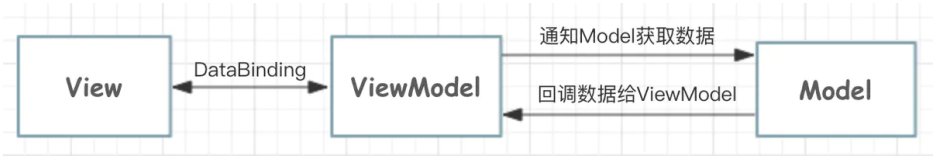
Android MVVM 入门教程

白夜叉小分队 关注
4 2019.06.15 19:33:00 字数 1,633 阅读 22,694

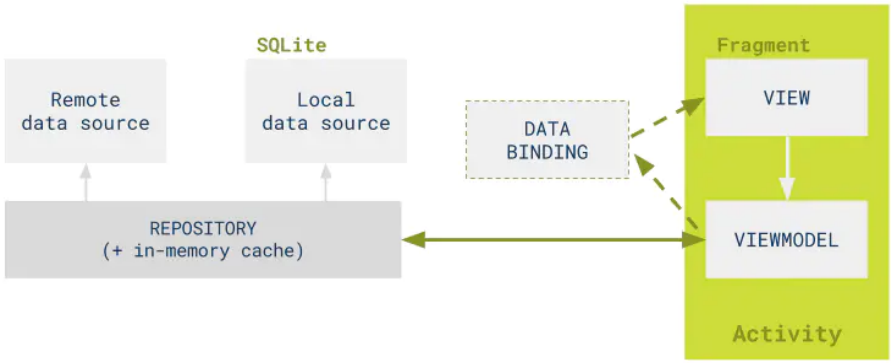
1. MVVM 模式

架构理解

MVVM 模式，即指 Model-View-ViewModel。它将 View 的状态和行为完全抽象化，把逻辑与界面的控制完全交给 ViewModel 处理。如下图：



官方：https://github.com/googlesamples/android-architecture/tree/todo-mvvm-databinding/



MVVM 由下面三层组成：

- View：主要进行视图控件的一些初始设置，不应该有任何的数据逻辑操作。
- Model：定义实体类，以及获取业务数据模型，比如通过数据库或者网络来操作数据等。
- ViewModel：作为连接 View 与 Model 的中间桥梁，ViewModel 与 Model 直接交互，处理完业务逻辑后，通过 DataBinding 将数据变化反应到用户界面上。

优点

1. **低耦合度**
   在 MVVM 模式中，数据处理逻辑是独立于 UI 层的。ViewModel 只负责提供数据和处理数据，不会持有 View 层的引用。而 View 层只负责对数据变化的监听，不会处理任何跟数据相关的逻辑。在 View 层的 UI 发生变化时，也不需要像 MVP 模式那样，修改对应接口和方法实现，一般情况下ViewModel 不需要做太多的改动。
2. **数据驱动**
   MVVM 模式的另外一个特点就是数据驱动。UI 的展现是依赖于数据的，数据的变化会自然的引发 UI 的变化，而 UI 的改变也会使数据 Model 进行对应的更新。ViewModel 只需要处

写下你的评论... 评论7 赞40 ···

3. **异步线程更新 Model**

   Model 数据可以在异步线程中发生变化，此时调用者不需要做额外的处理，数据绑定框架会将异步线程中数据的变化通知到 UI 线程中交给 View 去更新。

4. **方便协作**

   View 层和逻辑层几乎没有耦合，在团队协作的过程中，可以一个人负责 UI，一个人负责数据处理。并行开发，保证开发进度。

5. **易于单元测试**

   MVVM 模式比较易于进行单元测试。ViewModel 层只负责处理数据，在进行单元测试时，测试不需要构造一个 fragment/Activity/TextView 等等来进行数据层的测试。同理 View 层也一样，只需要输入指定格式的数据即可进行测试，而且两者相互独立，不会互相影响。

6. **数据复用**

   ViewModel 层对数据的获取和处理逻辑，尤其是使用 Repository 模式时，获取数据的逻辑完全是可以复用的。开发者可以在不同的模块，多次方便的获取同一份来源的数据。同样的一份数据，在版本功能迭代时，逻辑层不需要改变，只需要改变 View 层即可。

