Republic of Yemen IBB University Faculty of Science Departments of IT & CS Systems Analysis and Design		**الجمهورية اليمنية** **جامعة إب** **كلية العلوم التطبيقية** **قسم : علوم الحاسوب** **وتقنية المعلومات** **تحليل وتصميم نظم**

تكليف هندسة برمجيات 3

بسام سمير عبدالملك العريقي

أستاذ المادة / م.مالك المصنف

◇ أولًا: المقارنة بين 3 أنظمة قوالب ((DTL – Jinja2 – Mako

المعيار	Django Template Language (DTL)	Jinja2	Mako
السرعة	أبطأ قليلًا من Jinja2 وMako، لكنه كافي لمعظم التطبيقات.	أسرع من DTL بسبب طريقة المعالجة المسبقة.	أسرع بكثير في الحالات الكبيرة والمعقدة.
الأمان	عالي جدًا، يمنع الحقن (XSS) افتراضيًا.	آمن، مع خاصية الهروب التلقائي مثل DTL.	قوي لكن يحتاج ضبط يدوي لتفعيل الأمان في بعض الحالات.
المرونة	مقيد قليلًا لتفادي كتابة منطق برمجي في القالب.	أكثر مرونة ويدعم منطق برمجي أكبر داخل القوالب.	مرن جدًا ويسمح بكتابة Python مباشرة داخل القالب.
سهولة التعلم	سهل للمبتدئين وخاصة مع Django.	سهل إذا تعرف Python، وصيغته مشابهة لـ DTL.	يحتاج خبرة أكبر لأنه يدمج Python مباشرة.
التكامل مع الإطار	مدمج في Django افتراضيًا.	يُستخدم غالبًا مع Flask أو بشكل مستقل.	يُستخدم بشكل مستقل أو مع Pylons وغيره.
الدعم والمجتمع	كبير بفضل مجتمع Django.	كبير بفضل Flask ومجتمع Python العام.	أصغر نسبيًا.

DTL: خصائص وميزات الفلاتر في

1. التعديل على النصوص

a. `{{ name|upper }}` → يحول النص إلى أحرف كبيرة.

b. `{{ name|lower }}` → يحول النص إلى أحرف صغيرة.

2. تنسيق التواريخ

a. `{{ date|date:"Y-m-d" }}` → يعرض التاريخ بصيغة سنة-شهر-يوم.

3. التعامل مع القوائم

a. `{{ items|length }}` → يعرض عدد العناصر.

b. `{{ items|first }}` → يعرض أول عنصر في القائمة.

4. التعامل مع القيم الفارغة

a. `{{ value|default:"N/A" }}` → يعرض القيمة الافتراضية إذا كانت فارغة.

5. دمج النصوص

a. `{{ list|join:", " }}` → يدمج عناصر القائمة بفاصل معين.

6. إزالة المسافات

a. `{{ text|trim }}` → يحذف المسافات الزائدة من بداية ونهاية النص.

7. الدعم المخصص

a. يمكن إنشاء فلاتر مخصصة بإضافتها في ملف Python داخل تطبيق Django.

تمام، سأعطيك مقارنة قوية بين **4 محركات قوالب للمواقع**:

- **Django Template Language (DTL)**
- **Jinja2**
- **Mako**
- **Mustache**

وسأشرح الميزات، العيوب، متى تختار كل واحد، وأعطيك **أمثلة عملية** للاستخدام، مع أفكار **"أشياء جميلة"** (حيل أو مميزات) لكل محرك.

1. Django Template Language (DTL)

النوع: محرك القوالب الافتراضي في Django

اللغة: تعتمد على وسوم (tags) ومرشحات (filters) شبيهة بالـ HTML مع كود تحكمي محدود.

المميزات:

- آمن جدًا (يعمل Escaping تلقائي لمنع XSS).
- مدمج بالكامل مع Django (تكامل مع ORM، الفورمات، الـ CSRF).
- بسيط وسهل التعلم.
- يدعم الوراثة بين القوالب (Template Inheritance).

العيوب:

- محدود في العمليات البرمجية (لا يمكنك كتابة كود Python معقد داخله).
- أقل مرونة من محركات مثل Jinja2.

