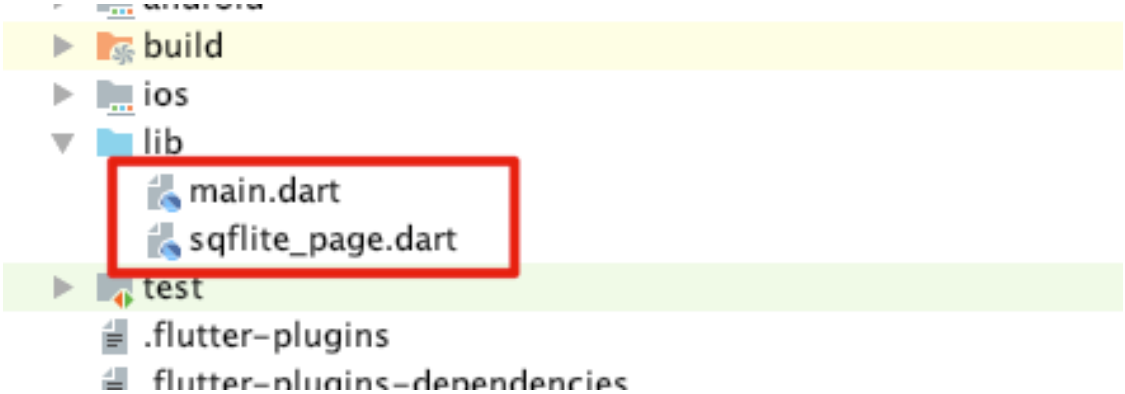
flutter 数据库sqflite

首先我要鄙视一下那些博客写一半的人（我叼他螺母），太坑了。

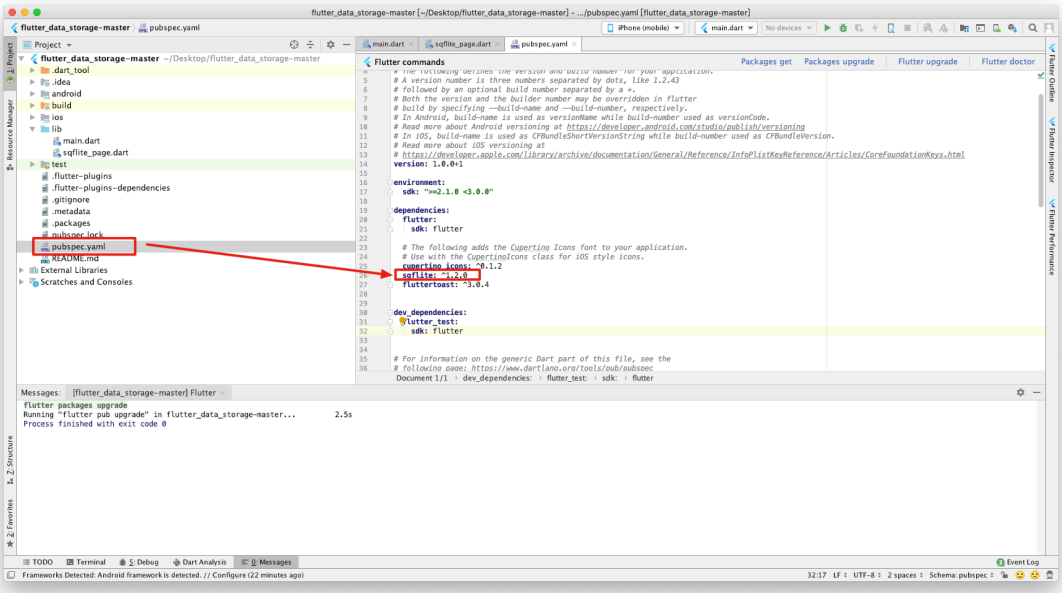
这边就两个文件main.dart和sqflite_page.dart



