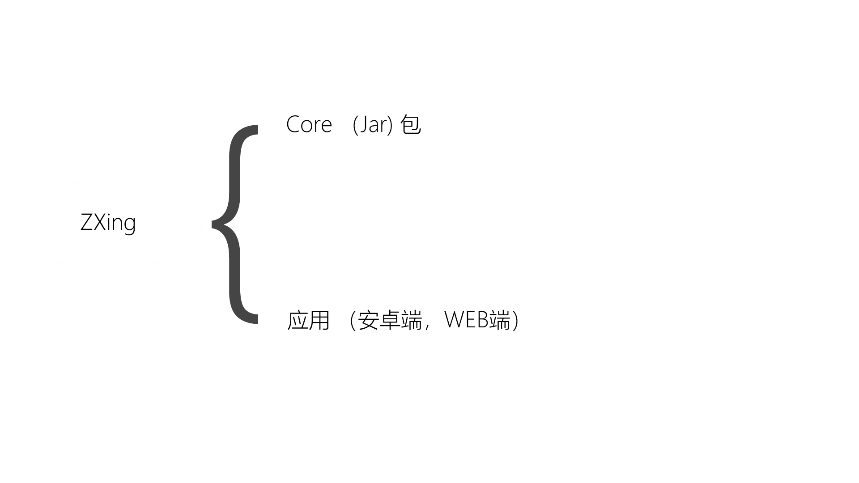
# 1 应用目的

1. 扫描条形码/二维码等以获取其中的信息
2. 将信息写进并生成二维码，用于分享
3. 核心算法打包成库，可以供其他人使用，制作网页端和安卓端应用

因此，整体视图图下：



Core

WEB端

Android:

环境：Android API 19 platform及以上。

目的：制作易使用的app，同时扫描Code的核心算法可以被其他应用程序所使用。（制作、打包成库）

用户需求：扫码，看到结果，对不同类型的结果提供不同操作。

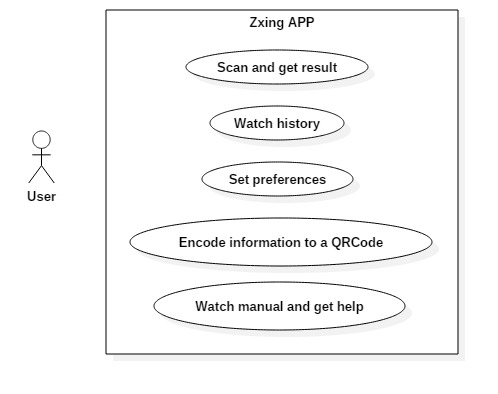
可以查看历史记录。

可以生成二维码。

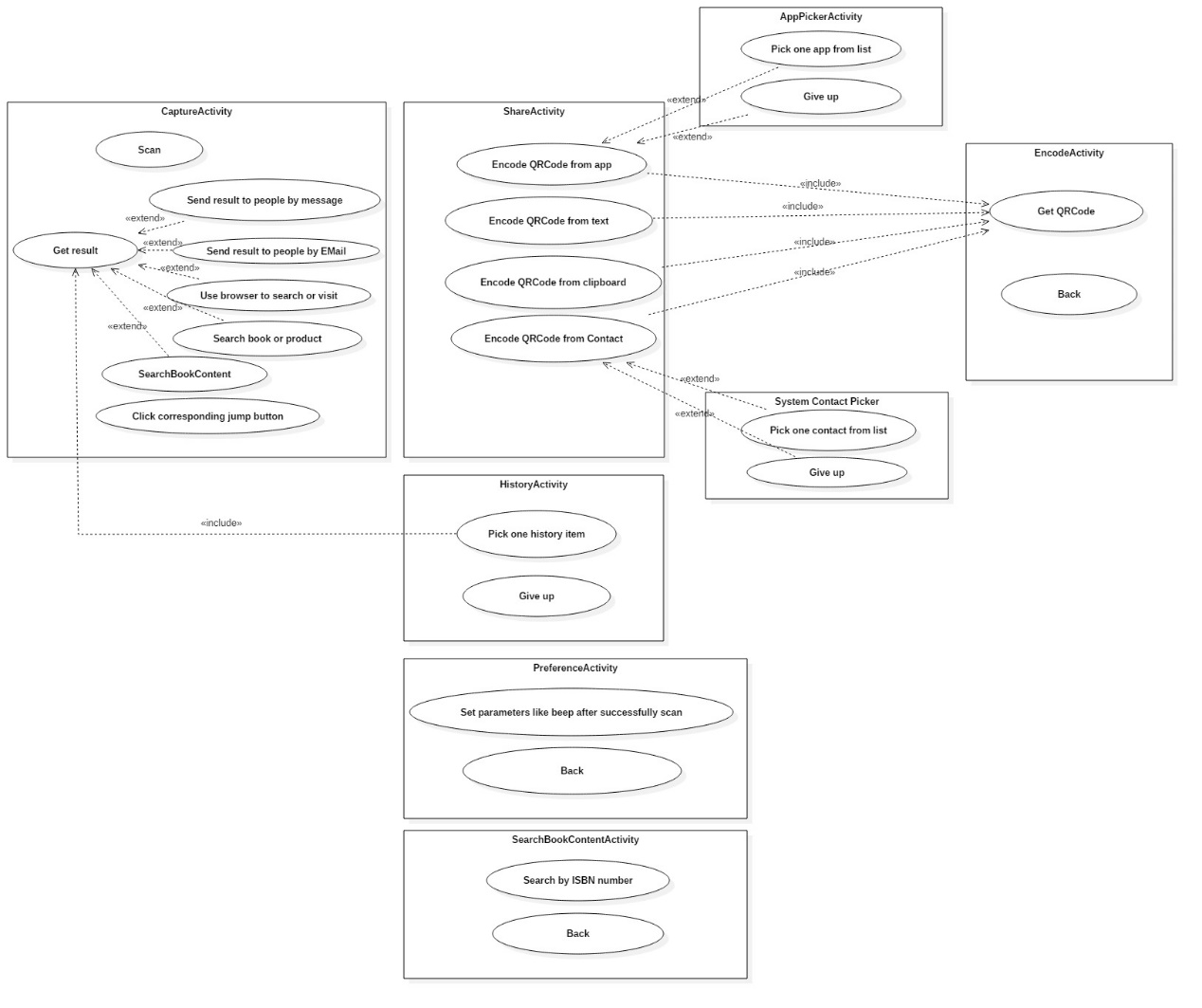
提供偏好设置和帮助页面。

## 1 用例视图

整个APP的用例如下：



分场景用例视图如下：



详细：CaptureActivity是主场景，也是程序一启动即进入的场景，在该场景中，相机保持启动，若发现1D/2D的码，则显示结果，除了显示必要信息外Code所携带的信息外，还有相关按钮，比如是个URL的话有用浏览器打开按钮和通过Message或Email分享按钮，WIFI的话有连接按钮等。上述使用功能为Android系统功能，特别地，如果是书本，提供了搜索书本内容，可以跳转到SearchBookActivity场景进行搜索。在CaptureActivity该场景中，有4个按钮可以分别跳转到生成二维码界面（ShareActivity），历史界面（HistoryActivity），设置界面（PreferenceActivity）和帮助界面

## 2 逻辑视图

2.1 概述

Package:

.client.android

.client.android.book

.client.android.camera

.client.android.camera.open

.client.android.clipboard

.client.android.encode

.client.android.history

.client.android.result

.client.android.result.supplement

.client.andoird.share

.client.android.wifi

Referenced Package：

com.google.zxing.core: 用于解析1D/2D的Code，并将结果以Result类的具体实现类返回。

com.google.zxing.client.android.camera.android-core: 相机工具类包。

.client.android 是核心。该包内的Java类包含主场景的内容及其控制等。

.client.android.book：提供SearchBookContent服务

.client.android.camera：是控制相机的核心包，其中的open包是用于打开相机的包。

.client.android.clipboard：提供剪贴板相关功能

.client.android.encode：提供生成二维码的编码功能，并提供全屏展示二维码功能

.client.android.history: 提供历史记录功能，管理历史记录。

.client.android.result：处理扫描结果的包。对于每一种存储不同类型信息的Code，都有一个相应的类。其中的.supplement包对搜索Code携带的额外信息提供服务。

