链滴

BaseRecyclerViewAdapterHelper 开源项目之 BaseQuickAdapter 源码学习上拉加载的实现代码 (三)

作者：angels

原文链接：https://ld246.com/article/1496286552440

来源网站：链滴

version:2.8.5

我们在上一章中分析了实现预加载功能的代码，相信自己，你也可以，每个人都是创造者。

本章我将分析BaseRecyclerViewAdapterHelper 中 实现加载更多功能的代码。

首先我们先了解几个有关加载更多功能的方法，

第一步：打开上拉加载的开关

```
/**
    * Set the enabled state of load more.
    *
    * @param enable True if load more is enabled, false otherwise.
    */
   public void setEnableLoadMore(boolean enable) {
       int oldLoadMoreCount = getLoadMoreViewCount();
       mLoadMoreEnable = enable;
       int newLoadMoreCount = getLoadMoreViewCount();

       if (oldLoadMoreCount == 1) {
           if (newLoadMoreCount == 0) {
               notifyItemRemoved(getHeaderLayoutCount() + mData.size() + getFooterLayoutCou
t());
           }
       } else {
           if (newLoadMoreCount == 1) {
               mLoadMoreView.setLoadMoreStatus(LoadMoreView.STATUS_DEFAULT);
               notifyItemInserted(getHeaderLayoutCount() + mData.size() + getFooterLayoutCoun
());
           }
       }
   }
```

通过上面方法打开我们的上拉加载的开关。首先我们先看下以下两个变量的意思。

1、oldLoadMoreCount 代表在改变这个开关时我们是否处于显示上拉加载的view的状态，1表示处该状态。

2、newLoadMoreCount 代表我们当前是否可以开启上拉加载功能，同样，1表示可以。

这段代码很有意思

```
       if (oldLoadMoreCount == 1) {
           if (newLoadMoreCount == 0) {
               notifyItemRemoved(getHeaderLayoutCount() + mData.size() + getFooterLayoutCou
t());
           }
```

我们为什么要做插入这段代码呢，他的作用其实是这样的：加入当前处于显示加载更多view的状态，时你想关闭该开关，那我们第一件事要做什么呢，当然是移除加载更多view 了，这段代码的作用就这个。

反过来，我们现在要开启上拉加载。走的是这段代码

```
else {
        if (newLoadMoreCount == 1) {
            mLoadMoreView.setLoadMoreStatus(LoadMoreView.STATUS_DEFAULT);
            notifyItemInserted(getHeaderLayoutCount() + mData.size() + getFooterLayoutCoun
());
        }
    }
```

因为们的的loadMoreView一直是处于最底部的一个view，所以我们通过调用

notifyItemInserted(getHeaderLayoutCount() + mData.size() + getFooterLayoutCount());

告诉recycerlView将loadViewMore显示出来。

当我们同过上拉加载加载新的数据完成后，我们需要告诉BaseQuickAdapter你可以恢复正常状态了此时我们将用到以下方法：

//加载完成第一个if是防止我们错误的调用该方法。可以看到，方法内部帮我们调用了更新数据源的法。而且是局部更新。

```
/**
    * Refresh complete
    */
   public void loadMoreComplete() {
       if (getLoadMoreViewCount() == 0) {
           return;
       }
       mLoading = false;
       mLoadMoreView.setLoadMoreStatus(LoadMoreView.STATUS_DEFAULT);
       notifyItemChanged(getHeaderLayoutCount() + mData.size() + getFooterLayoutCount());
   }
```

我们看到这么一句恢复我们的loadMoreView为默认值，我们可以跟进去看下一下

```
       mLoadMoreView.setLoadMoreStatus(LoadMoreView.STATUS_DEFAULT);
```

他内部是重置了loadMoreStatus这个字段

```
 public void setLoadMoreStatus(int loadMoreStatus) {
       this.mLoadMoreStatus = loadMoreStatus;
   }
```

