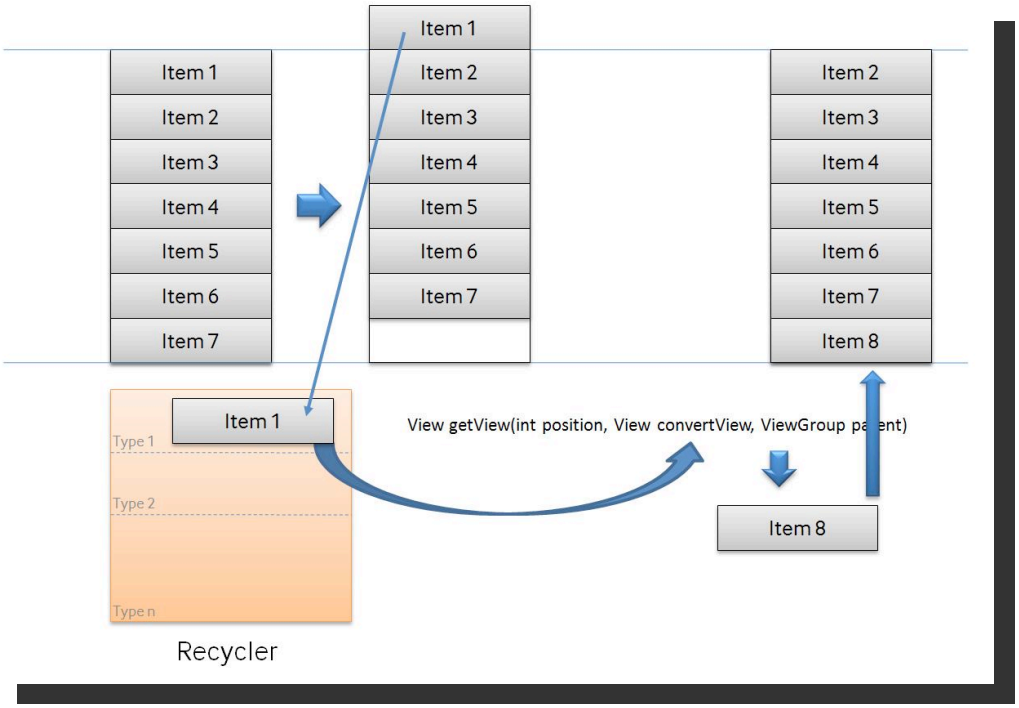
自定义 ListView 的选项显示

知识解析

ListView 原理

ListView 针对每个 item，要求 adapter “返回一个视图” (调用 adapter 上的 getView() 方法)，也就是说 ListView 在开始绘制的时候，系统首先调用 getCount（）函数，根据他的返回值得到 ListView 的长度，然后根据这个长度，调用 getView（）一行一行的绘制 ListView 的每一项。如果你的 getCount（）返回值是 0 的话，列表一行都不会显示，如果返回 1，就只显示一行。返回几则显示几行。



对于不在用户屏幕上显示的 ListView 选项，系统并不会创建这个选项。这是因为无论是在内存中保存大量的选项的 layout 中的组件，还是创建大量的 layout，均需要花费大量的资源（内存和时间），而此时对用户来说，在屏幕外面的选项并不可见，是“无用”的。因此，只需要创建在屏幕上能看到的那几个选项即可。当用户滑动 ListView 选项，把一些

选项滑出屏幕后，这些已经创建好的选项 layout 已经没用了，此时，它将会被放入 Recycler 中等待被系统回收。而正常情况下，下面又会有新的选项被滑进屏幕，此时也需要重新创建新的选项 layout 来放置选项内容。一方面，有选项要被回收，一方面，有选项要被创建，因此，比较好的方式是直接将要回收的选项交给新滑入到屏幕的选项使用。这就是 ListView 的一个提供性能的地方。这个在 Recycler 中的选项 layout 是通过 getView()的参数 convertView 传递给新的选项的。因此，可以判断 converView 是否为 null 就知道是否有从 Recycler 中的选项 layout 传递过来，如果有，就不需要再产生新的选项 layout 了，否则，才需要产生。

自定义 Adapter

前面我们在 ListView 中，使用的都是 android 提供的一些 layout 来显示 ListView 中选项的内容，它只能显示比较简单的内容，如果我们需要在 ListView 中显示复杂的内容，那么就需要对 ListView 中的 Item 进行定制。

