**Flutter笔记**

模板：

**3 某功能开发**

**3.1 开发过程**

**3.2 问题**

**3.3 后话**

主要内容：

对flutter进行开发和学习的记录笔记，记录flutter形成之旅。

寄语：

希望能提升编写能力和巩固知识，保持对编程的热爱。

编写时间始：2020-02-20

编写人：雷民用

目录

[1相机功能 4](#_Toc4381)

[1.1 开发过程 4](#_Toc17270)

[1.1.1 运行API提供的例子 4](#_Toc2185)

[1.1.2 使用流程 5](#_Toc28137)

[1.2 问题 6](#_Toc1329)

[1.3 后话 6](#_Toc31119)

[2 图片浏览 7](#_Toc31881)

[2.1 开发过程 7](#_Toc26446)

[2.1.1 运行库提供的例子 7](#_Toc20499)

[2.1.2 使用流程 7](#_Toc19954)

[2.1.3 困难的邂逅 8](#_Toc20861)

[2.2 问题 8](#_Toc7768)

[2.3 后话 9](#_Toc1158)

[3 GPS开发（尚未解决） 10](#_Toc12848)

[3.1 开发过程 10](#_Toc30023)

[3.1.1 困难的邂逅 11](#_Toc16816)

[3.2 问题 13](#_Toc28056)

[3.3 后话 13](#_Toc32742)

[4 列表开发 13](#_Toc27291)

[4.1 开发过程 13](#_Toc23179)

[4.1.1 使用方式 13](#_Toc31679)

[4.1.2 分析例子代码 13](#_Toc392)

[4.1.3 SmartRefresher属性 16](#_Toc16052)

[4.1.4 配合数据库的开发 16](#_Toc15242)

[4.2 问题 18](#_Toc2087)

[4.3 后话 18](#_Toc30813)

[5 provide使用 18](#_Toc1861)

[5.1 provide心得 18](#_Toc22239)

[5.1.1 是否更新 18](#_Toc25163)

[5.2 问题 18](#_Toc17355)

[5.3 后话 18](#_Toc31923)

[6 dart基础 18](#_Toc16306)

[6.1 数据处理 18](#_Toc23491)

[6.1.1 List 和String的转化 18](#_Toc3541)

[6.2 问题 19](#_Toc1153)

[6.3 后话 19](#_Toc23235)

[7 SharedPreferences 19](#_Toc18012)

[7.1 开发过程 19](#_Toc8418)

[7.1.1 使用方式例子 19](#_Toc16396)

[7.2 问题 20](#_Toc592)

[7.3 后话 20](#_Toc6254)

[8 定时器 20](#_Toc19136)

[8.1 开发过程 20](#_Toc29560)

[8.1.1 定时一次 20](#_Toc8505)

[8.1.2 循环定时 20](#_Toc11114)

[8.2 问题 21](#_Toc28904)

[8.3 后话 21](#_Toc22762)

[9 网络监听 21](#_Toc2402)

[9.1 开发过程 21](#_Toc18611)

[9.1.1 获取当前网络状态 21](#_Toc14852)

[9.1.2 实时的网络监听 21](#_Toc13303)

[9.2 问题 22](#_Toc2393)

