

**RecyclerView的相册风格扩展设计文档**

Version 1.0.2

修订历史

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| 2020-07-23 | 1.0.0 | 初始版本 | 麦青焕 |
| 2020-07-27 | 1.0.1 | 添加数据转换类SortDataHelper, 添加监听数据刷新时刷新坐标的机制。 | 麦青焕 |
| 2020-08-02 | 1.0.2 | 修改外部更新数据会crash的bug, 和更改选中机制，改用数据bean作为标识 | 麦青焕 |

目录

[1. 简介 4](#_Toc12903)

[1.1 目的 4](#_Toc30119)

[1.2 范围 4](#_Toc15184)

[2. 环境 4](#_Toc22280)

[2.1 编译环境 4](#_Toc11513)

[2.2 运行环境 4](#_Toc8777)

[3. 架构 5](#_Toc17825)

[3.1 架构说明 5](#_Toc22448)

[4. 基本使用 6](#_Toc23409)

[4.1 添加数据 6](#_Toc22481)

[4.2 初始化RecyclerView (修改) 6](#_Toc27509)

[5. 自定义使用 6](#_Toc30512)

[5.1 自定义标题 6](#_Toc7486)

[5.2 自定义内容 6](#_Toc14278)

[5.3 相关类与接口: 7](#_Toc18631)

[6. 添加粘性头效果 7](#_Toc8166)

[6.1 相关类与接口： 7](#_Toc14420)

[6.2 用法： 7](#_Toc8470)

[7. 旋钮/按键事件响应 7](#_Toc28253)

[7.1 相关类与接口： 7](#_Toc22226)

[7.2 用法： 7](#_Toc14279)

[8. 选择删除 8](#_Toc25691)

[8.1 相关类与接口: 8](#_Toc21754)

[8.2 用法： 8](#_Toc12792)

[8.3 其他删除方法： 8](#_Toc6296)

[9. 转换数据 8](#_Toc3420)

[9.1 相关类与接口: 8](#_Toc4310)

[9.2 用法： 9](#_Toc3590)

[10. 点击事件 9](#_Toc14543)

[11. 选中/取消选中监听回调(新增) 9](#_Toc26071)

[11.1 接口：在GridRecyclerAdapter.java中 9](#_Toc22859)

[11.2 接口调用样例： 10](#_Toc18231)

[12. 参考资料 10](#_Toc15561)

# 简介

## 目的

本项目开发是为了公司的第三次考核，后期可以扩展为项目所用。

项目实现一个类似相册风格的RecyclerView布局，界面如下：



## 范围

此文档适用于本项目开发、维护、测试人员。

# 环境

## 编译环境

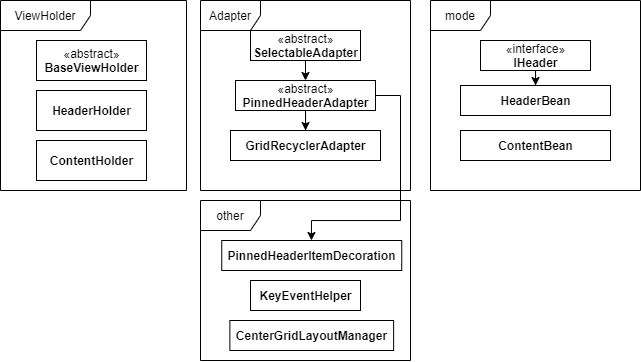
|  |  |
| --- | --- |
| 编辑系统 | Windows 10 |
| IDE | Android Studio 4.0 |
| Gradle版本 | com.android.tools.build:gradle:3.1.2 |

## 运行环境

|  |  |
| --- | --- |
| 兼容系统 | Android 5.1 |
| RecyclerView版本 | com.android.support:recyclerview-v7:22.+ |

# 架构

在RecyclerView的适配器模式基础上，把ViewHolder独立出来，实现ViewHolder与Adapter的解耦



## 架构说明

### ViewHolder层任务

1. 展示UI
2. 把数据绑定到UI上

### Adapter层任务

1. 多类型Item
2. 粘性头
3. 实体按键/旋钮事件处理，和选中item滚动居中
4. 选择删除，和全选状态监听

### Model层任务

1. 标识标题item和普通item
2. 提供绑定到view的数据

# 基本使用

## 添加数据

1. List<Object> dataList = **new** ArrayList<>();
2. dataList.add(**new** HeaderBean("标题"));
3. dataList.add(**new** ContentBean("图片路径"));

## 初始化RecyclerView (修改)

mAdapter = new GridRecyclerAdapter<>(dataList, *SPAN\_COUNT*);  
mLayoutManager = new CenterGridLayoutManager(context, *SPAN\_COUNT*, mAdapter);  
recyclerView.setLayoutManager(mLayoutManager);  
recyclerView.setAdapter(mAdapter);

