Python

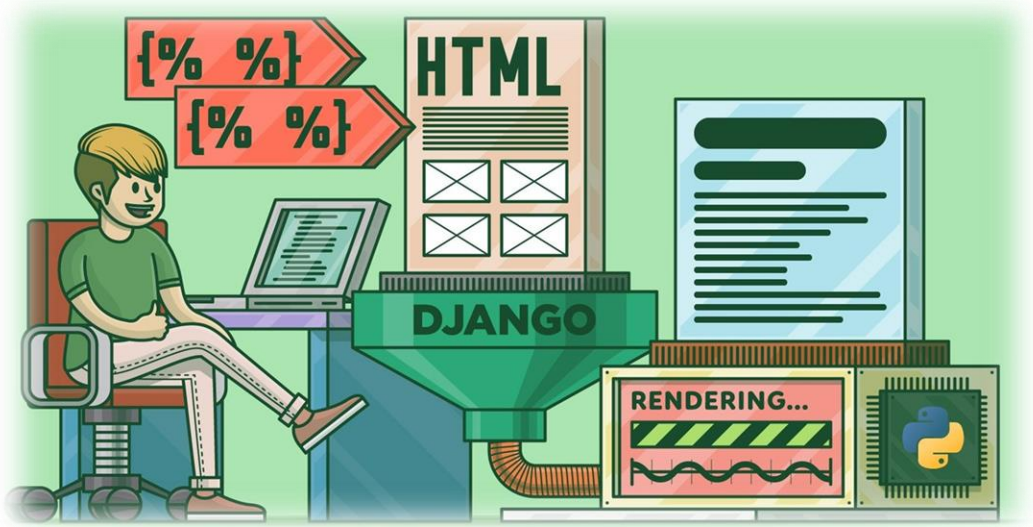
Full stack Skills Bootcamp

Introduction to Django Templates

What is a Template ?

Django templates allow you to create dynamic HTML pages that display content from your views.

- They enable separation of logic (in views) and presentation (in templates).
- Templates use the Django Template Language (DTL) to insert data from the view.

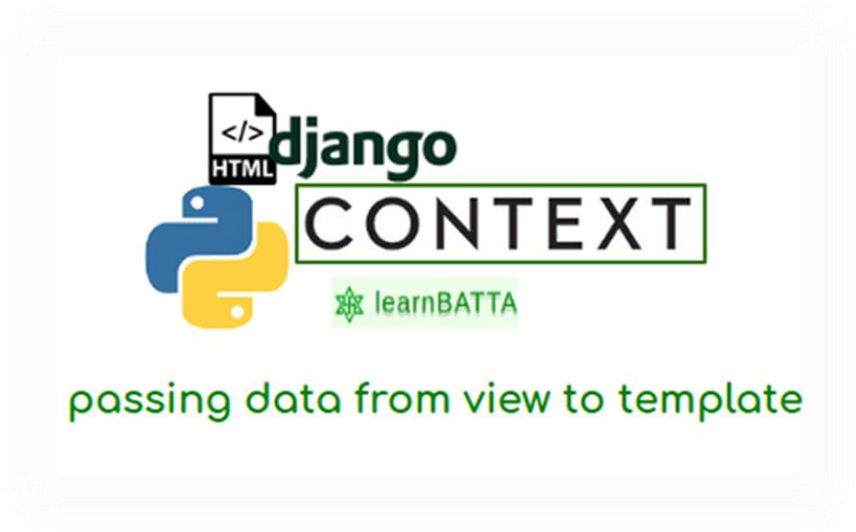


Passing Data from View to Template

- You can pass data from the view to the template using a context dictionary.
- This data can be displayed dynamically using Django template syntax..

```python
from django.shortcuts import render

def my_view(request):
    context = {'name': 'Nikhith'}
    return render(request, 'index.html', context)
```



Rendering Data in Templates

- Django uses double curly braces to output variables in the template.
- For example, to display the value of a variable like this: {{ variable_name }}.
- The template language also supports loops, conditionals, and filters.

```html
<h1>Hello, {{ name }}</h1>
```

- With the context {'name': 'Nikhith'}, this will render as:

```html
<h1>Hello, Nikhith</h1>
```

Template Inheritance

- Django templates support inheritance, allowing you to reuse common HTML elements.
- You can define a base template with common elements (e.g., header, footer) and extend it in other templates.
- This allows you to keep your code DRY (Don't Repeat Yourself).

```html
<html>
<head><title>{% block title %}My Site{% endblock %}</title></head>
<body>
  <header>Header content</header>
  <div>{% block content %}{% endblock %}</div>
  <footer>Footer content</footer>
</body>
</html>
```

```html
{% extends 'base.html' %}
{% block content %}
  <h1>Welcome to My Site</h1>
{% endblock %}
```

Template Tags and Filters

- Django templates support various tags (e.g., {% for %}, {% if %}) for logic control.
- Filters can be used to modify values before displaying them, like converting to uppercase or formatting dates.

```html
{% for item in items %}
  <p>{{ item }}</p>
{% endfor %}
```

```html
<p>{{ name|upper }}</p>  <!-- Renders as 'NIKHITH' -->
```

Folder Structure for Bootstrap

Organize your project by creating a dedicated folder for static files.

```
exampleproject/
├── mysite/
│   ├── static/
│   │   └── css/
│   │       └── bootstrap.min.css
│   ├── templates/
│   │   └── base.html
└── manage.py
```

Adding Bootstrap CSS

You can use a CDN or download Bootstrap and place it in the static/css/ directory.

```html
<link href="{% static 'css/bootstrap.min.css' %}" rel="stylesheet">
```

Ensure you have the django.contrib.staticfiles app in your settings.py.

Configuring settings.py for Static Files

Ensure STATIC_URL is set in settings.py:

```python
STATIC_URL = '/static/'
```

- Optionally define STATICFILES_DIRS for additional static file locations:

```python
STATICFILES_DIRS = [
    BASE_DIR / "static",
]
```

Using Template Inheritance with Bootstrap

Use Django's template inheritance to create a base layout..

- In base.html we can inherit the static css files:

- Extend this base template in other HTML files.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <link href="{% static 'css/bootstrap.min.css' %}" rel="stylesheet">
    <title>{% block title %}My Site{% endblock %}</title>
</head>
<body>
    {% block content %}{% endblock %}
</body>
</html>
```