

4- Django (cargadores de plantillas y herencia)

En la actividad anterior al momento de cargar las plantillas se usaba toda la ruta de la plantilla:


```
plantilla = open("C:/miapp/miapp/templates/plantilla1.html")  
template = Template(plantilla.read())
```

Esta práctica no es muy correcta ni óptima. Así que debemos optimizarlo.

1- Importar la clase loader

```
from django.template import loader
```

2- Configurar la ruta de las plantillas en el archivo settings.py
En el apartado de DIRS



```
TEMPLATES = [  
    {  
        'BACKEND': 'django.template.backends.django.Dj  
        'DIRS': ['C:/miapp/miapp/templates/'],  
        'APP_DIRS': True,  
        'OPTIONS': {  
            'context_processors': [  

```

3- Editar la vista que anteriormente cargaba la plantilla, quedaría de la siguiente manera:

```
def contHTML(request):  
    nombre= "Vianey"  
    fechaActual = datetime.datetime.now()  
    alumnos= ["Martha", "Esther", "Miguel", "Jonathan"]  
    maestros= ["Javier", "Antonio", "Lucy", "Sofía"]  
    plantilla = loader.get_template("plantilla1.html")  
    documento = plantilla.render({"nombreAlumno": nombre,  
                                "fechaA": fechaActual,"alumnos": alumnos,"maestros": maestros})  
    return HttpResponse(documento)
```

Podrás visualizar todo de la misma manera que estaba antes:

Lista de alumnos

Total de alumnos: 4

- martha
- esther
- miguel
- jonathan

El alumno ganador es: Martha

- 4- Se puede mejorar aún más el proceso de cargar plantillas si ajustas la importación de la siguiente manera

Cambia esto:

```
from django.template import loader
```

Por esto:

```
from django.template.loader import get_template
```

Y en la vista únicamente llama a la función **get_template**

```
plantilla = get_template("plantilla1.html")
```

Todo se debe visualizar de la misma manera.

Shortcuts Django

Si queremos optimizar aún más el proceso anterior existen funciones de atajo conocidas como Shortcuts. La documentación la encuentras en el siguiente link: <https://docs.djangoproject.com/en/4.2/topics/http/shortcuts/>

Render() es una función que nos permite devolver un objeto HttpResponse de una mejor forma.

- 1- Importa la función render

```
from django.shortcuts import render
```

- 2- Revisa la documentación, para usar esta función se necesitan 2 argumentos, el request y el nombre de la plantilla:

Argumentos requeridos

request

El objeto de solicitud utilizado para generar esta respuesta.

template_name

El nombre completo de una plantilla para usar o secuencia de nombres de plantilla. Si se da una secuencia, se utilizará la primera plantilla que exista. Consulte la [documentación de carga de plantillas](#) para obtener más información sobre cómo se encuentran las plantillas.

Argumentos opcionales

context

Un diccionario de valores para agregar al contexto de la plantilla. Por defecto, este es un diccionario vacío. Si se puede llamar a un valor en el diccionario, la vista lo llamará justo antes de representar la plantilla.

content_type

El tipo MIME que se usará para el documento resultante. El valor predeterminado es 'text/html'.

status

El código de estado de la respuesta. El valor predeterminado es 200.

using

El **NAME** de un motor de plantillas que se utilizará para cargar la plantilla.

Los otros argumentos son opcionales, por ejemplo el contexto (el diccionario de valores).

En este caso vamos a colocar, el request, el nombre de la plantilla y el diccionario de datos

```
def contHTML(request):  
    nombre= "Vianey"  
    fechaActual = datetime.datetime.now()  
    alumnos= ["Martha", "Esther", "Miguel", "Jonathan"]  
    maestros= ["Javier", "Antonio", "Luz", "Sofía"]  
    return render(request, "plantilla1.html", {"nombreAlumno": nombre,  
        "fechaA": fechaActual,"alumnos": alumnos,"maestros": maestros})
```

Todo sigue funcionando perfectamente:

Lista de alumnos

Total de alumnos: 4

- martha
- esther
- miguel
- jonathan

El alumno ganador es: Jonathan

Lista de maestros

- javier
- antonio
- luz
- sofia

De esta manera se renderiza una plantilla de una manera más eficiente y abreviada.

