

Software Engineering

Assignment 3

مقدم من: بشار فيصل محمد سيف

اشراف: مالمالك المصنف

1- مقارنة بين اربع انواع من انواع Django template

| الجانب | Chameleon | Mako | Jinja2 | Django Template Language (DTL) |
|------------------------------|--|--|--|---|
| السرعة | عالي السرعة، موجه للأداء العالي | أسرع جدًا، مثالي للتطبيقات التي تعتمد على الأداء | أسرع من DTL بكثير | أبطأ من البقية (لكنه جيد للمواقع البسيطة والمتوسطة) |
| مدعوم في Django ؟ | لا، يحتاج إعداد يدوي | لا، يحتاج تكامل يدوي | لا، يحتاج إضافته يدويًا | نعم، هو المحرك الافتراضي |
| بنية القالب | يشبه XHTML ، يعتمد على السمات | أقرب إلى كود بايثون داخل HTML | تشبه Python بشكل كبير، وأكثر مرونة | سهلة وبسيطة تشبه HTML مع وسوم خاصة من Django |
| تعامل مع بايثون | متوسط، لا يدعم كل خصائص بايثون | قوي جدًا، يدعم كتابة تعابير معقدة | مرن جدًا، يدعم الحلقات والتعابير المتقدمة | محدود، لا يسمح بتعقيد المنطق البرمجي داخل القالب |
| سهولة التعلم | يحتاج وقت للتعود | أكثر تعقيدًا بقليل | سهل لمن يعرف Python | سهل جدًا للمبتدئين |
| المرونة | مرن، لكنه مقيد ببنية XML | عالي المرونة، لكنه قد يؤدي إلى فوضى | مرن جدًا ويشبه كتابة كود داخل القالب | محدود في المنطق البرمجي داخل القالب |
| الدوال والمرشحات | محدود نسبيًا | قابل للتوسعة، يدعم دوال خاصة | يحتوي على عدد كبير من المرشحات والدوال | يحتوي على عدد محدود من المرشحات |
| الدوال المخصصة (custom tags) | مدعومة ولكن ليست بسيطة جدًا | مدعومة، مرنة جدًا | مدعومة وأسهل من DTL | مدعومة، لكن تتطلب كتابة كود بايثون في ملفات منفصلة |
| الأمان (Auto Escaping) | تلقائي | لا، يحتاج تمكين يدوي | تلقائي | تلقائي |
| الدعم المجتمعي | صغير نسبيًا | متوسط، أقل انتشارًا | كبير، مستخدم أيضًا في Flask وأطر أخرى | ضخم جدًا، متكامل مع Django |
| استخدام في المشاريع | مناسب للمشاريع التي تعتمد على XHTML أو XML | جيد للتطبيقات عالية الأداء | مثالي للمواقع الديناميكية الكبيرة، مشاريع البيانات | مثالي لمواقع الإدارة، CMS، والواجهات المبنية على Django |
| التوافق مع أدوات Django | محدود التوافق | أقل توافقًا مع ميزات Django | جيد، لكن بعض أدوات Django قد لا تعمل مباشرة | عالي التوافق، كل الميزات تعمل معه |
| (i18n) الدعم متعدد اللغات | محدود جزئيًا | مدعوم | مدعوم | مدعوم |
| التحديثات والتطوير | نادر التحديث | أبطأ من غيره | نشط جدًا | نشط جدًا |

2- أنواع فلاتر Django Templates

- يمكن تقسيم الفلاتر في Django إلى مجموعات رئيسية:

1- فلاتر النصوص (String Filters)

| الوظيفة | الفلتر |
|---|---------------|
| يحوّل النص إلى حروف صغيرة. | lower |
| يحوّل النص إلى حروف كبيرة. | upper |
| يجعل أول حرف من كل كلمة كبيراً. | title |
| يجعل أول حرف في النص كبيراً. | capfirst |
| يقطع النص بعد عدد معين من الأحرف. | truncatechars |
| يقطع النص بعد عدد معين من الكلمات. | truncatewords |
| يعطي طول السلسلة (عدد الأحرف) | length |
| يقتطع جزء من السلسلة (مثلاً أول 3 أحرف) | slice:"3" |

2- فلاتر التاريخ والوقت (Date & Time Filters)

| الوظيفة | الفلتر |
|---|--------------|
| تنسيق التاريخ. | date:"Y-m-d" |
| تنسيق الوقت. | time:"H:i" |
| الوقت المنقضي منذ تاريخ معين (مثل "قبل 3 أيام") | timesince |
| الوقت المتبقي حتى تاريخ معين. | timeuntil |