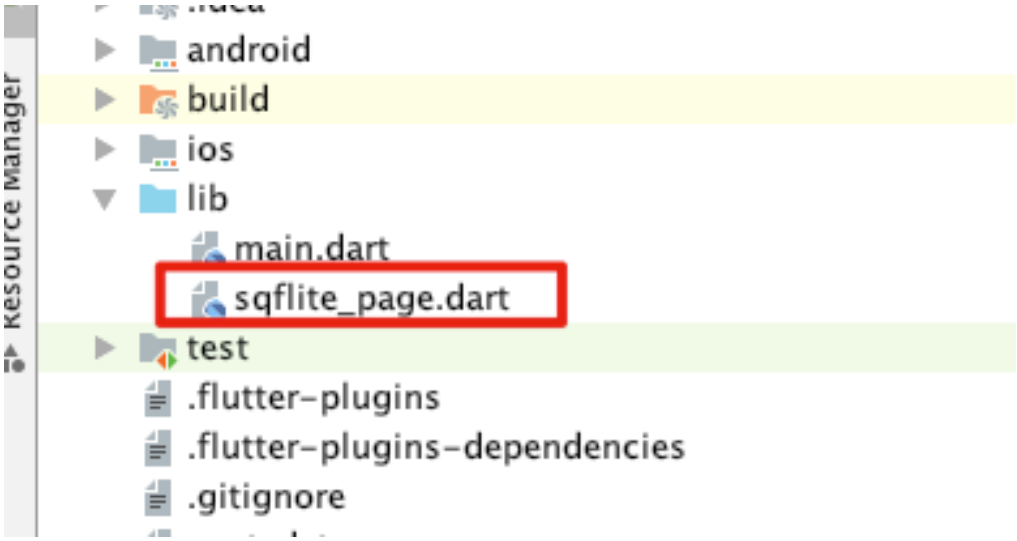
1.添加sqflite插件

在pubspec.yaml文件中添加sqflite，当前版本1.2.0：(别忘了点击右上角的“Packages get”或者“Packages upgrade”)

