Merhaba Arkadaşlar,

Bu videomuzda django da kullanıcı yönetimi hakkında bilgiler vereceğiz. Django built in olarak kimlik doğrulama ve yetkilendirme barındırmaktadır.

Kullanabilmek için settings.py de INSTALLED\_APPS kısmında django.contrib.auth ve django.contrib.contenttypes, MIDDLEWARE kısmında SessionMiddleware ve AuthenticationMiddleware bulunmalıdır.

Ana route dosyamıza built-in olarak gelen auth url lerini ekleyebiliriz.

path('accounts/'**,** include('django.contrib.auth.urls'))**,**

Ekledğimizde django otomatik olarak view metodlarını ve karşılık gelen URL adresilerini oluşturur.

urlpatterns = [  
 path('login/'**,** views.LoginView.as\_view()**,** name='login')**,** path('logout/'**,** views.LogoutView.as\_view()**,** name='logout')**,** path('password\_change/'**,** views.PasswordChangeView.as\_view()**,** name='password\_change')**,** path('password\_change/done/'**,** views.PasswordChangeDoneView.as\_view()**,** name='password\_change\_done')**,** path('password\_reset/'**,** views.PasswordResetView.as\_view()**,** name='password\_reset')**,** path('password\_reset/done/'**,** views.PasswordResetDoneView.as\_view()**,** name='password\_reset\_done')**,** path('reset/<uidb64>/<token>/'**,** views.PasswordResetConfirmView.as\_view()**,** name='password\_reset\_confirm')**,** path('reset/done/'**,** views.PasswordResetCompleteView.as\_view()**,** name='password\_reset\_complete')**,**]

Sadece login ve logout kullanmak istiyorsak direk route dosyamızda da tanımlayabiliriz. Varsıyalan olarak her view için template atanmıştır. Templateler hazır olmayıp bunları bizim hazırlamamız gerekmektedir. İstersek template yollarını template\_name argümanı ile değiştirebiliriz.

from django.contrib.auth import auth\_views  
  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 path('learning/', include('learning.urls')),  
 path('accounts/', include('django.contrib.auth.urls')),  
 path('\_\_debug\_\_/', include(debug\_toolbar.urls)),  
  
 path('account/login/', auth\_views.LoginView.as\_view(template\_name='project/auth/login.html'), name='login'),  
 path('account/logout/', auth\_views.LogoutView.as\_view(), name='logout'),  
]

Varsayılan tema yolları:

*Login : 'registration/login.html'  
logout : 'registration/logged\_out.html'*

*Password\_change :'registration/password\_change\_form.html'  
password\_change\_done :'registration/password\_change\_done.html'  
password\_reset : 'registration/password\_reset\_form.html'  
email\_template : 'registration/password\_reset\_email.html'  
subject\_template: 'registration/password\_reset\_subject.txt'  
reset\_done : 'registration/password\_reset\_done.html'  
password\_reset\_confirm: 'registration/password\_reset\_confirm.html'  
password\_reset\_complete: 'registration/password\_reset\_complete.html'*

Bizim djangonun entegre olan auth viewlerini kullanma zorunluluğumuz yok. Django istersek kendi viewlerimizide yazmamıza izin vermektedir. Biz ilk etapta djangonun entegre olarak sunduğu imkanlar ile devam edelim. üye giriş ve çıkışı, üye kaydı, şifre değiştirme ve şifre hatırlatmasını sağlayacak bir kullanıcı yönetimi oluşturalım.

Kullanıcı Girişi ile başlayabiliriz. Django auth template dosyalarının registration içerisinde yer aldğını söylemiştik. templates klasörü içerisine registration klasörünü oluşturalım ve içerisine üye girişi için varsayılan tema adı olan login.html dosyasını ekleyelim. Django bize giriş formunu hazır olarak sunmaktadır. Daha önce formlarla çalışmak videomuzdan hatırladığımız gibi formları template içerisine getirmek için {{ form.as\_p }} veya table veya ul kullanıyorduk. Csrf token eklemeyi unutmayalım. Evet giriş formumuz bu kadar oluşturmuş olduk ☺

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<body>  
 <h2>Üye Girişi</h2>  
 <form method="post">  
 {% csrf\_token %}  
 {{ form.as\_p }}  
 <button type="submit">Giriş</button>  
 </form>  
</body>  
</html>

Serverımızı çalıştıralım ve kontrol edelim.   
Gördüğünüz gibi giriş sayfamız geldi. Üye girişi yapmayı deneyelim. (önce yanlış giriş yapıp hatayı daha sorna doğru giriş yapıp yönlendiği sayfayı gösterelim)

Evet gördüğünüz giriş yaptığında bizi profile sayfasına yönlendirdi. Bu sayfayı settings.py den LOGIN\_REDIRECT\_URL ayarı ile değiştirebiliriz. Ürün listeleme sayfasına yönlendirelim.

