

Clase 17. PYTHON

Portfolio Parte I



Django

¿Recuerdas que la clase pasada te pedimos que instales Django en tu computadora?

¡Comenzaremos a poner manos al código! 🕶️

Psst... en caso de que no lo hayas hecho, puedes hacerlo ahora mientras esperamos a que lleguen todos los estudiantes

[Ver tutorial](#)

Esta clase va a ser

- grabada

a

Objetivos de la clase

- **Realizar** los comandos básicos de Django.
- **Utilizar** MVC creando las primeras vistas.
- **Aplicar** el uso de templates

Repositorio Github

Te dejamos el acceso al Repositorio de Github donde encontrarás todo el material complementario y scripts de la clase.

✓ [Repositorio Python](#)



Temario

16

Git & GitHub

- ✓ Git
- ✓ GitHub

17

Django – Portfolio Parte I

- ✓ [Django](#)
- ✓ [Plantillas Django](#)

18

Django – Portfolio Parte II

- ✓ Mejoras en las plantillas
- ✓ Modelo
- ✓ ¿Cómo crear una app?

Django

¿Qué es Django?

Definición

Ya sabemos programar en Python, es momento de empezar a aplicar un poco de todo esto para poder crear un **proyecto web**.

Para realizar un proyecto web necesitaremos usar un **framework** que nos facilite esta tarea, el más sencillo de todos los frameworks web de Python es Flask. Pero en este apartado usaremos **Django** que es mucho más completo y es de **código abierto**.

¿Qué es un framework?

Es un entorno de trabajo, un ecosistema que nos facilita crear algo, en este caso un sitio web, sin mayores esfuerzos.



Fundamentos de Django (MVC)

¿QUÉ ES EL MODELO VISTA CONTROLADOR?

(MVC) es una arquitectura de software que nos permite tener una organización adecuada de nuestro código.



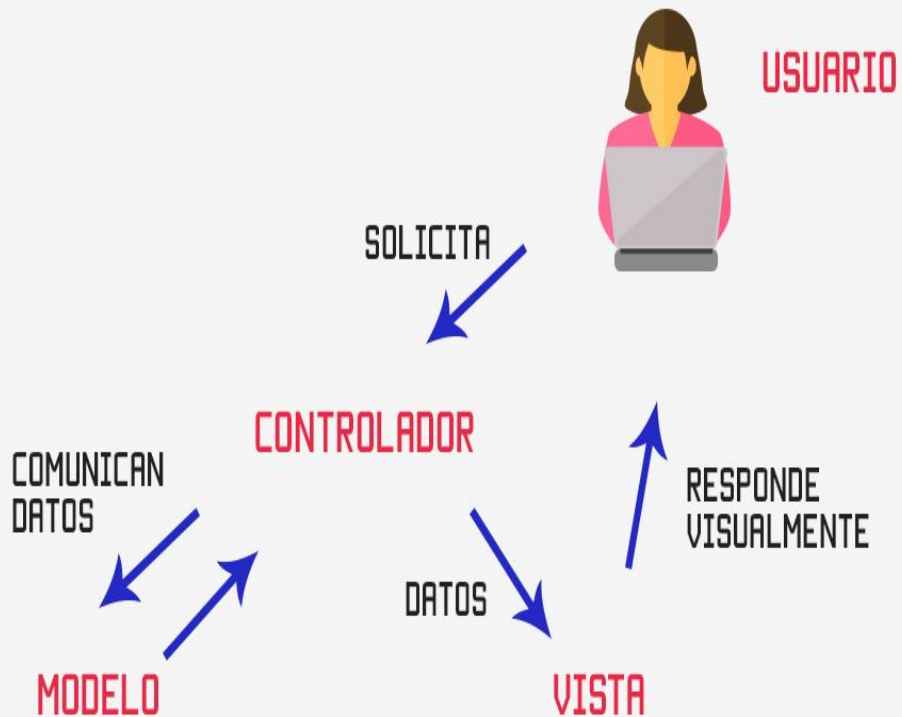
Modelo vista Controlador

Django se basa en el patrón de diseño web, denominado, modelo vista controlador.

Modelo: Maneja los datos.

