SQFlite เป็น package สำหรับการใช้งาน SQLITE database ใน flutter

CRUD คือพื้นฐานการทำงานกับฐานข้อมูล ประกอบด้วย Create, Read, Update, Delete

1. สร้าง Project
   1. สร้าง project ใหม่
   2. แก้ไข ไฟล์ pubspec.yaml เพิ่ม dependencies

sqflite: ^2.0.2

path: ^1.8.0

path\_provider: ^2.0.9

provider: ^6.0.2

**Text

Description automatically generated**

* 1. สร้างไฟล์ใหม่ ที่ โฟลเดอร์ lib ชื่อ user\_provider.dart เพื่อ กำหนด ข้อมูลที่ต้องการเก็บ

import 'package:flutter/foundation.dart';

class User {

  final String? id;

  final String? name;

  final int? age;

  final int? salary;

  final String? mobile;

  const User({this.id, this.name, this.age, this.salary, this.mobile});

}

class UserProvider with ChangeNotifier {

  List<User> \_nuser = [];

  //Map<String, dynamic> \_user = {};

  List<User> get users {

    return \_nuser;

  }

  // Map<String, dynamic> get users {

  //   return \_user;

  // }

  addNewUser(User data) {

    \_nuser.add(data);

    notifyListeners();

  }

}

* 1. สร้างไฟล์ใหม่ ที่ โฟลเดอร์ lib ชื่อ user.dart เพื่อ แสดงข้อมูล user และแก้ไขโฟลเดอร์ตามชื่อ project

import 'package:flutter/material.dart';

import 'package:lab10/user\_provider.dart';

import 'package:provider/provider.dart';

class BookPage extends StatelessWidget {

  const BookPage({Key? key}) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        appBar: AppBar(

          title: const Text("Users "),

          actions: [

            IconButton(

                icon: const Icon(Icons.add),

                onPressed: () {

                  Navigator.pushNamed(context, '/adduser');

                })

          ],

        ),

        body: Consumer<UserProvider>(

          builder:

              (BuildContext context, UserProvider provider, Widget? child) {

            return ListView.builder(

              itemCount: provider.users.length,

              itemBuilder: (BuildContext context, int index) {

                var useri = provider.users[index];

                var name = '${useri.age} ${useri.salary}';

                return ListTile(

                  title: Text('${useri.id!} ${useri.name!}'),

                  subtitle: Text(name),

                  trailing: Text(useri.mobile!),

                  // provider.users[index];

                );

              },

            );

          },

        ));

  }

}

* 1. สร้างไฟล์ใหม่ ที่ โฟลเดอร์ lib ชื่อ add\_user.dart เพื่อ แสดงข้อมูล user และแก้ไขโฟลเดอร์ตามชื่อ project

// ignore\_for\_file: avoid\_print, deprecated\_member\_use

import 'package:flutter/material.dart';

import 'package:lab10/user\_provider.dart';

import 'package:provider/provider.dart';

class AddBookPage extends StatefulWidget {

  const AddBookPage({Key? key}) : super(key: key);

  @override

  AddBookPageState createState() => AddBookPageState();

}

class AddBookPageState extends State<AddBookPage> {

  final \_form = GlobalKey<FormState>();

  final \_id = TextEditingController();

  final \_name = TextEditingController();

  final \_age = TextEditingController();

  final \_salary = TextEditingController();

  final \_mobile = TextEditingController();

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('Add User'),

      ),

      body: Form(

        key: \_form,

        child: ListView(

          children: <Widget>[

            buildIdField(),

            buildNameField(),

            buildAgeField(),

            buildSalaryField(),

            buildMobileField(),

            buildSaveButton()

          ],

        ),

      ),

    );

  }

  ElevatedButton buildSaveButton() {

    return ElevatedButton(

        child: const Text('Save'),

        onPressed: () async {

          if (\_form.currentState!.validate()) {

            print('save button press');

            User data = User(

                id: \_id.text,

                name: \_name.text,

                age: int.parse(\_age.text),

                mobile: \_mobile.text,

                salary: int.parse(\_salary.text));

            var nuser = Provider.of<UserProvider>(context, listen: false);

            nuser.addNewUser(data);

            Navigator.pop(context);

          } else {

            ScaffoldMessenger.of(context).showSnackBar(

              const SnackBar(

                content: Text('Please validate value'),

              ),

            );

          }

        });

  }

  TextFormField buildIdField() {

    return TextFormField(

      controller: \_id,

      decoration: const InputDecoration(

        labelText: 'id',

        icon: Icon(Icons.book),

      ),

      validator: (value) => value!.isEmpty ? 'Please fill in id' : null,

    );