LOGIN\_REDIRECT\_URL = '/learning/product/'

Tekrar giriş yaptığımızda yönlendiğini görüyoruz. Evet şimdi bu login sayfamızı daha önce oluşturmuş olduğumuz şablon içerisine alalım.

{% extends 'learning/layout/base.html' %}  
  
{% block title %}  
Giriş  
{% endblock %}  
  
{% block top %}  
 <div class="jumbotron">  
 <div class="container">  
  
 <h1 class="display-3">Üye Girişi!</h1>  
 <p>Hoşgeldiniz. Aşağıdaki formu kullanarak üye girişi yapabilirsiniz. </p>  
 </div>  
 </div>  
{% endblock %}  
  
{% block content %}  
 <form method="post">  
 {% csrf\_token %}  
 {{ form.as\_p }}  
 <button type="submit">Giriş</button>  
 </form>  
{% endblock %}

Eğer giriş yapıldı ise giriş formunun tekrardan görüntülenmesine gerek yok. Bunun için redirect yapabiliriz veya üye girişi yapışmıştır çıkış için tıklayın yazdırabiliriz.

Django kullanıcı bilgilerini user nesnesinde tutar templatelerden istediğimiz zaten user nesnesinin özelliklerine ulaşabiliriz. Kullanıcı girişi yapıldığının tespiti için user.is\_authenticated kullanılır. Basit bir if bloğu oluşturalım ve is\_authenticated true ise username yazdıralım değilse formu getirelim. User nesnesinin username özelliği ile kullanıcı adını alabiliriz. Sayfamızı yenilendiğimizde merhaba admin geldiğini görüyoruz.

{% block content %}  
 {% if user.is\_authenticated %}  
 Merhaba {{ user.username }}  
 {% else %}  
 <form method="post">  
 {% csrf\_token %}  
 {{ form.as\_p }}  
 <button type="submit">Giriş</button>  
 </form>  
 {% endif %}  
  
{% endblock %}

Üye çıkışı için logout linki oluşturalım. Tıkla nın üzerine geldiğinde logout linkini görebiliyoruz.

Merhaba {{ user.username }}. Çıkış yapmak için <a href="{% url 'logout' %}">tıkla.</a>

Yukarıya baktığımızda üye girişi yapabilirsiniz yazıyor. O bölümde üye girişi yapıldıysa değiştirelim.

{% block top %}  
 <div class="jumbotron">  
 <div class="container">  
  
 <h1 class="display-3">Üye Girişi!</h1>  
 {% if user.is\_authenticated %}  
 Merhaba {{ user.username }}. Üye girişi yapıldığı için tekrar giriş yapamazsan.  
 {% else %}  
 <p>Hoşgeldiniz. Aşağıdaki formu kullanarak üye girişi yapabilirsiniz. </p>  
 {% endif %}  
 </div>  
 </div>  
{% endblock %}

Logout için tıkladığımızda yönetim paneli içerisindeki logout sayfasına yönleniyor. Logged\_out.html ile kendi içeriğimizi gösterebiliriz.

Kendi istediğimiz logout sayfasına yönlendirmek için settings.py de LOGOUT\_REDIRECT\_URL ayarı ile değiştirebiliriz. Çıkış yapıldığında tekrardan login sayfasına yönlendirelim.

LOGOUT\_REDIRECT\_URL = '/accounts/login/

Üye giriş ve çıkış sayfalarını tamamladık. Şimdi yukarıdaki menümüze kullanıcı işlemleri için bir adet menü oluşturalım ve sadece giriş yapan kullanıcılara görüntülenebilsin. Bu menüden çıkış ve şifre değiştirme işlemleri yapılabilsin. Bunun için üst menünün bulunduğu header.html açalım.

