumu ve versiyonu kontrol edilebilir. Bu komutlar diğer işletim sistemleri içinde geçerlidir.

2. olarak **pip** kurulumuna değinelim.

**pip** kısaca python modüllerini kurmamız için geliştirilmiş bir paket yöneticisi.

Ubuntu:

```
sudo apt-get install python-pip
```

Windows:

```
python -m pip install -U pip
```

Mac OS:

```
sudo easy_install pip
```

pip kurulumundan sonra pip ile kurulumlara başlayabiliriz.

Kullanım örneği:

```
sudo pip install Django  
sudo pip install Django=1.7.7
```

Mac OS:

```
sudo pip install django
```

Windows:

```
pip install django
```

Toplu kurulum için  
**sudo pip install -r requirements.txt**

Ben genelde pip'i kullanmayı tercih ediyorum. Hem eski versiyonu otomatik kaldırıyor hemde tek bir dosya oluşturup tümünü bir arada kurabiliyoruz.

Kurulum aslında bu kadar zaten bir çok arkadaşta farklı platformlarda kurulumları gayet başarılı anlattılar.

Şu an için gerekli olarak görmedeğim **virtualenv**, shell kullanımı gibi konulara girmeyeceğim.

Kontrolleri sağladıktan sonra **Django** artık hazır, sizi bekliyor.

Django'nun bilinen bir açığı yok yani sayfanızın hacklenme gibi bir şey söz konusu değil. Tabi yazılımcı hataları olmazsa. Zaten html dosyasına yönlendirme sizin elinizde olduğu için biri gelip index.htm\* dosyalarının tümünü değiştirtse bile sayfanıza bir şey olmayacaktır.

Eğer bir hata alıyorsanız bu güzel bir şey demek ve mutlaka birileri bunu tecrübe edinmiş, önlemini almış ve açıklamasını yapmış. Mümkün olduğunca hatalar alıp bunların açıklamalarını yapacağım.

Okunabilirliği artırmak için yazı boyutlarını büyük tutup sık boşluk kullanmaya çalışıp, çok terim ve uzun cümleler kullanmaktan kaçındım sanırım böyle daha anlaşılır olmuştur.

## Düzenleme Aracı

Düzenleme aracı olarak ( editör ) ben **JetBrains**'e ait **Pycharm** ( Professional) kullanıyorum. İlk videolarda **Gedit** kullanmıştım ama artık **Pycharm** ile anlatacağım.

JetBrains editör konusunda çok başarılı. Tüm ürünlerini denedim ve fiyatları da gerçekten çok uygun.

<https://www.jetbrains.com/pycharm/download/#section=linux>

buradan kendi işletim sisteminize göre Pycharm'ı indirebilirsiniz. Otomatik olarak linux gelecek.

Hemen bir proje oluşturarak artık çalışmaya başlayabiliriz ama bazı terimleri şimdiden anlatmak gerekiyor.

Proje: **Wikipedia**' da “bir probleme çözüm bulma ya da beliren bir fırsatı değerlendirmeye yönelik, bir ekibin, başlangıcı ve bitişi belirli bir süre ve sınırlı bir finansman dahilinde, birtakım kaynaklar kullanarak, müşteri memnuniyetini ve kaliteyi göz önünde bulundururken olası riskleri yönetmek şartıyla, tanımlanmış bir kapsama uygun amaç ve hedefler doğrultusunda özgün bir planı başlatma, yürütme, kontrol etme ve sonuca bağlama sürecidir” diye tanımlanıyor.

Sürekli duyduğumuz bu terim aslında “**bir fikrim var**” yerine kullanılıyor. Oysa ki fikir henüz başlanmamış, eyleme geçilmemiş ve akılcılıkla ilgilidir. Oysaki proje zamanı belirlenmiş, başlanmış, planlanmış ve ekip olarak tasarlanmış fikirler ve eylemler.

Burada da proje; başlangıç olarak yapmayı istediğimiz web sitesinin genel adıdır.

Örnek olarak; teknik servis takibi, sağlık ocağı sıra takibi, kombin ürün satış sitesi vs..

## Artık başlayalım:

```
mkdir django
cd django
django-admin startproject teknikservistakibi
cd teknikservistakibi/
ls -la
```

projemize ait bir klasör oluşturu ve içinde **manage.py** dosyası ve proje adı ile aynı bir klasör daha oluşturuldu.

\* Django'nun eski versiyonlarında bu klasör oluşturulmuyor ve dosyalar direk dışarıda tutuluyordu.

\* Proje oluşturmayı Pycharm'dan da yapabiliriz ama komut olarak öğrenmeniz daha iyi. Çünkü her zaman bir editörünüz olmayacak ve her zaman local de çalışamayacaksınız.

```
muslu@muslu-MS-7641:~$ tree django/
django/
├── teknikservistakibi
│   ├── manage.py
│   └── teknikservistakibi
│       ├── __init__.py
│       ├── settings.py
│       ├── urls.py
│       └── wsgi.py
```

2 directories, 5 files

\* **tree** komutu için **sudo apt-get install tree**

**manage.py:** Proje ve uygulamalar ile ilgili komutları çalıştıracığımız yönetim dosyası.

**\_\_init\_\_.py:** Genel kullanımı bu klasörde python dosyaları var demek ( Python paketlerini içeren dizinler ) ama yine bir py dosyası olduğu için içine özel komutlar yada açıklamalar eklenebilir.

**settings.py:** Adından da anlaşılacağı gibi ayarların bulunduğu dosya. Projenin tüm ayrıntıları burada. Wordpress'teki config.php gibi..

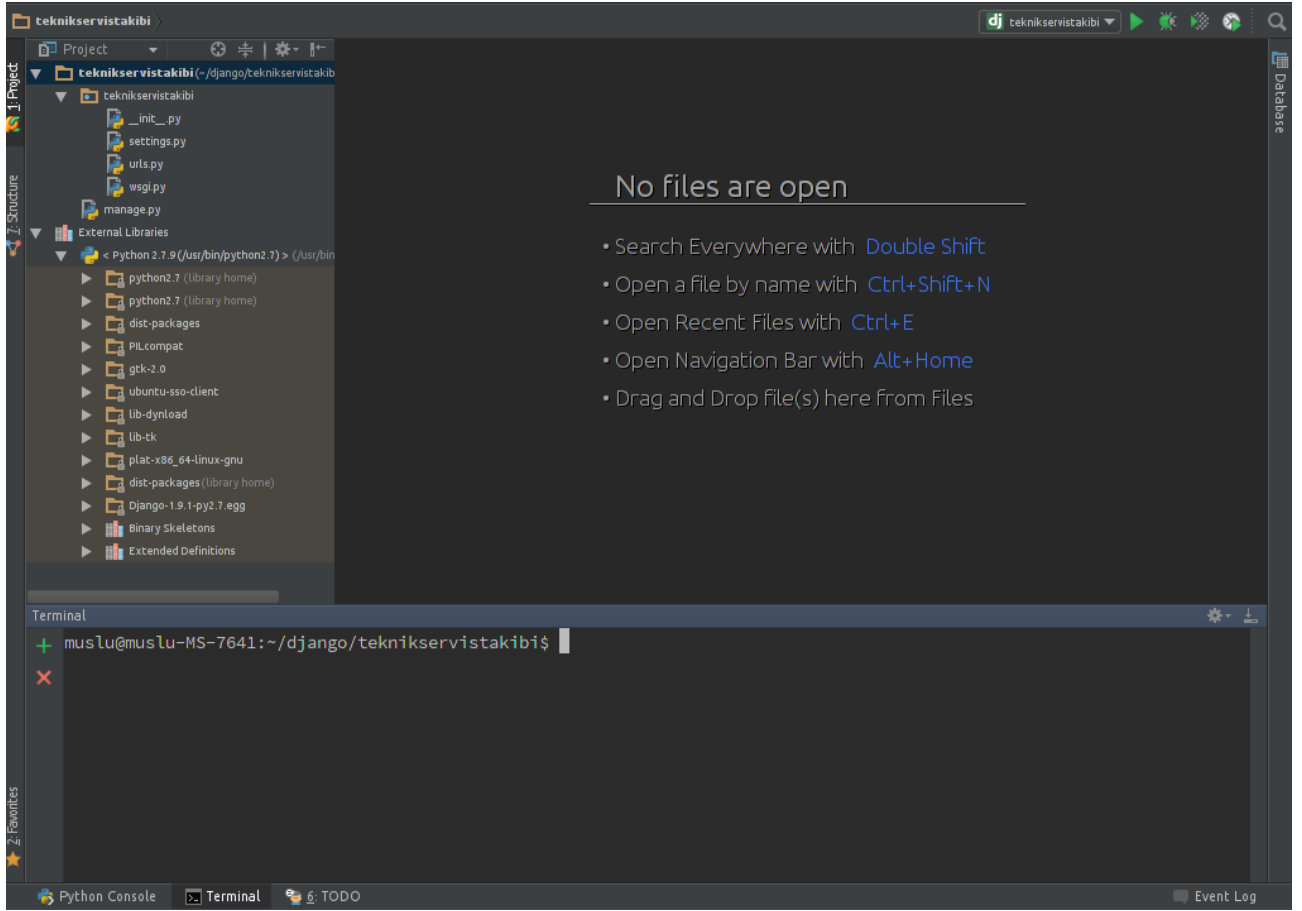
**urls.py:** Url yönlendirmelerinin yapılacağı dosya. Aynı zamanda fonksiyon da yazabiliriz.

**wsgi.py:** Http serverlar ( Örneğin **Apache** ve **Nginx**. \*libapache2-mod-wsgi ) için proje yönlendirme dosyası diyebiliriz, şu an çok detaya girmeye gerek duymuyorum.

İlk projemiz oluşturuldu. Şimdi bir kaç ayar yaparak ilk testi yapabiliriz.

Pycharm'ı başlatarak gelen ekrandan **Open** ile projenizin klasörünü (/django/teknikservisformu/) seçiyoruz.

\* Alt+F12 ile terminali açabilirsiniz.



Settings.py de ufak bir kaç ayar yaparak Türkçeleştirme yapıyoruz.

ctrl+g

Satır: 107-109

```
LANGUAGE_CODE = 'tr_TR'
```

```
TIME_ZONE = 'Europe/Istanbul'
```

Terminalde;

**python manage.py makemigrations && python manage.py migrate**

yazarak değişiklikleri onaylatıp, hata olup olmadığını kontrol ediyoruz.

\* 1.8 den sonra syncdb artık tamamen kullanılmıyor.



## python manage.py runserver

komutu ile Django'nun basit bir http serverını çalıştırıyoruz.

```
System check identified no issues (0 silenced).  
January 12, 2016 - 13:58:43  
Django version 1.9.1, using settings 'teknikservistakibi.settings'  
Starting development server at http://127.0.0.1:8000/  
Quit the server with CONTROL-C.
```

Burada karşılaşılabilecek hatalara değinelim.

### Error: That port is already in use.

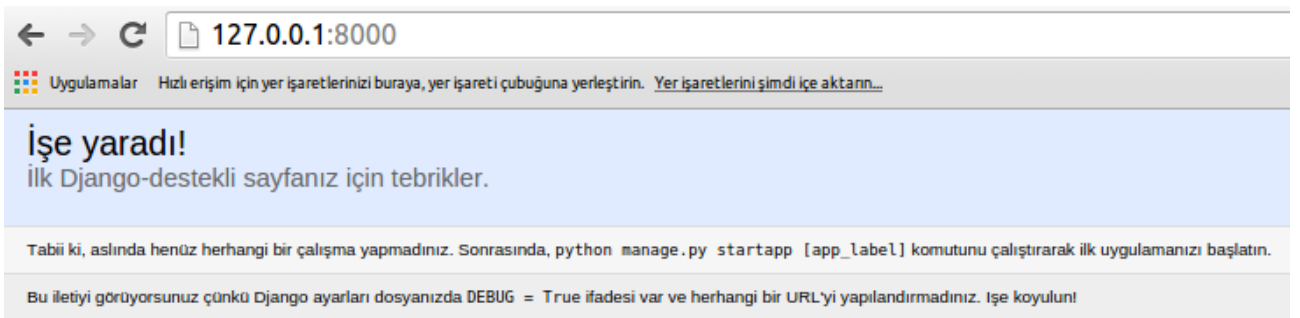
Eğer farklı bir komut satırında aynı proje ve/veya farklı bir proje çalışıyorsa bu hatayı alırsınız. Yani bu port zaten kullanılıyor.

İlla 2 proje çalıştırılması gerekiyorsa

## python manage.py runserver 127.0.0.1:8001

ile farklı bir port üzerinden çalıştırılabilir.

Diğer bir hususta; ip adresi alan başka bir cihazdan ( pc, telefon, tablet vs.. ) projenizi kontrol etmek isterseniz komut satırınızda ip adresinizi öğrenerek bu ip üzerinden yayın yapabilirsiniz. Böylelikle farklı tarayıcılarda nasıl gözüktüğüne bakabilirsiniz.



muslu@muslu-MS-7641:~/django/teknikservistakibi\$ ifconfig

```
muslu@muslu-MS-7641:~/django/teknikservistakibi$ ifconfig
eth0      Link encap:Ethernet  HWaddr d4:3d:7e:51:d4:3b
          inet addr:192.168.2.168  Bcast:192.168.2.255  Mask:255.255.255.0
          inet6 addr: fe80::d63d:7eff:fe51:d43b/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10623053  errors:0  dropped:0  overruns:0  frame:0
          TX packets:13422150  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:1000
          RX bytes:2270926581 (2.2 GB)  TX bytes:12170714775 (12.1 GB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128  Scope:Sunucu
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:44712  errors:0  dropped:0  overruns:0  frame:0
          TX packets:44712  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:0
          RX bytes:4300496 (4.3 MB)  TX bytes:4300496 (4.3 MB)

muslu@muslu-MS-7641:~/django/teknikservistakibi$
```

Diğer alınabilecek hatalar;

Port u belirtilmezseniz alacağınız hata

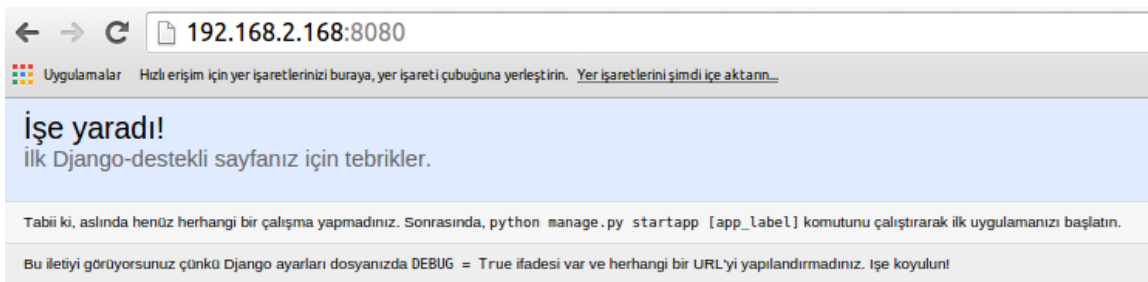
**CommandError: "192.168.2.168" is not a valid port number or address:port pair.**

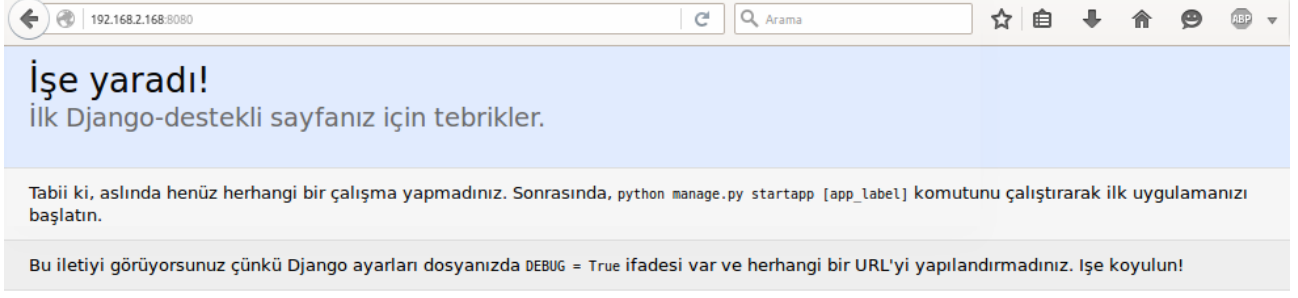
Port adresi belirttiğiniz halde farklı bir komut satırında yada bir http server ( apache ) yüklü ise yani bu port kullanılıyorsa.

