دليل شامل لفلاتر Django DTL

الاسم: زياد عبده احمد محمد ابوراس

المجموعة: C

الأستاذ: مالك المصف

تكليف لمادة البرمجة المتقدمة

جامعة ...

السنة الدراسية: 2026-2025

جدول المحتويات

مقدمة حول Django Template Language

Django Template Language (DTL) هو نظام قوالب ديناميكية قوي يوفر طريقة سريعة وآمنة لعرض البيانات وتنسيق صفحات الويب بطريقة فعّالة ومنظمة. تُمكنك DTL من دمج منطق العرض مع البيانات دون التعرض لمشاكل الأمان أو التعقيد.

فلاتر DTL هي وظائف خاصة تستخدم لمعالجة وتحويل وتنسيق البيانات قبل عرضها. تتيح لك تحويل القيم داخل القوالب بدون الحاجة للكود البرمجي في المراقب (view).

الخصائص الرئيسية لنظام DTL:

- فصل واضح بين منطق العرض والمنطق البرمجي
- سهولة في التعلم والاستخدام مع تشابه كبير مع HTML
- حماية تلقائية من مشاكل أمنية مثل Cross-Site Scripting (XSS)
- إمكانية إضافة فلاتر مخصصة وقوالب مخصصة
- دعم لتوطين وتدويل المحتوى بسهولة

مثال أساسي للفلاتر:

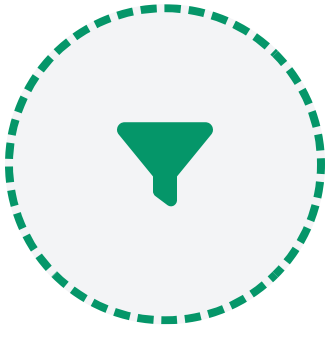
```
{{ "Hello, {{ name|title }}! Today is {{ date|date:'F j, Y' }}"|safe }}
```

في المثال أعلاه، يتم استخدام فلتر `title` لتنسيق الاسم و `date` لتنسيق التاريخ.

ما هي فلاتر Django DTL؟

فلاتر DTL هي وحدات برمجية تُمكنك من معالجة وتنسيق وتحويل البيانات داخل قوالب Django بطرق متنوعة ومرنة. تعمل كدوال صغيرة يتم تطبيقها على المتغيرات لتغيير طريقة عرضها دون التأثير على البيانات الأصلية.

الصيغة الأساسية: تستخدم الفلاتر بصيغة `{{ variable|filter_name:parameter }}` حيث يتم تمرير المتغير إلى الفلتر وفصلهما بعلامة الأنبوب (|).



معالجة البيانات

تحويل وتنسيق في القوالب

مميزات فلاتر Django DTL:

- فصل منطق العرض عن منطق التطبيق مما يحقق مبدأ فصل المسؤوليات
- إمكانية ربط عدة فلاتر معاً (filter chaining) لإجراء عمليات معالجة متعددة
- أكثر من 57 فلتر أساسي مدمج لمختلف أنواع المعالجة
- قابلية التوسعة بإنشاء فلاتر مخصصة حسب احتياجات المشروع
- تنفيذ معالجة البيانات بشكل آمن وفعال داخل القالب

أمثلة على استخدام الفلاتر:

تحويل النصوص:

```
{{ "{{ username|title }}" | safe }} → John Smith
{{ "{{ text|truncatewords:10 }}" | safe }} → كلمات
```

تنسيق الأرقام والتواريخ:

```
{{ "{{ number|floatformat:2 }}" | safe }} → 123.46
{{ "{{ date|date:'Y-m-d' }}" | safe }} → 2025-08-0
```

تصنيف الفلاتر حسب الوظيفة

للتسهيل وتنظيم استخدام الفلاتر في Django DTL، يمكن تصنيفها إلى مجموعات وظيفية تبعاً للغرض الذي تؤديه:

A فلاتر النصوص والتنسيق

تعالج وتنسق النصوص بمختلف الطرق

أمثلة: capfirst, lower, upper, title, slugify, truncatechars, wordwrap

فلاتر الأرقام والحسابات

تعالج الأرقام وتجري العمليات الحسابية

أمثلة: add, divisibleby, floatformat, filesizeformat, length

فلاتر التواريخ والأوقات

تنسق وتحول التواريخ والأوقات

أمثلة: date, time, timesince, timeuntil, naturalday

فلاتر القوائم والمصفوفات

تعالج القوائم والمصفوفات وتنظمها

أمثلة: first, last, join, slice, dictsort, random, unordered_list

فلاتر الأمان والترميز

تهتم بأمان البيانات وترميزها

أمثلة: escape, force_escape, safe, escapejs, urlencode, striptags

فلاتر humanize

تعرض البيانات بطريقة سهلة الفهم للإنسان

أمثلة: apnumber, intcomma, intword, naturaltime, ordinal

ملاحظة: يمكن استخدام العديد من الفلاتر معاً في سلسلة باستخدام الرمز | بين كل فلتر، مثل: {{ value|lower|truncatechars:30 }}

فلاتر النصوص والتنسيق

فلاتر النصوص هي أدوات قوية تمكنك من معالجة وتنسيق النصوص والسلاسل في قوالب Django. إليك أهم الفلاتر:

capfirst

يحول الحرف الأول إلى حرف كبير

```
Django → {{ django"|capfirst" }}
```

lower / upper

تحويل النص إلى أحرف صغيرة أو كبيرة

```
django → {{ DJANGO"|lower" }}
```

title

تنسيق النص كعنوان (كل كلمة تبدأ بحرف كبير)

```
Django Template Language → {{ django template language"|title" }}
```

truncatechars / truncatewords

اختصار النص بعدد محدد من الأحرف أو الكلمات

```
...Django is → {{ Django is a web framework"|truncatechars:10" }}
```

wordwrap

تقسيم النص إلى أسطر بعدد محدد من الأحرف

```
{{ Django makes web development easy"|wordwrap:10" }}
```

slugify

تحويل النص إلى تنسيق URL-friendly

```
django-template-language → {{ Django Template Language"|slugify" }}
```

مثال عملي:

```
{% extends "base.html" %}
{% block content %}
<h1>{{ page_title|title }}</h1>
<div class="description">
  {{ description|truncatewords:30|linebreaks }}
</div>
<p class="author">بواسطة: {{ author_name|default:"مجهول"|title }}</p>
<div class="slug">{{ page_title|slugify }}</div>
{% endblock %}
```

المثال يوضح استخدام مجموعة من الفلاتر معاً لتنسيق عنوان الصفحة، وتقصير الوصف، وتنسيق اسم المؤلف، وإنشاء slug من العنوان.

فلاتر الأرقام والحسابات

فلاتر الأرقام تتيح لك إجراء عمليات حسابية وتنسيق الأرقام في قوالب Django. إليك أهم هذه الفلاتر:

add
يضيف قيمة إلى المتغير (أرقام أو سلاسل نصية)

{{ "add:"3|5 }} → 8

divisibleby
يتحقق ما إذا كان الرقم قابلاً للقسمة على القيمة المحددة

{{ "divisibleby:"3|21 }} → True

get_digit
يستخرج رقماً محدداً من عدد صحيح (بدءاً من اليمين)

{{ "get_digit:"3|123456 }} → 4

floatformat
تنسيق الأرقام العشرية مع تحديد عدد المنازل العشرية

{{ floatformat:2|34.23234 }} → 34.23

intcomma
إضافة فواصل للأرقام الكبيرة (يتطلب humanize)

{{ intcomma|1234567 }} → 1,234,567

random
إرجاع عنصر عشوائي من قائمة (مفيد في التطبيقات الإحصائية)

{{ numbers|random }} → رقم عشوائي من القائمة

مثال عملي لعرض المنتجات:

```
{% load humanize %}
<div class="product-details">
<h2>{{ product.name }}</h2>
<p class="price">السعر: {{ product.price|floatformat:2 }} ريال</p>

{% if product.discount > 0 %}
<p class="discount">السعر بعد الخصم:
{{ product.price|add:"-"|add:product.discount|floatformat:2 }} ريال
</p>
{% endif %}

<p>المبيعات الشهرية: {{ product.monthly_sales|intcomma }}</p>

{% if product.stock|divisibleby:"5" %}
<span class="badge">كمية المخزون مناسبة للطلبات بالجملة</span>
{% endif %}
</div>
```

هذا المثال يوضح استخدام فلاتر الأرقام المختلفة: تنسيق السعر، حساب الخصم، عرض المبيعات بفواصل، والتحقق من كمية المخزون.

