

*Django*

Erabiltzaileen kontrola



# Erabiltzaileen kontrola Django

- Django berez dakar erabiltzaile-autentifikaziorako sistema.
- Sistema horrek, besteak beste, honakoak lantzen ditu:
  - Erabiltzaileak
  - Baimenak
  - Taldeak
  - Cookieetan oinarritutako erabiltzaile-saioak



# Instalazioa

- `django.contrib.auth` paketea jasotzen dira erabiltzaileen autentifikazioari buruzkoak.
  - Proiektuaren *settings.py* fitxategian, `INSTALLED_APPS` aukeran, `'django.contrib.auth'` and `'django.contrib.contenttypes'` aktiboan jarri behar dira.
  - Gero, proiektuaren datu-basea sinkronizatzean, erabiltzaileen kudeaketarako taulak sortzen dira.



# Erabiltzaileak (*Users*)

- *User* eredua

- Hainbat eremu eta metodo ditu aurredefinituta.
- Eremuak:
  - `username (beharrezkoa), first_name, last_name, email, password (beharrezkoa), is_staff, is_active, is_superuser, last_login, date_joined`
- Metodoak.
  - Adibidez, honako hauek: `is_authenticated()`, `get_full_name()`, `set_password(pasahitz_gordina)`, `check_password(pasahitz_gordina)`
- *Manager* metodoak:
  - `create_user(username, email=None, password=None)`
  - Erabilpena, erabiltzaile berri bat sortzeko:

```
>>> from django.contrib.auth.models import User
>>> erab = User.objects.create_user('jon', 'jon001@ehu.es', 'pasahitza')
```
- Super-erabiltzaileak sortzeko:

```
manage.py createsuperuser --username=jone --email=jone002@ehu.es
```



# Erabiltzaileak informazio osagarriarekin.

## *User profile*

- Erabiltzaileak (user) aldez aurretik datoz definituak Django ingurunean. Baina webgune bateko erabiltzaileei informazio osagarria erantsi nahi bazaie, profil (***user profile***) moduko eredu bat defini daiteke.
- Erabiltzaile-profilak definitzeko:
  - Definitu eredu (*model*) bat, eta bertan:
    - Jarri eremu bat **OneToOneField** motakoa **User** eredura.
      - Horrela bermatzen da profil horretako instantzia bakarra egongo dela *user* bakoitzeko.
    - Jarri eremu eta metodo osagarriak.

```
from django.contrib.auth.models import User

class ErabiltzaileProfila(models.Model):
    # Beharrezko eremua
    user = models.OneToOneField(User)

    # Eremu osagarriak
    gogoko_animalia = models.CharField(max_length=20)
```

- Webgune horretan erabiltzaile-profila zein den adierazteko, *settings.py* fitxategian:

```
AUTH_PROFILE_MODULE = 'aplikazioa.ErabiltzaileProfila'
```



# Erabiltzaileen autentifikazioa

- Django, `django.contrib.auth` paketea, bi funtzio daude aurredefinituta erabiltzaileen autentifikazioarekin lan egiteko: `authenticate()` eta `login()`.
- Adibide honek erakusten du nola erabil daitezkeen bi funtzio horiek erabiltzaile bat aplikazioan sartzeko:

```
from django.contrib.auth import authenticate, login

def nire_bista(request):
    erabiltzailea = request.POST['username']
    pasahitza = request.POST['password']
    user = authenticate(username=erabiltzailea, password=pasahitza)
    if user is not None:
        if user.is_active:
            login(request, user)
            # Berbideratu login ondorengo orri batera.
        else:
            # Errore-mezua bueltatu: 'kontua desgaituta'.
    else:
        # Errore-mezua bueltatu: 'login desegokia'.
```



## Erabiltzaileen autentifikazioa (II)

- Aplikaziotik kanporatzeko erabili `logout()` funtzioa:

```
from django.contrib.auth import logout

def logout_bista(request):
    logout(request)
    # Berbideratu arrakasta orri batera.
```

- Bista hau exekutatu ondoren, erabiltzaile-saioa (*session*) bukatu egiten da, eta informazio guztia ezabatu egiten da.



