Introducción a Django

Antonio Espín Herranz

Introducción

- Introducción.
- Instalación de Django.
- Crear el proyecto con Django.
- Estructura del proyecto.
- Lanzar el sitio Web
- Ruta de instalación

Introducción

- Django es un framework para el desarrollo de aplicaciones Web.

- El frameWork nos ayuda a separar las distintas partes de la aplicación.
  - Por un lado, el acceso a la base de datos:
    - SQLite3, MySQL, Oracle, PostGreSQL, etc.
    - La más cómoda es SQLite3, ya que no necesitamos instalar nada.
    - Para las demás a parte del motor de la BD, necesitaremos librerías de Python para poder acceder y conectar con estas otras BD.
  - Por otro lado, la capa de presentación al cliente.
  - Hay un acoplamiento bajo entre cada una de las partes: esto facilita el mantenimiento y la reutilización.

- Este frameWork se basa en el patrón **MVT**, modelo – vista – template.

Instalación

- Desde una consola de administración:
  - **pip install Django==5.0.1 // pip install Django**
  - Para comprobar la instalación de Django, desde una consola de python:
  - **import django**

```
>>> import django
>>> django.__version__
'5.0.1'
>>> django.VERSION
(5, 0, 1, 'final', 0)
```

La versión oficial es:
5.0.1

  - Para **actualizar** una versión anterior de Django:
    - **pip install –U Django**
  - **Django 5 → requiere Python >= 3.10**

Otro sotfware

- Para trabajar con Django a parte del intérprete de python + el paquete de django.

- Necesitaremos una BD: SQLite3 o MySQL, …
  - En el caso de SQLite3, con tener el browser para consultar la BD es suficiente.
  - Para otras BD necesitaríamos el gestor de la BD y las librerías de Python.

- Y para trabajar con los proyectos:
  - Un navegador Web: Chrome, Mozilla, etc.
  - Un editor o IDE (tener en cuenta que el IDE reconozca el proyecto).
    - **Visual Studio Code es una buena opción.**
    - Instalar el plugin: **pylance**

Características de Django

- Aunque lo normal es que un sitio Web tengamos una BD, Django NO obliga a tener una BD (es una opción un poco rara, pero es posible).
  - Aunque esto si puede hacer que no tengamos disponibles ciertas características añadidas como suele ser la administración del sitio web.
    - Usuarios, grupos, etc. (van sobre la BD)

Crear el proyecto

- Desde una consola de MS-DOS:
  - Situarnos en el directorio donde queramos ubicar el proyecto.
  - Lanzar el script:
    - **django-admin.py startproject Misitio**
    - Misitio será el nombre del proyecto y así se nombrará la carpeta raíz del proyecto.
  - *Revisar variables de entorno*.