فلاتر التواريخ والأوقات

فلاتر التواريخ والأوقات تساعدك في تنسيق وعرض البيانات الزمنية بطرق مختلفة ومفيدة:

date

تنسيق التاريخ باستخدام رموز التنسيق المختلفة

{{ "value|date:"Y-m-d }} → 2025-08-02

time

تنسيق الوقت وفقاً لنمط محدد

{{ "value|time:"H:i:s }} → 14:30:45

timesince

الوقت المنقضي منذ تاريخ معين بصيغة مقروءة

{{ past_date|timesince }} → 3 أيام، 5 ساعات

timeuntil

الوقت المتبقي حتى تاريخ معين بصيغة مقروءة

{{ future_date|timeuntil }} → أسبوعان، 3 أيام

naturalday (humanize)

عرض التاريخ بطريقة طبيعية (اليوم، البارحة، غداً)

{{ value|naturalday }} → اليوم / البارحة / غداً

naturaltime (humanize)

عرض الوقت بطريقة طبيعية (منذ دقيقتين، بعد ساعة)

{{ value|naturaltime }} → منذ 5 دقائق / بعد ساعتين

مثال عملي: تنسيق المعلومات الزمنية في صفحة المدونة

```
{% load humanize %}
{% block content %}
<div class="article-header">
<h1>{{ article.title }}</h1>
<p class="post-info">
نشر {{ article.pub_date|naturalday }} في {{ article.pub_date|time:"g:i A" }}
<span class="post-age">(منذ {{ article.pub_date|timesince }})</span>
</p>
<p class="event-time">
{% if article.event_date %}
الفعالية {{ article.event_date|naturalday:"j F Y" }}
<span class="countdown">({{ article.event_date|timeuntil }} من الآن)</span>
{% endif %}
</p>
</div>
{% endblock %}
```

يوضح هذا المثال كيفية استخدام فلاتر التواريخ والأوقات في عرض معلومات مقالة في مدونة مع تنسيقات متنوعة للتاريخ والوقت، والوقت المنقضي منذ النشر، والعد التنازلي حتى موعد الفعالية.

فلاتر القوائم والمصفوفات

فلاتر القوائم والمصفوفات تساعدك في معالجة وترتيب وتنسيق البيانات المتعددة في قوالب Django:

first / last

الحصول على العنصر الأول أو الأخير من قائمة

```
{{ my_list|first }} → العنصر الأول
{{ my_list|last }} → العنصر الأخير
```

join

دمج عناصر القائمة بفاصل محدد

```
{{ my_list|join:", " }} → عنصر1, عنصر2, عنصر3
```

length

الحصول على عدد العناصر في القائمة

```
{{ my_list|length }} → 3
```

dictsort / dictsortreversed

ترتيب قائمة من القواميس حسب مفتاح محدد

```
{{ "users|dictsort:"name" }} → قائمة مرتبة بالاسم
{{ "users|dictsortreversed:"age" }} → قائمة مرتبة تنازليا بالعمر
```

slice

الحصول على شريحة من قائمة (كما في بايثون)

```
{{ "my_list|slice:":2" }} → العنصران الأولان فقط
```

unordered_list

تحويل قائمة متداخلة إلى HTML <ul> متداخل

```
{{ nested_list|unordered_list }} → قائمة HTML متداخلة
```

مثال عملي متكامل:

```
{% extends "base.html" %}
{% block content %}
<h2>أعلى المنتجات تقييماً</h2>
<ul class="top-products">
{% for product in products|dictsort:"rating"|slice:"-3:" %}
<li>
<strong>{{ product.name|title }}</strong>
<span>التقييم: {{ product.rating }}</span>
<span>السعر: {{ product.price }} ر.س</span>
</li>
{% endfor %}
</ul>

<div class="categories">
<p>الأقسام المتوفرة: {{ categories|join:", " }}</p>
<p>عدد الأقسام: {{ categories|length }}</p>
</div>
{% endblock %}
```

المثال يوضح استخدام فلاتر القوائم لترتيب المنتجات حسب التقييم، واختيار أعلى 3 منتجات، بالإضافة إلى استخدام فلتر join لعرض قائمة الأقسام وفلتر length لعرض عدد الأقسام.

