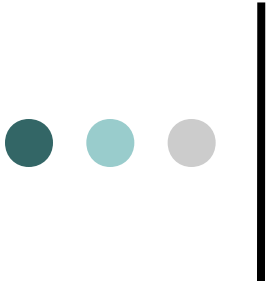


Django

Txantiloι-geruza (*template layer*)



Txantilo-geruza (*template layer*)

- Djangoren txantilo-geruzan:
 - **Aurkezpenaren logika** lantzen da.
 - Bista-geruzan sortutako emaitzak erakusteaz arduratzen da.
 - Datuak eta datuen aurkezpena ondo bereiztea da xedea.



Datuen aurkezpena txantiloirik gabe. Adibidea

```
from bozketak.models import Bozketa
from django.http import HttpResponse

def index(request):
    azken_bozketen_lista = Bozketa.objects.all().order_by('-arg_data')[:5]
    emaitza = ', '.join([p.galdera for p in azken_bozketen_lista])
    return HttpResponse(emaitza)
```

- Bistaren kodea eta aurkezpenaren xehetasunak nahastu egiten dira!

Fitxategien antolaketa txantiloiak erabiltzeko

nireproiektua/

manage.py

nireproiektua/

__init__.py

settings.py

urls.py

wsgi.py

aplikazioa/

templates/

Djangok atzitzeko moduko kokagunean

```
TEMPLATE_DIRS = (  
    "../nireproiektua/templates"  
)
```



Bistak eta txantiloiak

- Bistek erabiltzaile-eskaera bat jasotzen dute (*HttpRequest*) eta *HttpResponse* bat itzultzen dute.
- *HttpResponse* objektu hori *errendatzeko* (*render*) erabiltzen dira **txantiloiak**.
 - Informazioa aurkezteko marko orokorrak dira.
 - Gero ikusiko dugunez, aldagaiak eta bestelako bloke-etiketak edukitzen dituzte.
- *HttpResponse* objektua itzultzeko prozesua hiru urratsetan:
 1. **Txantiloia** kargatu
 2. **Testuingurua** osatu; alegia, txantiloia hori betetzeko erabiliko diren datuak zehaztu.
 - Testuinguru batean aldagai-izenei beren balioak ezartzen zaizkie.
 3. *HttpResponse* objektua itzuli, osatutako txantiloia errendatuz.
 - `render(testuingurua)` metodoa erabiltzen da txantiloia errendatzeko.
- Urrats horiek batera egiteko, `render_to_response()` lasterbidea (*shortcut*) erabil daiteke.



Txantiloien erabilera. Adibideak

```
from django.template import Context, loader
from bozketak.models import Bozketa
from django.http import HttpResponse

def index(request):
    azken_bozketen_lista = Bozketa.objects.all().order_by('-arg_data')[:5]
    t = loader.get_template('bozketak/index.html')
    c = Context({
        'azken_bozketen_lista': azken_bozketen_lista,
    })
    return HttpResponse(t.render(c))
```



Txantiloien erabilera. Adibideak (II)

- `render_to_response()` lasterbidea erabiliz

```
from django.shortcuts import render_to_response
from bozketak.models import Bozketa

def index(request):
    azken_bozketen_lista = Bozketa.objects.all().order_by('-pub_date')[:5]
    return render_to_response('bozketak/index.html',
                              {'azken_bozketen_lista': azken_bozketen_lista})
```



Txantiloien erabilera. Adibideak (III)

- 404 salbuespena landuz, erantzuna *renderizatu* aurretik.

```
from django.http import Http404
# ...
def xehetasunak(request, bozketa_id):
    try:
        p = Bozketa.objects.get(pk=bozketa_id)
    except Bozketa.DoesNotExist:
        raise Http404
    return render_to_response('bozketak/xehetasunak.html', {'bozketa': p})
```




Txantiloien erabilera. Adibideak (IV)

- `get_object_or_404()` lasterbidea erabiliz

```
from django.shortcuts import render_to_response, get_object_or_404
# ...
def xehetasunak(request, bozketa_id):
    p = get_object_or_404(Bozketa, pk=bozketa_id)
    return render_to_response('bozketak/xehetasunak.html', {'bozketa': p})
```



Txantiloiak

- Txantiloiak testu-fitxategiak dira, eta testu-oinarria duen edozein formatu sor dezakete (HTML, XML, CSV, etab.).
- Marko orokorra eskaintzen dute, eta beren barruan honako osagaiak eduki ditzakete:
 - Aldagaiak
 - Etiketak (*tag*)
 - Iragazkiak (*filter*)
 - Iruzkinak edo oharrak



Aldagaiak

- Sintaxia: `{{ aldagaia }}`
- Txantiloien motorrak aldagai bat aurkitzen duenean:
 - Ebaluatu egiten du
 - Aldagaiaren tokian ebaluazioaren emaitza jartzen du
- Aldagaien identifikadoreak: karaktere alfanumerikoak eta *underscore* ("_") erabil daitezke aldagai-izenak osatzeko.
- Aldagai baten atributuak atzitzeko, puntu-notazioa erabili.



