**ListView优化**

Adapter的作用就是ListView界面与数据之间的桥梁，当列表里的每一项显示到页面时，都会调用Adapter的getView方法返回一个View。想过没有？ 在我们的列表有1000000项时会是什么样的？是不是会占用极大的系统资源？

List.xml：

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

<LinearLayout

  xmlns:android=*"http://schemas.android.com/apk/res/android"*

  android:layout\_width=*"fill\_parent"*

  android:layout\_height=*"fill\_parent"*

android:orientation=*"vertical"*>

<LinearLayout android:id=*"@+id/linearLayout1"* android:layout\_width=*"fill\_parent"* android:layout\_height=*"wrap\_content"* android:orientation=*"horizontal"*>

 <ImageView android:layout\_height=*"wrap\_content"*

android:layout\_width=*"wrap\_content"*

android:src=*"@drawable/icon"*

android:id=*"@+id/imageView1"*

android:layout\_margin=*"5px"*>

</ImageView>

<TextView android:text=*"TextView"*

android:id=*"@+id/textView1"*

android:layout\_width=*"wrap\_content"* android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center"*

android:textSize=*"20px"*>

</TextView>

</LinearLayout>

</LinearLayout>

|  |
| --- |
| 1。最简单的方法，最慢且最不实用 |

    @Override

**public** View getView(**int** position, View convertView, ViewGroup parent) {

       // **TODO** Auto-generated method stub

       //找到R.layout.list布局文件

       layout\_Adapter = (LinearLayout)**this**.layoutInflater.inflate(R.layout.list, **null**);

       imageView = (ImageView)layout\_Adapter.findViewById(R.id.imageView1);

       textView = (TextView)layout\_Adapter.findViewById(R.id.textView1);

       imageView.setImageResource(R.drawable.icon);

       textView.setText(title[position]);

**return** layout\_Adapter;

    }

2。利用convertview回收视图，效率提高200%。

@Override

**public** View getView(**int** position, View convertView, ViewGroup parent) {

       // **TODO** Auto-generated method stub

**if**(convertView == **null**){

           convertView = (LinearLayout)**this**.layoutInflater.inflate(R.layout.list, **null**);

       }

       imageView = (ImageView)convertView.findViewById(R.id.imageView1);

       textView = (TextView)convertView.findViewById(R.id.textView1);

       imageView.setImageResource(R.drawable.icon);

       textView.setText(title[position]);

**return** convertView;

    }

上面的代码是不是好了很多？系统将会减少创建很多View。性能得到了很大的提升。

|  |
| --- |
| 3。利用viewholder模式，效率在提高50% |

**public** View getView(**int** position, View convertView, ViewGroup parent) {

       // **TODO** Auto-generated method stub

       ViewHolder holder;

**if**(convertView == **null**){

           convertView = (LinearLayout)**this**.layoutInflater.inflate(R.layout.*list*, **null**);

           holder = **new** ViewHolder();

           holder.tv = (TextView)convertView.findViewById(R.id.*textView1*);

           holder.im = (ImageView)convertView.findViewById(R.id.*imageView1*);

           convertView.setTag(holder);

       }**else**{

           holder = (ViewHolder)convertView.getTag();

           Log.*d*("list", "convertView....");

       }

       holder.im.setImageResource(R.drawable.*icon*);

       holder.tv.setText(title[position]);

**return** convertView;

    }

**static class** ViewHolder{

       TextView tv;

       ImageView im;

    }

这种方法为View设置了Tag属性，通过setTag()方法将View的ImageView、TextView组件的对象添加到Tag 中，当回收利用时直接用getTag()方法取出，不用重新获取组件。

三种方式加载效率对比如下图所示：