# 自定义使用

## 自定义标题

1. **public** **class** 标题数据类 implements IHeader {
2. // do something ....
3. }
4. **public** **class** 标题Holder extends BaseViewHolder<标题数据类> {
6. **public** 标题Holder(Context context, ViewGroup root) {
7. super(context, root, 布局文件ID);
8. // do something ... e.g. findViewById
9. }
11. @Override
12. **public** **void** bindData(标题数据类 s, **int** position,
13. GridRecyclerAdapter adapter) {
14. // 绑定数据到View
15. }
16. }

## 自定义内容

1. **public** **class** 内容数据类 {
2. // do something ....
3. }
4. **public** **class** 内容Holder extends BaseViewHolder<内容数据类> {
6. **public** ContentHolder(Context context, ViewGroup root) {
7. super(context, root, 布局文件ID);
8. // do something ... e.g. findViewById
9. }
11. @Override
12. **public** **void** bindData(内容数据类s, **int** position,
13. BaseGridRecyclerAdapter adapter) {
14. // 绑定数据到View
15. }
16. }

## 相关类与接口:

BaseViewHolder.java

GridRecyclerAdapter.java

IHeader.java

# 添加粘性头效果

## 相关类与接口：

PinnedHeaderItemDecoration.java

PinnedHeaderAdapter.java

## 用法：

1. recyclerView.addItemDecoration(**new** PinnedHeaderItemDecoration());

# 旋钮/按键事件响应

## 相关类与接口：

KeyEventHelper.java

GridRecyclerAdapter.java

## 用法：

1. @Override
2. **public** ActivityOperationBean onSelectedChange(CoordinateBean coordinateBean) {
3. ......
4. KeyEventHelper.PositionBean positionBean =
5. mAdapter.moveToNext(coordinateBean.getX(), coordinateBean.getY());
6. **int** position = positionBean.getPosition();
7. mAdapter.setCurrentHighLight(position);
8. mAdapter.notifyDataSetChanged();
9. mRvGrid.smoothScrollToPosition(position);
10. ......
11. }

注意：这里是最简写法，实际应用需要更加实际情况

# 选择删除

## 相关类与接口:

SelectableAdapter.java

GridRecyclerAdapter.java

/\* 把选中的类删除\*/

public void deleteSelected()

## 用法：

1. mAdapter.deleteSelected();

## 其他删除方法：

// 1. 获取选中item的index

List<int> indexList = mAdapter.getSelectedItems();

// 2. 用indexList从你的数据集合中拿到要更新的数据,并且删除

for (int i = 0; i < indexList .size(); ++i) {  
 mDataList.remove(indexList.get(i));  
}

// 3. 更新数据到adapter

1. mAdapter.setDataList(你的数据List);

# 转换数据

## 相关类与接口:

SortDataHelper.java

## 用法：

1. **public** **class** 你的数据类 extends ContentBean{
2. // do something ....
3. }

List<ContentBean> dataList= new ArrayList<>();  
list.add(new ContentBean("2016年12月11日", ""));  
list.add(new ContentBean("2016年12月11日", ""));  
list.add(new ContentBean("2016年12月11日", ""));

initAdapter(GridRecyclerActivity.this, mRvGrid, SortDataHelper.*sortData*(dataList));

# 点击事件

mAdapter.setOnItemClickListener(new GridRecyclerAdapter.OnItemClickListener() {  
 @Override  
 public void onItemClick(View v, int position) {  
 Toast.*makeText*(v.getContext(), "点击了" + position, Toast.*LENGTH\_SHORT*).show();  
 }  
});

# 选中/取消选中监听回调(新增)

## 接口：在GridRecyclerAdapter.java中

public interface OnSelectListener {  
 */\*\*  
 \* 选择回调  
 \*  
 \** ***@param*** *v 当前被点击的view  
 \** ***@param*** *position 当前被点击的位置  
 \** ***@param*** *isChecked true：选择， false：取消选择  
 \** ***@param*** *allSelectedItems 已经勾选的item  
 \** ***@param*** *contentItemCount 除标题item外，全部item总数  
 \*/* void onSelect(View v, int position, boolean isChecked, List allSelectedItems, int contentItemCount);  
}  
  
public void setOnSelectListener(OnSelectListener onSelectListener) {  
 mOnSelectListener = onSelectListener;  
}

## 接口调用样例：

mAdapter.setOnSelectListener(new GridRecyclerAdapter.OnSelectListener() {  
 @Override  
 public void onSelect(View v, int position, boolean isChecked, List allSelectedItems, int contentItemCount) {  
 }  
});

# 参考资料

粘性头：

参考博客：

<https://www.jianshu.com/p/1558cbd20d17>

<https://blog.csdn.net/little762/article/details/79391807>

<https://blog.csdn.net/briblue/article/details/70211942>

Github: <https://github.com/tuacy/RecyclerPinnedHeader>

滚动居中：

参考博客：

<https://blog.csdn.net/iblade/article/details/90449089>