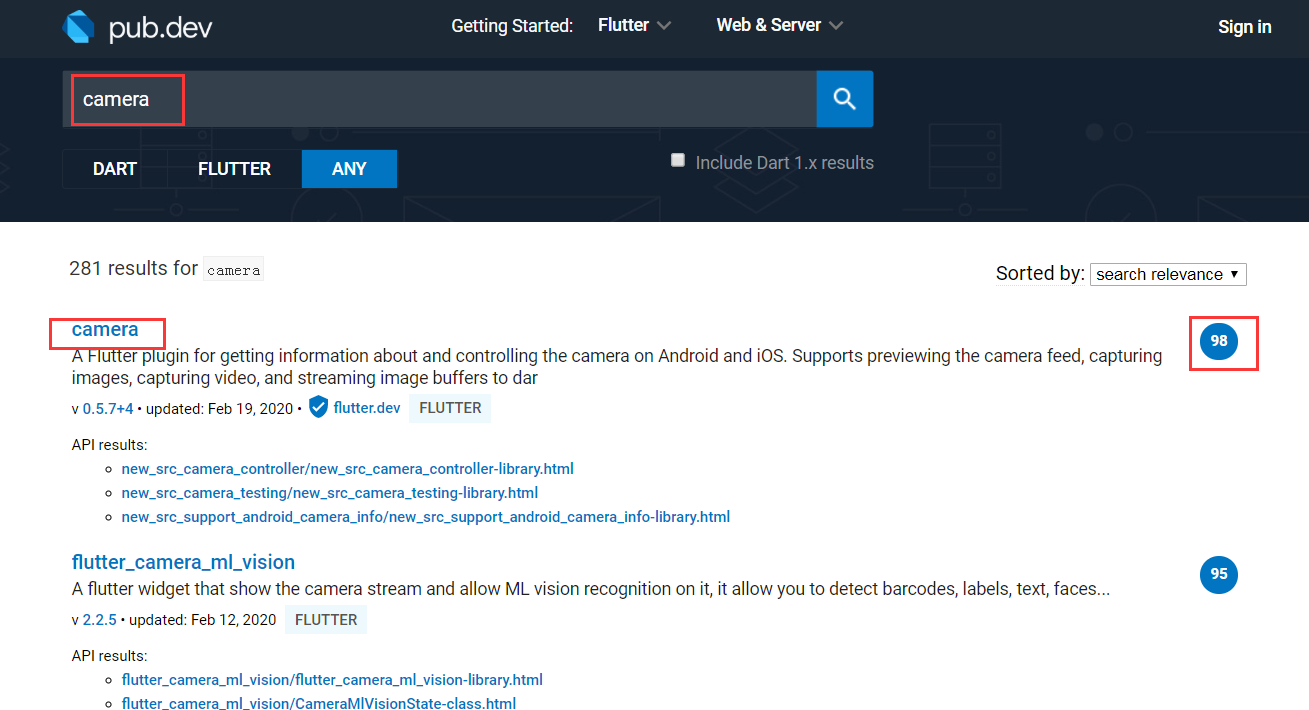
[9.3 后话 22](#_Toc30895)

**1****相机功能**

使用谷歌API camera （开发时间2020-02-20）。

* 1. **开发过程**

搜索pub.dev（<https://pub.flutter-io.cn/>）是不是有合适的第三方库，搜索关键词“camera”，有很多的三方库供我们使用。我直接排行最高的测试是否合适本项目的需求，因此进行了对camera API进行了了解和使用。



**图1.1 搜索结果**

* + 1. **运行API提供的例子**

注意：本次开发的项目无没包含androidx。

（1）初步尝试

直接在项目中进行测试，无法运行。报错并有修改提示，百度搜索试图解决，但是不成功。（大概的意思：是没用启动androidx相关的API，网络上也有suport和androidxAPI不能共存，我也猜测发行的最新版本的API可能使用androidx）。

1. 第二次尝试

提出：既然我们知道问题的所在，API的使用需要提供对androidx的支持。

我重新建立一个新的空项目，此空项目提供了对androidx的支持。使用空项目运行API提供的例子，发现是能成功运行的。

1. 第三次尝试

提出：是否是一定是只要提供对androidx的支持就行了，还是我开发的项目本身存在问题？

重新建立一个不对androidx的支持，测试无法运行，并且报的和开发项目的一样的错。我也根据提示进行了修改和百度修改，还是无法运行。

结论：由此得到的是，使用API必须提供对androidx的支持结论。

1. 第四次尝试

因为开发的项目，是不能提供对androidx的支持的（提供了前面开发的插件无法使用，并且一开始开发项目也设定好不能对androidx的支持。），这个API是官方的API，不至于这么狭窄吧？肯定是有解决的方法的？

提出：是不是API版本太高（高版本需要androidx的支持）？

降低API的版本，发现新建的没有androidx支持的新项目可以运行。惊喜出现。

结论：降低API的版本。

* + 1. **使用流程**

通过看例子的功能流程我们可以知道，相机主要有以下几个步骤。

1. 获取摄像头。

|  |
| --- |
| List<CameraDescription> cameras = [];  cameras = await availableCameras(); |

