BaseRecyclerViewAdapterHelper

说明

1、`BaseQuickAdapter<T, VH>`为最基本的类型，提供最基础的功能，所有子类都继承于此。

2、`T`为数据类型；`VH`为`ViewHolder`类型，如果不需要自定义，直接使用`BaseViewHolder`即可。

3、框架提供了一个基础`BaseViewHolder`，所有自定义的`ViewHolder`都应该继承于此！

BaseQuickAdapter<T, VH>

1、快速使用

`BaseQuickAdapter<T, VH>`为最基础的类型，直接使用`BaseQuickAdapter<T, VH>`即可简单快速实现一个`Adapter`：

```
public class DemoAdapter extends BaseQuickAdapter<String, BaseViewHolder> {

    /**
     * 构造方法，此示例中，在实例化Adapter时就传入了一个List。
     * 如果后期设置数据，不需要传入初始List，直接调用 super(layoutResId); 即可
     */
    public DemoAdapter(list List<String>) {
        super(R.layout.layout_demo, list);
    }

    /**
     * 在此方法中设置item数据
     */
    @Override
    protected void convert(@NotNull BaseViewHolder helper, @NotNull String item) {
        helper.setText(R.id.tweetName, "This is an Item, pos: " + (helper.getAdapterPosition() - getHeaderLayout
    }
}
```

`Activity`中设置：

```
DemoAdapter adapter = new DemoAdapter(new ArrayList<String>());
mRecyclerView.setAdapter(adapter);

 // 设置新的数据方法
adapter.setNewData(list)
```

2、纯View创建Item

如果不想使用xml布局，想直接使用View代码方式创建，那么构造函数使用`super(0)`，并且在`Adapter`中重写`onCreateDefViewHolder`方法：

```
public class DemoAdapter extends BaseQuickAdapter<String, BaseViewHolder> {

    public DemoAdapter(list List<String>) {
        //布局传递0
        super(0);
    }

    /**
     * 重写此方法，自己创建 View 用来构建 ViewHolder
     */
    @NotNull
    @Override
    protected BaseViewHolder onCreateDefViewHolder(@NotNull ViewGroup parent, int viewType) {
        // 创建自己的布局
        FrameLayout layout = new FrameLayout(getContext());
        ...
        ...
        return createBaseViewHolder(layout);
    }
```

```
    @Override
    protected void convert(@NotNull BaseViewHolder helper, @NotNull String item) {
        ...
    }
}
```

3、BaseQuickAdapter 点击事件

Item 点击事件

代码如下：

```
DemoAdapter adapter = new DemoAdapter();

// 设置点击事件
adapter.setOnItemClickListener(new OnItemClickListener() {
    @Override
    public void onItemClick(@NonNull BaseQuickAdapter adapter, @NonNull View view, int position) {
        Tips.show("onItemClick " + position);
    }
});
```

Item 长按事件

代码如下：

```
adapter.setOnItemLongClickListener(new OnItemLongClickListener() {
    @Override
    public boolean onItemLongClick(@NonNull BaseQuickAdapter adapter, @NonNull View view, int position) {
        Tips.show("onItemLongClick " + position);
```

```
            return true;
        }
    });
```

Item 内子View的点击事件：

注意，请不要在**convert**方法里注册控件**id**

```
// 先注册需要点击的子控件id（注意，请不要写在convert方法里）
adapter.addChildClickViewIds(R.id.btn, R.id.iv_num_add, R.id.item_click);
// 设置子控件点击监听
adapter.setOnItemChildClickListener(new OnItemChildClickListener() {
    @Override
    public void onItemChildClick(@NonNull BaseQuickAdapter adapter, @NonNull View view, int position) {
        if (view.getId() == R.id.btn) {
            Tips.show("onItemChildClick " + position);
        }
    }
});
```

Item 内子View的长按事件：

注意，请不要在**convert**方法里注册控件**id**

```
// 先注册需要长按的子控件id（注意，请不要写在convert方法里）
adapter.addChildLongClickViewIds(R.id.btn, R.id.iv_num_add, R.id.item_click);
// 设置子控件长按监听
adapter.setOnItemChildLongClickListener(new OnItemChildLongClickListener() {
    @Override
    public boolean onItemChildLongClick(@NonNull BaseQuickAdapter adapter, @NonNull View view, int position) {
        if (view.getId() == R.id.btn) {
```

```
            Tips.show("onItemChildLongClick " + position);
        }
        return true;
    }
});
```

4、BaseQuickAdapter主要属性、方法说明

	Java	Kotlin	说明
获取Context	getContext()	context	
数据相关			
获取Adapter中数据	getData()	data	只能get
设置新的数据实例	setNewData()	setNewData()	将会替换List指针引用
添加数据	addData()	addData()	
移除数据	remove()	remove()	
改变某一位置的数据	setData()	setData()	
替换整个数据	replaceData()	replaceData()	不会更改原数据的引用
设置Diff数据（异步，推荐）	setDiffCallback()	setDiffCallback()	配置数据差异化比较的Callback
	setDiffConfig()	setDiffConfig()	更高程度的自定义化配置

	Java	Kotlin	说明
	setDiffNewData(List)	setDiffNewData(List?)	必须先设置setDiffCallback() 或者 setDiffConfig()，否则不生效
设置Diff数据	setDiffNewData(DiffResult, List)	setDiffNewData(DiffResult, List?)	通过DiffResult设置数据，Adapter内部不关心Diff过程，只要结果。
空布局			
设置空布局视图	setEmptyView()	setEmptyView()	仅当 data 为空时，才会显示
是否有空视图	hasEmptyView()	hasEmptyView()	
获取空视图	getEmptyLayout()	getEmptyLayout()	
是否使用空布局	setUseEmpty()	isUseEmpty	
头布局			
添加头布局	addHeaderView()	addHeaderView()	
设置头布局	setHeaderView()	setHeaderView()	
是否有头布局	hasHeaderLayout()	hasHeaderLayout()	
移除头布局	removeHeaderView()	removeHeaderView()	
移除所有头布局	removeAllHeaderView()	removeAllHeaderView()	
脚布局			
添加脚布局	addFooterView()	addFooterView()	

	Java	Kotlin	说明
设置脚布局	setFooterView()	setFooterView()	
是否有脚布局	hasFooterLayout()	hasFooterLayout()	
移除脚布局	removeFooterView()	removeFooterView()	
移除所有脚布局	removeAllFooterView()	removeAllFooterView()	
布局其他属性			
当显示空布局时，是否显示 头布局	setHeaderWithEmptyEnable()	headerWithEmptyEnable	
当显示空布局时，是否显示 脚布局	setFooterWithEmptyEnable()	footerWithEmptyEnable	
点击事件			
item点击事件	setOnItemClickListener()	同java	
item长按事件	setOnItemLongClickListener	同java	
item子view的点击事件	setOnItemChildClickListener	同java	
item子view的长按事件	setOnItemChildLongClickListener	同java	
添加需要响应点击事件的子View id	addChildClickViewIds()	同java	添加以后，setOnItemChildClickListener才会响应
添加需要响应长按事件的子View id	getChildLongClickViewIds()	同java	

	Java	Kotlin	说明
动画			
是否打开动画	setAnimationEnable()	animationEnable	默认:false
动画是否仅第一次执行	setAnimationFirstOnly()	isAnimationFirstOnly	
设置自定义动画	setAdapterAnimation()	adapterAnimation	
设置使用内置默认动画	setAnimationWithDefault()	setAnimationWithDefault()	参数为枚举