

*Django*

*Internazionalizazioa*

*eta*

*lokalizazioa*



# Web-aplikazioak eta hizkuntzak

- Gaur egun, behar-beharrezkoa da web-aplikazioak hizkuntza desberdinetara egokitzea.
- Garatzaileek kontuan hartu behar dituzte web-aplikazioaren edukiak hizkuntza bat baino gehiagotan eskaintzeko baliabideak.
- Besteak beste, beharrezkoa izaten da:
  - Testuak itzultzea
  - Datak egoki formateatzea
  - Zenbakiak eta denbora-unitateak egoki tratatzea
  - ...
- Hurbilpen egokia:
  - Aplikazioaren “hezurdura” bakarra
  - Hizkuntza bakoitzeko testuak eta xehetasunak aparte eduki, eta, komeni denean, kargatu



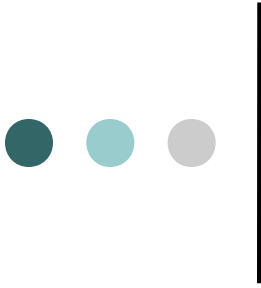
# Internazionalizazioa eta lokalizazioa

- Txanpon beraren bi aldeak dira:
  - Internazionalizazioa: Softwarea prestatzea lokalizaziorako. Eginkizun hau software-garatzaileri dagokio.
    - Ikuspegia: lokaletik globalera.
  - Lokalizazioa: Itzulpenak eta egokitzapenak egitea. Eginkizun hau itzultzaileei egokitu ohi zaie.
    - Ikuspegia: globaletik lokalera.
- Djangok eskaintzen ditu tresnak eta baliabideak eginkizun honetarako.



# Testuinguru honetan erabiltzen diren terminoak eta kontzeptuak.

- *locale*
  - Parametroen multzo bat, adierazteko zein **hizkuntza**, herrialdea, etab. nahi dituen erabiltzaileak interfazeaz.
- *locale name*
  - *locale* balioa adierazteko identifikadorea: ll formatuan edo ll\_HH formatuan. Adibidez: eu, pt\_BR. Hizkuntzaren aldea beti minuskulaz, herrialdearena beti maiuskulaz.
- *language code*
  - Hizkuntza baten izena errepresentatzen du. Adibidez, eu, pt-br.
- *message file*
  - Testu-fitxategia da. Hizkuntza bateko itzulpen-kate guztiak biltzen ditu, eta string horiek nola adierazi behar diren. Fitxategi hauek .po luzapena dute.
- *translation string*
  - Itzuli beharreko literalak.
- *format file*
  - Python modulu bat, *locale* jakin baterako datuen formatua definitzen duena.



# Django aplikazioa hizkuntza bat baino gehiagotan. Nola egokitu?

- Ingurunea prestatu.
  - *settings.py*:
    - `MIDDLEWARE_CLASSES` ezarpenean: `django.middleware.locale.LocaleMiddleware`
      - `django.contrib.sessions.middleware.SessionMiddleware`-ren ondoren
      - `django.middleware.common.CommonMiddleware` baino lehen
- Erabaki behar da zein diren itzuli eta egokitu beharreko stringak (*translation strings*)
- String horiek itzuli egin behar dira, eta ondoren konpilatu **GNU gettext** tresnaren bidez.
- Prozesu hori burutu ostean, erabiltzaileak aukeratzen duen hizkuntzaren arabera erakusten dira testuak aplikazioan.
  - Erabiltzaileak eskatu ahala kargatzen dira testuak.



# Lokalizazioa: nola sortu mezu-fitxategiak (*message files*)

- Itzulpenak egin aurretik, helburu-hizkuntza berrirako mezu-fitxategia sortu behar da.
- Horretarako, Django:
  - `$ django-admin makemessages -l en`
  - Proiektuaren edo aplikazioaren erro-direktorioan exekutatu behar da.
    - `locale` azpidirektorio bat eskuz sortu behar da.
    - `settings.py`-n, `LOCALE_PATHS` ezarpena sortu eta `locale` azpidirektorioren izenbidea adierazi
  - Komando horrek egiten duena:
    - Proiektuaren edo aplikazioaren mezuak aztertzen ditu, eta *itzultzeko markatuta daudenak* biltzen ditu. Besterik ezean, `.html` eta `.txt` dokumentuak aztertzen ditu.
    - Horiekin mezu-fitxategia sortzen du hizkuntza horretarako. Luzapena: `.po`
  - Itzultzeko stringek **aldez aurretik markatuta egon behar dute**.
    - Adibidez, aplikazioan horrelako string bat edukiz gero: `_("Ongi etorri filmen gunera.")`
    - Horrelakoa litzateke `.po` fitxategia:

```
#: path/to/python/module.py:23
msgid "Ongi etorri filmen gunera."
msgstr ""
```