**Error: You don't have permission to access that port.**

Münasip bir port bulduğunuzda deneme yapabilirsiniz.

**python manage.py runserver 192.168.2.168:8080**





Gördüğünüz gibi **Django** projemiz çalışmaya başladı ve **Türkçe** olarak yayınlanıyor.

Setting.py dosyamızdan dili değiştirip test edebilirsiniz. Neredeyse bilinen tüm dillere destek veriyor.

Alınabilecek diğer hata ise:

**CommandError: You must set settings.ALLOWED\_HOSTS if DEBUG is False.**

Eğer DEBUG modundan çıkmak isterseniz, yani hataların apaçık şekilde yayınlanmasını istemiyor, özelleştirilmiş bir html dosyasında gösterilmesini istiyorsanız ALLOWED\_HOSTS listesine kabul edilen ip ve adresleri yazmanız gerekir.

Örnek:

```
ALLOWED_HOSTS = ['192.168.2.168', '127.0.0.1', '.izmirteknikservis.tk']
```

www kullanmanız gerektiğinde **.domain.uzantisi** şeklinde yazabilirsiniz. İleride daha detaylı değineceğiz.

Değişikliği yaptıktan sonra artık bir sayfa gelmeyecek ve Not found uyarısı verecektir. Çünkü url olarak herhangi bir yönlendirme yapmadık.

Settings.py ile ilgili bir kaç noktaya daha değinelim ama ihtiyaç oldukça gerekli eklemeleri yapacağız.

import os

```
# Projenin bulunduğu klasöre ulaşmak için değişken
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
# benim proje yolum. BASE_DIR = /home/muslu/django/teknikservisformu/

# Hataların ekrana yansıtılması
DEBUG = False

# Çalışılacak domain isimler listesi
ALLOWED_HOSTS = ['192.168.2.168']

# Veritabanı seçimi, ayarları
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

Muslu YÜKSEKTEPE – 2016

[www.muslu.org](http://www.muslu.org) | [www.djangoturkiye.com](http://www.djangoturkiye.com)

Bağış Hesabı: TR03 0006 2000 7500 0006 6675 10

```

    }
}

# Kurulu uygulamalar. Yazdığımız uygulamaların listesi. Öncelik sırası var.
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

# Tüm projede geçerli olacak kodlar. Sıralamaya göre öncelik middleware lerdedi.
MIDDLEWARE_CLASSES = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.auth.middleware.SessionAuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

# urls.py dosyası
ROOT_URLCONF = 'teknikservistakibi.urls'

# Html dosyaları içinde gönderilecek veriler, ayarlar vs..
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

# Http serverlar için wsgi dosya adı ve uygulaması
WSGI_APPLICATION = 'teknikservistakibi.wsgi.application'

# Yetkilerde geçerli olan şifreleme yöntemleri
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

```

```
# Uluslararasılaşma
# https://docs.djangoproject.com/en/1.9/topics/i18n/
LANGUAGE_CODE = 'tr_TR'
TIME_ZONE = 'Europe/Istanbul'
USE_I18N = True
USE_L10N = True
USE_TZ = True
```

# Statik (CSS, JavaScript, Resimler) dosyaların çağıracağı url yolu. Fiziksel yol değildir dikkat edin. <http://127.0.0.1:80/static/> çağrıldığında css, js, resim vss gibi dosyalara ulaşılacak.

```
# https://docs.djangoproject.com/en/1.9/howto/static-files/
STATIC_URL = '/static/'
```

# Gizli kod. İleride gerekecek.

```
SECRET_KEY = '9f5$6e&r&x3*_a%j1ocv*p3aftgk1y5n&)+^jehhc@&z%@@@8p'
```

okunabilirlik için **boşluklar** ekledim ama bu yazım hali **PEP** standartlarına uygun değil zaten **Pycharm** da bu konuda uyaracaktır ancak **hata** olarak değil **uyarı** olarak.

**Yazım aşamasında iken Debug modunu True yapmanız gerekir.**

Alınacak hata:

**SyntaxError: Non-ASCII character '\xc4' in file /home/muslu/django/teknikservistakibi/teknikservistakibi/settings.py on line 3, but no encoding declared; see http://python.org/dev/peps/pep-0263/ for details**

Yani diyor ki; setting.py dosyamızın 3. satırında pep standartlarına uygun olmayan kodlama sorunu var. ( non-ascii dediği ) Yorum satırı olsa bile türkçe karakter kullanamayız.

# Projenin bulunduğu klasöre ulaşmak için değişken

Bu hata ile sık karşılaşacağız, bu sebeple her dosyanızın başına **# -\*- coding: utf-8 -\*-** (Bu dosyanın kodlama şekli utf-8 dir) eklemeniz gerekiyor. Her zaman birinci satıra eklenmesi gerekir.

Django'nun bir güzel tarafı da yönetim panelinin hazır gelmesi. Kullanıcılar, gruplar ve bunların yetkileri için auth modülü bizim için hazırlanmış.

```
INSTALLED_APPS = [
    'django.contrib.admin',
    ....
]
```

\* Bonus: `django.contrib.admin` aslında `/usr/local/lib/python2.7/dist-packages/Django-1.9.1-py2.7.egg/django/contrib/admin/` fiziksel yolundaki dosyaları eklemek ( import ) demek.

Urls.py dosyasını açtığınızda göreceğiniz gibi admin sayfasına ait url aktif geliyor.

Test etmek için tarayıcınızda <http://192.168.2.168:8080/admin/> adresini açabilirsiniz.

Tasarım bozuk geldi, çünkü static dediğimiz dosyalar yüklenmedi.

Komut satırından ( terminal ) kontrol edebiliriz.

```
[12/Jan/2016 15:47:41] "GET /admin/login/?next=/admin/ HTTP/1.1" 200 1697
[12/Jan/2016 15:47:41] "GET /static/admin/css/base.css HTTP/1.1" 404 99
[12/Jan/2016 15:47:41] "GET /static/admin/css/login.css HTTP/1.1" 404 100
```

admin sayfasındaki statik dosyalarını kendi projemize aktararak istediğimiz gibi düzenleyebiliriz.

Bunun için;  
terminalde ctrl+c ile çalışan komutu durdurup,

**python manage.py collectstatic**

yazmamız gerekir ama hata alacağız. Çünkü **STATIC\_ROOT** tanımlamasını yapmadık.

**django.core.exceptions.ImproperlyConfigured: You're using the staticfiles app without having set the STATIC\_ROOT setting to a filesystem path.**

Yani; **STATIC\_URL** isteği ile gelen linkin **fiziksel** karşılığını yazmalıyız. `/static/` olarak gelecek soruya, cevap olarak proje klasörümüzün altındaki static klasörünün ( izin demeyi pek tercih etmiyorum ) fiziksel yolunu vermek.

Settings.py dosyamızda aşağıdaki tanımlamaları yapıyoruz.

```
STATIC_ROOT = BASE_DIR + "/static/"
STATIC_URL = '/static/'
```

ek olarak projemizin medya dosyaları için ayrı bir klasör oluşturarak admin statik dosyalarından ayırmamız daha iyi olacak.

```
MEDIA_ROOT = BASE_DIR + '/media/'  
MEDIA_URL = '/media/'
```

```
URL = Fiziksel yol  
http://192.168.2.168:8080/static/ = /home/muslu/django/teknikservisformu/static/  
http://192.168.2.168:8080/media/css/stil.css = /home/muslu/django/teknikservisformu/media/css/stil.css
```

terminalden yada dosya yöneticisi ile static ve media adında klasörlerimizi oluşturalım.

\* static klasörünü oluşturmasanız bile collectstatic komutu oluşturacak.

```
mkdir static  
mkdir media
```

Tekrar collectstatic ile statik dosyaları projemize kopyalayabiliriz.

**python manage.py collectstatic**

Uyarı olarak /home/muslu/django/teknikservistakibi/static klasörüne kopyalanacak ne dersiniz diye soruyor.

yes

**ls static/**

diyerek yada proje klasörümüzdeki static klasörüne bakarak admin klasörünün oluşup oluşmadığını kontrol edebiliriz.

**python manage.py runserver 192.168.2.168:8080**

ile tekrar projemizin yayınını başlatalım ve tarayıcıda <http://192.168.2.168:8080/admin/> sekmemizi yenileyelim.

Alınabilecek hata:

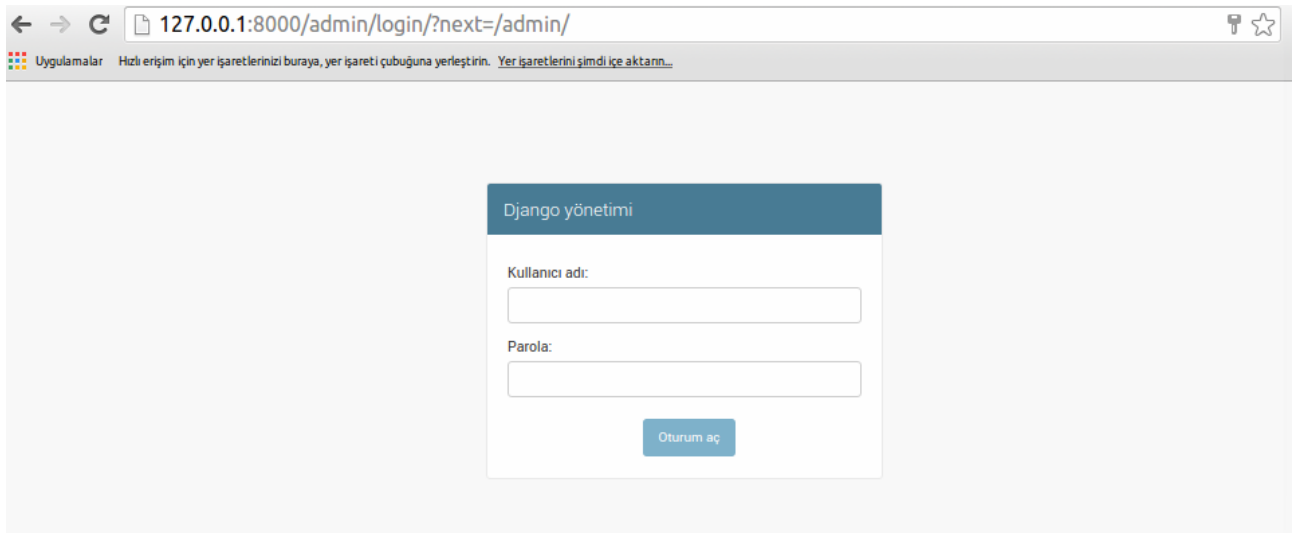
**Bad Request (400)**

Debug modu hala False!

**Statik dosyalar hala 404 veriyor.**

Debug modu hala False.

Herhangi bir sorun yaşamadınız ise aşağıdaki gibi yönetim paneli gelecek.



Yönetim paneline giriş yapabilmemiz için super yetkili bir kullanıcı oluşturmalıyız.

Daha sonra super olmasa da yetkili kişilerin girmesi için panelden kullanıcı oluşturacağız.

\*Django 1.8 öncesinde **syncdb** ile yetkili kullanıcıda oluşturabiliyorduk ama artık komut ile oluşturmamız gerekiyor.

Terminalde ctrl+c ile yayını durdurup

**python manage.py createsuperuser**

komutu ile super kullanıcı oluşturabiliriz.

Sorulara cevap verdikten sonra yetkili kullanıcı oluşturabilirsiniz. Bu komutu unutmayın ileride şifreyi unutursanız başvuracaksınız.



Alınabilecek hatalar:

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

**This password is entirely numeric.**

Şifreniz çok kısa. En az 8 karakter olması gerekiyor.

**This password is too common.**

Klasik bir şifre seçimi yapıldı.

**The password is too similar to the email address.**

Email adresi ile benzer şifre seçildi.

**Error: Your passwords didn't match.**

Yazılan bilgiler aynı değil.

Bu parola doğrulama ve oluşturma seçenekleri 1.9 ile geldi.

<https://allmychanges.com/p/python/django/>

Settings.py de ki **AUTH\_PASSWORD\_VALIDATORS** listesinden istediğiniz ( istemediğiniz ) kontrol şekillerini kaldırabilirsiniz.

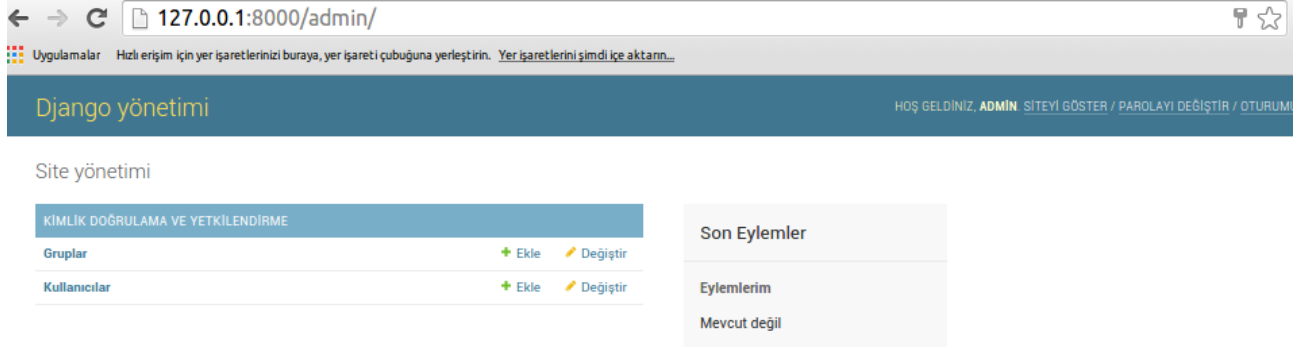
```
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    # {
    #     'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    # },
    # {
    #     'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    # },
]
```

Uygun bir şifre seçtikten sonra

**Superuser created successfully.**

Bilgisini alacağız.

Yönetim sayfamızı açarak <http://127.0.0.1:8000/admin/login/?next=/admin/> giriş yapabiliriz.



Yönetim panelimiz açıldı.

13 Ocak 2016

Django'nun kendine has yönetim paneli temasını aslında beğenmiyorum, ileri de **suit** panelinin kurulum ve kullanılmasına değineceğim.

Önce admin paneli temasını nasıl değiştiririz konusuna bakalım.

Django'nun admin panel dosyalarını kendi projemizdeki templates klasörüne taşıdığımızda (statik dosyaları gibi) istediğimiz gibi düzenleme yapabiliriz.

Proje klasörümüzde **templates** adında bir klasör oluşturalım.

Bu klasörde **html, txt ve xml** dosyalarımızı saklayacağız.

index.html, robots.txt, sitemap.xml, vs..

Bu templates klasörümüzün Django tarafından geçerli olması için de settings.py dosyamızda düzenleme yapmamız gerekiyor.

Templates ayar listemizdeki DIRS değişkenini proje klasörümüzün altındaki templates klasörü olarak güncelliyoruz.

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
        'APP_DIRS': True,
        .....,
    },
]
```

Şimdi Django'nun kurulduğu klasör yolunu bulalım.

