Una Ambiciosa Introducción a Python
Parte II

Introducción a ORM con Peewee

ORM (Object-Relational Mapping) es una técnica que permite interactuar con bases de datos relacionales utilizando objetos en lugar de consultas SQL directas. En Python, hay varias librerías ORM, como SQLAlchemy y **Peewee**. Nos centraremos en Peewee, un ORM ligero y fácil de usar.

Diferencias entre manejo directo de SQLite y ORM
En los archivos address_book y address_book_orm, hay dos enfoques para manejar una base de datos SQLite:

1. **Uso directo de SQLite (sqlite3)**: Se ejecutan consultas SQL manualmente para crear tablas, insertar, eliminar y actualizar datos.
2. **Uso de ORM con Peewee**: Se define una clase que representa la tabla en la base de datos y se utilizan métodos del ORM para interactuar con los datos.

Vamos a analizar ambos enfoques comparando los archivos address_book.py y address_book_orm.py.

1. Enfoque Manual con sqlite3

En el archivo address_book.py, la interacción con SQLite se hace de la siguiente manera:

- Se establece la conexión con sqlite3.connect().
- Se ejecutan consultas SQL para crear la tabla (CREATE TABLE), insertar (INSERT INTO), actualizar (UPDATE), eliminar (DELETE) y recuperar (SELECT) contactos.
- Se utilizan consultas SQL directas almacenadas en queries.py.
- La manipulación de los datos requiere manejar cursores y ejecutar las sentencias SQL manualmente.

Consideraciones del enfoque manual

- Se deben escribir consultas SQL manualmente para cada operación.
- Se necesita manejar las conexiones y cursores manualmente.
- Mayor posibilidad de errores en consultas SQL.
- Código más verboso y menos reutilizable.

2. Enfoque con ORM (Peewee)

En el archivo address_book_orm.py, se utiliza Peewee para simplificar la interacción con la base de datos. La definición de la tabla y las operaciones se hacen de forma orientada a objetos.

Definición de Modelos en Peewee

En Peewee, cada tabla de la base de datos se representa con una clase que hereda de Model. Cada columna de la tabla es un atributo de la clase.

Definición de un Modelo

```
from peewee import SqliteDatabase, Model, IntegerField, CharField, DateField

db = SqliteDatabase('address_book.db')

class Contact(Model):
    id = IntegerField(primary_key=True)
    name = CharField(max_length=64, null=False)
    last_name = CharField(max_length=64, null=False)
```

```
    phone = CharField(max_length=16, null=True)
    email = CharField(max_length=64, null=True)
    birthday = DateField(null=True)

    class Meta:
        database = db  # Especifica la base de datos
        table_name = 'contact'  # Nombre de la tabla en la BD
```

Explicación de los Campos

- IntegerField(primary_key=True): Define una clave primaria.
- CharField(max_length=64, null=False): Campo de texto con límite de caracteres y que no permite valores nulos.
- DateField(null=True): Campo de tipo fecha que permite valores nulos.
- class Meta: Permite definir configuraciones adicionales como el nombre de la tabla y la base de datos asociada.

Creación de la Base de Datos y la Tabla

Para crear las tablas a partir de los modelos:

```
def create_address_book():
    db.create_tables([Contact])
```

Operaciones CRUD con Peewee

- **Insertar un contacto:**

```
def insert_contact(name, last_name, phone, email, birthday):
    Contact.create(name=name, last_name=last_name, phone=phone, email=email, birthday=birthday)
```

- **Buscar un contacto:**

```
def search_contact(contact_id):
    contact = Contact.get_or_none(Contact.id == contact_id)
    return contact
```

- **Actualizar un contacto:**

```
def update_contact(contact_id, name, last_name, phone, email, birthday):
    contact = Contact.get_or_none(Contact.id == contact_id)
    if contact:
        contact.name = name
        contact.last_name = last_name
        contact.phone = phone
        contact.email = email
        contact.birthday = birthday
        contact.save()
```

- **Eliminar un contacto:**

```
def delete_contact(contact_id):
    contact = Contact.get_or_none(Contact.id == contact_id)
    if contact:
        contact.delete_instance()
```

Ventajas del uso de Peewee

- No es necesario escribir SQL manualmente.
- Las operaciones con la base de datos se hacen de forma más intuitiva.
- Se reduce la posibilidad de errores tipográficos en consultas SQL.
- Código más claro y mantenible.
- Permite trabajar con diferentes bases de datos (SQLite, MySQL, PostgreSQL) con un mismo código.
- Soporte para relaciones entre tablas, validaciones y consultas avanzadas.

Una Ambiciosa Introducción a Python
Parte II

Conclusión

Peewee permite trabajar con bases de datos de manera más sencilla y organizada utilizando un enfoque orientado a objetos.
Mientras que el uso de sqlite3 requiere escribir consultas SQL manualmente, con Peewee podemos definir modelos y usar métodos que simplifican la manipulación de los datos.
Este enfoque facilita la escalabilidad y el mantenimiento del código en proyectos más grandes.
Aprender ORM con Peewee les permitira desarrollar aplicaciones con bases de datos de manera eficiente y estructurada.

Lic. Franco Herrera