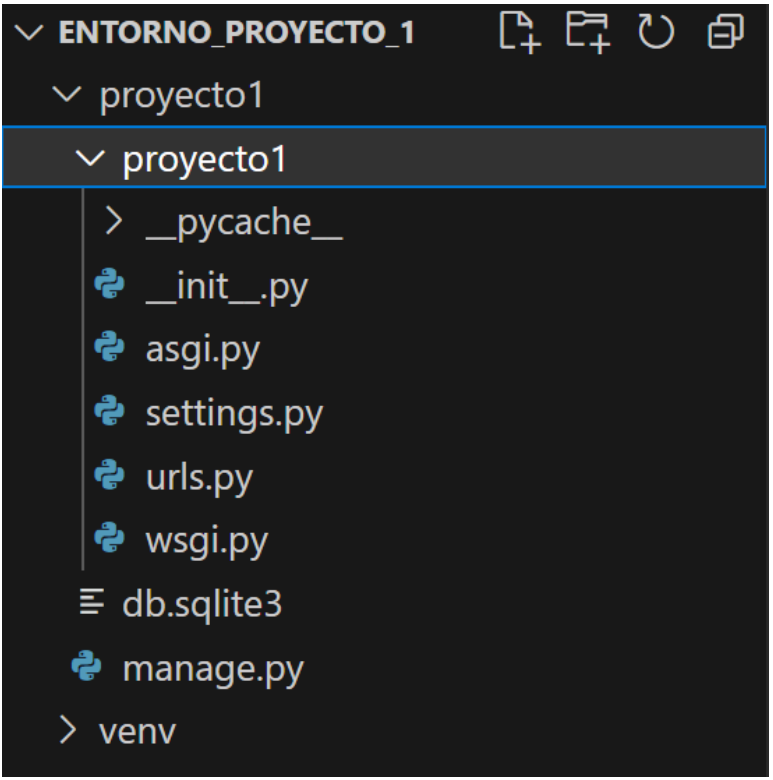
Variables de Entorno

- El programa: **django-admin** se copia en la carpeta: … **Python36-32**\Scripts

  (***carpeta fuente de Python***)

- Hay que añadir rutas a la variable de entorno **PATH**, con estas rutas funciona bien:
  - C:\Program Files (x86)\Python36-32\;
  - C:\Program Files (x86)\Python36-32\Tools\scripts;
  - C:\Program Files (x86)\Python36-32\Lib\site-packages

Estructura del proyecto



- En este caso el proyecto se llama **proyecto1**
  - Es la carpeta que contiene a todos los ficheros del proyecto.
  - El script **manage.py**, utilidad de línea de comandos que permite interactuar con un proyecto Django de varias formas.
  - Se puede utilizar con el parámetro **help**.
    - **python manage.py help**

Estructura del proyecto

- **proyecto1/proyecto1/**: El directorio interno misitio/ contiene el paquete Python para tu proyecto. El nombre de este paquete Python se usará para importar cualquier cosa dentro del. (Por ejemplo: import misitio.settings).

- **__init__.py:** Un archivo requerido para que Python trate el directorio app como un paquete o como un grupo de módulos. Es un archivo vacio y generalmente no se necesita agregarle nada.

- **settings.py**: Las opciones/configuraciones para nuestro proyecto Django. Tipos de configuraciones disponibles y sus valores predefinidos.
  - La configuración de la BD.
  - Las aplicaciones que tenemos instaladas.
    - Django configura opciones de seguridad y acceso a la aplicación de forma automática.

Estructura del proyecto II

- Dentro de las aplicaciones instaladas (que vienen por defecto en settings.py):
  - **django.contrib.admin** -- La interfaz administrativa.
  - **django.contrib.auth** -- El sistema de autentificación.
  - **django.contrib.contenttypes** -- Un framework para tipos de contenidos.
  - **django.contrib.sessions** -- Un framework. para manejar sesiones
  - **django.contrib.messages** -- Un framework para manejar mensajes
  - **django.contrib.staticfiles** -- Un framework para manejar archivos estáticos.

Estructura del proyecto III

- Algunas de estas aplicaciones hacen uso de la BD. Por ejemplo: almacenan grupos y usuarios en las tablas.
- Las tablas se crearán de forma automática cuando llamemos al comando:
  - **python manage.py migrate**
  - Este comando permite crear tablas en la BD de forma automática.
  - También lo utilizaremos para crear las tablas de nuestro modelo.

Estructura del proyecto IV

- **urls.py**: Declaración de las URLs para este proyecto de Django. Es como una ‘‘tabla de contenidos’’ de tu sitio hecho con Django.
  - Se declaran y se asocian peticiones de los clientes con vistas.
    - Se pueden utilizar expresiones regulares y comodines.

- **wsgi.py**: El punto de entrada WSGI para el servidor Web, encargado de servir nuestro proyecto.
  - Tiene que ver con la conexión con el servidor de Apache.
  - Web Server Gateway Interface

Estructura del proyecto V

- **asgi.py**: Interfaz de puerta de enlace de servidor asíncrono.
  - La invocación por defecto es asíncrona.