مثال استخدام:

```
<!-- base.html -->
<html>
endblock %} %}موقعي{% <head><title>{% block title

head>/><title/>
<body>
h1>/>{{ user.username }} مرحبا<    <h1
{%     {% block content %}{% endblock
<body/>
html>/>

<!-- home.html -->
{% extends 'base.html' %}
endblock %} %}الصفحة الرئيسية{% {% block title
{% block content %}
    {% for product in products %}
}} - {{         <p>{{ product.name
p>/>{{ product.price|floatformat:2
    {% empty %}
p>/>لا توجد منتجات<        <p
    {% endfor %}
{% endblock %}
```

شيء جميل:

- يمكنك استخدام **المرشحات الجاهزة**:

```
d/m/Y" }}": {{ date|date
```

- وتستطيع إنشاء **مرشحاتك الخاصة** في Python.

2. Jinja2

النوع: محرك قوالب عالي الأداء، يستعمل في Flask و يمكن استخدامه مع Django أيضًا.

اللغة: شبيهة جدًا بـ DTL لكن أقوى في كتابة المنطق.

المميزات:

- أسرع من DTL.
- يدعم كتابة تعبيرات Python مباشرة.
- مرن جدًا (Loops متقدمة، Macros، Expressions معقدة).

العيوب:

- أقل أمانًا إذا لم تستخدم Autoescaping.
- مرونته العالية قد تؤدي إلى خلط الكثير من المنطق داخل القوالب.

مثال استخدام:

```
<!DOCTYPE html>
<html>
<body>
h1>/>{{ user.name|upper }} مرحباً<    <h1
    {% for item in items if item.stock > 0 %}
(2f"|format(item.price.%" }} - {{         <p>{{ item.name
p>/>{{
    {% else %}

p>/>لا يوجد منتجات متاحة<p>
    {% endfor %}
```

شيء جميل:

- يمكنك إنشاء **Macros** لإعادة استخدام القوالب:

```
{% macro render_product(p) %}
div>/>{{ p.price }} - {{     <div>{{ p.name
{% endmacro %}

{{ render_product(product) }}
```

3. Mako

النوع: محرك قوالب سريع يدعم Python داخل القالب مباشرة.

اللغة: Python مدموج مع HTML.

المميزات:

- أداء عالي جدًا.
- حرية كاملة في كتابة Python.
- مناسب إذا أردت منطق معقد في القوالب.

العيوب:

- الحرية الكبيرة قد تؤدي إلى خلط منطق الأعمال مع العرض.
- أقل أمانًا من DTL (تحتاج إدارة الـ Escaping يدويًا).

مثال استخدام:

```
<%!
:(    def format_price(p
return f"${p:.2f}"
%>
<html>
<body>
:% for product in products
- {p>${product.name>
p>/>{(format_price(product.price}$
% endfor
<body/>
html>/>
```

شيء جميل:

- يمكنك كتابة **Functions** مباشرة في القالب.
- تستطيع حتى استدعاء مكتبات Python داخل القالب.

4. Mustache

النوع: محرك قوالب بسيط جدًا يعتمد على منطق العرض فقط، بدون كود برمجي.

اللغة: HTML + placeholders.

المميزات:

- بسيط جدًا (Logic-less).
- نفس الصيغة يمكن استخدامها في عدة لغات (JavaScript، Python...).
- يمنع خلط المنطق بالعرض تمامًا.

العيوب:

- لا يمكنك كتابة أي منطق أو شروط أو حلقات معقدة داخله.
- تحتاج تجهيز البيانات بالكامل قبل تمريرها للقالب.

مثال استخدام:

```
h1>/>{{name}} مـرحـبـاً<<h1
<ul>
  {{#products}}
<li/>{{price}} - {{    <li>{{name
products}}/}}
  {{^products}}
<li/>لا يـوجد مـنـتـجـات<    <li
{{products/}}
ul>/>
```

شيء جميل:

- يمكنك استخدام نفس القالب في **الفرونت إند** و **الباك إند** لأن الصيغة واحدة.
- يدعم التكرار الشرطي بطريقة أنيقة جدًا.

جدول المقارنة

المحرك	الأداء	درجة الأمان	سهولة التعلم	المرونة	التكامل مع Django
DTL	جيد	عالي جدًا	سهل	متوسط	ممتاز
Jinja2	ممتاز	جيد	متوسط	عالي	جيد
Mako	ممتاز	متوسط	متوسط	عالي جدًا	ضعيف
Mustache	جيد	عالي	سهل جدًا	ضعيف	ضعيف