**实验报告**

目录

[一、系统需求分析 2](#_Toc10810)

[1.1开发目的 2](#_Toc15579)

[1.2 产品功能描述 2](#_Toc26587)

[1.3系统功能描述 2](#_Toc24674)

[二、界面设计 2](#_Toc25510)

[2.1 activity\_article.xml 3](#_Toc18175)

[2.2 item\_layout01.xml 3](#_Toc31349)

[2.3 news\_list.xml 3](#_Toc6332)

[2.4 nav\_header.xml 4](#_Toc24671)

[2.5 item\_article.xml 4](#_Toc31936)

[2.6 activity\_main.xml 4](#_Toc32172)

[三、关键代码技术 5](#_Toc10477)

[3.1 tools 5](#_Toc19596)

[3.1.1 ActivityCollector 5](#_Toc9021)

[3.1.2 BaseActivity 5](#_Toc1390)

[3.1.3 DataCleanManager 6](#_Toc27451)

[3.2 usermodel 7](#_Toc5931)

[3.2.1 Article 7](#_Toc13318)

[3.2.2 ArticleAdapter 7](#_Toc4446)

[3.2.3 ArticleActivity 9](#_Toc23498)

[3..2.4 EditArticleActivity 9](#_Toc6813)

[3.2.5 ArticleDetailActivity 11](#_Toc29580)

[3.2.6 UserInfo 11](#_Toc6521)

[3.2.7 UserDetailActivity 12](#_Toc694)

[3.2.8 RegisterActivity 12](#_Toc11877)

[3.2.9 LoginActivity 12](#_Toc19414)

[3.2.10 FavoriteNewsListAdapter 13](#_Toc19961)

[3.2.11 UserFavoriteActivity 14](#_Toc17503)

[3.3 主模块 15](#_Toc1581)

[3.3.1 NewsBean 15](#_Toc16245)

[3.3.2 NewsCollectBean 15](#_Toc3705)

[3.3.3 NewsInfoBean 15](#_Toc3076)

[3.3.4 NewsFragment 16](#_Toc521)

[3.3.5 MainActivity 21](#_Toc2233)

[3.3.6 TabAdapter 25](#_Toc23643)

[3.3.7 WebActivity 26](#_Toc1172)

[3.4 数据库模块 27](#_Toc21504)

[3.4.1 数据库配置 27](#_Toc14629)

[四、 完成效果展示 28](#_Toc19559)

[五、 总结 32](#_Toc21440)

**一、系统需求分析**

**1.1开发目的**

新闻APP开发的主要目的是信息传递，以其专业、全面的新闻报道平台，为用户提供阅读，将最新、最热的新闻传递到用户的智能设备。 目前我国的手机用户已经达到8亿多，查阅新闻资讯所占比例最高，达54.5%，新闻APP软件几乎成为装机必备第三方应用软件。简易新闻App的开发是为了让用户能够实时查看最新消息以了解社会动态，增长知识，增广见闻，顺便娱乐一下、放松自己。

**1.2 产品功能描述**

简易新闻App是基于Android Studio编辑器上开发的一款可以查看、添加、删除新闻，注册登录，编辑资料，查看个人收藏，清除缓存的安卓App。App通过获取聚合数据的新闻接口来实现新闻数据的展示。由于每日限制请求次数，故每次请求时就缓存新闻数据到本地数据库中。当每次下拉刷新就分页查询20条新闻数据重新显示在对应的tab碎片中。用户可以根据自己的兴趣点击对应的tab标签来阅读新闻，可以对新闻项进行收藏和分析，清除本地缓存，登录后（取消）收藏新闻，查看个人收藏，添加、查看、删除个人文章等。

**1.3系统功能描述**

登录、注册模块：注册过的用户登录后才能（取消）收藏新闻，查看个人收藏，添加、查看、删除个人发布的文章等。

个人信息模块：用户登录后可以查看、修改个人资料。

个人文章模块：用户登录后可以发布、查看、删除文章。

个人收藏模块：用户登录后可以（取消）收藏新闻接口数据子项，并且查看自己收藏的所有新闻。

清除缓存模块：该功能主要清除浏览网页时在本地留下的离线内容和图片缓存。

展示新闻列表模块：该功能主要在用户打开App时，就请求聚合数据提供的新闻接口来显示在tab页面并将请求的数据缓存到本地数据库中。若当日请求次数已用完，则分页查询本地数据库。

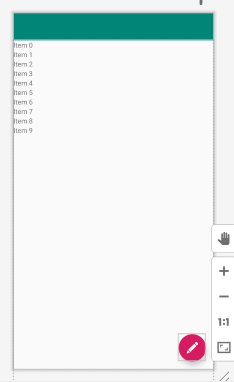
**二、界面设计**

**2.1 activity\_article.xml**

采用CoordinatorLayout + FloatingActionButton的方式进行布局，使用CoordinatorLayout 来布局，其主要有以下两个作用：

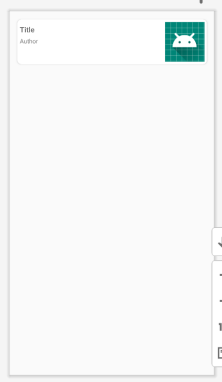
1. 作为顶层布局
2. 作为协调子 View 之间交互的容器

CoordinatorLayout 提供了设置 FAB 的位置的属性。



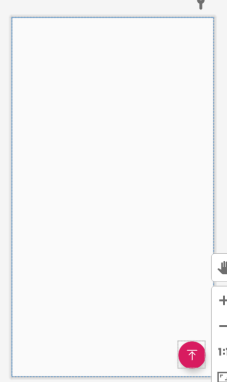
**2.2 item\_layout01.xml**

用于新闻中包含一张图片的界面，整体采用CarView，图片和文本之间采用相对布局，文本在图片的左边。文本之间同样是相对布局，标题在发布者的上方。



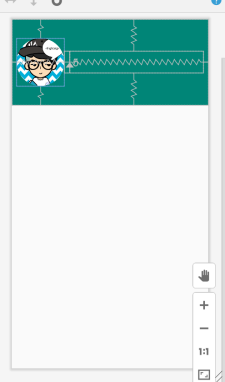
**2.3 news\_list.xml**

帧布局，存放新闻的list，并用下拉刷新的控件SwipeRefreshLayout来包裹住list，使用户通过下拉能够刷新新闻list，右下角有FloatingActionButton控件，能够一键回到顶部。



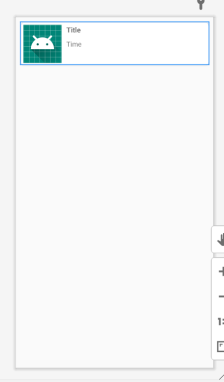
**2.4 nav\_header.xml**

相对布局，昵称相对于头像在右边，签名相对于在昵称的下面，使用CircleImageView将头像圆形化。



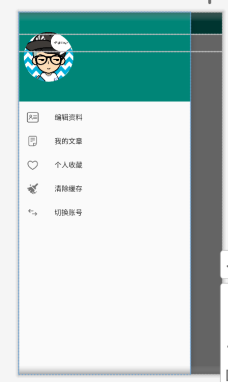
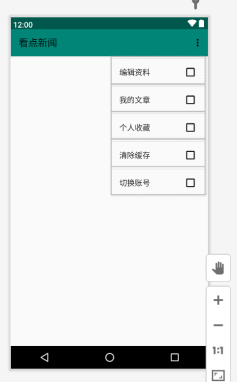
**2.5 item\_article.xml**

个人文章列表，卡片布局，作为子项的布局，使得每张图片都是在卡片当中，和nav\_header.xml类似的相对布局。



**2.6 activity\_main.xml**

DrawerLayout，官方给我们提供的一个侧滑菜单控件，主页面部分从上到下依次为为：Toobar+TabLayout+ViewPager，侧边栏部分从上到下为：nav\_header+nav\_menu。

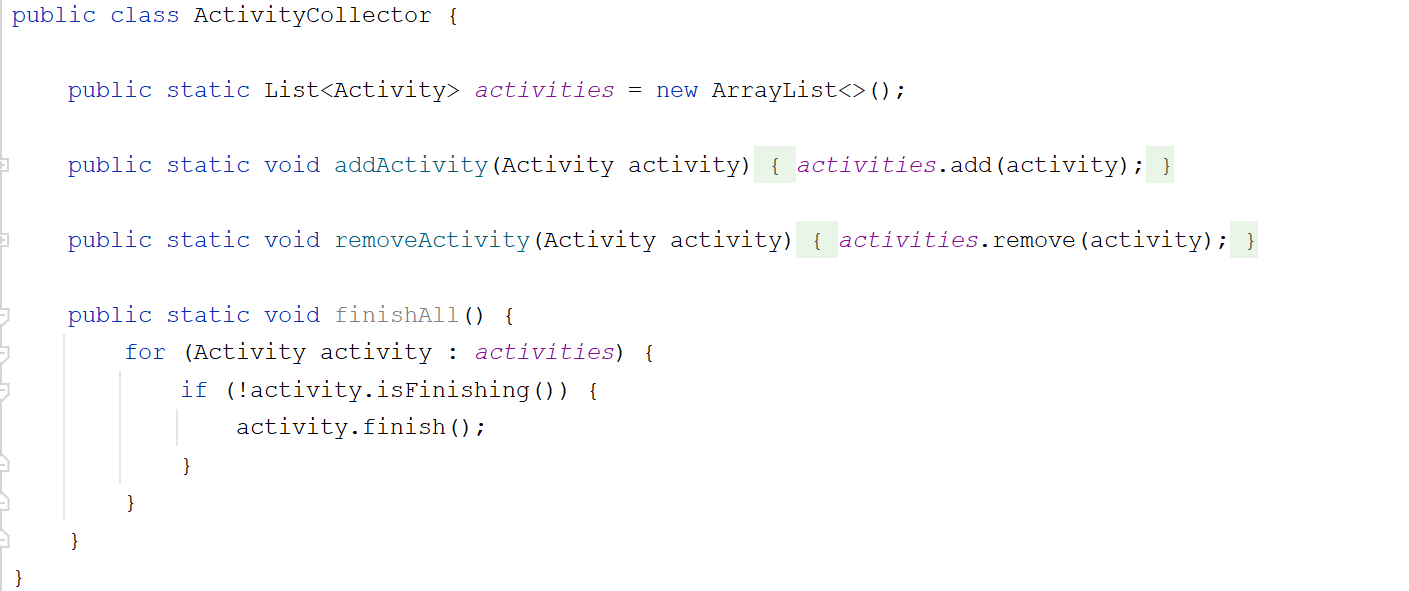
 

**三、关键代码技术**

**3.1 tools**

**3.1.1 ActivityCollector**

定义了public静态变量——列表activities，通过列表activities来统计activity的单个增减、全部释放。



**3.1.2 BaseActivity**

基本activity类，其他的Activity只要继承BaseActivity就可以，在其onCreate和onDestroy方法中分别实现了对activities的动态增减。