- Diferencias entre **WSGI** y **ASGI**:
  - https://medium.com/analytics-vidhya/difference-between-wsgi-and-asgi-807158ed1d4c

Lanzar el proyecto

- Antes de lanzar el proyecto tenemos que iniciar un servidor:

  python manage.py runserver

- Después desde un navegador:

  http://127.0.0.1/8000

- El servidor siempre se para con **Control+C**

Cambiar el puerto

- Por defecto, utiliza el puerto 8000, se puede cambiar al lanzar el comando:
  - ***python manage.py runserver 8080***
  - Y en el navegador: http://127.0.0.1:8080
    - Se lanza el localhost.
  - También se puede cambiar la dirección IP:
    - ***python manage.py runserver 192.148.1.103:8000***

Ruta de instalación

- **Es importante saber dónde tenemos instalado Django durante el curso nos hará falta saberlo**.
- Por ejemplo, para la personalización de plantillas.
- Tendremos que copiar plantillas HTML y alojarlas en nuestro proyecto para poder personalizarlas.
- Dentro de la carpeta de la instalación de Python tendremos:
  - …\…\Python36-32\Lib\site-packages**django**
  - **O dentro del entorno virtual …**

Práctica nº 1

Crear una aplicación

- Una vez hemos creado el proyecto y comprobamos que se despliega correctamente en el Servidor de desarrollo lo siguiente es crear una aplicación que se alojará dentro de la carpeta del proyecto.

- Django nos creará una estructura para esa aplicación añadiendo ficheros a los que iremos añadiendo nuestro código.

Crear una aplicación

- Navegamos dentro del proyecto, hasta tener visible el fichero **manage.py**
- Y lanzamos el comando:
  - ***python manage.py startapp nombre_app***
- Crea una estructura como esta:

Nombre_app

- __init__.py
- admin.py
- apps.py
- migrations/
  - __init__.py
- models.py
- tests.py
- views.py

Estructura de la Aplicación

- **models.py**
  - Se definen las clases del modelo que heredan de **models.Model**
  - Y se especifican las relaciones entre ellas. Uno a muchos, muchos a muchos.
  - A partir de estas clases django realiza modificaciones automáticamente en la BD (a esto lo llama **migraciones**).

- **admin.py**
  - Tiene que ver con la administración del sitio Web.
  - Que datos y como los queremos organizar en el panel de administración (sería la parte privada del sitio, donde se gestionan los datos que se mostrarán en la parte publica).
  - Se representan por clases que heredan:
    - **admin.ModelAdmin**

Estructura de la Aplicación II

- **views.py**
  - Las vistas de nuestra aplicación.
  - Se pueden definir como **funciones** o **clases**.
  - Básicamente desde una vista, accederemos al modelo para recuperar datos, crearemos un **contexto** (representado por un **dict**) y se pasará a una **plantilla** que dará como una página HTML.

- **tests.py:**
  - Se pueden definir test para **testear la aplicación**.
  - Se especificarán en este fichero.

Estructura de la Aplicación III

- **urls.py**
  - Se define la tabla de contenidos del sitio.
  - Ante una petición se indica la vista que responde.

- **apps.py**
  - Las aplicaciones que tenemos definidas. Este fichero se rellena automáticamente al crear la aplicación.

- La carpeta **migrations**:
  - Almacena todos los cambios que se han ido produciendo en las clases de modelo.

- También puede aparecer una carpeta **static**:
  - Con contenidos estáticos del sitio: CSS, imágenes, etc.

- **Templates**:
  - Otra carpeta donde podemos almacenar plantillas.

Estructura de la Aplicación IV

- En el desarrollo de la aplicación en **django** tendremos que trabajar en los ficheros:
  - urls.py,
  - models.py,
  - admin.py,
  - views.py

Proyecto vs Aplicación

- Diferencias entre proyecto y aplicación:
  - **Un proyecto**: es una instancia de un cierto conjunto de aplicaciones Django. El único requisito es que exista un fichero settings.py para la configuración. Que define los parámetros de la BD, las aplicaciones instaladas, etc.
    - Dentro de un mismo proyecto podemos tener distintas aplicaciones definidas como partes desacopladas para reutilizarlas en otros proyectos.
  - **Una aplicación**: es una funcionalidad que incluye modelos y vistas.

