Plantillas III.

En este modulo veremos como llamar métodos desde plantillas, la jerarquía u orden en llamadas desde plantillas, el uso de listas en contexto y plantillas y estructuras de control de flujo en plantillas.

from django.http import HttpResponse

from django.template import Template, Context

import datetime

class Persona(object):

    def \_\_init\_\_(self, nombre, apellido):

        self.**nombre** = nombre

        self.**apellido** = apellido

    def saludo(request):

        p1 = Persona("Valeria", "Rivero")

*# nombre = "Alexis"*

*# apellido = "Reynoso"*

        fecha\_actual = datetime.datetime.now()

        doc\_externo = open("C:/Users/Alexis/Desktop/Programacion/Backend/Frameworks/Django/Archivos Django/ProyectoDjango5/ProyectoDjango5/templates/index.html")

        plt = Template(doc\_externo.read())

        doc\_externo.close()

        ctx = Context({

            'nombre\_persona' : p1.**nombre**,

            'apellido\_persona' : p1.**apellido**,

            'fecha' : fecha\_actual,

        })

        documento = plt.render(ctx)

        return HttpResponse(documento)

Para este modulo usaremos este codigo Python.

<!DOCTYPE html>

<html *lang*="en">

<head>

    <meta *charset*="UTF-8">

    <meta *name*="viewport" *content*="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <h2 *style*="color: blueviolet;">

        Hola mundo de django, esta es mi primera pagina hmtl con django

    </h2>

    <p>

        El nombre del profesor es: {{nombre\_persona}} {{apellido\_persona}}

    </p>

    <p>

        La fecha y hora actual es: {{fecha.day}}/{{fecha.month}}/{{fecha.year}}

    </p>

</body>

</html>

Y este será nuestra plantilla html.

Ahora comenzaremos a ver como usar una lista para nuestro plantilla, para ello debemos de agregarlas dentro del diccionario.

        ctx = Context({

            'nombre\_persona' : p1.**nombre**,

            'apellido\_persona' : p1.**apellido**,

            'fecha' : fecha\_actual,

            "temas" : ["Plantillas", "Modelos", "Formularios", "Vistas", "BBDD", "Despliegue aplicación"]

        })

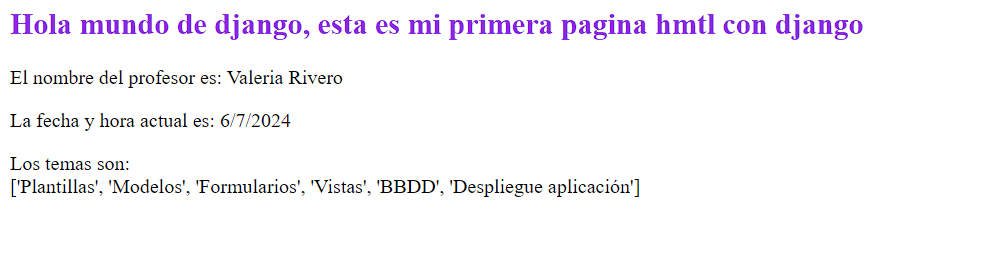
Ahora procederemos a rescatar la lista desde nuestra plantilla.

    <p>

        Los temas son:<br>

        {{temas}}

    </p>



Esto mostrara en pantalla la lista con su mismo formato, si nosotros quisiéramos acceder a los elementos individuales de la lista usaremos la nomenclatura del punto y los índices de la lista.

    <p>

        Los temas son:<br>

        {{temas.0}}<br>

        {{temas.1}}<br>

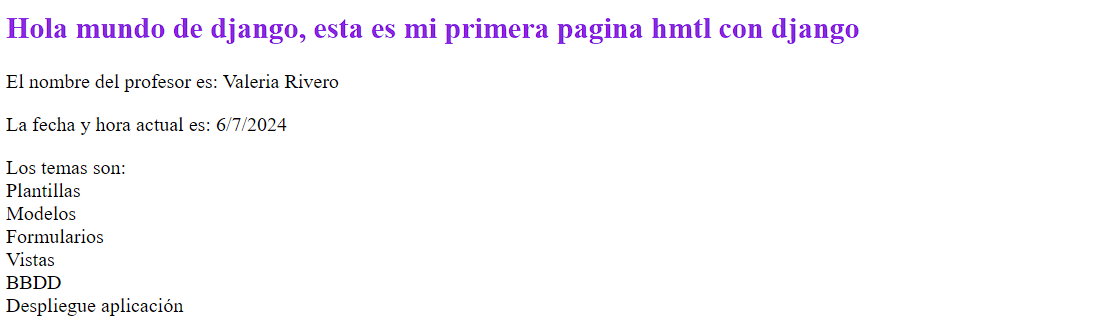
        {{temas.2}}<br>

        {{temas.3}}<br>

        {{temas.4}}<br>

        {{temas.5}}<br>

    </p>



O podemos meter la lista dentro de una variable y despues enlazar la variable en el diccionario.