2. DataBinding

在使用 MVVM 模式之前，我们必须了解 DataBinding。

简介

首先要明确一个 DataBinding 与 MVVM 之间的关系 ↓
MVVM 是一种思想，一种架构模式，而 DataBinding 是谷歌推出的方便实现 MVVM 的工具。
在 DataBinding 库之前，我们经常会写一些重复性很高而且毫无营养的代码，比如：findViewById()、setText()、setOnClickListener() 等。直到2015谷歌 I/O大会推出了 DataBinding，一个实现视图和数据双向绑定的工具。使用 DataBinding 库以后，可以使用声明式布局文件来减少粘结业务逻辑和布局文件的胶水代码，有利于开发者更方便地实现 MVVM 模式。

环境配置

在 Module:app 的 build.gradle 文件添加如下代码：

```
android {
    // ...
    dataBinding {
        enabled = true
    }
}
```

使用方法

使用 DataBinding 的布局文件和普通的布局文件有点不同，DataBinding 布局文件的根标签是 layout 标签，layout 里面有一个 data 元素和 View 元素，这个 View 元素就是我们没使用 DataBinding时候的布局文件。例子代码如下：

```
<layout xmlns:android="http://schemas.android.com/apk/res/android">

    <data>
        <variable
            name="user"
            type="com.example.mvvmdemo.UserBean"/>
    </data>

    <LinearLayout
        android:orientation="vertical" android:layout_width="match_parent"
        android:layout_height="match_parent">
```

```
            android:layout_height="wrap_content"
            android:text="@{user.name}"/>
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@{user.sex}"/>
    </LinearLayout>

</layout>
```

data 元素里面的 user 就是我们自定义的 user 实体类，当我们向 DataBinding 中设置好 user 类以后，我们的两个 TextView 会自动设置 text 的值。
UserBean实体类代码如下：

```
public class UserBean {

    public ObservableField<String> name = new ObservableField<>();
    public ObservableField<String> sex = new ObservableField<>();

    public UserBean(){
        name.set("王小明");
        sex.set("男");
    }
}
```

这个实体类的元素是 DataBinding 中的 ObservableField 类，ObservableField 的作用是，当我们实体类中的值发生改变时，会自动通知View刷新。所以使用 DataBinding 的时候，建议使用 ObservableField 来定义实体类。
之后，我们只需要在 Activity 中绑定 layout 就可以了。下面是使用代码：

```
ActivityMainBinding activityMainBinding = DataBindingUtil.setContentView(this, R.layou
UserBean user = new UserBean();
activityMainBinding.setUser(user);
```

在使用 DataBinding 的时候，我们设置布局使用 DataBindingUtil 工具类中的 setContentView() 方法。设置好了 user 后，layout 中的 TextView便显示为"王小明"和"男"。

优点

1. 再也不需要编写 findViewById
2. 更新 UI 数据时不需再切换至 UI 线程

在某篇博客上看到这样一段评价，直接引用：

> 针对第一个优点，有人说，已经有 ButterKnife 了。针对第二个优点，也有人说，有 RxJava 了。但是 DataBinding，不仅仅能解决这2个问题，它的核心优势在于，它解决了将数据分解映射到各个 view 的问题。针对每个 Activity 或者 Fragment 的布局，在编译阶段，它会生成一个ViewDataBinding 类的对象，该对象持有 Activity 要展示的数据和布局中的各个 view 的引用。同时还有如下优势：将数据分解到各个 view、在 UI 线程上更新数据、监控数据的变化，实时更新，这样一来，你要展示的数据已经和展示它的布局紧紧绑定在了一起。这才是 DataBinding 真正的魅力所在。

PS：个人感觉有点像前端 React Redux 的单向数据流。

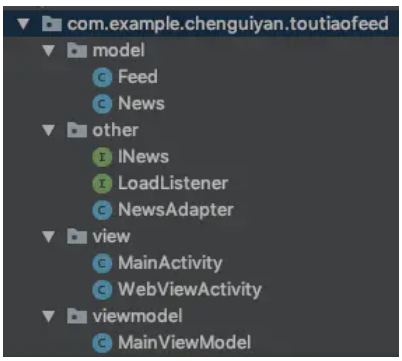
3. 简单实践

项目

Demo：使用 MVVM 模式，利用 Retrofit 获取今日头条首页10条热门新闻推荐，并以

目的：希望通过实践，对 MVVM 模式能够理解得更深刻。

项目文件目录如下：



布局文件

`activity_main.xml`

```
        xmlns:tools="http://schemas.android.com/tools">

    <data>
        <variable
            name="viewModel"
            type="com.example.chenguiyan.toutiaofeed.viewmodel.MainViewModel"/>
    </data>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <android.support.v7.widget.RecyclerView
            android:id="@+id/recycler_view"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"/>
    </LinearLayout>

</layout>
```

item_news.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">

<data>
    <variable
        name="news"
        type="com.example.chenguiyan.toutiaofeed.model.News"/>
</data>

<LinearLayout
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/news_title"
        android:text="@{news.title}"
        android:paddingTop="5dp"
        android:paddingLeft="5dp"
        android:paddingRight="5dp"
        android:textSize="15sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

    <LinearLayout
        android:paddingBottom="5dp"
        android:paddingLeft="5dp"
        android:paddingRight="5dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <TextView
            android:textColor="#acacac"
            android:text="来源: "
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>
        <TextView
            android:text="@{news.source}"
            android:textColor="#acacac"
            android:id="@+id/news_source"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>
    </LinearLayout>

    <ImageView
        android:background="#acacac"
        android:layout_width="match_parent"
        android:layout_height="1dp"/>

</LinearLayout>

</layout>
```

View

MainActivity.java

```java
public class MainActivity extends AppCompatActivity {

    private static final String TAG = "MainActivity";

    public ActivityMainBinding mActivityMainBinding;
    private MainViewModel mViewModel;

    public NewsAdapter mNewsAdapter;
    public List<News> mNewsList = new ArrayList<>();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // 设置dataBinding、viewModel
        mActivityMainBinding = DataBindingUtil.setContentView(this, R.layout.activity_
        mViewModel = new MainViewModel(this);
        mActivityMainBinding.setViewModel(mViewModel);
        // 初始化RecyclerView
        LinearLayoutManager layoutManager = new LinearLayoutManager(this);
        mActivityMainBinding.recyclerView.setLayoutManager(layoutManager);
        mNewsAdapter = new NewsAdapter(this, mNewsList);
        mActivityMainBinding.recyclerView.setAdapter(mNewsAdapter);
        // 加载数据
        mViewModel.loadNews();
    }
}
```

Model

Feed.java

```java
public class Feed {
    private boolean has_more;
    private String message;
    private List<News> data;

    public void setHas_more(boolean has_more) {
        this.has_more = has_more;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public void setData(List<News> data) {
        this.data = data;
    }

    public boolean isHas_more() {
        return has_more;
    }

    public String getMessage() {
        return message;
    }

    public List<News> getData() {
        return data;
    }

    // 通过传进来的url，利用retrofit获取网络数据，回调给viewModel
    public void loadData(String feedUrl, final LoadListener<News> loadListener) {
        OkHttpClient okHttpClient = new OkHttpClient();
        Retrofit retrofit = new Retrofit.Builder()
                .client(okHttpClient)
                .baseUrl(feedUrl)
                .addConverterFactory(GsonConverterFactory.create())
                .build();
        INews iNews = retrofit.create(INews.class);
        Call<Feed> feed = iNews.getFeed();
        feed.enqueue(new Callback<Feed>() {
            @Override
            public void onResponse(Call<Feed> call, Response<Feed> response) {
                // 获取成功
                List<News> newsList = new ArrayList<>();
                for (int i = 0; i < response.body().getData().size(); i++) {
                    newsList.add(response.body().getData().get(i));
```

```
            @Override
            public void onFailure(Call<Feed> call, Throwable t) {
                // 获取失败
                loadListener.loadFailure(t.getMessage());
            }
        });
    }
}
```

News.java

```
public class News {
    private String title;
    private String item_id;
    private String source;

    public News(String title, String item_id, String source) {
        this.title = title;
        this.item_id = item_id;
        this.source = source;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public void setItem_id(String item_id) {
        this.item_id = item_id;
    }

    public void setSource(String source) {
        this.source = source;
    }

    public String getTitle() {
        return title;
    }

    public String getItem_id() {
        return item_id;
    }

    public String getSource() {
        return source;
    }
}
```

ViewModel

MainViewModel.java

```
public class MainViewModel {

    private static final String TAG = "MainViewModel";
    private MainActivity mActivity;
    private String feedUrl;

    public MainViewModel(MainActivity activity) {
        mActivity = activity;
    }

    public void loadNews() {
        // 获取url
        feedUrl = mActivity.getResources().getString(R.string.feed_api_url);
        // 加载数据
        Feed feed = new Feed();
        feed.loadData(feedUrl, new LoadListener<News>() {
            @Override
            public void loadSuccess(List<News> list) {
                // 加载数据成功
                mActivity.mNewsList.addAll(list);
                mActivity.mNewsAdapter.notifyDataSetChanged();
            }
            @Override
```

