转载请注明链接：http://blog.csdn.net/feather_wch/article/details/79630459

总结列表和RecylerView的要点。

1. ListView的简略介绍，市面上资料很多
2. RecyclerView给出最基本的步骤，和最简单的实现方法。

列表和RecyclerView

版本:2018/3/28-1(22:10)

- 列表和RecyclerView
  - ListView
  - RecyclerView
    - 分割线定制：ItemDecoration
    - 拖拽和滑动删除
  - 参考资料

ListView

比较简单，四步骤：

1. ListView所在的布局和Item的布局
2. 自定义适配器(继承BaseAdapter)
3. 给ListView设置适配器对象

自定义适配器主要四部分工作：

1. getCount()-返回数据集的总数，如List的size等
2. getItem(position)-返回数据集中position位置的数据
3. getItemId(int position)-return position即可
4. getView()-通过Item的布局去自定义每行的效果

convertView+ViewHolder是重写getView()的最好办法，能减少 `findViewById` 的次数，也减少了 重绘`View`

RecyclerView

1、RecyclerView的特点

> 1. 名称 回收循环视图 ---只管回收与复用View
> 2. 强制使用 `ViewHolder` ，并将其封装，该控件会自动回收和复用

2、RecyclerView的优点

> 1. Item 复用性高
> 2. 灵活、可定制化高、可扩展性高 -提供插拔式体验；高度解耦；
> 3. 可控制 显示的方式 -通过布局管理器 LayoutManager
> 4. 可控制 Item间的间隔(可绘制) -通过 ItemDecoration
> 5. 可控制 Item的增删动画 -通过 ItemAnimator

3、RecyclerView的缺点

> 实现 控制点击 、 长按事件 比较麻烦(要自己写)

4、RecyclerView的实现步骤

> 1. 定义包含 RecyclerView 的布局
> 2. 实现 RecyclerView 的 Item布局
> 3. 继承并实现 RecyclerView.Adapter (ViewHolder的实现， 点击事件 的实现，定义 列表等数据结构 用于存储数据)
> 4. 给 RecyclerView 设置 Adapter，并且设置监听器

5、RecyclerView的最简单实现

> 1-Activity的布局和 Item 的布局

```
//activity_recler_view.XMML
<android.support.constraint.ConstraintLayout  ...>

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

    </android.support.v7.widget.RecyclerView>

</android.support.constraint.ConstraintLayout>
```

```
//reclerview_item.xml
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <TextView
        android:id="@+id/item_text"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:gravity="center"
        android:text="TextView"
        android:textColor="@color/colorAccent"
        android:textSize="20dp" />

</LinearLayout>
```

> 2-自定义Adapter

```
public class RecyclerViewAdapter extends RecyclerView.Adapter{
    //1. 布局
    private LayoutInflater mInflater;
    //2. 数据
    private ArrayList<String> mDatas;
    public RecyclerViewAdapter(Context context, ArrayList<String> stringArrayList){
        super();
        mInflater = LayoutInflater.from(context);
        mDatas = stringArrayList;
    }
    //3. 自定义ViewHolder
    class ViewHolder extends RecyclerView.ViewHolder{
        private TextView name;

        public ViewHolder(View itemView) {
            super(itemView);
            name = itemView.findViewById(R.id.item_text);
            itemView.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    if(mItemClickListener != null){
                        mItemClickListener.onItemClick(v, getPosition());
                    }
                }
            });
        }

        public TextView getName(){
            return name;
        }
    }

    //4. 通过Item的布局创建ViewHolder
    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        return new ViewHolder(mInflater.inflate(R.layout.reclerview_item, parent, false)));
    }
    //5. 完成类似于getView里面显示数据的功能
    @Override
    public void onBindViewHolder(RecyclerView.ViewHolder holder, int position) {
        ViewHolder viewHolder = (ViewHolder) holder;
        viewHolder.name.setText(mDatas.get(position));
    }
    //6. 获取数据size
    @Override
    public int getItemCount() {
        return mDatas.size();
    }

    //7. 点击事件的监听
    private ItemClickListener mItemClickListener;
    public interface ItemClickListener{
        public void onItemClick(View view, int position);
    }
```

```
    public void setOnItemClickListener(ItemClickListener listener){
        mItemClickListener = listener;
    }
}
```

> 3-Activity.java

```
public class ReclerViewActivity extends Activity {

    RecyclerView mRecyclerView;
    RecyclerViewAdapter mRecyclerViewAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_recler_view);
        mRecyclerView = findViewById(R.id.activity_recler_view);

        ArrayList<String> arrayList = new ArrayList<>();
        arrayList.add("Wang");
        arrayList.add("Gu");
        arrayList.add("Li");
        arrayList.add("Wen");

        //1. 布局
        LinearLayoutManager linearLayoutManager = new LinearLayoutManager(this);
        mRecyclerView.setLayoutManager(linearLayoutManager);
        //2. 适配器
        mRecyclerViewAdapter = new RecyclerViewAdapter(this, arrayList);
        mRecyclerViewAdapter.setOnItemClickListener(new RecyclerViewAdapter.ItemClickListener()
            @Override
            public void onItemClick(View view, int position) {
                Toast.makeText(ReclerViewActivity.this, "onItemClick position=" + position, Toa
            }
        });
        mRecyclerView.setAdapter(mRecyclerViewAdapter);
    }
}
```

6、RecyclerView的基本使用

```
mRecyclerView = findView(R.id.id_recyclerview);
//1. 设置布局管理器
mRecyclerView.setLayoutManager(layout);
//2. 设置adapter
mRecyclerView.setAdapter(adapter)
//3. 设置Item增加、移除动画
mRecyclerView.setItemAnimator(new DefaultItemAnimator());
//4. 添加分割线
mRecyclerView.addItemDecoration(new DividerItemDecoration(getActivity(), DividerItemDecoration.
```

分割线定制：ItemDecoration

7、ItemDecoration的使用和主要方法的要点(可以参考官方提供的 `DividerItemDecoration` 的内部实现)

> 1. `ItemDecoration` 是抽象类
> 2. 使用方法； `mRecyclerView.addItemDecoration()`
> 3. 四个方法： `onDraw()\onDrawOver()\getItemOffsets\getItemOffsets`
> 4. `addItemDecoration()` 后，绘制顺序 `onDraw()->onDrawOver()` -复写其中一个即可。
> 5. `getItemOffsets` 可以通过 `outRect.set()` 为每个Item设置一定的偏移量，主要用于 绘制`Decorator` 。

```
mRecyclerView = findViewById(R.id.activity_imageview_recyclerview);
mRecyclerView.setLayoutManager(new LinearLayoutManager(this));
mRecyclerView.setAdapter(mRVAdapter = new RVAdapter(this, cityNames));
//1. 这里设置分割线
mRecyclerView.addItemDecoration(new DividerItemDecoration(this, DividerItemDecoration.VERTICAL)
```

8、LayoutManager的布局管理器

拖拽和滑动删除

9、RecyclerView的拖拽和滑动删除如何实现?

> 1. 都是通过 `ItemTouchHelper` 实现
> 2. 拖拽与滑动需要在 `getMovementFlags` 中进行开关
> 3. 拖拽的实现在 `onMove()` 中完成
> 4. 滑动删除是在 `onSwiped()` 中实现

```
//1. 帮助快速处理拖拽和滑动删除的类
ItemTouchHelper itemTouchHelper = new ItemTouchHelper(new ItemTouchHelper.Callback() {
    /**====================================
     * 2、通过返回值设置是否处理拖拽或者滑动事件
     *====================================*/
    @Override
    public int getMovementFlags(RecyclerView recyclerView, RecyclerView.ViewHolder viev
        if(recyclerView.getLayoutManager() instanceof GridLayoutManager){
            //1. Grid布局中上下左右都可以拖拽
            int dragFlags = ItemTouchHelper.UP | ItemTouchHelper.DOWN
                    | ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT;
            int swipeFlags = 0;
            return makeMovementFlags(dragFlags, swipeFlags);
        }else{
            //2. 列表只有UP/DOWN两种方式
            int dragFlags = ItemTouchHelper.UP | ItemTouchHelper.DOWN;
            //3. 滑动删除只有两个方向
            int swipeFlags = ItemTouchHelper.START | ItemTouchHelper.END;
            return makeMovementFlags(dragFlags, swipeFlags);
        }
    }
    /**====================================
     * 3、在拖拽过程中会不断回调
     *====================================*/
    @Override
    public boolean onMove(RecyclerView recyclerView, RecyclerView.ViewHolder viewHolder
        //1. 获取到当前Item和目标Item的下标
        int fromPosition = viewHolder.getAdapterPosition();
        int toPosition = target.getAdapterPosition();
        if(fromPosition < toPosition){
            for(int i = fromPosition; i < toPosition; i++){
                Collections.swap(mAdapter.getDatas(), i, i + 1);
            }
        }else{
            for(int i = fromPosition; i > toPosition; i--){
                Collections.swap(mAdapter.getDatas(), i, i - 1);
            }
        }
        mAdapter.notifyItemMoved(fromPosition, toPosition);
        return true;
    }

    //长按Item开始拖拽的时候调用
    @Override
    public void onSelectedChanged(RecyclerView.ViewHolder viewHolder, int actionState)
        //设置为灰色
        if(actionState != ItemTouchHelper.ACTION_STATE_IDLE){
            viewHolder.itemView.setBackgroundColor(Color.LTGRAY);
        }
        super.onSelectedChanged(viewHolder, actionState);
    }
    //拖拽结束松开手指的时候调用
    @Override
    public void clearView(RecyclerView recyclerView, RecyclerView.ViewHolder viewHolder
```

```
            super.clearView(recyclerView, viewHolder);
            viewHolder.itemView.setBackgroundColor(Color.WHITE);
        }

        /**==============================
         * 滑动删除的回调
         *==========================*/
        @Override
        public void onSwiped(RecyclerView.ViewHolder viewHolder, int direction) {
            //彻底删除目标Item
            int targetPostion = viewHolder.getAdapterPosition();
            mAdapter.notifyItemRemoved(targetPostion);
            mAdapter.getDatas().remove(targetPostion);
        }
    });
    itemTouchHelper.attachToRecyclerView(mRecyclerView);
```