Iragazkiak (*filter*)

- Sintaxia: “|” ikurra erabili behar da iragazkia aplikatzeko.
- Adibidez: `{{ izena|lower }}`
 - `izena` aldagaiaren balioa erakutsiko du, baina lehenago `lower` iragazkia pasata, hau da, karaktere guztiak minuskula bihurtuta.
- Iragazkiak kateatu egin daitezke: baten irteera hurrengoari pasatzen zaio.
 - Adibidez: `{{ testua|escape|linebreaks }}`
 - `testua` aldagaiko karaktere bereziak HTMLko entitate bihurtzen ditu, eta lerro-jauziak `<p>` etiketa.
- Iragazki batzuek argumentuak hartzen dituzte, adibidez:
 - `{{ bio|truncatewords:30 }}`: `bio` aldagaiaren lehenengo 30 hitz.
 - Argumentuek zuriuneak badituzte, komatxoaren artean jarri behar dira.
 - Adibidez, lista bat komekin eta zuriuneekin lotzeko: `{{ list|join:", " }}`.
- Aurrez definitutako iragazkien zerrenda:
 - <http://teknikariak.informatika.ehu.es/Django/ref/templates/builtins.html#built-in-filter-reference>



Iruzkinak

- Sintaxia: `{# iruzkina #}`
- Iruzkinak ez dira bistaratzen. Oharrak dira, txantiloia-aren edukia deskribatzeko-eta erabiltzen direnak.



Etiketak (*tag*)

- Sintaxia: `{% etiketa %}`
- Etiketek erabilera anitz izan ditzakete:
 - Irteeran testua sortu
 - Kontrol-egiturak definitu, iterazioak-eta exekutatzeko
 - Kanpoko informazioa kargatu, gero aldagaien bidez erabili ahal izateko
- Zenbaitetan hasierako eta bukaerako etiketak erabili behar dira:
 - `{% etiketa %} ... edukiak ... {% endetiketa %}`
- Aurrez definitutako etiketen zerrenda:
 - <http://teknikariak.informatika.ehu.es/Django/ref/templates/builtins.html#built-in-tag-reference>
 - Adibidez:
 - `for`, `if` and `else`, `block` and `extends`



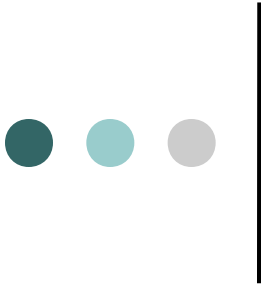
Etiketak erabiltzeko adibideak

- for egitura:

```
<ul>
  {% for atleta in atleta_lista %}
    <li>{{ atleta.izena }}</li>
  {% endfor %}
</ul>
```

- Denetarik:

```
{% if atleta_lista|length > 1 %}
  Taldea: {% for atleta in atleta_lista %} ... {% endfor %}
{% else %}
  Atleta: {{ atleta_lista.0.izena }}
{% endif %}
```



block – extends, eta txantiloien herentzia.

Adibidea.

- Adibide bat, ***oinarria.html*** txantiloia (bi zutabeko dokumentuen hezurdura sinplea):

```
<!DOCTYPE html>
<html lang="en">
<head>
  <link rel="stylesheet" href="style.css" />
  <title>{% block izenburua %}Nire gunea{% endblock %}</title>
</head>

<body>
  <div id="sidebar">
    {% block sidebar %}
    <ul>
      <li><a href="/">Home</a></li>
      <li><a href="/blog/">Blog</a></li>
    </ul>
    {% endblock %}
  </div>

  <div id="edukia">
    {% block edukia %}{% endblock %}
  </div>
</body>
</html>
```




block – extends (II)

- *oinarria.html* dokumentuan hiru `block` elementu definitzen dira.
- Bloke horiek *txantilo* umeein bete daitezke.

```
{% extends "oinarria.html" %}

{% block edukia %}
{% for sarrera in blog_sarrerak %}
    <h2>{{ sarrera.izenburua }}</h2>
    <p>{{ sarrera.gorputza }}</p>
{% endfor %}
{% endblock %}
```

- Txantilo
- honetan *oinarria.html* dokumentua hedatzen du. Honela:
 - *oinarria.html* txantiloia (gurasoa) kargatzen du.
 - Ikusten du zein diren `block` elementuak gurasoan, eta horien ordean umean definitutakoak jartzen ditu.
 - Definitu gabe dauden blokeak gurasotik hartzen ditu: gure adibidean, `izenburua` eta `sidebar`.
- <http://teknikariak.informatika.ehu.es/Django/topics/templates.html#template-inheritance>



Txantiloiak eta segurtasuna

- Aldagaien balioak txantiloietan kargatzen direnean, *Cross Site Scripting (XSS)* arriskua dago:
 - Adibidez, imajinatu honako txantiloia, itxuraz arazorik gabea:

```
Kaixo, {{ izena }}
```
 - Eta orain suposatu `izena` aldagaiak honako balioa duela:

```
<script>alert('kaixo')</script>
```
 - Txantiloia horrela renderizatuko litzateke:

```
Kaixo, <script>alert('kaixo')</script>
```
 - Django, erabilera desegoki hori eragozteko, deskodetu egiten ditu karaktere arriskutsuak besterik ezean.
- *Cross Site Request Forgery* babesaz gehiago jakiteko:
 - <http://teknikariak.informatika.ehu.es/Django/ref/contrib/csrf.html>



Txantiloiei buruzko ezagutza sakontzeko

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