Vista: Lo que el usuario ve.

Controlador: Interacción entre los datos y lo que ve el usuario.



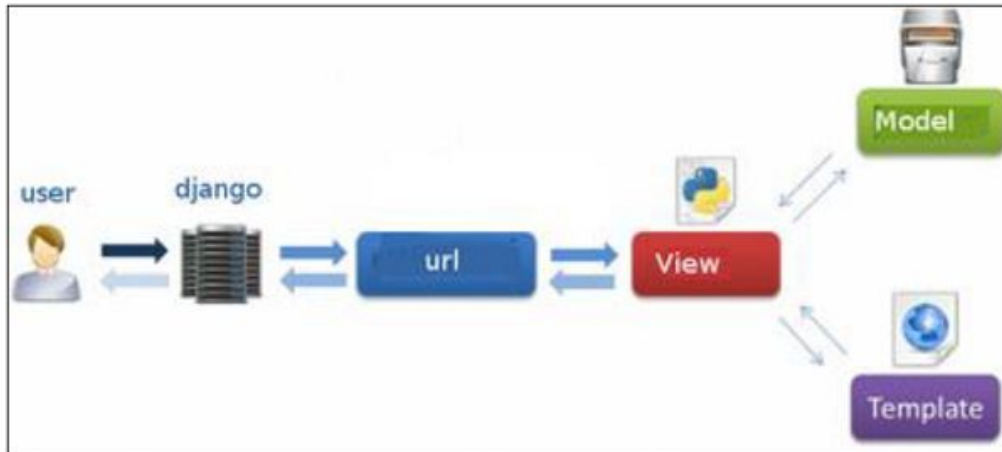
Modelo vista controlador

Esto nos permite que la aplicación sea más funcional, mantenible, escalable y colaborativa.

Modelo vista controlador

Este es el patrón de diseño clásico Web para cualquier proyecto web en cualquier lenguaje.

Django modifica este patrón por el modelo MTV, Model template, View.





Ejemplo en vivo

A continuación vamos a crear nuestro primer proyecto

Duración: **10 minutos**



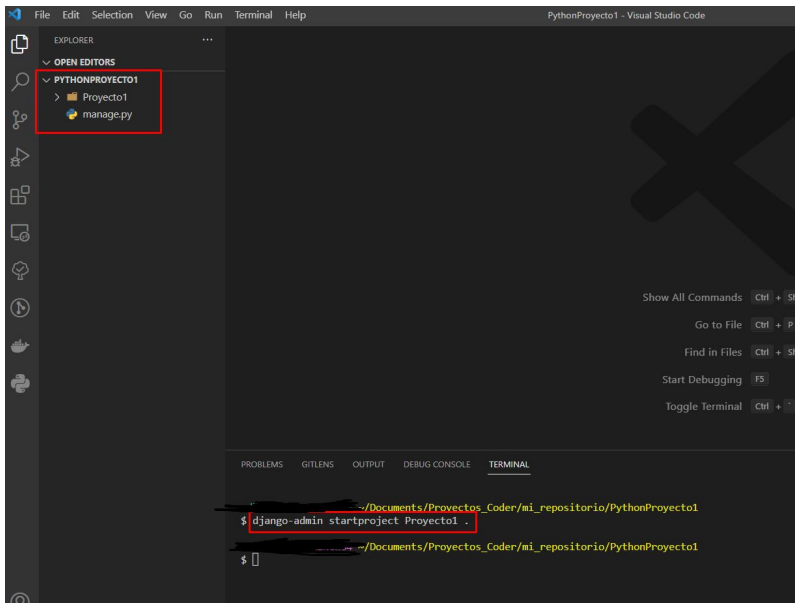
Pasos para crear nuestro proyecto

1. Crear una carpeta, en mi caso, `PythonProyecto1`
2. Iniciar VSC o tu editor (o incluso cmd) y pararte en esa carpeta.
3. Escribir `django-admin startproject Proyecto1.` (si, al final va un punto). Para esto deberemos tener instalar Django por medio de la terminal ejecutando el comando `pip3 install Django`