辅助插件：fluttertoast：^3.0.4



在lib下新建sqflite.dart文件：



则新建的sqflite.dart代码如下：

```
import 'package:flutter/material.dart';
import 'package:sqflite/sqflite.dart';
import 'package:path/path.dart';
import 'package:fluttertoast/fluttertoast.dart';
```

```
class SqflitePage extends StatefulWidget{
```

```
  @override
  State<StatefulWidget> createState() {
    // TODO: implement createState
    return _SqflitePageState();
  }
}
```

```
class _SqflitePageState extends State<SqflitePage> {
```

```
  TextEditingController _nameController=new
TextEditingController();
  TextEditingController _ageController=new
TextEditingController();
  String _data = "暂无数据";
  String _dbName = 'user.db';//数据库名称
```

```
  String _createTableSQL ='CREATE TABLE student_table (id
```

```
INTEGER PRIMARY KEY, name TEXT,age INTEGER)';//创建学生表;
  int _dbVersion = 1;//数据库版本
```

```
  @override
  void initState() {
    super.initState();
    //创建数据库、学生表
    _createDb(_dbName, _dbVersion, _createTableSQL);
  }

  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return Scaffold(
      appBar: AppBar(
        iconTheme: IconThemeData(color: Colors.white),
        title: Text("sqflite数据存储",style: TextStyle(color:
Colors.white),),
      ),
      body: Container(
        padding: EdgeInsets.all(20),
        child: Column(
          children: <Widget>[
            Center(
              child: Text("设置进入程序默认创建数据库和一张学生
表，如下可对表的姓名、年龄进行增删改查操作：",style:
TextStyle(fontSize: 14,color: Color(0xff666666)),),
            ),
            Padding(padding: EdgeInsets.only(top: 10)),
            _getNameInputView(),
            Padding(padding: EdgeInsets.only(top: 10)),
            _getAgeInputView(),
            Padding(padding: EdgeInsets.only(top: 30)),
            _getAddBtnView(),
            Padding(padding: EdgeInsets.only(top: 10)),
            _getdeleteBtnView(),
            Padding(padding: EdgeInsets.only(top: 10)),
            _getUpdateBtnView(),
            Padding(padding: EdgeInsets.only(top: 10)),
            _getQueryBtnView(),
            Padding(padding: EdgeInsets.only(top: 30)),
            Text(_data,style: TextStyle(color:
Colors.red,fontSize: 18),)
          ],
        ),
```

```
      ),
      resizeToAvoidBottomPadding: false,
    );
  }

  _getNameInputView() {
    return TextField(
      keyboardType: TextInputType.text,
      style: TextStyle(color: Color(0xFF888888)),
      controller: _nameController,
      decoration: InputDecoration(
        hintText: "姓名",
        hintStyle: TextStyle(color: Color(0xFF888888)),
        contentPadding: EdgeInsets.only(left: 10,right:
10,bottom: 10,top: 10),
        border: OutlineInputBorder(
          borderRadius: BorderRadius.circular(8.0),
        ),
      ),
    );
  }

  _getAgeInputView() {
    return TextField(
      keyboardType: TextInputType.text,
      style: TextStyle(color: Color(0xFF888888)),
      controller: _ageController,
      decoration: InputDecoration(
        hintText: "年龄",
        hintStyle: TextStyle(color: Color(0xFF888888)),
        contentPadding: EdgeInsets.only(left: 10,right:
10,bottom: 10,top: 10),
        border: OutlineInputBorder(
          borderRadius: BorderRadius.circular(8.0),
        ),
      ),
    );
  }

  _getAddBtnView(){
    return RaisedButton(
        onPressed: (){
          if(_nameController.text==null||
_nameController.text==""){
            Fluttertoast.showToast(msg: "插入数据不能为
空! ",backgroundColor: Colors.orange);
```

```
            return;
          }
          if(_ageController.text==null||
_ageController.text==""){
            Fluttertoast.showToast(msg: "插入数据不能为
空！",backgroundColor: Colors.orange);
            return;
          }
          String sql = "INSERT INTO student_table(name,age)
VALUES('${_nameController.text}','$
{_ageController.text}')";
          _add(_dbName, sql);

        },
        child: Text("插入数据",style: TextStyle(color:
Colors.white,fontSize: 18),),
        color: Colors.orange,
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(5),
        )
    );
  }

  _getdeleteBtnView(){
    return RaisedButton(
        onPressed: (){

          String sql = "DELETE FROM student_table";//无条件删
除学生表数据
          _delete(_dbName, sql);
        },
        child: Text("删除数据",style: TextStyle(color:
Colors.white,fontSize: 18),),
        color: Colors.orange,
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(5),
        )
    );
  }

  _getUpdateBtnView(){
    return RaisedButton(
        onPressed: (){
          if(_nameController.text==null||
_nameController.text==""){
```

```
            Fluttertoast.showToast(msg: "姓名不能为
空！",backgroundColor: Colors.orange);
            return;
          }