如果要定义自己的 Adapter，定义一个 Adapter 的子类，覆盖其构造器，覆盖相关的 getView()/getCount()/getItemId()方法。

在 getView()方法中，将返回各个 Item 的视图，根据需要将不同的组件放到这个视图中，并返回。

对于复杂视图的 Item，以及选项较多的情况，应该使用这种方式来实现。

将这个自定义 adapter 初始化后，作为 ListView 的数据来源。

给 ListView 的各个 Item（选项）加上点击事件监听。

下面看一个例子。

```
class MyAdapter extends ArrayAdapter<ImageText> {
      List<ImageText> list;
      public MyAdapter(Context context, int resource,
List<ImageText> objects) {
          super(context, resource, objects);
          // TODO Auto-generated constructor stub
          this.list = objects;
```

```
        }
        @Override
        public int getCount() {
            // TODO Auto-generated method stub
            return list.size();
        }
        @Override
        public long getItemId(int position) {
            // TODO Auto-generated method stub
            return position;
        }
        @Override
        public View getView(int position, View convertView,
ViewGroup parent) {
            // TODO Auto-generated method stub
            View view = null;
            if (convertView == null) {
                // 自己创建一个 Item View
                LayoutInflater inflater = (LayoutInflater)
MainActivity.this.getSystemService(LAYOUT_INFLATER_SE
RVICE);
                view = inflater.inflate(R.layout.item,
parent, false);
            } else {
                view = convertView;
            }
            ImageView iv = (ImageView)
view.findViewById(R.id.imageView1);
            TextView tv = (TextView)
view.findViewById(R.id.textView1);
            ImageText it = list.get(position);
            iv.setImageResource(it.getImageId());
```

```
            tv.setText(it.getText());
            return view;
        }
    }
```

在 getView()方法中，我们对 convertView 进行判断，如果不为 null，则直接使用这个从 Recycler 中传递过来的 layout，否则，从资源文件中 inflate。

其中，ImageText 是封装了文本和图片数据的类，类定义如下：

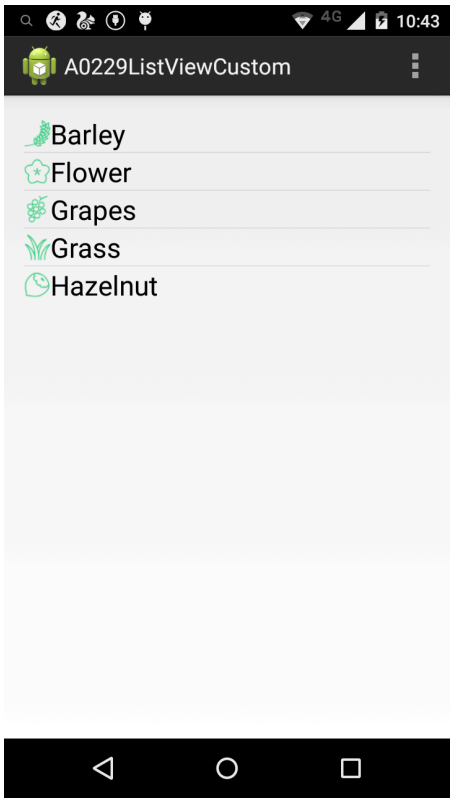
```
class ImageText {
        private int imageId;
        private String text;
        public ImageText(int imageId, String text) {
            this.setImageId(imageId);
            this.setText(text);
        }
        public int getImageId() {
            return imageId;
        }
        public void setImageId(int imageId) {
            this.imageId = imageId;
        }
        public String getText() {
            return text;
        }
        public void setText(String text) {
            this.text = text;
        }
    }
```

对应的 R.layout.item 文件如下：

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/and
roid"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >
    <ImageView
        android:layout_gravity="center"
        android:id="@+id/imageView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_launcher" />
    <TextView
        android:id="@+id/textView1"
        android:layout_gravity="center"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Large Text"
            android:textAppearance=
            "?android:attr/textAppearanceLarge" />
</LinearLayout>
```

功能演示



职业素质

在工作中我们可以自己定义一些 listView 的样式来满足开发的需求。比如我们实现以下比较复杂样式的 ListView 列表的，需要我们自定义 item 的布局，自定义适配器 Adapter，从而实现我们想要的效果。