（2）获取摄像头的控制器（controller）。

|  |
| --- |
| controller = CameraController(  CameraDescription,  ResolutionPreset.medium,  enableAudio: false  ); |

（3）初始化，更新UI

|  |
| --- |
| controller.initialize().then((\_){  if(!mounted){  return;  }  setState((){});  }) |

（4）显示画面

|  |
| --- |
| CameraPreview(controller); |

（5）拍照

|  |
| --- |
| await controller.takePicture(filePath); |

（6）显示照片

|  |
| --- |
| Image.file(File(imagePath)); |

* 1. **问题**

（1）API版本太高，于项目不兼容androidx。

解决方式：降低API使用版本

* 1. **后话**

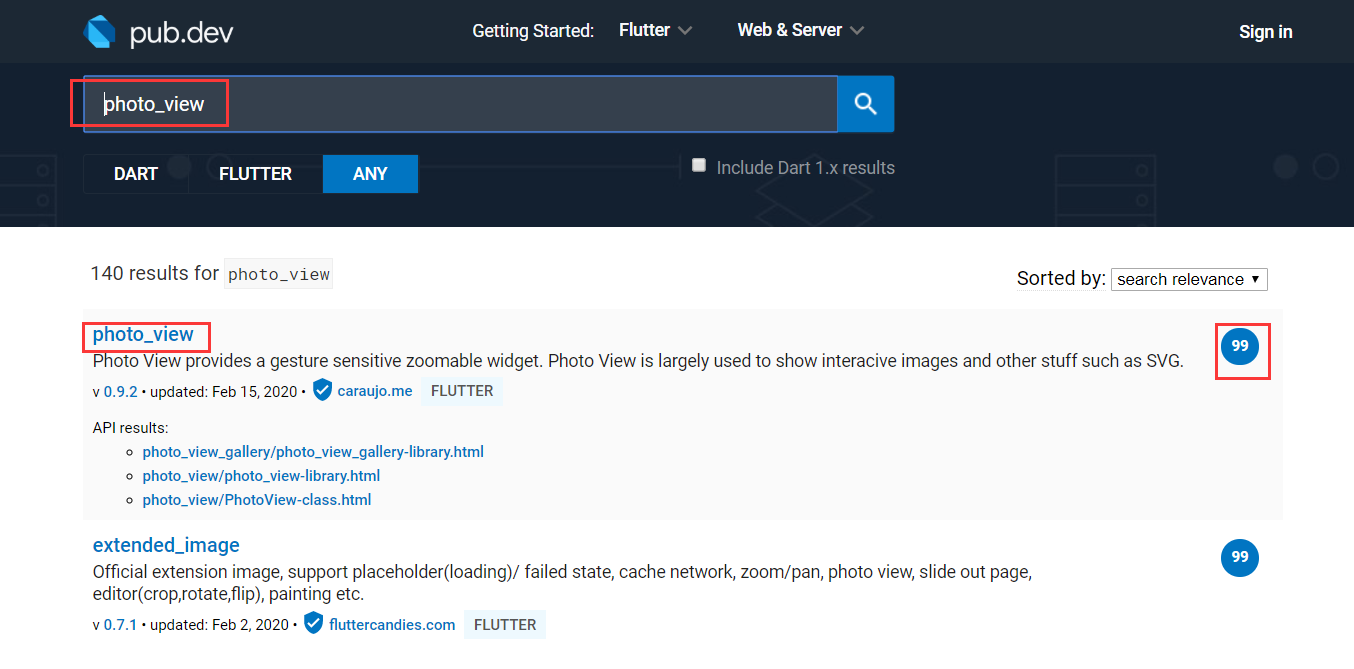
（1）2020-02-27号，停下。我对这个API，没太过深入。目前完成的功能有，打开摄像头，并进行拍照。

**2 图片浏览**

使用第三方photo\_view （开发时间2020-02-27）。

**2.1 开发过程**

百度搜索“flutter图片浏览”，得知第三方库“photo\_view”。在搜索pub.dev（<https://pub.flutter-io.cn/>），搜索关键字：“photo\_view”。



