View控件

ListView

RecyleView

RecyclerView的优点

RecyclerView并不会完全替代ListView（这点从ListView没有被标记为@Deprecated可以看出），两者的使用场景不一样。但是RecyclerView的出现会让很多开源项目被废弃，例如横向滚动的ListView, 横向滚动的GridView, 瀑布流控件，因为RecyclerView能够实现所有这些功能。

比如：有一个需求是屏幕竖着的时候的显示形式是ListView，屏幕横着的时候的显示形式是2列的GridView，此时如果用RecyclerView，则通过设置LayoutManager一行代码实现替换。

RecylerView相对于ListView的优点罗列如下：

- RecyclerView封装了viewholder的回收复用，也就是说RecyclerView标准化了ViewHolder，编写Adapter面向的是ViewHolder而不再是View了，复用的逻辑被封装了，写起来更加简单。
  直接省去了listview中convertView.setTag(holder)和convertView.getTag()这些繁琐的步骤。
- 提供了一种插拔式的体验，高度的解耦，异常的灵活，针对一个Item的显示RecyclerView专门抽取出了相应的类，来控制Item的显示，使其的扩展性非常强。
- 设置布局管理器以控制Item的布局方式，横向、竖向以及瀑布流方式
  例如：你想控制横向或者纵向滑动列表效果可以通过LinearLayoutManager这个类来进行控制(与GridView效果对应的是GridLayoutManager,与瀑布流对应的StaggeredGridLayoutManager等)。
  也就是说RecyclerView不再拘泥于ListView的线性展示方式，它也可以实现GridView的效果等多种效果。
- 可设置Item的间隔样式（可绘制）
  通过继承RecyclerView的ItemDecoration这个类，然后针对自己的业务需求去书写代码。
- 可以控制Item增删的动画，可以通过ItemAnimator这个类进行控制，当然针对增删的动画，RecyclerView有其自己默认的实现。
  但是关于Item的点击和长按事件，需要用户自己去实现。

基本使用

```
recyclerView = (RecyclerView) findViewById(R.id.recyclerView);
LinearLayoutManager layoutManager = new LinearLayoutManager(this );
//设置布局管理器
recyclerView.setLayoutManager(layoutManager);
//设置为垂直布局，这也是默认的
layoutManager.setOrientation(OrientationHelper. VERTICAL);
//设置Adapter
recyclerView.setAdapter(recycleAdapter);
 //设置分隔线
recyclerView.addItemDecoration( new DividerGridItemDecoration(this ));
//设置增加或删除条目的动画
recyclerView.setItemAnimator( new DefaultItemAnimator());
```

在使用RecyclerView时候，必须指定一个适配器Adapter和一个布局管理器LayoutManager。适配器继承RecyclerView.Adapter类，具体实现类似ListView的适配器，取决于数据信息以及展示的UI。布局管理器用于确定RecyclerView中Item的展示方式以及决定何时复用已经不可见的Item，避免重复创建以及执行高成本的findViewById()方法。

可以看见RecyclerView相比ListView会多出许多操作，这也是RecyclerView灵活的地方，它将许多动能暴露出来，用户可以选择性的自定义属性以满足需求。

- item中的布局
  android:layout_width="match_parent"android:layout_height="wrap_content"有作用

[honyang recyleView详解]（https://blog.csdn.net/lmj623565791/article/details/45059587）

小结

ListView和RecyleView我更多的是参考网络上的笔记，因此，此笔记仅仅贴了个参考链接，其中在练习recyleView的时候感受到recyleView的强大，自己在其中自定义划线的部分还是有点不是很清楚，需要以后再次练习；