链滴

Android 教程 2020 - RecyclerView 显示多种 item

作者：RustFisher

原文链接：https://ld246.com/article/1580975989820

来源网站：链滴

Android教程2020 - 系列总览

本文链接

前面我们已经用RecyclerView显示一些数据。也知道如何获取滑动的距离。

前面我们的列表中显示的都是同类数据。如果要在一个列表中显示不同类别的数据，该怎么做呢？

RecyclerView已经有应对这类场景的能力，本文描述如何用RecyclerView显示多种内容（item）。

综述

这个例子中涉及到的类和文件比较多。

放在同一个包里的文件

BaseMultiData.java
BaseVH.java
ItemTypeDef.java
MultiData1.java
MultiData2.java
MultiItemAdapter.java
OneVH.java
ReViewDemoMulti.java
TwoVH.java

layout资源文件

item_multi_1.xml
item_multi_2.xml

场景描述，数据准备

首先假设场景。假设我们有2种数据。第一种数据只有1个字符串，第二种数据有2个字符串。

那么我们可以先设计出一个数据的抽象类。并且设计枚举类型来区分这2种数据。

利用枚举类型来区分数据的类别，ItemTypeDef.java。

```
public class ItemTypeDef {

    public static final int ITEM_TYPE_ONE_TEXT = 1;
    public static final int ITEM_TYPE_2_TEXT = 2;

    public enum Type {
        ONE_TEXT(ITEM_TYPE_ONE_TEXT),
        TWO_TEXT(ITEM_TYPE_2_TEXT);
        int code;

        Type(int code) {
            this.code = code;
        }

        public int getCode() {
            return code;
        }

        public static Type getItemTypeByCode(int code) {
            switch (code) {
                case ITEM_TYPE_ONE_TEXT:
                    return Type.ONE_TEXT;
                case ITEM_TYPE_2_TEXT:
                    return Type.TWO_TEXT;
            }
            return Type.ONE_TEXT;
        }
    }
}
```

枚举类自己带有ordinal()方法来表明枚举的顺序值。

但这个例子中我们将枚举包裹在一个类里面，并且强行使用了int来标记类别。

数据的抽象类

```
public abstract class BaseMultiData {
    public abstract int typeCode();
}
```

表示具体数据的class。

第1种数据class。

```
public class MultiData1 extends BaseMultiData {

    private String str1;

    public MultiData1(String str1) {
        this.str1 = str1;
    }

    public String getStr1() {
        return str1;
```

```
    }

    @Override
    public int typeCode() {
        return ItemTypeDef.Type.ONE_TEXT.getCode();
    }
}
```

第2种数据class。

```
public class MultiData2 extends BaseMultiData {

    private String str1;
    private String str2;

    public MultiData2(String str1, String str2) {
        this.str1 = str1;
        this.str2 = str2;
    }

    public String getStr1() {
        return str1;
    }

    public String getStr2() {
        return str2;
    }

    @Override
    public int typeCode() {
        return ItemTypeDef.Type.TWO_TEXT.getCode();
    }
}
```

可以看到，两种数据类分别持有1和2个String。

资源文件

前面设计了2种数据，对应的，接下来我们设计2种样式。

layout

先准备2种layout。

给第1种数据MultiData1准备了item_multi_1.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="40dp"
    android:background="#33691E"
    android:orientation="horizontal">
```

```
    <TextView
        android:id="@+id/tv1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_marginStart="12dp"
        android:textColor="#ffffff" />

</LinearLayout>
```

给第2种数据MultiData2准备了item_multi_2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:background="#ffffff"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/tv1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_marginStart="12dp"
        android:text="tv1"
        android:textColor="#33691E" />

    <TextView
        android:id="@+id/tv2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_marginStart="12dp"
        android:text="tv2"
        android:textColor="#8BC34A" />

</LinearLayout>
```

VH

设计对应的ViewHolder。

首先是抽象类BaseVH。它持有一个ItemTypeDef.Type来表明自己的类别。

```
public abstract class BaseVH extends RecyclerView.ViewHolder {
    private ItemTypeDef.Type type;
    public View root;

    public BaseVH(View itemView, ItemTypeDef.Type type) {
        super(itemView);
        this.root = itemView;
        this.type = type;
    }
```

```
    public ItemTypeDef.Type getType() {
        return type;
    }
}
```