**图2.1 搜索结果**

**2.1.1 运行库提供的例子**

结果发现，例子没这么简单，不懂怎么去使用；最后是进行百度求医。直接搜索“photo\_view”的使用，让你知道网友的强大。

1. 存在版本兼容问题：photo\_view的下载库，下载了最后版本，发现和项目的不兼容。最后通过降低库版本进行解决问题。

**2.1.2 使用流程**

（1）新建图片浏览界面。此界面使用的是photo\_view库的API进行编写。关键的是：PhotoView()，用来显示图片的。

PhotoView关键属性：

imageProvider: [使用AssetImage()或者Image.network()]

1. 在需要浏览图片处进行跳转到浏览界面。

总结：

①首先明白，图片展示和图片浏览是两个界面。

②进行图片浏览就是跳转到浏览界面，同时把图片传过去。



**图2.2 使用流程**

**2.1.3 困难的邂逅**

（1）无法加载出图片。

图片是刚刚照相机拍好的图片，并且存储在手机的内部存储空间（即只有app自己内访问的空间上）。现在只知道图片的的存储路径下，使用组件：PhotoView进行显示出来，结构图片无法显示（浏览时，一直在加载，转圈圈）。

解决：

解决时间：2020-03-02 17:50

解决方式：FileImage 从本地文件系统加载图片。

|  |
| --- |
| imageProvider: FileImage(File(“本地图片的文件路径”));  如：  imageProvider: FileImage(File(“/data/user/0/www.jiliantech.com.jl\_light\_app/app\_flutter/Pictures/1583120843134.jpg”)); |

**2.2 问题**

（1）API版本太高，于项目不兼容androidx。

解决方式：降低API使用版本

（2）知道绝对路径下的图片加载。

知识补充：

AssetImage 加载Asset图片

NetworkImage 加载网络图片

FileImage 从本地文件系统加载图片

MemoryImage 从内存中加载图片

所以使用：FileImage 进行拍照后对图片的显示。

**2.3 后话**

（1）2020-02-28号，停下。我对这个API，没太过深入。目前完成的功能有，只进行单一的一张图片进行浏览，自由放大缩小，翻转。photo\_view库有很多强大的功能，可以进行很多张图片的浏览效果等。再往后可以对其进行深入了解和开发。

（2）2020-03-02号，发现bug，问题。就是对图片显示使用的组件需要有所了解。本次是对手机上的图片进行显示，所以使用组件：FileImage ，就能很轻松的解决问题。