3- فلاتر القوائم والمجموعات (List/Sequence Filters)

| الوظيفة | الفلتر |
|--|------------------|
| يعطي أول عنصر في القائمة. | first |
| يعطي آخر عنصر. | last |
| يعطي عدد العناصر. | length |
| يدمج عناصر القائمة في نص مفصول بفواصل. | join:", " |
| يعيد عنصرًا عشوائيًا من القائمة. | random |
| يرتب القاموس حسب المفتاح. | dictsort |

4- فلاتر منطقية (Boolean / Conditional Filters)

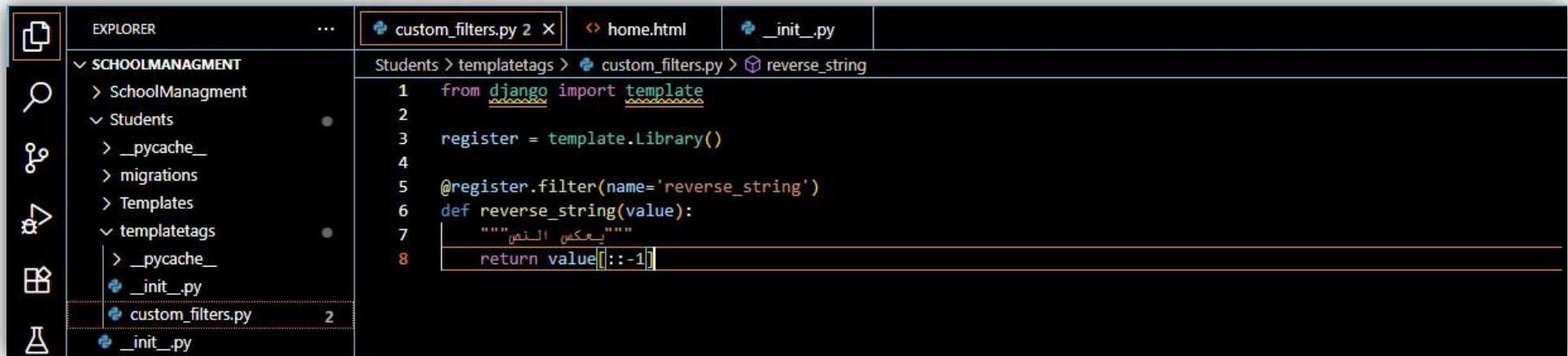
| الوظيفة | الفلتر |
|--|----------------------------|
| يعطي قيمة افتراضية إذا كانت القيمة فارغة أو None. | default:"-" |
| يترجم القيمة المنطقية إلى كلمات. | yesno:"نعم,لا,ربما" |
| (داخل القالب، ليس فلتراً مباشراً لكنه أداة منطقية) | if |

5- فلاتر متنوعة (Miscellaneous)

| الوظيفة | الفلتر |
|---|-----------------------|
| يحول الرموز الخاصة إلى HTML آمن (مثل > و <) | escape |
| يعرض HTML كما هو دون فلترة (خطر أمني إذا لم تُستخدم بحذر) | safe |
| يحول الفواصل إلى فقرات HTML. | linebreaks |
| يعدّ عدد الكلمات في النص. | wordcount |
| تنسيق الأرقام العشرية. | floatformat |
| يعرض الحجم بطريقة مفهومة (مثل 3.2 MB) | Filesizeformat |


3- إنشاء فلتر مخصص (Custom Template Filter)

1- كود إنشاء الفلتر



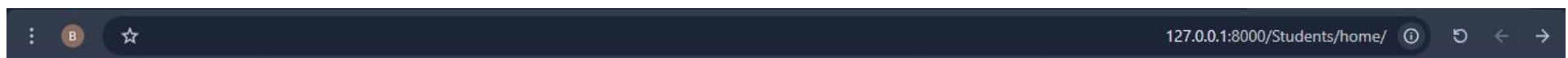
```
1 from django import template
2
3 register = template.Library()
4
5 @register.filter(name='reverse_string')
6 def reverse_string(value):
7     """عكس النص"""
8     return value[::-1]
```

2- استخدام الفلتر في



```
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>home</title>
7 </head>
8 <body>
9
10     {% load custom_filters %}
11
12     <p>النص الأصلي: {{ "Hello Django" }}</p>
13     <p>النص المعكوس: {{ "Hello Django"|reverse_string }}</p>
14
```

3- التنفيذ:



النص الأصلي: Hello Django

النص المعكوس: ognajD olleH