Documentación Django (Phyton)

Tabla de contenidos

Introducción

Tabla de contenidos

Django es un framework web de alto nivel de desarrollo web de código abierto, escrito en Python, que respeta el patrón de diseño conocido como modelo–vista–controlador (MVC).

El modelo-vista-controlador es un patrón de arquitectura de software, que separa los datos y la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones.

- **Modelo**: Es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio).
- **Vista**: Presenta el modelo en un formato adecuado para interactuar (usualmente la interfaz de usuario) y también puede encargarse de filtrar la entrada de datos y enviarla al modelo.
- **Controlador**: Responde a eventos (usualmente acciones del usuario) e invoca peticiones al modelo cuando se hace alguna solicitud de información (por ejemplo, editar un documento o un registro en una base de datos).

Instalación en Linux

Tabla de contenidos

Python viene instalado por defecto en la mayoría de distribuciones GNU/Linux. Para comprobar si lo tenemos instalado, abrimos una terminal y escribimos:

```
python3 --version
```

En el caso de que no lo tengamos instalado, podemos instalarlo desde los repositorios oficiales de nuestra distribución. En el caso de Ubuntu, escribimos:

```
sudo apt install python3
```

Una vez verificado que tenemos instalado Python podemos empezar a instalar Django primero debemos instalar python3.10-venv para poder crear entornos virtuales. Para ello, escribimos:

```
sudo apt install python3.10-venv
```

Una vez instalado, creamos un entorno virtual llamado mysite y lo activamos:

> [!NOTE] El entorno virtual se puede llamar como queramos, pero es recomendable llamarlo como el proyecto que vamos a crear. Y un entorno virtual es un entorno aislado donde podemos instalar paquetes de Python sin afectar al sistema operativo.
>
> Ademas es recomendable crear un entorno virtual para cada proyecto que vayamos a crear.

```
$ python3 -m venv mysite
$ source mysite/bin/activate
(mysite)$ pip install django
(mysite)$ python -m django --version
4.2.7
```

Explicación de los comandos

- **python3 -m venv django**: Creamos un entorno virtual llamado mysite.
- **source django/bin/activate**: Activamos el entorno virtual esto hay que hacerlo cada vez que queramos trabajar con Django.
- **pip install django**: Instalamos Django.
- **python -m django --version**: Comprobamos la versión de Django instalada.

Si en la terminal nos aparece (django) delante del nombre de nuestro usuario, significa que el entorno virtual está activado. En caso contrario, debemos activarlo con el comando source django/bin/activate. Para desactivar el entorno virtual, escribimos deactivate.

Creación de un proyecto

Tabla de contenidos

En primer lugar vamos a ver como crear un proyecto Django. Para ello, nos situamos en el directorio donde queremos crear el proyecto y escribimos:

```
(mysite)$ django-admin startproject mysite
```

Esto creará un directorio llamado mysite con la siguiente estructura:

```
mysite/
    manage.py
    mysite/
        __init__.py
        settings.py
        urls.py
        asgi.py
        wsgi.py
```

Explicación de los ficheros al crear un proyecto Django

Tabla de contenidos

- **manage.py**: Es un script que ayuda con la gestión del sitio. Con él podemos arrancar un servidor de desarrollo, crear aplicaciones, crear migraciones de la base de datos, etc.
- El directorio **mysite/** es un paquete de Python para nuestro proyecto.
- **settings.py**: Contiene la configuración del proyecto.
- **urls.py**: Contiene las definiciones de las URLs del proyecto.
- **wsgi.py**: Es un punto de entrada para los servidores web compatibles con WSGI para servir el proyecto.
- **asgi.py**: Es un punto de entrada para los servidores web compatibles con ASGI para servir el proyecto.

Configuración del fichero settings.py

En el fichero settings.py podemos configurar nuestro proyecto. En este fichero podemos configurar la base de datos, el idioma, la zona horaria, etc.

Por defecto el fichero settings.py viene configurado de la siguiente manera:

- **DATABASES**: Configuración de la base de datos que se va a utilizar en el proyecto. Por defecto se utiliza una base de datos sqlite llamada db.sqlite3.
- **INSTALLED_APPS**: La lista de las aplicaciones que tiene instalada el proyecto, por ejemplo vemos que se ha incluido la aplicación polls (polls.apps.PollsConfig). También tenemos una aplicación que nos permite tener un panel de control de la aplicación (django.contrib.admin). Y otras cuantas aplicaciones…
- **DEBUG**: True: Si está activo los errores que se produzcan en la aplicación se verán con todo lujo de detalles en el navegador. Si tenemos la aplicación en producción debería ser False.
- **ALLOWED_HOSTS**: Es una lista de cadenas que especifica los nombres de host válidos para el sitio.

Configuración del fichero urls.py

En el fichero urls.py podemos configurar las URLs de nuestro proyecto. En este fichero podemos añadir las URLs de las aplicaciones que vayamos creando.

