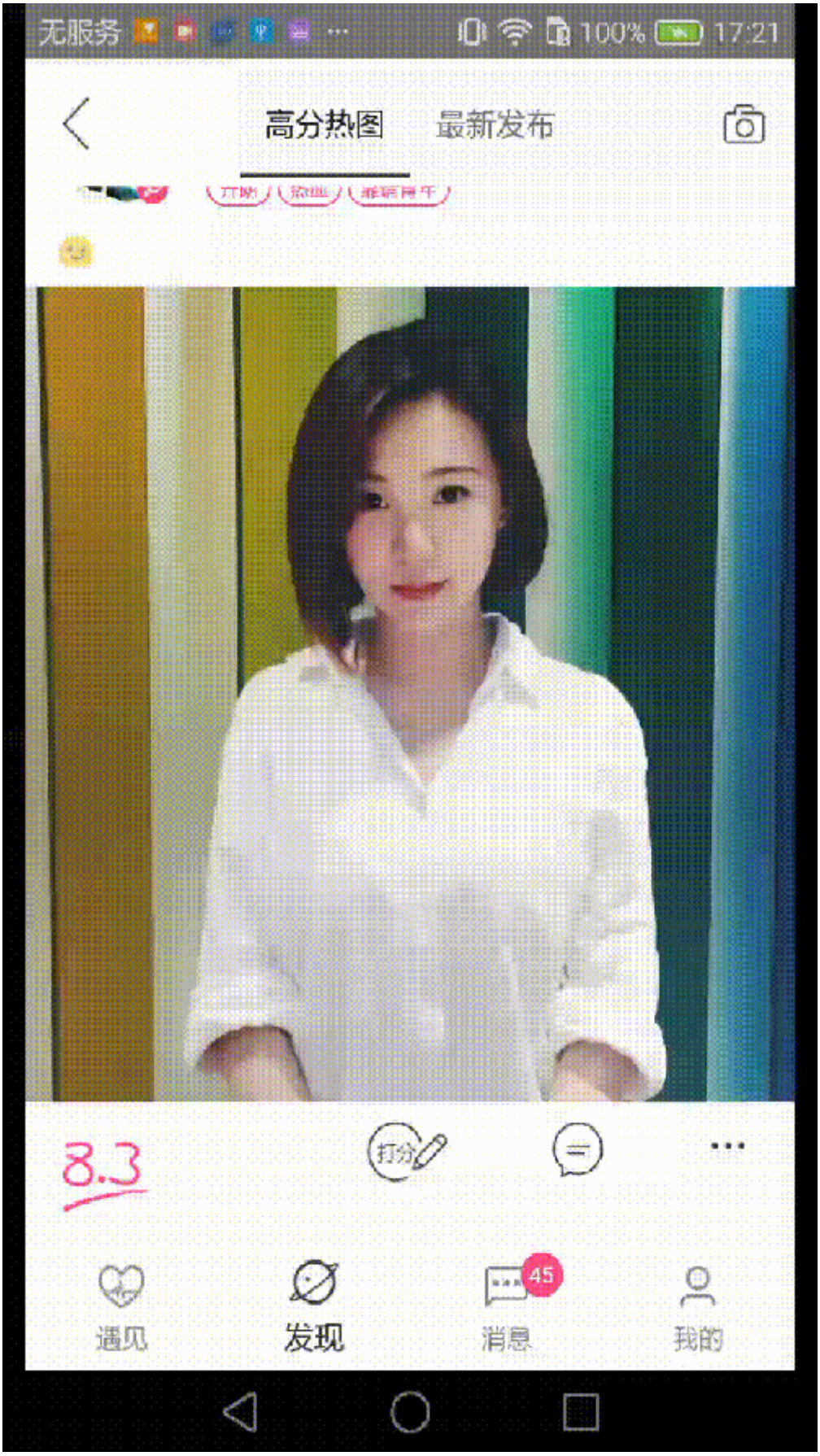
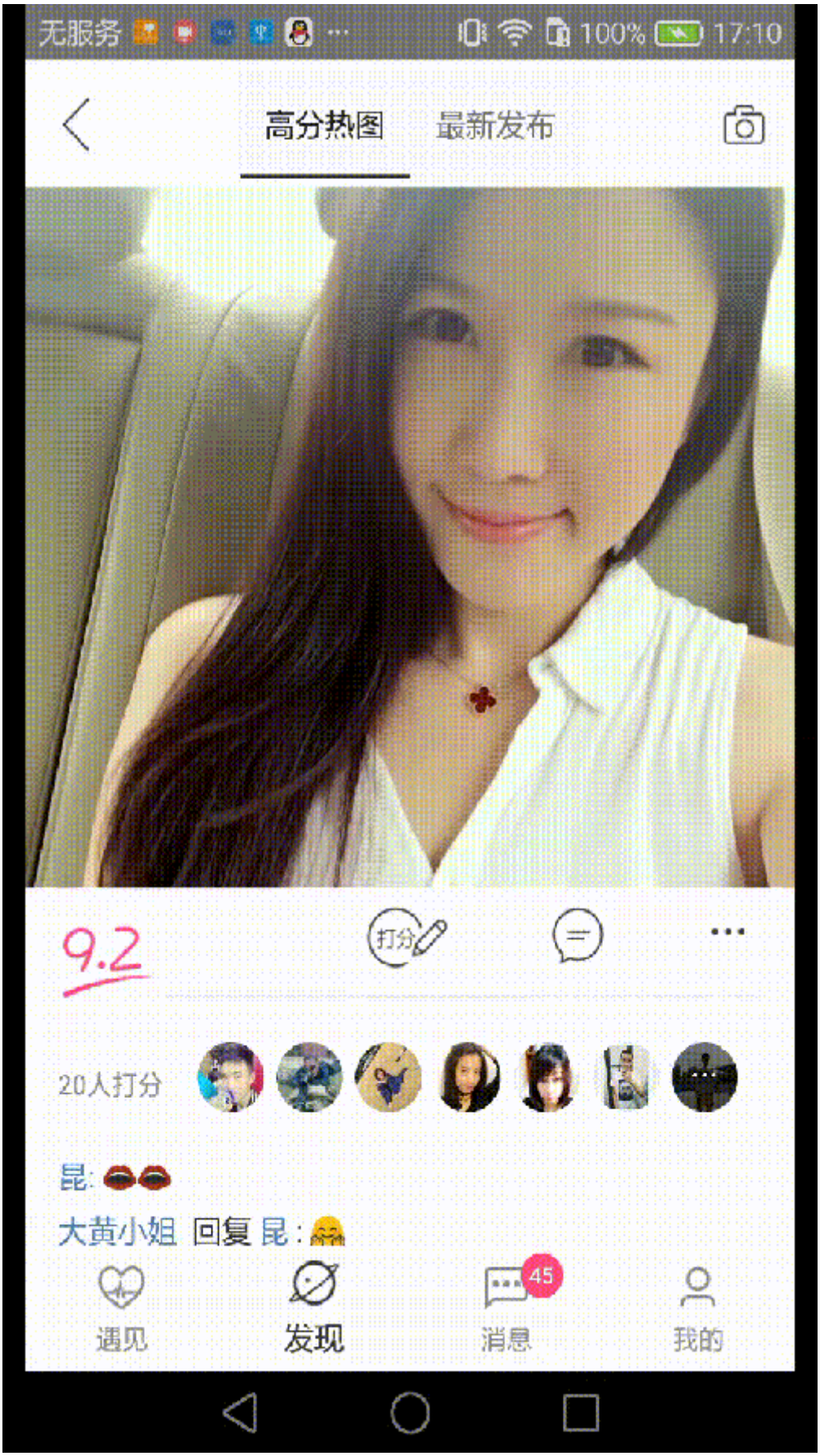
关键：public final void notifyItemChanged(int position, Object payload)

