RecyclerView

2018年4月17日 18:16

RecyclerView，一个滑动组件，同样拥有item回收复用的功能

优点：

- RecyclerView封装了viewholder的回收复用，即标准化了viewholder，编写adapter面向的是viewholder而不再是view
- 针对一个item的显示Recyclerview专门有相应的类，来控制item的显示
- 设置布局管理器（**LinearLayoutManager**）以控制item的布局方式，横向、竖向以及瀑布流方式（(与GridView效果对应的是GridLayoutManager,与瀑布流对应的还StaggeredGridLayoutManager等)
- 可设置item的间隔样式（通过继承recyclerview的ItemDecoration类）
- 可以控制Item增删的动画，可以通过ItemAnimator这个类进行控制

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。

```
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());
```

创建适配器：

①创建Adapter：创建一个继承RecyclerView.Adapter<VH>的Adapter类（VH是ViewHolder的类名）

② 创建ViewHolder：在Adapter中创建一个继承RecyclerView.ViewHolder的静态内部类，记为VH。ViewHolder的实现和ListView的ViewHolder实现几乎一样。

③ 在Adapter中实现3个方法：

> onCreateViewHolder（）

这个方法主要生成为每个Item inflater出一个View，但是该方法返回的是一个ViewHolder。该方法把View直接封装在ViewHolder中，然后我们面向的是ViewHolder这个实例，当然这个ViewHolder需要自己编写

```
View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.item_1, parent, false);
```

> onBindViewHolder（）

这个方法主要用于适配渲染数据到View中，方法提供给你一个viewHolder，而不是原来的convertView

> getItemCount（）

这个方法就类似于BaseAapter的getCount方法，即总共有多少个条目

RecyclerView将ListView中getView()的功能拆分成了onCreateViewHolder()和onBindViewHolder()

基本的Adapter实现：

//①创建Adapter

```
publicclassNormalAdapterextendsRecyclerView.Adapter<NormalAdapter.Vh>{
```

//②创建ViewHolder

```
publicstaticclassVhextendsRecyclerView.ViewHolder{
publicfinalTextViewtitle;
publicVH(Viewv){
super(v);
title=(TextView)v.findViewById(R.id.title);
}
}
privateList<String>mDatas;
publicNormalAdapter(List<String>data){
this.mDatas=data;
}
```

//③在Adapter中实现3个方法

```
@Override
publicVhonCreateViewHolder(ViewGroupparent,intviewType){
//LayoutInflater.from指定写法
Viewv=LayoutInflater.from(parent.getContext()).inflate(R.layout.item_1,parent,false);
returnnewVH(v);
}
@Override
publicvoidonBindViewHolder(VHholder,intposition){
holder.title.setText(mDatas.get(position));
holder.itemView.setOnClickListener(newView.OnClickListener(){
@Override
publicvoidonClick(Viewv){
//item点击事件
}
});
}
@Override
publicintgetItemCount(){
returnmDatas.size();
}
}
```

设置RecyclerView

为RecyclerView进行四大设置，即四大组成：

- ✧ Layout Manager(必选)
- ✧ Adapter(必选)
- ✧ Item Decoration(可选，默认为空)
- ✧ Item Animator(可选，默认为DefaultItemAnimator)

```
List<String> data = initData();
RecyclerView rv = (RecyclerView) findViewById(R.id.rv);
rv.setLayoutManager(new LinearLayoutManager(this));
rv.setAdapter(new NormalAdapter(data));
```

四大组成

RecyclerView的四大组成是：

- Layout Manager：Item的布局。
- Adapter：为Item提供数据。
- Item Decoration：Item之间的Divider。
- Item Animator：添加、删除Item动画。

Layout Manager布局管理器（用于控制RecyclerView效果）

LayoutManager负责RecyclerView的布局，其中包含了Item View的获取与回收。

RecyclerView提供了三种布局管理器：

- **LinearLayoutManager** 以垂直**或者**水平列表方式展示Item
- **GridLayoutManager** 以**网格**方式展示Item
- **StaggeredGridLayoutManager** 以**瀑布流**方式展示Item

① LinearLayoutManager线性

- LinearLayoutManager(Context context)
  - 该构造函数**默认**是**竖直**方向
- LinearLayoutManager(Context context, int orientation, boolean reverseLayout)
  - orientation方向，水平（OrientationHelper.HORIZONTAL）或者竖直（OrientationHelper.VERTICAL）
  - reverseLayout是否逆向，true：布局逆向展示，false：布局正向显示

② GridLayoutManager

- GridLayoutManager(Context context, int spanCount)
  - spanCount，每列或者每行的item个数，设置为1，就是列表样式
  - 该构造函数**默认**是**竖直**方向的网格样式
- GridLayoutManager(Context context, int spanCount, int orientation,boolean reverseLayout)
  - spanCount每列或者每行的item个数，设置为1，就是列表样式
  - orientation方向，水平（OrientationHelper.HORIZONTAL）或者竖直（OrientationHelper.VERTICAL）
  - reverseLayout是否逆向，true：布局逆向展示，false：布局正向显示

③ StaggeredGridLayoutManager瀑布流

StaggeredGridLayoutManager(int spanCount, int orientation)

- spanCount每列或者每行的item个数
- orientation方向，水平（OrientationHelper.HORIZONTAL）或者竖直（OrientationHelper.VERTICAL）

LayoutManager常用API：

```
canScrollHorizontally();//能否横向滚动
canScrollVertically();//能否纵向滚动
scrollToPosition(int position);//滚动到指定位置

setOrientation(int orientation);//设置滚动的方向
getOrientation();//获取滚动方向

findViewByPosition(int position);//获取指定位置的Item View
findFirstCompletelyVisibleItemPosition();//获取第一个完全可见的Item位置
findFirstVisibleItemPosition();//获取第一个可见Item的位置
findLastCompletelyVisibleItemPosition();//获取最后一个完全可见的Item位置
findLastVisibleItemPosition();//获取最后一个可见Item的位置
```

Item Decoration间隔样式

RecyclerView通过**addItemDecoration()**方法添加item之间的分割线。Android并没有提供实现好的Divider，因此**任何分割线样式都需要自己实现。**

自定义间隔样式需要继承**RecyclerView.ItemDecoration**类，该类是个**抽象类**，主要有三个方法。

- **onDraw**(Canvas c, RecyclerView parent, State state)，在**Item绘制之前**被调用，该方法主要用于绘制间隔样式。
- **onDrawOver**(Canvas c, RecyclerView parent, State state)，在**Item绘制之前**被调用，该方法主要用于绘制间隔样式。
- **getItemOffsets**(Rect outRect, View view, RecyclerView parent, State state)，设置**item的偏移量**，**偏移的部分用于填充间隔样式**，即设置分割线的宽、高；在RecyclerView的onMesure()中会调用该方法。

onDraw()和onDrawOver()这两个方法都是用于绘制间隔样式，我们只需要复写其中一个方法即可。

Item Animator动画

RecyclerView能够通过**mRecyclerView.setItemAnimator(ItemAnimator animator)设置添加、删除、移动、改变的动画效果。**

RecyclerView提供了**默认**的ItemAnimator实现类：**DefaultItemAnimator**。如果没有特殊的需求，默认使用这个动画即可。

```
// 设置Item添加和移除的动画
mRecyclerView.setItemAnimator(new DefaultItemAnimator());
```

>>>

博客： https://blog.csdn.net/moira33/article/details/79403892