Terminal açarak aşağıdaki komutu yazın.

**Linux, Mac OS, Windows:**

**python -c "import django; print(django.\_\_path\_\_)"**

Bonus: python -c "....." ile python kodlarını çalıştırabilirsiniz.

Bonus: "... ; ....." ile tek satırda kod yazabilirsiniz.

**Windows:**

C:\Python27\lib\site-packages\django\contrib\admin\templates\

Dosya yöneticiniz ile bu klasörü açarak **admin** klasörünü proje klasörünüzdeki **templates** klasörünü kopyalamanız yeterli.

**Linux, Mac OS:**

['/usr/local/lib/python2.7/dist-packages/Django-1.9.1-py2.7.egg/django']

İsterseniz dosya yöneticiniz ile kopyalamayı yapabilirsiniz

/usr/local/lib/python2.7/dist-packages/Django-1.9.1-py2.7.egg/django/contrib/admin/templates/admin klasörünü proje klasörünüzün altında oluşturduğunuz templates klasörüne kopyalayabilir

yada terminalde;

**cd ~/django/teknikservistakibi/templates**

komutu ile **proje klasörümüzdeki templates** klasörüne geçiyoruz.

**cp -a /usr/local/lib/python2.7/dist-packages/Django-1.9.1-py2.7.egg/django/contrib/admin/templates/admin/. !**

**Dikkat: 2 adet nokta var. Birisi admin/ klasörünün altındaki tüm dosyalar demek, diğeri bulunduğumuz klasör demek.**

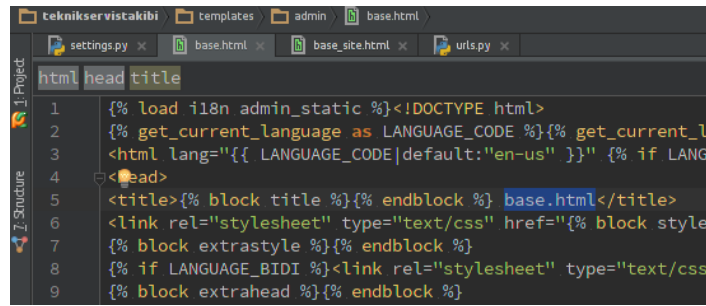
Hata almaktan çekiniyor yada anlaşılmadıysa alttaki komutları kullanabilirsiniz.

- `cd /usr/local/lib/python2.7/dist-packages/Django-1.9.1-py2.7.egg/django/contrib/admin/templates/`
- `cp -a admin/ ~/django/teknikservistakibi/templates/`
- `cd ~/django/teknikservistakibi/templates/`
- `cd` ile djangonun admin templates klasörüne gittik
- `cp -a` ile admin ve alt klasörlerini proje klasörümüzdeki templates klasörümüze kopyaladık.
- `cd` ile tekrar proje klasörümüzdeki templates klasörümüze döndük.

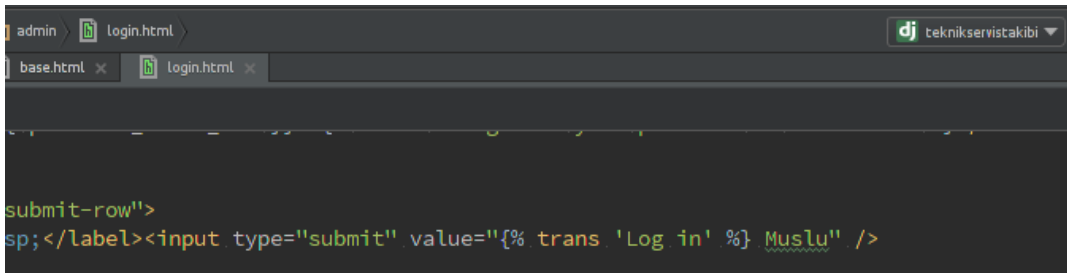
`ls`, `tree` veya dosya yöneticiniz ile kopyala yapılıp yapılmadığını kontrol edebilirsiniz.

Kopyalamayı tamamladıktan sonra deneme yapabiliriz.

Pycharm da templates admin base.html ve base\_site.html dosyalarını açalım ve title etiketlerini değiştirelim.



```
1 {% load i18n admin_static %}<!DOCTYPE html>
2 {% get_current_language as LANGUAGE_CODE %}{% get_current_la
3 <html lang="{% LANGUAGE_CODE|default:'en-us' %}" {% if LANGU
4 <head>
5 <title>{% block title %}{% endblock %} base.html</title>
6 <link rel="stylesheet" type="text/css" href="{% block styles
7 {% block extrastyle %}{% endblock %}
8 {% if LANGUAGE_BIDI %}<link rel="stylesheet" type="text/css"
9 {% block extrahead %}{% endblock %}
```

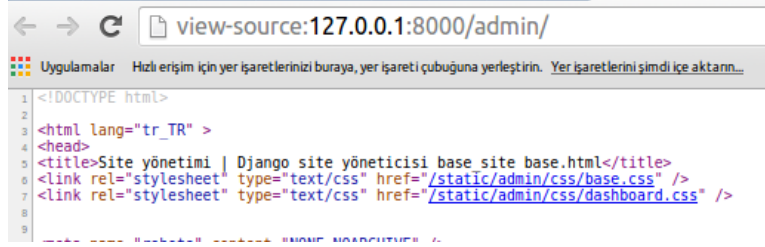


```
submit-row">
sp;</label><input type="submit" value="{% trans 'Log in' %} Muslu" />
```

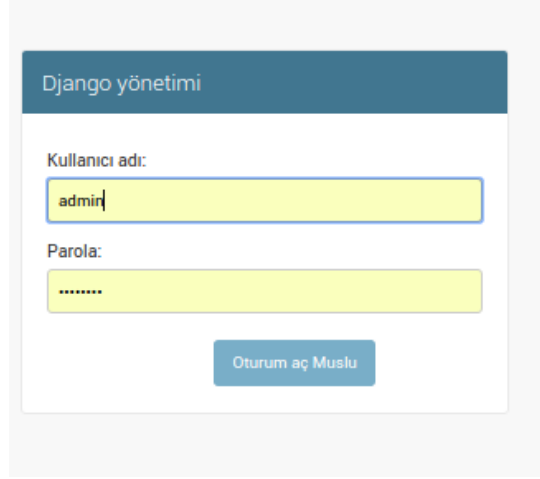
python manage.py runserver ile serverı başlatalım.

<http://127.0.0.1:8000/admin/>

tarayıcınızın üst kısmına baktığımızda yada kaynak kodları kontrol ettiğimizde değişikliklerin çalıştığını göreceğiz.



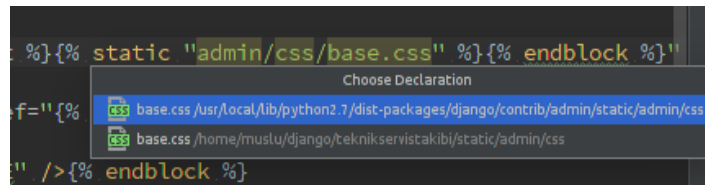
```
<!DOCTYPE html>
<html lang="tr_TR" >
<head>
<title>Site yönetimi | Django site yöneticisi base_site base.html</title>
<link rel="stylesheet" type="text/css" href="/static/admin/css/base.css" />
<link rel="stylesheet" type="text/css" href="/static/admin/css/dashboard.css" />
```



Bundan sonra istediğiniz gibi değişiklikleri yapabilirsiniz. İleri seviyelerde debug\_tool kullanarak yayınlanan sayfaları görerek hangi sayfaları değiştirmemiz gerektiğini de öğreneceğiz.

**Bonus:** Pycharm'da ctrl tuşuna basılı tutarken bir fonksiyon, script yolu, stil classı vs.. tıkladığınızda dosya yolu gösterir yada direk fonksiyonun bulunduğu dosyaya götürür.

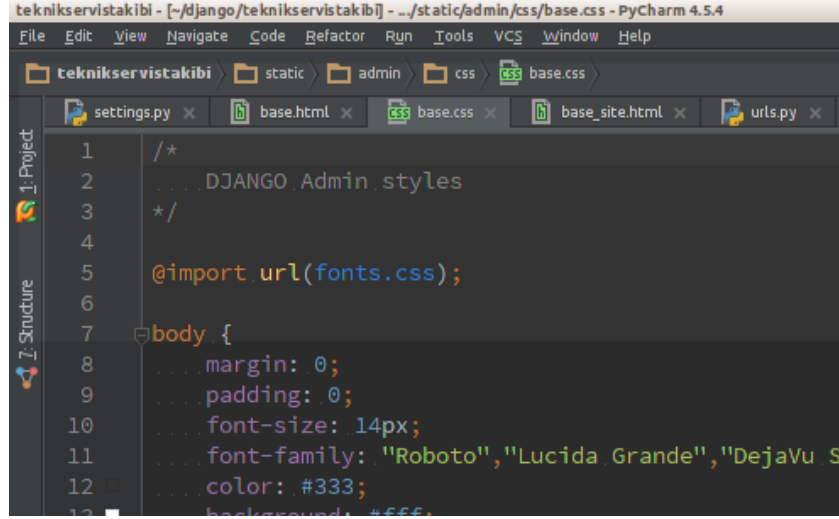
base.css metni üzerinde ctrl tuşuna basılı tutarak fare sol tuşu ile tıkladığınızda aşağıdaki gibi yolları gösterecektir ve hangisi tıklarsanız o dosyayı açacaktır.



Burada görüldüğü gibi bu stil dosyası 2 yerde mevcut.

Collectstatic komutu ile statik dosyalarını static klasörümüze kopyalamıştık.

Biz static klasörümüzdekileri kullanmak istiyoruz ancak bu konuya daha sonra Apache ile değineceğiz. Yönetim paneline ait html dosyaları değıştirmeniz yeterli.



```
tekni servistakibi - [~/django/tekniservistakibi] - ../static/admin/css/base.css - PyCharm 4.5.4
File Edit View Navigate Code Refactor Run Tools VCS Window Help
tekni servistakibi static admin css base.css
settings.py x base.html x base.css x base_site.html x urls.py x
1 /*
2 ... DJANGO Admin styles
3 */
4
5 @import url(fonts.css);
6
7 body {
8     margin: 0;
9     padding: 0;
10    font-size: 14px;
11    font-family: "Roboto","Lucida Grande","DejaVu S
12    color: #333;
13    background-color: #fff;
```

Şimdi kısaca hazır gelen kimlik doğrulama ve yetkilendirme uygulamalarına bakıp bir uygulama yazmaya başlayalım.

Gruplar --> Ekle linki ile grup ekleme sayfasını açıyoruz. Grup adının altında ManyToMany ile çekilmiş verileri görüyoruz. Burada tüm uygulama yetkileri mevcut. Yazacağımız uygulamalarda burada gözükecek.

Örnek olarak finans, teknik servis, kargo, satış vs.. gibi gruplar oluşturulup kişilere ekleme, silme ve/veya düzenleme yetkileri toplu olarak verilebilir.

## grup ekle

Adı:

İzinler:

Mevcut izinler

Q Süzgeç

- admin | günlük girdisi | Can add log entry
- admin | günlük girdisi | Can change log entry
- admin | günlük girdisi | Can delete log entry
- auth | grup | Can add group
- auth | grup | Can change group
- auth | grup | Can delete group
- auth | izin | Can add permission
- auth | izin | Can change permission
- auth | izin | Can delete permission
- auth | kullanıcı | Can add user
- auth | kullanıcı | Can change user
- auth | kullanıcı | Can delete user

Tümünü seçin

Seçilen izinler

Tümünü kaldır

Birden fazla seçmek için "Control (Ctrl)" veya Mac'deki "Command" tuşuna basılı tutun.

[Kaydet ve başka birini ekle](#) [Kaydet ve düzenlemeye devam et](#) [KAYDET](#)

## Kullanıcı Ekleme:

## kullanıcı ekle

Önce, bir kullanıcı adı ve parola girin. Ondan sonra, daha fazla kullanıcı seçeneğini düzenleyebilirsiniz.

Kullanıcı adı:   
Zorunlu. 30 karakter ya da daha az olmalı. Sadece harfler, rakamlar ve @/./+/\_ karakterleri kullanılabilir.

Parola:

Parola onayı:   
Doğrulama için önceki gibi aynı parolayı girin.

Bu ekranda kullanıcı için ad ve parola girildikten sonra detaylı bilgilerin geleceği bir sayfa gelecek.

**Kullanıcı adı:**

Zorunlu. 30 karakter ya da daha az olmalı. Sadece harfler, rakamlar ve @/./+/\_/ karakterleri kullanılabilir.

**Parola:** **algorithm:** pbkdf2\_sha256 **yinelemeler:** 24000 **tuz:** 2oLOV0\*\*\*\*\* **adresleme:** RE96ve\*\*\*\*\*

Ham parolalar saklanmazlar, bu yüzden bu kullanıcının parolasını görmenin yolu yoktur, fakat **bu formu** kullanarak parolayı değiştirebilirsiniz.

---

**Kişisel bilgiler**

**Adı:**

**Soyadı:**

**E-posta adresi:**

---

**İzinler**

☒ **Etkin**  
Bu kullanıcının etkin olarak işlem görüp görmediğini belirler. Hesapları silmek yerine bunun işaretini kaldırın.

☐ **Görev durumu**  
Kullanıcının bu yönetici sitesine oturum açıp açamayacağını belirler.

☐ **Süper kullanıcı durumu**  
Bu kullanıcıya ayrı ayrı izin atamadan tüm izinlerin verilip verilmeyeceğini belirler.

**Gruplar:**

**Mevcut gruplar** ⓘ

**Kullanıcı Ekle Değiştir**  
**Kullanıcı Sil**

**Seçilen gruplar** ⓘ

**Artık bir uygulama yazmaya başlayalım.**

Terminalde proje klasörünüze geçerek

**./manage.py startapp servisformu**  
**python manage.py startapp servisformu**  
**django-admin startapp servisformu**

herhangi birini yazarak uygulamayı başlatabilirsiniz.



Alınabilecek hatalar:

**CommandError: '/home/muslu/django/teknikservistakibi/servisformu' already exists**  
Uygulama zaten oluşturulmuş, farklı bir isim seçilmeli.

**CommandError: 'django' conflicts with the name of an existing Python module and cannot be used as an app name. Please try another name.**

Python modül isimleri uygulama adı olarak kullanılamaz. Örneğin; django, math

İlk uygulamanın oluşturulmasıyla proje klasörümüzde uygulamamızın adı ile bir klasör daha oluşturuldu.

```
Terminal
+ muslu@muslu-MS-7641:~/django/teknikservistakibi$ tree
X .
├── db.sqlite3
├── manage.py
├── media
├── servisformu
│   ├── admin.py
│   ├── apps.py
│   ├── __init__.py
│   ├── migrations
│   │   └── __init__.py
│   ├── models.py
│   ├── tests.py
│   └── views.py
└── static
    ├── admin
    │   └── css
    │       └── base.css
```

servisformu klasöründe aşağıdaki dosya ve klasörler oluşturuldu.

**admin.py:** Uygulamanın admin sayfasına ait ayarların yapılacağı dosya. Models.py den gelen alanları tanımlayıp, filtreleyip, kısıtlayabiliriz.

