TP4_Task_1

By Khertish Lobine

03/02/24

Université des Mascareignes

Faculty of Information and Communication Technology

Department of Applied Computer Science

Course : PMH

To Mr. Shiam Beehary

1. Adding the dependencies

The sqflite package provides classes and functions to interact with a SQLite database. The path package provides functions to define the location for storing the database on disk.

To add the packages as a dependency, run flutter pub add:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\Dev\db_test_app> flutter pub add sqflite path
Resolving dependencies...
  flutter_lints 2.0.3 (3.0.1 available)
  lints 2.1.1 (3.0.0 available)
  matcher 0.12.16 (0.12.16+1 available)
  material_color_utilities 0.5.0 (0.8.0 available)
  meta 1.10.0 (1.11.0 available)
  path 1.8.3 (from transitive dependency to direct dependency) (1.9.0 available)
+ sqflite 2.3.1
+ sqflite_common 2.5.0+2
+ synchronized 3.1.0+1
  test_api 0.6.1 (0.7.0 available)
```

2. Define the data dog model and opening the database

Define the path to the database file using **getDatabasesPath()** from the sqflite package, combined with the join function from the path package.Open the database with the **openDatabase()** function from sqflite.

```
lib > models.dart > Dog
1   class Dog {
2     final int id;
3     final String name;
