**\_\_init\_\_.py:** Projenin bir Python projesi olarak algılanmasını sağlar. Boş bir dosyadır ve biz de içerisine herhangi bir şey yazmayacağız.  
**settings.py:** Django projemiz ile ilgili yapılandırmalar bulunur.  
**urls.py:** Projedeki adresler bulunur. Boş olarak gelir. Sitemizdeki adresleri buraya yazacağız.  
**wsgi.py** Bu dosya Django geliştirme sunucusu için WSGI(Web Server Gateway Interface,Web Sunucusu Geçit Arayüzü) uygulamasını içerir.  
**manage.py:** Django projemiz ile etkileşime geçmek için oluşturulmuş bir dosyadır. Bu dosyayı düzenlemeyiz. Daha fazla bilgiye [buradan](https://docs.djangoproject.com/en/1.6/ref/django-admin/) ulaşabilirsiniz.  
Artık oluşturduğumuz projeyi sunucu da başlatabiliriz. Projemizin bulunduğu dizindeki manage.py dosyasını çalıştırıyoruz.

Django'da durum bu işlemlerden biraz daha farklı.Herhangi bir sayfanın görüntülenebilmesi için görünümü(views) hazırlamamız gerekiyor. Hazırlamış olduğumuz bu görünüme ulaşabilmek için adres(URL)e bağlamamız gerekli. Şimdi gelin ilk görünümümüzü hazırlayalım.

İlk görünümümüzü oluşturduk. Bu oluşturduğumuz fonksiyon tarayıcıya "Merhaba Django" yazısını gönderecek.

Tarayıcıya gönderme işlemini HTTPResponse fonksiyonu yapar.

Oluşturduğumuz bu görünüme URL yapısı ile ulaşacağız.

Web tarayıcısına bir adres girdiğimiz zaman hangi fonksiyonun çağrılacağına projeyi oluştururken oluşan urls.py dosyası içerisinde belirteceğiz. url.py varsayılan olarak şu şekildedir.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | from django.conf.urls import patterns, include, url    from django.contrib import admin  admin.autodiscover()    urlpatterns = patterns('',      # Examples:      # url(r'^$', 'ilkProjem.views.home', name='home'),      # url(r'^blog/', include('blog.urls')),        url(r'^admin/', include(admin.site.urls)),  ) |

Gördüğünüz gibi varsayılan bir adres tanımlaması yapılmıştır. Burada adres tanımlaması için kullanılan url fonksiyonu iki parametre alır. İlk parametre tarayıcıdan istenen adresi ikincisi ise bu adres çağrıldığında işletilecek olan fonksiyondur. Eğer tarayıcınızdan http://localhost:8000/admin/ yazarsanız şu şekilde bir görünüm ile karşılaşırsınız.

**Dinamik Adresler**  
Dinamik adres tanımlamaları yaparken Düzenli İfadeler(regular expressions)den yararlanacağız. Eğer düzenli ifadeler hakkında bir fikriniz yoksa [Düzenli İfadeler](http://www.pythondersleri.com/p/python-baslangc.html#re) yazılarına bakabilirsiniz.  
Elimizde değişken adresler var ve bizim bu adreslere göre görünümlerimiz varsa bu şekilde urller oluşturmamız gerekiyor. Örneğin elimizde şarkıların bulunduğu bir yapı olsun. Şarkıların kendilerine ait kodları olsun.

[?](https://www.pythondersleri.com/2014/02/url-yaps-ve-gorunumler.html)

|  |  |
| --- | --- |
| 1  2  3  4 | http://localhost:8000/sarkilar/S01  http://localhost:8000/sarkilar/S02  http://localhost:8000/sarkilar/S03  http://localhost:8000/sarkilar/S04 |

Eğer biz bu adreslerin hepsini urls.py dosyasına eklemeye kalkarsak işimiz epey zorlaşır. Burada sarkilar kısmı hep sabit fakat şarkı sürekli değişir. Tabi sisteme şarkı eklenip çıkartıldığında bizim de urls.py dosyamızı tekrardan düzenlememiz gerekir. İşte bu tarz zahmetlerden kurtulmak için Düzenli İfadeleri kullanacağız. Şimdi yukarıdaki adres yapımızı oluşturalım.

[?](https://www.pythondersleri.com/2014/02/url-yaps-ve-gorunumler.html)

|  |  |
| --- | --- |
| 1 | url(r'^sarkilar/(S[0-9]+)/',views.sarkilar\_icerik), |

İleride kendi hata sayfamızı oluşturup kullanıcıya onu göstermeyi öğreneceğiz.  
Adresimizi bu şekilde istediğimiz kadar uzatabiliriz.

[?](https://www.pythondersleri.com/2014/02/url-yaps-ve-gorunumler.html)

|  |  |
| --- | --- |
| 1  2 | http://localhost:8000/kulupler/galatasaray/drogba  http://localhost:8000/kulupler/fenerbahce/emenike |

Bu adresleri urls.py dosyamızda şu şekilde tanımlayabiliriz.

[?](https://www.pythondersleri.com/2014/02/url-yaps-ve-gorunumler.html)

|  |  |
| --- | --- |
| 1 | url(r'kulupler/(.+)/(.+)',views.kulupler\_icerik) |

Bu örnekte gördüğümüz gibi adresi istediğimiz kadar uzatabiliyoruz. Burada her / sonrası oluşturduğumuz görünüm fonksiyonuna parametre olarak gönderilir. Yani burada oluşturduğumuz kulupler\_icerik fonksiyonumuzun yapısı şu şekilde olacaktır.

[?](https://www.pythondersleri.com/2014/02/url-yaps-ve-gorunumler.html)

|  |  |
| --- | --- |
| 1 | def kulupler\_icerik(request,kulup,oyuncu) |

Şu şekilde oluşturduğumuz tüm adresler kulup\_icerik fonksiyonuna yönlendirilir.

[?](https://www.pythondersleri.com/2014/02/url-yaps-ve-gorunumler.html)

|  |  |
| --- | --- |
| 1  2  3  4 | http://localhost:8000/kulupler/galatasaray  http://localhost:8000/kulupler/galatasaray/drogba  http://localhost:8000/kulupler/galatasaray/drogba/forvet/N12  http://localhost:8000/kulupler/galatasaray/drogba/forvet |

Sablonlar

Şablonlar sayfa tasarımını ve kod kısımlarını birbirinde ayırmamızı sağlarlar. Görünümler konusunda görümün fonksiyonları içerisinde doğrudan HTML taglarını kullanmıştık.

[?](https://www.pythondersleri.com/2014/02/sablon-nedir-ve-nasl-kullanlr.html)

|  |  |
| --- | --- |
| 1  2 | def ana\_sayfa(request):      return HttpResponse("<h1>Ana Sayfa</h1><br> Ana sayfa içerikleri") |

Django kullananlar için bu problemler Şablonlar ile ortadan kaldırılmıştır.  
Elimizde bulunan bir html dosyasını Şablonlar ile düzenleyelim.