10、RecyclerView如何自定义ItemView的拖拽效果(如第一个Item禁止拖拽)?

> 1.需要在 `ItemTouchHelper` 中的 `isLongPressDragEnabled()` 禁止有所Item的拖拽

```
/**
 *  默认禁止所有Item的拖拽，并给RecyclerView设置长按监听事件，并进行处理
 */
@Override
public boolean isLongPressDragEnabled() {
      return false;
}
```

> 2.给RecyclerView设置 `ItemTouchListener(本质通过手势探测器GestureDetectorCompat完成)` -在长按事件中用 `ItemTouchHelper` 的 `startDrag` 方法进行拖拽

```
mRecyclerView.addOnItemTouchListener(new OnRecyclerItemClickListener(mRecyclerView) {
        @Override
        public void onItemClick(RecyclerView.ViewHolder viewHolder) {
            //TODO 点击事件
        }

        @Override
        public void onLongClick(RecyclerView.ViewHolder viewHolder) {
            if(viewHolder.getLayoutPosition() != 0){
                itemTouchHelper.startDrag(viewHolder);
            }
        }
    });
```

> 3-Item点击事件监听器的具体实现:

```
public abstract class OnRecyclerItemClickListener implements RecyclerView.OnItemTouchListener{
    private RecyclerView mRecyclerView;
    private GestureDetectorCompat mGestureDetectorCompat; //手势探测器

    public OnRecyclerItemClickListener(RecyclerView recyclerView){
        mRecyclerView = recyclerView;
        //手势探测器
        mGestureDetectorCompat = new GestureDetectorCompat(mRecyclerView.getContext(), new Gest
            //1. 普通点击操作
            @Override
            public boolean onSingleTapUp(MotionEvent e) {
                View curItemView = mRecyclerView.findChildViewUnder(e.getX(), e.getY());
                if(curItemView != null){
                    RecyclerView.ViewHolder viewHolder = mRecyclerView.getChildViewHolder(curIt
                    onItemClick(viewHolder);
                }
                return true;
            }
            //2. 长按操作
            @Override
            public void onLongPress(MotionEvent e) {
                View curItemView = mRecyclerView.findChildViewUnder(e.getX(), e.getY());
                if(curItemView != null){
                    RecyclerView.ViewHolder viewHolder = mRecyclerView.getChildViewHolder(curIt
                    onLongClick(viewHolder);
                }
            }
        });
    }

    @Override
    public boolean onInterceptTouchEvent(RecyclerView rv, MotionEvent e) {
        return mGestureDetectorCompat.onTouchEvent(e);
    }

    @Override
    public void onTouchEvent(RecyclerView rv, MotionEvent e) {
        mGestureDetectorCompat.onTouchEvent(e);
    }

    @Override
    public void onRequestDisallowInterceptTouchEvent(boolean disallowIntercept) {
    }

    public abstract void onItemClick(RecyclerView.ViewHolder viewHolder);
    public abstract void onLongClick(RecyclerView.ViewHolder viewHolder);
}
```

参考资料

1. Android RecyclerView 使用完全解析

2. clipToPadding解决列表滚动无法触及到Padding的问题
3. 用RecyclerView做一个小清新的Gallery效果
4. RecyclerView 梳理：点击&长按事件、分割线、拖曳排序、滑动删除
5. 【wanandroid】RecyclerView资料列表
6. RecyclerView进阶(一)RecyclerView实现双列表联动