{% if user.is\_authenticated %}  
 <li class="nav-item dropdown">  
 <a class="nav-link dropdown-toggle" href="#" id="dropdown01" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">Üye İşlemleri</a>  
 <div class="dropdown-menu" aria-labelledby="dropdown01">  
 <a class="dropdown-item" href="{% url 'password\_change' %}">Şifre Değiştir</a>  
 <a class="dropdown-item" href="{% url 'logout' %}">Çıkış</a>  
 </div>  
 </li>  
{% endif %}

Üye işlemlerinden şifre değiştire tıkladığıdna yönetim panelindeki şifre değiştirme ekranına gitmektedir. Biz herkesin yönetim paneli görüntülerine ulaşmasını istemeyebiliriz. Kendi şifre değiştirme alanımızı oluşturmak için registration/ klasörüne password\_change için kullanılan password\_change\_form.html oluşturmamız yeterli. Sayfayı yenilediğimizde bu templatenin içeriğinin geldiğini görüyoruz.

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<body>  
Buraya şifre değiştirme gelecek  
</body>  
</html>

Formumuzu template dosyamıza girelim.

{% extends 'learning/layout/base.html' %}  
  
{% block title %}  
Şifre Değiştir  
{% endblock %}  
  
{% block top %}  
 <div class="jumbotron">  
 <div class="container">  
  
 <h1 class="display-3">Şifre Değiştirme!</h1>  
 <p>Merhaba {{ user.username }}. Aşağıdaki formdan şifrenizi değiştirebilirsiniz. </p>  
 </div>  
 </div>  
{% endblock %}  
  
{% block content %}  
 <form method="post">  
 {% csrf\_token %}  
 {{ form.as\_p }}  
 <button type="submit">Şifre Değiştir</button>  
 </form>  
{% endblock %}

Şifre başarılı olarak değiştiğinde ise password\_change\_done.html si çağırılacaktır. Bu dosyamızı da oluşturalım. Şifre sıfırlama sayfası da yönetim paneli kullanılmaktadır. Registration/ klasörüne password\_reset için kullanılan password\_reset\_form.html oluşturarak kendi formumuzu kullanabiliriz. Sayfayı yenilediğimizde bu templatenin içeriğinin geldiğini görüyoruz.

{% extends 'learning/layout/base.html' %}  
  
{% block title %}  
Şifre Değiştir  
{% endblock %}  
  
{% block top %}  
 <div class="jumbotron">  
 <div class="container">  
  
 <h1 class="display-3">Şifre Değiştirme</h1>  
 <p>Tebrikler şifreniz başarıyla değiştirilmiştir.</p>  
 </div>  
 </div>  
{% endblock %}  
  
{% block content %}  
 Merhaba {{ user.username }}. Şifreniz başarıyla değiştirildi.  
  
{% endblock %}

Şifremizi değiştirelim. Evet şifre değiştirme işlemini başarıyla tamamladık. Son olarak şifremi unuttum kısmı yaparak kullanıcı yönetim sayfalarını tamamlayalım.

{% extends 'learning/layout/base.html' %}  
  
{% block title %}  
Şifre Sıfırlama  
{% endblock %}  
  
{% block top %}  
 <div class="jumbotron">  
 <div class="container">  
  
 <h1 class="display-3">Şifre Sıfırlama!</h1>  
 <p>Şifre sıfırlamayı aşağıdaki form ile yapabilirsiniz. </p>  
 </div>  
 </div>  
{% endblock %}  
  
{% block content %}  
 {% if user.is\_authenticated %}  
 Şifre sıfırlama için çıkış yapılması gerekmektedir.  
 {% else %}  
 <form method="post">  
 {% csrf\_token %}  
 {{ form.as\_p }}  
 <button type="submit">Şifre Değiştir</button>  
 </form>  
 {% endif %}  
{% endblock %}

Şifre unuttum formu başarılı olarak gönderildiğinde password\_reset\_done.html si çağırılacaktır. Bu dosyamızı da oluşturalım. Bu dosyamızı da oluşturalım.

{% extends 'learning/layout/base.html' %}  
  
{% block title %}  
Şifre Sıfırlama  
{% endblock %}  
  
{% block top %}  
 <div class="jumbotron">  
 <div class="container">  
 <h1 class="display-3">Şifre Sıfırlama!</h1>  
 <p>Şifre sıfırlama emaili gönderildi. </p>  
 </div>  
 </div>  
{% endblock %}  
  
{% block content %}  
 Email adresinize şifrenizi sıfırlamak için gerekli olan talimatlar gönderilmiştir.  
 <br><br>  
 Eğer sıfırlama maili gelmezse kayıt olduğunuz mail adresini doğru girdiğinizden emin olun ve  
 spam klasörünü kontrol edin.  
{% endblock %}

Şifre unutulduğunda mail adresine şifre yenileme linki gönderilir. Bu mailin içeriğini değiştirmek için password\_reset\_email.html oluşturulur.