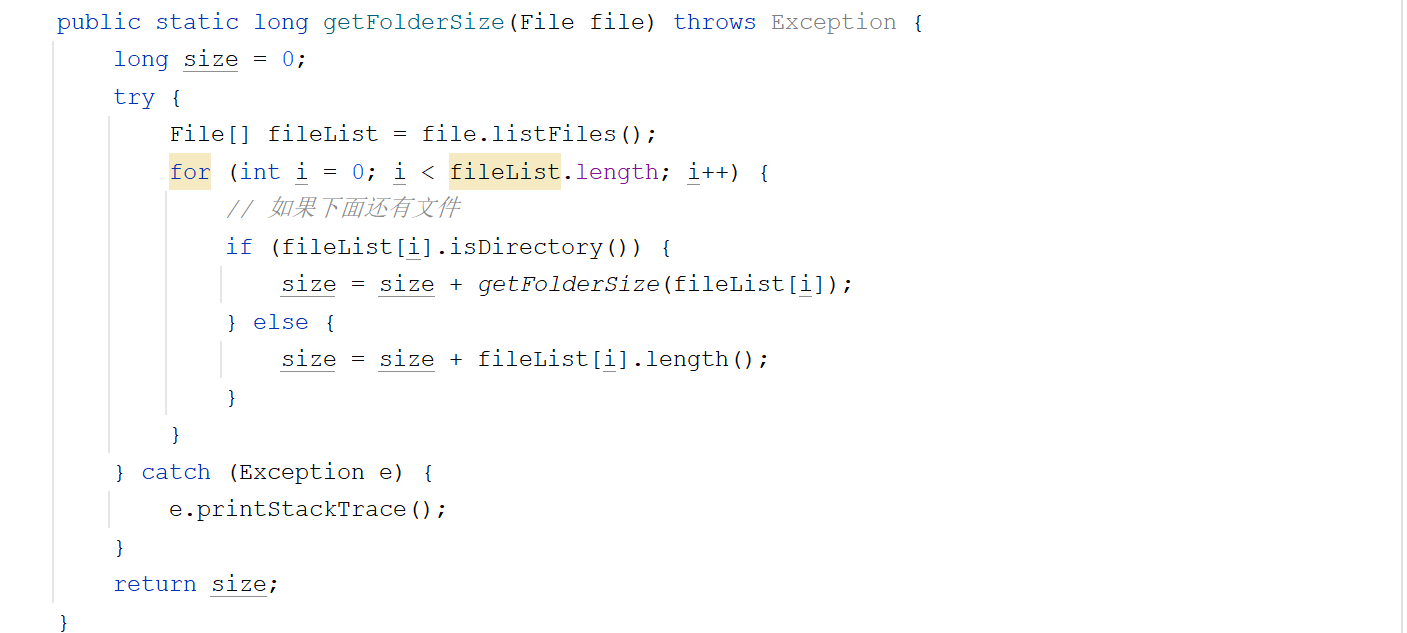
**3.1.3 DataCleanManager**

数据清除管理器：

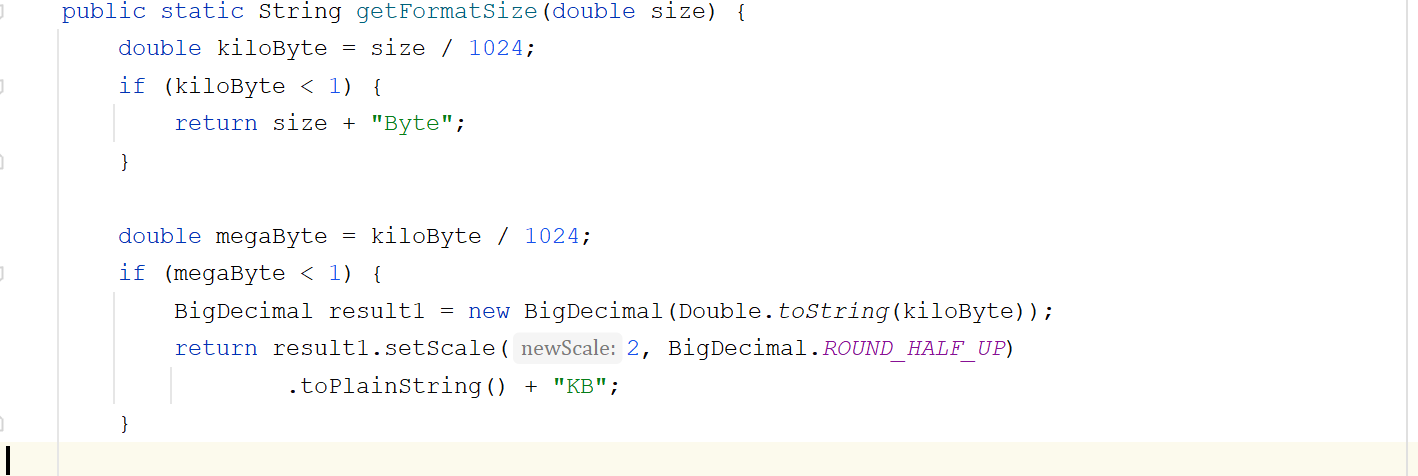
1. 清除本应用内部缓存



1. 获取文件大小



1. 格式化单位



**3.2 usermodel**

**3.2.1 Article**

在数据库中创建Article表，包含的键有：用户账号、标题、作者昵称、编写时间、文章图片地址、文章内容。其设计结构如下

**个人文章表：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 是否为null | 字段描述 |
| userId | varchar | 20 | 否 | 用户账号 |
| articleTitle | Varchar | 50 | 否 | 文章标题 |
| articleAuthor | varchar | 20 | 否 | 用户昵称 |
| articleTime | varchar | 50 | 否 | 发布时间 |
| articleImagePath | varchar | 100 | 是 | 文章图片地址 |
| articleContent | varchar | 100 | 否 | 文章内容 |

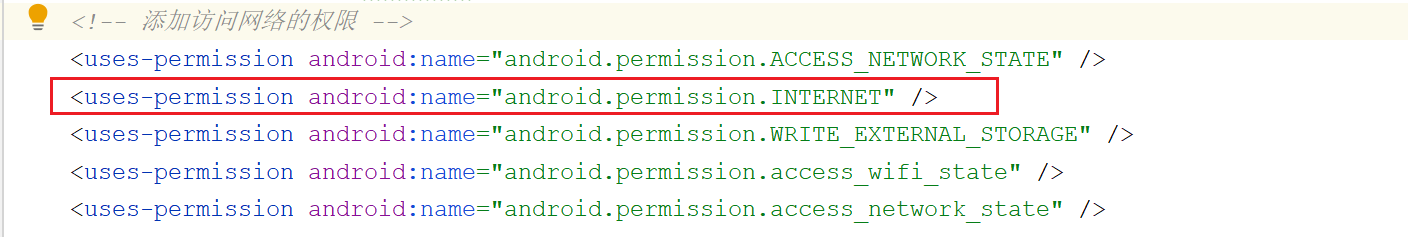
**3.2.2 ArticleAdapter**

ArticleActivity页面的控制器，要想使用 ListView 就需要编写一个 Adapter 将数据适配到 ListView上，此处使用的是RecyclerView组件。为了节省资源提高运行效率，编写ViewHolder类来减少findViewById()的使用以及避免过多地 inflate view，从而实现提示性能的目标。



在onCreateViewHolder方法中，让RecyclerView数据视图项cardView监听点击事件，使其实现点击跳转到文章详情页ArticleDetailActivity的功能。

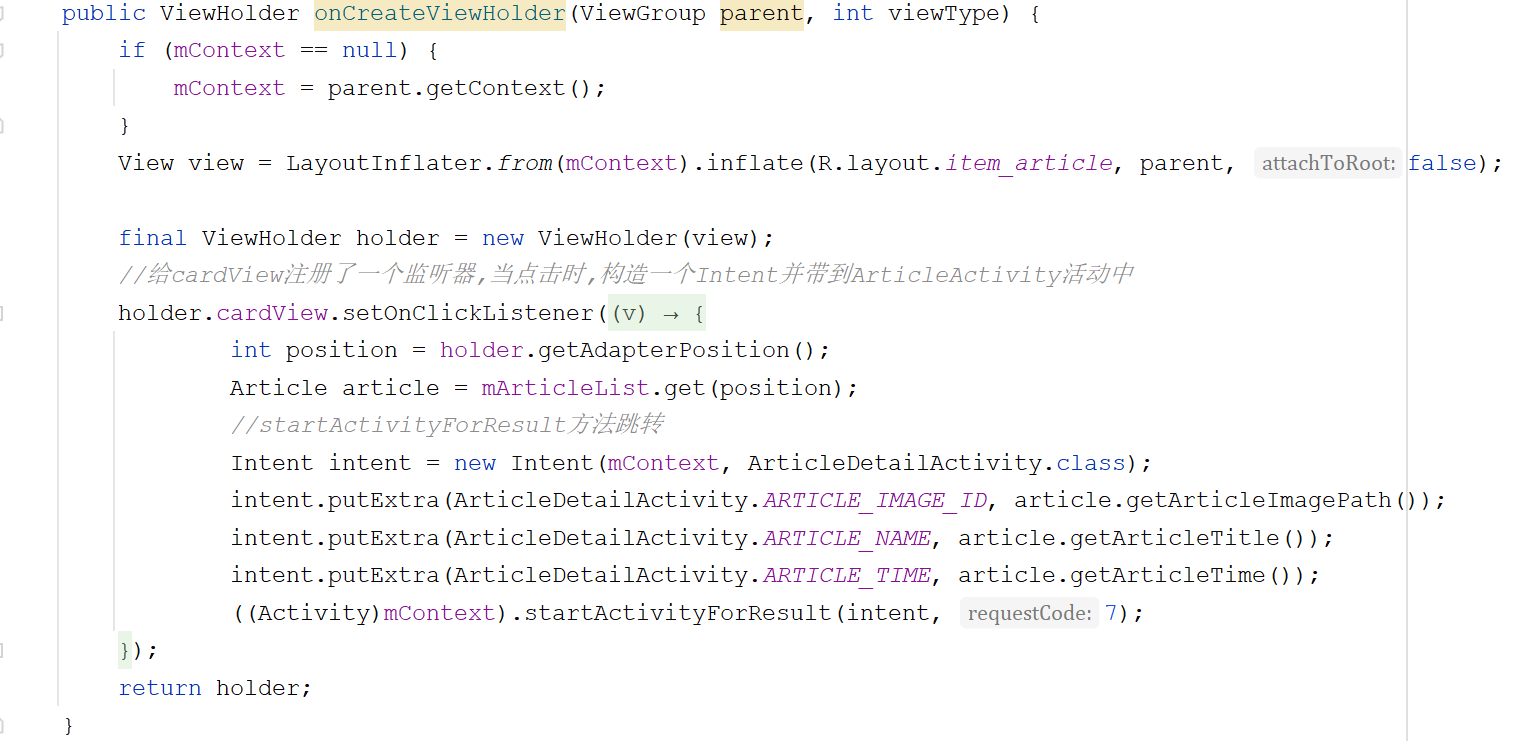
使用Glide来加载图片，需要网络权限，所以在AndroidManifest.xml中声明一下网络权限：<uses-permission android:name="android.permission.INTERNET" />

