Modulo 6: Desarrollo de Aplicaciones Web con Python Django
ABPro - Ejercicio Individual 1

Adolfo García P.

13 - 07 - 23

Aprendizaje esperado:

- Reconocer los principales componentes de un proyecto desarrollado en Django

1 Creación de un entorno virtual.

```
adolfofgp@DESKTOP-LTK64MK:/mnt/c/users/92fel/onedrive/escritorio/talentodigital/modulo6
/m6_individual$ virtualenv env6individual
created virtual environment CPython3.8.10.final.0-64 in 19544ms
  creator CPython3Posix(dest=/mnt/c/users/92fel/onedrive/escritorio/talentodigital/modu
lo6/m6_individual/env6individual, clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=c
opy, app_data_dir=/home/adolfofgp/.local/share/virtualenv)
    added seed packages: pip==23.1.2, setuptools==67.8.0, wheel==0.40.0
  activators BashActivator,CShellActivator,FishActivator,NushellActivator,PowerShellAct
ivator,PythonActivator
adolfofgp@DESKTOP-LTK64MK:/mnt/c/users/92fel/onedrive/escritorio/talentodigital/modulo6
/m6_individual$ ls
env6individual
adolfofgp@DESKTOP-LTK64MK:/mnt/c/users/92fel/onedrive/escritorio/talentodigital/modulo6
/m6_individual$ source env6individual/bin/activate
```

2 Instalación de Django y revisión de versiones.

```
(env6individual) adolfofgp@DESKTOP-LTK64MK:/mnt/c/users/92fel/onedrive/escritorio/talen
todigital/modulo6/m6_individual$ pip install django
Collecting django
  Using cached Django-4.2.3-py3-none-any.whl (8.0 MB)
Collecting asgiref<4,>=3.6.0 (from django)
  Using cached asgiref-3.7.2-py3-none-any.whl (24 kB)
Collecting sqlparse>=0.3.1 (from django)
  Using cached sqlparse-0.4.4-py3-none-any.whl (41 kB)
Collecting backports.zoneinfo (from django)
  Using cached backports.zoneinfo-0.2.1-cp38-cp38-manylinux1_x86_64.whl (74 kB)
Collecting typing-extensions>=4 (from asgiref<4,>=3.6.0->django)
  Using cached typing_extensions-4.7.1-py3-none-any.whl (33 kB)
Installing collected packages: typing-extensions, sqlparse, backports.zoneinfo, asgiref
, django
Successfully installed asgiref-3.7.2 backports.zoneinfo-0.2.1 django-4.2.3 sqlparse-0.4
.4 typing-extensions-4.7.1
(env6individual) adolfofgp@DESKTOP-LTK64MK:/mnt/c/users/92fel/onedrive/escritorio/talen
todigital/modulo6/m6_individual$ python -m django --version
4.2.3
(env6individual) adolfofgp@DESKTOP-LTK64MK:/mnt/c/users/92fel/onedrive/escritorio/talen
todigital/modulo6/m6_individual$ python --version
Python 3.8.10
(env6individual) adolfofgp@DESKTOP-LTK64MK:/mnt/c/users/92fel/onedrive/escritorio/talen
todigital/modulo6/m6_individual$
```

3 Identifique qué paquetes se descargan automáticamente. Investigue la utilidad que tienen estos paquetes.

```
(env6individual) adolfofgp@DESKTOP-LTK64MK:/mnt/c/users/92fel/onedrive/escritorio/talen
todigital/modulo6/m6_individual$ pip freeze
asgiref==3.7.2
backports.zoneinfo==0.2.1
Django==4.2.3
sqlparse==0.4.4
typing_extensions==4.7.1
```

asgiref: es una biblioteca que proporciona una abstracción de nivel superior para manejar las comunicaciones entre servidores web y aplicaciones web en Python. Django utiliza asgiref para gestionar las solicitudes y respuestas HTTP.

backports.zoneinfo: es un paquete que proporciona una implementación en Python de la zona horaria de la biblioteca estándar zoneinfo introducida en Python 3.9. Django utiliza backports.zoneinfo para manejar las zonas horarias en versiones anteriores de Python.

sqlparse: es una biblioteca que analiza y formatea consultas SQL. Django utiliza sqlparse para mejorar la legibilidad de las consultas SQL generadas por el ORM de Django.

typing_extensions: es una biblioteca que proporciona extensiones al módulo typing incorporado en Python. Django utiliza typing_extensions para anotar el código y proporcionar mejoras en la verificación de tipos.

4 ¿Qué facilidades nos proporciona Django?

Arquitectura MVC: Django utiliza el patrón de diseño Modelo-Vista-Controlador (MVC), lo que ayuda a separar la lógica de negocio, la presentación y el manejo de datos en diferentes componentes, lo que facilita la organización y mantenimiento del código.

ORM integrado: Django incluye un Object-Relational Mapping (ORM) integrado que permite interactuar con la base de datos utilizando código Python en lugar de SQL directamente. Esto facilita la creación, modificación y consulta de datos en la base de datos, y proporciona una capa de abstracción para trabajar con diferentes motores de base de datos.