**apps.py:** Uygulamanın adı ve diğer ayarlarının yapılacağı dosya

**\_\_init\_\_.py:** Klasik python dosyası, zaten değinmiştik.

**migrations:** Veritabanına ait güncellemelerin ve değişikliklerin kolay kullanım için tutulacağı klasör

**models.py:** Veritabanında oluşturacağımız tablo ve alanları yazacağımız dosya. Herhangi bir sql bilginiz olmasa bile kolayca yönetebileceğiz

**tests.py:** Uygulamanın bazı testleri deneyebileceğimiz dosya

**views.py:** Veritabanından gelen bilgilerin yada kendi tanımladığımız değişken yada verilerin html, txt yada xml dosyalarına yönlendirileceği dosya. Ayrıca sadece ekrana bilgi de bastırabiliriz.

Url den gelen sorguların sırayla ilerlemesini şöyle anlayabiliriz.

Tarayıcıya yazılan url önce middleware da denetlenir. Bu konuda şu an uzun durmayacağız. Daha sonra urls.py ye eklediğimiz url yönlendirmesi ile view.py ye, burada da models.py den gelen veriler tekrar templates klasöründeki dosyalara yönlendirebiliriz.

Aklınızın karışmaması için grafiksel anlatmak daha doğru olacaktır.

Basit şekilde şöyle sıralanabilir.

url.py --> views.py --> models.py --> views.py --> html

**http://127.0.0.1/formlar/ --> def formlar(request) --> class FormBilgileri(models.Model) --> def formlar(request) --> formlistesi.html**

Açıklama olarakta şöyle olabilir.

Tarayıcıdan gelen link urls.py de hangi fonksiyona tanımlandı ise buradan da models.py de tanımlanan tablodaki alanlar alınıp render edilerek html dosyasına gönderilir.

Hemen bir örnek yazarak anlaşılır hale getirelim.

Pycharm da **teknikservis** klasöründen **models.py** yi açalım ve aşağıdaki gibi kodları ekleyelim.

```
# -*- coding: utf-8 -*-
```

```
### utf-8 kodlama
```

```
from django.utils import timezone
```

```
### KayıtTarihi alanımız için otomatik bugünü seçtirme fonksiyonu
```

```
from django.db import models
```

```
### django'nun hazır modelleri. Buradan bir çok hazır alanları seçebiliriz.
```

```
class Teknisyen ( models.Model ) :
```

```
### Teknisyen adında bir tablo oluşturuyoruz.
```

```
    Aktif = models.BooleanField(default = 1)
```

```
    ### Teknisyenin aktif olup olmadığını seçmek için booleanfield kullanacağız. Default olarakta seçili gelecek.
```

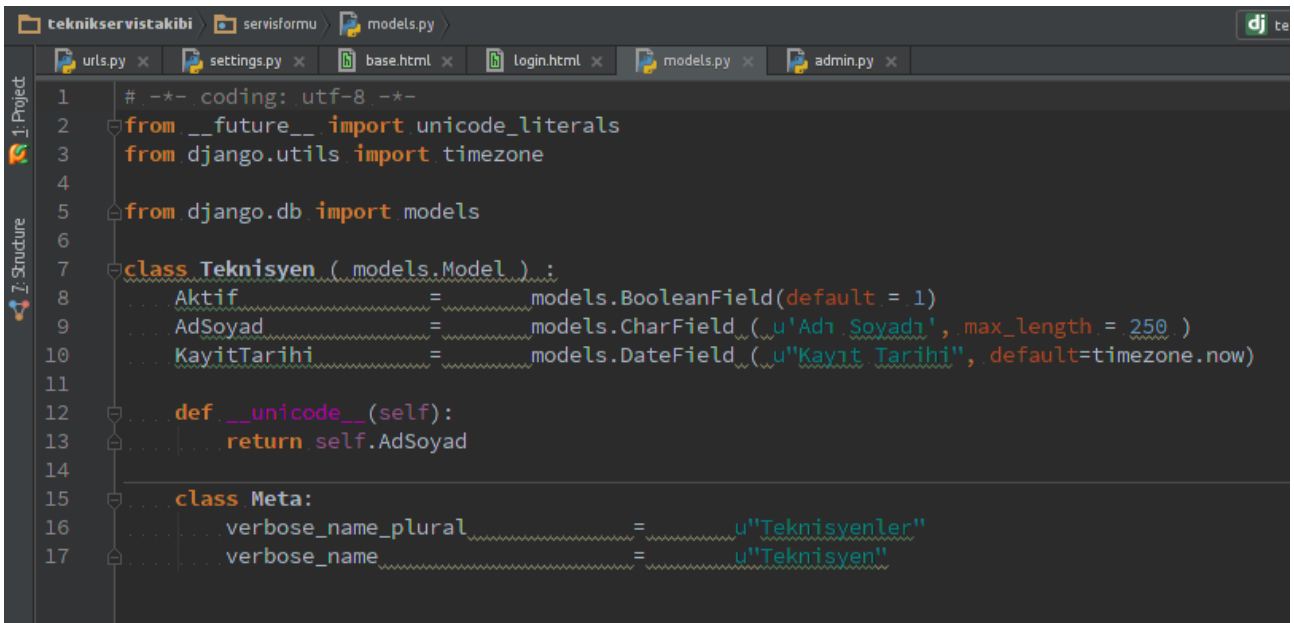
```
    AdSoyad = models.CharField ( u'Adı Soyadı', max_length = 250 )
```

```
    ### Teknisyenin ad ve soya dını gireceğimiz en fazla 250 karakterlik bir varchar alanı. 255 karaktere kadar yazılabilir.
```

```
KayitTarihi = models.DateField ( u"Kayıt Tarihi", default=timezone.now)
### kayıt işlemi yapıldığında otomatik bugünü seçecek ve gözükmeyecek.
```

```
def __unicode__(self):
    ### daha önceden __str__ kullanılıyordu. Models çağrıldığında burada seçilen alan ve/veya alanlar
    döndürülür. Birazdan göreceğiz.
    return self.AdSoyad
```

```
class Meta:
    ### admin sayfasında bu uygulamanın nasıl isimlendirilip çağırılacağı tanımlamalar
    verbose_name_plural = u"Teknisyenler"
    verbose_name = u"Teknisyen"
```



```
1  # -*- coding: utf-8 -*-
2  from __future__ import unicode_literals
3  from django.utils import timezone
4
5  from django.db import models
6
7  class Teknisyen ( models.Model ) :
8      Aktif = models.BooleanField(default = 1)
9      AdSoyad = models.CharField ( u'Adı Soyadı', max_length = 250 )
10     KayitTarihi = models.DateField ( u'Kayıt Tarihi', default=timezone.now)
11
12     def __unicode__(self):
13         return self.AdSoyad
14
15     class Meta:
16         verbose_name_plural = u"Teknisyenler"
17         verbose_name = u"Teknisyen"
```

\*\* kafa karıştırmamak için from \_\_future\_\_ import unicode\_literals kodlamalarına şu an değinmiyoruz.

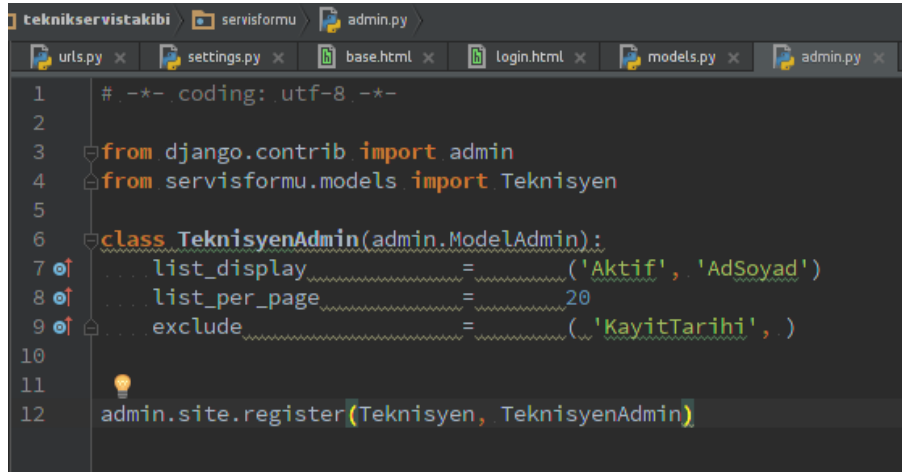
admin.py dosyasını açıp aşağıdaki gibi eklemeleri yapalım.

```
# -*- coding: utf-8 -*-
```

```
from django.contrib import admin
from servisformu.models import Teknisyen
### models.py dosyamızdaki Teknisyen ( class - sınıf ) tablomuzu ve alanları ekliyoruz.
```

```
class TeknisyenAdmin(admin.ModelAdmin):
    ### Admin sayfasında gösterilecek detaylar
    list_display = ('Aktif', 'AdSoyad')
    ### sırayla gösterilecek alanlar
    list_per_page = 20
    ### sayfadaki kayıt adeti, otomatik sayfalama yapacak
    exclude = ('KayitTarihi',)
    ### KayitTarihi alanını gizliyoruz
```

```
admin.site.register(Teknisyen, TeknisyenAdmin)
### Teknisyen ve TeknisyenAdmin sınıflarını kayıt ettiriyoruz.
```

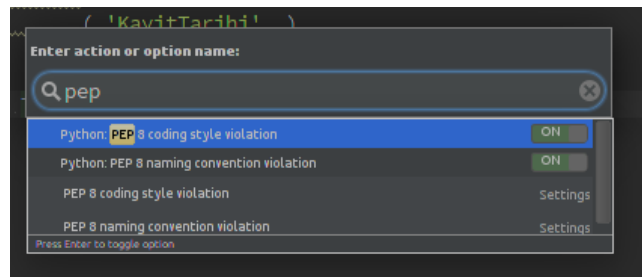


```
1  # -*- coding: utf-8 -*-
2
3  from django.contrib import admin
4  from servisformu.models import Teknisyen
5
6  class TeknisyenAdmin(admin.ModelAdmin):
7      list_display = ('Aktif', 'AdSoyad')
8      list_per_page = 20
9      exclude = ('KayitTarihi', )
10
11
12  admin.site.register(Teknisyen, TeknisyenAdmin)
```

Artık uygulamamızı projemize dahil edebiliriz.  
settings.py dosyamıza eklememizi yapalım.

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'servisformu'
]
```

Bonus: pep standartlarını geçici olarak pasif etmek isterseniz, ctrl+shift+a ile arama ekranını açıp pep yazabilir ve ON olan kısımları OFF yapabilirsiniz.



Terminalde;

./manage.py makemigrations servisformu && ./manage.py migrate && ./manage.py runserver  
yazarak yaptığımız değişiklikleri ekleyip, onaylatıp serverımızı çalıştırıyoruz.

<http://127.0.0.1:8000/admin/>



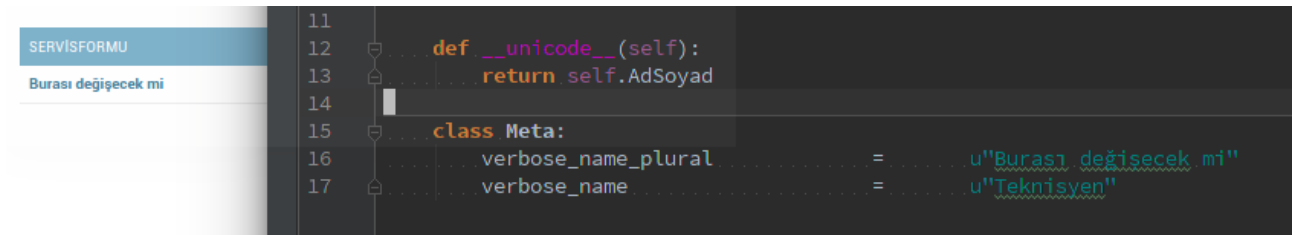
Uygulamamız yönetim sayfamızda artık hazır.

Veri ekleyip html dosyasına gönderilmeden önce yönetim sayfasında yapabileceğimiz değişikliklere bakalım.

Models.py deki

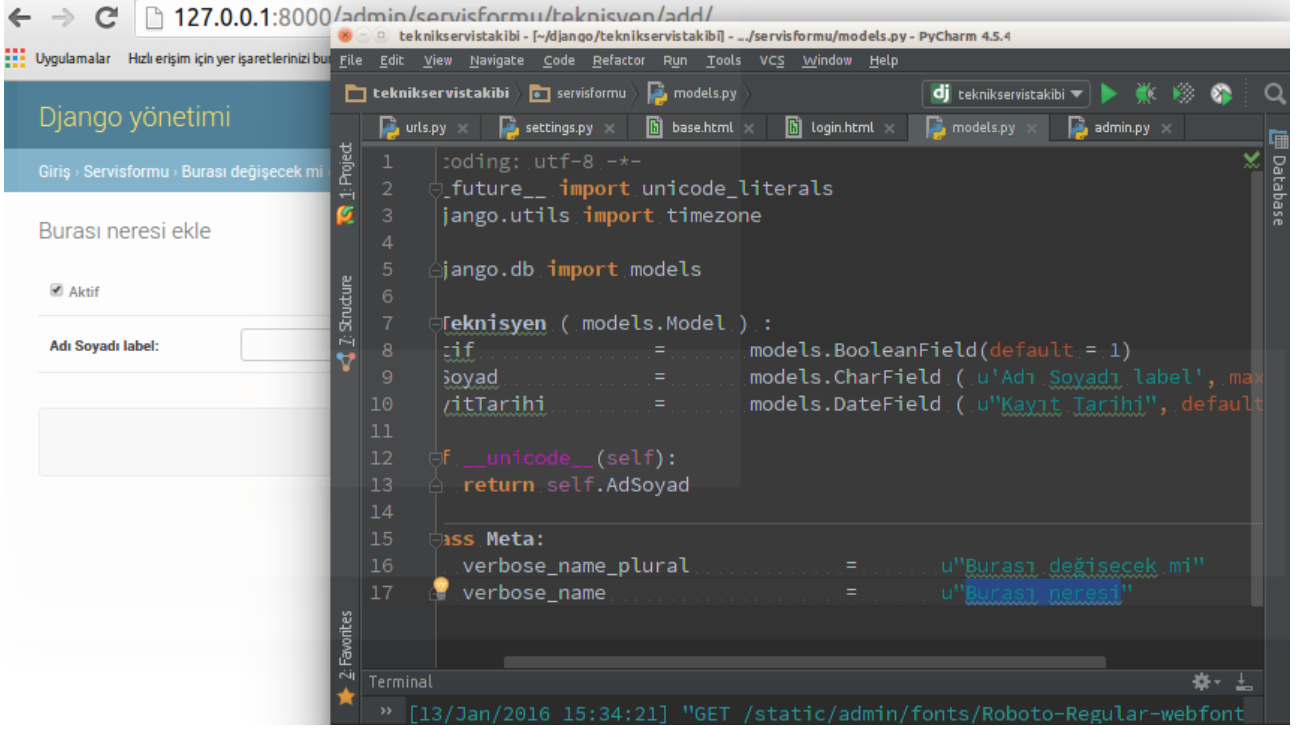
```
verbose_name_plural = u"Teknisyenler"
```

tanımlamamızı değiştirerek deneme yapalım.



