Guia de Instalacion de Django con Pipenv

Este documento proporciona una guia paso a paso con formato detallado y explicaciones precisas para instalar Django en un entorno virtual Pipenv en Windows. Incluye configuracion de variables de entorno, gestion del entorno local vs global y procedimientos de revision para cambiar la ubicacion del entorno virtual.

1. Antecedentes y Objetivo

En proyectos de Python, es vital aislar dependencias y versiones de librerias para evitar conflictos entre proyectos. Pipenv combina las funcionalidades de virtualenv y pip, proporcionando un entorno virtual dedicado por proyecto y un control de versiones de dependencias mediante Pipfile y Pipfile.lock.

Esta guia muestra:

- Como instalar y configurar Pipenv.

- Como crear un entorno local dentro del proyecto (.venv) o mantenerlo en ubicacion global.

- Paso a paso de instalacion de Django en dicho entorno.

- Instrucciones de revision para regresar a la situacion global.

- Buenas practicas de versionado con Git y GitHub.

2. Requisitos Previos

Antes de comenzar, asegurese de cumplir con los siguientes requisitos:

1. Tener instalado Python 3.11 (o superior) y anadido al PATH del sistema.

   - Verifique con: python --version

   - En caso de no estar, descargue desde https://www.python.org/downloads/windows y marque "Add Python to PATH".

2. Tener instalado Git y anadido al PATH.

   - Verifique con: git --version

   - En caso contrario, instale desde https://git-scm.com/download/win siguiendo las opciones por defecto.

3. Instalacion y Verificacion de Pipenv

3.1. Instalacion de Pipenv a nivel de usuario:

1. Abra PowerShell (sin privilegios de administrador).

2. Ejecute:

python -m pip install --user pipenv

- "python -m pip" asegura usar el pip de la version de Python en PATH.

- "--user" instala Pipenv en la carpeta del usuario (%APPDATA%), evitando permisos elevados.

3. Verifique la instalacion:

pipenv --version

- Debera mostrar un numero de version. Si no, revise PATH y reinicie PowerShell.

3.2. Razonamiento:

- Instalar Pipenv en modo usuario mantiene la instalacion limpia y separada del Python global del sistema, minimizando riesgos al actualizar paquetes globales.

4. Creacion de la Carpeta del Proyecto

4.1. Seleccion de ubicacion:

- Una buena practica es tener una carpeta central para proyectos, por ejemplo: C:\Users\<TuUsuario>\Projects.

4.2. Comandos:

1. Navegue a la ubicacion deseada:

cd C:\Users\<TuUsuario>\Projects

2. Cree la carpeta del proyecto:

mkdir ChatBot_Django

3. Ingrese en la carpeta:

cd ChatBot_Django

- Ahora su carpeta de proyecto es C:\Users\<TuUsuario>\Projects\ChatBot_Django.

5. Configuracion de la Variable PIPENV_VENV_IN_PROJECT

5.1. Objetivo de la variable:

- Por defecto, Pipenv crea entornos virtuales en una carpeta global (p.ej. %USERPROFILE %\.virtualenvs\). Esto puede dispersar entornos y dificultar el versionado de proyectos.

- Al activar PIPENV_VENV_IN_PROJECT, indicamos a Pipenv que ubique el entorno virtual dentro de la carpeta del proyecto, en '.venv'.

5.2. Comandos:

1. En PowerShell (en una sesion nueva), ejecute:

setx PIPENV_VENV_IN_PROJECT 1

- Esto guarda la variable de entorno en el registro de Windows, en el ambito de usuario.

2. Cierre la ventana de PowerShell y abra una nueva.

3. Verifique que la variable exista:

echo $Env:PIPENV_VENV_IN_PROJECT

- Debera mostrar '1'. Si no, repita el comando setx y reabra PowerShell.

5.3. Explicacion tecnica:

- 'setx' almacena permanentemente en HKEY_CURRENT_USER\Environment.

- Las nuevas sesiones de PowerShell leen estas variables al iniciar.

- Pipenv chequea esta variable para definir la ruta del entorno virtual.

6. Eliminacion de Entornos Pipenv Previos

6.1. Verificar si existe entorno anterior:

- Antes de crear un nuevo entorno local, es importante eliminar cualquier entorno global previamente generado para este proyecto.

6.2. Comando:

pipenv --rm

- Este comando borra el entorno virtual que Pipenv asoció a la carpeta actual.

- Si el mensaje indica 'No virtualenv has been created for this project yet!', significa que no habia ningun entorno asociado.

6.3. Consecuencia:

- Se asegura que la proxima invocacion de Pipenv creara un nuevo entorno en '.venv'.

7. Creacion del Entorno Virtual con Python 3.11

7.1. Generar Pipfile y entorno:

1. En la carpeta del proyecto:

pipenv --python 3.11

- Esto crea un Pipfile con la seccion [requires] indicando 'python_version = "3.11"'.

- Tambien inicia la creacion del entorno virtual en './.venv' (gracias a la variable configurada).

2. Verifique la estructura:

C:\Users\<TuUsuario>\Projects\ChatBot_Django\

Pipfile

.venv\

Pipfile.lock (se crea tras instalar dependencias)

7.2. Razonamiento:

- Especificar la version de Python garantiza consistencia en el equipo de trabajo.

- El Pipfile ahora define estrictamente que el proyecto usa Python 3.11.

8. Instalacion de Django

8.1. Comando para instalar:

pipenv install django

- Al ejecutar este comando:

- Pipenv instala Django (ultima version estable) en el entorno local '.venv'.

- Agrega 'django = "*"' bajo [packages] en el Pipfile.