  }

  TextFormField buildNameField() {

    return TextFormField(

      controller: \_name,

      decoration: const InputDecoration(

        labelText: 'name',

        icon: Icon(Icons.people),

      ),

      validator: (value) => value!.isEmpty ? 'Please fill in name' : null,

    );

  }

  TextFormField buildMobileField() {

    return TextFormField(

      controller: \_mobile,

      keyboardType: TextInputType.number,

      decoration: const InputDecoration(

        labelText: 'mobile',

        icon: Icon(Icons.list),

      ),

      validator: (value) =>

          value!.isEmpty ? 'Please fill in mobile number' : null,

    );

  }

  TextFormField buildAgeField() {

    return TextFormField(

        controller: \_age,

        keyboardType: TextInputType.number,

        decoration: const InputDecoration(

          labelText: 'age',

          icon: Icon(Icons.list),

        ),

        validator: (value) {

          if (value!.isEmpty) {

            return 'Please fill in age';

          } else {

            int age = int.parse(value);

            if (age < 0) {

              return 'Please fill in age';

            } else {

              return null;

            }

          }

        });

  }

  TextFormField buildSalaryField() {

    return TextFormField(

        controller: \_salary,

        keyboardType: TextInputType.number,

        decoration: const InputDecoration(

          labelText: 'salary',

          icon: Icon(Icons.list),

        ),

        validator: (value) {

          if (value!.isEmpty) {

            return 'Please fill in salary';

          } else {

            int salary = int.parse(value);

            if (salary < 0) {

              return 'Please fill in salary';

            } else {

              return null;

            }

          }

        });

  }

}

* 1. ลบข้อมูลในไฟล์ lib/main.dart แทนที่ด้วย

import 'package:flutter/material.dart';

import 'package:lab10/add\_user.dart';

import 'package:lab10/user.dart';

import 'package:lab10/user\_provider.dart';

import 'package:provider/provider.dart';

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

class MyApp extends StatelessWidget {

  const MyApp({Key? key}) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return MultiProvider(

      providers: [

        ChangeNotifierProvider(

          create: (BuildContext context) {

            return UserProvider();

          },

        )

      ],

      child: MaterialApp(

          title: 'SQFlite Demo',

          theme: ThemeData(

            primarySwatch: Colors.blue,

          ),

          initialRoute: '/',

          routes: {

      '/': (context) => const BookPage(),

            '/adduser': (context) => const AddBookPage(),

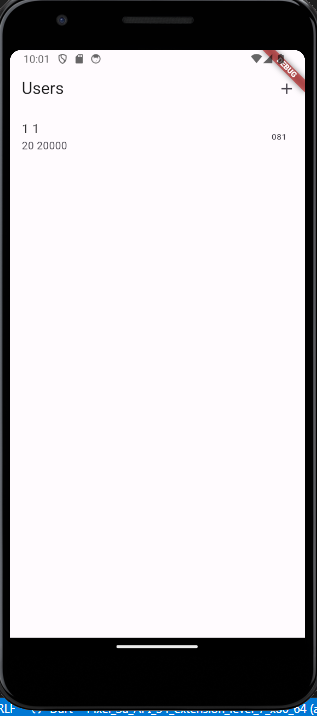
          }),

    );

  }

}

* 1. Run program และ แสดงผลลัพธ์



1. สร้าง Project
   1. แก้ไข ไฟล์ pubspec.yaml เพิ่ม dependencies

sqflite: ^2.0.2

path: ^1.8.0

path\_provider: ^2.0.9

provider: ^6.0.2

**Text

Description automatically generated**

1. สร้างไฟล์ใหม่ ที่ โฟลเดอร์ lib ชื่อ db\_helper.dart

import 'dart:io';

import 'package:path/path.dart';

import 'package:sqflite/sqflite.dart';

import 'package:path\_provider/path\_provider.dart';

class DatabaseHelper {

  static const \_databaseName = "MyDatabase.db";

  static const \_databaseVersion = 1;

  static const table = 'my\_table';

  static const columnId = '\_id';

  static const columnName = 'name';

  static const columnAge = 'age';

  // make this a singleton class

  DatabaseHelper.\_privateConstructor();

  static final DatabaseHelper instance = DatabaseHelper.\_privateConstructor();

  // only have a single app-wide reference to the database

  static Database? \_database;

  Future<Database> get database async {

    if (\_database != null) return \_database!;

    // lazily instantiate the db the first time it is accessed

    \_database = await \_initDatabase();

    return \_database!;

  }

  // this opens the database (and creates it if it doesn't exist)

  \_initDatabase() async {

    Directory documentsDirectory = await getApplicationDocumentsDirectory();