4     final int age;
5
6     const Dog({
7       required this.id,
8       required this.name,
9       required this.age,
10    });
11  }
```

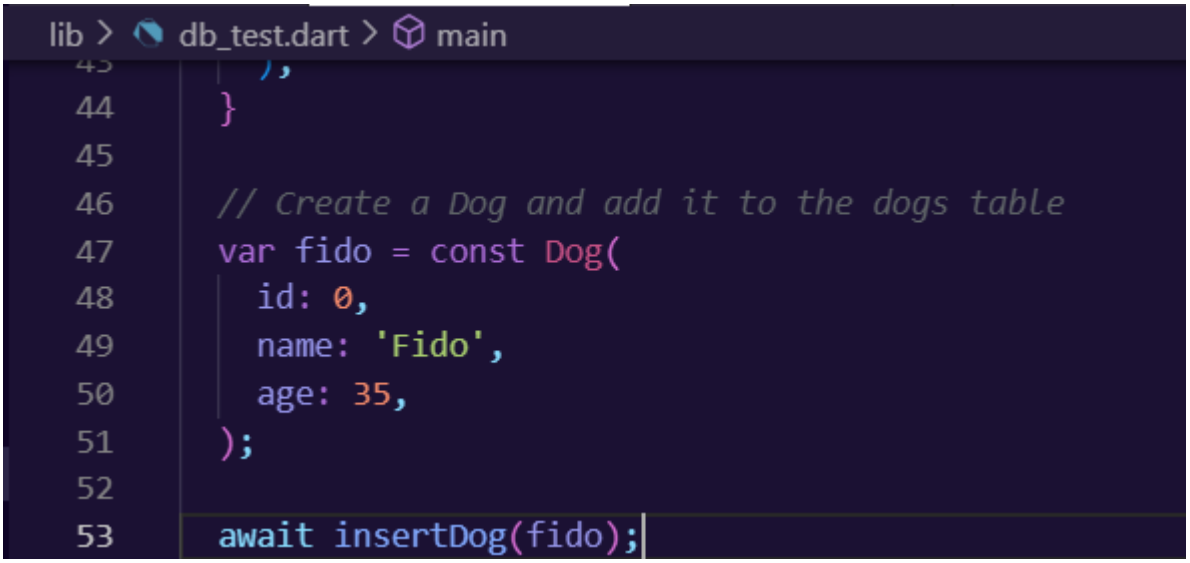
```dart
Run | Debug | Profile
void main() async {
  // Avoid errors caused by flutter upgrade.
  // Importing 'package:flutter/widgets.dart' is required.
  WidgetsFlutterBinding.ensureInitialized();
  // Open the database and store the reference.
  final database = openDatabase(
    // Set the path to the database. Note: Using the `join` function from the
    // `path` package is best practice to ensure the path is correctly
    // constructed for each platform.
    join(await getDatabasesPath(), 'doggie_database.db'),
  );
}
```

3. Creating the tables for dog

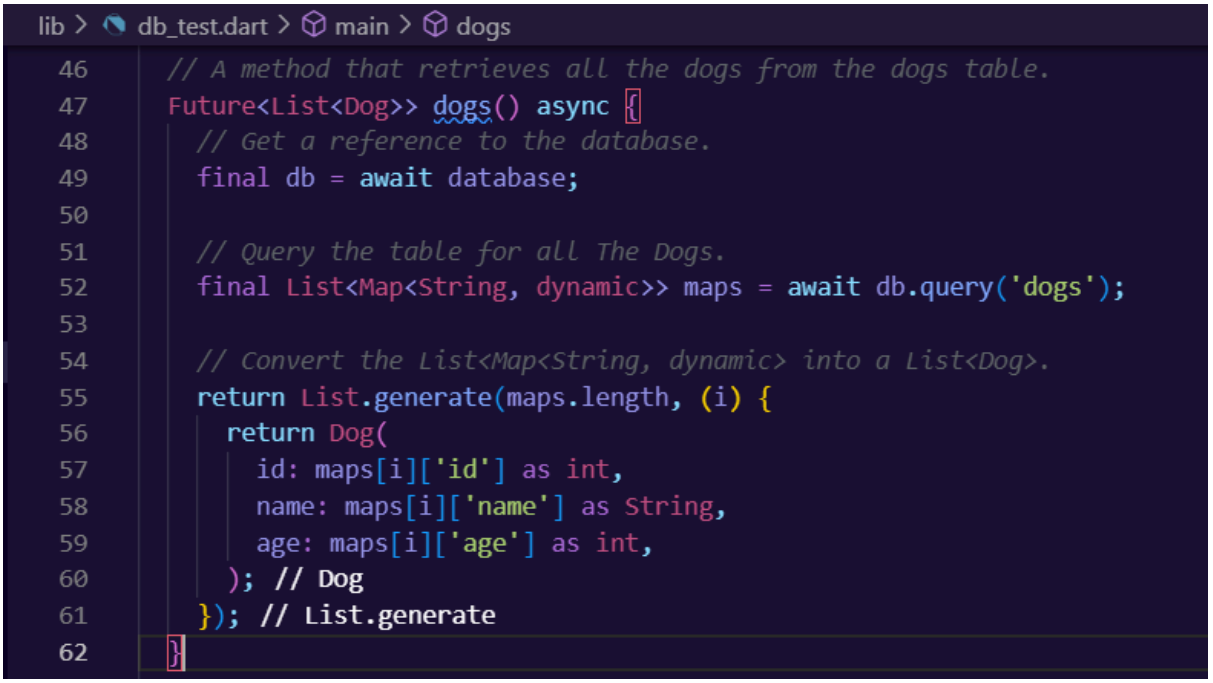
```dart
  // When the database is first created, create a table to store dogs.
  onCreate: (db, version) {
    // Run the CREATE TABLE statement on the database.
    return db.execute(
      'CREATE TABLE dogs(id INTEGER PRIMARY KEY, name TEXT, age INTEGER)',
    );
  },
  // Set the version. This executes the onCreate function and provides a
  // path to perform database upgrades and downgrades.
  version: 1,
);
```

4. Inserting dogs into the database by first converting dogs into a map and using the insert() to store the map