# Mezu-fitxategien konpilazioa

- `gettext` utilitateak mezu-fitxategia erabil dezan, konpilatu egin behar da (sortzen denean, eta aldatzen den guztietan).
- Horretarako:
  - `$ django-admin compilemessages`
- Konpilazioan fitxategi bitar bat sortzen da, `.mo` luzapena edukiko duena.
- Windows-en instalatzeko,  
<http://teknikariak.informatika.ehu.es/Django/topics/i18n/translation.html#gettext-on-windows>



# Itzuli beharreko stringak nola markatu

- Bistetan:

```
from django.utils.translation import ugettext as _
```

```
def nire_bista(request):  
    output = _("Ongi etorri.")  
    return HttpResponse(output)
```

```
def nire_bista(request, h, e):  
    output = _('%(hil)aren %(egun)a da.') % {'hil': h, 'egun': e}  
    return HttpResponse(output)
```





# Itzuli beharreko stringak nola markatu (II)

## ○ Txantiloietan:

- Itzulpena behar duten txantilo guztietan jarri hasieran: `{% load i18n %}`
  - Txantilo batek beste bat luzatzekotan, “**extends**” etiketaren ondoren
- Itzulpenetarako bi etiketa (*tag*) erabiltzen dira: `trans` eta `blocktrans`.
- `{% trans %}`: karaktere-kateak (komatxoaren artean) edo aldagaien edukiak itzultzen ditu.
  - Agerikoa ez bada ere, etiketa hau erabiltzean `gettext()` funtzioari dei egiten zaio.

```
<title>{% trans "Ongi etorri." %}</title>
<title>{% trans nirealdagaia %}</title>
```

- `{% blocktrans %}`: esaldi konplexuagoak markatzeko aukera ematen du, leku-markak (*placeholders*) erabiliz.

```
{% blocktrans %}Aukera honek {{ balioa }} hau du.{% endblocktrans %}
```



# Nola landu uneko hizkuntza

- Djangok modu malgua dauka zein hizkuntza erabili behar duen erabakitzeko: instalazioa, erabiltzaile partikularra, biak.
  - Instalazioa: *settings.py* fitxategian, zehaztu `LANGUAGE_CODE` ezarpena.
    - Hizkuntza zein den jakiteko: `settings.LANGUAGE_CODE`
  - Erabiltzaileak aukeratu dezan zein den bere gustuko hizkuntza:
    - Aktibatu `LocaleMiddleware`:
      - *settings.py* fitxategian, `MIDDLEWARE_CLASSES` ezarpenean erantsi `'django.middleware.locale.LocaleMiddleware'`
    - Sistemak begiratzen du zein den erabili beharreko hizkuntza ordena honi jarraituz: URLaren hizkuntza-aurrizkia, saioaren `django_language` **gakoa**, `LANGUAGE_COOKIE_NAME` **cookiea**, `Accept-Language` **HTTP burukoa**, eta `LANGUAGE_CODE` **ezarpen globala**.
    - Hizkuntza zein den jakiteko: `request.LANGUAGE_CODE`

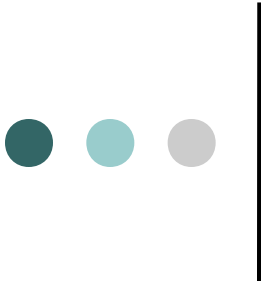


# URL patroiak eta itzulpenak

- `i18n_patterns` funtzioak aukera ematen du automatikoki hizkuntza-aurrizkia URLei gehitzeko:

```
from django.conf.urls import patterns, include, url
from django.conf.urls.i18n import i18n_patterns
```

```
urlpatterns = i18n_patterns('',
    url(r'^about/$', 'about.view', name='about'),
    url(r'^news/$', include(..., namespace='news'))),
```



# Internalizazio-lokalizazio kontuetan sakontzeko

- <http://teknikariak.informatika.ehu.es/Django/topics/i18n/>
- <http://teknikariak.informatika.ehu.es/Django/topics/i18n/translation.html>