    String path = join(documentsDirectory.path, \_databaseName);

    return await openDatabase(path,

        version: \_databaseVersion, onCreate: \_onCreate, onUpgrade: \_onUpgrade);

  }

  // SQL code to create the database table

  Future \_onCreate(Database db, int version) async {

    await db.execute('''

          CREATE TABLE $table (

            $columnId INTEGER PRIMARY KEY,

            $columnName TEXT NOT NULL,

            $columnAge INTEGER NOT NULL

          )

          ''');

  }

  Future \_onUpgrade(Database db, int oldversion, int newversion) async {

    await db.execute('''

          DROP TABLE IF EXISTS $table

          ''');

    \_onCreate(db, newversion);

  }

  // Helper methods

  // Inserts a row in the database where each key in the Map is a column name

  // and the value is the column value. The return value is the id of the

  // inserted row.

  Future<int> insert(Map<String, dynamic> row) async {

    Database db = await instance.database;

    return await db.insert(table, row);

  }

  // All of the rows are returned as a list of maps, where each map is

  // a key-value list of columns.

  Future<List<Map<String, dynamic>>> queryAllRows() async {

    Database db = await instance.database;

    return await db.query(table);

  }

  // All othe methods (insert, query, update, delete) can also be done using

  // raw SQL commands. This method uses a raw query to give the row count.

  Future<int?> queryRowCount() async {

    Database db = await instance.database;

    return Sqflite.firstIntValue(

        await db.rawQuery('SELECT COUNT(\*) FROM $table'));

  }

  // We are assuming here that the id column in the map is set. The other

  // column values will be used to update the row.

  Future<int> update(Map<String, dynamic> row) async {

    Database db = await instance.database;

    int id = row[columnId];

    return await db.update(table, row, where: '$columnId = ?', whereArgs: [id]);

  }

  // Deletes the row specified by the id. The number of affected rows is

  // returned. This should be 1 as long as the row exists.

  Future<int> delete(int id) async {

    Database db = await instance.database;

    return await db.delete(table, where: '$columnId = ?', whereArgs: [id]);

  }

}

1. สร้างไฟล์ใหม่ ที่ โฟลเดอร์ lib ชื่อ update\_user.dart เพื่อใช้ในการแก้ไข ข้อมูลของ user

และแก้ไขชื่อ โฟลเดอร์ตรงไฮไลท์สีเหลือง เป็นชื่อของโปรเจค หาก เรียกใช้อัตโนมัติไม่ต้องแก้ไข

// ignore\_for\_file: avoid\_print, deprecated\_member\_use

import 'package:flutter/material.dart';

import 'package:lab65\_10\_2/db\_helper.dart';

class EditUserPage extends StatefulWidget {

  final Map<String, dynamic>? us;

  const EditUserPage({Key? key, @required this.us}) : super(key: key);

  @override

  State createState() => \_EditUserPageState();

  // receive data from the FirstScreen as a parameter

}

class \_EditUserPageState extends State<EditUserPage> {

  final \_form = GlobalKey<FormState>();

  @override

  Widget build(BuildContext context) {

    final id = TextEditingController(text: widget.us'\_id'].toString());

    final name = TextEditingController(text: widget.us!['name'].toString());

    final age = TextEditingController(text: widget.us!['age'].toString());

    return Scaffold(

      appBar: AppBar(

        title: const Text('Update User'),

      ),

      body: Form(

        key: \_form,

        child: ListView(

          children: <Widget>[

            TextFormField(

              controller: name,

              //   initialValue: widget.us['name'],

              decoration: const InputDecoration(

                labelText: 'name',

                icon: Icon(Icons.people),

              ),

              validator: (value) =>

                  value!.isEmpty ? 'Please fill in name' : null,

            ),

            TextFormField(

                controller: age,

                //  initialValue: widget.us['age'].toString(),

                keyboardType: TextInputType.number,

                decoration: const InputDecoration(

                  labelText: 'age',

                  icon: Icon(Icons.list),

                ),

                validator: (value) {

                  if (value!.isEmpty) {

                    return 'Please fill in age';

                  } else {

                    int age = int.parse(value);

                    if (age < 0) {

                      return 'Please fill in age';

                    } else {

                      return null;

                    }

                  }

                }),

            ElevatedButton(

                child: const Text('Save'),

                onPressed: () {

                  if (\_form.currentState!.validate()) {

                    print('save button press');

                    Map<String, dynamic> row = {

                      DatabaseHelper.columnId: int.parse(id.text),

                      DatabaseHelper.columnName: name.text,

                      DatabaseHelper.columnAge: int.parse(age.text),

                    };

                    final dbHelper = DatabaseHelper.instance;

                    final rowsAffected = dbHelper.update(row);

                    print('updated $rowsAffected row(s)');

