
Django Learning Report

Name : Prajwal Bashyal
Institution: Bhaktapur Multiple Campus, BIT
Date : 2025-12-18
Topic : Django – Templates (complete guide)

Django Templates – Complete Guide (Project-level / Parallel to Apps)

This document explains **Django templates from the very beginning**, assuming the **templates folder is parallel to apps (project-level templates)**. The content is suitable for:

- Django beginners
- Revision and quick review
- Self-study and guided learning

---

1. What are Templates in Django?

In Django, **templates are HTML files** used to define **how data is displayed to users**.

Django follows the **MVT (Model–View–Template)** architecture:

- **Model** → handles database and data structure
- **View** → handles logic and processing (Python)
- **Template** → handles presentation (HTML)

Templates ensure that **logic and design are kept separate**.

Example idea:

- View sends data → `{'name': 'Prajwal'}`
- Template displays it as → `Hello Prajwal`

---

2. Why Templates are Important

Without templates:

- HTML and Python logic mix together
- Code becomes hard to maintain

With templates:

- Clean separation of concerns
- Easier collaboration between backend and frontend
- Reusable and scalable code

---

3. Django Template Language (DTL)

Django uses **Django Template Language (DTL)** inside HTML files.

DTL allows:

- Printing variables → `{{ variable }}`
- Conditions → `{% if %}`
- Loops → `{% for %}`
- Template inheritance

Example:

```
<h1>{{ title }}</h1>
```

---

4. Templates Folder Structure (Parallel to Apps)

In this approach, the **templates folder is placed at the project level**, parallel to apps.

```
project_folder/
│
├── manage.py
├── project_name/
│   └── settings.py
│
├── blog/                # Django app
│   ├── views.py
│   └── models.py
│
└── templates/           # Global templates folder
    └── blog/
        ├── base.html
        ├── home.html
        └── post_detail.html
```

5. Why Use App Name Inside Templates Folder?

Even though templates are global, **each app must have its own subfolder**.

✅ Correct:

`templates/blog/home.html`

❌ Incorrect:

`templates/home.html`

Reasons:

- Prevents template name conflicts
- Improves readability
- Makes large projects manageable

---

6. Configuring Templates in `settings.py`

Since templates are **outside apps**, Django must be told where to find them.

Step 1: BASE_DIR (default)

```
from pathlib import Path
BASE_DIR = Path(__file__).resolve().parent.parent
```

`BASE_DIR = project root folder`

Step 2: TEMPLATES Configuration

```
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',
            ],
        },
    },
]
```

Explanation:

- `DIRS` → path to global templates folder
- `APP_DIRS = True` → allows app-level templates if needed

---

7. Creating a Template File

Example: `templates/blog/home.html`

```
<!DOCTYPE html>
<html>
<head>
    <title>Home</title>
</head>
<body>
    <h1>Welcome to Django</h1>
</body>
</html>
```

8. What is Rendering in Django?

Rendering means converting a template into a final HTML page by combining:

- Template (HTML)
- Context data (Python)

This is done using Django's `render()` function.

---

9. The `render()` Function

Syntax:

```
render(request, template_name, context=None)
```

Example View

```python
from django.shortcuts import render

def home(request):
    return render(request, 'blog/home.html')
```

What Django Does Internally:

1. Locates the template using `DIRS`
2. Injects context data
3. Converts it to pure HTML
4. Sends it to the browser

---

10. Passing Data from View to Template

View:

```python
def home(request):
    context = {
        'title': 'Home Page',
        'author': 'Prajwal'
    }
    return render(request, 'blog/home.html', context)
```

Template:

```
<h1>{{ title }}</h1>
<p>Author: {{ author }}</p>
```

11. Template Tags

If Condition

```
{% if user.is_authenticated %}
<p>Welcome {{ user.username }}</p>
{% else %}
<p>Please login</p>
{% endif %}
```

For Loop

```
{% for post in posts %}
<h2>{{ post.title }}</h2>
{% endfor %}
```

---

12. Template Inheritance

Template inheritance avoids repeating HTML.

`base.html`

```
<!DOCTYPE html>
<html>
<body>
<header>My Blog</header>

{% block content %}{% endblock %}

<footer>Footer</footer>
</body>
</html>
```

`home.html`

```
{% extends 'blog/base.html' %}

{% block content %}
<h1>Home Page</h1>
{% endblock %}
```

---

13. Static Files in Templates

```
{% load static %}
<link rel="stylesheet" href="{% static 'css/style.css' %}">
```

---

14. Common Template Errors

Error	Cause
TemplateDoesNotExist	Wrong path or `DIRS` missing
Invalid block tag	Missing `{% load %}`
Variable not showing	Context not passed

15. Use of Templates in Django Blog Website

Templates are used for:

- Home page
- Post list page
- Post detail page
- Create / edit post
- Dashboard

Each page corresponds to a **template rendered by a view**.

---

16. Final Summary

- Templates define the **user interface**
- Placed in a **project-level templates folder**
- Configured using `DIRS` in `settings.py`
- `render()` combines template and data