Bu mail şifrenizi unuttuğunuz için gönderilmiştir.  
Eğer şifre sıfırlamasını siz talep etmediyseniz emaili silebilirsiniz.  
  
Sıfırlama talep edilen email : {{ email }}  
  
Şifrenizi sıfırlamak için aşağıdaki linki takip ediniz.  
  
{{ protocol}}://{{ domain }}{% url 'password\_reset\_confirm' uidb64=uid token=token %}

Emailde yer alan linke tıklandığında şifre oluşturma formu açılır. Bu sayfanın içeriğini password\_reset\_confirm.html den oluşturabilirsiniz.

{% extends 'learning/layout/base.html' %}  
  
{% block title %}  
Şifre Oluştur  
{% endblock %}  
  
{% block top %}  
 <div class="jumbotron">  
 <div class="container">  
  
 <h1 class="display-3">Şifre Oluşturma!</h1>  
 <p>Şifrenizi aşağıdaki form ile oluşturabilirsiniz. </p>  
 </div>  
 </div>  
{% endblock %}  
  
{% block content %}  
 {% if user.is\_authenticated %}  
 Şifre sıfırlama için çıkış yapılması gerekmektedir.  
 {% else %}  
 <form method="post">  
 {% csrf\_token %}  
 {{ form.as\_p }}  
 <button type="submit">Şifre Oluştur</button>  
 </form>  
 {% endif %}  
{% endblock %}

Şifreler girilip başarıyla kayıt edilirse şifrenin değiştiği bilgisini ise password\_reset\_complete.html den oluşturabilirsiniz.

{% extends 'learning/layout/base.html' %}  
  
{% block title %}  
Şifre Oluşturuldu  
{% endblock %}  
  
{% block top %}  
 <div class="jumbotron">  
 <div class="container">  
  
 <h1 class="display-3">Şifre Oluşturma!</h1>  
 <p>Şifreniz oluşturuldu. </p>  
 </div>  
 </div>  
{% endblock %}  
  
{% block content %}  
<p>Şifreniz başarıyla oluşturuldu. Giriş yapmak için <a href="{% url 'login' %}">tıklayın</a>.</p>  
{% endblock %}

Evet djangonun bize sağladığı formlar ile bu işlemleri tamamlamış olduk. Şimdi viewlerde kullanabileceğimiz built-in methodlara bakalım. Bunun için learning/views/ klasöründe authentication.py dosyası oluşturalım.

Kullanıcı girişi yapmak için authenticate metodu kullanılır. öncelikle methodumuzu dosyamıza import edelim.

Authenticate methodu username ve password argümanları alır. eğer kullanıcı ve şifre doğrulanırsa true doğrulanmaz ise false döner. Çıkış işlemi için logout metodu kullanılır.

from django.contrib.auth import authenticate, logout

def login(request):  
 username = request.POST['username']  
 password = request.POST['password']  
 user = authenticate(request**,** username=username**,** password=password)  
  
 if user is not None:  
 # Kullanıcı girişi yapılmış  
 pass  
 else:  
 # Kullanıcı girişi yapılmamış.  
 pass

def logout(request):  
 logout(request)

function based viewlerde giriş yapmış kullanıcılar için kısıtlamayı @login\_required decoratörğ ile yapabiliriz. Eğer kullanıcı giriş yapmadı ise settings.py’de yer alan login url yönlendirilir. Başka bir yere yönlendirmek istersek login\_url argümanı ekleyebiliriz.

from django.contrib.auth.decorators import login\_required

@login\_required(login\_url='/accounts/login/')  
def view\_profile(request):  
 pass

Class based viewlerde ise LoginRequiredMixin ile sadece giriş yapmış kullanıcıların view kullanmasını sağlayabiliriz.

class ViewProfile(LoginRequiredMixin**,** View):  
 login\_url = '/login/'  
  
 def get(self):  
 pass

Bazı parçaların giriş yapmış kullanıcılar için kullanılmasını istiyorsan is\_authenticated ile giriş yapmış kullanıcıları yakalayabiliriz.

