2022年夏季Java小学期大作业-实验报告

杜一阳 计11 2021011778 duyy21@mails.thu.edu.cn

Table of Contents

2022年夏季Java小学期大作业-实验报告
- 代码结构
  - 总览
  - 用户界面（UI）
    - 窗口类：在单一Activity下的多个Fragment实现
    - 适配器类：使用RecyclerView的核心
    - 文章类
  - 网络与API接口
  - 数据库与本地存储
    - ROOM数据库
    - 图片本地存储
  - 设置
- 具体实现
  - 用户界面（UI）
    - MainActivity
    - FragmentWithArticles
    - HomeFragment
      - 可滑动、可增删的分类标签
      - 同时使用两种RecyclerView.Adapter
      - 上拉加载，下拉刷新
    - RssFragment
    - SlideshowFragment
    - ArticleFragment
  - 网络与API接口
  - 数据库与本地存储
    - Article类
    - ArticleDao接口
    - ArticleDatabase & Converters
    - FileController
  - 设置
    - 清除本地存储

设置Rss源
其他难点与亮点
总结与心得

代码结构

总览

本项目通过Android Studio开发，使用了其"Navigation Drawer Activity"模板生成了左侧导航栏，其余部分均为自主开发。Java代码在名为com.java.duyiyang的包下，主要分类有：

- 用户界面（UI）
  - `MainActivity`
  - `FragmentWithArticles` 及其子类 `HomeFragment`, `GalleryFragment`, `ListFragment`, `FavoriteFragment`, `RssFragment`
  - `SlideshowFragment`
  - `CustomAdapter`, `TabAdapter`
  - `ArticleFragment`
- 网络与API接口
  - `ApiAdapter`
- 数据库与本地存储
  - `Article`, `ArticleDao`, `ArticleDatabase`, `Converters`
  - `FileController`
- 设置
  - `SettingsFragment`

这一部分介绍各个类的设计思路、功能及代码结构。

用户界面（UI）

窗口类：在单一Activity下的多个Fragment实现

App采用Android Jetpack's Navigation来管理各窗口。在实现多窗口切换时，采用多个Fragment是比多个Activity更优的做法，因此我们只创建`MainActivity`这一个Activity类，在其中实现多个不同的窗口Fragment。

由于新闻App有许多界面需要展示文章列表及相应功能，于是我将这部分功能抽象成一个`FragmentWithArticles`抽象类，大大提高了代码的复用率。主页、搜索结果、历史记录、收藏、Rss源这五个界面分别通过`HomeFragment`，`GalleryFragment`，`ListFragment`，`FavoriteFragment`，`RssFragment`这五个继承`FragmentWithArticles`的类来实现。

此外，搜索界面在`SlideshowFragment`类中实现。

适配器类：使用RecyclerView的核心

在展示文章列表时，由于列表内元素个数可能很大，如果将他们都加载出来将会十分浪费资源，而RecyclerView可以通过展示容器的复用，以屏幕容纳数量的资源使用（相对小）来加载整个列表内容。使用RecyclerView需要定义其适配器（RecyclerView.Adapter）用于复用加载内容，故`FragmentWithArticles`及其子类需要与Adapter配合完成显示。

因此，本App中实现了两种适配器类：`CustomAdapter`和`TabAdapter`，前者是为了满足显示文章列表的需要，后者是为了满足分类列表"添加和删除操作"这一需求。

文章类

`ArticleFragment`用于显示文章的标题、发布者、发布时间、内容（文字图片视频），并且负责处理收藏行为，并将已阅读的新闻（及其图片与收藏状态）存储到本地。

网络与API接口

`ApiAdapter`类实现了这部分的全部功能：设置参数并向课程提供的Api请求新闻数据，异步地获取新闻数据，并在完成后通过`Method.invode`进行回调。同时实现了必要的构造函数等。

数据库与本地存储

ROOM数据库

> 处理大量结构化数据的应用可极大地受益于在本地保留这些数据。最常见的使用场景是缓存相关的数据，这样一来，当设备无法访问网络时，用户仍然可以在离线状态下浏览该内容。
> ——Android Developers