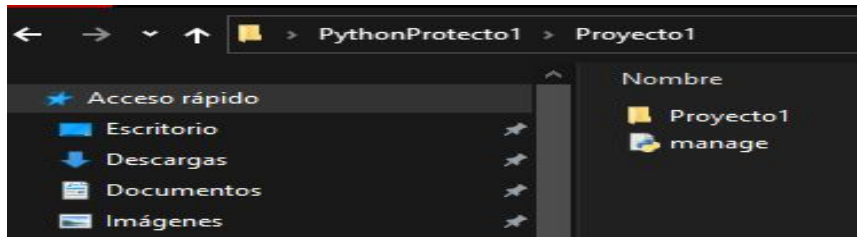
“Donde Proyecto1 será una nueva carpeta que se creará en PythonProyecto1”



Pasos para crear nuestro proyecto



Hasta acá deberíamos tener algo así.
Con este comando se creará una carpeta y un
archivo **manage**.



Manage es importante porque nos permite interactuar con los proyectos, con algunos comandos muy simples. Por ejemplo: ayudas, migraciones, testeos, etc.

En la carpeta **Proyecto1** veremos que hay varios archivos **.py**.

__init__.py para que sepa que es un paquete,
__settings.py para manipular la configuración, y **url/wsgi** que pronto sabremos su uso.



Pasos para crear nuestro proyecto



4. Tipeamos `python manage.py migrate`



5. Verificamos con: `python manage.py runserver`



Pasos para crear nuestro proyecto

```
~/Documents/Proyectos_Coder/mi_repositorio/PythonProyecto1
$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
```

```
~/Documents/Proyectos_Coder/mi_repositorio/PythonProyecto1
$
```

```
~/Documents/Proyectos_Coder/mi_repositorio/PythonProyecto1
$ python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
April 10, 2023 - 05:24:04
Django version 4.1.1, using settings 'Proyecto1.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```



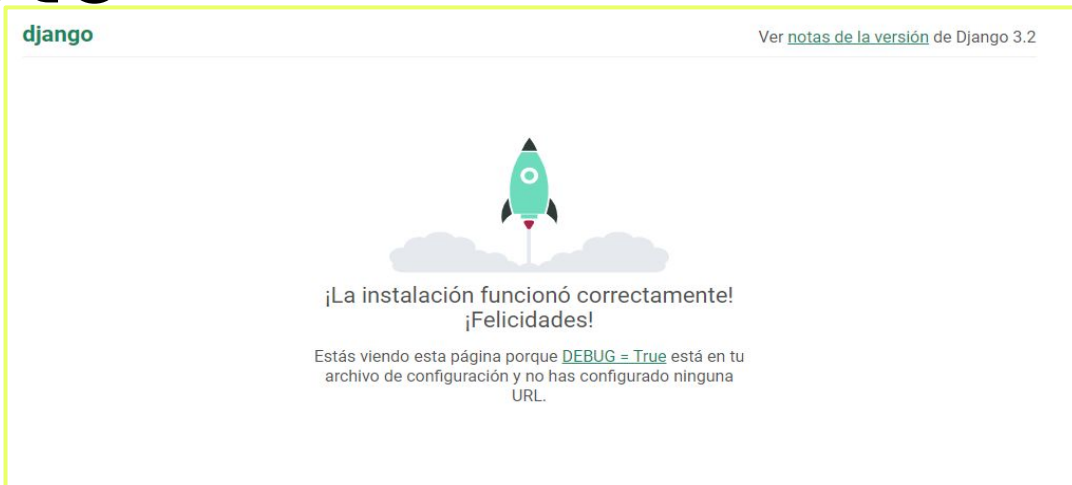
Pasos para crear nuestro proyecto

Aquí también podemos ver el comando a utilizar para la creación de nuestro proyecto