def auth\_method(request):  
  
 if request.user.is\_authenticated:  
 # Kullanıcı girişi yapılmış  
 pass  
 else:  
 # Kullanıcı girişi yapılmamış  
 pass

@user\_passes\_test dekoratörü ile function based viewleri bazı kısıtlamalara tabi tutabiliriz. Argüman olarak girilen method eğer true dönerse method çalıştırılır. False dönerse ise settings.py deki login url yönlendirilir. Farklı bir adrese yönlendirmek için login\_url argümanı girilir.

from django.contrib.auth.decorators importuser\_passes\_test

def email\_check(user):  
 return user.email.endswith('@example.com')  
  
  
@user\_passes\_test(email\_check**,** login\_url='/login/')  
def my\_view(request):  
 pass

class based viewler için ise UserPassesTestMixin kullanılır. Sınıf içeirsindeki test\_func çalıştırılır. Eğer false dönerse login url yönlendirilir.

from django.contrib.auth.mixins importUserPassesTestMixin

class MyView(UserPassesTestMixin**,** View):  
  
 def test\_func(self):  
 return self.request.user.email.endswith('@example.com')

Django hazır olarak yetkilendirmede sunmaktadır. Daha önceki videolarda her model için django varsayılan olarak add, delete, edit ve view yetkilerini otomatik olarak oluşturduğunu farklı yetkileride meta ayarlarından ekleyebildiğimizden bahsetmiştik. Product modeli için varsayılan yetkiler: add\_product, change\_product, delete\_product, view\_product

Gruplar ve kullanıcı bazında yetkilendirme yapılabilir. Kullanıcı bir gruba eklendiğinde o grubun yetkilerinide sahiplenir. Kullanıcıların bir yetkiye sahip olup olmadığı has\_perm metodu ile sorgulanabilir. Eğer true dönerse yetkisi var false dönerse yok demektir.

user.has\_perm('learning.add\_product')

function based viewlerde yetki kontrolü için @permission\_required dekoratörü kullanılır. Eğer yetki yok ise login\_url argümanına yönlenir. Varsayılan olarak settings.py’deki login url dir. Dekoratörün alabileceği diğer argüman ise raise\_exceptiondır eğer True ise PermissionDenied hatası döndürür. (http 403)

learning uygulamasının product modeline ekleme yetkisi yoksa;

settings.py deki login url yönlenir.

from django.contrib.auth.decorators import permission\_required  
@permission\_required('learning.add\_product')

accounts/login yönlenir.

@permission\_required('learning.add\_product', login\_url='/accounts/login/')

PermissioDenied hatası verir

@permission\_required('learning.add\_product', raise\_exception=True)

Kullanıcı girişi yapılmamışsa login url yönlenir.  
Aksi taktirde PermissioDenied hatası verir

@login\_required   
@permission\_required('learning.add\_product', raise\_exception=True)

Class based viewler için PermissionRequiredMixin kullanılır.

from django.contrib.auth.mixins importPermissionRequiredMixin

class MyView(PermissionRequiredMixin**,** View):  
 permission\_required = 'learning.add\_product'  
 # çoğul yetki için  
 permission\_required = ('learning.add\_product'**,** 'learning.view\_product')

Template dosyalarında da yetki sorgulaması yapılabilir.

{% if perms.learning %}  
 <p>Learning uygulamasında bazı yetkilere sahipsiniz.</p>  
 {% if perms.learning.add\_product %}  
 <p>Ürün ekleyebilirsiniz!</p>  
 {% endif %}  
 {% if perms.foo.view\_product %}  
 <p>Ürün görüntüleyebilirsiniz!</p>  
 {% endif %}  
{% else %}  
 <p> Learning uygulamasında hiçbir yetkiye sahip değilsiniz.</p>  
{% endif %}

Bir sonraki videomuzda uygulamamızı yazmaya başlayacağız. Şimdiye kadar öğrendiklerimiz Djangonun temel bilgileriydi uygulamamız için bu temel bilgileri bilmemiz şart. Eğer kafanızda oturmayan yerler varsa videoları tekrardan izleyip örnekleri yazmaya çalışın. Uygulama yazmaya başladığımızda daha hızlı ilerleyeceğiz. Uygulamayı yazarken Django hakkında bilgiler vermeye devam edeceğim. Sadece bu öğrendiklerimizle uygulama yapmayacağız. Ama detaya girebilmek için temelin sağlam olması gerekiyor.