Django Web Framework

Template Tags

In the previous chapter we discussed about Templates and template variable.

Now we are going to discuss about What is template tags?

Template Tags-

Tags provide logic in the template. They are enclosed in **{% %}** and can perform a variety of functions such as loops, conditional statements, including other templates, and more.

Example- Suppose we want to display the name of the user at the top right corner once user is logged in but if not logged in then a login button should be there, we can do it by using template tags or we want to display a success message once user clicks the submit button.

Practical 1:

Displaying list of subjects using template tags.

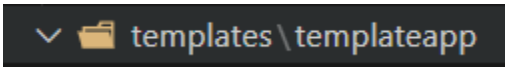
Step 1: Create new Django project and application. Add the app in settings.py.

```
C:\Users\LENOVO\Documents\djangocourse>django-admin startproject templatetags1

C:\Users\LENOVO\Documents\djangocourse>cd templatetags1

C:\Users\LENOVO\Documents\djangocourse\templatetags1>python manage.py startapp templateapp
```

Step 2: Create template folder inside the root folder and add the directory in settings.py.



```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR,'templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

Step 3: Create a view and HTML file.

templateapp > views.py > ...

```
from django.shortcuts import render

# Create your views here.
def display_view(request):
    subjects=['Java','Networking','Python','Web development']
    name='Harsh Trivedi'
    course='MCA'
    year='Final year'
    context={
        'subjects':subjects,
        'name':name,
        'course':course,
        'year':year,
    }
    return render(request,'templateapp/show.html',context)

```

Here we have created subjects list that contain different types of subject.

Now we are going to use template tags to display it on our template/web page.

templates > templateapp > show.html > html > body > div.data > ul

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Template tags</title>
</head>
<body>
    <div class="data">
        <h2>Name: {{name}}</h2>
        <h2>Course: {{course}}</h2>
        <h2>Year: {{year}}</h2>
        <h2>Subjects:</h2>
        <ul>
            {%for subject in subjects%}
                <li> {{subject}} </li>
            {%endfor%}
        </ul>>
    </div>
</body>
</html>
```

Here we have used template variable {{}} as well as template tags {%%}

and by using for loop we have displayed all the subjects.

Template tags allow you to control the flow of logic, iterate over data, and perform various tasks within your template.

Step 4: Map URLs for our view-

Application side URL-

templateapp > urls.py > ...

```
from django.urls import path
from templateapp import views

urlpatterns=[
    path('',views.display_view),
]
```

Project Level URL-

```python
from django.contrib import admin
from django.urls import path,include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('',include('templateapp.urls'))
]

```

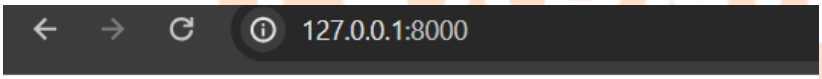
Step 5: Start the development server and visit the URL-

```
C:\Users\LENOVO\Documents\djangocourse\templatetags1>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin,
 auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
May 20, 2024 - 21:34:05
Django version 5.0, using settings 'templatetags1.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

[20/May/2024 21:34:18] "GET / HTTP/1.1" 200 606
Not Found: /favicon.ico
[20/May/2024 21:34:21] "GET /favicon.ico HTTP/1.1" 404 2264
```



Name: Harsh Trivedi

Course: MCA

Year: Final year

Subjects:

- Java
- Networking
- Python
- Web development

Practical 2:

In the same project or you can create a new one, we are going to see another example of template tags. In this practical we will see the use of if condition.

We are going to check if a person is eligible to vote or not.

Step 1: Create a new view function and new template-

```python
def Eligible_to_vote_view(request):
    age=23
    return render(request,'templateapp/vote.html',context={'age':age,})
```

We can pass the data to context dictionary directly like this also.

Create a new template-

templates > templateapp > vote.html > html > body

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Check for vote</title>
</head>
<body>
    <h2>Eligible for vote or not?</h2> <br/>
    {%if age >= 18%}
        You are eligible to vote
    {%else%}
        You are not eligible to vote
    {%endif%}
</body>
</html>
```

Step 2: Update application level urls.py-

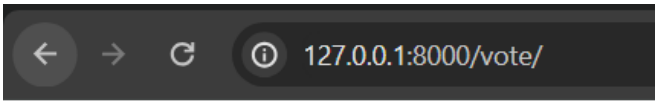
templateapp > urls.py > ...

```python
from django.urls import path
from templateapp import views

urlpatterns=[
    path('',views.display_view),
    path('vote/',views.Eligible_to_vote_view)
]
```

Step 3: Run the development server and visit the url-

```
[20/May/2024 21:56:41] "GET /vote/ HTTP/1.1" 200 343
[20/May/2024 21:56:57] "GET /vote/ HTTP/1.1" 200 294
```



Eligible for vote or not?

You are eligible to vote

Template tags keep presentation logic separate from template code, promoting cleaner and more maintainable templates.

Template tags are really important and there are many use cases of template tags.

Template tags are also used for relative URLs, to load and use static content on our templates. We will learn about this in upcoming chapters.

The next topic we are going to learn is static files.

We will learn about template filters and template inheritance when we cover advance template features topic.

If you are following the course series and learning the concepts easily, please subscribe to our YouTube channel. This will motivate us to create more educational, course videos and study material.

Join our growing community of tech enthusiasts! Subscribe to our YouTube channel, where we simplify programming languages in the easiest way possible. Gain a deeper understanding with our clear explanations and receive exclusive notes to enhance your learning journey. Don't miss out on valuable insights – hit that subscribe button now and embark on a programming adventure with us!

Subscribe to our channel:
https://www.youtube.com/@HTEASYLEARNING