                    Navigator.pop(context);

                  } else {

                    ScaffoldMessenger.of(context).showSnackBar(

                      const SnackBar(

                        content: Text('Please validate value'),

                      ),

                    );

                  }

                })

          ],

        ),

      ),

    );

  }

}

1. สร้างไฟล์ใหม่ ที่ โฟลเดอร์ lib ชื่อ add\_user.dart เพื่อใช้ในการแสดงรายการข้อมูลของ user

และแก้ไขชื่อ โฟลเดอร์ตรงไฮไลท์สีเหลือง เป็นชื่อของโปรเจค หาก เรียกใช้อัตโนมัติไม่ต้องแก้ไข

// ignore\_for\_file: deprecated\_member\_use, avoid\_print

import 'package:flutter/material.dart';

import 'package:lab65\_10\_2/db\_helper.dart';

class AddUserPage extends StatefulWidget {

  const AddUserPage({Key? key}) : super(key: key);

  @override

  AddUserPageState createState() => AddUserPageState();

}

class AddUserPageState extends State<AddUserPage> {

  final \_form = GlobalKey<FormState>();

  final \_id = TextEditingController();

  final \_name = TextEditingController();

  final \_age = TextEditingController();

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('Add User'),

      ),

      body: Form(

        key: \_form,

        child: ListView(

          children: <Widget>[

            //    buildIdField(),

            buildNameField(),

            buildAgeField(),

            buildSaveButton()

          ],

        ),

      ),

    );

  }

  ElevatedButton buildSaveButton() {

    return ElevatedButton(

        child: const Text('Save'),

        onPressed: () {

          if (\_form.currentState!.validate()) {

            print('save button press');

            Map<String, dynamic> row = {

              DatabaseHelper.columnName: \_name.text,

              DatabaseHelper.columnAge: int.parse(\_age.text),

            };

            final dbHelper = DatabaseHelper.instance;

            final id = dbHelper.insert(row);

            print('inserted row id: $id');

            Navigator.pop(context);

          } else {

            ScaffoldMessenger.of(context).showSnackBar(

              const SnackBar(

                content: Text('Please validate value'),

              ),

            );

          }

        });

  }

  TextFormField buildIdField() {

    return TextFormField(

      controller: \_id,

      decoration: const InputDecoration(

        labelText: 'id',

        icon: Icon(Icons.book),

      ),

      validator: (value) => value!.isEmpty ? 'Please fill in id' : null,

    );

  }

  TextFormField buildNameField() {

    return TextFormField(

      controller: \_name,

      decoration: const InputDecoration(

        labelText: 'name',

        icon: Icon(Icons.people),

      ),

      validator: (value) => value!.isEmpty ? 'Please fill in name' : null,

    );

  }

  TextFormField buildAgeField() {

    return TextFormField(

        controller: \_age,

        keyboardType: TextInputType.number,

        decoration: const InputDecoration(

          labelText: 'age',

          icon: Icon(Icons.list),

        ),

        validator: (value) {

          if (value!.isEmpty) {

            return 'Please fill in age';

          } else {

            int age = int.parse(value);

            if (age < 0) {

              return 'Please fill in age';

            } else {

              return null;

            }

          }

        });

  }

}

1. สร้างไฟล์ใหม่ ที่ โฟลเดอร์ lib ชื่อ user.dart เพื่อใช้ในการแสดงรายการข้อมูลของ user

และแก้ไขชื่อ โฟลเดอร์ตรงไฮไลท์สีเหลือง เป็นชื่อของโปรเจค หาก เรียกใช้อัตโนมัติไม่ต้องแก้ไข

// ignore\_for\_file: avoid\_print

import 'package:flutter/material.dart';

import 'package:lab65\_10\_2/db\_helper.dart';

import 'package:lab65\_10\_2/update\_user.dart';

class UserPage extends StatefulWidget {

  const UserPage({Key? key}) : super(key: key);

  @override

  UserPageState createState() => UserPageState();

}

class UserPageState extends State<UserPage> {

  List<Map<String, dynamic>> us = [];

  final dbHelper = DatabaseHelper.instance;

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text("Users "),

        actions: [

          IconButton(

              icon: const Icon(Icons.add),

              onPressed: () async {

                //            dbHelper.onDrop();

                await Navigator.pushNamed(context, '/adduser');

                \_query();

              })

        ],

      ),

      body: \_myListView(context),

    );

  }

  Future<void> \_query() async {

    final allRows = await dbHelper.queryAllRows();

    setState(() {

      us = allRows;

    });

  }

  void \_delete(Map<String, dynamic> us) async {

    var id = us['\_id'];

    // Assuming that the number of rows is the id for the last row.

