عمل الطالب:ريان امين سيف عبدالقوي

اشراف الاستاذ:مالك المصنف

دليل شامل لإتقان فلاتر Django في القوالب الديناميكية والفعّالة

1. المقدمة

تعتبر **الفلاتر (Filters)** في Django أداة قوية ضمن نظام القوالب (Templates). تساعد هذه الفلاتر على تنسيق البيانات وعرضها بطريقة ديناميكية وفعّالة. هذا الدليل موجّه خصيصًا لمطوري Django في اليمن، ويهدف إلى رفع مهاراتهم في التعامل مع القوالب بشكل احترافي.

---

2. الفلاتر المدمجة (Built-in Filters)

يوفر Django مجموعة كبيرة من الفلاتر الجاهزة التي تغطي معظم الاحتياجات.

2.1 أمثلة شائعة:

- **date**: تنسيق التاريخ.

```
{{ article.created_at|date:"d-m-Y" }}
```

- **truncatechars**: تقصير النصوص.

```
{{ post.content|truncatechars:100 }}
```

- **default**: تحديد قيمة افتراضية إذا كانت المتغيّر فارغًا.

```
{{ user.phone|default:"غير متوفر" }}
```

- **length**: إرجاع طول القائمة أو النص.

```
{{ comments|length }} تعليق
```

---

3. إنشاء فلاتر مخصصة (Custom Filters)

في بعض الحالات، قد تحتاج إلى فلاتر غير موجودة ضمن المكتبة المدمجة.

3.1 الخطوات:

1. أنشئ ملفًا باسم `templatetags/custom_filters.py` داخل التطبيق.
2. اكتب الفلتر المطلوب:

```
from django import template

register = template.Library()

@register.filter(name='currency')
def currency(value):
    return f"{value:,} ريال"
```

1. استدعاء الفلتر في القالب:

```
{% load custom_filters %}
{{ product.price|currency }}
```

---

4. أفضل الممارسات لجعل القوالب ديناميكية وسريعة

- استخدم الفلاتر بدلًا من كتابة منطق معقد داخل القالب.
- قلل الاستعلامات في القوالب عبر **select_related** و **prefetch_related** في الـ views.
- استعمل التخزين المؤقت (caching) عند التعامل مع بيانات كبيرة.
- راقب الأداء باستخدام أدوات مثل **Django Debug Toolbar**.

---

5. حالات استخدام حقيقية من البيئة المحلية (اليمنية)

5.1 أنظمة إدارة المحتوى (CMS)

- استخدام فلتر **truncatewords** لتلخيص المقالات.
- فلتر **date** لتنسيق التاريخ بالتقويم المحلي.

5.2 المتاجر الإلكترونية

- فلتر **currency** لإظهار الأسعار بالريال اليمني.
- فلتر **floatformat** لعرض الخصومات بدقة.

5.3 منصات التعليم الإلكتروني

- فلتر **pluralize** للتفريق بين صيغة المفرد والجمع:

```
{{ lessons.count }} درس{{ lessons.count|pluralize:"ان" }}
```

---

6. مقارنة بين الفلاتر (Filters) والوسوم (Tags)

- **Filters**: تُستخدم عادةً لمعالجة قيمة واحدة (تنسيق نصوص، تواريخ، أسعار).
- **Tags**: تُستخدم في المنطق الأكثر تعقيدًا (loops، شروط، تحميل مكتبات).

مثال:

- فلتر: `{{ text|upper }}`
- وسم: `{% for item in items %} ... {% endfor %}`

---

7. نصائح لتحسين الأداء والأمان

- لا تضع منطق معقد داخل القوالب.
- لا تعرض البيانات الحساسة مباشرة.
- استخدم الفلاتر المدمجة للـ **escaping** مثل `escape` و `safe` بحذر.
- اعتمد على **caching** لتقليل الحمل على السيرفر.

---

8. تمارين عملية مع حلول مقترحة

تمرين 1:

- **المطلوب**: أنشئ فلتر يعرض الوقت المتبقي لانتهاء اشتراك الطالب.

```
from django.utils.timezone import now

@register.filter(name='remaining_days')
def remaining_days(expiry_date):
    delta = expiry_date - now().date()
    return f"{delta.days} يوم متبقي"
```

تمرين 2:

- **المطلوب**: استخدم فلتر `truncatechars` لتلخيص وصف منتج في 50 حرفًا.

```
{{ product.description|truncatechars:50 }}
```

---

9. الخاتمة

باستخدام الفلاتر (المبنية والمخصصة)، يمكن لمطوري Django في اليمن بناء قوالب **ديناميكية، آمنة، وسريعة الأداء** تناسب التطبيقات المحلية مثل المتاجر الإلكترونية، أنظمة المحتوى، ومنصات التعليم. هذا الدليل خطوة نحو رفع كفاءة وجودة المشاريع البرمجية.