APLICACIONES WEB

SALAS FIGUEROA JESUS NAHATAEN

Universidad Tecnológica de Tijuana.
TSU. Tecnologías de la información.
Entornos virtuales y negocios digitales.
Docente: DR. Parra Galviz Ray Burnet.
Email: 0322103855@ut-tijuana.edu.mx

I. INTRODUCCIÓN

IN this document I will cover a series of activities about django templates and we will see all the information

TEMPLATES DJANGO SYSTEM

Being a web framework, Django needs a convenient way to generate HTML dynamically. The most common approach relies on templates. A template contains the static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted. For a hands-on example of creating HTML pages with templates, see Tutorial 3.

A Django project can be configured with one or several template engines (or even zero if you don't use templates). Django ships built-in backends for its own template system, creatively called the Django template language (DTL), and for the popular alternative Jinja2. Backends for other template languages may be available from third-parties. You can also write your own custom backend, see Custom template backend

Django defines a standard API for loading and rendering templates regardless of the backend. Loading consists of finding the template for a given identifier and preprocessing it, usually compiling it to an in-memory representation. Rendering means interpolating the template with context data and returning the resulting string. [1]

The Django template language is Django's own template system. Until Django 1.8 it was the only built-in option available. It's a good template library even though it's fairly opinionated and sports a few idiosyncrasies. If you don't have a pressing reason to choose another backend, you should use the DTL, especially if you're writing a pluggable application and you intend to distribute templates. Django's contrib apps that include templates, like django.contrib.admin, use the DTL. [1]

For historical reasons, both the generic support for template engines and the implementation of the Django template language live in the django.template namespace.

Syntax¶

A Django template is a text document or a Python string marked-up using the Django template language. Some constructs are recognized and interpreted by the template engine. The main ones are variables and tags. [1]

A template is rendered with a context. Rendering replaces variables with their values, which are looked up in the context, and executes tags. Everything else is output as is.

The syntax of the Django template language involves four constructs.

Variables

A variable outputs a value from the context, which is a dict-like object mapping keys to values.

Variables are surrounded by {{ and }} like this

Tags

Tags provide arbitrary logic in the rendering process.

This definition is deliberately vague. For example, a tag can output content, serve as a control structure e.g. an "if" statement or a "for" loop, grab content from a database, or even enable access to other template tags.

Conclusion

In summary, Django, as a web framework, uses templates to generate dynamic HTML content. These templates can be configured with different template engines, with Django's own template language (DTL) being the default option. Templates in Django involve variables surrounded by {{ and }}, which display values from the context, and tags that provide logic and control structures for rendering content. Using Django's built-in template system is recommended unless you have a specific reason to choose another backend. [1]

REFERENCES

[1] Django, «Django project,» [En línea]. Available: Django. (s. f.). Django Project. Recuperado 5 de noviembre de 2023, de https://docs.djangoproject.com/en/4.2/topics/templates/. [Último acceso: 05 11 2023].