RecyclerView局部刷新大家都遇到过，有时候还说会遇见图片闪烁的问题。

优化之前的效果：



优化之后的效果：



如果想单独更新一个item，我们通常会这样做，代码如下：

```
mLRecyclerViewAdapter.notifyItemChanged(position);1
```

这里的position就是那个列表项的索引，调用这个方法可以更新一个Item的UI（当然，你要是直接调用notifyDataSetChanged()方法也可以，但这样会造成其他不需要更新的item也会刷新）。

即便如此，图片闪烁还是出现了，什么原因引起来的呢，这里猜测可以有如下几个原因：

1. 流传甚为广泛的一种说法，imageView的宽高不固定导致的（wrap_content）？
2. 是RecyclerView自带的更新动画效果导致的？
3. 是因为图片加载框架(glide 的 animte)的动画效果导致的？
4. getView中（RecyclerView中是onBindViewHolder）加载图片的时候，设置一个tag，当发现这个imageView的tag和之前的tag一致时就不加载。

这里我们不再对上面的原因进行具体的分析，针对上面可能引起闪烁的原因进行一一验证后的结果是令人感到失望的：都不是引起图片闪烁的根本原因。

那么怎么解决这个图片闪烁的问题呢？通过查看api，我们发现了另一个方法：

```
/**
 * Notify any registered observers that the item at <code>position</code> has changed with an
 * optional payload object.
 *
 * <p>This is an item change event, not a structural change event. It indicates that any
 * reflection of the data at <code>position</code> is out of date and should be updated.
 * The item at <code>position</code> retains the same identity.
 * </p>
 *
 * <p>
 * Client can optionally pass a payload for partial change. These payloads will be merged
 * and may be passed to adapter's {@link #onBindViewHolder(ViewHolder, int, List)} if the
 * item is already represented by a ViewHolder and it will be rebound to the same
 * ViewHolder. A notifyItemRangeChanged() with null payload will clear all existing
 * payloads on that item and prevent future payload until
 * {@link #onBindViewHolder(ViewHolder, int, List)} is called. Adapter should not assume
 * that the payload will always be passed to onBindViewHolder(), e.g. when the view is not
 * attached, the payload will be simply dropped.
 *
 * @param position Position of the item that has changed
 * @param payload Optional parameter, use null to identify a "full" update
 *
 * @see #notifyItemRangeChanged(int, int)
 */
public final void notifyItemChanged(int position, Object payload) {
    mObservable.notifyItemRangeChanged(position, 1, payload);
}
```