Resumen de Comandos

- Después de configurar el entorno de ejecución.
- Crear el proyecto:
  - ***django-admin startproject nombre_proyecto***
- Crear la aplicación:
  - Dentro del proyecto, lanzar el comando:
    - ***cd nombre_proyecto***
    - ***python manage.py startapp nombre_app***
- Registrar la aplicación en el fichero **settings.py**
  - En **INSTALLED_APPS**
- Crear las tablas de administración:
  - ***python manage.py migrate***

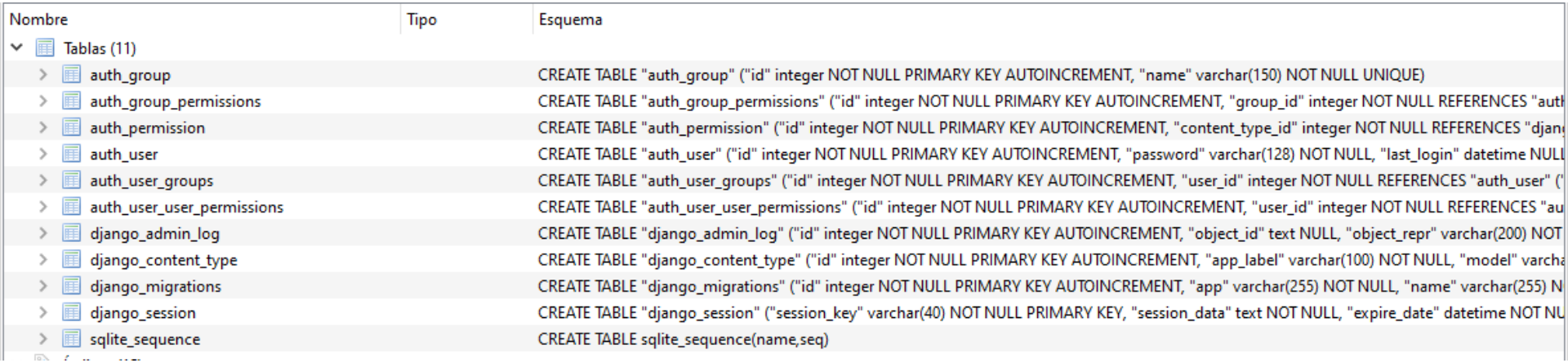
Migraciones

- Al ejecutar el comando:
  - **python manage.py migrate**
  - Veremos algo parecido a esto:

```
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
```

Tablas

- Por defecto crea todas estas tablas:



Nombre	Tipo	Esquema
Tablas (11)		
auth_group		CREATE TABLE "auth_group" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "name" varchar(150) NOT NULL UNIQUE)
auth_group_permissions		CREATE TABLE "auth_group_permissions" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "group_id" integer NOT NULL REFERENCES "auth
auth_permission		CREATE TABLE "auth_permission" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "content_type_id" integer NOT NULL REFERENCES "djan
auth_user		CREATE TABLE "auth_user" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "password" varchar(128) NOT NULL, "last_login" datetime NULL
auth_user_groups		CREATE TABLE "auth_user_groups" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "user_id" integer NOT NULL REFERENCES "auth_user" (
auth_user_user_permissions		CREATE TABLE "auth_user_user_permissions" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "user_id" integer NOT NULL REFERENCES "au
django_admin_log		CREATE TABLE "django_admin_log" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "object_id" text NULL, "object_repr" varchar(200) NOT
django_content_type		CREATE TABLE "django_content_type" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "app_label" varchar(100) NOT NULL, "model" varcha
django_migrations		CREATE TABLE "django_migrations" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "app" varchar(255) NOT NULL, "name" varchar(255) N
django_session		CREATE TABLE "django_session" ("session_key" varchar(40) NOT NULL PRIMARY KEY, "session_data" text NOT NULL, "expire_date" datetime NOT NU
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)

Son las tablas de autenticación y aparte se crearán otras tablas que estén relacionadas con la aplicación (***cuando vayamos especificando los objetos del modelo***)