本App通过ROOM数据库来存储文章（除图片本身）的全部信息。ROOM数据库是Android Jetpack在SQLite上提供的一个抽象层。为了使用它，我们实现了如下类/接口：

- `Article`类定义了文章数据的内容：新闻ID、标题、发布时间、关键词、图像地址、视频地址、内容、发布者、分类和收藏状态。同时其构造函数还实现了直接传入`JSONObject`的支持，即`Article`类本身负责由Json格式到数据库格式的转换。
- `ArticleDao`为App内其他组件调用数据库定义了接口，提供了插入、弱插入（不覆盖）、删除、全部删除、根据新闻ID获取文章、获取全部文章等方法。
- `ArticleDatabase`采用单例模式，通过调用其静态`getDatabase()`函数返回唯一的room数据库实例，用与MainActivity初始化中的数据库构建环节。
- `Converters`类负责将数据库不能直接存储的ArrayList类型转化成为数据库可直接存储的String类型。

图片本地存储

由于在数据库中直接存储图片数据并不是一个好的选择（会导致数据库大而慢），我们将图片存储于本地的Internal Storage中。又因为App需要实现离线加载图片的功能，我们有这样的策略：当需要加载图片时，先尝试加载本地的图片，若失败再尝试于网络获取。于是，我们将这些策略以静态方法的形式封装到了`FileController`类当中。

`FileController`类主要负责三项异步任务：图片的加载，图片的保存和图片的删除。

设置

通过App右上角的按钮可以进入设置界面，允许用户设置如下内容：

- 删除图片、删除历史记录（及图片）
- 设置RSS源
- （显示）关于

具体实现

这一部分以前文中代码结构为序，介绍具体实现方法及难点、亮点。

用户界面（UI）

MainActivity

这部分代码主要由模板自动生成，这里只介绍重点：

```
private ArticleDatabase mDataBase;
public ArticleDao mArticleDao;

protected void onCreate(...) {
    mDataBase = ArticleDatabase.getDatabase(getApplicationContext());
    mArticleDao = mDataBase.articleDao();
}
```

在Activity创建时，初始化数据库。

FragmentWithArticles

这里先介绍其子类的一般布局设计：

```
<LinearLayout>
    ...
    <com.scwang.smart.refresh.layout.SmartRefreshLayout>
        ...
        <androidx.recyclerview.widget.RecyclerView/>
    </com.scwang.smart.refresh.layout.SmartRefreshLayout>
</LinearLayout>
```

子类一般基于LinearLayout，内含一个`SmartRefreshLayout`用于实现下拉刷新/上拉加载更多，包含了一个显示文章列表的RecyclerView。

这个抽象类的主要成员变量有：

- 用于和api交互的ApiAdapter
- 用于操作图形组件的RecyclerView、CustomAdapter、RefreshLayout
- 数据库接口ArticleDao，已加载的文章ArrayList

主要实现了如下方法：

- `parseJson(String)`将String类型的Json格式通过JSONObject获取，并调用Article的构造函数，将文章信息储存在ArrayList中。并且在处理结束后对Fragment的一些属性做修改，如：重新允许改变标签，重新允许再次刷新/加载更多等。
- `parseXML(InputStream)`为了从Rss源获取信息，我们需要处理XML格式的文件。这里的逻辑是：先将XML格式读入为RssArticle类，再调用Article类的构造函数直接将其转化为Article，这样在后续的处理过程中就没有差别了。

此外还声明了`afterParseJson()`等abstract方法，用于适配不同子类在加载完成后的不同功能需求。

亮点：这里最大的亮点就是抽象出FragmentWithArticles这个基类，将共有的初始化、“RefreshLayout行为”、“RecyclerView行为”等代码做了复用，使得显示文章列表、下拉刷新、上拉加载更多等功能的开发得到了大大简化，并使得项目的代码结构更加清晰。此外，采用RecyclerView的设计方法也节约了资源。

难点：抽象出FragmentWithArticles并不是一开始就设计好的，这带来了一些代码重构的工作量。

HomeFragment

这个类是展示主页的Fragment，也是所有FragmentWithArticles的子类中最复杂的一个——因为它是不仅是唯一一个用到了两种RecyclerView.Adapter的Fragment，还是唯一一个使用tabLayout实现可滑动、可增删的分类标签的Fragment。

