# 11.1 文件操作

Dart的IO库包含了文件读写的相关类，它属于Dart语法标准的一部分，所以通过Dart IO库，无论是Dart VM下的脚本还是Flutter，都是通过Dart IO库来操作文件的，不过和Dart VM相比，Flutter有一个重要差异是文件系统路径不同，这是因为Dart VM是运行在PC或服务器操作系统下，而Flutter是运行在移动操作系统中，他们的文件系统会有一些差异。

#### APP目录

Android和iOS的应用存储目录不同，[PathProvider](https://pub.dartlang.org/packages/path_provider) 插件提供了一种平台透明的方式来访问设备文件系统上的常用位置。该类当前支持访问两个文件系统位置：

* **临时目录:** 可以使用 getTemporaryDirectory() 来获取临时目录； 系统可随时清除的临时目录（缓存）。在iOS上，这对应于[NSTemporaryDirectory()](https://developer.apple.com/reference/foundation/1409211-nstemporarydirectory) 返回的值。在Android上，这是[getCacheDir()](https://developer.android.com/reference/android/content/Context.html#getCacheDir())返回的值。
* **文档目录:** 可以使用getApplicationDocumentsDirectory()来获取应用程序的文档目录，该目录用于存储只有自己可以访问的文件。只有当应用程序被卸载时，系统才会清除该目录。在iOS上，这对应于NSDocumentDirectory。在Android上，这是AppData目录。
* **外部存储目录**：可以使用getExternalStorageDirectory()来获取外部存储目录，如SD卡；由于iOS不支持外部目录，所以在iOS下调用该方法会抛出UnsupportedError异常，而在Android下结果是android SDK中getExternalStorageDirectory的返回值。

一旦你的Flutter应用程序有一个文件位置的引用，你可以使用[dart:io](https://api.dartlang.org/stable/dart-io/dart-io-library.html)API来执行对文件系统的读/写操作。有关使用Dart处理文件和目录的详细内容可以参考Dart语言文档，下面我们看一个简单的例子。

#### 示例

我们还是以计数器为例，实现在应用退出重启后可以恢复点击次数。 这里，我们使用文件来保存数据：

1. 引入PathProvider插件；在pubspec.yaml文件中添加如下声明：

* path\_provider: ^0.4.1
* 添加后，执行flutter packages get 获取一下, 版本号可能随着时间推移会发生变化，读者可以使用最新版。

1. 实现：

* import 'dart:io';  
  import 'dart:async';  
  import 'package:flutter/material.dart';  
  import 'package:path\_provider/path\_provider.dart';  
    
  class FileOperationRoute extends StatefulWidget {  
   FileOperationRoute({Key key}) : super(key: key);  
    
   @override  
   \_FileOperationRouteState createState() => new \_FileOperationRouteState();  
  }  
    
  class \_FileOperationRouteState extends State<FileOperationRoute> {  
   int \_counter;  
    
   @override  
   void initState() {  
   super.initState();  
   //从文件读取点击次数  
   \_readCounter().then((int value) {  
   setState(() {  
   \_counter = value;  
   });  
   });  
   }  
    
   Future<File> \_getLocalFile() async {  
   // 获取应用目录  
   String dir = (await getApplicationDocumentsDirectory()).path;  
   return new File('$dir/counter.txt');  
   }  
    
   Future<int> \_readCounter() async {  
   try {  
   File file = await \_getLocalFile();  
   // 读取点击次数（以字符串）  
   String contents = await file.readAsString();  
   return int.parse(contents);  
   } on FileSystemException {  
   return 0;  
   }  
   }  
    
   Future<Null> \_incrementCounter() async {  
   setState(() {  
   \_counter++;  
   });  
   // 将点击次数以字符串类型写到文件中  
   await (await \_getLocalFile()).writeAsString('$\_counter');  
   }  
    
   @override  
   Widget build(BuildContext context) {  
   return new Scaffold(  
   appBar: new AppBar(title: new Text('文件操作')),  
   body: new Center(  
   child: new Text('点击了 $\_counter 次'),  
   ),  
   floatingActionButton: new FloatingActionButton(  
   onPressed: \_incrementCounter,  
   tooltip: 'Increment',  
   child: new Icon(Icons.add),  
   ),  
   );  
   }  
  }
* 上面代码比较简单，不再赘述，需要说明的是，本示例只是为了演示文件读写，而在实际开发中，如果要存储一些简单的数据，使用shared\_preferences插件会比较简单。
* 注意，Dart IO库操作文件的API非常丰富，但本书不是介绍Dart语言的，故不详细说明，读者需要的话可以自行学习。