链滴

BaseRecyclerViewAdapterHelper 开源项目之 BaseQuickAdapter 源码学习之预加载的实现 (二)

作者：angels

原文链接：https://ld246.com/article/1496286364240

来源网站：链滴

version：2.8.5

又是美好的第一天，今天我们来学习下有关自动加载更多以及预加载相关的代码。

首先我们今天的切入点是：autoLoadMore(int position) 见名知意，是与自动加载更多相关的。我先看下该函数的代码实现

```
 private void autoLoadMore(int position) {
        //只有开启了上拉加载且loadMoreView没有gone且data.size>0 时返回1
        if (getLoadMoreViewCount() == 0) {
            return;
        }
        if (position < getItemCount() - mAutoLoadMoreSize) {
            return;
        }
        if (mLoadMoreView.getLoadMoreStatus() != LoadMoreView.STATUS_DEFAULT) {
            return;
        }
        mLoadMoreView.setLoadMoreStatus(LoadMoreView.STATUS_LOADING);
        if (!mLoading) {
            mLoading = true;
            mRequestLoadMoreListener.onLoadMoreRequested();
        }
    }
```

第一句先判断getLoadMoreViewCount判断是否==0.其实他并不是简单的判断是是否有加载更多视的数量。进入方法里：

```
   */
    public int getLoadMoreViewCount() {
        if (mRequestLoadMoreListener == null || !mLoadMoreEnable) {
            return 0;
        }
        if (!mNextLoadEnable && mLoadMoreView.isLoadEndMoreGone()) {
            return 0;
        }
        if (mData.size() == 0) {
            return 0;
        }
        return 1;
    }
```

我们可以看到，他的返回结果跟很多因素有关，从代码很容易看出:

return 0的情况:

1、你没有设置mRequestLoadMoreListener 或者没有开启mLoadMoreEnable开关；

2、mNextLoadEnable = false ， mNextLoadEnable 在加载更多结束时，你调用

loadMoreEnd(boolean gone) 时会置为false。且 mLoadMoreView 是处于gone状态的。

3、当数据源大小为0时

接下来 autoLoadMore方法中的第二句代码很重要，

```
if (position < getItemCount() - mAutoLoadMoreSize) {
        return;
    }
```

理解起来大概是这样的，mAutoLoadMoreSize是标识开启自动加载更多的一个数量阀值。这个返回巧妙。

假设你的data.size =20 ,mAutoLoadMoreSize =10，当前position=9, 按照理解，这个pisition=9个临界值，因为我们设置了剩余数量<10个时自动加载更多，此时计算9<20-10，position等于9，明后面还有10个数据没渲染，当position=10时(未加载数据还剩9个，此时应该预加载更多)，10<20-0，不成立，代码继续往下走，

执行

```
if (mLoadMoreView.getLoadMoreStatus() != LoadMoreView.STATUS_DEFAULT) {
        return;
    }
```

我们为什么还要做这个判断的。假如不做这个判断。直接执行下面的代码。

```
mLoadMoreView.setLoadMoreStatus(LoadMoreView.STATUS_LOADING);
    if (!mLoading) {
        mLoading = true;
        mRequestLoadMoreListener.onLoadMoreRequested();
    }
```

那么会出现很好玩的 现象，当你快速上滑时，由于position>=10后满足条件，执行加载更多的回调position=11时也会执行，以此类推，那么你将收到多次加载更多的回调。所以我们需要判断此时是当前的加载更能多回调已完成，保证每次到达阀值后只调用一次加载更多回调方法。

理解了这个方法之后，我们接下来看下该方法在哪里被调用呢?

```
@Override
  public int getItemViewType(int position) {
    if (getEmptyViewCount() == 1) {
        //
        boolean header = mHeadAndEmptyEnable && getHeaderLayoutCount() != 0;
        switch (position) {
            case 0:
                if (header) {
                    return HEADER_VIEW;
                } else {
                    return EMPTY_VIEW;
                }
            case 1:
                if (header) {
                    return EMPTY_VIEW;
                } else {
                    return FOOTER_VIEW;
                }
            case 2:
                return FOOTER_VIEW;
            default:
                return EMPTY_VIEW;
```

```
            }
        }
        autoLoadMore(position);
        int numHeaders = getHeaderLayoutCount();
        if (position < numHeaders) {
            return HEADER_VIEW;
        } else {
            int adjPosition = position - numHeaders;
            int adapterCount = mData.size();
            if (adjPosition < adapterCount) {
                return getDefItemViewType(adjPosition);
            } else {
                adjPosition = adjPosition - adapterCount;
                int numFooters = getFooterLayoutCount();
                if (adjPosition < numFooters) {
                    return FOOTER_VIEW;
                } else {
                    return LOADING_VIEW;
                }
            }
        }
    }
```

在 getItemViewType方法中，为什么在这个方法里面呢，因为根据recyclerView.Adapter的执行逻，在渲染一个新的itemview时，会先调用getItemViewType询问我需要加载什么类型的ViewHolder在这里调用能更早的调用我们的加载更多的方法，当前，你在onBindViewHolder数据绑定方法中调也可以实现这个功能。接下来就很好理解了，当RecyclerView在渲染一个新的itemView时，就会调下

autoLoadMore(position);判断是不是需要调用加载更多回调，需要就调用，有关预加载的分析就OK，我们把加载更多的其他分析放到下一篇中进行分析。