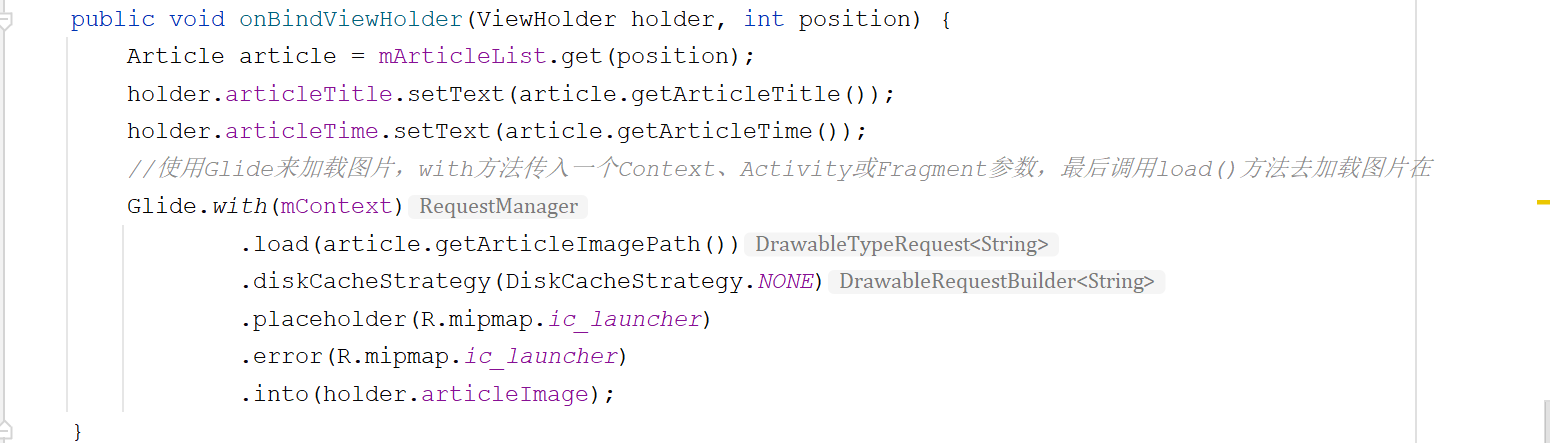


给cardView注册了一个监听器,当点击时,构造一个Intent并带到ArticleDetailActivity活动中，为了实时刷新，采用startActivityForResult的方式来进行跳转，由于是在adpator中跳转，需要把上下文对象转为activity对象进行传递：

Intent mIntent = new Intent(mContext,clazz);

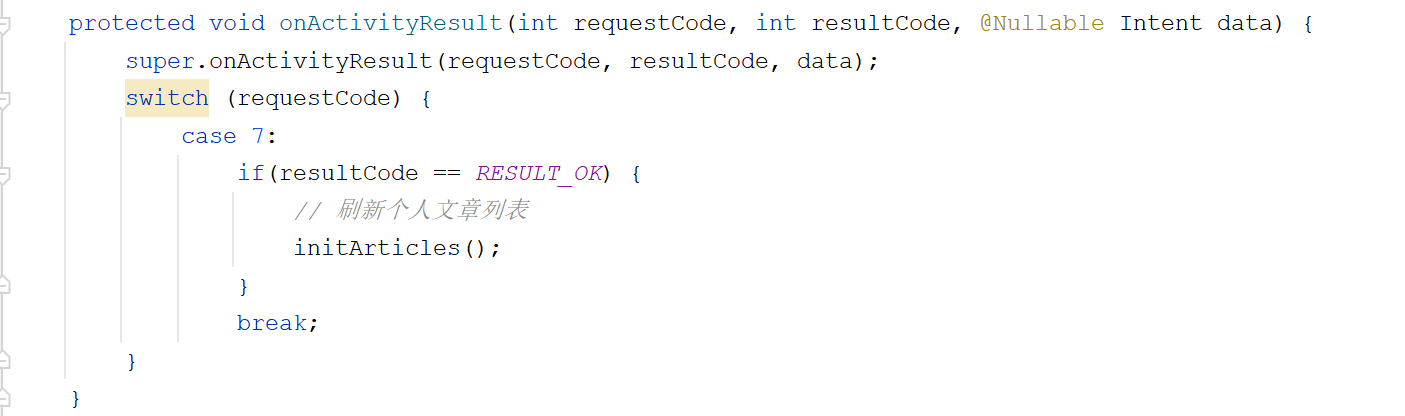
((Activity) mContext).startActivityForResult(mIntent, requestCode);





**3.2.3 ArticleActivity**

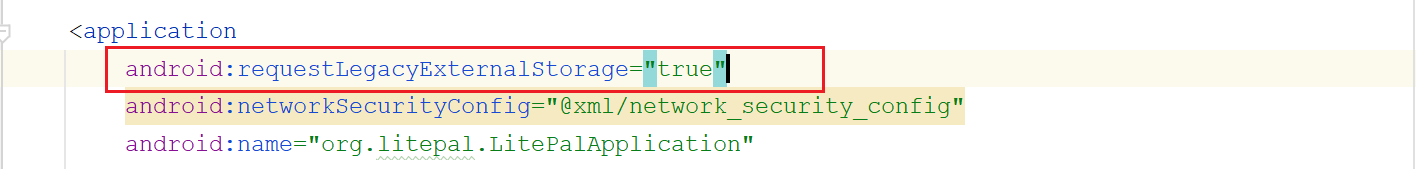
监听悬浮按钮点击事件跳转到EditArticleActivity页面，在编辑完成返回后需要对页面进行刷新。



跳转到EditArticleActivity页面使用startActivityForResult()方法来进行跳转，相较于startActivity()仅仅是跳转到目标页面，若是想跳回当前页面，则必须再使用一次startActivity()而言，startActivityForResult()可以一次性完成这项任务，当程序执行到这段代码的时候，假若从T1Activity跳转到下一个Text2Activity，而当这个 Text2Activity调用了finish()方法以后，程序会自动跳转回T1Activity，并调用前一个T1Activity中的 onActivityResult()方法。

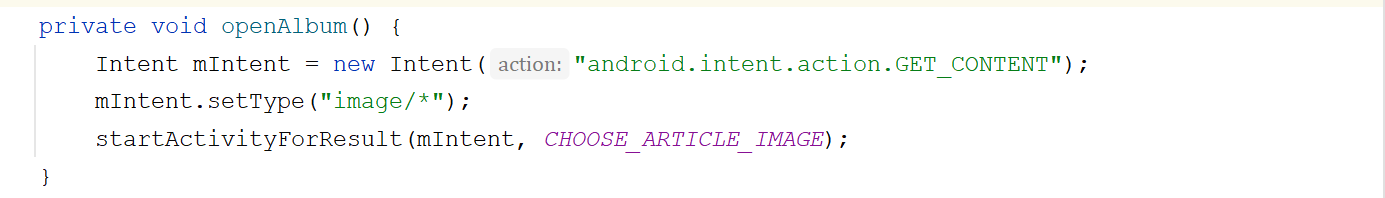
**3..2.4 EditArticleActivity**

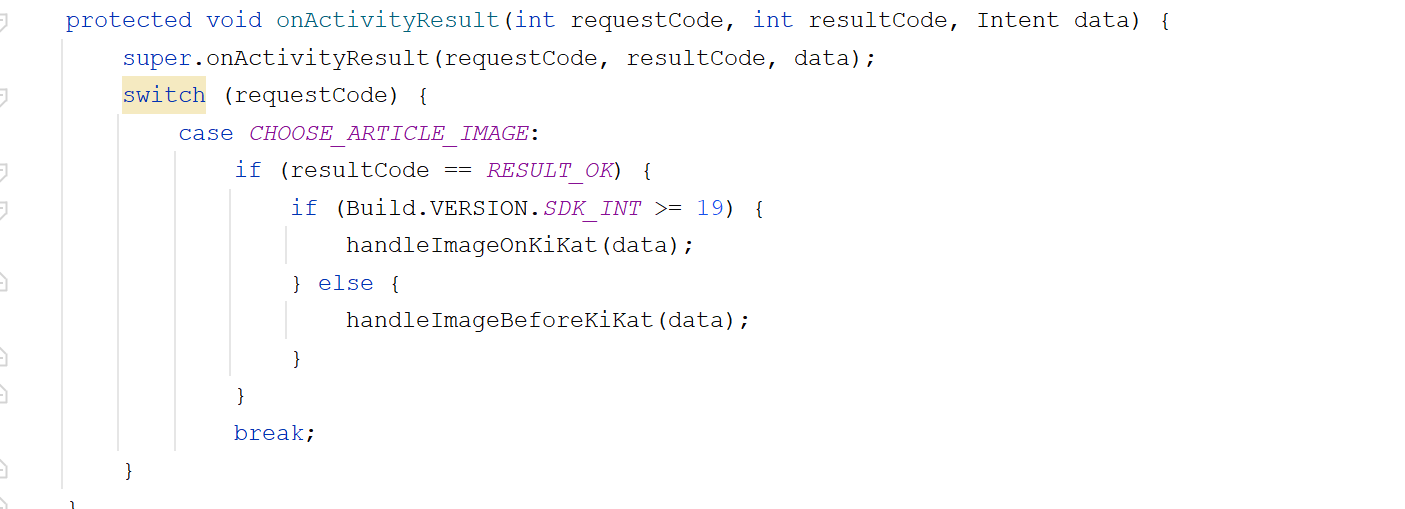
注意：谷歌在对Android 10权限问题上进一步加强，在Android 10上如果不在manifest文件中加上requestLegacyExternalStorage=“true”，则会出现分区存储的问题，导致是压缩图时获取的的bitmap为null，而无法显示图片。

