使用 ViewHolder 模式

知识解析

在上一个自定义的 Adapter 中，在 getView()方法中每次都需要去从从 layout 中去使用 findViewById()去寻找对应的组件，这也是一项比较耗时的工作，尤其是当 layout 层次比较多比较复杂的时候。而采用 ViewHolder 模式，可以大大降低搜索的时间。它的基本原理是，将 layout 中的子组件保存到 ViewHoder 的属性中，然后将 ViewHoder 设置成对应 View 的 tag（通过 view 上的 setTag(Object)方法），后续只需要通过 getTag()取出 View 对应的 Tag，即可得到保存了它的子组件的 ViewHolder 对象，这样即避免了使用 findViewById()去搜索的时间。

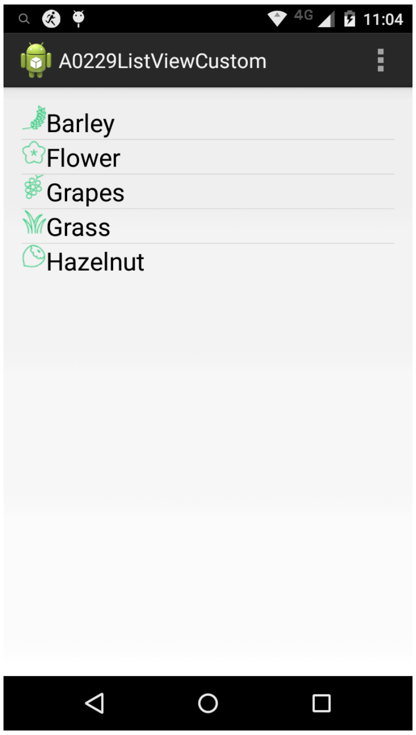
下面是一个使用 ViewHolder 的例子。

```
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;
            ViewHolder vh = 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);
                ImageView iv = (ImageView)
view.findViewById(R.id.imageView1);
                TextView tv = (TextView)
view.findViewById(R.id.textView1);
                vh = new ViewHolder();
                vh.vh_iv = iv;
                vh.vh_tv = tv;
                view.setTag(vh);
            }else{
                view = convertView;
                vh = (ViewHolder)view.getTag();
            }
            ImageText it = list.get(position);
            vh.vh_iv.setImageResource(it.getImageId());
            vh.vh_tv.setText(it.getText());
            return view;
        }
        class ViewHolder{
```

```
        TextView vh_tv;
        ImageView vh_iv;
    }
}
```

功能演示



实战操作

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        int[] imgs =
{R.drawable.barley,R.drawable.flower,R.drawable.grapes,R.drawable.gr
ass,R.drawable.hazelnut};
        String[] names =
{"Barley","Flower","Grapes","Grass","Hazelnut"};
        ArrayList<ImageText> list = new ArrayList<ImageText>();
        for(int i=0;i<imgs.length;i++){
            ImageText it = new ImageText(imgs[i],names[i]);
            list.add(it);
        }
MyAdapter adapter = new MyAdapter(this,0,list);
        ListView lv = (ListView) this.findViewById(R.id.listView1);
        lv.setAdapter(adapter);
    }
    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;}
   }
class MyAdapter extends ArrayAdapter<ImageText>{
        List<ImageText> list;
        public MyAdapter(Context context, int resource, List<ImageText>
objects) {
            super(context, resource, objects);
        this.list = objects;}

@Override
        public int getCount() {
            return list.size();
        }
        @Override
        public long getItemId(int position) {
            return position;
        }
        @Override
        public View getView(int position, View convertView, ViewGroup
parent) {
            View view = null;
            ViewHolder vh = null;
            if(convertView == null){
                //自己创建一个Item View
                LayoutInflater inflater = (LayoutInflater)
MainActivity.this.getSystemService(LAYOUT_INFLATER_SERVICE);
                view = inflater.inflate(R.layout.item, parent,false);
                ImageView iv = (ImageView)
view.findViewById(R.id.imageView1);
                TextView tv = (TextView)
view.findViewById(R.id.textView1);
                vh = new ViewHolder();
                vh.vh_iv = iv;
                vh.vh_tv = tv;
                view.setTag(vh);
            }else{
                view = convertView;
                vh = (ViewHolder)view.getTag();
            }
ImageText it = list.get(position);
            vh.vh_iv.setImageResource(it.getImageId());
            vh.vh_tv.setText(it.getText());
            return view;
        }
        class ViewHolder{
            TextView vh_tv;
            ImageView vh_iv;
        }
    }
}
```

职业素质

ViewHolder通常出现在适配器里，为的是listview滚动的时候快速设置值，而不必每次都重新创建很多对象，从而提升性能。

在 android 开发中 Listview 是一个很重要的组件，它以列表的形式根据数据的长自适应展示具体内容,用户可以自由的定义 listview 每一列的布局，但当 listview 有大量的数据需要加载的时候，会占据大量内存，影响性能，这时候就需要按需填充并重新使用 view 来减少对象的创建。