重点看payload参数介绍：

```
payload Optional parameter, use null to identify a "full" update1
```

翻译过来就是如果payload参数是null，那么就会来一个“完整的”更新，也就是说会全部更新。

我们再看一下`mLRecyclerViewAdapter.notifyItemChanged(position)`的源码：

```
/**
 * Notify any registered observers that the item at <code>position</code> has changed.
 * Equivalent to calling <code>notifyItemChanged(position, null);</code>.
 *
 * <p>This is an item change event, not a structural change event. It indicates that any
 * reflection of the data at <code>position</code> is out of date and should be updated.
 * The item at <code>position</code> retains the same identity.</p>
 *
 * @param position Position of the item that has changed
 *
 * @see #notifyItemRangeChanged(int, int)
 */
public final void notifyItemChanged(int position) {
    mObservable.notifyItemRangeChanged(position, 1);
}
```

从源码中看到，`notifyItemChanged(position)`调用了`notifyItemRangeChanged(int positionStart, int itemCount)`方法，源码如下：

```
public void notifyItemRangeChanged(int positionStart, int itemCount) {
    notifyItemRangeChanged(positionStart, itemCount, null);
}
```

`notifyItemRangeChanged(int positionStart, int itemCount)`方法最终还是调用了`notifyItemRangeChanged(int positionStart, int itemCount, Object payload)`方法，只是payload参数是null。

那么如果payload传一个不为null的参数，就可以实现对列表项中的具体控件更新了吗？我们通过代码验证下。

模拟更新一条数据：

```
ItemModel itemModel = mDataAdapter.getDataList().get(position);
itemModel.id = 100;
itemModel.title = "refresh item " + itemModel.id;
mDataAdapter.getDataList().set(position,itemModel);

//RecyclerView局部刷新
// notifyItemChanged(int position, Object payload) 其中的payload相当于一个标记，类型不限
mLRecyclerViewAdapter.notifyItemChanged(mLRecyclerViewAdapter.getAdapterPosition(false,position) , "jdsjlzx");
```

这里，我们将payload参数赋值为"jdsjlzx"，当然你也可以赋值为其他值，只要不空就行。

重写adapter中的`onBindViewHolder(RecyclerView.ViewHolder holder, int position, List payloads)`方法：

```
//局部刷新关键：带payload的这个onBindViewHolder方法必须实现
@Override
public void onBindViewHolder(RecyclerView.ViewHolder holder, int position, List payloads) {

    if (payloads.isEmpty()) {
        onBindViewHolder(holder,position);
    } else {

        //注意：payloads的size总是1
        String payload = (String)payloads.get(0);
        TLog.error("payload = " + payload);

        //需要更新的控件
        ItemModel itemModel = mDataList.get(position);
        ViewHolder viewHolder = (ViewHolder) holder;
        viewHolder.textView.setText(itemModel.title);

    }
}
```

如果payloads列表不是空的，如上图所示，你就可以在else代码块里面刷新你想更新的控件了（记得不需要更新的控件就不要写在这里了）。

总结

由此看来，RecyclerView做局部刷新还是非常容易的，其实就是使用好带payload参数的这个notifyItemChanged方法，以及重写带payload的这个onBindViewHolder方法，在onBindViewHolder中去刷新你想更新的控件即可。

PS：

拿朋友圈来说，我发一张照片，这就是一个item，但这个item里还要加上赞和评论。
当我有评论和赞要刷新时，我需要判断当前要改动的item是否是屏幕中的可见位置。如果是，通过调用带payload参数的这个notifyItemChanged方法更新item，就能达到只刷赞或者只刷评论，而不用重新加载照片（也就是图片闪烁的原因）的效果。

怎么判断当前position是位于屏幕中呢？下面给出参考代码：

```
private void doAnim(int position) {
        int firstItemPosition = layoutManager.findFirstVisibleItemPosition();
        if (position - firstItemPosition >= 0) {
            //得到要更新的item的view
            View view = mRecyclerView.getChildAt(position - firstItemPosition +
1);
            if (null != mRecyclerView.getChildViewHolder(view)) {
                ProductsViewHolder viewHolder = (ProductsViewHolder)
mRecyclerView.getChildViewHolder(view);

                //do something

            }

        }

    }123456789101112131415
```

上面代码同时也获取到了ViewHolder视图，有了ViewHolder，你还可以做其他操作哦（比如item动画效果）。