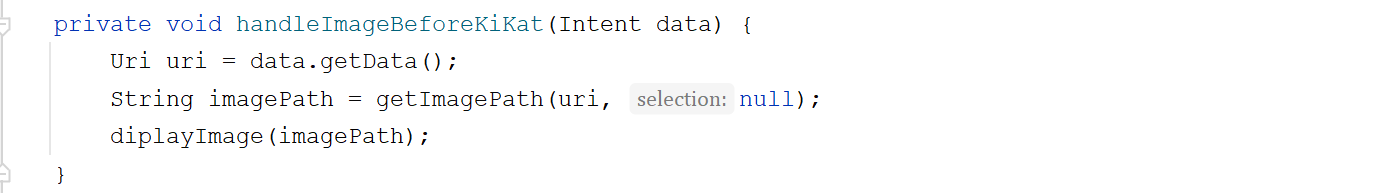


接收intent传递过来的userID来进行数据库查询用户信息，使文章与用户绑定。

用户点击ImageButton之后可以选择图片上传，使用startActivityForResult的方法进行跳转，应用需要获取打开相册的权限，获取到图片路径，再将图片显示出来。

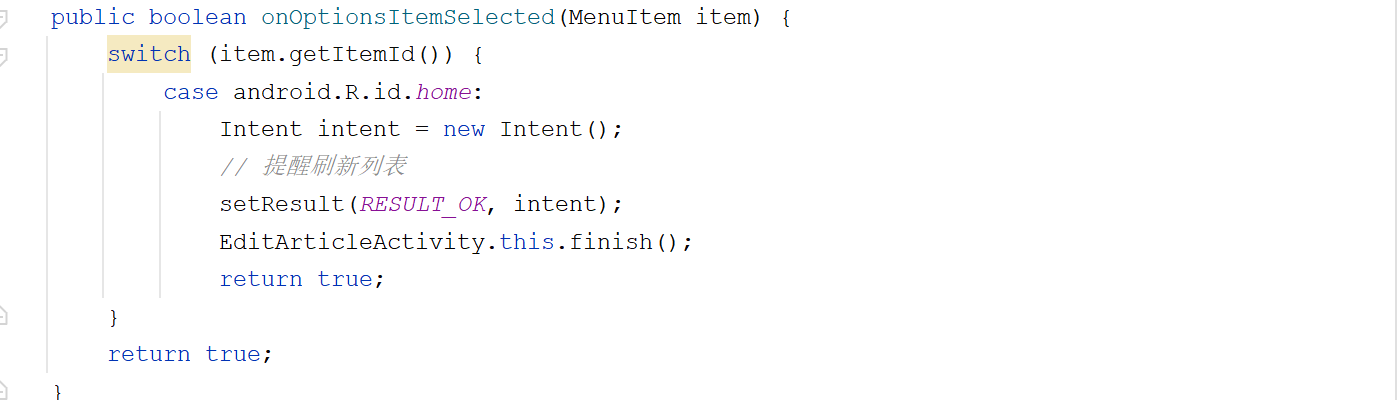








点击发布之后，如果操作成功，则finish当前页面，并回跳到ArticleActivity页面，发起一个刷新提醒。



**3.2.5 ArticleDetailActivity**

点击删除按钮之后回跳回AritcleActivity需要对AritcleActivity进行刷新，由于在ArticleAdapter中定义了startActivityForResult，并将其绑定给了AritcleActivity，在AritcleActivity中实现了onActivityResult函数。因此当删除成功返回之后会触发刷新。



**3.2.6 UserInfo**

用户信息类，包含的键有：账号、昵称、登录密码、性别、生日、个性签名、保存头像的路径。

**用户表**：

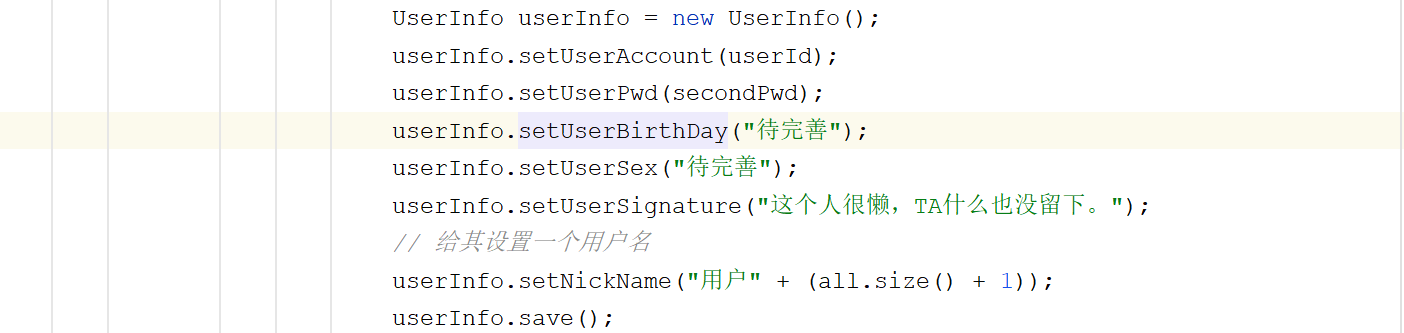
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 是否为null | 字段描述 |
| userAccount | varchar | 20 | 否 | 账号 |
| nickname | varchar | 20 | 是 | 昵称 |
| userPwd | varchar | 20 | 否 | 登录密码 |
| userSex | varchar | 6 | 是 | 性别 |
| userBirthDay | varchar | 20 | 是 | 生日 |
| userSignature | varchar | 50 | 是 | 个性签名 |
| imagePath | varchar | 50 | 是 | 保存头像的路径 |

**3.2.7 UserDetailActivity**

能够实现编辑、修改个人信息（头像、昵称、性别、生日、个性签名）的功能，给这些属性对应得线性布局分别绑定点击事件，使点击的时候弹出填写框或选择按钮等来编辑个人信息。同样的，当前页面返回也需要刷新上一页面的信息。

**3.2.8 RegisterActivity**

对用户输入进行检查，倘若用户输入为空，则不允许注册，两次输入的密码应该保持一致，上述条件满足后，对账号进行数据库查重，如若账号已被注册，则提示已被注册，注册成功后将用户信息进行初始化存储。



注册完成之后跳转到登录界面需要保存注册信息以便直接登录，参数通过intent的putExtra()方法传递。登录界面直接使用getExtra()方法获取填入即可。