而这个字段是在什么时候用到呢，LoadMoreView的代码很少，可以看到

```
 public void convert(BaseViewHolder holder) {
       switch (mLoadMoreStatus) {
           case STATUS_LOADING:
               visibleLoading(holder, true);
               visibleLoadFail(holder, false);
               visibleLoadEnd(holder, false);
               break;
           case STATUS_FAIL:
               visibleLoading(holder, false);
```

```
            visibleLoadFail(holder, true);
            visibleLoadEnd(holder, false);
            break;
        case STATUS_END:
            visibleLoading(holder, false);
            visibleLoadFail(holder, false);
            visibleLoadEnd(holder, true);
            break;
        case STATUS_DEFAULT:
            visibleLoading(holder, false);
            visibleLoadFail(holder, false);
            visibleLoadEnd(holder, false);
            break;
    }
}
```

他是在一个convert方法中根据mLoadMoreStatus来改变loadMoreView的显示和隐藏，convert方大家应该很熟悉，参数holder其实就是我们的loadMoreView本身，那么loadMoreView.convert在被调用呢。

其实是在我们绑定数据时，如果判断当前viewholder时loadMore类型，就会调用。

```
@Override
  public void onBindViewHolder(K holder, int positions) {
    Log.d(TAG,"#test onBindViewHolder");
    int viewType = holder.getItemViewType();

    switch (viewType) {
        case 0:
            convert(holder, mData.get(holder.getLayoutPosition() - getHeaderLayoutCount()));
            break;
        case LOADING_VIEW:
            mLoadMoreView.convert(holder);
            break;
        case HEADER_VIEW:
            break;
        case EMPTY_VIEW:
            break;
        case FOOTER_VIEW:
            break;
        default:
            convert(holder, mData.get(holder.getLayoutPosition() - getHeaderLayoutCount()));
            break;
    }
}
```

很好理解，我们的loadMoreView是一直存在的。作为我们recyclerView的最后一个item，当加载到后一个item的时候，他就调用loadView的convert方法，方法内部根据我们当前是否应该显示loadVi w来做相应的操作。这样我们就理解了loadMore的隐藏和显示的逻辑了。后面还有两个方法也很好解，请看。

加载失败调用，可能你有需求在加载失败后要显示一个加载失败的view提示用户，而不是直接关闭loa MoreView。此时你可以调用该方法。

```
/**
```

```
     * Refresh failed
     */
    public void loadMoreFail() {
        if (getLoadMoreViewCount() == 0) {
            return;
        }
        mLoading = false;
        mLoadMoreView.setLoadMoreStatus(LoadMoreView.STATUS_FAIL);
        notifyItemChanged(getHeaderLayoutCount() + mData.size() + getFooterLayoutCount());
    }

/**
     * Refresh end, no more data
     *
     * @param gone if true gone the load more view
     */
    public void loadMoreEnd(boolean gone) {
        if (getLoadMoreViewCount() == 0) {
            return;
        }
        mLoading = false;
        mNextLoadEnable = false;
        mLoadMoreView.setLoadMoreEndGone(gone);
        if (gone) {
            notifyItemRemoved(getHeaderLayoutCount() + mData.size() + getFooterLayoutCount(
);
        } else {
            mLoadMoreView.setLoadMoreStatus(LoadMoreView.STATUS_END);
            notifyItemChanged(getHeaderLayoutCount() + mData.size() + getFooterLayoutCount()
;
        }
    }
```

之后就到我们关心的回调部分了。首先我们需要设置我们的回调监听器

```
public void setOnLoadMoreListener(RequestLoadMoreListener requestLoadMoreListener) {
        this.mRequestLoadMoreListener = requestLoadMoreListener;
        mNextLoadEnable = true;
        mLoadMoreEnable = true;
        mLoading = false;
    }
```

在设置监听器的时候，代码也帮我们做了一个字段的赋值操作。默认开启上拉加载，mLoading是表当前是否处于上拉加载中。

接下来你可能要问，那这个mrequestLoadMoreListener在什么时候被调用呢，其实这个也设计的比好，上一章我们分析了预加载功能，其实就在上次介绍的代码中。

```
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();
    }
}
```

如果你想关闭预加载，当时是mAutoLoadMoreSize =0 ,此时要调用最后一句代码，条件就变成了position