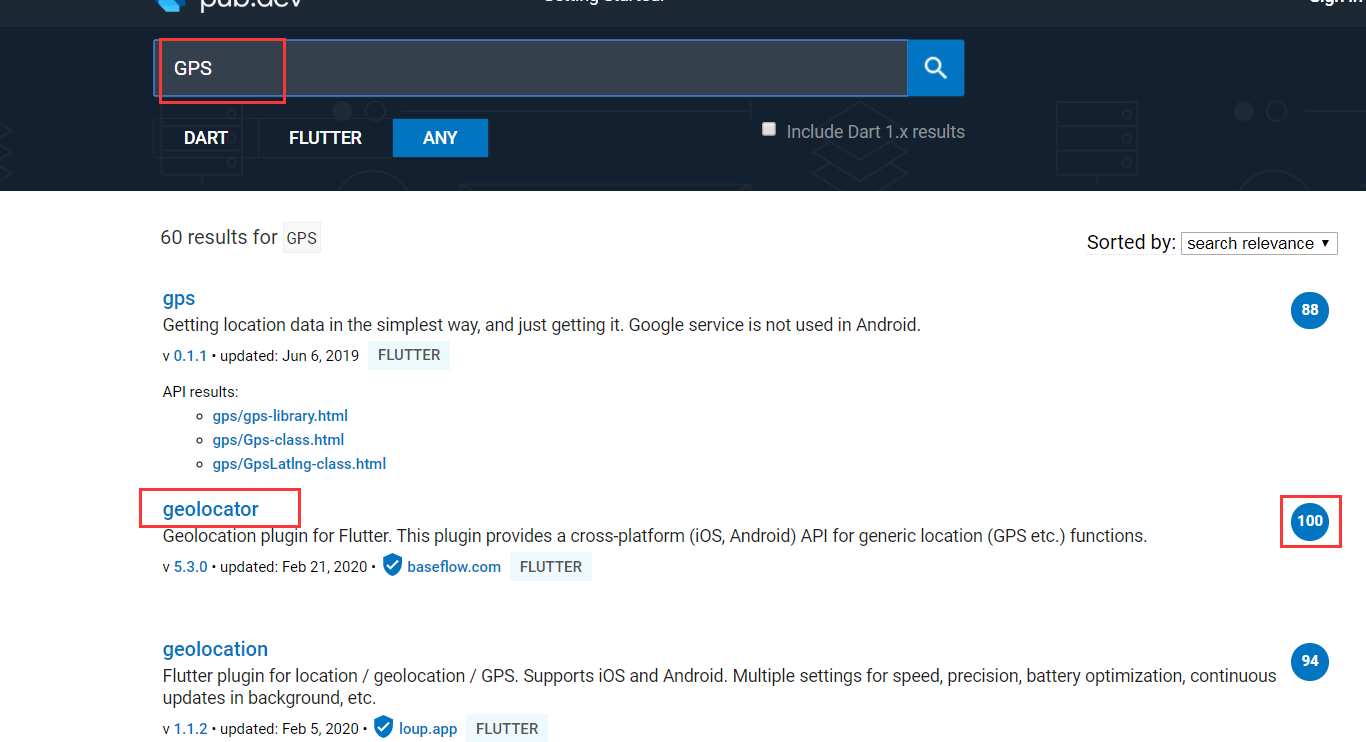
**3 GPS开发（尚未解决）**

开始开发时间：2020-03-02

使用第三方库gps 0.1.1

**3.1 开发过程**

在pub.dev直接搜索：GPS，寻找合适自己的第三方库。



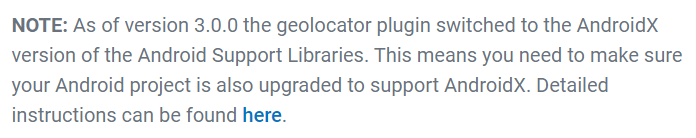
**图3.1 搜索结果**

因为，使用评价高，所以尝试这个第三方库。

库使用提示：

从版本3.0.0开始需要android支持库提升到androidx库的支持。

注意：现在的项目是不对androidx的支持的，所以下载版本要在3.0.0以下。



**图3.2 库使用提示**

**3.1.1 困难的邂逅**

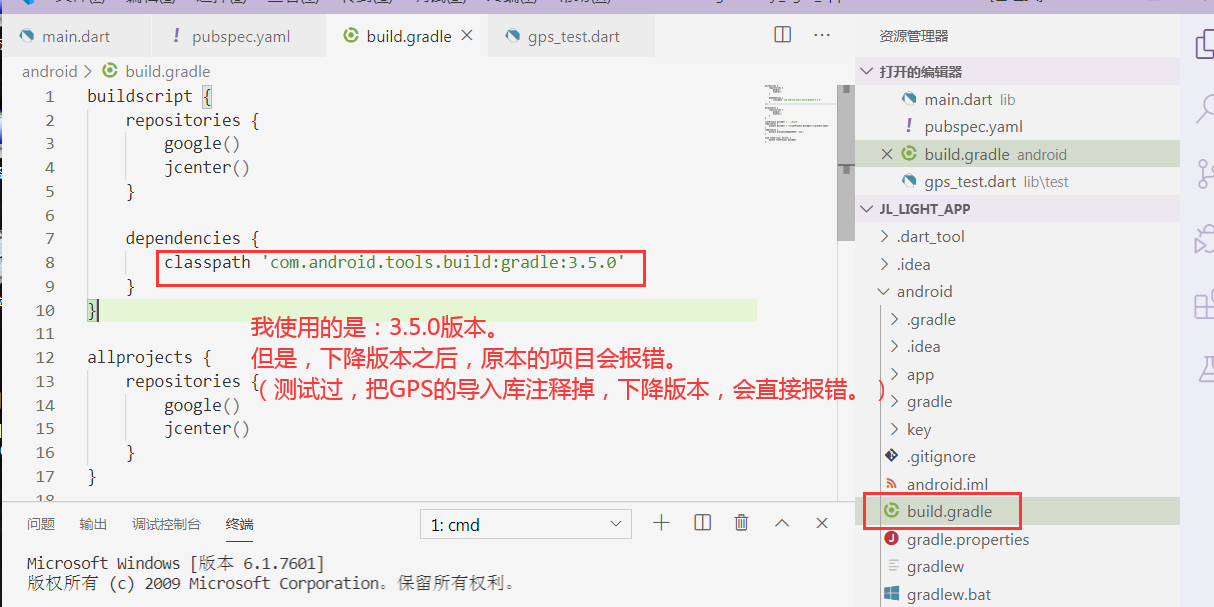
（1）报错。

|  |
| --- |
| \* What went wrong:  The Android Gradle plugin supports only Kotlin Gradle plugin version 1.3.10 and higher.  The following dependencies do not satisfy the required version:  project ':google\_api\_availability' -> org.jetbrains.kotlin:kotlin-gradle-plugin:1.2.71 |

搜索到网友的解答：

|  |
| --- |
| 您可以通过在项目的根目录上运行flutter packages pub deps来查找依赖于google\_api\_availability的程序包 – 这将在树视图中列出项目的所有直接和传递依赖项.  我找不到显示包的插件依赖关系的方法 – 我想你只会在尝试构建它时找到它.  问题是您使用的是Android Gradle插件的3.3.1版本,该插件强制执行Kotlin 1.3.0或更高版本.