TEXT CLASS REVIEW

TEMAS A TRATAR EN EL CUE

- Aislación del entorno Python.
- Manejo de distintas versiones.
- El comando venv.
- Iniciando el entorno virtual.
- Saliendo del entorno virtual.
- Instalaciones en el entorno global.
- El enrutador de Django.
- MVC en Django para aplicaciones monolíticas.
- Herencia de componentes en el modelo MVC.
- El principio DRY y su aplicación en el entorno.
- Python/Django.
- ¿Qué son los Templates de Django?
- Renderización en Django.

LIBRERIAS PROPIAS DE CADA PROYECTO

AISLACIÓN DEL ENTORNO PYTHON

Con el fin de facilitar el despliegue, se crea un entorno aislado para realizar pruebas. Lo ideal es disponer de un entorno en local lo más parecido posible al entorno remoto.

Se utiliza, en este caso, `virtualenvwrapper` que incluye `virtualenv`, el cual se analizará más adelante.

MANEJO DE DISTINTAS VERSIONES

- Existen varias versiones peal textMe. Las más actualizadas son las 2.7.5 y la 3.10 En la versión 3, existen diferencias importantes.

- Versión 1.4, y superior, funciona con cualquier versión de Python superior a la 2.5 hasta la 2.7.
- El soporte opcional para GIS (Sistemas de Información Geográfica) requiere Python 2.5 a 2.7.
- La versión más reciente en la serie 2.x, la versión 2.7. y versión 3.10.

EL COMANDO VIRTUALENV

Los entornos virtuales de Python, administrados por `virtualenv`, están configurados para instalar paquetes y ejecutar programas de una manera que los aísle de otros paquetes instalados en el resto del sistema. Debido a que cada entorno tiene su propio ejecutable de intérprete, y directorio para instalar paquetes, es fácil crear entornos configurados con varias combinaciones de Python y versiones de paquetes, todo en la misma computadora.

INICIANDO EL ENTORNO VIRTUAL

Un entorno virtual es una herramienta para mantener las dependencias requeridas por diferentes proyectos en lugares separados, mediante la creación de entornos virtuales de Python para ellos.

Éstos son lo suficientemente útiles como para ser usados en cada proyecto, pues permiten:

- Gestionar dependencias sin necesidad de acceso root.
- Instalar diferentes versiones de la misma dependencia. Por ejemplo: al trabajar en diferentes proyectos, con diferentes requisitos.
- Trabajar con diferentes versiones de Python.

INICIANDO UN ENTORNO VIRTUAL:

Se crea un entorno, como por ejemplo:

```
1 $ virtualenv ENTORNO
```

Se sustituye `"ENTORNO"`, por el nombre del entorno a utilizar. Se utiliza la opción `-p` para indicar la versión de python a utilizar:

```
1 $ virtualenv ENTORNO -p /usr/bin/python2.7
```

También se podría usar la opción **"-no-site-packages"**, que nos aísla aún más del sistema de paquetes del sistema. Para activar el entorno virtual se ejecuta el comando **source** sobre uno de los archivos de nuestro entorno.

```
$ source ENTORNO/bin//activate
```

SALIENDO DEL ENTORNO VIRTUAL

Para salir del entorno que estamos ejecutando, simplemente ejecutamos la siguiente instrucción:

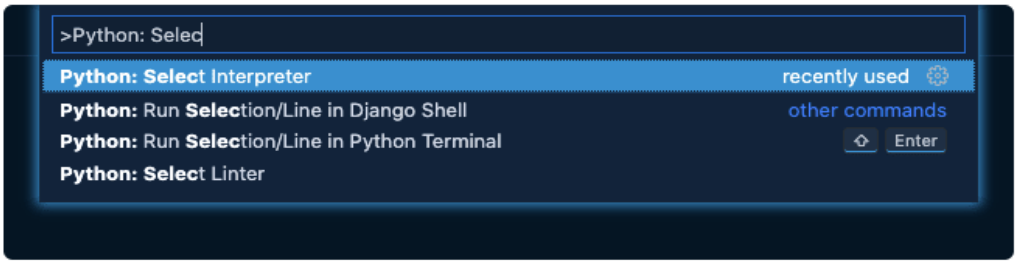
```
(entorno2)$ deactivate
```

INSTALACIONES EN EL ENTORNO GLOBAL

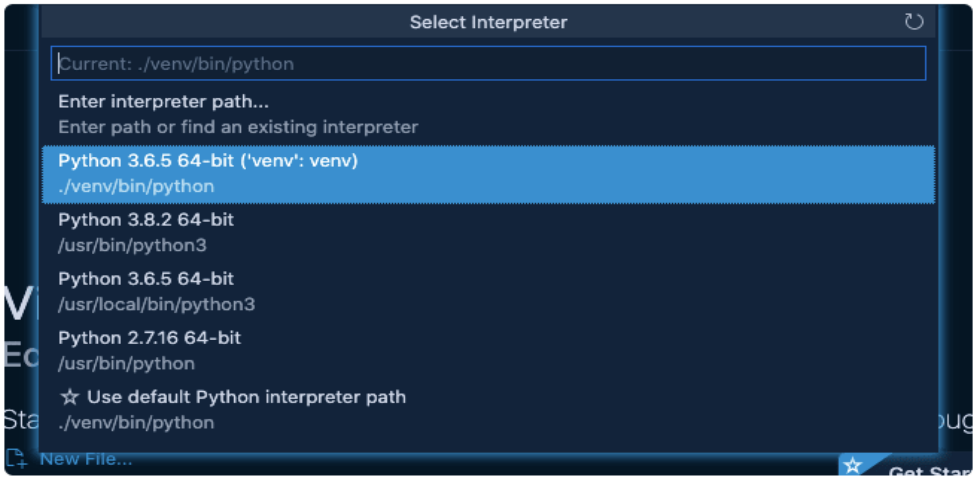
Las variables de entorno global se establecen en los archivos profile o /etc/ profile como variables globales, y pueden ser accedidos por cualquier proceso o subproceso activado por el terminal de inicio de sesión. Todo el código de un proyecto de Python se ejecuta dentro del contexto de un entorno concreto.