.client.android.share：提供生成二维码时的信息选择功能，先选择要存进二维码的信息，再使用encode包进行编码并展示。

.client.android.wifi：提供控制wifi，连接wifi等功能

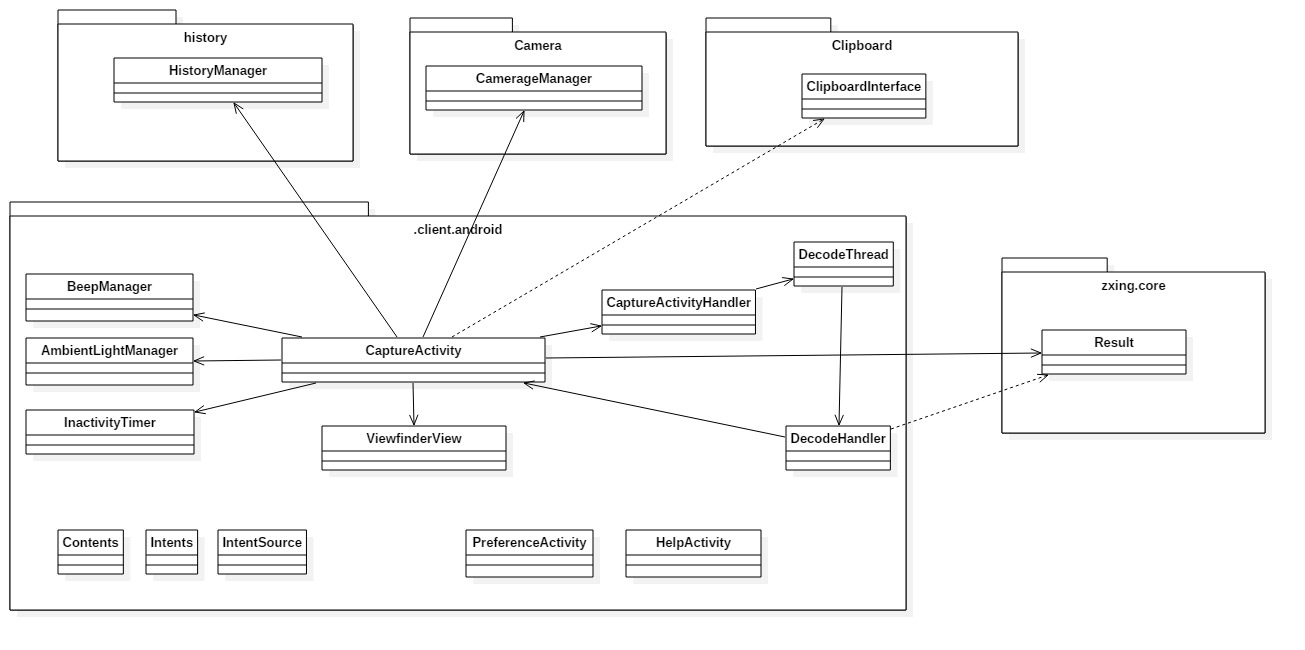
.client.android.share和.client.android.encode用于生成二维码并展示

.client.android.camera是为.client.android提供相机功能服务的。后者主要控制主场景，显示相机内容，展示扫码结果并对不同结果提供不同服务。除此之外，后者包内还有偏好设置，帮助页面。