```
sudo pip install django  
pip3  
python -m pip install django  
  
python -m django startproject  
Proyecto1
```



Pasos para crear nuestro proyecto



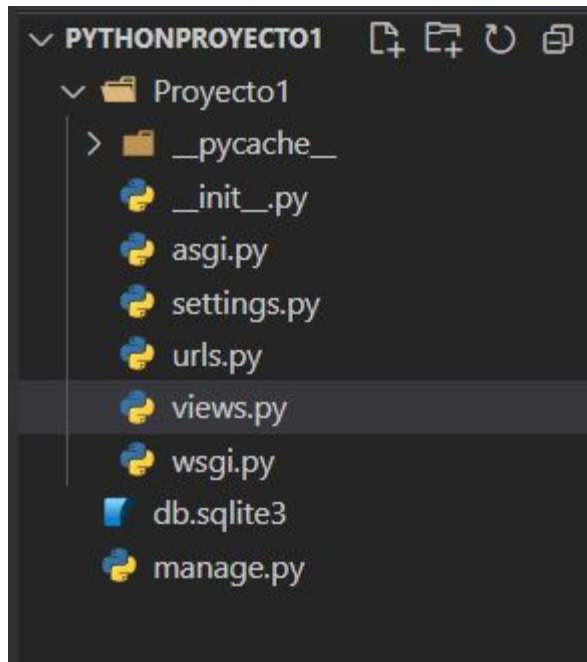
¡Tenemos nuestro primer proyecto en funcionamiento!



Break

¡10 minutos y volvemos!

Nuestro primer view



Nuestro primer view

Lo primero que tendremos que hacer es crear un archivo para esta nueva vista o view.

Usualmente se suele llamar **view.py**, debemos crearlo en la carpeta del proyecto, es decir en la misma ruta que tenemos **urls**, **__init__**, **wsgi**, etc. Debería quedar así





Nuestro primer view

Vamos a nuestro archivo `views.py`, e importamos los elementos de un `Response` de la siguiente manera:

```
from django.http import HttpResponse
```



Nuestro primer view

Luego iniciamos creando nuestra primer vista, por medio de un método que recibe como parámetro la request y nos da por resultado un response:

```
def saludo(request):  
    return HttpResponse("Hola Django - Coder")
```

Nuestro primer view

Listo, ahora solo necesitamos avisarle a Python y Django cual será la URL que nos llevará a la vista que creamos. Eso lo hacemos en el archivo **urls.py**.

```
from django.contrib import admin
from django.urls import path
from Proyecto1.views import saludo    #Para acceder a la vista hay que importar el modulo y el método

urlpatterns = [
    path('admin/', admin.site.urls),
    path('saludo/', saludo),          #ojo la url no hace falta que se llame saludo/ el nombre es libre,
                                     #pero la vista sí debe llamarse por su nombre
]
```

Nuestro primer view

Para esto debemos primero importar **Proyecto1.views** y generar el vínculo entre una url y la vista, nos debería quedar así:

```
from django.contrib import admin
from django.urls import path
from Proyecto1.views import saludo    #Para acceder a la vista hay que importar el modulo y el método

urlpatterns = [
    path('admin/', admin.site.urls),
    path('saludo/', saludo),    #ojo la url no hace falta que se llame saludo/ el nombre es libre,
                                #pero la vista sí debe llamarse por su nombre
]
```

Nuestro primer view

Solo resta probar que funcione lo que hicimos 🙄

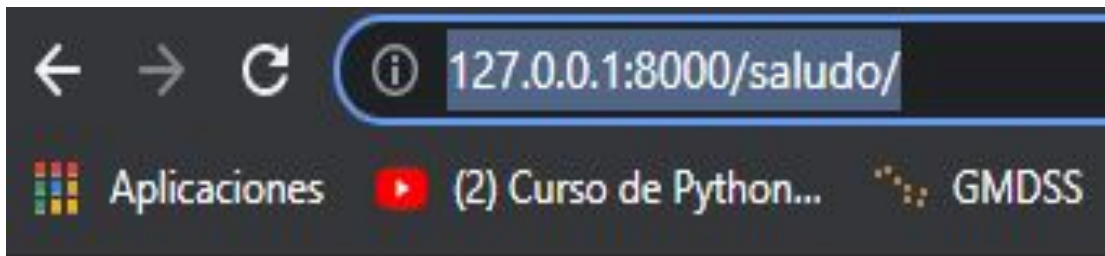
Para esto nos apoyamos en lo visto anteriormente y arrancamos el servidor de la siguiente manera:

```
python manage.py runserver
```

Nuestro primer view

Luego entramos a: <http://127.0.0.1:8000/saludo/>

Ya podemos ver nuestra primer página, o mejor dicho nuestra vista



Agregar otra view

Agregar otra view

Hagamos lo mismo para ver lo sencillo que ha sido:

```
from django.http import HttpResponse

def saludo(request): #Nuestra primera vista :)
    return HttpResponse("Hola Django - Coder")
```

```
def segunda_vista(request):
    return HttpResponse("<br><br>Ya entendimos esto, es muy simple :) ")
```

Podemos poner comandos HTML 🕶️

```
from django.contrib import admin
from django.urls import path
from Proyecto1.views import saludo, segunda_vista

urlpatterns = [
    path('admin/', admin.site.urls),
    path('saludo/', saludo), #ojo la url no hace falta el nombre de la vista
                             #pero la vista sí debe tener el nombre
    path('segundavista/', segunda_vista),
]
```

Pasaje de parámetros

¿Qué es?

Muchas veces queremos mostrar en una vista el resultado de algún proceso interno realizado en Python, esto es básicamente enviar parámetros por medio de la vista.

Veamos cómo se hace, es muy simple y parecido a la anterior.



Ejemplo

Así es el proceso de creación de una nueva vista que muestra **DíaDeHoy**.

```
def diaDeHoy(request):  
  
    dia = datetime.datetime.now()  
  
    documentoDeTexto = f"Hoy es día: <br> {dia}"  
  
    return HttpResponse(documentoDeTexto)
```

```
from django.contrib import admin  
from django.urls import path  
from Proyecto1.views import saludo, segunda_vista, diaDeHoy  
  
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('saludo/', saludo), #ojo la url no hace falta que  
                             #pero la vista sí debe llamar  
  
    path('segundavista/', segunda_vista),  
    path('diaDeHoy/', diaDeHoy),  
]
```

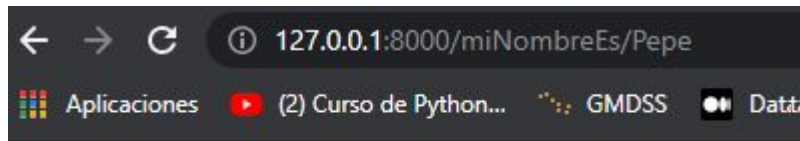


Hoy es día:
2021-09-01 20:07:54.838810

Parámetros desde la URL

¿Qué es?

Mi vista puede trabajar sobre algún dato que nos enviar por **url**, veamos cómo funciona así la respuesta aparece sola:



Mi nombre es:

Pepe

```
def miNombreEs(self, nombre):  
    documentoDeTexto = f"Mi nombre es: <br><br> {nombre}"  
    return HttpResponse(documentoDeTexto)
```

```
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('saludo/', saludo), #ojo la url no ha  
                             #pero la vista ss  
  
