# 12.4 插件开发：Android端API实现

本节我们接着上一节“获取电池电量”插件的示例，来完成Android端API的实现。以下步骤是使用Java的示例，如果您更喜欢Kotlin，可以直接跳到后面Kotlin部分。

首先在Android Studio中打开您的Flutter应用的Android部分：

1. 启动 Android Studio
2. 选择 File > Open…
3. 定位到您 Flutter app目录, 然后选择里面的 android文件夹，点击 OK
4. 在java目录下打开 MainActivity.java

接下来，在onCreate里创建MethodChannel并设置一个MethodCallHandler。确保使用和Flutter客户端中使用的通道名称相同的名称。

import io.flutter.app.FlutterActivity;  
import io.flutter.plugin.common.MethodCall;  
import io.flutter.plugin.common.MethodChannel;  
import io.flutter.plugin.common.MethodChannel.MethodCallHandler;  
import io.flutter.plugin.common.MethodChannel.Result;  
  
public class MainActivity extends FlutterActivity {  
 private static final String CHANNEL = "samples.flutter.io/battery";  
  
 @Override  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 new MethodChannel(getFlutterView(), CHANNEL).setMethodCallHandler(  
 new MethodCallHandler() {  
 @Override  
 public void onMethodCall(MethodCall call, Result result) {  
 // TODO  
 }  
 });  
 }  
}

接下来，我们添加Java代码，使用Android电池API来获取电池电量。此代码和在原生Android应用中编写的代码完全相同。

首先，添加需要导入的依赖。

import android.content.ContextWrapper;  
import android.content.Intent;  
import android.content.IntentFilter;  
import android.os.BatteryManager;  
import android.os.Build.VERSION;  
import android.os.Build.VERSION\_CODES;  
import android.os.Bundle;

然后，将下面的新方法添加到activity类中的，位于onCreate 方法下方：

private int getBatteryLevel() {  
 int batteryLevel = -1;  
 if (VERSION.SDK\_INT >= VERSION\_CODES.LOLLIPOP) {  
 BatteryManager batteryManager = (BatteryManager) getSystemService(BATTERY\_SERVICE);  
 batteryLevel = batteryManager.getIntProperty(BatteryManager.BATTERY\_PROPERTY\_CAPACITY);  
 } else {  
 Intent intent = new ContextWrapper(getApplicationContext()).  
 registerReceiver(null, new IntentFilter(Intent.ACTION\_BATTERY\_CHANGED));  
 batteryLevel = (intent.getIntExtra(BatteryManager.EXTRA\_LEVEL, -1) \* 100) /  
 intent.getIntExtra(BatteryManager.EXTRA\_SCALE, -1);  
 }  
  
 return batteryLevel;  
}

最后，我们完成之前添加的onMethodCall方法。我们需要处理平台方法名为getBatteryLevel的调用消息，所以我们需要先在call参数判断调用的方法是否为getBatteryLevel。 这个平台方法的实现只需调用我们在前一步中编写的Android代码，并通过result参数返回成功或错误情况的响应信息。如果调用了未定义的API，我们也会通知返回：

@Override  
public void onMethodCall(MethodCall call, Result result) {  
 if (call.method.equals("getBatteryLevel")) {  
 int batteryLevel = getBatteryLevel();  
  
 if (batteryLevel != -1) {  
 result.success(batteryLevel);  
 } else {  
 result.error("UNAVAILABLE", "Battery level not available.", null);  
 }  
 } else {  
 result.notImplemented();  
 }  
}

现在就可以在Android上运行该应用程序了，如果使用的是Android模拟器，则可以通过工具栏中的“…”按钮访问Extended Controls面板中的电池电量。

### 使用Kotlin添加Android平台特定的实现

使用Kotlin和使用Java的步骤类似，首先在Android Studio中打开您的Flutter应用的Android部分：

1. 启动 Android Studio。
2. 选择 the menu item “File > Open…”。
3. 定位到 Flutter app目录, 然后选择里面的 android文件夹，点击 OK。
4. 在kotlin目录中打开MainActivity.kt。

接下来，在onCreate里创建MethodChannel并设置一个MethodCallHandler。确保使用与在Flutter客户端使用的通道名称相同。

import android.os.Bundle  
import io.flutter.app.FlutterActivity  
import io.flutter.plugin.common.MethodChannel  
import io.flutter.plugins.GeneratedPluginRegistrant  
  
class MainActivity() : FlutterActivity() {  
 private val CHANNEL = "samples.flutter.io/battery"  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 GeneratedPluginRegistrant.registerWith(this)  
  
 MethodChannel(flutterView, CHANNEL).setMethodCallHandler { call, result ->  
 // TODO  
 }  
 }  
}

接下来，我们添加Kotlin代码，使用Android电池API来获取电池电量，这和原生开发是一样的。

首先，添加需要导入的依赖。

import android.content.Context  
import android.content.ContextWrapper  
import android.content.Intent  
import android.content.IntentFilter  
import android.os.BatteryManager  
import android.os.Build.VERSION  
import android.os.Build.VERSION\_CODES

然后，将下面的新方法添加到activity类中的，位于onCreate 方法下方：

private fun getBatteryLevel(): Int {  
 val batteryLevel: Int  
 if (VERSION.SDK\_INT >= VERSION\_CODES.LOLLIPOP) {  
 val batteryManager = getSystemService(Context.BATTERY\_SERVICE) as BatteryManager  
 batteryLevel = batteryManager.getIntProperty(BatteryManager.BATTERY\_PROPERTY\_CAPACITY)  
 } else {  
 val intent = ContextWrapper(applicationContext).registerReceiver(null, IntentFilter(Intent.ACTION\_BATTERY\_CHANGED))  
 batteryLevel = intent!!.getIntExtra(BatteryManager.EXTRA\_LEVEL, -1) \* 100 / intent.getIntExtra(BatteryManager.EXTRA\_SCALE, -1)  
 }  
  
 return batteryLevel  
 }

最后，我们完成之前添加的onMethodCall方法。我们需要处理平台方法名为getBatteryLevel的调用消息，所以我们需要先在call参数判断调用的方法是否为getBatteryLevel。 这个平台方法的实现只需调用我们在前一步中编写的Android代码，并通过result参数返回成功或错误情况的响应信息。如果调用了未定义的API，我们也会通知返回： ​

MethodChannel(flutterView, CHANNEL).setMethodCallHandler { call, result ->  
 if (call.method == "getBatteryLevel") {  
 val batteryLevel = getBatteryLevel()  
 if (batteryLevel != -1) {  
 result.success(batteryLevel)  
 } else {  
 result.error("UNAVAILABLE", "Battery level not available.", null)  
 }  
 } else {  
 result.notImplemented()  
 }  
}

您现在就可以在Android上运行该应用程序。如果您使用的是Android模拟器，则可以通过工具栏中的“…”按钮访问Extended Controls面板中的电池电量。