```
        });
    }
}
```

Other

NewsAdapter.java

```
public class NewsAdapter extends RecyclerView.Adapter {

    private Context mContext;
    private List<News> newsList;
//    private OnItemClickListener mOnItemClickListener = null;

    public static class ViewHolder extends RecyclerView.ViewHolder {
        ItemNewsBinding mItemNewsBinding;

        public ViewHolder(ItemNewsBinding itemNewsBinding) {
            super(itemNewsBinding.getRoot());
            this.mItemNewsBinding = itemNewsBinding;
        }
    }

    public NewsAdapter(Context mContext, List<News> newsList) {
        this.mContext = mContext;
        this.newsList = newsList;
    }

    @NonNull

    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, in
        ItemNewsBinding itemNewsBinding = DataBindingUtil.inflate(LayoutInflater.from(
        // View view = LayoutInflater.from(mContext).inflate(R.layout.item_news, viewG
        return new ViewHolder(itemNewsBinding);
    }

    @Override
    public void onBindViewHolder(@NonNull RecyclerView.ViewHolder viewHolder, final in
        ViewHolder mViewHolder = (ViewHolder) viewHolder;
        // dataBinding绑定
        News news = newsList.get(position);
        mViewHolder.mItemNewsBinding.setNews(news);
        // 设置点击事件，将接口方法回调给MainActivity
//        if (mOnItemClickListener != null) {
//            mViewHolder.mItemNewsBinding.getRoot().setOnClickListener(new View.OnCli
//                @Override
//                public void onClick(View v) {
//                    mOnItemClickListener.onShortClick(position);
//                }
//            });
//            mViewHolder.mItemNewsBinding.getRoot().setOnLongClickListener(new View.O
//                @Override
//                public boolean onLongClick(View v) {
//                    mOnItemClickListener.onLongClick(position);
//                    return false;
//                }
//            });
//        }
        // 直接在adapter里设置点击事件
        mViewHolder.mItemNewsBinding.getRoot().setOnClickListener(new View.OnClickList
            @Override
            public void onClick(View v) {
                String newsUrlPrefix = mContext.getResources().getString(R.string.news
                String httpUrl = newsUrlPrefix + newsList.get(position).getItem_id();
                Intent intent = new Intent(mContext, WebViewActivity.class);
                intent.putExtra("httpUrl", httpUrl);
                mContext.startActivity(intent);
            }
        });
        mViewHolder.mItemNewsBinding.getRoot().setOnLongClickListener(new View.OnLongC
            @Override
            public boolean onLongClick(View v) {
                return false;
            }
        });
    }
```

```
        return newsList.size();
    }

    // 定义点击事件的接口
//    public interface OnItemClickListener {
//        void onShortClick(int position); // 单击
//        void onLongClick(int position); // 长按
//    }

//    public void setOnItemClickListener(OnItemClickListener onItemClickListener) {
//        this.mOnItemClickListener = onItemClickListener;
//    }
}
```

INews.java

```
public interface INews {
    @GET(".")
    Call<Feed> getFeed();
}
```

LoadListener.java

```
public interface LoadListener<T> {
    void loadSuccess(List<T> list);
    void loadFailure(String message);
}
```

strings.xml

```
<resources>
    <string name="feed_api_url">https://www.toutiao.com/api/pc/feed/</string>
    <string name="news_url_prefix">https://www.toutiao.com/a</string>
</resources>
```

Json解析


```
{
    "has_more":false,
    "message":"success",
    "data":[
        Object{...},
        Object{...},
        Object{...},
        Object{...},
        Object{...},
        Object{...},
        Object{...},
        Object{...},
        Object{...},
        Object{...}
    ],
    "next":{
        "max_behot_time":1551432969
    }
}
```

```
⊞Object{...},
⊞Object{...},
⊟{
    "chinese_tag":"时政",
    "media_avatar_url":"//p1.pstatp.com/large/8532/7581013616",
    "is_feed_ad":false,
    "tag_url":"search/?keyword=%E6%97%B6%E6%94%BF",
    "title":"财政部三定方案公布：设部长1名副部长4名",
    "single_mode":false,
    "middle_mode":false,
    "abstract":"第一条根据党的十九届三中全会审议通过的《中共中央关于深化党和国家机构改革的决定》、《深化党和国家机构改革方案》和第十三届全国人民代表大会第一次会议批准的《国务院机构改革方案》，制定本规定。",
    "tag":"news_politics",
    "behot_time":1551433139,
    "source_url":"/group/6663336896573211148/",
    "source":"新京报",
    "more_mode":false,
    "article_genre":"article",
    "comments_count":4,
    "group_source":2,
    "item_id":"6663336896573211148",
    "has_gallery":false,
    "group_id":"6663336896573211148",
    "media_url":"/c/user/5540918998/"
},
⊞Object{...},
```


写下你的评论... 评论7 赞40