Flutter Sqlite Tutorial

Flutter SQLite Tutorial

In this tutorial, we shall learn basic SQLite operations with the help of a complete Flutter Application.

Use Case

We maintain a car database where each car has an id, name and number of miles driven. Our Application should be able to insert a row into the database, query rows, update a row or delete a row based on the required fields provided.

About UI

Following example application contains a TabBar with TabBarView for operations Insert, View, Query, Update and Delete operations that we shall perform on the car_table.

In Insert Screen, we shall take car name and number of miles it has gone through TextField widgets. Then when you click on the Insert button, we shall prepare the Car object through _insert()

method of main() and call insert() method of Database Helper.

In View Screen, there a Refresh button. When you click on it, it queries all the rows and displays them in a ListView.

In Query Screen, there is a TextField to get the name of the car from user. We have attached onChanged() method, so that, when user starts typing, it dynamically queries the table and shows the rows below the button in a ListView.

In Update Screen, we have three TextFields for reading id, name and miles from user. When user enters these fields and click update button, we shall update name and miles for the id provided. You can go the View tab, press on Refresh button to see if the update happened.

In Delete Screen, we have a TextField to read id. When user provides the id and presses Delete button, the row shall be deleted, based on id, if present in the database table.

Dependencies

Under dependencies section, below flutter property, add `sqlite` and `path` packages.

```
dependencies:
  flutter:
    sdk: flutter
  sqflite:
  path:
```

And click on **Packages get** button on the top right corner of editor.

No need to mention the version number for the packages sqflite and path.

Class File

Under lib folder, create a file named car.dart with class as shown below.

car.dart

```
import 'package:flutter_sqlite_tutorial/dbhe

class Car {
  int id;
  String name;
  int miles;

  Car(this.id, this.name, this.miles);

  Car.fromMap(Map<String, dynamic> map) {
    id = map['id'];
    name = map['name'];
    miles = map['miles'];
  }

  Map<String, dynamic> toMap() {
    return {
      DatabaseHelper.columnId: id,
      DatabaseHelper.columnName: name,
      DatabaseHelper.columnMiles: miles,
    };
  }
}
```

We shall use this class type to transfer data between UI (main.dart) and Database Helper Class.

Database Helper Class

Create a Database helper Class as shown below.

It contains methods to create a Database if not present, connect to the database, perform SQLite operations like query rows from table, update a row, delete a row, insert a row, etc.

dbhelper.dart

```
import 'package:flutter_sqlite_tutorial/car.
import 'package:path/path.dart';
import 'package:sqflite/sqflite.dart';

class DatabaseHelper {

  static final _databaseName = "cardb.db";
  static final _databaseVersion = 1;

  static final table = 'cars_table';

  static final columnId = 'id';
  static final columnName = 'name';
  static final columnMiles = 'miles';

  // make this a singleton class
  DatabaseHelper._privateConstructor();
  static final DatabaseHelper
instance = DatabaseHelper._privateConstructo

  // only have a single app-wide reference t
  static Database _database;
  Future<Database> get database async {
    if (_database != null) return _database;
    // lazily instantiate the db the first t
accessed
    _database = await _initDatabase();
    return _database;
  }

  // this opens the database (and creates it
exist)
  _initDatabase() async {
    String path = join(await
getDatabasesPath(), _databaseName);
    return await openDatabase(path,
        version: _databaseVersion,
        onCreate: _onCreate);
  }

  // SQL code to create the database table
```

```
  Future _onCreate(Database db, int version)
    await db.execute('''
          CREATE TABLE $table (
            $columnId INTEGER PRIMARY KEY AU
            $columnName TEXT NOT NULL,
            $columnMiles INTEGER NOT NULL
          )
          ''');
  }

  // Helper methods

  // Inserts a row in the database where eac
is a column name
  // and the value is the column value. The
the id of the
  // inserted row.
  Future<int> insert(Car car) async {
    Database db = await instance.database;
    return await
db.insert(table, {'name': car.name, 'miles':
  }

  // All of the rows are returned as a list
each map is
  // a key-value list of columns.
  Future<List<Map<String, dynamic>>> queryAl
    Database db = await instance.database;
    return await db.query(table);
  }

  // Queries rows based on the argument rece
  Future<List<Map<String, dynamic>>> queryRo
    Database db = await instance.database;
    return await db.query(table, where: "$co
'%$name%'");
  }

  // All of the methods (insert, query, upda
also be done using
  // raw SQL commands. This method uses a ra
the row count.
  Future<int> queryRowCount() async {
    Database db = await instance.database;
    return Sqflite.firstIntValue(await db.ra
COUNT(*) FROM $table'));
  }

  // We are assuming here that the id column
set. The other
  // column values will be used to update th
```

```
  Future<int> update(Car car) async {
    Database db = await instance.database;
    int id = car.toMap()['id'];
    return await
db.update(table, car.toMap(), where: '$colum
?', whereArgs: [id]);
  }

  // Deletes the row specified by the id. Th
affected rows is
  // returned. This should be 1 as long as t
  Future<int> delete(int id) async {
    Database db = await instance.database;
    return await db.delete(table, where: '$c
?', whereArgs: [id]);
  }
}
```