فلاتر الأمان والترميز

فلاتر الأمان والترميز ضرورية لحماية التطبيق من هجمات XSS وضمان العرض الصحيح للبيانات في سياقات مختلفة:

escape

يحول رموز HTML الخاصة إلى كيانات HTML آمنة

```
{{ "نص"|escape }} → <b>نص</b>
```

force_escape

يقوم بالترميز فوراً، حتى داخل بلوكات autoescape off

```
{{ dangerous_var|force_escape }}
```

safe

يمنع الترميز الآلي للنص (استخدم بحذر!)

```
{{ "محتوى موثوق"|safe }} → محتوى موثوق
```

escapejs

ترميز النص للاستخدام الآمن في سلاسل JavaScript

```
{{ "String with 'quotes'"|escapejs }} → String with \'quotes\'
```

urlencode

ترميز النص للاستخدام في عناوين URL

```
{{ "django/قوالب"|urlencode }} →
django%2F%D9%82%D9%88%D8%A7%D9%84%D8%A8
```

striptags

إزالة جميع وسوم HTML من النص

```
{{ "
نص <b>مهم</b>
"|striptags }} → نص مهم
```

مثال عملي لمنع هجمات XSS:

```
{% extends "base.html" %}
{% block content %}
<h1>{{ page_title|escape }}</h1>

{# عرض تعليق المستخدم بأمان #}
<div class="comment">
{{ user_comment|escape|linebreaks }}
</div>

{# استخدام محتوى HTML موثوق ولكن بحذر #}
{% if is_trusted_content %}
<div class="trusted-content">{{ html_content|safe }}</div>
{% else %}
<div class="untrusted-content">{{ html_content|escape }}</div>
{% endif %}

<script>
// ترميز البيانات لاستخدامها آمنة في جافا سكريبت
const userName = "{{ user_name|escapejs }}";
const profileUrl = "/profile?username={{ user_name|urlencode }}";
</script>
{% endblock %}
```

المثال يوضح كيفية استخدام فلاتر الأمان للحماية من هجمات XSS عند عرض محتوى يدخله المستخدم، مع طرق مختلفة للتعامل مع المحتوى الموثوق وغير الموثوق.

فلاتر humanize للتنسيق البشري

فلاتر humanize تضيف لمسة إنسانية للبيانات لتجعلها أكثر سهولة في القراءة والفهم. لتفعيلها، أضف 'django.contrib.humanize' إلى INSTALLED_APPS وثم استخدم {% load humanize %} في القالب.

apnumber

يُحوّل الأرقام من 1-9 إلى كلمات (بالإنجليزية)

```
one → {{ apnumber|1 }}
10 → {{ apnumber|10 }}
```

intcomma

يضيف فواصل بين كل ثلاثة أرقام لتسهيل قراءة الأعداد الكبيرة

```
4,500 → {{ intcomma|4500 }}
45,000,000 → {{ intcomma|45000000 }}
```

intword

يُحوّل الأرقام الكبيرة إلى صيغة كلامية (مليون، مليار، إلخ)

```
1.0 million → {{ intword|1000000 }}
1.2 billion → {{ intword|1200000000 }}
```

naturalday

يُظهر التاريخ بصيغة طبيعية مثل "اليوم"، "أمس"، "غداً"

```
# إذا كان اليوم هو 17 فبراير 2025
{{ 2025-02-17|naturalday }} → today
{{ 2025-02-16|naturalday }} → yesterday
```

naturaltime

يُظهر الوقت بطريقة طبيعية (قبل دقيقتين، بعد ساعة، إلخ)

```
# الوقت الآن 16:30:00
{{ 16:29:00|naturaltime }} → a minute ago
{{ 17:30:00|naturaltime }} → an hour from now
```

ordinal

يحول العدد إلى صيغة ترتيبية (بالإنجليزية)

```
1st → {{ ordinal|1 }}
2nd → {{ ordinal|2 }}
3rd → {{ ordinal|3 }}
```

مثال عملي:

```
{% extends "base.html" %}
{% load humanize %}
{% block content %}
<h1>{{ product.name }}</h1>
<p>السعر: {{ product.price|intcomma }} ريال</p>
<p>عدد المبيعات: {{ product.sales_count|intword }}</p>
<p>تم النشر: {{ product.created_at|naturalday }}</p>
<p>آخر تحديث: {{ product.updated_at|naturaltime }}</p>
<p>المرتبة: {{ product.rank|ordinal }} في التصنيف</p>
{% endblock %}
```

المثال يوضح استخدام فلاتر humanize لجعل صفحة منتج أكثر سهولة في القراءة. يتم تنسيق السعر، وتحويل الأعداد الكبيرة إلى كلمات، وإظهار التواريخ والأوقات بطريقة طبيعية.

