# 12.3 开发Flutter插件

下面我们通过一个获取电池电量的插件来介绍一下Flutter插件的开发流程。该插件中我们在Dart中通过getBatteryLevel 调用Android BatteryManager API和iOS device.batteryLevel API。

### 创建一个新的应用程序项目

首先创建一个新的应用程序:

* 在终端中运行：flutter create batterylevel

默认情况下，模板支持使用Java编写Android代码，或使用Objective-C编写iOS代码。要使用Kotlin或Swift，请使用-i和/或-a标志:

* 在终端中运行: flutter create -i swift -a kotlin batterylevel

### 创建Flutter平台客户端

该应用的State类拥有当前的应用状态。我们需要延长这一点以保持当前的电量

首先，我们构建通道。我们使用MethodChannel调用一个方法来返回电池电量。

通道的客户端和宿主通过通道构造函数中传递的通道名称进行连接。单个应用中使用的所有通道名称必须是唯一的; 我们建议在通道名称前加一个唯一的“域名前缀”，例如samples.flutter.io/battery。

import 'dart:async';  
  
import 'package:flutter/material.dart';  
import 'package:flutter/services.dart';  
...  
class \_MyHomePageState extends State<MyHomePage> {  
 static const platform = const MethodChannel('samples.flutter.io/battery');  
  
 // Get battery level.  
}

接下来，我们调用通道上的方法，指定通过字符串标识符调用方法getBatteryLevel。 该调用可能失败(平台不支持平台API，例如在模拟器中运行时)，所以我们将invokeMethod调用包装在try-catch语句中。

我们使用返回的结果，在setState中来更新用户界面状态batteryLevel。

// Get battery level.  
 String \_batteryLevel = 'Unknown battery level.';  
  
 Future<Null> \_getBatteryLevel() async {  
 String batteryLevel;  
 try {  
 final int result = await platform.invokeMethod('getBatteryLevel');  
 batteryLevel = 'Battery level at $result % .';  
 } on PlatformException catch (e) {  
 batteryLevel = "Failed to get battery level: '${e.message}'.";  
 }  
  
 setState(() {  
 \_batteryLevel = batteryLevel;  
 });  
 }

最后，我们在build创建包含一个小字体显示电池状态和一个用于刷新值的按钮的用户界面。

@override  
Widget build(BuildContext context) {  
 return new Material(  
 child: new Center(  
 child: new Column(  
 mainAxisAlignment: MainAxisAlignment.spaceEvenly,  
 children: [  
 new RaisedButton(  
 child: new Text('Get Battery Level'),  
 onPressed: \_getBatteryLevel,  
 ),  
 new Text(\_batteryLevel),  
 ],  
 ),  
 ),  
 );  
}

至此Flutter部分的测试代码写好了，接下来我们需要实现Android和iOS平台下的API，由于平台API实现部分篇幅较大，我们将在接下来的两节中，分别介绍Android和iOS端API的实现。