    path('segundavista/', segunda_vista),  
    path('diaDeHoy/', diaDeHoy),  
    path('miNombreEs/<nombre>', miNombreEs),  
]
```

Plantillas Django

Plantillas

¿Qué son?

Son archivos que nos permiten separar la vista de la estética, es decir guardar en un archivo separado de todo lo que guardábamos en “documento”, para así enviar por la HttpResponse.

Entonces, ¿podríamos decir que así se crea un template? 😊

¡Exacto! 🙌

Recordemos que Django se basaba en Modelo, Vista, Template.



Creando nuestro primer template.

Duración: 20 minutos



ACTIVIDAD EN CLASE

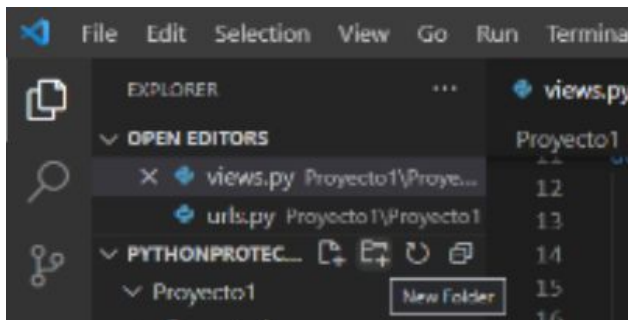
Creando nuestro primer template.

Descripción de la actividad.

Vamos a crear nuestro primer template. Esto necesitará de un “template”, propiamente dicho, es decir, lo que se muestra. Además, necesitaremos un “contexto”, esto es, para manejar contenido que cambia, por ejemplo nuestro nombre en el ejercicio anterior. Y por último crear un “render” es decir, una transformación a página web.

Paso a paso

1 Dentro del proyecto creamos una carpeta que se puede llamar plantillas.



2 Dentro de esa nueva carpeta creamos un archivo `template1.html`.

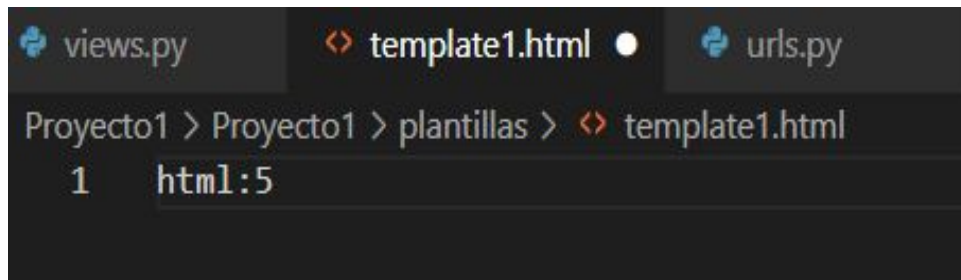


Paso a paso

3

Dentro de `template1`, escribimos `html:5` y damos clic en "enter".

Se crea automáticamente un esqueleto.



The screenshot shows a code editor with three tabs at the top: `views.py`, `template1.html` (which is the active tab), and `urls.py`. The breadcrumb navigation at the top of the editor reads: `Proyecto1 > Proyecto1 > plantillas > template1.html`. The main editing area shows a single line of code: `1 html:5`, where the line number '1' is on the left and the code 'html:5' is on the right.

Paso a paso

4

Dentro de `<body>` `</body>`, escribimos lo que queremos que se vea en nuestra página web.

```
views.py x template1.html urls.py
Proyecto1 > Proyecto1 > plantillas > template1.html > html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8 </head>
9 <body>
10
11 </body>
12 </html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>

  Excelente!!!!!!.... Es muy fácil usar plantillas.

</body>
</html>
```

Paso a paso

5

5a. Llamamos a template1 desde una nueva vista (`probandoTemplate`).

```
def probandoTemplate(self):  
  
    miHtml = open("C:/Users/nico_/Desktop/PythonProtecto1/Proyecto1/Proyecto1/plantillas/template1.html")  
  
    plantilla = Template(miHtml.read()) #Se carga en memoria nuestro documento, template1  
    ##OJO importar template y contex, con: from django.template import Template, Context  
  
    miHtml.close() #Cerramos el archivo  
  
    miContexto = Context() #EN este caso no hay nada ya que no hay parametros, IGUAL hay que crearlo  
  
    documento = plantilla.render(miContexto) #Aca renderizamos la plantilla en documento  
  
    return HttpResponse(documento)
```

Paso a paso

5

5b. Aquí crearemos el contexto y el render.

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('saludo/', saludo),      #ojo la url no hace
    path('segundavista/', segunda_vista),
    path('diaDeHoy/', diaDeHoy),
    path('miNombreEs/<nombre>', miNombreEs),
    path('probandoTemplate/', probandoTemplate),
]
```



Excelente !!!!!!! Es muy fácil usar plantillas.

¿Qué hemos logrado?



Logramos separar lo que construye a la página web, es decir el **HTML**, del procesamiento de los datos dado por la vista.

De esta manera podemos tener un diseñador trabajando en el HTML y nosotros desarrollando en **Python/Django** sin saber nada de HTML.

**Por último, hablemos de
entornos Virtuales y
Paquetes en Python**

Entornos virtuales

Los entornos virtuales se pueden describir como directorios de instalación aislados. Este aislamiento te permite localizar la instalación de las dependencias de tu proyecto, sin obligarte a instalarlas en todo el sistema.

Muchas veces se recomienda a la hora de programar, usar este tipo de herramientas. Es importante mencionar que los entornos virtuales, pueden crearse en diferentes lenguajes de programación, en este caso en particular, veremos como crearlos en Python.