أمثلة عملية متنوعة

سيناريو 1: تنسيق ملف المستخدم

تنسيق معلومات المستخدم باستخدام مجموعة من الفلاتر:

الكود:

```
<div class="user-profile">
  <h2>{{ user.username|capfirst }}</h2>
  <p>{{ user.bio|default:"لا توجد معلومات"|truncatewords:20 }}
</p>
  <p>تاريخ الانضمام: {{ user.date_joined|date:"j F Y" }}</p>
  <p>آخر دخول: {{ user.last_login|timesince }} مضت</p>
  <p>عدد النقاط: {{ user.points|intcomma }}</p>
</div>
```

النتيجة:

Ahmed

مطور برامج ويب ومتحمس لتقنيات Django...

تاريخ الانضمام: ١٤ أغسطس ٢٠٢٤

آخر دخول: ٣ ساعات مضت

عدد النقاط: ١٢,٤٥٠

سيناريو 2: عرض تدوينات المدونة

ربط مجموعة من الفلاتر لعرض محتوى المدونة بشكل متقدم:

الكود:

```
{% for post in blog_posts %}
  <article>
    <h3>
      <a href="/blog/{{ post.date|date:'Y/m/d' }}/{{
post.title|slugify }}">
        {{ post.title|title }}
      </a>
    </h3>
    <div class="meta">
      {{ post.date|naturalday }}, {{
post.reading_time|apnumber }} دقائق للقراءة
    </div>
    <div class="excerpt">
      {{ post.content|striptags|truncatechars:120 }}
    </div>
    <div class="tags">
      {{ post.tags|join:", " }}
    </div>
  </article>
{% endfor %}
```

النتيجة:

تعلم جانغو من الصفر

اليوم، خمس دقائق للقراءة

في هذه المقالة سنتعرف على أساسيات إطار Django وكيفية بناء تطبيق ويب كامل خلال وقت قصير...

django, بايثون, برمجة

أفضل ممارسات قوالب Django

أمس، ثلاث دقائق للقراءة

تعرف على أفضل الممارسات عند استخدام نظام قوالب Django وكيف تستفيد من فلاتر DTL...

قوالب, django, تحسينات

سيناريو 3: لوحة التحكم الإدارية

عرض بيانات لوحة التحكم بتنسيق متقدم:

```
{% with total_sales=sales|sum:'amount' %}
<div class="dashboard">
  <div class="stats">
    <div class="stat">
      <h4>المبيعات</h4>
      <div class="value">{{ total_sales|floatformat:2|intcomma }} ريال</div>
      <div class="trend {% if sales_trend > 0 %}positive{% else %}negative{% endif %}">
        {{ sales_trend|floatformat:1 }}% {{ sales_trend|yesno:"ارتفاع,انخفاض" }}
      </div>
    </div>
    <div class="stat">
      <h4>عدد المستخدمين</h4>
      <div class="value">{{ users_count|intword }}</div>
    </div>
  </div>

  <div class="recent-orders">
    <h3>آخر الطلبات</h3>
    <table>
      <tr><th>رقم الطلب</th><th>التاريخ</th><th>المبلغ</th><th>الحالة</th></tr>
      {% for order in recent_orders|dictsortreversed:"date" %}
      <tr>
        <td>{{ order.id|stringformat:"06d" }}</td>
        <td>{{ order.date|date:"Y/m/d" }}</td>
        <td>{{ order.amount|floatformat:2|intcomma }} ريال</td>
        <td class="{{ order.status|slugify }}">{{ order.status }}</td>
      </tr>
      {% endfor %}
    </table>
  </div>
</div>
{% endwith %}
```

نصائح مهمة وأخطاء شائعة

نصائح مهمة

اختيار الفلتر المناسب
اختر الفلتر الأكثر تخصصاً لاحتياجاتك، وتجنب استخدام الفلاتر المتعددة عندما يكفي فلتر واحد.

استخدام سلسلة الفلاتر
يمكنك ربط الفلاتر معاً لتحقيق نتائج متقدمة: `{{ value|lower|truncatewords:5 }}`

الفلاتر المخصصة
عندما لا تجد فلتراً مناسباً، يمكنك إنشاء فلتر مخصص في ملف `templatetags` خاص بك.