其他包都是针对扫描Code所得到的Result服务的。.client.clipboard将Result中的信息赋值到剪切板。.client.android.history包将Result保存至数据库，并提供读取功能，读取之后在.client.android控制的主场景中展示。.client.android.result包对于每种不同type的Result都有不同的类规定了相应的操作，并展示于.client.android包中的主场景中。特别地，对于wifi，利用了.client.android.wifi包对wifi进行操作（打开系统的wifi，根据相应的wifi信息进行连接）；对于书本，利用了.client.android.book包搜索书本内容。除此之外，client.android.result对于Result的操作，大部分功能是利用系统功能，例如通过浏览器打开或搜索，通过Message或者Email分享

2.2 重要包的设计

## .client.android



CaptureActivity是中心。

BeepManager,AmbientLightManager,InactivityTimer三个类辅助主场景。BeepManager在扫Code成功时发出提示音（根据偏好设置），AmbientLightManager在光线较暗时打开闪光灯，在光线足够时关闭闪光灯，InactivityTimer启动一个异步任务，如果在手机未充电下，5分钟内没有成功扫出Code的话关闭程序。

通过CameraManager操作相机，包括初始化，开始，关闭等。

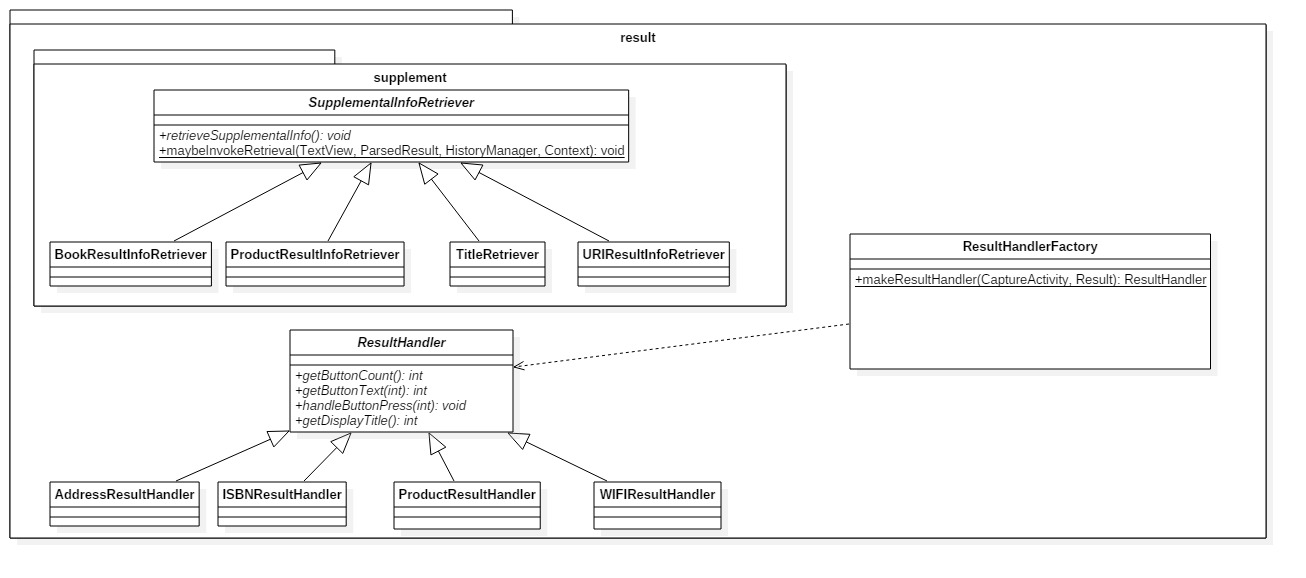
CaptureActivityHandler是归属于CaptureActivity所在线程的一个handler，主要用来接受通知，现在发生了什么，同时告知主场景该采取什么方法了。那么在这个handler创建的时候需要新建一个线程去处理解码，由该handler来控制该线程的生命周期。

DecodeThread继承于Thread类，该线程用来处理解码，该线程的handler是DecodeHandler，该handler主要接受来自于相机的信息，处理图片数据，得到Result结果。同时告诉CaptureActivityHandler结果。其实DecodeHandler关联的应该是CaptureActivityHandler，不过可以通过CaptureActivity获取其handler，且从Activity的生命周期和内存泄漏方面考虑，关联CaptureActivity。