          String sql = "UPDATE student_table SET name =?
WHERE id = ?";
          _update(_dbName, sql,[_nameController.text,1]);
        },
        child: Text("修改姓名数据",style: TextStyle(color:
Colors.white,fontSize: 18),),
        color: Colors.orange,
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(5),
        )
    );
  }

  _getQueryBtnView(){
    return RaisedButton(
        onPressed: (){
          String sql = 'SELECT * FROM student_table';
          _query(_dbName, sql);
        },
        child: Text("查询数据",style: TextStyle(color:
Colors.white,fontSize: 18),),
        color: Colors.orange,
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(5),
        )
    );
  }

  ///创建数据库db
  _createDb(String dbName,int vers,String dbTables) async {
    //获取数据库路径
    var databasesPath = await getDatabasesPath();
    String path = join(databasesPath, dbName);
    print("数据库路径：$path数据库版本$vers");
    //打开数据库
    await openDatabase(
        path,
        version:vers,
        onUpgrade: (Database db, int oldVersion, int
newVersion) async{
```

```
          //数据库升级,只回调一次
          print("数据库需要升级！旧版：$oldVersion,新版：
$newVersion");
        },
        onCreate: (Database db, int vers) async{
          //创建表，只回调一次
          await db.execute(dbTables);
          await db.close();

        }
    );

    setState(() {
      _data = "成功创建数据库db！\n数据库路径：$path \n数据库版本
$vers";
    });

  }

  ///增
  _add(String dbName,String sql) async {
    //获取数据库路径
    var databasesPath = await getDatabasesPath();
    String path = join(databasesPath, dbName);
    print("数据库路径：$path");

    Database db = await openDatabase(path);
    await db.transaction((txn) async {
      int count = await txn.rawInsert(sql);
    });
    await db.close();

    setState(() {
      _data = "插入数据成功！";
    });
  }
  ///删
  _delete(String dbName,String sql) async {
    var databasesPath = await getDatabasesPath();
    String path = join(databasesPath, dbName);

    Database db = await openDatabase(path);
    int count = await db.rawDelete(sql);
    await db.close();
```

```
    if (count > 0) {
      setState(() {
        _data = "执行删除操作完成，该sql删除条件下的数目为:
$count";
      });
    } else {
      setState(() {
        _data = "无法执行删除操作，该sql删除条件下的数目为:
$count";
      });
    }
  }

  ///改
  _update(String dbName,String sql,List arg) async {
    var databasesPath = await getDatabasesPath();
    String path = join(databasesPath, dbName);

    Database db = await openDatabase(path);
    int count = await db.rawUpdate(sql,arg);//修改条件，对应参
数值
    await db.close();
    if (count > 0) {
      setState(() {
        _data = "更新数据库操作完成，该sql删除条件下的数目为:
$count";
      });
    } else {
      setState(() {
        _data = "无法更新数据库，该sql删除条件下的数目为: $count";
      });
    }
  }

  ///查条数
  _getQueryNum(String dbName,String sql) async {
    var databasesPath = await getDatabasesPath();
    String path = join(databasesPath, dbName);

    Database db = await openDatabase(path);
    int count = Sqflite.firstIntValue(await
db.rawQuery(sql));
    await db.close();
    return count;
  }
```

```
  ///查全部
  _query(String dbName,String sql) async {
    var databasesPath = await getDatabasesPath();
    String path = join(databasesPath, dbName);

    Database db = await openDatabase(path);
    List<Map> list = await db.rawQuery(sql);
    await db.close();
    setState(() {
      _data = "数据详情: $list";
    });
  }
}
```

main.dart中调用代码如下：

```
import 'package:flutter/material.dart';
import 'package:flutter_data_storage/sqflite_page.dart';

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

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.orange,
      ),
      home: MyHomePage(title: 'Flutter学习'),
      routes: <String,WidgetBuilder>{
        sqfliteRoute:(BuildContext context) =>
SqflitePage(),
      },
    );
  }
}

const String sqfliteRoute = "SqfliteRoute";

class MyHomePage extends StatefulWidget {
  MyHomePage({Key key, this.title}) : super(key: key);
  final String title;
```

```
  @override
  _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {

  @override
  Widget build(BuildContext context) {

    return Scaffold(
      appBar: AppBar(
        // Here we take the value from the MyHomePage
object that was created by
        // the App.build method, and use it to set our
appbar title.
        title: Text(widget.title),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            RaisedButton(
              onPressed: (){

Navigator.of(context).pushNamed(sqfliteRoute);
              },
              child: Text("sqflite",style:
TextStyle(fontSize: 18),),

            )
          ],
        ),
      )// This trailing comma makes auto-formatting nicer
for build methods.
    );
  }
}
```