可滑动、可增删的分类标签

这里使用了TabLayout组件实现，自身已经可以支持左右滑动，还实现了Tab的动态增删newTab()等函数。此外，由于“在页面刷新过程中狂乱点击不同标签”这种行为会造成一些奇怪现象，需要在加载过程中禁用整个TabLayout，于是实现了enableAllTab()和disableAllTab()两个函数。

同时使用两种RecyclerView.Adapter

之所以有这样的需求是因为在设计TabLayout的时候，需要多一个Tab用于跟用户交互实现增删标签，然而同一个RecyclerView的Adapter理论上只能实现一种布局的复用，因此HomeFragment在使用加载文章列表的CustomAdapter的同时，还需要使用加载标签增删UI的TabAdapter。由于一个RecyclerView只能绑定一个Adapter，我们引入了自带的ConcatAdapter类。ConcatAdapter类可以在初始化时传入多个Adapter，然后将ConcatAdapter绑定在RecyclerView中。

难点：要实现不同的Adapter，仅仅使用ConcatAdapter是不够的，还需要处理二者切换时的逻辑。注意到RecyclerView.Adapter的底层逻辑可以概括为“在自身数据改变时重新根据getItemCount()来将对应数据绑定到ViewHolder中”，于是我们在切换Tab时只需让重写的getItemCount()返回对应的数量（显示时正常返回，隐藏时返回0），然后就可以通过`notifyDataSetChanged()`告知RecyclerView更新数据重新绘制了。

亮点：下拉加载更多的速度优化：在大作业给出的api接口文档中，要想实现加载更多文章只能通过增加size参数的方法来实现，这就带来了问题——已经被加载的新闻还会被再次加载，造成了O(N^2)的时间复杂度。然而我发现api接口可以设置*page*参数，这样在实现加载更多文章的行为时，不需要改变size，只需增加page，然后刷新即可，这将时间复杂度降到了**O(n)**。事实上，从体验上也能看出来：在优化前，下拉加载更多是越来越慢的，在size参数大于140之后加载速度会不太能够接受；在优化后，加载速度几乎没有能明显感受到的差异。

上拉加载，下拉刷新

用到了第三方包`SmartRefreshLayout`，只需自定义其`onRefresh()`和`onLoadMore()`两个函数即可。在`onRefresh()`中，我们将api设置到加载page1，禁止tab操作，然后调用ApiAdapter的刷新函数；在`onLoadMore()`中，我们将api的page参数+1，其余相同。在调用ApiAdapter的刷新函数时，我们传入HomeFragment类的`parseJson()`函数作为callback，回调时将取回的数据存入Article中，更新RecyclerView并重新允许tab操作等。

难点：最初，由于刷新的过程是异步进行的，如果在刷新进行中切换到另一个Tab当中，当刷新完成后仍会重新加载，造成一定混乱。这有两种解决方案：一是禁止刷新时选中tab，二是在加载完成之后判断是否还在原tab中来决定是否更新RecyclerView。由于方案二并没有解决用户可能带来的的短时间多次加载，且需要在callback中加一个参数比较麻烦，本App选择方案一。

亮点：上拉加载和下拉刷新可以有很多（炫酷的）动画。

RssFragment

亮点：作为一个新闻App，除了从给定的API中获取新闻，如果能够有更多的渠道——甚至是让用户自定义新闻来源，岂不美哉？而能想到的最方便的获取新闻的渠道就是Rss了，于是本App还有一个功能，便是加载Rss订阅源的XML格式数据，并以相似的形式呈递给用户。同时，在右上角的设置中还可以选择预设的Rss源，或者自己输入Rss源地址。~~其实这里也可以有一个TabLayout的但是我比较懒。~~

实现起来也很简单，就把parseJson的调用改为parseXML，然后用XmlPullParser处理。不过似乎不同Rss源的格式略有不同，这里就采用最通用的一种格式来处理了。（就是item标签下有title，description这种）

说明：这部分内容并不是大作业要求内容。由于RSS源内容格式不定且往往采用html语言描述，在显示时采用WebView控件，且不提供历史记录、收藏的功能（这不应当被视为未完成要求）。

SlideshowFragment

这是搜索界面的Fragment。分别用TextInputEditText、MaterialDatePicker和Spinner实现了关键词、日期范围和类别的搜索设置。最后把参数传给ApiAdapter，进一步传给ListFragment实现搜索。

亮点：使用了MaterialDatePicker，使得日期范围的选择变得轻松，并且UI也更美观。

ArticleFragment

显示文章的标题、发布者、发布时间、内容（文字图片视频），并且负责处理收藏行为，并将已阅读的新闻（及其图片与收藏状态）存储到本地。

`FragmentWithArticles`类在通过Navigation调用ArticleFragment前，会向其传递一个Bundle，这个bundle里面塞了要显示的Article等内容，相当于函数调用时的实参。

ArticleFragment在一个LinearLayout的容器下，通过TextView和ImageView来显示。通过在LinearLayout外面套一层ScrollView来实现滑动显示。

ArticleFragment的一大重要功能就是将以阅读的新闻存储到本地，因此它也能接触到数据库接口ArticleDao。基本地，在开始时ArticleFragment会将文章尝试存入数据库中，但由于这里的Article是从Api中获取的，并没有收藏信息，如果直接存入数据库会将收藏信息覆盖掉，于是这里的冲突策略选择的是Ignore，即“weak Insert”，并且在加载时询问数据库是否有收藏信息。相对的，另一大功能——收藏，所对应的冲突策略是Replace。此外，为了避免用户狂点收藏按钮导致的数据库高频读写，我将Insert操作的时机放在了Fragment生命周期中的onPause()中。

```
public interface ArticleDao {
    @Insert (onConflict = OnConflictStrategy.REPLACE)
    Completable insert(Article article); // 用于收藏
    @Insert (onConflict = OnConflictStrategy.IGNORE)
    Completable weakInsert(Article article); // 用于初始化保存
    ...
}
```

网络与API接口

`ApiAdapter`具有以下几个属性：size, startDate, endDate, words, categories, page，分别对应着api接口中的不同参数，并且有相应的修改属性的方法。它还拥有一个私有方法`getUrl()`，能够将这些参数转化为对应的URL，在调用其`getContent()`方法时将会向`getUrl()`的地址发出GET请求，将返回的内容以String的形式存储，完成后调用传入的callback，告知请求方异步的网络读取操作已经完成。

亮点：将与API相关的操作和网络请求封装了起来，具有良好的复用性。

数据库与本地存储

Article类

Article类的一个重要实现是以JSONObject为参数的构造函数，这相当于是将JSONObject处理成了Article。一般的参数，如title，pubDate等，通过`getString()`函数就可得到，（**难点：**）但是图片不然，因为image参数可能很奇怪（比如"[, , ,]"这种），因此我们需要在用逗号做分割之后再用正则表达式进行一些“纯化”，主要代码如下：

```
String[] res = imageRaw.substring(1, imageRaw.length() - 1).split(",");
for(int i = 0, len = res.length; i < len; i++) {
    Pattern pattern = Pattern.compile("\\s*(\\S+)\\s*");
    Matcher matcher = pattern.matcher(res[i]);
    if(matcher.find())
        image.add(matcher.group(1));
}
```

此外，为了方便向ArticleFragment传递参数，还实现了从bundle的构造函数和与之配套的bundle打包函数`packBundle()`。（这一方法在ApiAdapter类也有用到）

ArticleDao接口

`ArticleDao`类的"实现"本质上就是写了几条SQLite语句，这里列举三个：

```
@Query ("SELECT * FROM article_table WHERE idx =:idx")
Single<Article> selectById(String idx); // 根据NewsID查找Article

@Query ("SELECT * FROM article_table WHERE marked = 1 ORDER BY rowid DESC")
Single<Article[]> selectAllFavorite(); // 倒序获取所有收藏

@Query ("DELETE FROM article_table")
Completable clear(); // 清除本地数据库
```

这里的降序写法是为了满足历史记录和收藏"后看的放在前面"的需求（SQL和SQLite的降序写法好像还不一样，ROOM用的是SQLite。。。）

ArticleDatabase & Converters

ArticleDatabase基本上就纯按照Android Developers上面的例子写的，主要实现就是一个静态方法`getDatabase()`用于获取单例模式下的数据库。

Converters实现了ArrayList和String的相互转换（room数据库要求实现这个converter）。具体方法为调用Gson库，将ArrayList转化为Json格式的字符串。

FileController

FileController的`saveFile()`和`loadFile()`函数都使用了Picasso库来实现，二者的区别在于Picasso的`.into()`里面的参数，save是一个重载了onBitmapLoaded()的Target，load是一个ImageView。此外，为了实现"看过的图片离线也能加载"的需求，在`loadFile()`成功后，会调用`saveFile()`保存到本地。另外之前已经提到过，`loadFile()`传入url和本地文件名，优先加载本地图片，失败后再加载网络资源。

难点与亮点：这部分有很多的异步操作：Picasso的.into(Target)，文件IO，网络……，容易写出Callback Hell，好在回调的层数不是那么多。事实上，这里的`Target`和上面ArticleDao中返回的`Completable`，`Single<T>`，都是支持委托的，他们本身就是为了线型编程，避免写出金字塔形的Callback Hell而存在的。

设置

从UI右上角的三个点里面可以进到设置界面，主要由三个部分组成：清除本地存储、设置RSS源、About（就是个简介）。

亮点：以下应该都算得上是亮点。

清除本地存储

清除本地图片的实现就是获取所有的Article，然后对着文件地址都删一遍就行了（有try-catch无脑删就行了）。清除所有缓存就是在清除所有本地图片之后调用ArticleDao接口中的清空数据库。（又是一个Async Task）

设置Rss源

有两种设置方式：一种通过下拉菜单选择预设的15+个RSS源，另一种就是通过用户输入RSS源地址自定义。其实本来还想对后者加一个检查是否为合法RSS源的判断函数，但是发现自己判断的条件总是不能涵盖所有情形，遂放弃了这个Feature。

其他难点与亮点

- 亮点：采用Material3主题，并且换了个中文字体，使得UI界面更漂亮。
- 难点：做完了感觉花费最多精力的地方就是HomeFragment里面那个两种RecyclerAdapter来回切导致的一大堆问题，感觉这并不是一个好的实现方法。

总结与心得

从8月26号下载Android Studio开始，到8月29日大作业发布，再到9月5日大作业基本完成，能够明显地感觉到对Android Java开发的逻辑逐渐熟悉，对项目的掌控程度也越来越高。对我而言，提升最大的时候是刚开始熟悉安卓开发时有些陡峭的学习曲线——最初真的是连模板里面的几个类都看不懂，但是通过“面向Google”，“面向Android Developers”和“面向StackOverflow”的开发方式，我在改Bug的同时逐渐了解了这个app构建的方式。这要感谢老师的课程设置，使得我不用因为考试压力在前期拘泥于Java语言的细支末节（当然，它们同样重要），可以专心打磨大作业和（浅薄的）学习能力、工程能力。

回首上大学以来的Coding大作业：从Decimal开立方，到Win32编程、Qt编程，这次的Java Android开发给我留下印象最深的是诸多的异步操作。文件IO、HTTP GET、数据库读写、Picasso Load，这可以算是我首次接触各种各样的的异步编程（QT那个网络部分就不用太异步），虽然说代码写得没有理想中那么漂亮，底层原理也有待学习，但至少接触了许多新的理念还是挺有趣的。

顺便吐槽一下这东西真得自学，而且就是网上的资源方便一些：之前还在图书馆借了本《Java与Android移动应用开发》，唯一的作用就是知道了怎么parse Json和XML，而且也没有涉及jetpack内容，好在是Android Developers的文档已经足够详尽了，未尽之处也可以Google。以及课程设置也许可以提前一些？比如第三周把课上完这种。

还有一些Debug的总结在这里。最后感谢教学团队，特别是许斌老师刘明辉助教对课程的所有付出！