Observe that we have imported the sqflite and path at the start of our main.dart file.

Flutter UI

Following is the complete main.dart file.

main.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_sqlite_tutorial/car.
import 'package:flutter_sqlite_tutorial/dbhe

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'TutorialKart - Flutter',
      theme: ThemeData(
        primarySwatch: Colors.purple,
      ),
      home: MyHomePage(),
    );
  }
}
```

```dart
class MyHomePage extends StatefulWidget {
  @override
  _MyHomePageState createState() => _MyHomeP
}

class _MyHomePageState extends State<MyHomeP
  final dbHelper = DatabaseHelper.instance;

  List<Car> cars = [];
  List<Car> carsByName = [];

  //controllers used in insert operation UI
  TextEditingController nameController = Tex
  TextEditingController milesController = Te

  //controllers used in update operation UI
  TextEditingController idUpdateController =
  TextEditingController nameUpdateController
  TextEditingController milesUpdateControlle

  //controllers used in delete operation UI
  TextEditingController idDeleteController =

  //controllers used in query operation UI
  TextEditingController queryController = Te

  final GlobalKey<ScaffoldState> _scaffoldKe

  void _showMessageInScaffold(String message
    _scaffoldKey.currentState.showSnackBar(
        SnackBar(
          content: Text(message),
        )
    );
  }

  @override
  Widget build(BuildContext context) {
    return DefaultTabController(
      length: 5,
      child: Scaffold(
        key: _scaffoldKey,
        appBar: AppBar(
          bottom: TabBar(
            tabs: [
              Tab(
                text: "Insert",
              ),
              Tab(
                text: "View",
              ),
```