Como vimos en el módulo anterior y en este podemos usar la nomenclatura para trabajar con varias cosas, para este caso nos convendría saber cuando django esta en una plantilla con un punto cual es el orden a seguir, por ejemplo:

    <p>

        Los temas son:<br>

        {{temas.0}}<br>

        {{temas.1}}<br>

        {{temas.2}}<br>

        {{temas.3}}<br>

        {{temas.4}}<br>

        {{temas.5}}<br>

    </p>

Aquí internamente cual es el orden que sigue django para hacer la llamada.

Bueno, lo primero que pasa cuando se encuentra con un punto django lo primero que hace es buscar un diccionario para ver si ese punto corresponde a un elemento de un diccionario, despues ve si ese punto pertenece a un atributo o propiedad correspondiente a una clase, si no corresponde a ninguno pasa a ver si corresponde a un método perteneciente a un método o clase, y por ultimo analiza si ese punto pertenece a un índice de una lista.

Este es el orden que sigue django al usar la nomenclatura del punto.

Ahora la manera que vimos antes de mostrar todos los temas en pantallas es una de las tantas maneras de hacerlo, pero recordar que cuando estamos programando si queremos recorrer elementos en este caso de una lista, haremos uso de un bucle for.

*{% comment %} <p>*

*Los temas son:<br>*

*{{temas.0}}<br>*

*{{temas.1}}<br>*

*{{temas.2}}<br>*

*{{temas.3}}<br>*

*{{temas.4}}<br>*

*{{temas.5}}<br>*

*</p> {% endcomment %}*

NOTA: ASI SE HACE UN COMENTARIO CON DJANGO.

Ahora lo primero que debemos hacer para usar un bucle es saber que los bucles van dentro de una sola llave simple. Pero a diferencia de un programa normal debemos especificar donde comienza el bucle y donde termina. De esta manera.

    <p>

        {% for %}

        {% endfor %}

    </p>

Y ahora debemos de completer ese bucle for recorriendo los elementos de la lista.

    <p>

        Los temas del curso son:

        {% for tema in temas %}

            {{tema}}<br>

        {% endfor %}

    </p>

Siendo tema lo que se imprimirá en cada vuelta del bucle, y temas la clave que le asignamos a la lista dentro de nuestro diccionario

    <ol>

        Los temas del curso son:

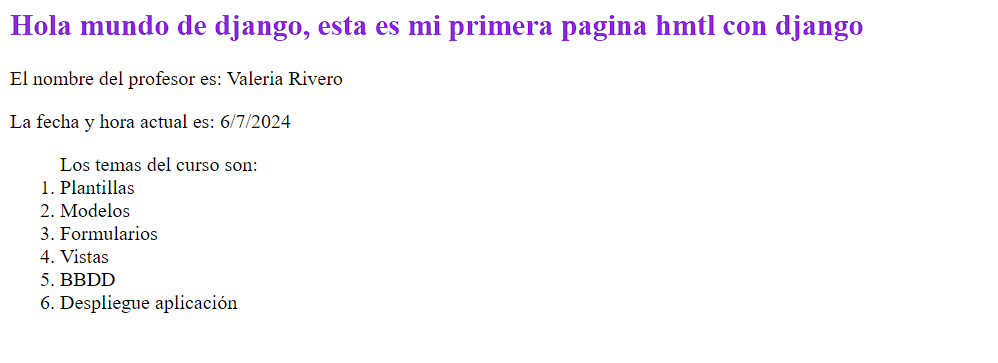
        {% for tema in temas %}

            <li>{{tema}}</li>

        {% endfor %}

    </ol>

De esta manera lo estaríamos creando como si fuera una lista ordenada.



Igual que usamos bucles tambien podemos usar estructuras condicionales.

Ahora para ver esto quitemos los elementos de la lista y quedémonos con una lista vacia.

        ctx = Context({

            'nombre\_persona' : p1.**nombre**,

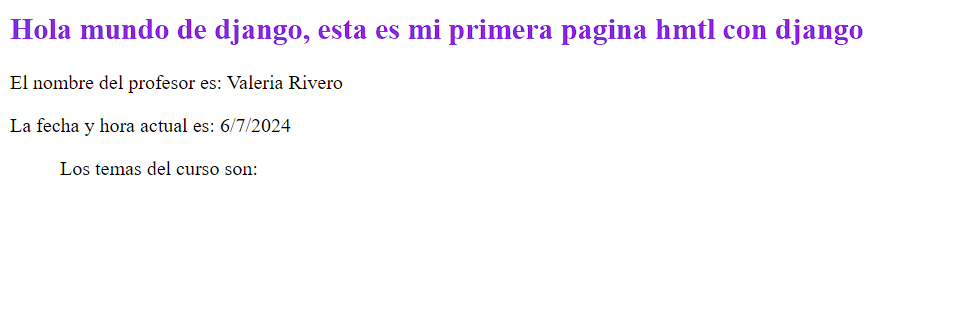
            'apellido\_persona' : p1.**apellido**,

            'fecha' : fecha\_actual,

            'temas' : []

        })

En este caso al no haber ningún elemento no se usará el bucle for y en la plantilla no mostrará nada.