```
verbose_name = u"Teknisyen"
```

tanımlamasını değiştirerek test edebiliriz.



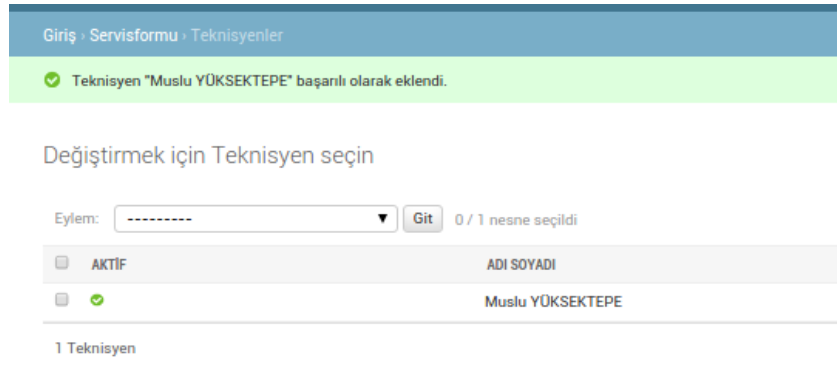
Değişiklikleri geri alarak kayıt girildikten sonra tüm listeyi kontrol edelim.

Teknisyen "Muslu YÜKSEKTEPE" başarılı olarak eklendi.

Bilgisini geldiği yer \_\_unicode\_\_ da eklediğimiz alanın geri dönüşü

Grid sistemindeki gelen bilgiler ise list\_display de eklediğimiz alanlar ve sıralaması ile oluşmakta.

Bu gird sistemi djangonun kendi oluşturduğu alandır ve klasik olarak tüm uygulamalarda kullanılmaktadır.



Hemen buraya kendi alanımızı ekleyelim.

```
class Meta:
    verbose_name_plural = u'Teknisyenler'
    verbose_name = u'Teknisyen'

def Yazdir ( self ) :
    return '<a href="/yazdir/%s" target="_blank">Yazdır</a>' % self.id
Yazdir.short_description = u'Yazdır'
Yazdir.allow_tags = True
```

models.py de class içine aşağıdaki gibi bir fonksiyon oluşturalım

```
def Yazdir ( self ) :
```

```
    return '<a href="/yazdir/%s" target="_blank">Yazdır</a>' % self.id
```

### Fonksiyon çağırıldığında döndürülecek metin.

```
    Yazdir.short_description = u'Yazdır'
```

### Fonksiyonun kısa açıklaması

```
    Yazdir.allow_tags = True
```

### Fonksiyonumuz html etiket içeriyor

admin.py deki list\_display oluşturduğumuz fonksiyon adını ekleyelim.

```
list_display = ('Aktif', 'AdSoyad', 'Yazdir')
```

<http://127.0.0.1:8000/admin/servisformu/teknisyen/>

Bir alan daha ekleyelim.

```
def EkAlanTest(self):
```

```
    return self.AdSoyad.replace(' ', '_____')
```

```
EkAlanTest.short_description = u'Burası alanın başlığı'
```

```
list_display = ('Aktif', 'AdSoyad', 'Yazdir', 'EkAlanTest')
```

Eylem:   0 / 1 nesne seçildi

<input type="checkbox"/>	AKTİF	ADI SOYADI	YAZDIR	BURASI ALANIN BAŞLIĞI
<input checked="" type="checkbox"/>		Muslu YÜKSEKTEPE	Yazdır	Muslu_____YÜKSEKTEPE

1 Teknisyen

Alınabilecek hatalar:

<class 'servisformu.admin.TeknisyenAdmin'>: (admin.E116) The value of 'list\_filter[2]' refers to 'Yazdir', which does not refer to a Field.

TeknisyenAdmin'deki list\_filter değişken listesindeki 3. değişken geçerli bir field alanı değil.

Yani Yazdir fonksiyonu özel bir tanımlama olduğu için list\_filter da kullanılamaz.

Kayıtlara daha kolay ulaşmak ve filtreleme yapmak için Django'nun hazır bir kaç fonksiyonlarına daha bakalım...

```
list_filter = ['Aktif', 'AdSoyad',]
```

### Filtre ( süzgeç ) yapabilmemiz için hazır sorgu alanı

```
search_fields = ['AdSoyad',]
```

### Arama yapabilmemiz için otomatik bir met

in alanı oluşturur. Arayacağımız kelimeler AdSoyad'a girdiğimiz kayıtlarda sorgulayacak.

```
date_hierarchy = 'KayitTarihi'
```

### Kayıtları yıl, ay ve gün olarak otomatik filtreleme yaptırmak için gird üstünde listeleme yapar

Değiştirmek için Teknisyen seçin

2016 Ocak 13 date\_hierarchy

Eylem:  0 / 1 nesne seçildi

<input type="checkbox"/>	AKTİF	ADI SOYADI	YAZDIR	BURASI ALANIN BAŞLIĞI
<input checked="" type="checkbox"/>		Muslu YÜKSEKTEPE	Yazdır	Muslu____YÜKSEKTEPE

1 Teknisyen

SÜZ

Aktif süzgecine göre

☐ Tümü

☐ Evet

☐ Hayır

Adı Soyadı süzgecine göre

☐ Tümü

☐ Muslu YÜKSEKTEPE



Şimdi de kayıt girerken yapabileceğimiz otomatik yetkilendirmelere bakalım

```
class TeknisyenAdmin(admin.ModelAdmin):
    list_display = ('Aktif', 'AdSoyad', 'Yazdir', 'EkAlanTest')
    list_per_page = 20
    exclude = ('KayitTarihi',)

    list_filter = ('Aktif', 'AdSoyad',)
    search_fields = ('AdSoyad',)
    date_hierarchy = 'KayitTarihi'

    def has_add_permission(self, request):
        return False

    def has_
admin.
    has_change_permission(self, request, obj=None)
    has_delete_permission(self, request, obj=None)
    has_module_permission(self, request)
    __hash__(self)
    predefined

forming system checks...

tem check identified no issues (0 silenced).
uary 13, 2016 - 16:11:57
```

Giriş · Servisformu · Teknisyenler

Değiştirmek için Teknisyen seçin

TEKNİSYEN EKLE +

Q | Ara

2016 Ocak 13

Eylem: | Git 0 / 1 nesne seçildi

	AKTİF	ADI SOYADI	YAZDIR	BURASI ALANIN BAŞLIĞI
	<input checked="" type="checkbox"/>	Muslu YÜKSEKTEPE	Yazdır	Muslu____YÜKSEKTEPE

1 Teknisyen

SÜZ

Aktif süzgecine göre

Tümü

Evet

Hayır

Adı Soyadı süzgecine göre

Tümü

Muslu YÜKSEKTEPE

Kayıt Ekleme yetki fonksiyonu

```
def has_add_permission(self, request):
    return False
```

Giriş · Servisformu · Teknisyenler

Değiştirmek için Teknisyen seçin

TEKNİSYEN EKLE +

Q | Ara

2016 Ocak 13

Eylem: | Git 0 / 1 nesne seçildi

	AKTİF	ADI SOYADI	YAZDIR	BURASI ALANIN BAŞLIĞI
	<input checked="" type="checkbox"/>	Muslu YÜKSEKTEPE	Yazdır	Muslu____YÜKSEKTEPE

1 Teknisyen

SÜZ

Aktif süzgecine göre

Tümü

Evet

Hayır

Adı Soyadı süzgecine göre

Tümü

Muslu YÜKSEKTEPE

eklendiğinde süzgeç üstündeki butonun artık gelmediğini göreceksiniz.

Yani artık yeni kayıt ekleme yetkimiz yok. Bu tüm kullanıcılar için geçerlidir. Bazen tek bir kayıt olması ve bu kaydın sadece güncellenmesi gerektiğinde bu yöntemi kullanabilirsiniz. Şu an aklıma gelen bir örnek hakkımızda yazısı olabilir. Müşteriniz artık hakkımızda yazısı ekleyemez sadece düzenleyebilir ve silebilir.

Teknisyen değiştir

GEÇMİŞ

☒ Aktif

Adı Soyadı:

Muslu YÜKSEKTEPE

SİL

Kaydet ve düzenlemeye devam et

KAYDET

```
def has_delete_permission(self, request, obj=None):  
    return False
```

Fonksiyonunu ekleyerek artık bu uygulama için silme yetkisini de iptal etmiş olduk.

Teknisyen değiştir

GEÇMİŞ

☒ Aktif

Adı Soyadı:

Muslu YÜKSEKTEPE

Kaydet ve düzenlemeye devam et

KAYDET

Tüm kullanıcıları değilde bazı kullanıcıları engellemek istersek, grup, üyelik, super kullanıcı yada sadece kaydı giren kişinin bu yetkilerin dışında yada içinde kalması için fonksiyona bir kaç müdahale etmemiz gerekiyor.

Aşağıdaki kodları herhangi bir fonksiyonu ekleyerek istediğimiz gibi yetkilendirme yapabiliriz ama ben `has_add_permission` için anlatacağım.

```

if not request.user.is_superuser:
    ### kullanıcı super inek değilse

    try:
        KayitSayisi = self.model.objects.count()
    ### hiç kayıt olmadığında hata verecektir.
    except:
        KayitSayisi = 0
    ### geçerli uygulamadaki obje sayısı, yani toplam kayıt sayısı

    ## KalanLimit = Uyeler.objects.get(user = request.user).KayitLimiti
    ### fikir olması açısından ekledim. İleri de kullanıcılara kayıt limiti verebiliriz. Grafson.com daki gibi

    KalanLimit = 5
    ### super olmayan kullanıcılar kayıt sayısı en fazla 5 olabilir.

    if KayitSayisi >= KalanLimit:
        return False
    else:
        return True
***return True olduğu sürece fonksiyon çalışacaktır.

```

Teknisyen "Cem Emir YÜKSEKTEPE" başarılı olarak değiştirildi.

Değiştirmek için Teknisyen seçin

Ara

2016 Ocak 13

Eylem: 

-----

Git

 0 / 2 nesne seçildi

<input type="checkbox"/>	AKTİF	ADI SOYADI	YAZDIR	BURASI ALANIN BAŞLIĞI
<input type="checkbox"/>	<div><div></div><div></div></div>	Muslu YÜKSEKTEPE	<a href="#">Yazdır</a>	Muslu____YÜKSEKTEPE
<input type="checkbox"/>	<div><div></div><div></div></div>	Cem Emir YÜKSEKTEPE	<a href="#">Yazdır</a>	Cem____Emir____YÜKSEKTEPE

2 Teknisyenler

SÖZ

Aktif sözgecine göre

Tümü

Evet

Hayır

Adı Soyadı sözgecine göre

Tümü

Cem Emir YÜKSEKTEPE

Muslu YÜKSEKTEPE

Muslu YÜKSEKTEPE – 2016  
[www.muslu.org](http://www.muslu.org) | [www.djantoturkiye.com](http://www.djantoturkiye.com)  
 Başış Hesabı: TR03 0006 2000 7500 0006 6675 10

1 adet Teknisyen bağırlı olarak silindi.

Değıştirmek için Teknisyen seçin

Ara

2016 Ocak 13

Eylem:  Git 0 / 1 nesne seçildi

	AKTİF	ADI SOYADI	YAZDIR	BURASI ALANIN BAŞLIĞI
	<div></div>	Cem Emir YÜKSEKTEPE	Yazdır	Cem____Emir____YÜKSEKTEPE

1 Teknisyen

SÜZ

Aktif süzgecine göre

Tümü

Evet

Hayır

Adı Soyadı süzgecine göre

Tümü

Cem Emir YÜKSEKTEPE

Yönetim sayfasında yetkilendirme ve kısıtlama işlemleri de bu şekilde yapılabilir.

Eğer sonuçların gösterildiğı bu sayfa da stil değışikliği yapmak isterseniz proje klasörünüzdeki **templates/admin/change\_list\_results.html** dosyasını kullanabilirsiniz.

Örnek stil:

```
<style>
  .object-tools a.addlink {
    font-size: 22px;
    background-color: #ff9900;
  }
</style>
```

TEKNİSYEN EKLE+

SÜZ

Aktif süzgecine göre

Tümü

Evet

Hayır

Adı Soyadı süzgecine göre

Tümü

Cem Emir YÜKSEKTEPE

Yine aynı dosyada bulunan sonuç sayısını yazdırabiliriz.

....

...

```
{% if results %}
<div class="results">

<p>{{ results|length }} adet kayıt bulundu!</p>

<table id="result_list">
```

.....

..

.

Değiştirmek için Teknisyen seçin

Q  Ara

2016 Ocak 13

Eylem:  Git 0 / 1 nesne seçildi

1 adet kayıt bulundu!

<input type="checkbox"/>	AKTİF	ADI SOYADI	YAZDIR	BURA!
<input checked="" type="checkbox"/>	✓	Cem Emir YÜKSEKTEPE	Yazdır	Cem_

1 Teknisyen

Şimdi de kullanıcıya göre gösterilecek alanları kısıtlamayı görelim.

list\_display'i liste olarak tanımlayarak remove ile kullanıcıya göre gizle/göster yapabiliriz.

list\_display = ['Aktif', 'AdSoyad', 'Yazdir', 'EkAlanTest']

olarak değiştirelim.

```
def has_add_permission(self, request):
    # if not request.user.is_superuser:
    KayitSayisi = self.model.objects.count()
    KalanLimit = 2

    if KayitSayisi >= KalanLimit:
        return False
    else:
        return True

def get_list_display(self, request):
    g_l = super(TeknisyenAdmin, self).get_list_display(request)

    try:
        if request.user.is_superuser:
            g_l.remove('EkAlanTest')
    except:
        pass
    return g_l

admin.site.register(Teknisyen, TeknisyenAdmin)
```

```
def get_list_display(self, request):
```

```
    g_l = super(TeknisyenAdmin, self).get_list_display(request)
```

```
    ### list_display listesini g_l adında bir listeye aktarıyoruz.
```

```
    Try:
```

```
    ## kullanıcı tekrar sayfayı yüklediğinde silinen alan yeniden silinmeye çalışılacak ve hata oluşacak
```

```
    if not request.user.is_superuser:
```

```
    ### kullanıcı super kullanıcı değilse
```

```
        g_l.remove('EkAlanTest')
```

```
    ### list_display listemizden EkAlanTest alanını gizle
```

```
    except:
```

```
    ### hata oluştuğunda pas geç
```

```
        pass
```

```
    return g_l
```

```
    ### g_l listesi düzenlenmiş olarak yada normal hali ile geri gönderilsin.
```

Değiştirmek için 1 teknisyen seçin

2016 Ocak 13

Eylem:   0 / 1 nesne seçildi

1 adet kayıt bulundu!

<input type="checkbox"/>	AKTİF	ADI SOYADI	YAZDIR
<input type="checkbox"/>	✓	Cem Emir YÜKSEKTEPE	Yazdır

1 Teknisyen

Aktif s

Tümü

Evet

Hayır

Adı So

Tümü

Cem Er

Alınabilecek hatalar:

**'tuple' object has no attribute 'remove'**

tuple olarak tanımlanan list\_display'i listeye çevirmeyi atladınız, yani ['Aktif', 'AdSoyad'.....] gibi olmalı

**list.remove(x): x not in list**

Kullanıcı sayfayı tekrar yüklemeye çalıştığında listeden EkAlanTest alanını tekrar silmeye çalışıyor ama yok. try except kullanmayı unuttunuz.

14 Ocak 2016

Bu fonksiyonlar tüm yetkileri kısıtlayabilirsiniz.

Şimdi de uygulamanın yönetim panelindeki isimlendirmesine bakalım.

Uygulamamızın adı **servisformu** olduğu için uygulama için oluşturulan div de SERVISFORMU olarak isimlendirildi. İstersek burayı değiştirebiliriz.

Site yönetimi

KİMLİK DOĞRULAMA VE YETKİLENDİRME		
Gruplar	+ Ekle	Değiştir
Kullanıcılar	+ Ekle	Değiştir
SERVISFORMU		
Teknisyenler	+ Ekle	Değiştir

Son E

Eylemle

musl

Kullan

Musl

Teknis

Cem

Teknis

Bunun için `__init__.py` ve `apps.py` dosyalarımızı açalım.

***apps.py:***