Para un proyecto Python, y haciendo uso de Visual Studio Code, tenemos que seleccionar el intérprete de Python que se utilizará para ejecutar el proyecto. Podemos crear un entorno virtual, e indicar a VS Code la versión de Python con la que se cuenta dentro del entorno virtual, junto con todos sus paquetes instalados.

Dentro de VS Code, vamos a la opción View > Command Palette, y seleccionamos el comando Python: Select Interpreter.



El comando mostrará una lista de los intérpretes disponibles que VS Code puede localizar automáticamente. Desde la lista, selecciona el entorno virtual que se encuentra dentro de la carpeta del proyecto:



Por el momento se ha creado el entorno virtual, aún no lo hemos activado. Iniciamos la línea de comandos de VS Code con la opción Terminal > New Terminal. Esto abre una consola integrada en nuestro editor, y automáticamente activa el entorno virtual ejecutando el script de activación.

EL ENRUTADOR DE DJANGO

El enrutamiento es el proceso de mapear la lógica a un conjunto de URL. El marco REST agrega soporte para enrutamiento de URL automático a Django. Las sintaxis del enrutador serían las siguientes:

```
1 enrutador = enrutadores.SimpleRouter()
```

`router.register` (prefijo, conjunto de vistas)

`router.urls #` el conjunto generado de urls para el conjunto de vistas registrado.

MVC EN DJANGO PARA APLICACIONES MONOLÍTICAS

Una aplicación monolítica es un sistema de aplicaciones, en el que todos los módulos relevantes se empaquetan como una única unidad de ejecución que se puede implementar.

En MVC separamos toda la lógica de la aplicación en tres componentes:

- Modelos (Datos y Lógica): representa y mantiene los datos de la aplicación en la base de datos. Cómo se almacena la información, y cómo se puede recuperar.
- Vistas (interfaz de usuario): muestra los datos que utilizan el modelo al usuario, como una salida o una GUI.
- Controlador (administrador de requests): maneja la solicitud del usuario, y actúa como una interfaz entre modelos y vistas.

Django utiliza el MVC para aplicaciones monolíticas cuando comienza a crear una aplicación con una única base de código, los requisitos son bastantes simples, y el tráfico es limitado. Por ejemplo: la aplicación de cálculo de impuestos.

HERENCIA DE COMPONENTES EN EL MODELO MVC

La herencia facilita la creación de objetos a partir de otros ya existentes, obteniendo características (métodos y atributos) similares a ellos.

Es uno de los mecanismos de la programación orientada a objetos, por medio del cual una clase se deriva de otra, llamada entonces clase base o clase padre, (a veces se le denomina superclase o clase base), de manera que extiende su funcionalidad.

EL PRINCIPIO DRY Y SU APLICACIÓN EN EL ENTORNO

El principio No te repitas (en inglés Don't Repeat Yourself o DRY, también conocido como Una vez y sólo una) es una filosofía de definición de procesos que promueve la reducción de la duplicación, especialmente en programación. Según este principio, toda pieza de información nunca debería ser duplicada, debido a que ello incrementaría la dificultad en los cambios y evolución.

Con respecto al código, no se trata de que los pedazos de código NUNCA se deban repetir, pues a veces es necesario y menos complejo dejar que algunas líneas se repitan en diferentes clases; donde en realidad lo debes aplicar es cuando tratas de hacer representaciones funcionales de tu aplicación. Ejemplo: si tienes una rutina que va a recuperar los datos de cliente, y hacer cierta transformación a esos datos, y además lo harás a menudo en la aplicación, vale la pena hacer una función llamada (por ejemplo) "RecuperaClienteInfo".

En cuanto al principio DRY, a la hora de desarrollar éste es fundamental, entendiendo claramente que no podemos ser tan dogmáticos y que en alguno de los casos es necesario la duplicación de código, sobre todo cuando hacemos uso de Framework.

PYTHON/DJANGO

Python es un lenguaje que se utiliza comúnmente para crear prototipos y desarrollar aplicaciones rápidamente. Mientras que Django es un framework que da la velocidad y la potencia de Python, con diversas características integradas adicionales para ayudar a crear aplicaciones web y APIs mucho más rápido.

¿QUÉ SON LOS TEMPLATES DE DJANGO?

En HTML no se puede escribir código Python, pues los navegadores no lo entienden. Sólo saben HTML. Sabemos que HTML es bastante estático, mientras que Python es mucho más dinámico.

Las template tags de Django nos permiten comunicar elementos de Python a HTML, para que se puedan construir sitios web dinámicos de forma más rápida y fácil.

RENDERIZACIÓN EN DJANGO

Dentro de los template podemos hacer uso de sentencias Python, como lo pueden ser ciclos y/o condicionales. La renderización en Django se utiliza en las capas de plantillas, enviando la capa de datos a la capa de plantilla, luego renderizándola, y finalmente devolviendo la solicitud de respuesta a través de la función HttpResponse.