**3.2.9 LoginActivity**

检测用户输入内容是非为空，不为空时查询数据库验证登录信息。



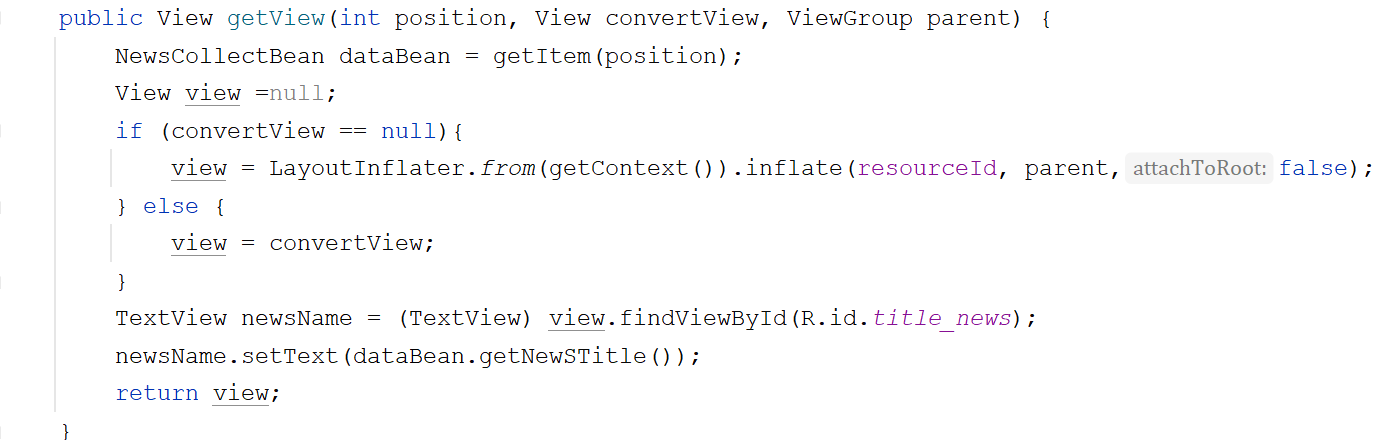


为了方便下次用户直接登录，登录界面销毁时对登陆的账号进行本地存储，下次启动app时可以直接读取账号以初始化数据。



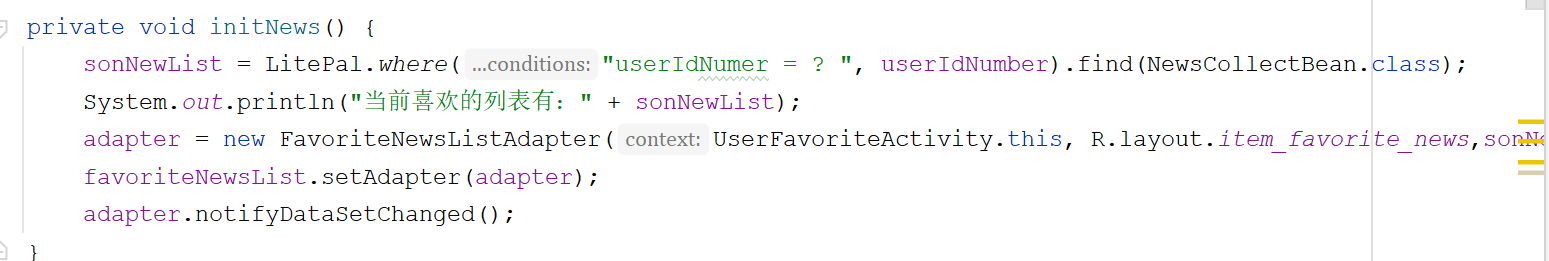
**3.2.10 FavoriteNewsListAdapter**

个人收藏文章界面的适配器。



**3.2.11 UserFavoriteActivity**

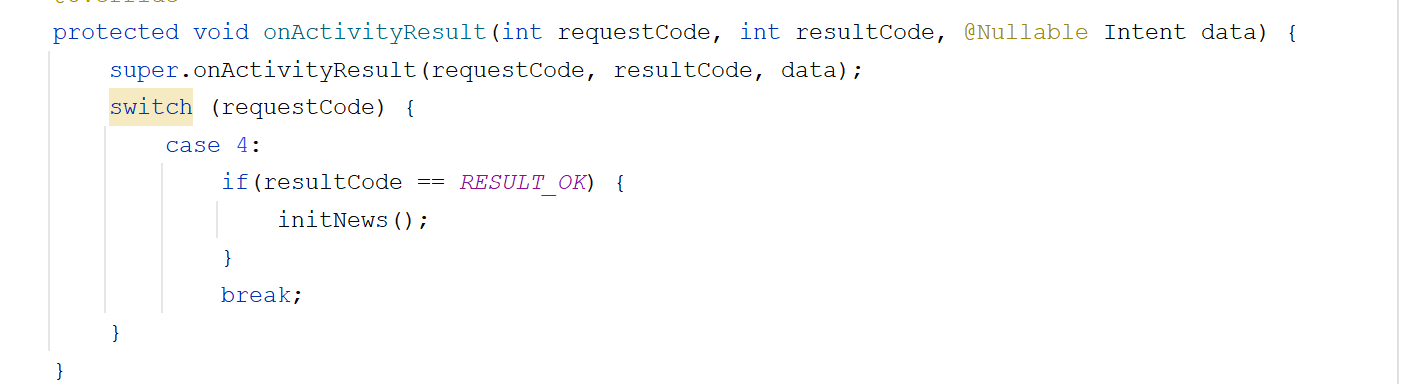
文章收藏夹，当页面调用create方法的时候进行初始化，调用initNews()，将数据库查询到的文章链接使用适配器展示到列表里面。



在onStart函数中初始化事件——文章绑定点击事件，点击之后能够阅读该文章。



考虑到用户阅读之后，可能会取消收藏，所以当再次返回到该页时，需要初始化页面。





**3.3 主模块**

**3.3.1 NewsBean**

新闻响应属性包含字段：数据响应理由、结果集、响应状态、新闻id、新闻标题、请求日期、新闻标签、新闻作者、新闻链接、附图1的地址、附图2的地址、附图3的地址。

**新闻表**：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 是否为null | 字段描述 |
| reason | varchar | 20 | 否 | 数据响应理由 |
| result | varchar | 20 | 否 | 结果集 |
| stat | int | 4 | 否 | 响应状态 |
| uniquekey | varchar | 100 | 否 | 新闻id |
| title | varchar | 100 | 否 | 新闻标题 |
| date | varchar | 50 | 否 | 请求日期 |
| category | varchar | 10 | 否 | 新闻标签 |
| author\_name | varchar | 30 | 否 | 新闻作者 |
| url | varchar | 200 | 否 | 新闻链接 |
| thumbnail\_pic\_s | varchar | 200 | 否 | 附图1的地址 |
| thumbnail\_pic\_s02 | varchar | 200 | 否 | 附图2的地址 |
| thumbnail\_pic\_s03 | varchar | 200 | 否 | 附图3的地址 |

**3.3.2 NewsCollectBean**

收藏的新闻包含字段：用户账号、新闻id、新闻标题、新闻链接。

**新闻收藏表：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 是否为null | 字段描述 |
| userIdNumer | varchar | 20 | 否 | 用户账号 |
| newsId | varchar | 100 | 否 | 新闻id |
| newSTitle | varchar | 100 | 否 | 新闻标题 |
| newsUrl | varchar | 200 | 否 | 新闻链接 |

**3.3.3 NewsInfoBean**

由于接口访问次数有限，此处需要将我们从接口访问接收到的数据进行缓存，当接口次数使用完的时候从本地读取新闻即可。即将展示给用户的本地新闻应包含以下信息：新闻标题、请求日期、新闻标签、新闻作者、新闻链接、附图1的地址、附图2的地址、附图3的地址。有关字段已经在NewsBean中说明。

**3.3.4 NewsFragment**

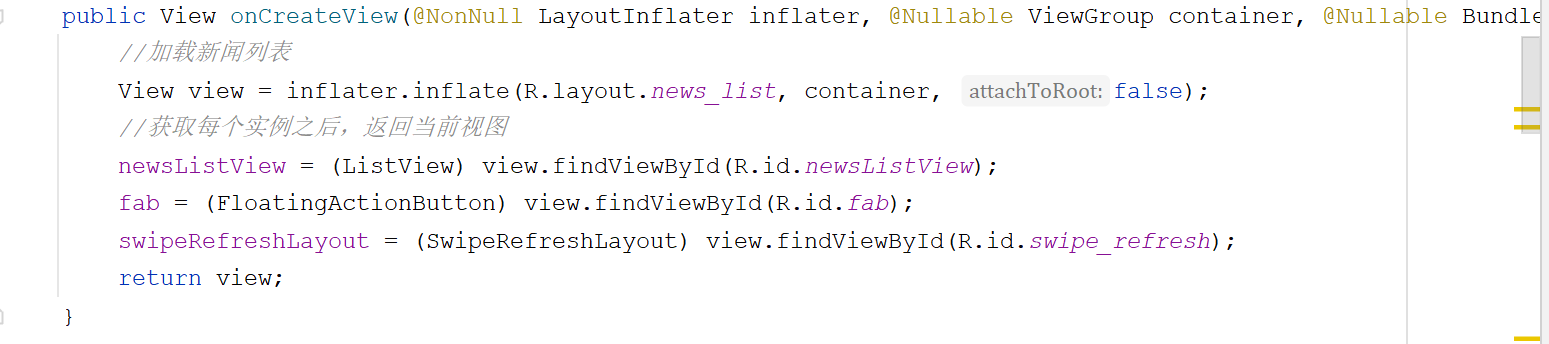
定义了每个tab下的碎片Fragment。由于主线程无法进行时间比较繁长的任务，所以需要子线程进行处理，然而子线程无法进行UI的界面更新，所以我们需要使用handler来传递消息给主线程，让其完成UI的更新。

使用Handler在Android来进行消息的发送与异步处理，当从服务器获取到新闻数据就构造一个TabAdpator对象来填充新闻列表，将新闻列表进行更新显示，同时保存到本地客户端。



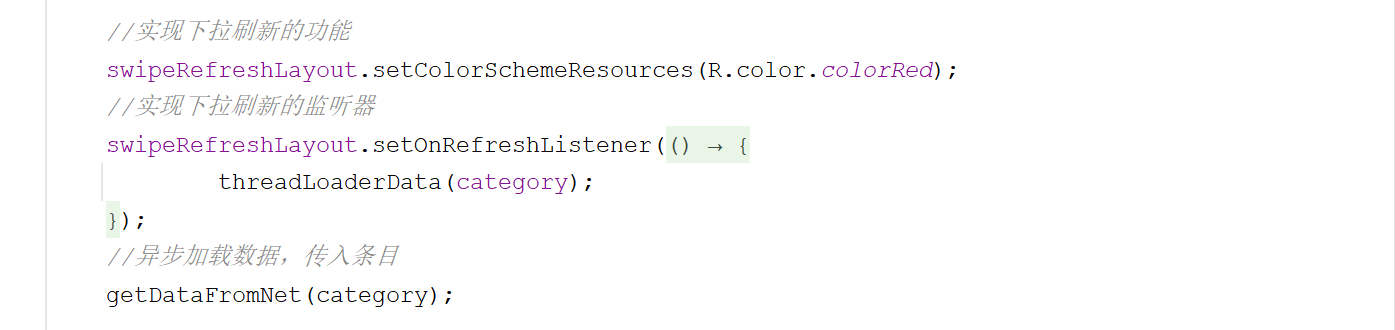
每次创建时调用onCreate方法，绘制Fragment的View组件，通过bundle来获取当前Tab是哪一个，异步处理请求新闻，将显示的View返回从而展示出当前页面。给条目添加点击事件，点击后跳转到新闻详情内容页。

Fragment的生命周期函数onCreateView，绘制该Fragment的View组件时回调，将显示的View返回系统创建Fragment的时候调用此函数，一般在此函数内实例化一些变量。



Fragment的生命周期函数onActivityCreated，在里面定义一些与Activity相关的UI交互操作。







使用线程的方式来加载新闻数据，主线程非常繁忙，因为它要处理绘制UI，响应用户的交互，默认情况下执行我们写下的大部分代码。因此就需要将重负荷的任务移除到工作线程避免主线程阻塞，同时获得更流畅的用户体验，避免ANR的发生。但是，当需要更新UI的时候我们需要“返回”到主线程，因为只有它才可以更新应用 UI。所以此处我们通过调用 Activity 的 runOnUiThread() 方法来切换到主线程。



加载数据，实现从本地数据库中读取数据刷新到newsListView的适配器中。采用分页的方法来循环读取，此处设置为一页20条数据。



对于耗时的操作，我们的一般方法是开启“子线程”。如果需要更新UI，则需要使用handler。但是如果耗时的操作太多，那么我们需要开启太多的子线程，这就会给系统带来巨大的负担，随之也会带来性能方面的问题。在这种情况下我们就可以考虑使用类AsyncTask来异步执行任务，不需要子线程和handler，就可以完成异步操作和刷新UI。