Entornos virtuales

Primero de todo, instalamos la librería: **pip install virtualenv**

```
PROBLEMAS 7 SALIDA CONSOLA DE DEPURACIÓN TERMINAL JUPYTER
• PS C:\Users\layla> pip install virtualenv
Collecting virtualenv
  Downloading virtualenv-20.16.3-py2.py3-none-any.whl (8.8 MB)
    |████████████████████| 8.8 MB 3.3 MB/s
Collecting platformdirs<3,>=2.4
  Downloading platformdirs-2.5.2-py3-none-any.whl (14 kB)
Collecting distlib<1,>=0.3.5
  Downloading distlib-0.3.5-py2.py3-none-any.whl (466 kB)
    |████████████████████| 466 kB 6.4 MB/s
Requirement already satisfied: filelock<4,>=3.4.1 in c:\users\layla\anaconda3\lib\site-packages (from virtualenv) (3.6.0)
Installing collected packages: platformdirs, distlib, virtualenv
Successfully installed distlib-0.3.5 platformdirs-2.5.2 virtualenv-20.16.3
○ PS C:\Users\layla> 
```

Entornos virtuales

Para crear un entorno virtual, debemos decidir en qué carpeta lo queremos crear y ejecutar el módulo venv como script con la ruta a la carpeta. Por ejemplo, creamos una carpeta en el Escritorio y allí lo ejecutamos.

PROBLEMAS

7

SALIDA

CONSOLA DE DEPURACIÓN

TERMINAL

JUPYTER

```
PS C:\Users\layla> cd Escritorio
PS C:\Users\layla\Escritorio> mkdir ejemplo_entorno_virtual_python
```

Entornos virtuales

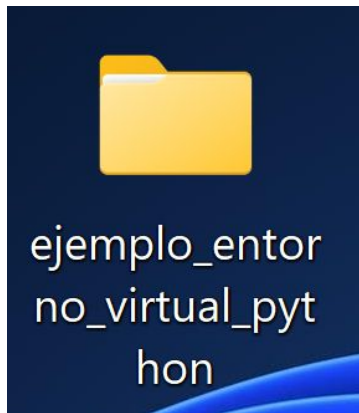
Mode	LastWriteTime	Length	Name
d----	7/8/2022 16:31		ejemplo_entorno_virtual_python

Ingresamos a la carpeta y ejecutamos el siguiente comando: `python -m venv tutorial-env`

```
• PS C:\Users\layla\Escritorio> cd ejemplo_entorno_virtual_python
• PS C:\Users\layla\Escritorio\ejemplo_entorno_virtual_python> python -m venv tutorial-env
```

Entornos virtuales

Chequeamos el Escritorio y la creación de las carpetas:



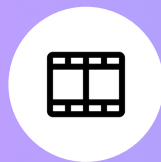
<input type="checkbox"/>	Nombre	Fecha de modifica...	Tipo	Tamaño
<input type="checkbox"/>	tutorial-env	7/8/2022 16:36	Carpeta de arc...	

El comando anterior creó el directorio tutorial-env si no existe, y también directorios dentro de él que contienen una copia del intérprete de Python y varios archivos de soporte.



#Codertraining

¡No dejes para mañana lo que puedes practicar hoy! Te invitamos a revisar la [Guía de Ejercicios Complementarios](#), donde encontrarás un ejercicio para poner en práctica lo visto en la clase de hoy.



¿Quieres saber más?
**Te dejamos material
ampliado de la clase**



MATERIAL AMPLIADO

Recursos multimedia

Django

✓ [Tutorial Django básico \(Python\)](#) | Developer

Entornos virtuales

✓ [Creación de entornos virtuales.](#)

✓ [Entornos virtuales y paquetes](#)

Disponible en nuestro repositorio.

Resumen de la clase hoy

- ✓ Crear un primer proyecto en Django.
- ✓ Crear primeras vistas
- ✓ Relacionar un template con una vista.
- ✓ Entornos virtuales en Python

¿Preguntas?

Muchas gracias.

Opina y valora
esta clase

#DemocratizandoLaEducación