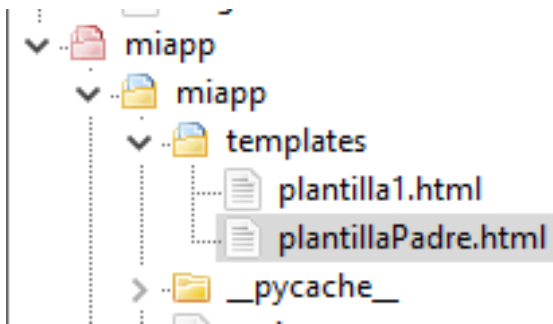
Herencia de Plantillas

Existe un problema común del desarrollo web al momento de usar plantillas y muchas páginas con código repetitivo: ¿Cómo reducimos la duplicación y redundancia de las áreas comunes de las páginas, como por ejemplo, los paneles de navegación?

La mejor forma de solucionar este problema con Django es usar una estrategia más elegante llamada herencia de plantillas.

En esencia, la herencia de plantillas te deja construir una plantilla base "esqueleto" que contenga todas las partes comunes de tu sitio y definir "bloques" que los hijos puedan sobrescribir.

Paso 1- Crear una nueva plantilla llamada plantillaPadre.html



Contiene lo siguiente:

```
<html>
  <head>
    <title>{% block title %} {% endblock %}</title>
  </head>
  <body>
    <h1>Bienvenido al curso de Python</h1>

    {% block content %}

    {% endblock %}

    <p>Esto aparece en todas las páginas</p>

  </body>
</html>
```

En una plantilla padre se establecen los valores estáticos (los que no van a cambiar y que serán lo mismo en todas las páginas, por ejemplo:

```
<html>
  <head>
    <title>{% block title %} {% endblock %}</title>
  </head>
  <body>
    <h1>Bienvenido al curso de Python</h1>
    {% block content %}

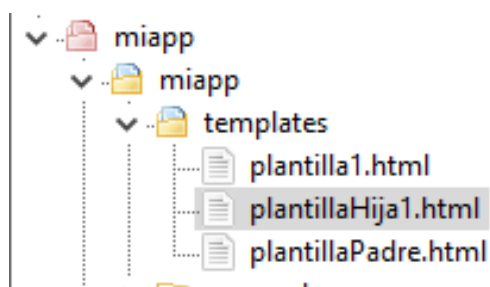
    {% endblock %}

    <p>Esto aparece en todas las páginas</p>
  </body>
</html>
```

Y se definen aquellos espacios dinámicos, los que serán modificados por las plantillas hijos, por ejemplo el título.

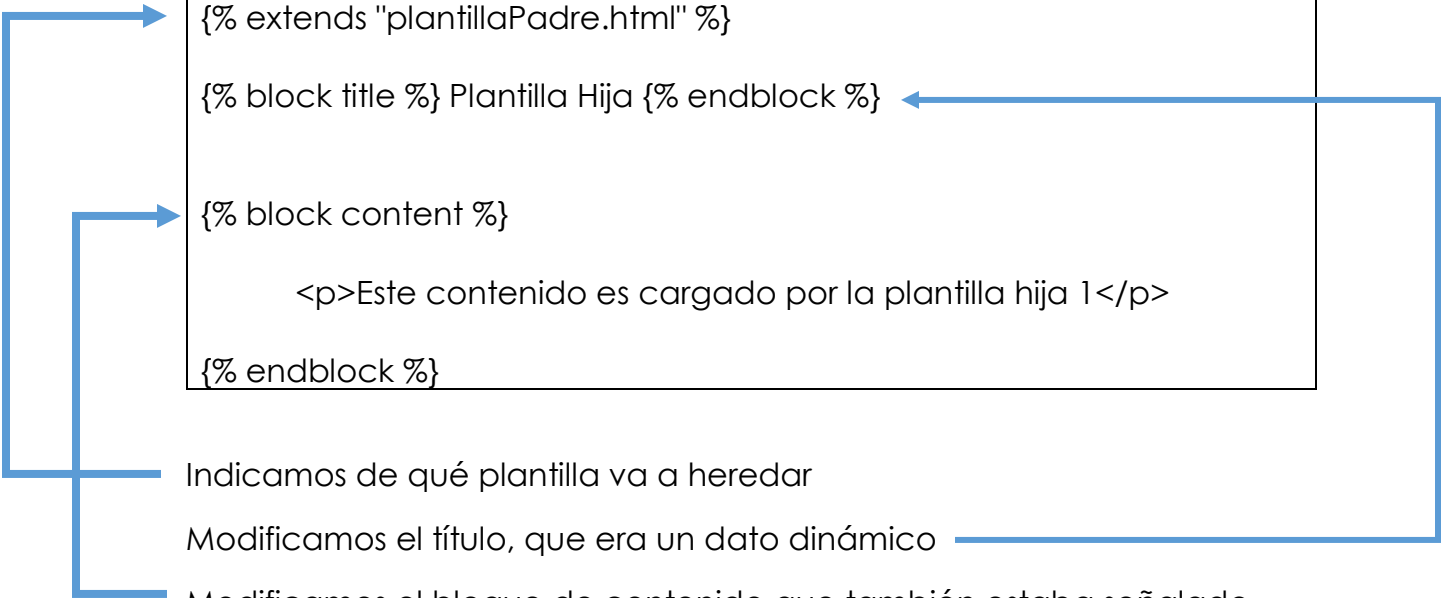
Y se crea un bloque de contenido en el cuerpo del documento.