在线的新闻接口采用的是聚合新闻接口api，由于接口每日免费的次数有限，所以采用了缓存到本地的思路，以便免费次数用完之后从本地读取。

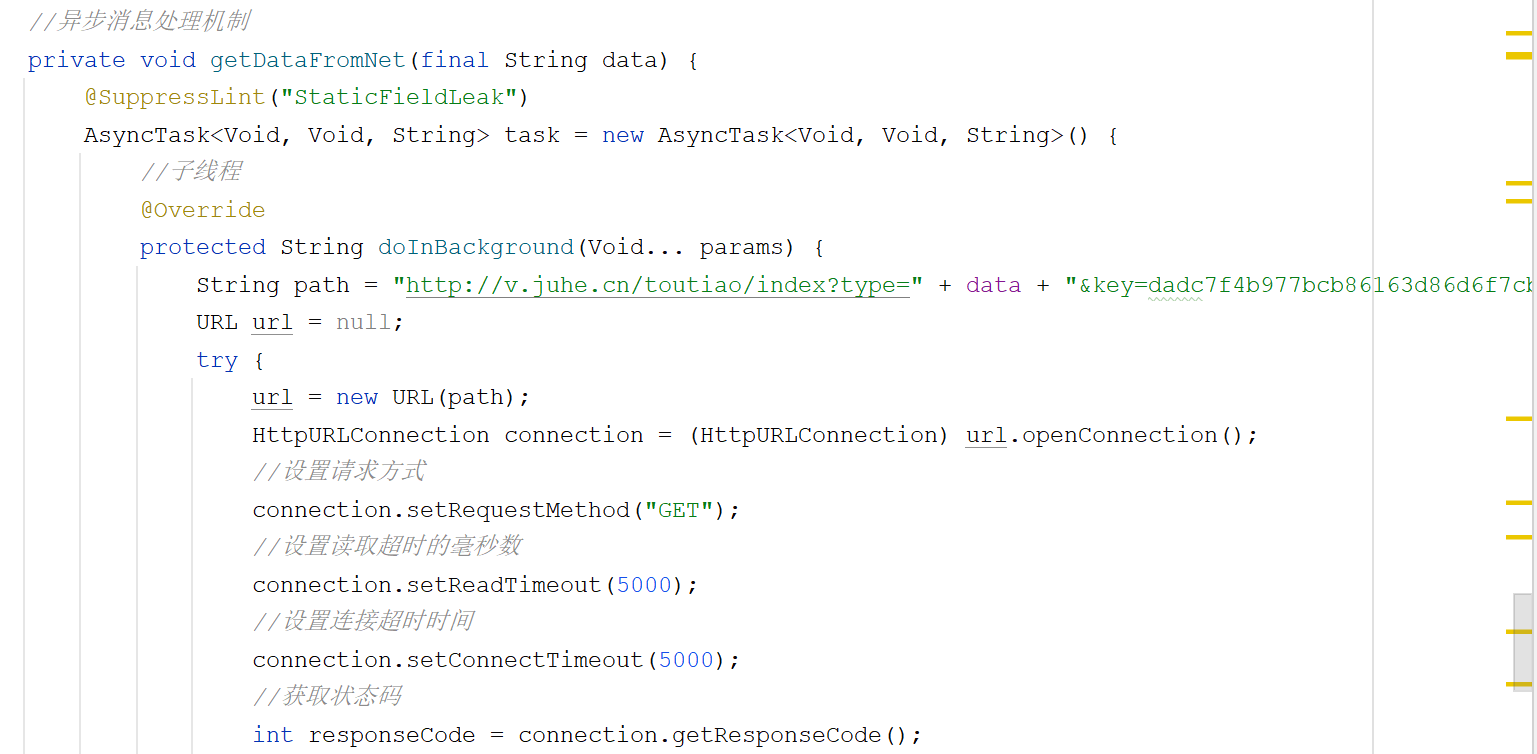


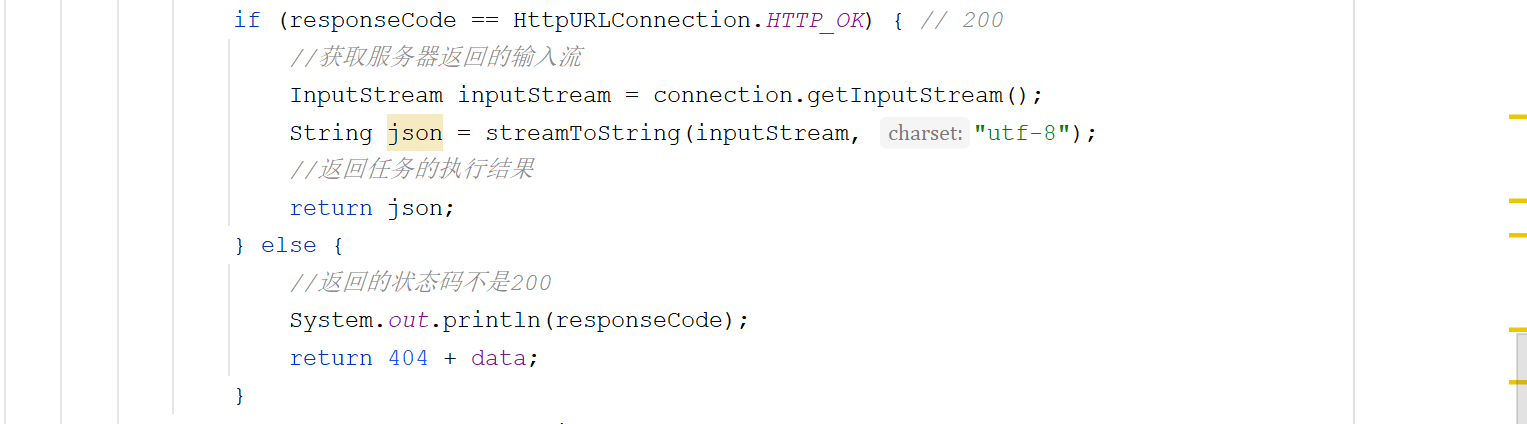




从网络接口获取新闻数据，便是采用AsyncTask的方式来进行异步消息处理机制。AsyncTask的定义：public abstract class AsyncTask<Params, Progress, Result>，三种泛型类型分别代表“启动任务执行的输入参数”、“后台任务执行的进度”、“后台计算结果的类型”。在特定场合下，并不是所有类型都被使用，如果没有被使用，可以用java.lang.Void类型代替。此处我们只需要Result，为String类型的json数据。

oInBackground(Params... params)，用于执行较为费时的操作，此方法将接收输入参数和返回计算结果，在其中我们用来访问新闻数据流，再将数据流转为字符串。





onPostExecute函数，当给后台任务执行完毕并通过return语句返回时，此方法将被调用，返回来的数据可以进行一些UI操作，并将处理的参数传入。查看返回的状态码是否为200，若不是则开子线程，然后从本地加载相应的数据。

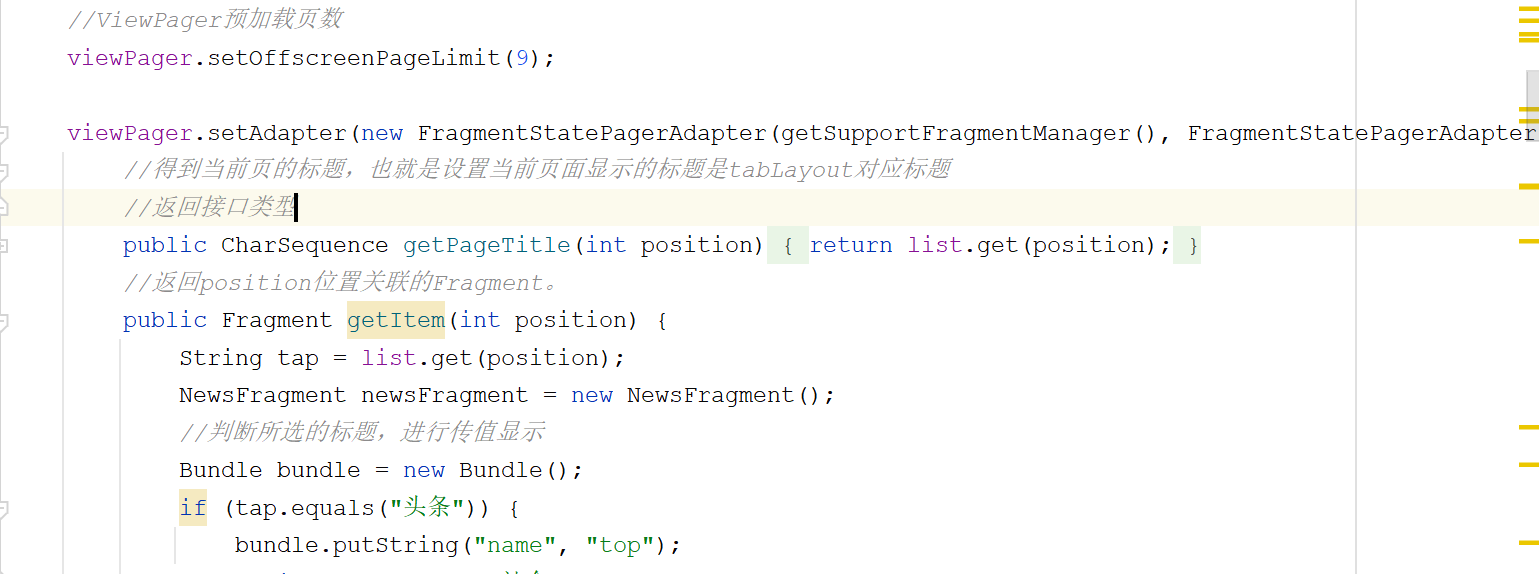


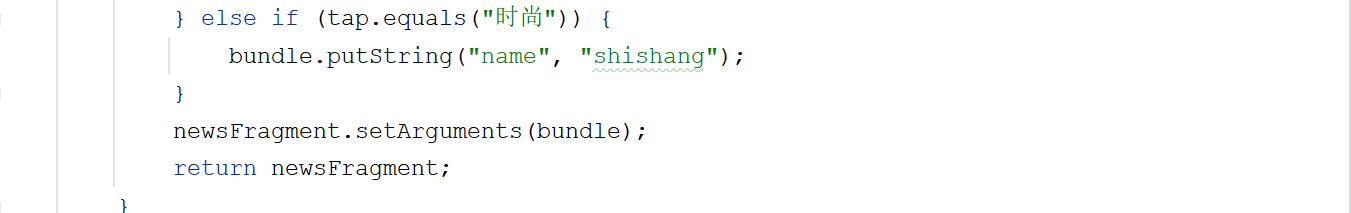
当fragment调用onCreate函数时要通过参数savedInstanceState恢复之前保存的状态。

