Investigación sobre Framework de Django

Django es sencillamente una colección de bibliotecas escritas en el lenguaje de programación Python. Para desarrollar un sitio usando Django escribes código Python que utiliza esas bibliotecas. Django es un miembro importante de una nueva generación de frameworks Web.

Django nació naturalmente de aplicaciones de la vida real escritas por un equipo de desarrolladores Web en Lawrence, Kansas. Nació en el otoño boreal de 2003, cuando los programadores Web del diario Lawrence Journal-World, Adrian Holovaty y Simon Willison, comenzaron a usar Python para crear sus aplicaciones

Estructura:

El archivo **models.py** contiene una descripción de la tabla de la base de datos, como una clase Python. A esto se lo llama el modelo. Usando esta clase se pueden crear, buscar, actualizar y borrar entradas de tu base de datos usando solo código Python en lugar de escribir declaraciones SQL repetitivas.

El archivo **views.py** contiene la lógica de la página, en la función ultimos_libros() . A esta función se la denomina vista.

El archivo **urls.py** específica qué vista es llamada según el patrón URL. En este caso, la URL /ultimos_libros/ será manejada por la función ultimos_libros(). En otras palabras, si el nombre de nuestro dominio es example.com, cualquier visita a la URL http://example.com/ultimos_libros/ llamara a la función ultimos_libros().

El archivo ultimos_libros.html es una **plantilla HTML** especial, que describe el diseño de la página. Usa el lenguaje de plantillas de Django, con declaraciones básicas y lógicas por ejemplo: {% for libro in lista_libros %}.

Una ventaja clave de este enfoque es que los componentes tienen un acoplamiento débil entre sí. Eso significa que cada pieza de la aplicación Web que funciona sobre Django tiene un único propósito clave, que puede ser modificado independientemente sin afectar las otras piezas.

Como primer paso se deberá crear un **proyecto,** este proyecto deberá ubicarse dentro de un **directorio.**

Es importante que una vez creando el directorio deberá generar un ambiente virtual $**virtualenv nombreEspacioVirtual** y enseguida activarlo $**source nombreEspacioVirtual/bin/activate,** para después instalar Django con $**pip install Django.**

Ahora se deberán seguir los siguientes pasos:

1. **Django-admin.py startproject misitio**

   El comando **startproject** crea un directorio de trabajo que contiene varios archivos:

```
misitio/
    manage.py
    misitio/
        __init__.py
        settings.py
        urls.py
        wsgi.py
```

   Estos archivos son los siguientes:

   misitio/: El directorio de trabajo externo misitio/, es solo un contenedor, es decir una carpeta que contiene nuestro proyecto. Por lo que se le puede cambiar el nombre en cualquier momento sin afectar el proyecto en sí.

   manage.py: Una utilidad de línea de comandos que te permite interactuar con un proyecto Django de varias formas. Usa manage.py help para ver lo que puede hacer. No deberías editar este archivo, ya que este es creado en el directorio convenientemente para manejar el proyecto.

   misitio/misitio/: El directorio interno misitio/ contiene el paquete Python para tu proyecto. El nombre de este paquete Python se usara para importar cualquier cosa dentro del.

   __init__.py: Un archivo requerido para que Python trate el directorio misitio como un paquete o como un grupo de módulos. Es un archivo vacio y generalmente no necesitaras agregarle nada.

   settings.py: Las opciones/configuraciones para nuestro proyecto Django.

   urls.py: Declaración de las URLs para este proyecto de Django.

   wsgi.py: El punto de entrada WSGI para el servidor Web, encargado de servir nuestro proyecto.

2. **python manage.py migrate** (Debes entrar primero al proyecto creado en el paso anterior)

   El comando **migrate** busca la variable INSTALLED_APPS y crea las tablas necesarias de cada una de las aplicaciones registradas en el archivo settings.py, que contiene todas las aplicaciones.

3. **python manage.py runserver**

   runserver inicia el servidor de desarrollo en el puerto 8000, escuchando sólo conexiones locales. Ahora que el servidor está corriendo, visita la dirección http://127.0.0.1:8000/ con tu navegador Web.

4. **python manage.py startapp biblioteca**
   Este comando no produce ninguna salida, pero crea un directorio llamado biblioteca dentro del directorio misitio y dentro de este crea otro directorio más, llamado migrations. Echemos un vistazo al contenido:

```
biblioteca/
    __init__.py
    admin.py
    models.py
    tests.py
    views.py
    migrations/
        __init__.py
```

   Estos archivos contendrán los modelos, las vistas, las pruebas, y las migraciones para esta aplicación.

Bibliografía

Garcia M., S. (2015). *La Guia definitiva de Django.* Celayita México: Django Software Corporation.