```
            Tab(
              text: "Query",
            ),
            Tab(
              text: "Update",
            ),
            Tab(
              text: "Delete",
            ),
          ],
        ),
        title: Text('TutorialKart - Flutte
      ),
      body: TabBarView(
        children: [
          Center(
            child: Column(
              children: <Widget>[
                Container(
                  padding: EdgeInsets.all(
                  child: TextField(
                    controller: nameContro
                    decoration: InputDecor
                      border: OutlineInput
                      labelText: 'Car Name
                    ),
                  ),
                ),
                Container(
                  padding: EdgeInsets.all(
                  child: TextField(
                    controller: milesContr
                    decoration: InputDecor
                      border: OutlineInput
                      labelText: 'Car Mile
                    ),
                  ),
                ),
                RaisedButton(
                  child: Text('Insert Car
                  onPressed: () {
                    String name = nameCont
                    int miles = int.parse(
                    _insert(name, miles);
                  },
                ),
              ],
            ),
          ),
          Container(
            child: ListView.builder(
```

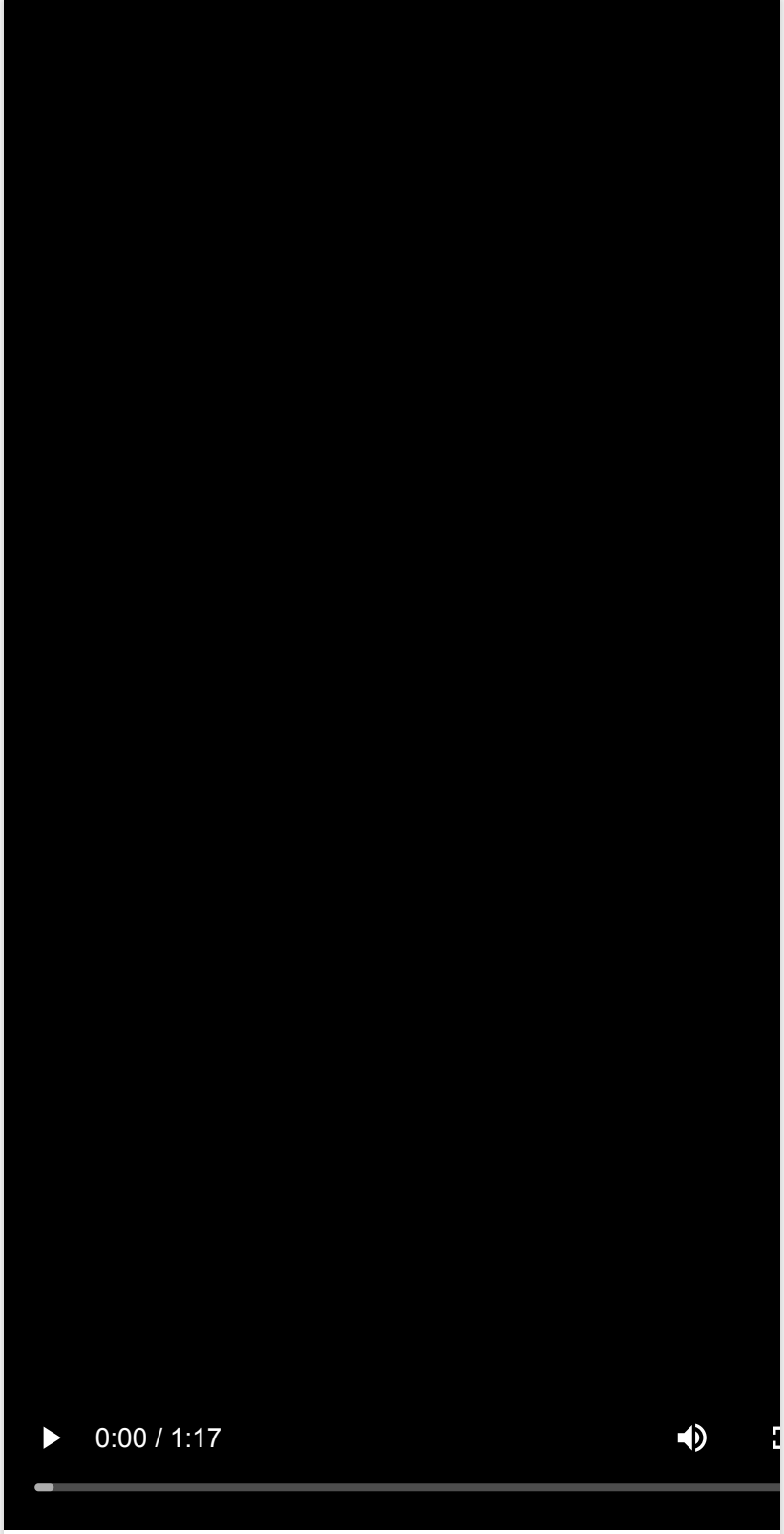
```
                padding: const EdgeInsets.al
                itemCount: cars.length + 1,
                itemBuilder: (BuildContext c
                  if (index == cars.length)
                    return RaisedButton(
                      child: Text('Refresh')
                      onPressed: () {
                        setState(() {
                          _queryAll();
                        });
                      },
                    );
                  }
                  return Container(
                    height: 40,
                    child: Center(
                      child: Text(
                        '[${cars[index].id}]
${cars[index].miles} miles',
                        style: TextStyle(fon
                      ),
                    ),
                  );
                },
              ),
            ),
            Center(
              child: Column(
                children: <Widget>[
                  Container(
                    padding: EdgeInsets.all(
                    child: TextField(
                      controller: queryContr
                      decoration: InputDecor
                        border: OutlineInput
                        labelText: 'Car Name
                      ),
                      onChanged: (text) {
                        if (text.length >= 2
                          setState(() {
                            _query(text);
                          });
                        } else {
                          setState(() {
                            carsByName.clear
                          });
                        }
                      },
                    ),
                    height: 100,
                  ),
```