Para trabajar con condicionales seguiremos la misma estructura que la del bucle for:

*{% comment "" %}Esto es la manera de crear un bucle for con django{% endcomment %}*

*{% comment "" %}Encerramos a este bucle for dentro de un if{% endcomment %}*

    <ol>

        Los temas del curso son:

        {% if  %}

        {% for tema in temas %}

            <li>{{tema}}</li>

        {% endfor %}

        {% endif %}

    </ol>

Recordando que estamos trabajando en html, o sea debe como abrimos el if antes que el for este lo encerrara y cerrara despues que el.

*{% comment "" %}Esto es la manera de crear un bucle for con django{% endcomment %}*

*{% comment "" %}Encerramos a este bucle for dentro de un if{% endcomment %}*

    <ol>

        Los temas del curso son:

        {% if  temas%}

            {% for tema in temas %}

                <li>{{tema}}</li>

            {% endfor %}

        {% endif %}

    </ol>

Aquí lo que queremos es evaluar si temas existe y que ademas de existir tiene informacion, si existe pero no tiene informacion no entra en el if lo mismo pasa si no existe.

Ahora borraremos los elementos de la lista temas.

        ctx = Context({

            'nombre\_persona' : p1.**nombre**,

            'apellido\_persona' : p1.**apellido**,

            'fecha' : fecha\_actual,

            'temas' : [] *#"Plantillas", "Modelos", "Formularios", "Vistas", "BBDD", "Despliegue aplicación"*

        })

*{% comment "" %}Esto es la manera de crear un bucle for con django{% endcomment %}*

*{% comment "" %}Encerramos a este bucle for dentro de un if{% endcomment %}*

*{% comment "" %}Antes de terminar el if agregamos un else{% endcomment %}*

    <ol>

        Los temas del curso son:

        {% if  temas%}

            {% for tema in temas %}

                <li>{{tema}}</li>

            {% endfor %}

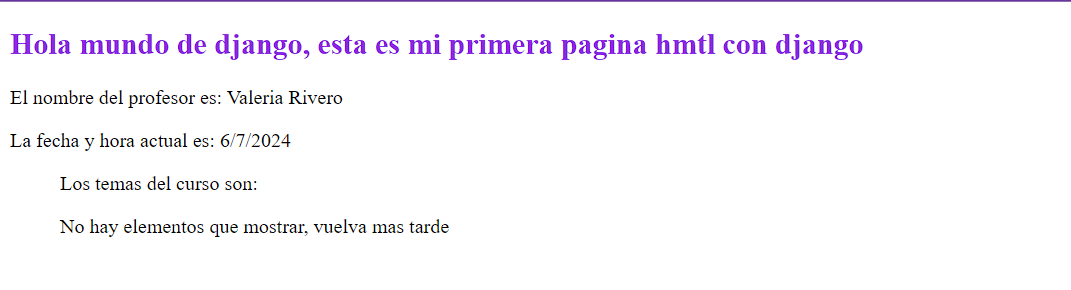
        {% else %}

            <p>No hay elementos que mostrar, vuelva mas tarde</p>

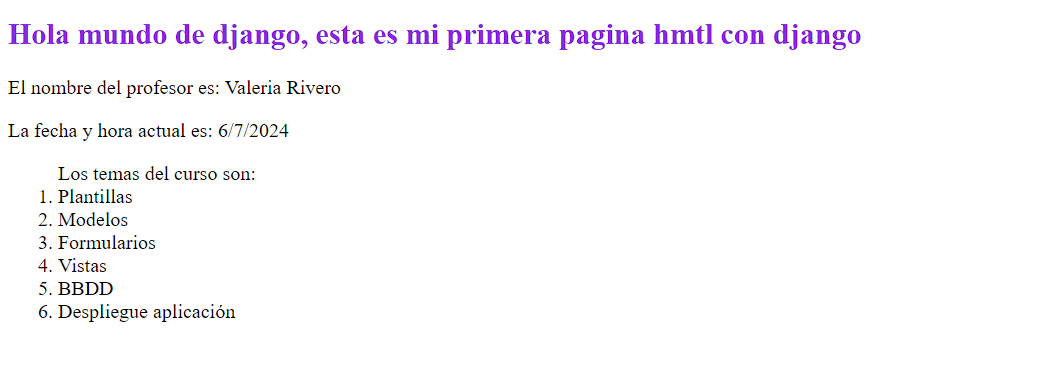
        {% endif %}

    </ol>

Ahora en este caso, como existe temas pero no tiene informacion aplicara el else y nos mostrara un párrafo que indica que no hay elementos.



Y si agregamos los elementos de nuevo nos realizara el for:



NOTA: LOS MÉTODOS LLAMADOS DESDE UNA PLANTILLA NO LLEVAN LOS PARENTESIS COMO NORMALMENTE SERIAN.