**3.3.5 MainActivity**

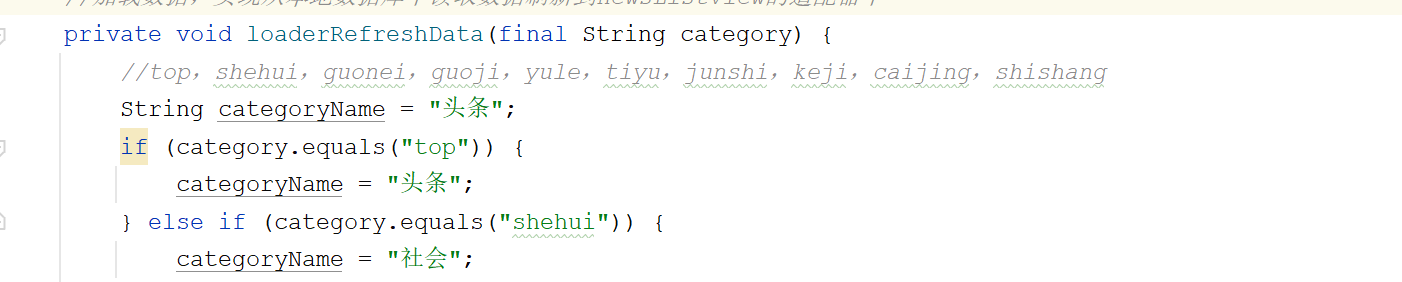
ViewPager 是android support中的类，直接继承自ViewGroup类，它是一个容器类，可以在其中添加其他的View，需要一个PagerAdapter适配器，来为他提供数据。经常和Fragment一起使用，并提供了专门的FragmentPagerAdapter和FragmentStatePagerAdapter类。

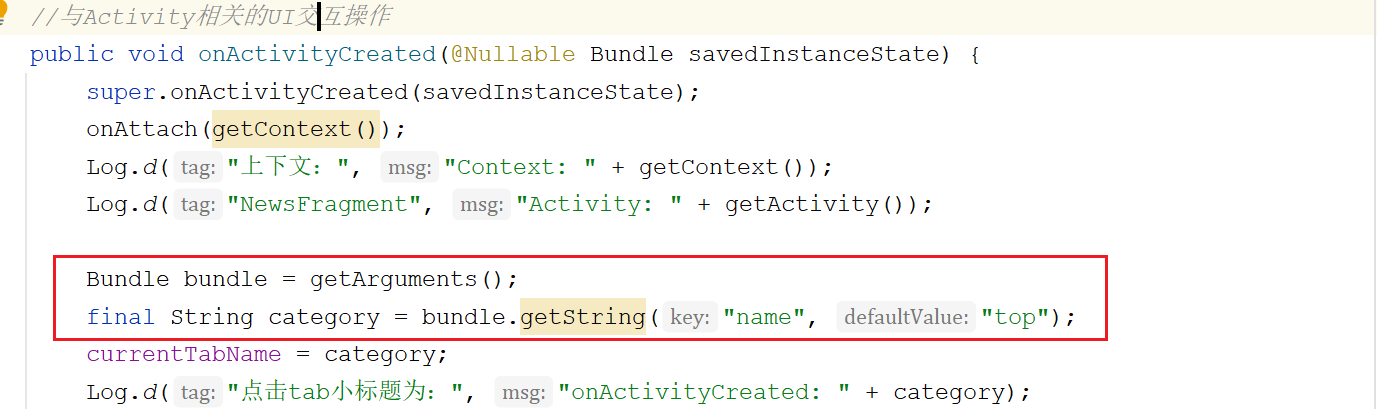
ViewPager预加载页数viewPager.setOffscreenPageLimit(9);ViewPage填充了fragment后，当fragment不可见时, 可能会将fragment的实例也销毁(执行 onDestory, 是否执行与setOffscreenPageLimit 方法设置的值有关)，所以内存开销会小些, 适合多fragment的情形。



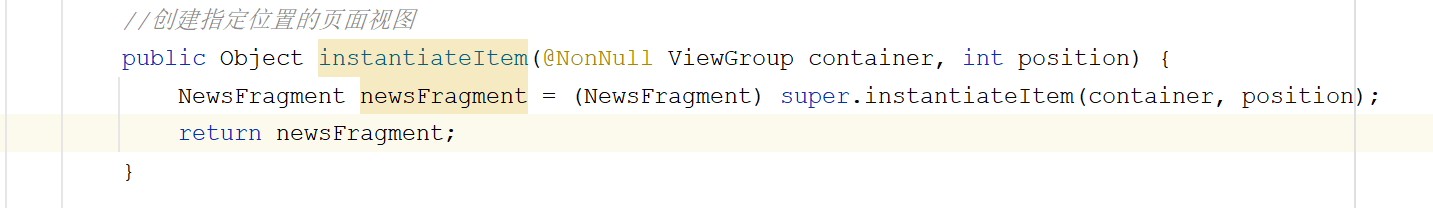


根据传入的不同的bundle值，改变查询新闻数据的参数categoryName，从而获取到指定的Fragment页面的数据。

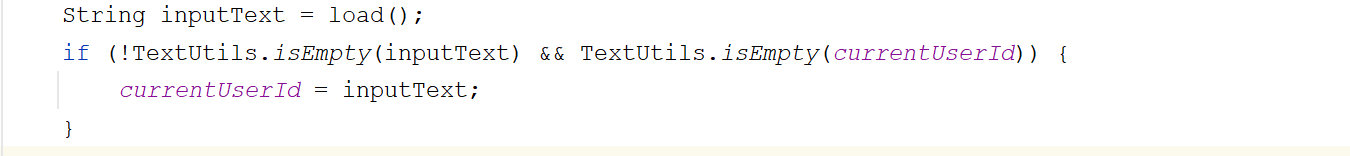




instantiateItem函数在每次 ViewPager 需要一个用以显示的 Object 的时候，都会被 ViewPager.addNewItem() 调用。



加载上次登录的账号，起到记住会话的功能

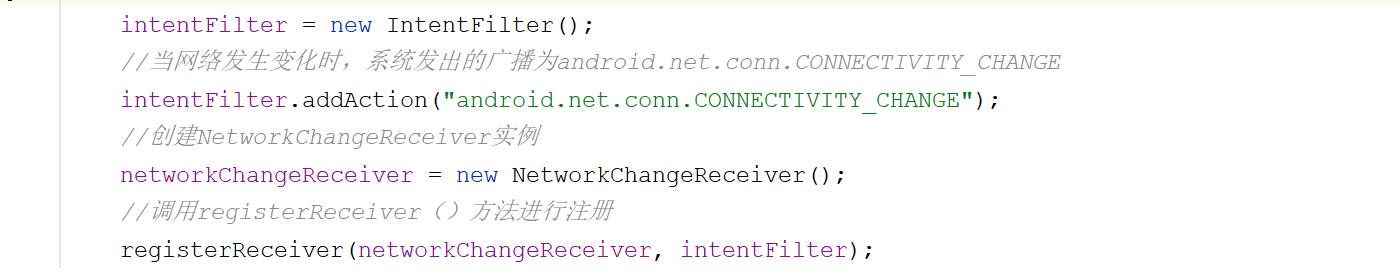




检测当前登录用户，并对抽屉信息进行初始化，同时对未登录状态的设置了默认初始化信息。

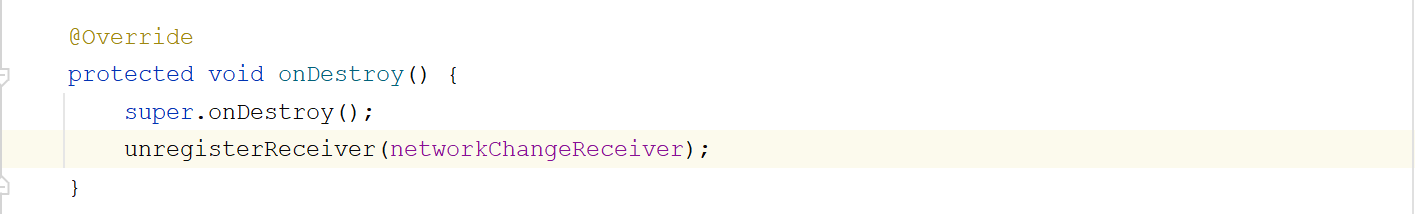


在onCreate方法中给app注册网络变化的系统广播，NetworkChangeReceiver类继承自BroadcastReceiver，用来接收广播。用来提示用户网络的变化，当用户处于未联网状态时，提示用户此时仅本地缓存可读，当处于联网状态时，提示用户注意网络流量的使用。