```
# -*- coding: utf-8 -*-
```

```
from __future__ import unicode_literals
```

```
from django.apps import AppConfig
```

```
class ServisformuConfig(AppConfig):
```

```
    name = 'servisformu'
```

```
### uygulamanın adı
```

```
    verbose_name = u'Servis Formları'
```

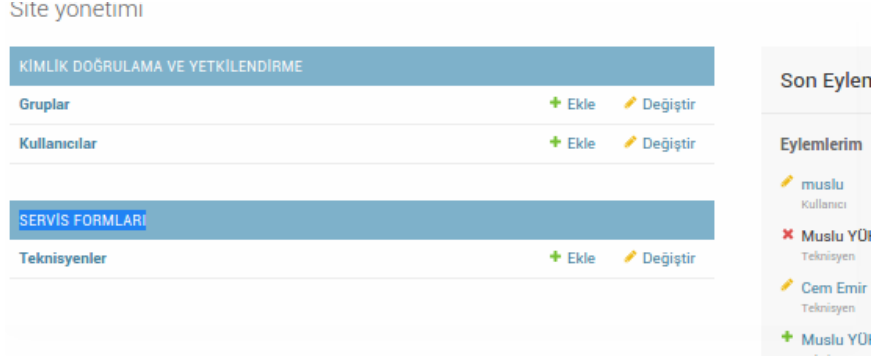
```
### uygulamamızın gösterilecek adı
```

***\_\_init\_\_.py:***

```
default_app_config = 'servisformu.apps.ServisformuConfig'
```

<http://127.0.0.1:8000/admin/>

Uygulama adımız deđiřti.



Projemizin konusuna geri dönersek bir teknik servis i takibi için formlar oluşturacağız. Bunun için önce projenin neleri kapsayacağı ve nelere sorun çözeceđini düşünerek ön bir fizibilete yapalım.

Çıktısı alınacak form için hangi alanlar olması gerektiđini düşünelim.

1. Müřteri bilgileri
2. Ürünler (tablet, laptop, otomobil, telefon vss.)
3. Ürünler ile gelen aksesuarlar (pil, çanta vs..)
4. Teknisyen (Müdahale edecek personel)
5. İşlem durumları (Hazır, yeni geldi, parça bekliyor vs..)

Projemizde önden kurgulamamız gereken hususlarda çok dikkatli olmalıyız. Daha sonradan eklenmesi gereken alan yada uygulamalar, veritabanında bilgi kaybına kadar yol açabilir.

Teknisyenler için bir tablo oluşturmuřtuk. řimdi de müşteriler için bir tablo oluşturalım. Bunu yeni bir uygulama oluşturarak yapabiliriz ancak ben aynı model üzerinden yeni bir sınıfla devam etmenin daha uygun olduđunu düşünüyorum.

Fazla uygulama ile yeni bařlayan arkadaşların kafaları karışmasını istemeyiz.

models.py dosyamızı açıp alt tarafa yazmaya bařlayalım.



models.py:

**import random, string**

```
class Musteriler ( models.Model ) :
    Aktif          =      models.BooleanField(default = 1)
    Kodu           =      models.CharField ( u'Müşteri Kodu', default=''.join(random.choice(string.digits)
for x in range(8)), max_length = 8 )
    Unvan          =      models.CharField ( u'Ticari Ünvan', max_length = 250 )
    Yetkili        =      models.CharField ( u'Yetkili Adı Soyadı', max_length = 250 )
    Telefon        =      models.CharField ( u'Telefon', max_length = 13 )
    KayitTarihi    =      models.DateField ( u"Kayıt Tarihi", default=timezone.now)

    def __unicode__(self):
        return self.Yetkili

    class Meta:
        verbose_name_plural      =      u"Müşteriler"
        verbose_name             =      u"Müşteri"
```

Bu sınıfta farklı olarak kullandığımız şey müşteri koduna ait rastgele bir kod oluşturmak. Yeni bir kayıt açtığımızda default olarak sürekli rastgele 8 haneli sadece rakamlardan oluşan bir kod oluşturulacak.

**default=''.join(random.choice(string.digits) for x in range(8))**

terminalde

**./manage.py makemigrations servisformu && ./manage.py migrate && ./manage.py runserver**

ile değişiklikleri onaylatıp serverı başlatıyoruz.

admin.py dosyasını açarak

```
class MusterilerAdmin(admin.ModelAdmin):
    list_display      =      ( 'Kodu', 'Unvan', 'Yetkili', 'Aktif' )
    list_per_page     =      80
    exclude           =      ( 'KayitTarihi', )
    search_fields      =      ( 'Yetkili', 'Unvan' )
```

**admin.site.register(MusterilerAdmin, Musteriler)**

Alınacak hata:

## TypeError: 'MediaDefiningClass' object is not iterable

Çünkü register ederken önce sınıf, sonra sınıfa ait admin ayarları sınıfı tanımlanmalı. Bu hata ile çok nadir karşılaşacaksınız ama aklıma gelmişken değinmek istedim.

admin.site.register(Musteriler, MusterilerAdmin)  
olarak düzeltelim.

./manage.py makemigrations servisformu && ./manage.py migrate && ./manage.py runserver

ve

<http://127.0.0.1:8000/admin/servisformu/>

The screenshot shows the Django admin interface for 'Servis Formları'. The breadcrumb trail is 'Giriş > Servis Formları'. The main heading is 'Servis Formları yönetimi'. Below this is a table with two rows: 'Müşteriler' and 'Teknisyenler'. Each row has '+ Ekle' and 'Değiştir' links. Below the table is a section for 'Müşteri ekle' with a 'Aktif' checkbox checked. The form fields are: 'Müşteri Kodu' (with value 94506257), 'Ticari Ünvan', 'Yetkili Adı Soyadı', and 'Telefon'.

Müşteri kodu otomatik 8 karakter ve sadece rakamlardan oluşuyor.

Labellarda gördüğümüz gibi tüm alanlar zorunlu. Çünkü charfield oluştururken blank=True kullanmadık.

models.py ye dönelim ve bir test daha yapalım.

Telefon alanımıza blank = True ekleyelim.

Telefon = models.CharField ( u'Telefon', max\_length = 13, blank = True )

### Müşteri ekle

☒ Aktif

Müşteri Kodu: 18572984

Ticari Ünvan:

Yetkili Adı Soyadı:

Telefon:

Artık telefon alanı zorunlu değil.

Bonus: Gerektiğinde bir alanı **null** kabul edilmiyor istersenirse **null = True** yazabilirsiniz.

✓ Müşteri "Muslu YÜKSEKTEPE" başanlı olarak eklendi.

Değiştirmek için Müşteri seçin

MÜŞTERİ EKLE+

Q  Ara

Eylem:  Git 0 / 1 nesne seçildi

1 adet kayıt bulundu!

<input type="checkbox"/>	MÜŞTERİ KODU	TİCARİ ÜNVAN	YETKİLİ ADI SOYADI	AKTİF
<input checked="" type="checkbox"/>	18572984	Yazki Bilişim Hizmetleri	Muslu YÜKSEKTEPE	✓

1 Müşteri

NOT: Kayıt düzenleme linki her zaman için satırın ilk alanındadır. Burada müşteri kodu.

Satırdaki tüm alanlarda yada seçilen alanlar tıklandığında düzenleme linkinin getirilmesi için admin.py dosyamızı açıp list\_display\_links ekleyebiliriz.

```
class MusterilerAdmin(admin.ModelAdmin):
    list_display = ('Kodu', 'Unvan', 'Yetkili', 'Aktif')
    list_per_page = 80
    exclude = ('KayitTarihi',)
    search_fields = ('Yetkili', 'Unvan')
    list_display_links = ('Unvan', 'Yetkili')
```

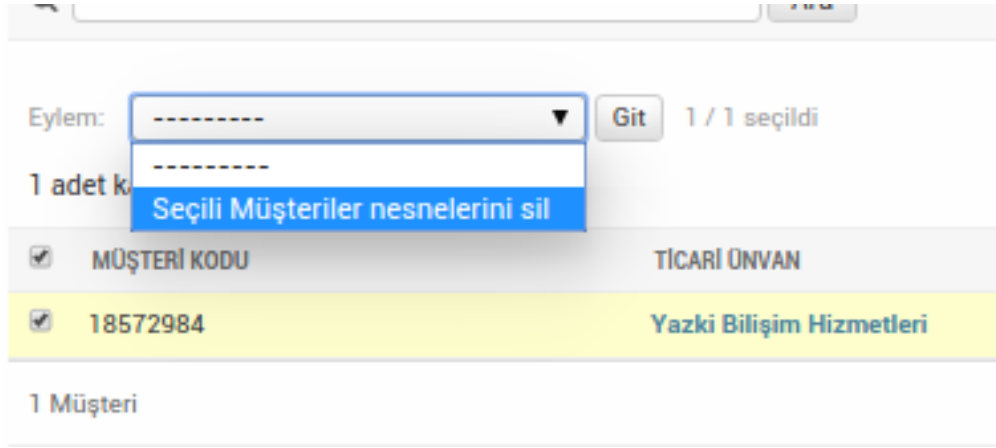
Eylem:   0 / 1 nesne seçildi

1 adet kayıt bulundu!

<input type="checkbox"/> MÜŞTERİ KODU	TİCARİ UNVAN	YETKİLİ ADI SOYADI	AKTİF
<input type="checkbox"/> 18572984	Yazki Bilişim Hizmetleri	Muslu YÜKSEKTEPE	<input checked="" type="checkbox"/>

1 Müşteri

Kayıtlarda toplu işlem yaptırmak için action ları kullanırız. Türkçe karşılığı eylem olarak kullanılmış ve bir açılır kutuda sunulmuş. İstersek bunlara kendimize özel toplu işlem eylemleri ekleyebiliriz.



admin.py dosyamızı açalım ve en üstlerden bir yer beğenip aşağıdaki fonksiyonu ekleyelim.

```
def SecilileriGuncelle(modeladmin, request, queryset):
    for k in queryset:
        k.save()
    return ""
SecilileriGuncelle.short_description = u"Seçilileri Güncelle"
```

```
class MusterilerAdmin(admin.ModelAdmin):
    list_display = ('Kodu', 'Unvan', 'Yetkili', 'Aktif')
    list_per_page = 80
    exclude = ('KayitTarihi',)
    search_fields = ('Yetkili', 'Unvan')
    list_display_links = ('Unvan', 'Yetkili')
    actions = (SecilileriGuncelle,)
    actions_on_bottom = True
    actions_on_top = True
```

```

def SecilileriGuncelle(modeladmin, request, queryset):
    for k in queryset:
        k.save()
    return ""
SecilileriGuncelle.short_description = u"Secilileri Güncelle"

class MusterilerAdmin(admin.ModelAdmin):
    list_display = ('Kodu', 'Unvan', 'Yetkili', 'Aktif')
    list_per_page = 80
    exclude = ('KayitTarihi',)
    search_fields = ('Yetkili', 'Unvan')
    list_display_links = ('Unvan', 'Yetkili')
    actions = (SecilileriGuncelle,)
    actions_on_bottom = True
    actions_on_top = True

```

Bonus: `actions_on_bottom` ve `actions_on_top` ile eylemler açılır kutusunu kayıt gridlerinin altında yada üstünde göster diyebiliriz.

Alınabilecek hatalar:

**NameError: name 'SecilileriGuncelle' is not defined**  
**SecilileriGuncelle** fonksiyonunu altta bıraktınız yada yazmadınız.

Değiştirmek için Müşteri seçin

MÜŞTERİ EKLE+

Q  Ara

Eylem:  Git 0 / 1 nesne seçildi

1 adet k  
 Seçili Müşteriler nesnelerini sil  
 Seçilileri Güncelle

MÜŞ	TİCARİ UNVAN	YETKİLİ ADI SOYADI	AKTİF
<input type="checkbox"/> 18572984	Yazki Bilişim Hizmetleri	Muslu YÜKSEKTEPE	<input checked="" type="checkbox"/>

Eylem:  Git 0 / 1 nesne seçildi

1 Müşteri

Biraz daha detaylı inceleyelim.

`def SecilileriGuncelle(modeladmin, request, queryset):`  
**### SecilileriGuncelle** adında bir fonksiyon oluşturuyoruz

`print queryset`  
**### querysetten neler geldiğine** bakıyoruz

`for k in queryset:`  
**### queryset ile seçilen tüm kayıtların listelerini** alıyoruz.

```
print k
### kayıtları döngüden tek tek alıyoruz
```

```
k.save()
### sınıfımıza ait save modülünü çalıştırıyoruz. Birazdan detaylı değineceğim.
```

```
return ""
### geri dön ama boş dön
```

```
SecilileriGuncelle.short_description = u"Seçilileri Güncelle"
#### fonksiyonun kısa bir açıklaması
```

1 kayıt var

```
January 14, 2016 - 11:47:18
Django version 1.9.1, using settings 'teknikservistakibi.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
[<Musteriler: Muslu YÜKSEKTEPE>]
Muslu YÜKSEKTEPE
[14/Jan/2016 11:47:26] "POST /admin/servisformu/musteriler/ HTTP/1.1" 302 0
[14/Jan/2016 11:47:26] "GET /admin/servisformu/musteriler/ HTTP/1.1" 200 7495
[14/Jan/2016 11:47:26] "GET /admin/jsi18n/ HTTP/1.1" 200 7259
```

2 kayıt var.

```
[14/Jan/2016 11:48:06] "POST /admin/servisformu/musteriler/add/ HTTP/1.1" 302 0
[14/Jan/2016 11:48:06] "GET /admin/servisformu/musteriler/ HTTP/1.1" 200 8143
[14/Jan/2016 11:48:06] "GET /admin/jsi18n/ HTTP/1.1" 200 7259
[<Musteriler: Serkan Emiņç>, <Musteriler: Muslu YÜKSEKTEPE>]
Serkan Emiņç
Muslu YÜKSEKTEPE
[14/Jan/2016 11:48:10] "POST /admin/servisformu/musteriler/ HTTP/1.1" 302 0
[14/Jan/2016 11:48:10] "GET /admin/servisformu/musteriler/ HTTP/1.1" 200 7988
```

Hatırlarsanız \_\_unicode\_\_ fonksiyonuna sadece yetkili alanımızı eklemiştik.

```
def __unicode__(self):
    return self.Yetkili
```

yerine

```
def __unicode__(self):
    # return self.Kodu + " " + self.Yetkili
    # return "%s %s" % (self.Kodu, self.Yetkili)
    return "Müşteri Kodu: %s - Adı: %s" % (self.Kodu, self.Yetkili)
```

Değiştirip çıktıları tekrar kontrol edelim.