# Nola mugatu atzipena erabiltzaile baimendunei

- Modu oinarritzkoa:

```
from django.http import HttpResponseRedirect

def nire_bista(request):
    if not request.user.is_authenticated():
        return HttpResponseRedirect(...)
    else: # baimena badu
        # ...
```

- login\_required dekoratzailea:

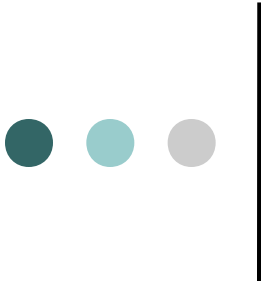
```
from django.contrib.auth.decorators import login_required

@login_required(login_url='...')
def nire_bista(request):
    ...
```

- Dekoratzailerak honakoa egiten du:

- Erabiltzailea ez badago barruan, login\_url helbidera birbideratzen du.
- Erabiltzailea barruan badago, bista normal exekutatzen du.
- Bistaren kodeak suposa dezake erabiltzailea barruan dagoela.



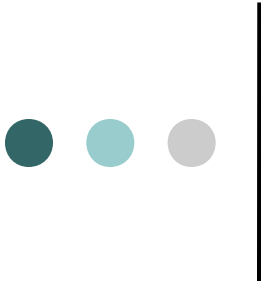


# Aldez aurretik definituriko bistak (*built-in views*)

- Honako paketeen daude: `django.contrib.auth.views`

- `login(request[, template_name, redirect_field_name, authentication_form])`
  - GET bidez deituta: login formulario bat bistaratzen du, eta formulario horrek POST bidez bidaltzen ditu formulario-datuak URL berera.
  - POST bidez deituta: erabiltzailearen sarrera egiten saiatzen da.
    - Arrakastatsua bada, *next* aldagaian zehaztutako helbidera birbideratzen da bista. *next* aldagairik ez bada adierazten, `settings.LOGIN_REDIRECT_URL` helbidera birbideratzen da.
    - Arrakastatsua ez bada, berriz bistaratzen du *login* formularioa.
  - *login* bistak `registration/login.html` txantiloia erabiltzen du besterik ezean. Txantiloia honi lau testuinguru-aldagai pasa dakizkioke: *form*, *next*, *site*, *site\_name*
  - *login* bistaren erabilera, `registration/login.html` ez den beste txantiloia bat erabiliz

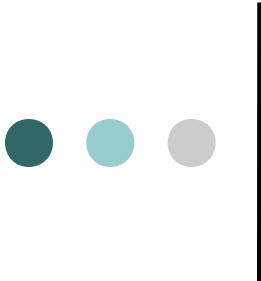
```
url(r'...', 'django.contrib.auth.views.login',  
    {'template_name': '...',  
     'extra_context': {'next': '...'}},  
    name='login')
```



# Aldez aurretik definituriko bistak (*built-in views*) (II)

- `logout(request[, next_page, template_name, redirect_field_name])`

```
url(r'...', 'django.contrib.auth.views.logout',  
    {'template_name': '...'}, name='...'),
```



# Aldez aurretik definituriko formularioak (*built-in forms*)

- Honako paketeen daude: `django.contrib.auth.forms`

- Adibidez, `UserCreationForm`

- Erabilera-adibidea:

```
url(r'...', CreateView.as_view(  
    template_name='...',  
    form_class=UserCreationForm,  
    success_url='...' ) ),
```



# Erabiltzaileen kudeaketan sakontzeko

- <http://teknikariak.informatika.ehu.es/Django/topics/auth.html>
- Saioen kontrola:
  - <http://teknikariak.informatika.ehu.es/Django/topics/http/sessions.html>