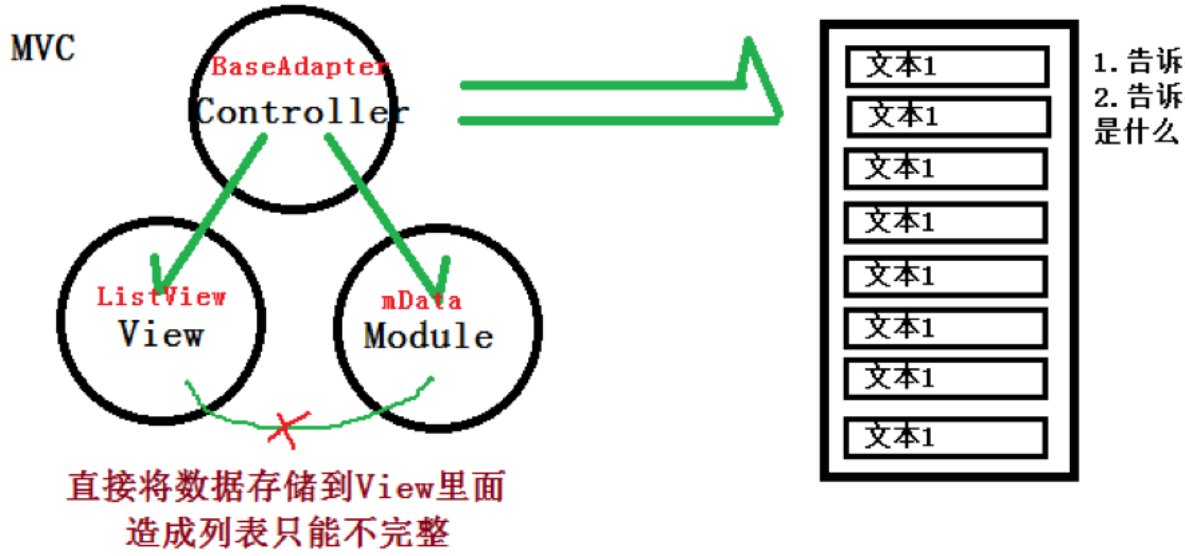
1. 常用适配器BaseAdapter方法详解



1. BaseAdapter 默认提供了四个方法:
    - getCount() 必须 告诉列表默认显示多少条数据
    - getView(int position, View convertView, ViewGroup parent) 必须 告诉列表每项怎么显示
    - getItem(int position) 可选
    - getItemId(int position) 可选
2. 细节:
    - getCount() 只有在setAdapter()与notifyDataSetChanged();调用时才调用
    - getView(int position, View convertView, ViewGroup parent) 只有getCount() 不为0的时候才调用
    - 每个View都可以携带一个对象 这样对象就可以不用作为全局变量了 只要有View就有对象
    - 子项里面有需要绑定的子控件可以创建一个ViewHolder类来封装

笔记

1. 每一个View 都可以获取Context. --->v.getContext();
2. FILL_PARENT(安卓2.3版本以前的写法)==MATCH_PARENT
3. 通过布局文件直接转换成View LayoutInflater.from(mContext).inflate(R.layout.xxx,root);

```
需求:点击某个按钮 动态添加一些控件
1. new TextView()
2. 当要添加的控件很多 布局很复杂的情况下 通过先定义好容器布局(定好控件的位置 样式)
```

---

1. 通过某个容器containerLl找到它的子控件 containerLl.findViewById(R.id.xxx);
2. 安卓已经帮我们定义好了一些子项布局 android.R.layout.simple_list_item_1 (android.R.id.text1) android.R.layout.simple_list_item_2 (android.R.id.text1 android.R.id.text2)
3. getView里面有个参数叫convertView ,可以认为它是一个缓存的View 特点:列表滑动显示一个子项布局的时候 就会调用getView方法,当执行getView的时候发现上面有一个ItemView被隐藏了 此时convertView 就是那个被隐藏的ItemView(不管是上拉还是下拉)
4. Tag可以绑定一个对象 在任何地方都可以取出该绑定的对象