- Se genera/actualiza Pipfile.lock con versiones exactas de Django y dependencias.

8.2. Por que usar Pipfile.lock:

- Pipfile.lock registra versiones exactas (hashes) de todas las dependencias transitivas, facilitando la reproducibilidad en otros equipos.

- Para instalar exactamente las mismas versiones en otra maquina, basta usar 'pipenv install'.

9. Activacion del Entorno Virtual

9.1. Comando:

pipenv shell

- Esto abre un subshell con el entorno '.venv' activo.

- El prompt cambia a '(.venv) PS C:\Users\<TuUsuario>\Projects\ChatBot_Django'.

9.2. Verificacion de ubicacion de Python:

where python

- Debe apuntar a 'C:\Users\<TuUsuario>\Projects\ChatBot_Django\.venv\Scripts\python.exe'.

python --version debe mostrar 'Python 3.11.x'.

9.3. Comandos disponibles:

- Ahora puede ejecutar 'django-admin', 'manage.py', 'pip install' adicional, etc., dentro del entorno aislado.

10. Creacion del Proyecto Django

10.1. Con el entorno activo, ejecute:

django-admin startproject mysite .

- El punto '.' indica que el esquema principal de Django se cree en la carpeta actual.

10.2. Estructura resultante:

ChatBot_Django\

Pipfile

Pipfile.lock

.venv\

Scripts\

Lib\

...

manage.py

mysite\

__init__.py

settings.py

urls.py

wsgi.py

10.3. Primera prueba:

python manage.py runserver

- Abrir en navegador http://127.0.0.1:8000

- Validar que aparece la pagina de bienvenida de Django.

11. Configuracion de Git y .gitignore

11.1. Crear .gitignore en la raiz con estas reglas:

.venv/

__pycache__/

*.py[cod]

mysite/settings.py

- Esto evita subir el entorno y archivos temporales.

11.2. Inicializar repositorio Git:

git init

- Crea carpeta .git/ y prepara el versionado.

11.3. Primer commit:

git add .

git commit -m "Inicializar proyecto Django con Pipenv"

- Se guardan en Git todos los archivos excepto los ignorados.

11.4. Vincular con GitHub:

1. En GitHub, crear un nuevo repositorio llamado ChatBot_Django.
   - No agregar README ni .gitignore desde la web.
2. En PowerShell, ejecutar:

   git remote add origin https://github.com/TuUsuario/ChatBot_Django.git

   git branch -M main

   git push -u origin main

- Ahora el proyecto local esta vinculado al remoto y se suben todos los archivos.

12. Revertir la Creacion de Entorno en .venv

12.1. Para volver a la ubicacion global de entornos en caso de ser necesario:

1. Abra una nueva ventana de PowerShell.
2. Ejecute:

   setx PIPENV_VENV_IN_PROJECT ""

   - Esto borra el valor de la variable, dejandola vacia.
3. Cierre y abra una nueva ventana de PowerShell.
4. En la carpeta del proyecto, elimine la carpeta .venv:

   pipenv --rm

   - Borra el entorno actual en .venv.

5. La proxima vez que ejecute:

pipenv --python 3.11

pipenv install

- Pipenv creara el entorno en la ubicacion global: %USERPROFILE %\.virtualenvs\ChatBot_Django-<hash>

12.2. Razonamiento:

- Revertir esta configuracion puede ser util si desea liberar espacio en la carpeta del proyecto o seguir la convencion de entornos centralizados.

13. Flujo de Trabajo Continuo

13.1. Cada dia que trabaje en el proyecto:

1. Ingrese a la carpeta del proyecto:

cd C:\Users\<TuUsuario>\Projects\ChatBot_Django

2. Active el entorno:

pipenv shell

- El prompt mostrara '(.venv)' indicando entorno activo.

3. Realice cambios en codigo, modelos, vistas, templates, etc.

4. Si necesita agregar una libreria nueva:

pipenv install <paquete>

- Pipfile y Pipfile.lock se actualizan automaticamente.

5. Versione sus cambios:

git add .

git commit -m "Mensaje descriptivo de cambios"

git push

6. Para salir del entorno cuando termine:

exit

- Regresa al PowerShell normal.

14. Buenas Practicas y Recomendaciones

14.1. Mantenga actualizados Pipfile y Pipfile.lock:

- Cada vez que instale o actualice paquetes, haga commit de ambos archivos.

14.2. En entornos de produccion, use:

pipenv install --deploy --ignore-pipfile

- Esto fuerza el uso de Pipfile.lock y evita cambios no deseados.

14.3. Documente versiones de Python y dependencias en README:

- Ejemplo:

Requisitos

- Python 3.11

- Pipenv

- Django 4.x

- Facilita onboarding de nuevos colaboradores.

14.4. No suba .venv ni carpetas de entorno a GitHub:

- Asegurese de que .venv este en .gitignore para reducir tamano del repo.

14.5. Use variables de entorno para configuraciones sensibles:

- No almacene SECRET_KEY, credenciales o configuraciones privadas en settings.py.

- Utilice un archivo .env aparte y python-dotenv o similar para cargarlas.

15. Resumen Ejecutivo

- Este documento cubre la instalacion de Django usando Pipenv en Windows.

- PIPENV_VENV_IN_PROJECT = 1 -> entorno en .venv (en el propio proyecto)

- Para revertir: setx PIPENV_VENV_IN_PROJECT "" + pipenv --rm.

- Pipfile y Pipfile.lock mantienen control de versiones de dependencias.

- Git: ignorar .venv/, versionar solo codigo y archivos de configuracion.

- Buenas practicas garantizan reproducibilidad y facil colaboracion en el equipo.