Bases de datos

Loreto Pelegrín Castillo

Base de datos SQLite3 y Flask

En esta parte de base de datos, se va a combinar SQLite y Flask, vamos a tener en cuenta diferentes elementos que nos proporciona flask para que funcione correctamente la integración:

- **Variable especial g**

g es una variable especial proporcionada por Flask. Es un objeto que se utiliza para almacenar datos a nivel de la solicitud (es decir, específicos para cada usuario que hace una solicitud al servidor).

```
from flask import g
```

Se va a utilizar a la hora de conectar con la base de datos.

```
def get_db():
    if 'db' not in g:
        g.db = sqlite3.connect('mi_base_de_datos.db')
    return g.db
```

Este condicional verifica si ya existe una conexión a la base de datos en el objeto g. Si 'db' no está presente en g, significa que aún no se ha establecido la conexión para esta solicitud.

- **Decorador** @app.before_request**:**

El decorador @app.before_request en Flask se utiliza para registrar una función que se ejecutará antes de procesar cada solicitud. Esto significa que, antes de que Flask invoque la función asociada a una ruta específica, las funciones registradas con @app.before_request se ejecutarán automáticamente.

- **Decorador @app.teardown_appcontext:**

En Flask, el decorador @app.teardown_appcontext se utiliza para registrar una función que se ejecutará automáticamente al final del ciclo de vida de una solicitud, cuando se está limpiando el contexto de la aplicación. Es particularmente útil para realizar tareas de limpieza como cerrar conexiones de base de datos, liberar recursos o realizar cualquier tarea que deba ejecutarse al finalizar la solicitud, sin importar si esta terminó con éxito o debido a un error.

```
# Función para cerrar la conexión a la base de datos
@app.teardown_appcontext
def close_db():
    if 'db' in g:
        g.db.close()
```

Ejemplo:

Como enviar un el contenido de una tabla de la base de datos a HTML.

```
@app.route('/')
def index():
    conexion = get_db()
    cursor = conexion.cursor()
    cursor.execute('SELECT * FROM T_Usuarios ')
    usuarios = cursor.fetchall()
    return render_template('index.html', usuarios=usuarios)
```

Crear un nuevo usuario.

```
# Ruta para crear un nuevo usuario
@app.route('/crear_usuario', methods=['POST'])
def crear_usuario():
    nombre = request.form['nombre']
    email = request.form['email']
    conexion = get_db()
    cursor = conexion.cursor()
    cursor.execute('INSERT INTO T_Usuarios (id, nombre, email) VALUES
(?,?, ?)', (None,nombre, email))
    conexion.commit()
    return redirect('/')
```

Borrar un usuario a partir de su id.

```
@app.route('/borrar_usuario/<int:id>', methods=['POST'])
def borrar_usuario(id):
    conexion = get_db()
    cursor = conexion.cursor()
    cursor.execute('DELETE FROM T_Usuarios WHERE id = ?', (id,))
    conexion.commit()
    return redirect('/')
```

Actualizar usuario.

```
@app.route('/actualizar_usuario/<int:id>', methods=['POST'])
def actualizar_usuario(id):
    nombre = request.form['nombre']
    email = request.form['email']
    conexion = get_db()
    cursor = conexion.cursor()
    cursor.execute('UPDATE usuarios SET nombre = ?, email = ? WHERE id =
?', (nombre, email, id))
    conexion.commit()
    return redirect('/')
```