具体实现类。第1种数据的OneVH。

```
public class OneVH extends BaseVH {
    public TextView tv1;

    public OneVH(View itemView, ItemTypeDef.Type type) {
        super(itemView, type);
        tv1 = itemView.findViewById(R.id.tv1);
    }
}
```

第2种数据的TwoVH。

```
public class TwoVH extends BaseVH {
    public TextView tv1;
    public TextView tv2;

    public TwoVH(View itemView, ItemTypeDef.Type type) {
        super(itemView, type);
        tv1 = itemView.findViewById(R.id.tv1);
        tv2 = itemView.findViewById(R.id.tv2);
    }
}
```

设计适配器

现在2种数据的类和资源都准备好了。我们来设计适配器。

RecyclerView.Adapter中用来区分子项类别的方法是public int getItemViewType(int position)。

我们需要复写这个方法，告诉RecyclerView当前子项的类别。它返回的是int。

```
    @Override
    public int getItemViewType(int position) {
        return dataList.get(position).typeCode();
    }
```

适配器完整代码如下

```
public class MultiItemAdapter extends RecyclerView.Adapter<BaseVH> {

    private List<BaseMultiData> dataList = new ArrayList<>();

    public void setDataList(List<BaseMultiData> dataList) {
        this.dataList = dataList;
        notifyDataSetChanged();
    }
```

```
    @NonNull
    @Override
    public BaseVH onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        switch (ItemTypeDef.Type.getItemTypeByCode(viewType)) {
            case ONE_TEXT:
                return new OneVH(LayoutInflater.from(parent.getContext()).inflate(R.layout.item_mu
ti_1, parent, false), ItemTypeDef.Type.ONE_TEXT);
            case TWO_TEXT:
                return new TwoVH(LayoutInflater.from(parent.getContext()).inflate(R.layout.item_mu
ti_2, parent, false), ItemTypeDef.Type.TWO_TEXT);
        }
        return null;
    }

    @Override
    public void onBindViewHolder(@NonNull BaseVH holder, int position) {
        BaseMultiData data = dataList.get(position);
        switch (holder.getType()) {
            case ONE_TEXT:
                ((OneVH) holder).tv1.setText(((MultiData1) data).getStr1());
                break;
            case TWO_TEXT:
                TwoVH twoVH = (TwoVH) holder;
                MultiData2 data2 = (MultiData2) data;
                twoVH.tv1.setText(data2.getStr1());
                twoVH.tv2.setText(data2.getStr2());
                break;
        }
    }

    @Override
    public int getItemCount() {
        return dataList.size();
    }

    @Override
    public int getItemViewType(int position) {
        return dataList.get(position).typeCode();
    }
}
```

要显示多种item，得把它们装在一起。数据列表dataList声明装载的是抽象类BaseMultiData。

创建ViewHolder的onCreateViewHolder方法中，根据viewType来判断应该创建哪一种ViewHolde
。

装配数据的onBindViewHolder方法中，也是根据holder的类别来设置相应的数据。这里用到了强制换。

在activity中使用这个适配器

```
public class ReViewDemoMulti extends AbsActivity {

    private MultiItemAdapter mMultiItemAdapter = new MultiItemAdapter();
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_recyler_view_demo);

        RecyclerView recyclerView = findViewById(R.id.re_view);
        recyclerView.setAdapter(mMultiItemAdapter);
        recyclerView.setLayoutManager(new LinearLayoutManager(this, RecyclerView.VERTICAL,
alse));

        mMultiItemAdapter.setDataList(genData());
    }

    private List<BaseMultiData> genData() {
        List<BaseMultiData> list = new ArrayList<>();
        for (int i = 0; i < 60; i++) {
            if (i % 2 == 0) {
                list.add(new MultiData1("[Data1] " + i));
            } else {
                list.add(new MultiData2("[Data2] " + i, "Type2"));
            }
        }
        return list;
    }
}
```

运行起来即可看到效果。

这是一个简单的例子chestnut，工作中的开发需求会更复
或繁杂。可以本文介绍的方法为基础，实现具体的开发需求。

工程放这里： https://github.com/AnRFDev/Tutorial2020

相关阅读

RecyclerView - 使用入门

RecyclerView点击事件 - 如何设置点击事件

RecyclerView示例 - 实际使用