与此同时,geolocator包依赖于google\_api\_availability,这似乎是使用Kotlin 1.2.71.目前没有使用Kotlin 1.3.0或更高版本的google\_api\_availability版本,因此您只有1个解决方案 – 将Android Gradle插件降级到版本3.2.1. |

我做出的解决方式与结果：



**图3.3 解决截图与结果**

结果：从图中可以看出，我通过降低版本，之后，原本的项目直接报错了。

注意：我重新新建项目（提供对androidx的支持），我直接导入最新的（geolocator库版本，也就是说需要支持androidx的版本），是可以运行的。由此我有新的解决思路，就是重新建立一个androidx的支持，把项目迁移过来。而且，目前看到的库大多数都是需要对androidx的支持。

时间：2020-03-04

解决结果：毫无进展。

进行了一以下的解决方式：

（1）重新建立项目，想对项目提升到androidx，行不通。

串口插件是不兼容androidx的，如果提升到androidx的话，需要重新开发对串口的读写，或者使用其他人的串口读写库（我进行了搜索pub，是有可用的三方库。）。无论选着这两种方式的那种方式，时间是消耗很长的一段时间的，时间上不允许。

（2）想找到没有提升到androidx的GPS功能库。

我寻找了搜索了pub，看了几个库，都是需要androidx的支持的。

总结：

（1）官方也有兼容androidx的意见。

1. 把项目提升到androidx
2. 再使用第三方库时，尽量不要使用最新的库。

（2）由此可见，趋势都是androidx了。

**3.2 问题**

（1）API版本太高，于项目不兼容androidx。

待解决。。。

**3.3 后话**

**4 列表开发**

使用库：“pull\_to\_refresh”，主要实现的功能是，上拉加载功能。需要配合数据库，sqlline，上拉一次就查询一次数据库，进行数据的格式化和展示，当没有数据返回时提示，没有更多数据。