    //final id = await dbHelper.queryRowCount();

    final rowsDeleted = await dbHelper.delete(id);

    print('deleted $rowsDeleted row(s): row $id');

    \_query();

  }

  Widget \_myListView(BuildContext context) {

    // \_query();

    return ListView.builder(

      itemCount: us.length,

      itemBuilder: (context, index) {

        return ListTile(

          title: Text(

              'No: ${us[index]['\_id']} Name: ${us[index]['name']} Age: ${us[index]['age']}'),

          trailing: IconButton(

              icon: const Icon(Icons.delete\_rounded),

              onPressed: () {

                \_delete(us[index]);

              }),

          onTap: () async {

            await Navigator.push(

                context,

                MaterialPageRoute(

                  builder: (context) => EditUserPage(

                    us: us[index],

                  ),

                ));

            \_query();

          },

        );

      },

    );

  }

}

1. ลบข้อมูลในไฟล์ lib/main.dart แทนที่ด้วย code ด้านล่างและ แก้ไขชื่อ โฟลเดอร์ตรงไฮไลท์สีเหลือง เป็นชื่อของโปรเจค หาก เรียกใช้อัตโนมัติไม่ต้องแก้ไข

import 'package:flutter/material.dart';

import 'package:lab65\_10\_2/add\_user.dart';

import 'package:lab65\_10\_2/user.dart';

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

class MyApp extends StatelessWidget {

  const MyApp({super.key});

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

        title: 'SQFlite Demo',

        theme: ThemeData(

          primarySwatch: Colors.blue,

        ),

        initialRoute: '/',

        routes: {

          '/': (context) => const UserPage(),

          '/adduser': (context) => const AddUserPage(),

        });

  }

}

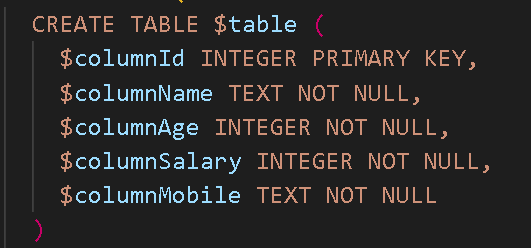
* 1. Run program และ แสดงผลลัพธ์

|  |
| --- |
|  |

1. แก้ไข projectดังนี้
   1. เปลี่ยนชื่อ table เป็น my\_table1 และ \_databaseVersion เป็น 2 ที่ไฟล์ db\_helper.dart
   2. แก้ไข table เพิ่ม ฟิลด์ salary และ mobile เพื่อเก็บเงินเดือน และเบอร์โทรศัพท์ ที่ db\_helper.dart ดังตัวอย่าง

![Text

Description automatically generated



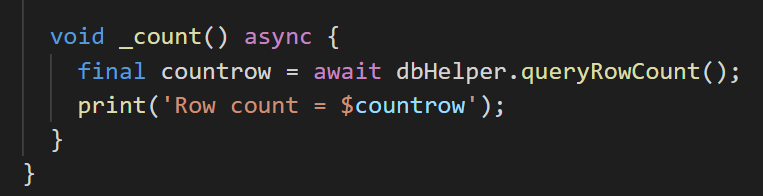
* 1. แก้ไขการแสดงผล โดยเพิ่มการแสดงข้อมูลเงินเดือนและเบอร์โทรศัพท์ที่ subtilte property ที่ ListTIle ใน user.dart
  2. เพิ่ม TextFormField ของ salary และ mobile ที่ add\_user.dart และ update\_user.dart
  3. เพิ่มตัวแปร ColumnSalary และ ColumnMobile เพื่อใช้ในการ insert ที่ปุ่ม save button ในไฟล์ add\_user.dart และ update ที่ update\_user.dart
  4. Run Program โดยการ Restart
  5. แสดง code และ ผลลัพธ์

|  |
| --- |
|  |

1. แก้ไข โปรแกรม method \_delete ที่ user.dart หากลบข้อมูล หมดแล้วให้แสดงคำว่า Empty table ที่ debug console
   1. Run program และแสดงผลลัพธ์

|  |
| --- |
|  |

1. เพิ่ม button ปุ่ม Count ที่ user.dart เพื่อเรียกใช้ method \_count()
   1. สร้าง method \_count ที่ user.dart เพื่อแสดงจำนวน row ที่ debug console



* 1. Run program และแสดงผลลัพธ์

|  |
| --- |
|  |