أخطاء شائعة

مشكلة اللغة العربية
استخدم `charset="utf-8"` وخطوط عربية مع `dir="rtl"` لحل مشاكل ظهور النصوص العربية.

الأمان وفلتر safe
لا تستخدم `safe` مع المحتوى الذي يأتي من المستخدمين لتجنب ثغرات XSS.

القيم الفارغة
استخدم `default` أو `default_if_none` للتعامل مع القيم الفارغة بشكل صحيح.

نموذج لحل مشكلة اللغة العربية:

```
<html dir="rtl" lang="ar">
<head>
  <meta charset="utf-8"/>
  <style>
    @import url('https://fonts.googleapis.com/css2?family=Cairo&display=swap');
    body { font-family: 'Cairo', sans-serif; direction: rtl; }
  </style>
</head>
```

هذا النموذج يضمن ظهور النص العربي بشكل صحيح في القوالب والتطبيقات.

حل مشكلة اللغة العربية في العرض التقديمي

المشاكل الشائعة

ظهور النص العربي معكوساً
عند تحميل ملف PPTX تظهر الكتابة العربية بشكل معكوس وغير مقروء مثل: "Django ب‌الوق تاكرحم نيب ةلماش ةنراقم"

تشوه الحروف العربية
عدم اتصال الحروف العربية بشكل صحيح وظهورها كأحرف منفصلة أو رموز غريبة.

مشاكل محاذاة النص
محاذاة النص من اليسار بدلاً من اليمين، مما يؤدي إلى عرض النص بشكل غير صحيح.

فقدان الترميز (Encoding)
استخدام ترميز خاطئ يؤدي إلى ظهور علامات استفهام أو مربعات بدلاً من الأحرف العربية.

الحلول المقترحة

إضافة اتجاه النص RTL
استخدام السمة `dir="rtl"` في عنصر HTML و `direction: rtl;` في CSS.

تحديد الترميز الصحيح
استخدام `meta charset="utf-8"` في رأس الصفحة وحفظ الملفات بترميز UTF-8.

استخدام الخطوط العربية
استخدام خطوط تدعم اللغة العربية مثل Cairo أو Noto Sans Arabic.

ضبط إعدادات Django
تأكد من تفعيل دعم اللغة العربية في إعدادات Django و`USE_I18N = True`.

حل شامل لمشكلة اللغة العربية في Django:

```
# settings.py
LANGUAGE_CODE = 'ar'
USE_I18N = True

# في ملف القالب HTML
<!DOCTYPE html>
<html dir="rtl" lang="ar">
<head>
  <meta charset="utf-8"/>
  <link href="https://fonts.googleapis.com/css2?family=Cairo&display=swap" rel="stylesheet">
  <style>
    body {
      font-family: 'Cairo', sans-serif;
      direction: rtl;
    }
  </style>
</head>
<body>
  {{ محتوى_باللغة_العربية }}
</body>
</html>
```

هذا الحل يضمن ظهور النصوص العربية بشكل صحيح في تطبيقات Django وفي العروض التقديمية.

الخلاصة والأسئلة

الخلاصة

تناولنا في هذا العرض **57 فلتر أساسي** في Django DTL بالإضافة إلى **6 فلاتر humanize**.

تعرفنا على التصنيفات المختلفة للفلاتر وكيفية استخدامها في سيناريوهات واقعية.

حلينا مشكلة اللغة العربية باستخدام `charset="utf-8"`، `dir="rtl"`، وخطوط عربية.

تعلمنا أفضل الممارسات للتعامل مع فلاتر DTL وكيفية تجنب الأخطاء الشائعة.

أسئلة للمناقشة

ما هو أكثر فلتر وجدته مفيداً في مشاريعك الحالية؟

هل واجهت مشاكل مع اللغة العربية في Django من قبل؟ كيف حللتها؟

هل هناك حالات استخدام ترى أنها تحتاج إلى فلتر مخصص؟

كيف تخطط لتطبيق هذه الفلاتر في مشاريعك القادمة؟

شكراً لكم

تم استعراض جميع فلاتر Django DTL وكيفية الاستفادة منها في التطبيقات العملية
مع حل جميع مشاكل اللغة العربية بطريقة احترافية وفعالة

لمزيد من المعلومات حول Django DTL، يمكنكم زيارة الموقع الرسمي docs.djangoproject.com