```
                    Container(
                      height: 300,
                      child: ListView.builder(
                        padding: const EdgeIns
                        itemCount: carsByName.
                        itemBuilder: (BuildCon
                          return Container(
                            height: 50,
                            margin: EdgeInsets
                            child: Center(
                              child: Text(
                                '[${carsByName
${carsByName[index].name} - ${carsByName[ind
                                style: TextSty
                              ),
                            ),
                          );
                        },
                      ),
                    ),
                  ],
                ),
              ),
              Center(
                child: Column(
                  children: <Widget>[
                    Container(
                      padding: EdgeInsets.all(
                      child: TextField(
                        controller: idUpdateCo
                        decoration: InputDecor
                          border: OutlineInput
                          labelText: 'Car id',
                        ),
                      ),
                    ),
                    Container(
                      padding: EdgeInsets.all(
                      child: TextField(
                        controller: nameUpdate
                        decoration: InputDecor
                          border: OutlineInput
                          labelText: 'Car Name
                        ),
                      ),
                    ),
                    Container(
                      padding: EdgeInsets.all(
                      child: TextField(
                        controller: milesUpdat
                        decoration: InputDecor
```

```
                    border: OutlineInput
                    labelText: 'Car Mile
                  ),
                ),
              ),
              RaisedButton(
                child: Text('Update Car
                onPressed: () {
                  int id = int.parse(idU
                  String name = nameUpda
                  int miles = int.parse(
                  _update(id, name, mile
                },
              ),
            ],
          ),
        ),
        Center(
          child: Column(
            children: <Widget>[
              Container(
                padding: EdgeInsets.all(
                child: TextField(
                  controller: idDeleteCo
                  decoration: InputDecor
                    border: OutlineInput
                    labelText: 'Car id',
                  ),
                ),
              ),
              RaisedButton(
                child: Text('Delete'),
                onPressed: () {
                  int id = int.parse(idD
                  _delete(id);
                },
              ),
            ],
          ),
        ),
      ],
    ),
  ),
);
}

void _insert(name, miles) async {
  // row to insert
  Map<String, dynamic> row = {
    DatabaseHelper.columnName: name,
    DatabaseHelper.columnMiles: miles
```

```
      };
      Car car = Car.fromMap(row);
      final id = await dbHelper.insert(car);
      _showMessageInScaffold('inserted row id:
    }

    void _queryAll() async {
      final allRows = await dbHelper.queryAllR
      cars.clear();
      allRows.forEach((row) => cars.add(Car.fr
      _showMessageInScaffold('Query done.');
      setState(() {});
    }

    void _query(name) async {
      final allRows = await dbHelper.queryRows
      carsByName.clear();
      allRows.forEach((row) => carsByName.add(
    }

    void _update(id, name, miles) async {
      // row to update
      Car car = Car(id, name, miles);
      final rowsAffected = await dbHelper.upda
      _showMessageInScaffold('updated $rowsAff
    }

    void _delete(id) async {
      // Assuming that the number of rows is t
      final rowsDeleted = await dbHelper.delet
      _showMessageInScaffold('deleted $rowsDel
    }
  }
```

Output



Conclusion

In this Flutter Tutorial, we learned how to use SQLite in Flutter Application.

❮ Previous

Next ❯

Download Flutter Sqlite Tutorial in PDF

Popular Tutorials

- Salesforce Tutorial
- SAP Tutorials
- Kafka Tutorial
- Kotlin Tutorial

Interview Questions

- Salesforce Visualforce Interview Questions
- Salesforce Apex Interview Questions
- Kotlin Interview Questions

Tutorial Kart

- About Us
- Contact Us
- Careers - Write for us
- Privacy Policy
- Terms of Use

✎ Write Us Feedback

www.tutorialkart.com - ©Copyright-TutorialKart 2018