```dart
lib > models.dart > Dog > toString
  final int age;

  const Dog({
    required this.id,
    required this.name,
    required this.age,
  });

  // Convert a Dog into a Map. The keys must correspond to the names of the
  // columns in the database.
  Map<String, dynamic> toMap() {
    return {
      'id': id,
      'name': name,
      'age': age,
    };
  }

  // Implement toString to make it easier to see information about
  // each dog when using the print statement.
  @override
  String toString() {
    return 'Dog{id: $id, name: $name, age: $age}';
  }
}
```

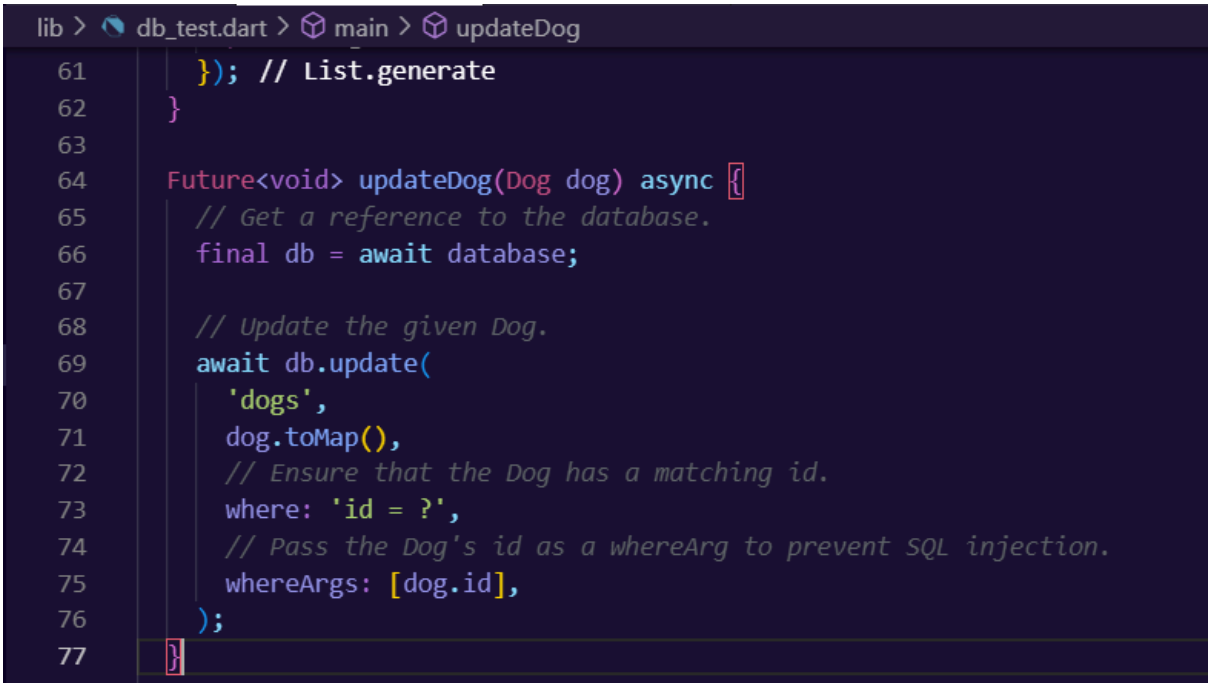


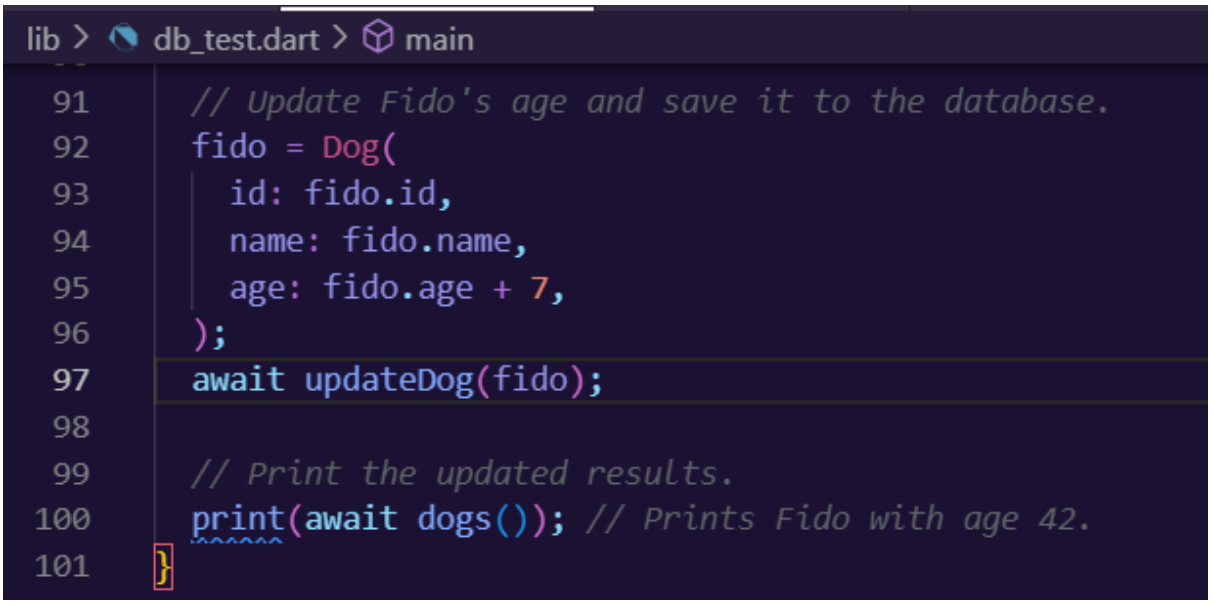
5. The list in the retrieved



7. A dog is then updated in the database

Using whereArgs to pass arguments to a where statement instead of using string interpolation, such as where: "**id = ${dog.id}**". This helps safeguard against SQL injection attacks.





8. A dog is then deleted from the database

```dart
lib > db_test.dart > main > deleteDog
  }

Future<void> deleteDog(int id) async {
  // Get a reference to the database.
  final db = await database;

  // Remove the Dog from the database.
  await db.delete(
    'dogs',
    // Use a `where` clause to delete a specific dog.
    where: 'id = ?',
    // Pass the Dog's id as a whereArg to prevent SQL injection.
    whereArgs: [id],
  );
}
```

```dart
lib > db_test.dart > main
  print(await dogs()); // Prints Fido

  // Delete Fido from the database.
  await deleteDog(fido.id);

  // Print the list of dogs (empty).
  print(await dogs());
}
```

9. Output of the code should contain

```
PS D:\Dev\db_test_app> flutter run lib/db_test.dart
Launching lib/db_test.dart on sdk gphone64 x86 64 in debug mode...
Running Gradle task 'assembleDebug'...                              5.4s
√  Built build\app\outputs\flutter-apk\app-debug.apk.
Installing build\app\outputs\flutter-apk\app-debug.apk...       1,785ms
I/flutter (12252): [Dog{id: 0, name: Fido, age: 35}]
I/flutter (12252): [Dog{id: 0, name: Fido, age: 42}]
I/flutter (12252): []
Syncing files to device sdk gphone64 x86 64...                     32ms

Flutter run key commands.
r Hot reload.
```