```
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
[<Musteriler: 69837932 Serkan Eminç>, <Musteriler: 18572984 Muslu YÜKSEKTEPE>]
69837932 Serkan Eminç
18572984 Muslu YÜKSEKTEPE
[14/Jan/2016 11:52:51] "POST /admin/servisformu/musteriler/ HTTP/1.1" 302 0
```

```
[14/Jan/2016 11:55:53] "GET /static/admin/img/logo-add.svg HTTP/1.1" 304 0
[<Musteriler: Müşteri Kodu: 69837932 - Adı: Serkan Eminç>, <Musteriler: Müşteri Kodu: 18572984 - Adı: Muslu YÜKSEKTEPE>]
Müşteri Kodu: 69837932 - Adı: Serkan Eminç
Müşteri Kodu: 18572984 - Adı: Muslu YÜKSEKTEPE
```

Alınabilecek hatalar:

**coercing to Unicode: need string or buffer, datetime.date found**

dönüşlerin mutlaka string olarak gönderilmesi gerekir. Burada datetimefield kullanılan bir alanı direk göndermeye çalışıyoruz.

Hazır güncelleme yaparken seçilen kayıtlarda değişim yaparak kayıt edilmesine bakalım.

admin.py dosyamızı tekrar açıp

```
def SecilileriGuncelle(modeladmin, request, queryset):
```

```
    for k in queryset:
```

```
        k.Yetkili = k.Yetkili + " ____"
```

```
        k.save()
```

```
    return ""
```

```
SecilileriGuncelle.short_description = u"Seçilileri Güncelle"
```

olarak değiştirip kayıt edelim ve test edelim.

İstediğiniz kayıtları seçip eylemlerden Seçilileri Güncelle 'yi seçip git diyelim.

2 adet kayıt bulundu!

<input type="checkbox"/>	MÜŞTERİ KODU	TİCARİ UNVAN	YETKİLİ ADI SOYADI	AKTİF
<input type="checkbox"/>	69837932	Esefix	Serkan Eminç____	<input checked="" type="checkbox"/>
<input type="checkbox"/>	18572984	Yazki Bilişim Hizmetleri	Muslu YÜKSEKTEPE____	<input checked="" type="checkbox"/>

Eylem:   0 / 2 nesne seçildi

Yetkili adı soyadı alanlarındaki kayıtların sonuna \_\_\_\_ eklendi.

Buraya da özel bir alan ekleyerek müşterilerin telefon numaralarını aratabiliriz. Bunu için tel:.... kullanacağız yani bilgisayarınızda arama yapan bir uygulamaya yada sanal santraliniz varsa bir apiye ihtiyacınız olacak.

admin.py:

```
class MusterilerAdmin(admin.ModelAdmin):
    list_display = ('Kodu', 'Unvan', 'Yetkili', 'Aktif', 'AramaYap')
    ...
    ...
```

models.py:

```
class Musteriler ( models.Model ) :
    ..
    def AramaYap ( self ) :
        if self.Telefon:
            return '<a href="tel:%s" target="_blank">Numarayı Ara</a>' % self.Telefon
        else:
            return 'Telefon No kayıt edilmedi'
    AramaYap.short_description = u'Ara'
    AramaYap.allow_tags = True
```

olarak değiştirelim.

Eylem:   0 / 2 nesne seçildi

2 adet kayıt bulundu!

<input type="checkbox"/>	MÜŞTERİ KODU	TİCARİ UNVAN	YETKİLİ ADI SOYADI	AKTİF	ARA
<input type="checkbox"/>	69837932	Esefix	Serkan Eminç____	✓	Telefon No kayıt edilmedi
<input type="checkbox"/>	18572984	Yazki Bilişim Hizmetleri	Muslu YÜKSEKTEPE____	✓	Numarayı Ara

Eylem:   0 / 2 nesne seçildi

Güncelleme yaparken sql karşılığını da görmek isterseniz.

admin.py

```
def SecilileriGuncelle(modeladmin, request, queryset):
    print queryset.query
    for k in queryset:
        k.Yetkili = k.Yetkili + " ____"
        k.save()
    return ""
```

SecilileriGuncelle.short\_description = u"Seçilileri Güncelle"

olarak fonksiyonu güncelleyip terminal çıktılarına bakabiliriz.

```
terminal
+ Starting development server at http://127.0.0.1:8000/
+ Quit the server with CONTROL-C.
SELECT "servisformu_musteriler"."id", "servisformu_musteriler"."Aktif", "servisformu_musteriler"."Kodu", "servisformu_musteriler"."Unvan", "servisformu_musteriler"."Yetkili", "servisformu_musteriler"."Telefon", "servisformu_musteriler"."KayitTarihi" FROM "servisformu_musteriler" WHERE "servisformu_musteriler"."id" IN (2, 1) ORDER BY "servisformu_musteriler"."id" DESC
[14/Jan/2016 12:26:12] "POST /admin/servisformu/musteriler/ HTTP/1.1" 302 0
```

Şimdide bazı koşulları sağlayan kayıtlarda toplu güncelleme yapalım.

Muslu YÜKSEKTEPE – 2016

[www.muslu.org](http://www.muslu.org) | [www.djangoturkiye.com](http://www.djangoturkiye.com)

Bağış Hesabı: TR03 0006 2000 7500 0006 6675 10



```
admin.py
def SecilileriGuncelle(modeladmin, request, queryset):
    print queryset.query
    for k in queryset:
        if k.Yetkili.startswith("Muslu"):
            k.Yetkili = k.Yetkili + u"__Musluilebaşıyordu"
        k.save()
    return ""
SecilileriGuncelle.short_description = u"Seçilileri Güncelle"
```

Sizde super kullanıcı ise ve telefon numaralarının başına 0 eklenmediyse 0 ekleyerek yeniden kaydet olarak deneme yapabilirsiniz.

Eylem:   0 / 2 nesne seçildi

2 adet kayıt bulundu!

<input type="checkbox"/>	MÜŞTERİ KODU	TİCARİ ÜNVAN	YETKİLİ ADI SOYADI	AKTİF	ARA
<input type="checkbox"/>	69837932	Esefix	Serkan Eminç	✓	Telefon No kayıt edilmedi
<input type="checkbox"/>	18572984	Yazki Bilişim Hizmetleri	Muslu YÜKSEKTEPE__Musluilebaşıyordu	✓	Numarayı Ara

Eylem:   0 / 2 nesne seçildi

Alınabilecek hatalar:

**'ascii' codec can't decode byte 0xc5 in position 13: ordinal not in range(128)**

Türkçe karakter kullandıysanız ama utf-8 kullanmanız gerekiyorsa bu hatayı alırsınız. Örnek olarak "\_\_Musluilebaşıyordu" başına u"...." eklenmediyse bu hata alınır.

Sanırım özel alan ekleyip işlem yaptırma daha iyi anlaşılmıştır.

Bu şekilde bir çok özellik ve güzellik ekleyebilirsiniz.

Teknisyen ve müşterilerin kayıtlarını yapabildiğimize göre artık diğer gereksinimleri yazabiliriz.

Alınan ürünün hangi durumda olduğunu kayıt edip, her işlemde güncelleyip ve sorguda göstermek için durumları hazırlamamız gerekiyor.

Sabit durumlar kullanılacak ise charfield e choices ekleyerek bir listeden seçim yaptırabiliriz ama biz siteyi kullanacak kişilerin kendi durumlarını ekleyebilmeleri için yeni bir tablo oluşturacağız.

Models.py dosyamızı açalım ve Durumlar adında bir sınıf oluşturalım

models.py:

```
..
..
class Durumlar (models.Model):
    Durumu = models.CharField(u'Durum', max_length=30, help_text='Metin alanının altında kayıt girerken yardımcı olabilecek konuları anlatan kısa bir açıklama yazabilirsiniz.')

    def __unicode__(self):
        return self.Durumu

    class Meta:
        verbose_name_plural = u'Durumlar'
        verbose_name = u'Durum'
```

help\_text kullanımına dikkat edin.

admin.py:

```
..
class DurumlarAdmin(admin.ModelAdmin):
    list_display = ('Durumu',)
    list_per_page = 80
```

admin.site.register(Durumlar, DurumlarAdmin)

Terminalde;

./manage.py makemigrations servisformu && ./manage.py migrate && ./manage.py runserver

Giriş > Servis Formları

Servis Formları yönetimi

SERVIS FORMLARI	
Durumlar	<a href="#">+ Ekle</a> <a href="#">Değiştir</a>
Müşteriler	<a href="#">+ Ekle</a> <a href="#">Değiştir</a>
Teknisyenler	<a href="#">+ Ekle</a> <a href="#">Değiştir</a>

Giriş > Servis Formları > Durumlar > Ekle Durum

Durum ekle

Durum:

Metin alanının altında kayıt girerken yardımcı olabilecek konuları anlatan kısa bir açıklama yazabilirsiniz.

Kaydet ve başka birini ekle

Kaydet ve düzenlemeye devam et

KAYDET

Bonus: Yukarıdaki **Giriş > Servis Formları > Durumlar > Ekle Durum** yazısı kafanıza takıldı ve **Durum Ekle** olarak değiştirmek isterseniz ( tüm kayıtlarda geçerli olacak ) **template/admin/change\_form.html** dosyasını açın, ctrl+g 21 ( farklı satır olabilir ) yazarak if add koşulunu bulunup aşağıdaki gibi değiştirin.

{% trans 'Add' %} etiketini yer değiştirin. Translate konusuna sonra gireceğiz.

```
&rsquo; {% if add %} {{ opts.verbose_name }} {% trans 'Add' %} {% else %}{{ original|truncatewords:"18" }}
{% endif %}
```

Servise bırakılan ürünlerin durumlarını da ayarladığımıza göre ürünle beraber bırakılan aksesuarları da kayıt altına alalım.

models.py:

```
...
...
class Aksesuarlar (models.Model):
    Adi = models.CharField(u'Adi', max_length=30, help_text='Ürünle beraber getirilen tüm aksesuarlar. Örn: Batarya, Çanta')

    def __unicode__(self):
        return self.Adi

    class Meta:
        verbose_name_plural = u"Aksesuarlar"
        verbose_name = u"Aksesuar"
```

admin.py:

```
...
..
.
class AksesuarlarAdmin(admin.ModelAdmin):
    list_display = ('Adi',)
    list_per_page = 5
```

admin.site.register(Aksesuarlar, AksesuarlarAdmin)

./manage.py makemigrations servisformu && ./manage.py migrate && ./manage.py runserver

SERVIS FORMLARI		
Aksesuarlar	+ Ekle	Değiştir
Durumlar	+ Ekle	Değiştir
Müşteriler	+ Ekle	Değiştir
Teknisyenler	+ Ekle	Değiştir

Giriş · Servis Formları · Aksesuarlar · Aksesuar Ekle

Aksesuar ekle

Adı:

Çanta

Ürüne beraber getirilen tüm aksesuarlar. Örn: Batarya, Çanta

Kaydet ve başka birini ekle

Kaydet ve düzenlemeye devam et

KAYDET

Aksesurları da eklediğimize göre artık servis formumuzu oluşturabiliriz. Her forma bir ürün eklenecekse önce ürün sınıfını hazırlamamız gerekiyor ama bir form da getirilen tüm ürünlerin kaydı tutulması istenirse önce formu hazırlamalıyız ki inline ile ürünleri sınırsız kayıt ettirebilelim. Kafanız karışmasın hemen yazmaya başlıyoruz.

models.py:

...

.

```
class ServisForm ( models.Model ) :
```

```
    Musteri      = models.ForeignKey ( Musteriler )
```

```
    TeslimEden   = models.CharField ( u'Teslim Eden', max_length = 130 )
```

```
    TeslimAlan   = models.ForeignKey ( Teknisyen, default=int(Teknisyen.objects.get(id=1).id) )
```

```
    FormNo       = models.CharField ( u'Form No', default=''.join(random.choice(string.digits) for x in range(8)), max_length = 8 )
```

```
    KayitTarihi  = models.DateTimeField ( u"Kayıt Tarihi", default=timezone.now)
```

```
    def __unicode__(self):
```

```
        return self.FormNo
```

```
    class Meta:
```

```
        verbose_name_plural      = u"Formlar"
```

```
        verbose_name             = u"Servis Formu"
```

```
    def Yazdir ( self ) :
```

```
        return '<a href="/yazdir/%s" target="_blank">Yazdır</a>' % self.id
```

```
    Yazdir.short_description      = u'Yazdır'
```

```
    Yazdir.allow_tags            = True
```

Not:

ForeignKey ile başka bir tabloya indexleme yaptırıyoruz. NoSQL kullanacağımız zaman bunu kullanamayız. TeslimAlan alanına default olarak teknisyen tablosundaki ilk kaydı getirmeceğiz.

Alınabilecek hatalar:

**This query requires pytz, but it isn't installed.**

Çözümü: terminalde **sudo pip install pytz**

admin.py:

```
class ServisFormAdmin(admin.ModelAdmin):
    list_display = ('FormNo', 'Musteri', 'TeslimEden', 'TeslimAlan', 'KayitTarihi', 'Yazdir')
    list_per_page = 50
    ordering = ('-KayitTarihi',)
    date_hierarchy = 'KayitTarihi'
    search_fields = ('FormNo', 'Musteri__Yetkili', 'Musteri__Telefon')
    exclude = ('KayitTarihi',)
```

admin.site.register(ServisForm, ServisFormAdmin)

search\_fields tanımlamasına dikkat ettiyseniz “\_\_” kullandık. Bu Musteri tablosundaki Yetkili ve Telefon alanlarında da ara demek.

**./manage.py makemigrations servisformu && ./manage.py migrate && ./manage.py runserver**

**Yeni kayıt oluşturmayın. Ürünleri ekledikten sonra oluşturacağız.**

Hatırlarsanız müşterilere ve teslim alan personel için aktif kısıtlaması eklemiştik. Bu kısıtı kullanmak için admin.py dosyamızı açalım.

```
class ServisFormAdmin(admin.ModelAdmin):
    list_display = ('FormNo', 'Musteri', 'TeslimEden', 'TeslimAlan', 'KayitTarihi', 'Yazdir')
    list_per_page = 50
    ordering = ('-KayitTarihi',)
    date_hierarchy = 'KayitTarihi'
    search_fields = ('FormNo', 'Musteri__Yetkili', 'Musteri__Telefon')
    exclude = ('KayitTarihi',)

    def formfield_for_foreignkey(self, db_field, request, **kwargs):
        if db_field.name == 'TeslimAlan':
            ### field ( alan ) adı TeslimAlan ise

            kwargs["queryset"] = Teknisyen.objects.filter(Aktif=True)
            ### Teknisyenlerin sadece aktif olanları filtreleyerek getir ve queryset e ekle
            ### kwargs konusuna şimdilik girmiyoruz

        if db_field.name == 'Musteri':
            kwargs["queryset"] = Musteriler.objects.filter(Aktif=True)
            return super(ServisFormAdmin, self).formfield_for_foreignkey(db_field, request, **kwargs)
```

olarak değiştirelim.

formfield\_for\_foreignkey fonksiyonu, sınıfımıza ait foreignkey fieldlarını kullanmamızı sağlar. Yani bu fieldlara kısıtlama yada yetkilendirme koyabiliriz.

Müşteri kısıtlamasını test etmek için

Giriş · Servis Formları · Müşteriler · Müşteri Kodu: 69837932 - Adı: Serkan Eminç

Müşteri değiştir

☒ Aktif

Müşteri Kodu: 69837932

Ticari Ünvan: Esefix

Yetkili Adı Soyadı: Serkan Eminç



Telefon:

Sil



bir müşterinin aktif durumunu pasife çekip kaydediyoruz.

Giriş · Servis Formları · Formlar · Servis Formu Ekle

Servis Formu ekle

Musteri:   

Teslim Eden:

Teslim Alan:   

Form No:

Foreignkey kısıtlamasını da böylelikle görmüş olduk.

Servis formu kaydını yapabiliyoruz ama teslim edilen ürünleri eklememiz gerekiyor. Biraz önce bahsettiğim gibi bir forma ait çok ürün ekleyebiliriz. Bu vesile ile inline tabloları da görmüş olacağız.

models.py dosyamızı açalım

Dikkat edin ServisForm dan sonra eklemeniz gerekiyor.

...  
...  
..

```
class Urunler ( models.Model ) :  
    ServisFormu      =      models.ForeignKey ( ServisForm )  
    Cins             =      models.CharField ( u'Cinsi', max_length = 30 )  
    Marka            =      models.CharField ( u'Marka', max_length = 50 )  
    Model            =      models.CharField ( u'Model', max_length = 50 )  
    SeriNo           =      models.CharField ( u'Seri No', max_length = 250 )  
    GarantiBitis     =      models.DateField ( u"Garanti Bitiş", default=timezone.now )  
    Sikayet          =      models.TextField ( u'Şikayet' )  
    Aksesuar         =      models.ManyToManyField ( Aksesuarlar, blank=True )  
    Durum            =      models.ForeignKey ( Durumlar )  
    Not              =      models.TextField ( u'Yapılan İşlemler', blank=True )
```

```
def __unicode__(self):  
    return "%s %s %s" % (self.Cins, self.Marka, self.Model)
```

```
class Meta:  
    verbose_name_plural      =      u"Ürünler"  
    verbose_name             =      u"Ürün"
```

Not:

ManyToManyField kullandık, yani bir indexleme yaparak başka bir tablodaki kayıtlardan çoklu seçim yapabiliriz.

TextField kullandık, Uzun açıklamalar girebiliriz.

admin.py dosyamızı açalım

admin.py:

..

..

.

```
class UrunlerInline(admin.StackedInline):
```

```
    model = Urunler
```

```
###hangi model
```

```
    extra = 0
```

```
### yeni eklenmek istendiğinde kaç tane ürün ekleme yapsın. İlk kayıta hazır olarak ürün ekleme sayfası getirmeyecek biz ekle deyince 1 tane ürün ekleme sayfası açacak.
```

```
    max_num = 5
```

```
### en fazla kayıt ürün eklenebilir.
```

```
class ServisFormAdmin(admin.ModelAdmin):
```

```
    inlines = [ UrunlerInline, ]
```

```
    list_display = ( 'FormNo', 'Musteri', 'TeslimEden', 'TeslimAlan', 'KayitTarihi', 'Yazdir' )
```

```
    list_per_page = 50
```

```
    ordering = ( '-KayitTarihi', )
```

```
    date_hierarchy = 'KayitTarihi'
```

```
    search_fields = ( 'FormNo', 'Musteri__Yetkili', 'Musteri__Telefon' )
```

```
    exclude = ( 'KayitTarihi', )
```

```
def formfield_for_foreignkey(self, db_field, request, **kwargs):
```

```
    if db_field.name == 'TeslimAlan':
```

```
        kwargs["queryset"] = Teknisyen.objects.filter(Aktif=True)
```

```
    if db_field.name == 'Musteri':
```

```
        kwargs["queryset"] = Musteriler.objects.filter(Aktif=True)
```

```
    return super(ServisFormAdmin, self).formfield_for_foreignkey(db_field, request, **kwargs)
```



Not:

UrunlerInline sınıfını register etmiyoruz.

**./manage.py makemigrations servisformu && ./manage.py migrate && ./manage.py runserver**

[Giriş](#) · [Servis Formları](#) · [Formlar](#) · [Servis Formu Ekle](#)

Servis Formu ekle

Musteri:

-----

+

Teslim Eden:

Müşteri Kodu: 18572984 - Adı: Muslu YÜKSEKTEPE\_\_Musluilebaşlıyordu

Teslim Alan:

Cem Emir YÜKSEKTEPE

+

Form No:

87814917

ÜRÜNLER

[+ Başka bir Ürün ekle](#)

Kaydet ve başka birini ekle

Kaydet ve düzenlemeye devam et

KAYDET

extra=0 dediğimiz için ürün sayfası açık gelmedi, **Başka bir Ürün ekle** linki ile ürün ekleme sayfamız gelecek.

## Not:

Tasarımsal olarak daha uygun olduğunu StackedInline kullandık.

TeslimAlan:

Cem Emir YÜKSEKTEPE

Form No:

87814917

ÜRÜNLER

Ürün: #1

Cinsi:

Marka:

Model:

Seri No:

Garanti Bitiş:

2016-01-14

Bugün

Şikayet:

Aksesuar:

Çanta

Adaptör

Batarya

Birden fazla seçmek için "Control (Ctrl)" veya Mac'deki "Command" tuşuna basılı tutun.

TabularInline kullandaysık yana doğru uzayacaktı.

CİNSİ	MARKA	MODEL	SERİ NO	GARANTİ BİTİŞ	
				2016-01-14	
				Bugün	

Başka bir Ürün ekle

Kaydet ve başka birini ekle

Kaydet ve düzenlemeye devam et

KAYDET

Servis formumuza yeni bir kayıt girdiğimizde aşağıdaki gibi gözükecek.

✓ Servis Formu "66024779" başarılı olarak eklendi.

Değiştirmek için Servis Formu seçin

SERVİS FORMU EKLE+

2016 Ocak 14

Eylem:   0 / 1 nesne seçildi

1 adet kayıt bulundu!

FORM NO	MÜŞERİ	TESLİM EDEN	TESLİMALAN	KAYIT TARİHİ	YAZDIR
66024779	Müşteri Kodu: 11278334 - Adı: Muslu YÜKSEKTEPE	Ceylan YÜKSEKTEPE	İsmail İSİRGAN	Oca. 14, 2016, 3:27 ö.s.	<a href="#">Yazdır</a>

1 Servis Formu

Bonus: Veritabanı işlemleri için ben RazorSQL kullanıyorum. Bilinen tüm veritabanlarını destekliyor ve kullanımı çok basit.

<http://www.razorsql.com>

Oluşturduğumuz servis formunun çıktısını almak için yazdır adında bir fonksiyon oluşturup grid e eklemiştik. Şimdi de bu kaydın html üzerinde gösterimine değinelim.

views.py dosyamızı açıp yazdıracağımız kaydın id si ile eşleştirme yaparak elde edilen değerleri html dosyasına göndereceğiz.

views.py:

```
def sayfaiyazdir(request, idsi):
```

```
### request şart. Olmazsa olmaz. Bu fonksiyona en azından bir id değeri gerekiyor.
```

```
formdurumu = ServisForm.objects.get(id = idsi)
```

```
## bu id ye ait form bilgileri
```

```
formbilgileri = Urunler.objects.filter(ServisFormu__id = idsi)
```

```
### buid ye ait ürünlerin listesi
```

```
#### ServisFormu__id ile gelen id yi indexlediğimiz servisformu tablosundaki id ile eşleştiriyoruz.
```

```
### örnek olması için bu şekilde yazdım.
```

```
return render(request, 'yazdir.html', {'formbilgileri': formbilgileri, 'formdurumu':formdurumu})
```

```
### bulduğumuz bilgileri yazdir.html dosyasına gödneriyoruz. Tabi ki templates klasörümüzde
```

urls.py dosyamızı açalım ve kodlarımızı yazalım.

```
from servismformu.views import sayfaiyazdir
```

```
urlpatterns = [  
    url(r'^yazdir/([\w-]+)/$', sayfaiyazdir, name='sayfaiyazdir'),  
    url(r'^admin/', admin.site.urls),  
]
```

/yazdir/\*\*\*/ diye bir link geldiğinde servismformu uygulama klasörümüzdeki views.py dosyasında ki sayfaiyazdir fonksiyonunu çalıştıracaktır.

Not:

([\w-]+) herşeyi yazabiliriz. (+id...) kullanabilirdik ama bu daha çok işinize yarayacak, aklınızda bulunsun.

Templates klasörümüzde yazdir.html adında bir dosya oluşturalım.

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <!--[if IE]><meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1"><![endif]-->  
    <meta charset="utf-8">  
    <title>İzmir Teknik Servis, Servis Formu kontrol, durum öğrenme | izmirteknikservis.tk</title>  
    <meta name="description" content="İzmir Teknik Servis, Servis Formu kontrol, durum öğrenme"/>  
    <link rel="canonical" href="http://www.izmirteknikservis.tk"/>  
    <link href="http://fonts.googleapis.com/css?family=Ubuntu+Mono" rel="stylesheet" type="text/css">  
  
    <link rel="shortcut icon" href="/media/favicon.ico">  
    <meta name="viewport" content="width=device-width, initial-scale=1">  
    <link rel="stylesheet" href="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">  
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>  
    <script src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js"></script>  
    <style>  
        .borderless tbody tr td, .borderless tbody tr th, .borderless thead tr th {  
            border: none;  
        }  
    </style>  
</head>  
  
<body onload="window.print()">  
  
<div class="container">  
  
    <h1>{{ formdurumu.FormNo }}</h1>
```

<p>{{ formdurumu.Musteri.Unvan|title }} {{ formdurumu.Musteri.Yetkili|title }} -  
{{ formdurumu.KayitTarihi }}</p>

```
<table class="table table-condensed">
  <thead>
    <tr>
      <th>Ürün</th>
      <th>Seri No</th>
      <th>Şikayet</th>
      <th>Aksesuar</th>
      <th>Not</th>
    </tr>
  </thead>
  <tbody>
    {% for k in formbilgileri %}
      <tr>
        <td>{{ k.Cins|upper }} | {{ k.Marka|upper }} | {{ k.Model|upper }}</td>
        <td>{{ k.SeriNo|upper }}</td>
        <td>{{ k.Sikayet|title }}</td>
        <td>{% for kk in k.Aksesuar.all %}{{ kk|title }} {% endfor %}</td>
        <td>{{ k.Not }}</td>
      </tr>
    {% endfor %}
  </tbody>
```

</table>

</div>

```
<div class="container">
  <table class="table table-condensed borderless">
    <tr>
      <th>Servise Teslim Eden</th>
      <th>Teknisyen</th>
      <th>Ürünü Teslim Alan</th>
    </tr>
    <tr>
      <th>{{ formdurumu.TeslimEden|title }}</th>
      <th>{{ formdurumu.TeslimAlan|title }}</th>
      <th></th>
    </tr>
  </table>
</div>
```

```
<div class="container">
  <table border="0" cellspacing="" width="100%">
```

```
<tr><td><i style="font-size:12px;">Bakım ve onarım süresi 3 (üç) aydır.</i></td></tr>
<tr><td><i style="font-size:12px;">Servisimizde asla kaçak yazılım kullanılmamakta ve destek
verilmemektedir.</i></td></tr>
<tr><td><i style="font-size:12px;">Tamire gelen cihazların harddisk, yazılım ve yedeklerinden tarafımız
sorumlu değildir.</i></td></tr>
<tr><td><i style="font-size:12px;">Servise gelen ürünlerde önceden bildirilmeyen sorunlardan tarafımız
sorumlu değildir.</i></td></tr>
<tr><td><i style="font-size:12px;">Servise teslim edilen ürünlerin emanet süresi 30 (otuz) gün olup,
sonrasında firmamızın sorumluluğu yoktur.</i></td></tr>
<tr><td><i style="font-size:12px;">Servise gelen cihazların chip değişim ve kalıplanmasında oluşabilecek
sorunlarda tarafımız sorumlu değildir.</i></td></tr>
```

```
</table>
</div>
```

```
</body>
</html>
```

Şimdi servis formuna kayıt girerek yazdır linkini tıklayabiliriz.

Örn:

<http://127.0.0.1:8000/yazdir/1/>

Aşağıda ilk kaydımıza ait çıktıyı görebiliriz.

Bonus:

Foreign ve ManyToMany gibi indexleme durumlarında gelen elementlerin yanında düzenlemek ve yeni kayıt eklemek için simgeler yardımcı olacaktır.

Herhangi bir kaydı seçtiğinizde düzenleme aktif olur. Ek pencerede yeni kayıt yada düzenleme işlemi yapabilirsiniz.

Musteri:

Müşteri Kodu: 11278334 - Adı: Muslu YÜKSEKTEPE ▼

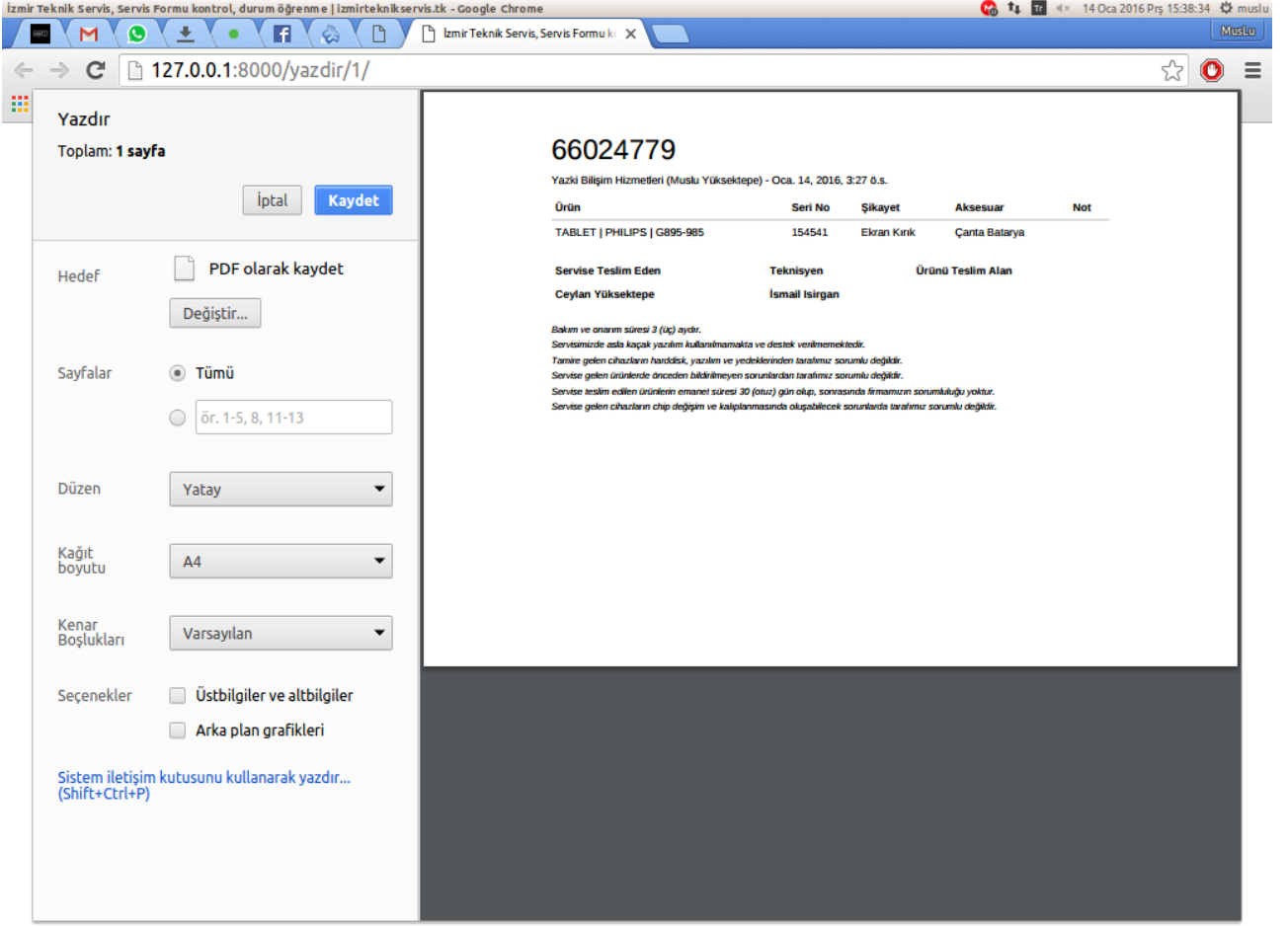


Teslim Eden:

Muslu YÜKSEKTEPE – 2016

[www.muslu.org](http://www.muslu.org) | [www.djangoturkiye.com](http://www.djangoturkiye.com)

Bağış Hesabı: TR03 0006 2000 7500 0006 6675 10



Aşağı yukarı bir serviste olması gereken tüm eksiklikleri giderin bir web sitesi hazırlamış olduk. Kayıt girerek eksikleri görebilir test edebilirsiniz.

Ve geri bildirim olarak dönerseniz sevinirim.

Yarın daha fazla admin sayfası düzenleme ve models.py da save modüllerine bakacağız ve hata alarak çalışmalara devam edeceğiz.

Muslu YÜKSEKTEPE  
14.01.2016