**4.1 开发过程**

**4.1.1 使用方式**

（1） 把下面这行添加到pubspec.yaml

|  |
| --- |
| dependencies:  pull\_to\_refresh: ^1.5.7 |

（2）import package

|  |
| --- |
| import 'package:pull\_to\_refresh/pull\_to\_refresh.dart'; |

**4.1.2 分析例子代码**

|  |
| --- |
| List<String> items = ["1", "2", "3", "4", "5", "6", "7", "8"];  //1、refreshController 控制器  RefreshController \_refreshController = RefreshController(initialRefresh: false);  void \_onRefresh() async{  await Future.delayed(Duration(milliseconds: 1000));//模拟数据加载的延迟  \_refreshController.refreshCompleted();//成加载  //\_refreshController.refreshNoData();//没有更多数据  //\_refreshController.refreshFailed();//加载失败  }  void \_onLoading() async{  await Future.delayed(Duration(milliseconds: 1000));  items.add((items.length+1).toString());  if(mounted)  setState(() {  });  \_refreshController.loadComplete();  //\_refreshController.loadNoData();//没有更多数据  //\_refreshController.loadFailed();//加载失败  }  @override  Widget build(BuildContext context) {  return Scaffold(  body: SmartRefresher(  enablePullDown: true,//下拉出发事件  enablePullUp: true,//上拉出发事件  header: WaterDropHeader(),//顶部加载loading UI  footer: CustomFooter( //底部加载loading UI  builder: (BuildContext context,LoadStatus mode){  Widget body ;  if(mode==LoadStatus.idle){  body = Text("pull up load");  }  else if(mode==LoadStatus.loading){  body = CupertinoActivityIndicator();  }  else if(mode == LoadStatus.failed){  body = Text("Load Failed!Click retry!");  }  else if(mode == LoadStatus.canLoading){  body = Text("release to load more");  }  else{  body = Text("No more Data");  }  return Container(  height: 55.0,  child: Center(child:body),  );  },  ),  controller: \_refreshController, //绑定控制器  onRefresh: \_onRefresh, //下拉回调函数  onLoading: \_onLoading,//上拉回调函数  child: ListView.builder(//子组件是普通的listView  itemBuilder: (c, i) => Card(child: Center(child: Text(items[i]))),  itemExtent: 100.0,  itemCount: items.length,  ),  ),  );  } |

**4.1.3 SmartRefresher属性**

这只是分析例子，总结的SmartRefresher属性归纳。同样的更定有更多是属性等待我们去发现。

|  |
| --- |
| SmartRefreshe(  enablePullDown,//是否触发下拉事件  enablePullUp,//是否触发上拉事件  header,//顶部loading UI  footer,//底部loading UI  controller,//控制器，通知组件现在是什么状态。  onRefresh,//下拉回调函数  onLoading,//上拉回调函数  child,//主题内容  ); |

**4.1.4 配合数据库的开发**

由上面的分析，只要在进行下拉或者上拉的的回调函数中，进行数据库的查询，再进行UI的渲染。同时，有几种情况要注意一下：

1. 数据加载成。

controller.loadComplete();//controller.refreshComplete();

显示加载成功后的底部或者顶部UI

1. 数据加载失败。

controller.loadFailed();//controller.refreshFailed();

显示加载失败后的底部或者顶部UI

1. 没有更多数据

controller.loadNoData();//controller.refreshNoData();

显示没有更多数据的底部或者顶部UI

简单的例子如：

|  |
| --- |
| //假设，这是下拉加载的一个回调函数 refreshController为控制器  void \_onLoading()async{  print("上拉加载");  startModel++;//参数数据库需要的参数  try{  List<Map> result = await Global.devDB.queryLimitModel(this.limitModel, this.startModel);//查询数据库  if(result != null){  CheckModelList newcheckModelList = CheckModelList.fromJson(result);//数据格式化  //数据格式化  checkModelList.modelListData.insertAll(checkModelList.modelListData.length, newcheckModelList.modelListData);  //加载数据成功  \_refreshController.loadComplete();  }  else{  //没有更多数据  \_refreshController.loadNoData();  }  }catch(e){  //加载失败  \_refreshController.loadFailed();  print("发生异常：$e");  }  //更新UI  setState(() {  });  } |

**4.2 问题**

**4.3 后话**

**5 provide使用**

**5.1 provide心得**

**5.1.1 是否更新**

（1）UI会更新

|  |
| --- |
| **例子：**  **Provide**<**DateTimeProvide**>(  builder: (context, child, modelCheckProvide){      return **Text**(  **Provide**.**value**<**DateTimeProvide**>(context).**getDateTime**()==null?"":  **Provide**.**value**<**DateTimeProvide**>(context).**getDateTime**().**toString**()      );  }, |

