Plantillas IV.

En este modulo veremos como usar condicionales en las plantillas, que son los filtros y que son los cargadores de plantillas.

Para este módulo trabajaremos con este archivo python

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/ProyectoDjango6/ProyectoDjango6/templates/index.html")

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

        doc\_externo.close()

        ctx = Context({

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

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

            'fecha' : fecha\_actual,

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

        })

        documento = plt.render(ctx)

        return HttpResponse(documento)

Esta será nuestra plantilla html.

<!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>

*{% 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 %}*

*{% 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>

</body>

</html>

Como vimos en el módulo anterior es posible dentro de una plantilla trabajar con estructuras de control de flujo, bucles, etc, ahora tambien podemos trabajar con operadores de comparación dentro de una plantilla para evaluar valores.

Ahora realizaremos un evaluación en este caso de que si el nombre del profesor es uno imprima un mensaje que indique es que cierto, en caso de que no usaremos un else.

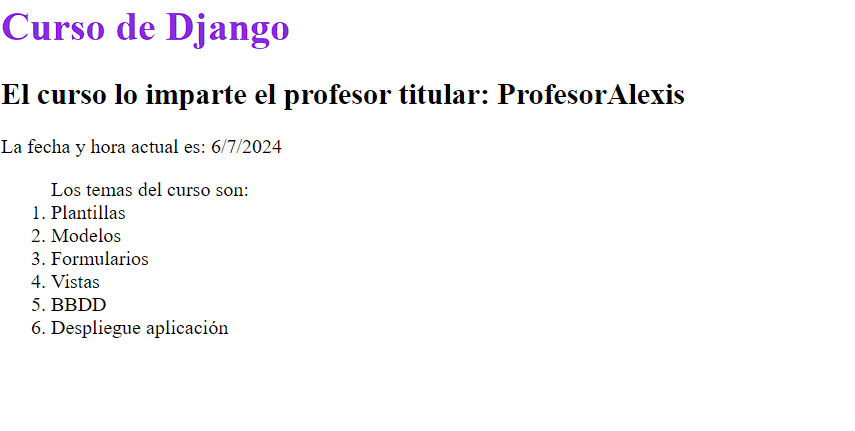
    {% if nombre\_persona == 'Alexis' %}

        <h2>El curso lo imparte el profesor titular: Profesor{{nombre\_persona}}</h2>

    {% else %}

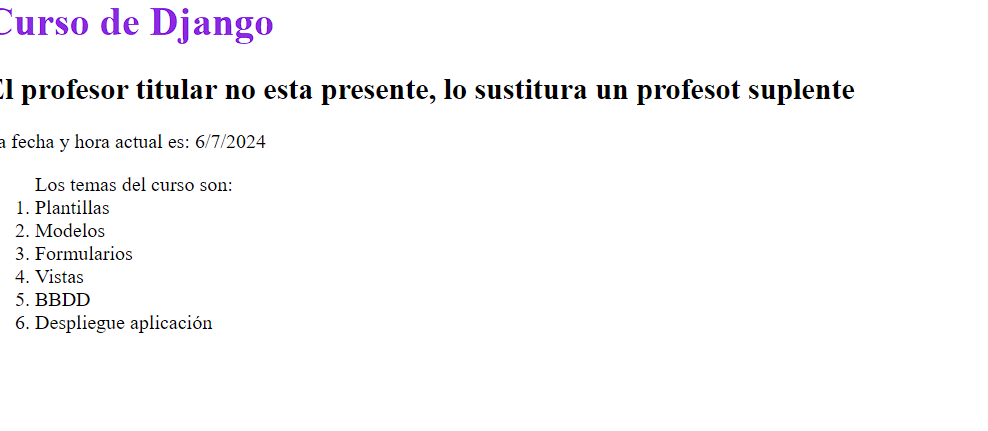
        <h2>El profesor titular no esta presente, lo sustitura un profesot suplente</h2>

    {% endif %}



Pero en dado caso que no sea asi.

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



Django tiene la filosofía de que toda la parte de la lógica debería de estar dentro del archivo views.py y la parte visual en las plantillas.

En las plantillas podemos usar los denominados filtros que como su nombre indica filtran la informacion de una manera.

Digamos que usamos el método upper para aplicarlo en el nombre del profesor.

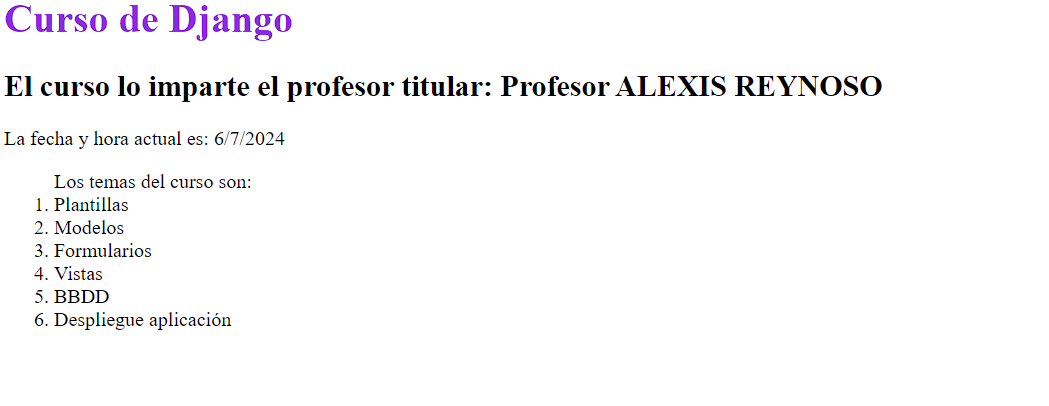
        <h2>El curso lo imparte el profesor titular: Profesor {{nombre\_persona.upper}} {{apellido\_persona.upper}}</h2>



Esta accion tambien se podía haber hecho con un filtro para usarlo debemos hacer uso en donde se encuentra el punto, reemplazarlo por un símbolo piper/tubería

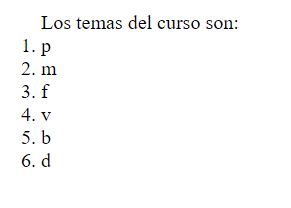
        <h2>El curso lo imparte el profesor titular: Profesor {{nombre\_persona|upper}} {{apellido\_persona|upper}}</h2>

El resultado será exactamente lo mismo.



