Flask. Manejo de templates con Jinja2

Loreto Pelegrín Castillo

Jinja2

Hemos visto cómo utilizar la función render_templates() para el manejo por parte de Flask de las plantillas web. Esta función está proporcionada por el motor de renderizado de plantillas independiente Jinja2, que viene incluido con Flask y se instala al mismo tiempo que este.

La manera en que funciona Jinja2 es manipulando ficheros que contienen contenido estático (HTML normales) junto a bloques de código para generar contenido dinámico. Cuando se renderiza una platilla, lo que ocurre es que ese contejido dinamico se ha procesado y representado en un documento HTML.

Características de Jinja2

1. **Variables**

En Jinja2, las variables se encierran entre doble llaves: {{ variable }}. El valor de la variable se reemplazará por su contenido real al renderizar la plantilla.

```python
from flask import Flask,render_template

app = Flask(__name__)


@app.route('/')
@app.route('/index')
def hola():
    nombre = "Loreto"
    edad = 32
    return render_template('index.html',nombre= nombre,edad=edad)

if __name__ == "__main__":
    app.run()
```

```html
<!DOCTYPE html>
<html>
<head>
    <title>Mi página</title>
</head>
<body>
    <h1>¡Hola, {{ nombre }}!</h1>
    <p>Tienes {{ edad }} años.</p>
</body>
</html>
```

2. Filtros

Se aplican a las variables de la plantilla y realizan algún tipo de modificación en ellas. Para usarlos hay que usar el nombre de la variable dentro de los delimitadores y separarlos por el carácter '|' (tubería).

Filtro	Definicion
Title	Pone en mayúscula la primera letra de cada palabra de la cadena
capitalize	Pone en mayúscula el primer carácter de la cadena
default	Define una cadena predeterminada si la variable no esta definida, no sirve para cadenas vacías
upper / lower	Poner todos los caracteres de una cadena en mayúsculas / minúsculas
Dictsort	Nos ordena las claves de un diccionario
truncate(num)	Convierte la longitud del texto en el numero que se le pasa por parámetro a la función
Sort	Ordenar una lista
max / min	Valor máximo y mínimo de la lista

Se pueden crear filtros personalizados. Estructura:

@app.template_filter("NombreFuncion")

def función():

contenido

Por ejemplo:

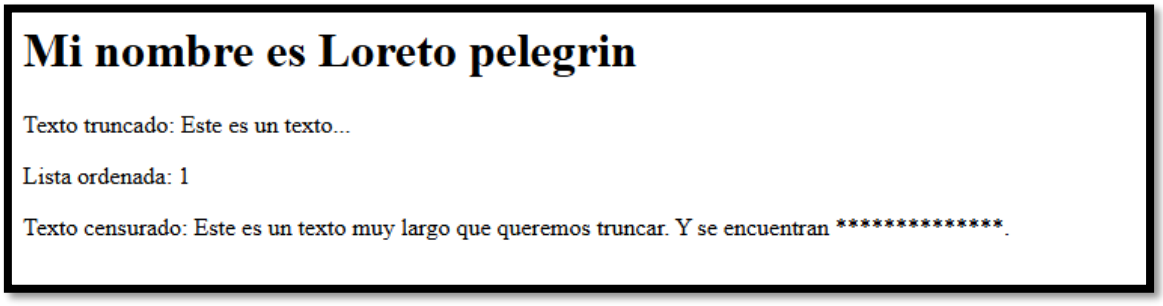
```
@app.template_filter('censurar')
def censurar_filter(s):
    palabras_censuradas = ['malas palabras', 'otra palabra']
    for palabra in palabras_censuradas:
        s = s.replace(palabra, '*' * len(palabra))
    return s
```

Código completo filtros:

```
from flask import Flask, render_template

app = Flask(__name__)

@app.template_filter('censurar')
def censurar_filter(s):
    palabras_censuradas = ['malas palabras', 'otra palabra']
    for palabra in palabras_censuradas:
        s = s.replace(palabra, '*' * len(palabra))
    return s
```

```python
@app.route('/')
def index():
    nombre = "Loreto Pelegrin"
    texto_largo = "Este es un texto muy largo que queremos truncar. Y se encuentran malas palabras."
    lista_de_numeros = [1, 5, 3, 9, 2]

    return render_template('index.html',
                           nombre=nombre,
                           texto_largo=texto_largo,
                           lista_de_numeros=lista_de_numeros)

if __name__ == '__main__':
    app.run(debug=True)
```

```html
<!DOCTYPE html>
<html>
<head>
    <title>Ejemplo de Filtros</title>
</head>
<body>
    <h1>Mi nombre es {{ nombre | capitalize }}</h1>
    <p>Texto truncado: {{ texto_largo | truncate(20) }}</p>
    <p>Lista ordenada: {{ lista_de_numeros | sort }}</p>
    <!-- En este último parrafo estamos llamando al filtro personalizado-->
    <p>Texto censurado: {{ texto_largo | censurar }}</p>
</body>
</html>
```



3. Sentencias if

Funciona de la misma manera que en Python, solamente que va en encerrada entre los delimitadores {% if condicion%} - *{% endif %}*

Estructura:

{% if condicion %}

{% elif %}

{% else %}

{% endif %}

Ejemplo:

```python
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def index():
    edad = 25
    return render_template('index.html', edad=edad)

if __name__ == '__main__':
    app.run(debug=True)
```

```html
<!DOCTYPE html>
<html>
<head>
    <title>Ejemplo de if</title>
</head>
<body>
    {% if edad < 18 %}
    <p>Eres menor de edad.</p>
    {% elif edad < 65 %}
    <p>Eres mayor de edad.</p>
    {% else %}
    <p>Eres una persona mayor.</p>
    {% endif %}
</body>
</html>
```

4. **Bucles for**

```python
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def index():
    frutas = ['manzana', 'banana', 'pera']
    return render_template('index.html', frutas=frutas)

if __name__ == '__main__':
    app.run(debug=True)
```

```html
<!DOCTYPE html>
<html>
<head>
<title>Ejemplo de For</title>
</head>
<body>
    <ul>
        {% for fruta in frutas %}
          <li>{{ fruta }}</li>
        {% endfor %}
      </ul>
</body>
</html>
```

Ejemplo con table:

```python
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def index():
    frutas = [
        {'nombre': 'manzana', 'color': 'red'},
        {'nombre': 'banana', 'color': 'yellow'},
        {'nombre': 'pera', 'color': 'green'}
    ]
    return render_template('index.html', frutas=frutas)

if __name__ == '__main__':
    app.run(debug=True)
```

```html
<!DOCTYPE html>
<html>
<head>
    <title>Ejemplo de for en Flask</title>
</head>
<body>
    <table border="1">
        <thead>
            <tr>
                <th>Fruta</th>
            </tr>
        </thead>
        <tbody>
            {% for fruta in frutas %}
                <tr style='background-color: {{ fruta.color }}'>
                    <td>{{ fruta.nombre }}</td>
                </tr>
            {% endfor %}
        </tbody>
    </table>
</body>
</html>
```

5. Herencia

Se utiliza en las platillas para compartir el contenido de una plantilla entre varias, con lo que nos ahorra repetir código.

Para usar herencia en el motor Jinja2, lo primero es usar {% extends “nombre_plantilla.html” %} dentro de las platillas que queremos heredar el código. Posteriormente, el código que sustituir por el código heredado debe estar dentro de los delimitadores {%block contect nombredelbloque %} y {%endblock}

```python
from flask import Flask,render_template

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/acerca')
def acerca():
    return render_template('acerca.html')
if __name__ == "__main__":
    app.run()
```

plantilla.html

```
<!DOCTYPE html>
<html>
<head>
    <title>{% block title %}{% endblock %}</title>
</head>
<body>
    <header>
        <h1>Mi sitio web</h1>
        <nav>
            <ul>
                <li><a href="/">Inicio</a></li>
                <li><a href="/acerca">Acerca de</a></li>
            </ul>
        </nav>
    </header>

    <div class="content">
        {% block content %}{% endblock %}
    </div>

</body>
</html>
```

Index.html

```
{% extends "plantilla.html" %}

{% block title %}INICIO{% endblock %}

{% block content %}
    <h2>Inicio</h2>
    <p>Página acerca Inicio</p>
{% endblock %}
```

Acerca.html

```
{% extends "plantilla.html" %}

{% block title %}Acerca de{% endblock %}

{% block content %}
    <h2>Acerca de nosotros</h2>
    <p>Esta es la página acerca de.</p>
{% endblock %}
```

Uso de ficheros estáticos con Flask

Si necesitamos usar ficheros de contenido estático, como ficheros CSS, en nuestra aplicación creada con Flask, deberemos crear un directorio que po convenio llamaremos "static" y que estará al mismo nivel que el directorio de templates creado previamente.

Para referenciarlo usaremos una declaración habitual de enlace en el head del HTML, dentro del cual usaremos la función url_for() entre llaves, teniendo en cuenta que primero debemos introducir el parámetro static y a continuación el parámetro filename con el nombre del fichero.

Ejemplo:

<link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">

Bootstrap

Documentación: https://getbootstrap.com/

Colores: https://getbootstrap.com/docs/5.3/customize/color/

Navbar: https://getbootstrap.com/docs/5.3/components/navbar/

Carrusel de imágenes: https://getbootstrap.com/docs/5.3/components/carousel/

Card: https://getbootstrap.com/docs/5.3/components/card/

Instalar Bootstrap

pip install Flask-Bootstrap

Ejemplo plantilla.html:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>{% block title %}Mi aplicación{% endblock %}</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min
.css" rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous">
    <link rel="stylesheet" href="{{ url_for('static',
filename='styles.css') }}">
</head>
```

```
<body>
    <header>
        <nav class="navbar navbar-expand-lg bg-body-tertiary">
            <div class="container-fluid">
              <a class="navbar-brand" href="#">Navbar</a>
              <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav"
aria-expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
              </button>
              <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav">
                  <li class="nav-item">
                    <a class="nav-link active" aria-current="page"
href="/">Home</a>
                  </li>
                  <li class="nav-item">
                    <a class="nav-link" href="acerca">Acerca de</a>
                  </li>
                </ul>
              </div>
            </div>
          </nav>
    </header>
    <div class="container">
        {% block content %}{% endblock %}
    </div>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundl
e.min.js" integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
crossorigin="anonymous"></script>
</body>
</html>
```