3- Crear una plantilla hija, llamada plantillaHija1.html



Esta plantilla ya no contiene las etiquetas básicas de la estructura HTML porque las va a heredar de la plantilla padre.

Para que las herede debemos indicarle de cuál plantilla va a heredar el contenido.



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

{% block title %} Plantilla Hija {% endblock %}

{% block content %}

    <p>Este contenido es cargado por la plantilla hija 1</p>

{% endblock %}
```

Indicamos de qué plantilla va a heredar

Modificamos el título, que era un dato dinámico

Modificamos el bloque de contenido que también estaba señalado como dinámico en nuestra plantilla padre.

4- Para poder visualizarlo en el navegador debemos crear la vista que cargue la plantilla hija

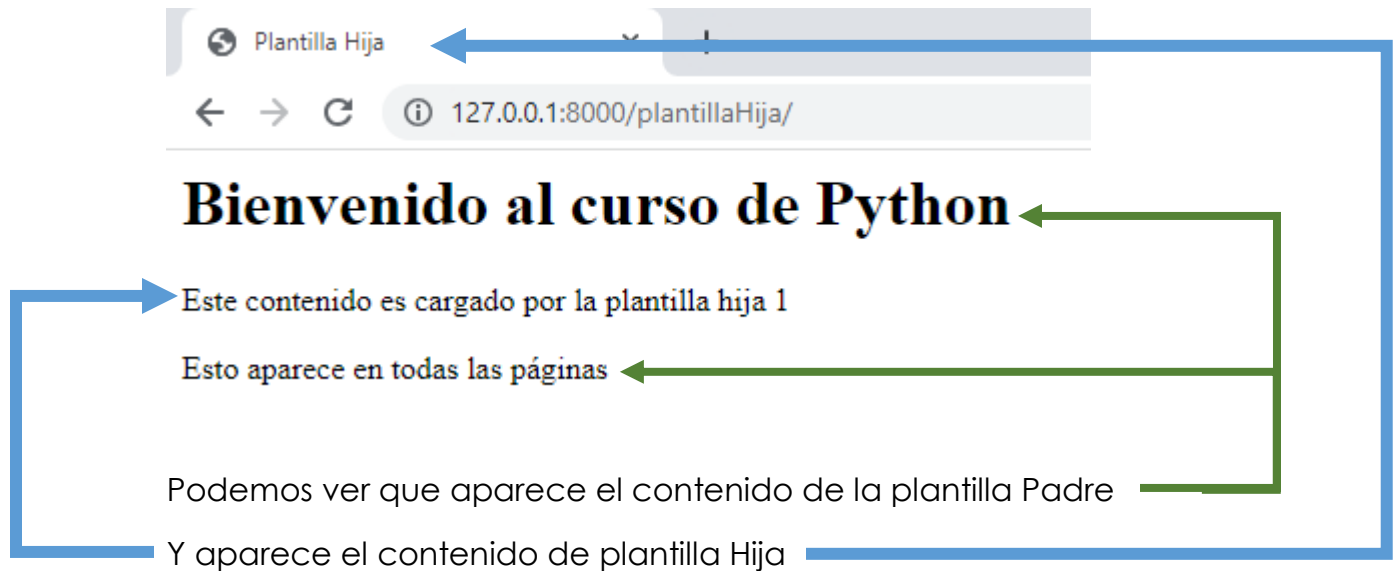
```
def plantillaHija(request):
    return render(request, "plantillaHija1.html")
```

5- Y agregar la ruta en el archivo de urls.py

```
from miapp.views import bienvenida, categoriaEdad, contHTML
from miapp.views import plantillaHija

urlpatterns = [
    path('admin/', admin.site.urls),
    path('bienvenida/', bienvenida),
    path('edad/<int:edad>', categoriaEdad),
    path('html/', contHTML),
    path('plantillaHija/', plantillaHija),
]
```

6- Realiza una prueba en tu navegador



ACTIVIDAD:

Cambia el color de texto del contenido que aporta la plantilla Padre.

Bienvenido al curso de Python

Este contenido es cargado por la plantilla hija 1

Esto aparece en todas las páginas

Crea otra plantilla hija 2, la cual se visualice de la siguiente manera:

