一统天下 flutter – dart: 多线程异步编程（isolate/compute）

示例如下:

lib\dart\isolate.dart

```
/*
 * dart 多线程异步编程（isolate/compute）
 *
 * dart 的多线程编程是通过 isolate 实现的，忘记传统的多线程编程吧（当然 isolate 的底层技术还是通过系统的线程和进程实现的）
 * dart 有一个主 isolate，你可以新开 isolate 实现并发编程
 * dart 的 isolate 与传统线程的主要区别就是，每个 isolate 都只维护自己的内存堆且不会与其它 isolate 共享，也就是说不需要锁这种东西了
 * dart 的不同 isolate 之间通过消息传递来实现通信
 */
import 'dart:async';
import 'dart:isolate';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:flutter_demo/helper.dart';
```

```dart
class DartIsolate extends StatelessWidget {
  const DartIsolate({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    // 演示 isolate 的用法
    sample1();
    // 演示 compute 的用法
    sample2();
    return const MyWidget(text: "dart_isolate");
  }

  void sample1() async {
    // isolate2
    Future<void> isolate2(SendPort sp) async {
      for (var i = 0; i < 5; i++) {
        await Future.delayed(const
Duration(seconds: 1));
        log("sample1: send $i");
        // 向消息通道发送消息（因为 isolate 之间不会共享
内存，所以这里的消息会先复制然后再发送副本，所以可能会比较
慢）
        sp.send(i);
      }
      log("sample1: send 999 and exit");
      // 终止当前 isolate 并关闭指定的消息通道，并且在关闭
```

```
消息通道之前发送指定的消息（这个最后的消息不会复制，而是直接发送，所以会很快）
      Isolate.exit(sp, 999);
    }
    // isolate1
    Future<int> isolate1() async {
      // 创建一个用于在不同 isolate 之间传递消息的通道
      // ReceivePort 继承自 Stream<dynamic> 你可以通过它获取其他 isolate 发来的消息
      var rp = ReceivePort();
      // 通过 spawn() 启动指定的 isolate 并传递指定消息通道的 SendPort 对象，以便新启动的 isolate 可以发送消息
      await Isolate.spawn(isolate2, rp.sendPort);
      var sum = 0;
      // 接收消息通道中发来的消息
      await for (final value in rp) {
        var v = value as int;
        log("sample1: recv $v");
        // 自定义约定，收到 999 就退出
        if (v == 999) {
          break;
        }
        sum += v;
      }
      return sum;
```

```
    }
    var a = await isolate1();
    log("sample1: $a"); // sample1: 10
  }
  // compute() 用于非常方便地将你自己的逻辑交给其他线程执行，其并不在 dart 库中，而是 flutter 提供的，需要 import 'package:flutter/foundation.dart';
  void sample2() async {
    // 第 1 个参数：需要在其他线程执行的方法的名称
    // 第 2 个参数：需要传递的数据
    var a = await compute(myCompute, "webabcd");
    log("sample2: $a"); // sample2: compute result: webabcd
  }
  String myCompute(String value) {
    return "compute result: $value";
  }
}
```

源码 https://github.com/webabcd/flutter_demo