在完成扫描后，CaptureActivity拿到了Result，拿到Result之后，要展示对吧。那么展示的信息是什么，我这个APP要对这个Code提供哪些操作。这个问题对不同的Result有不同的答案，因此有了result这个包，同时通过传递拿到的这个Result，让ResultHandlerFactory生成相应的ResultHandler给CaptureActivity。CaptureActivity再通过ResultHandler的接口展示相关信息。除此之外通过HistoryManager向历史记录增添一项。同时将结果放入剪切板（根据偏好设置和安全性）。还要通知InactivityTimer，当前这个APP这个人还在用，重新开始计时5分钟。

ViewfindView是自定义控件，用来凸显照相的中间部分，CaptureActivity负责控制其状态。在获得Result之后，CaptureActivity根据Result，在场景中展现结果，同时CaptureActivityHandler要记录上一次结果以便使用。

## .client.android.result



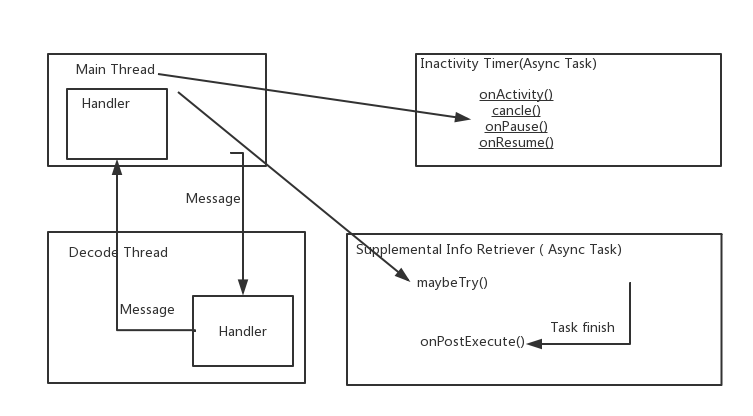
这个包名字叫result，只是个名字。主要是对Result的处理Handler。考虑到代码复用性以及未来Code可能存储其他类型信息的话，应该有一个抽象的类，这里就是ResultHandler。该类里有方法让具体结果的Handler去实现各自的处理方法。

那么，由于我们的界面是打算用按钮进行后续操作，所有这里有getCount，getButtonText，handleButtonPress，分别为这个结果的时候有几个按钮要显示，按钮上的文字是什么，这个按下这个按钮所进行的操作。在外面有一个工厂类，把Result进来，根据Result类型返回相应的ResultHanlder。

此外，从图里可见：里面还有一个supplement用作额外信息检索用。这个包跟外面的类一样，有个抽象类，利用多态使编码方便。这个类还兼顾跟外面的工厂类一样的工厂功能，传进来ParsedResult，以及相应的需要显示额外信息的View，并记录到History。

## 3 进程视图

3.1 进程图



2个Thread都拥有各自的handler，通过Handler传送信息，在各自线程上传送，在各自线程中处理。

我们有一个目标是5分钟无成功扫描关闭程序，那么需要有一个线程计时。我们使用AsyncTask，这个对线程的创建，运行等进行了包装。然后我们可以通过onActivity(), cancle(), onPause(), onResume()控制异步任务的生命周期。（这些函数是在主线程中运行的）。

对于需要搜索额外信息的时候，有可能要进行互联网操作，是费时操作，我们也放在异步任务中，通过maybeTry()从线程池中启动异步任务，在异步任务完成后，在onPostExecute（）（该方法在主线程中，此时异步任务已完成，异步任务的线程已结束）中进行界面更新等操作。

其实这个程序有对本地数据库的读写，虽然数据库比较小，而且一般不会有很多跳记录，但是Google没有用异步任务去完成，而是在主线程中完成的，感觉用异步会比较好。

