SOFTWARE ENGINEERING LAB3 ASSIGNMENT

prof. Malik AL-Mosanif

ABDULRAHMAN AMEEN AHMED MANSOUR

INFORMATION TECHNOLOGY 4TH YEAR

Group B

1- Comparison of **4 template engines** other than Django Template Language (DTL) with their **setup instructions in settings.py:**

1. Jinja2

Overview:

- Jinja2 is a **fast, modern, and designer-friendly** template engine.
- Very popular with Flask.
- Similar syntax to DTL but **more powerful and flexible**.

Installation:

pip install Jinja2

Configuration in settings.py:

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.jinja2.Jinja2',
        'DIRS': [BASE_DIR / 'jinja2_templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'environment': 'myproject.jinja2.environment',
        },
    },
]
```

Also create jinja2.py in your project (e.g. myproject/jinja2.py):

```
from jinja2 import Environment
from django.contrib.staticfiles.storage import staticfiles_storage
from django.urls import reverse

def environment(**options):
    env = Environment(**options)
    env.globals.update({
        'static': staticfiles_storage.url,
        'url': reverse,
    })
    return env
```

2. Chemeleon

Overview:

- XML-based template engine similar to **Zope Page Templates (ZPT)**.
- Focuses on **speed and reusability**.
- More **Pythonic**, but less flexible than Jinja2.

Installation:

pip install Chameleon

Configuration in `settings.py`:

```
TEMPLATES = [
    {
        'BACKEND': 'django_chameleon.backend.Chameleon',
        'DIRS': [BASE_DIR / 'chameleon_templates'],
        'APP_DIRS': True,
        'OPTIONS': {},
    },
]
```

We also need to install `django-chameleon`:

pip install django-chameleon

3. Cheetah

Overview:

- Python-powered, older but **highly expressive** template engine.
- Allows embedding actual Python code.
- Less secure and modern than Jinja2 or Chameleon.

Installation:

pip install Cheetah3

Configuration:

There is **no official Django backend** for Cheetah, but We can integrate it manually using custom template loaders.

We would typically subclass `django.template.backends.base.BaseEngine` or integrate Cheetah in a middleware/view.

Example (custom use):

```
from Cheetah.Template import Template
def render_cheetah(template_str, context):
    return str(Template(template_str, searchList=[context]))
```

Use case:

Useful only in legacy projects or when you need Python logic deeply embedded in templates.

4. Mako

Overview:

- Inspired by **JSP, Django templates**, and **Python code**.
- **Powerful and fast**, allows complex Python code in templates.
- More complex than Jinja2, not beginner-friendly.

Installation:

pip install Mako

Configuration in `settings.py`:

```
TEMPLATES = [
    {
        'BACKEND': 'django_mako_plus.MakoTemplates',
        'DIRS': [BASE_DIR / 'mako_templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                # other processors
            ],
        },
    },
]
```

Also install:

pip install django-mako-plus

Comparison Table:

Feature	Jinja2	Chameleon	Mako	Cheetah
Speed	High	Very High	High	Moderate
Syntax	Similar to DTL	XML-based	Pythonic	Python-embedded
Flexibility	Very High	Limited	High	Moderate
Community Support	Excellent	Limited	Medium	Low
Security	High	High	Needs care	Risky
Official Support in Django	Yes	Yes (via package)	Yes (via DMP)	No

2- Filters in Django Comparison and Examples:

1. Django Template Filters

Template filters are used in HTML templates using the `{{ variable|filter }}` syntax.

Common Template Filters with Examples and Comparisons

Filter	Usage	Example	Output	Comparison/Note
date	Formats a date	user.joined	date:"Y-m-d"	`2025-08-06`
length	Returns length of list or string	items	length	`3`
default	Uses default if value is None or empty	username	default:"Guest"	`Guest`
upper	Converts to uppercase	name	upper	`ABDULRAHMAN`
lower	Converts to lowercase	name	lower	`abdulrahman`
truncatechars	Truncates to n chars	msg	truncatechars:10	`Hello worl…`
truncatewords	Truncates to n words	msg	truncatewords:3	`This is my…`
add	Adds a number	age	add:"5"	`30`
join	Joins list with separator	names	join:", "	`Ali, Sara, John`
safe	Marks string as safe HTML	html_code	safe	Renders HTML
escape	Escapes HTML characters	html_code	escape	`&lt;b&gt;Hi&lt;/b&gt;`
linebreaks	Converts newlines to `<p>` tags	text	linebreaks	Renders `<p>...</p>` blocks
filesizeformat	Converts bytes to readable size	file.size	filesizeformat	`10 KB`

2. Django QuerySet Filters

Used in views/models to query the database using `Model.objects.filter(...)`.

Common QuerySet Filters with Examples and Comparisons

Filter Lookup	Usage	Example	SQL Equivalent / Note
exact	Exact match	`User.objects.filter(age__exact=30)`	`= 30`
iexact	Case-insensitive match	`User.objects.filter(name__iexact="ali")`	`ILIKE 'ali'`
contains	Substring match	`User.objects.filter(name__contains="rah")`	`LIKE '%rah%'`
icontains	Case-insensitive contains	`User.objects.filter(name__icontains="rah")`	`ILIKE '%rah%'`
in	Value in list	`User.objects.filter(id__in=[1, 2, 3])`	`IN (1, 2, 3)`
gt, gte	Greater than / equal	`filter(age__gt=18)`	`> 18`
lt, lte	Less than / equal	`filter(age__lte=25)`	`<= 25`
startswith	Prefix match	`filter(name__startswith="A")`	`LIKE 'A%'`
istartswith	Case-insensitive	`filter(name__istartswith="a")`	`ILIKE 'a%'`
endswith	Suffix match	`filter(email__endswith=".com")`	`LIKE '%.com'`
iendswith	Case-insensitive	`filter(email__iendswith=".com")`	`ILIKE '%.com'`
range	Within range	`filter(age__range=(20, 30))`	`BETWEEN 20 AND 30`
date	Filter by date part	`filter(joined__date='2025-08-06')`	Extract date from datetime
year, month, day	Filter by components	`filter(joined__year=2025)`	
isnull	Null check	`filter(last_login__isnull=True)`	`IS NULL`
regex, iregex	Pattern matching	`filter(name__regex=r"^A.*")`	Regexp

3- My Custom Filter:

custom_filters.py

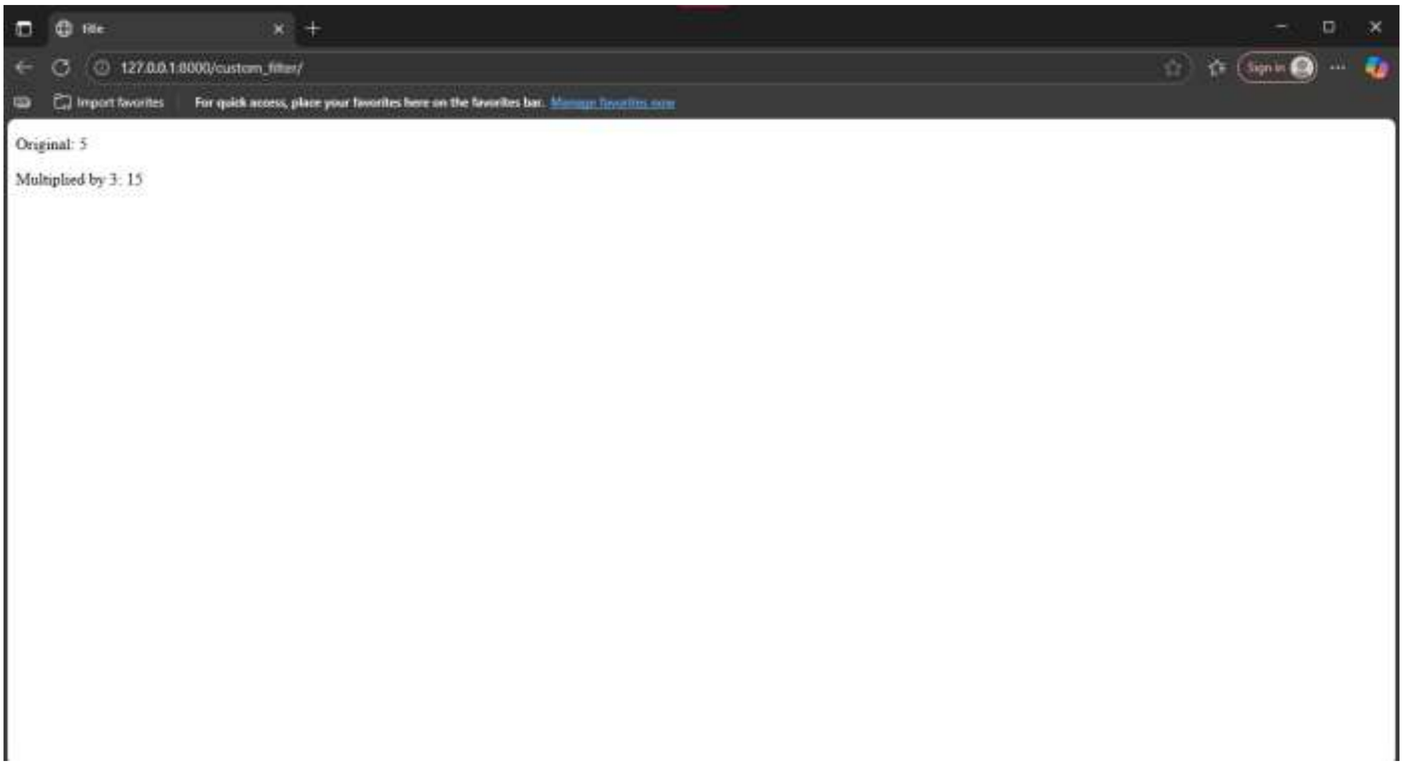
```python
from django import template

register = template.Library()

@register.filter(name='multiply')
def multiply(value, arg):
    try:
        return int(value) * int(arg)
    except (ValueError, TypeError):
        return ''
```

test_my_custom_filters.html

```html
{% load custom_filters %}
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <title>title</title>
</head>
<body>
    <p>Original: {{ 5 }}</p>
    <p>Multiplied by 3: {{ 5|multiply:3 }}</p>
</body>
</html>
```


title
127.0.0.1:8000/custom_filter/
Import favorites
For quick access, place your favorites here on the favorites bar. Manage favorites now
Sign in

Original: 5

Multiplied by 3: 15