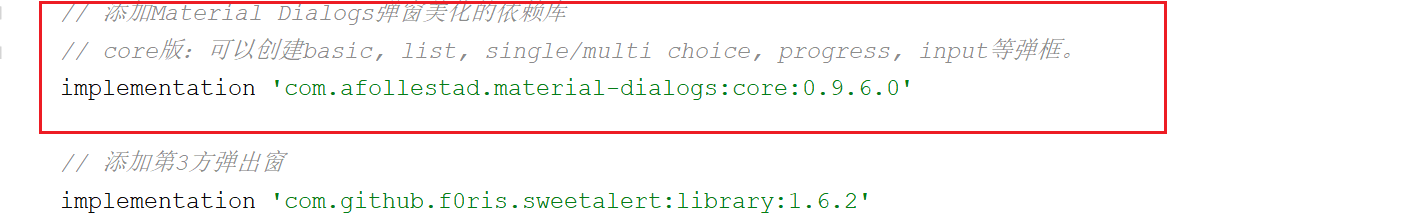




活动结束时，调用unregisterReceiver方法实现取消注册。

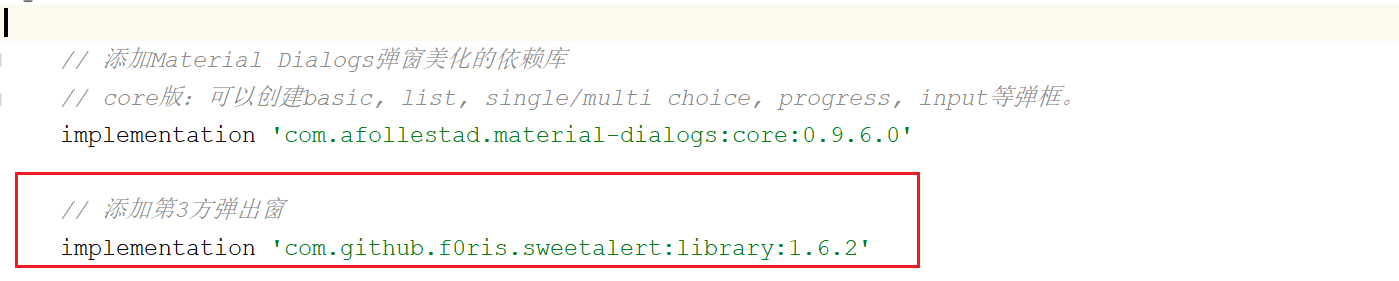


清除缓存功能的实现。采用第三方对话框组件MaterialDialog来进行提示以及监听用户选择。依赖模块：





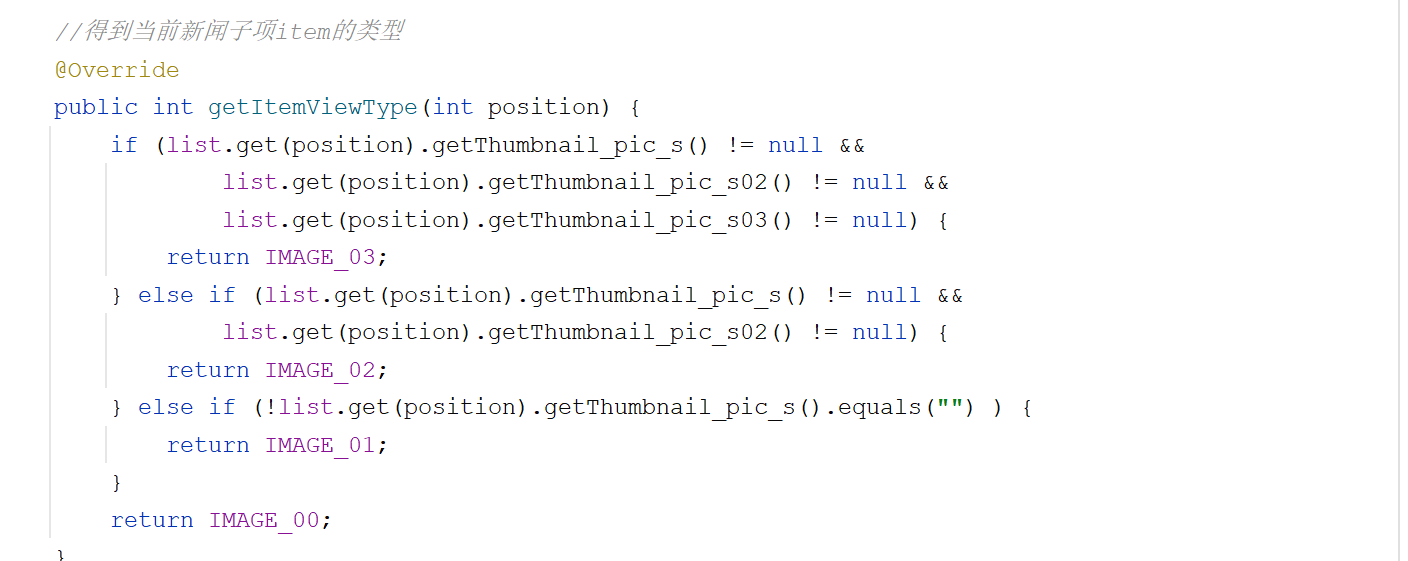
退出登录，采用第三方弹窗控件SweetAlert，依赖模块



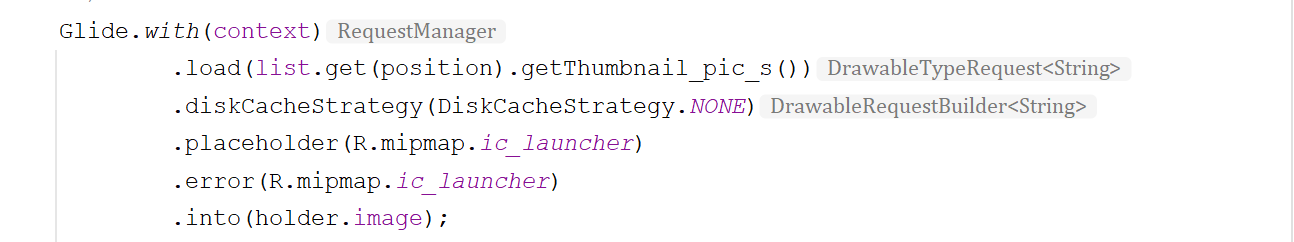


**3.3.6 TabAdapter**

自定义新闻列表的适配器，根据文章含有的图片数量，填充为图片数量为0、1、2、3的页面。





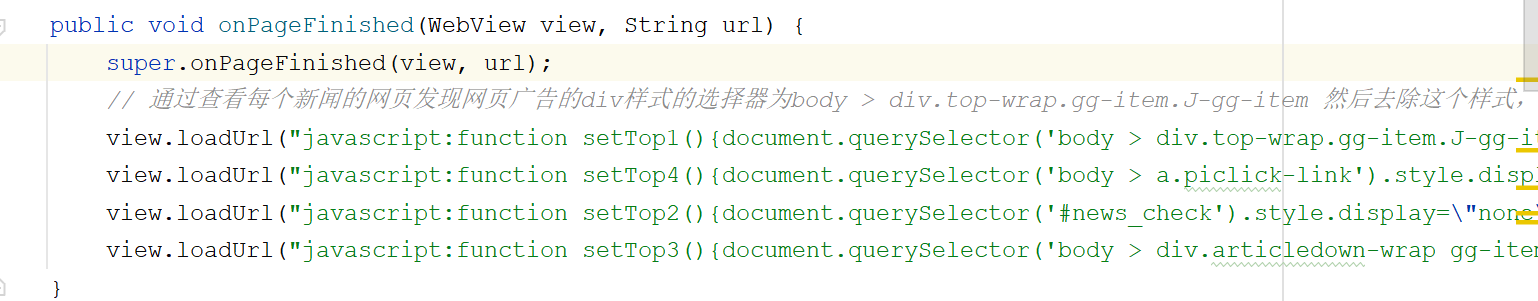


**3.3.7 WebActivity**

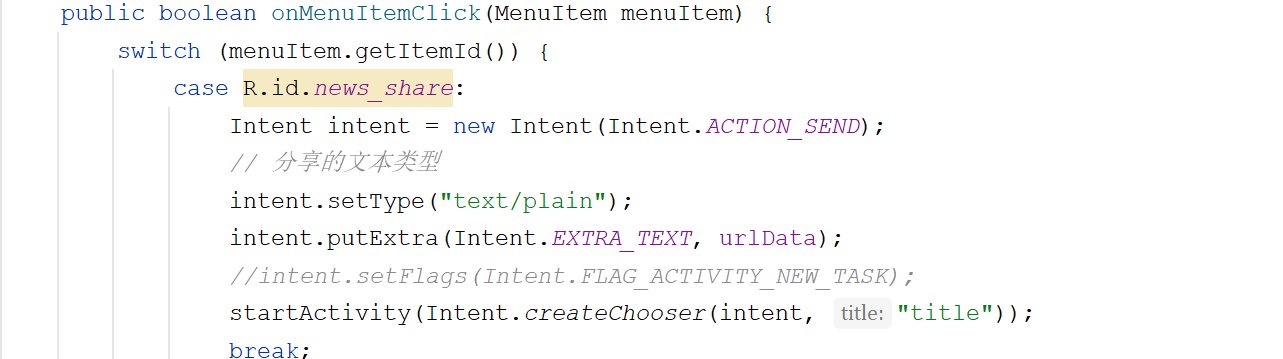
点击文章后跳到此处，用于访问文章链接。加载文章后，查询数据库，看是否已经收藏该文章，如果文章已被收藏，则收藏图标被渲染成红色。



出除网页里的广告



分享本文链接



收藏文章功能



**3.4 数据库模块**

**3.4.1 数据库配置**

1. 要在app/build.gradle文件中声明该开源库的引用：

implementation 'org.litepal.android:java:3.0.0'

2、配置litepal.xml文件



使用litepal数据库框架，在此处声明数据库名、版本号、指定所有的映射模型列表,声明要配置的映射模型类，注意一定要使用完整的类名。

3、修改AndroidManifest.xml文件中的内容：

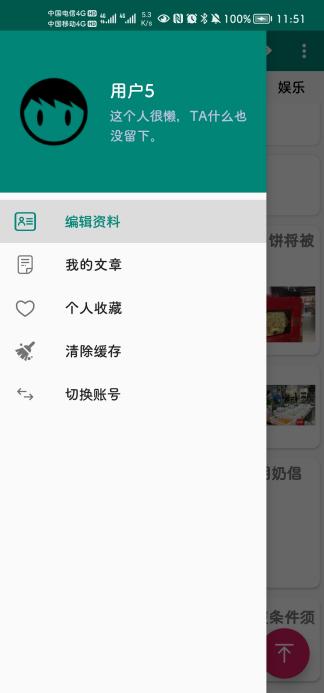
android:name="org.litepal.LitePalApplication"

1. **完成效果展示**

打开之后进入app，点击弹出侧边抽屉

** **

点击任意选项提示先登录，由于没有账号，我们先注册一个，注册完之后直接把数据带到登录界面，直接点击登录即可。

**  **

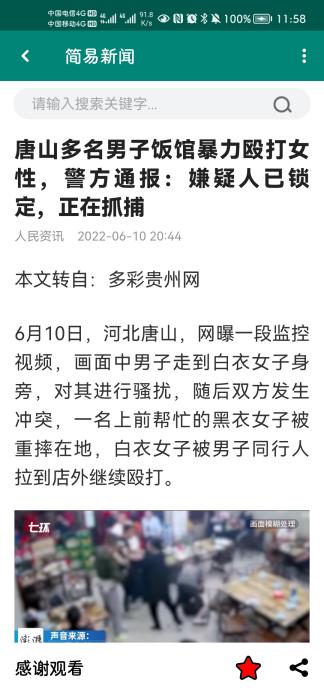
更新用户信息

我的文章功能，编辑、查看我的文章

个人收藏功能，收藏的文章会被在收藏夹显示

分享文章功能，分享的文章将会以连接的形式被发送出去



监听网络状态功能

清除缓存功能



退出登录功能



1. **总结**

在整个项目的实现过程中，遇到很多棘手的bug，最终在参考网上的项目后逐一实现了功能，虽然该项目功能略显简单，但是胜在基本功能都齐全，基本可以用于个人看新闻所用，项目中使用到的技术有activity技术、Broadcast Reciver技术、第三方控件技术unregisterReceiver和MaterialDialog、异步消息处理机制、litepal数据库等，页面处理用户的事件后更新页面及时。碍于找到的数据接口每日次数限制，不得不将已浏览的新闻数据存储在本地，这样一来势必会导致app占用的存储空间越来越大，当当日次数使用完毕之后，清除缓存会导致页面上没有数据显示。总而言之，通过本项目，我了解并亲身经历了一个项目从设计到实现的过程，对安卓开发有了更加深层次的理解。