Sistema de enrutamiento de URLs: Django tiene un sistema de enrutamiento de URLs que permite mapear URLs a vistas y controladores específicos. Esto simplifica la gestión de las rutas en una aplicación web y facilita la navegación entre diferentes páginas y funcionalidades.

Plantillas: Django proporciona un sistema de plantillas que permite separar la lógica de presentación del código Python. Esto facilita la creación de interfaces de usuario dinámicas y reutilizables, ya que se pueden definir plantillas HTML con marcadores especiales para la inserción de datos dinámicos.

Administrador de Django: Django incluye un administrador web automático que genera una interfaz de administración CRUD (Crear, Leer, Actualizar, Eliminar) para los modelos de la base de datos. Esto proporciona una forma rápida y sencilla de administrar los datos de la aplicación sin necesidad de escribir código adicional.

Seguridad integrada: Django incorpora características de seguridad como protección contra ataques de inyección de SQL, ataques de cross-site scripting (XSS), falsificación de solicitudes entre sitios (CSRF) y más. Estas características ayudan a proteger las aplicaciones web desarrolladas con Django contra las vulnerabilidades comunes de seguridad.

5 Con relación al levantamiento de un servidor. ¿Existe una forma de realizarlo con Python y sin Django? ¿Qué desventajas nos trae este tipo de proyectos sin Django?

Sí, es posible levantar un servidor utilizando Python sin Django, pero es importante considerar las siguientes desventajas:

Mayor tiempo de desarrollo: Al prescindir de un framework como Django, es posible que tengas que escribir más código personalizado para manejar aspectos como el enrutamiento de URLs, la gestión de formularios, la interacción con la base de datos, la autenticación de usuarios, entre otros. Esto puede requerir más tiempo y esfuerzo de desarrollo en comparación con el uso de funcionalidades ya implementadas en un framework.

Reinvención de la rueda: Los frameworks como Django han sido diseñados y probados para abordar una amplia gama de problemas comunes en el desarrollo web. Al prescindir de ellos, podrías tener que implementar soluciones propias para problemas ya resueltos en el contexto del desarrollo web, lo que puede llevar a una mayor complejidad y posibles errores.

Falta de funcionalidades integradas: Django viene con un conjunto de funcionalidades integradas que facilitan el desarrollo web, como el sistema de administración, el ORM, el enrutamiento de URLs, la gestión de formularios y la seguridad. Al construir un proyecto sin Django, tendrías que buscar y utilizar bibliotecas adicionales o escribir tu propio código para implementar estas funcionalidades.

Menor estandarización: Django promueve una estructura y un flujo de trabajo bien definidos, lo que facilita que los desarrolladores se familiaricen rápidamente con el código y puedan colaborar en proyectos de Django sin problemas. Al prescindir de Django, es posible que cada proyecto tenga su propia estructura y convenciones, lo que puede dificultar la colaboración y mantenimiento a largo plazo.

6 Levante un servidor utilizando Python. Debe basarse en el ejercicio realizado en las capsulas.

Muestre un mensaje que indique "Servidor levantado mediante http.server".

Ver archivo 'servidor.py' y 'landing.html'

7 Por último, debes crear un proyecto en Django. Indague en la utilidad de django-admin

startproject

```
todigital/modulo6/m6_individual$ django-admin startproject proy_individual
(env6individual) adolfofgp@DESKTOP-LTK64MK:/mnt/c/users/92fel/onedrive/escritorio/talen
todigital/modulo6/m6_individual$
```

Cuando se ejecuta el comando django-admin startproject <nombre_del_proyecto>, Django generará automáticamente una estructura de directorios y archivos base para tu proyecto. Algunas de las utilidades de django-admin startproject:

Creación de la estructura del proyecto: El comando startproject crea una estructura de directorios básica para tu proyecto de Django. Esto incluye el archivo manage.py, que es un script que te permite realizar diversas tareas de administración del proyecto, y un directorio con el mismo nombre que especificaste para el proyecto, que contendrá los archivos y configuraciones principales de tu proyecto.

Configuración inicial: El comando también genera un archivo llamado settings.py, que es donde se configuran los aspectos fundamentales de tu proyecto, como la base de datos, las aplicaciones instaladas, la configuración de idioma y tiempo, y más. Aquí puedes personalizar y ajustar la configuración según las necesidades de tu proyecto.

Ambiente virtual: Además, cuando ejecutas startproject, es una buena práctica hacerlo dentro de un entorno virtual de Python. De esta manera, se creará un entorno aislado donde puedes instalar y gestionar las dependencias específicas de tu proyecto sin interferir con otros proyectos. Esto te permite mantener un entorno limpio y organizado para tu desarrollo.

Configuración de la aplicación de administración: El comando startproject también incluye la configuración básica para la aplicación de administración de Django. Esta aplicación proporciona una interfaz de administración preconstruida y fácil de usar para gestionar los datos en tu proyecto.