Ejecución del servidor de desarrollo

Tabla de contenidos

Para arrancar el servidor de desarrollo, nos situamos en el directorio donde se encuentra el fichero manage.py y escribimos:

```
(mysite)$ python manage.py runserver
```

> [!CAUTION] Si al ejecutar el comando de **python manage.py runserver** nos aparece en la consola un mensaje de error como este:
>
> You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions. Run 'python manage.py migrate' to apply them.
>
> Esto significa que tenemos que aplicar las migraciones de la base de datos. Para ello, escribimos:
>
> ```
> (mysite)$ python manage.py migrate
> ```

> Las migraciones son como una versión controlada de tu base de datos, y Django las usa para crear, modificar y eliminar tablas y sus campos.
>
> Cuando creas modelos o cambias tus modelos existentes, Django genera estas migraciones. Sin embargo, estos cambios no se reflejan en tu base de datos hasta que apliques las migraciones.
>
> El mensaje te está indicando que debes aplicar estas migraciones para que tu proyecto funcione correctamente.

Ahora si todo a ido bien, podemos acceder a nuestro servidor de desarrollo desde un navegador web en la dirección http://127.0.0.1:8000/ en mi caso esa es la dirección, pero puede variar en función de la configuración de nuestro equipo. Si todo ha ido bien, veremos una página de bienvenida de Django.

Ahora bien si queremos que el servidor de desarrollo sea accesible desde cualquier dirección IP y ademas en un puerto determinado, escribimos:

```
(mysite)$ python manage.py runserver 0.0.0.0:8000
```

Instalación del IDE PyCharm

Tabla de contenidos

Ahora que tenemos todo lo necesario para desarrollar nuestra aplicacion web con Django vamos a instalar PyCharm que es un IDE multiplataforma creado por JetBrains. Es uno de los mejores IDEs para proyectos que utilizan el lenguaje de programación Python, primero debemos descargarlo desde la página oficial de JetBrains. Una vez descargado, nos situamos en el directorio donde se encuentra el fichero descargado y escribimos:

```
$ tar -xzf pycharm-2023.2.5.tar.gz
```

Esto creará un directorio llamado pycharm-2023.2.5 con la siguiente estructura:

```
pycharm-2023.2.5/
    bin/
    lib/
    jbr/
    license/
    plugins/
    help/
    debug-eggs/
```

Ahora para ejecutar PyCharm, nos situamos en el directorio pycharm-2023.2.5/bin y escribimos:

```
$ ./pycharm.sh
```

Esto nos abrirá una ventana de bienvenida de PyCharm, donde podemos crear un proyecto nuevo o abrir uno existente.

Es recomendable crear un alias para ejecutar PyCharm desde cualquier directorio. Para ello, abrimos una terminal y escribimos:

```
$ sudo nano ~/.bashrc
```

Esto nos abrirá el fichero .bashrc en el editor de texto nano. Ahora debemos añadir la siguiente línea al final del fichero:

```
alias pycharm="{tu sitio de instalacion}/pycharm-2023.2.5/bin/pycharm.sh"
```

Ahora guardamos los cambios y cerramos el editor de texto. Para que los cambios surtan efecto, escribimos:

```
$ source ~/.bashrc
```

Ahora podemos ejecutar PyCharm desde cualquier directorio escribiendo pycharm en la terminal.

Configuración de PyCharm

Tabla de contenidos

Una vez abierto PyCharm, nos aparecerá una ventana de bienvenida. En ella, podemos crear un proyecto nuevo o abrir uno existente. En nuestro caso, vamos a abrir el proyecto anteriormente creado **mysyte**. Para ello, abrimos la carpeta contenedora del proyecto.

Una vez abierto el proyecto, vamos a configurar el intérprete de Python. Para ello, nos situamos en **File > Settings > Project: mysite > Python Interpreter**. En el desplegable, seleccionamos el intérprete de Python que hemos instalado anteriormente.

Si no aparece en el desplegable que en mi caso me aparece así por que mi entorno virtual lo e llamado mysite donde hemos instalado Django **Python 3.10 (mysite)**, debemos añadirlo. Para ello, sin movernos de la ventana Python Interpreter le damos a **Add Interpreter** que debe aparecer al lado. En la ventana que se nos abre, seleccionamos **Add Local Interpreter** y pulsamos en **Existing**. En la siguiente ventana, seleccionamos el intérprete de Python que ya existe y tenemos instalado de antes.

Ahora vamos a crear un perfil para ello con que ejecutemos el programa nos saldra una ventana para añadir un perfil en el que tendremos que poner el host y el puerto en el que queremos que se ejecute el servidor de desarrollo de Django.

Lo guardamos y ya podemos ejecutar el servidor de desarrollo de Django desde PyCharm.