Existe tambien la opcion de poder encadenar filtros

                <li>{{tema|first|lower}}</li>



Siendo el filtro first aquel que nos devuelve la primera letra de nuestra cadena, seguida del filtro lower que las convierte e minúsculas.

Para mas informacion accede a la documentación oficial de django: <https://docs.djangoproject.com/en/5.0/ref/templates/builtins/>

O a la página devdocs que contiene la documentación de varios lenguajes y frameworks: <https://devdocs.io/>

Ahora pasaremos al ultimo tema de este módulo que serán los cargadores de plantillas.

Hemos estado cargando nuestras plantillas usando la función open()

doc\_externo = open("C:/Users/Alexis/Desktop/Programacion/Backend/Frameworks/Django/Archivos Django/ProyectoDjango7/ProyectoDjango7/templates/index.html")

Para despues pasábamos por un template la plantilla para que la leyese.

 plt = Template(doc\_externo.read())

Para despues renderizar ese objeto utilizando un contexto.

        doc\_externo.close()

        ctx = Context({

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

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

            'fecha' : fecha\_actual,

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

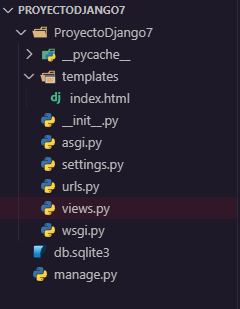
        })

        documento = plt.render(ctx)

Sin embargo, esta nos es la óptima más correcta de hacerlo debido a que a la hora de trabajar en un proyecto serio trabajaremos con varias plantillas. Para mejorar esto haremos uso de un cargador de plantillas un loader conseguimos que todo este proceso que hicimos se simplifique.

Básicamente consiste en decirle a nuestro proyecto de django que las plantillas a utilizar en el mismo se encuentran en un lugar en concreto.

En este caso recordemos nuestras plantillas se ubicarán en una subcarpeta llamada “Templates”.



Entonces esto simplificara el uso de plantillas ya que tendremos que evitar usar todo el proceso, y bastara con solo pasar el nombre de la plantilla.

Comenzaremos por importar el loader.

from django.template import loader

Una vez importado procederemos a modificar nuestro codigo de views.py

*# doc\_externo = open("C:/Users/Alexis/Desktop/Programacion/Backend/Frameworks/Django/Archivos Django/ProyectoDjango7/ProyectoDjango7/templates/index.html")*

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

*# doc\_externo.close()*

En este caso lo comentaremos, ahora debemos indicar donde estará la ruta donde se encuentren nuestras plantillas, esto lo modificaremos en el archivo settings.py

TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [],

        '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',

            ],

        },

    },

]

Buscaremos la lista llamada TEMPLATES que tendrá todo lo relativo a las plantillas, Usaremos DIRS para indicar donde se guardan nuestras plantillas, pegaremos la ruta de la carpeta donde se encuentran dichas plantillas.

Ahora una vez este la ruta de las plantillas definidas pasaremos al views.py y usaremos la función loader con el método get\_template.

        doc\_externo = loader.get\_template('index.html')

En este caso lo guardamos dentro de la variable doc\_externo y bastara con solo indicar el nombre de la plantilla.

Ahora lo que tendremos que hacer es quitar el contexto que le estábamos pasando, pero lo pasaremos como solo un diccionario.

        doc\_externo = loader.get\_template('index.html')

        ctx = {

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

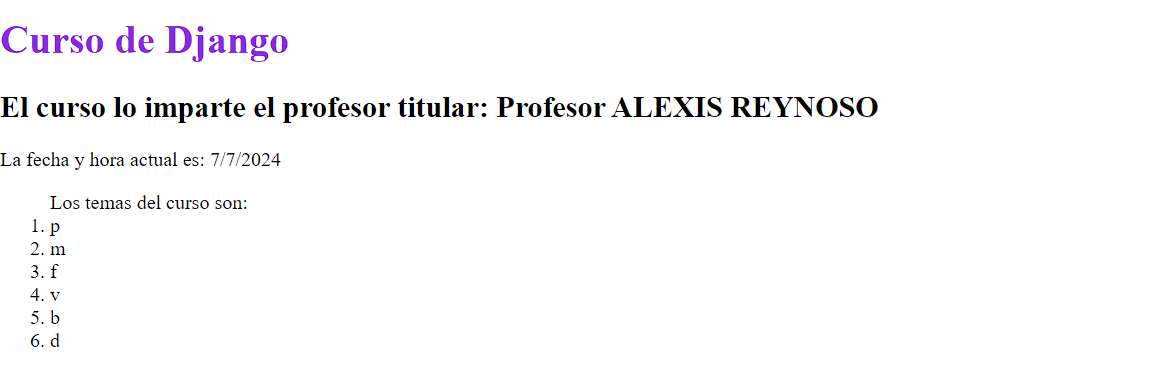
            'apellido\_persona' : p1.**apellido**,

            'fecha' : fecha\_actual,

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

        }

        documento = doc\_externo.render(ctx)



Vemos que esto hace que la pagina funcione igual que antes.