（2）UI不会更新

|  |
| --- |
| **直接：**  **Provide**.**value**<**DateTimeProvide**>(context).**getDateTime**()==null?"":  **Provide**.**value**<**DateTimeProvide**>(context).**getDateTime**().**toString**()  **结果：可以获取到值的，但UI不会更新。** |

**5.2 问题**

**5.3 后话**

**6 dart基础**

主要在开发过程中对，基础知识的收集和总结。

**6.1 数据处理**

**6.1.1 List 和String的转化**

主要函数：join(' ') split(' ')

|  |
| --- |
| List<int> listInt = [1, 2, 3];  String str = listInt.join(',');  print(str); // 1, 2, 3  print(str is String); // true  var str1 = '1, 2, 3';  var list1 = str1.split(',');  print(list1); // [香蕉, 苹果, 西瓜]  print(list1 is List); // true |

**6.2 问题**

**6.3 后话**

**7 SharedPreferences**

轻量级存储（存放在磁盘中，即之久化存储），键值对的存储方式。

**7.1 开发过程**

**7.1.1 使用方式例子**

步骤：

|  |
| --- |
| 1. 安装依赖   dependencies:  shared\_preferences:^0.5.6+3   1. 使用   Future<SharedPreferences> \_prefs = SharedPreferences.getInstance();  异步方式1：  \_prefs.then((prefs){  String value = prefs.getString("name") ?? "";  });  异步方式二：  final SharedPreferences prefs = await \_prefs;  prefs.setString("name", ”value”); |

**7.2 问题**

**7.3 后话**

|  |
| --- |
| Future<SharedPreferences> \_prefs = SharedPreferences.getInstance(); |

从这个当中可以猜出，这个类是单例化的，所以在那个地方进行get或者set相同的key，应该是同一个地方（值，key，对象，属性）。

**8 定时器**

定时一段时间就开始调用回调函数。

**8.1 开发过程**

**8.1.1 定时一次**

代码分析：

|  |
| --- |
| var duration = **Duration**(seconds: 3);  **Future**.**delayed**(duration, (){  //三秒后次回调函数被调用-只调用一次  }); |

**8.1.2 循环定时**

代码分析：

|  |
| --- |
| const period = const **Duration**(seconds: 10);  \_timer = **Timer**.**periodic**(period, (timer) {     timer.**cancel**();*//取消定时器，如果不取消，会没十秒调用一次回调函数。*  });  \_timer.**cancel**();*//这种也是取消定会器。* |

**8.2 问题**

**8.3 后话**

**9 网络监听**

可以进行实时的网络监听和判断当时的网络状态。

**9.1 开发过程**

添加依赖：

|  |
| --- |
| dependencies:  connectivity:^0.4.8+2 |

**9.1.1 获取当前网络状态**

代码分析：

|  |
| --- |
| var connectivityResult = await (**Connectivity**().**checkConnectivity**());  if (connectivityResult == **ConnectivityResult**.mobile) {  //当前连接是手机网络  }  else if(connectivityResult == **ConnectivityResult**.wifi){  //当前连接的是wifi  }  else{  //没有网络  } |

**9.1.2 实时的网络监听**

时的网络监

|  |
| --- |
| *//网络状态描述*  **String** \_connectStateDescription;    var subscription;      @override    void **initState**() {  *//监测网络变化*  subscription = **Connectivity**().onConnectivityChanged.**listen**((**ConnectivityResult** result) {     if (result == **ConnectivityResult**.mobile) {  **setState**(() {         \_connectStateDescription = "手机网络";       });     } else if (result == **ConnectivityResult**.wifi) {  **setState**(() {         \_connectStateDescription = "Wifi网络";       });     } else {  **setState**(() {         \_connectStateDescription = "无网络";       });     }  **print**(\_connectStateDescription);  });  }    @override    void **dispose**() {  super.**dispose**();  subscription.cancle();//一定要取消监听网络    } |

**9.2 问题**

**9.3 后话**

**10 插件开发**

**10.1 开发过程**

**10.2 问题**

**10.3 后话**