第一版本的代码

```
public class MyAdapter extends BaseAdapter {

    private ArrayList<String> mDatas;

    public MyAdapter(ArrayList<String> datas) {
        mDatas=datas;
    }

    @Override
    public int getCount() {
        Log.v("520it", "getCount");
        return mDatas.size();
    }


    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        Log.v("520it", "getView position="+position);
        //  TextView tv=new TextView(parent.getContext());
        //  tv.setLayoutParams(new AbsListView.LayoutParams(
        //  AbsListView.LayoutParams.MATCH_PARENT,
        //  AbsListView.LayoutParams.WRAP_CONTENT));
        //  tv.setTextSize(30);
        //  tv.setText(mDatas.get(position));

        //  LayoutInflater 布局充气的一个类  就是可以将一个布局转换成一个View
        LayoutInflater layoutInflater = LayoutInflater.from(parent.getContext());
        //  root 询问用户是否需要添加到某个容器里面去
        View view = layoutInflater.inflate(R.layout.lv_item_layout, null);
        TextView tv=(TextView) view.findViewById(R.id.tv);
        tv.setText(mDatas.get(position));
        return view;
    }

    @Override
    public Object getItem(int position) {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public long getItemId(int position) {
        // TODO Auto-generated method stub
        return 0;
    }


}
```

子项布局

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="24sp"
    android:textColor="#F00"
    android:gravity="center"
    android:id="@+id/tv" />
```

Tag属性的使用

```
public class MainActivity extends Activity implements OnClickListener {

    private Button mBtn;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mBtn =(Button) findViewById(R.id.btn);
        mBtn.setOnClickListener(this);
        //往Button里面绑定一个对象
        mBtn.setTag("haha");
    }

    //View v 传进来的View就是那个被点击的View
    @Override
    public void onClick(View v) {
        //取出绑定的对象
        String tag = (String) v.getTag();
        Toast.makeText(this, tag, 0).show();
    }

}
```

第二版本代码

```
public class MyAdapter extends BaseAdapter {

    private ArrayList<String> mDatas;

    public MyAdapter(ArrayList<String> datas) {
        mDatas = datas;
    }

    @Override
    public int getCount() {
        // Log.v("520it", "getCount");
        return mDatas.size();
    }

    class ViewHolder{
        TextView tv1;
        TextView tv2;
    }

    /**
     * position 索引 convertView parent 每个 ItemView里面的容器 返回的View直接添加到容器中来
     * convertView 缓存的ItemView
     * @return View 就是每个ItemView要显示的内容
     * */
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        Log.v("520it", "position="+position+ "   convertView=" + convertView);
        ViewHolder holder=null;
        if (convertView==null) {
            LayoutInflater layoutInflater = LayoutInflater
            .from(parent.getContext());
            convertView = layoutInflater.inflate(android.R.layout.simple_list_item_2,
            null);
            //在convertView刚开始创建的时候 就初始化ViewHolder
            holder=new ViewHolder();

            holder.tv1 = (TextView) convertView.findViewById(android.R.id.text1);
            holder.tv2 = (TextView) convertView.findViewById(android.R.id.text2);

            convertView.setTag(holder);
        }else {
            holder=(ViewHolder) convertView.getTag();
        }

        holder.tv1.setText(mDatas.get(position));
        holder.tv2.setText(mDatas.get(position));

        return convertView;
    }

    @Override
    public Object getItem(int position) {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public long getItemId(int position) {
        // TODO Auto-generated method stub
        return 0;
    }

}
```

2. 文字适配器ArrayAdapter

ArrayAdapter是 BaseAdapter的子类，其填充数据的方式是在ArrayAdapter创建的时候将数据填充进去。

细节：在ArrayAdapter填充数据后 如果想对数据进行添加 或者修改 需要调用adapter.notifyDataSetChanged();

代码

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ListView lv= (ListView) findViewById(R.id.lv);

        ArrayList<String> datas=new ArrayList<String>();
        for (int i = 0; i < 20; i++) {
            datas.add("测试 "+i);
        }

        //  ArrayAdapter    当一个列表里面只有显示一个TextView 就可以直接使用它
        //  resource 子项布局
        //  textViewResourceId 就是说要将每一项数据绑定到哪个TextView下 这里提供一个TextView的id
        //  datas 就是列表数据
        ArrayAdapter<String> adapter=new ArrayAdapter<String>(
        this,
        android.R.layout.simple_list_item_1,
        android.R.id.text1,
        datas);
        lv.setAdapter(adapter);
    }

}
```

3. SimpleAdapter适配器

SimpleAdapter是 BaseAdapter的子类，其填充数据的方式是在SimpleAdapter创建的时候将数据填充进去。

原理：SimpleAdapter首先获取datas里面的数据，通过position找到找到每项的Map<String,String> 通过倒数第二个参数获取每项的实际值，将其绑定到子项的子控件里面去。

代码

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ListView lv = (ListView) findViewById(R.id.lv);
        //  什么时候使用SimpleAdapter都可以
        //  1. datas代表整个列表的数据
        ArrayList<HashMap<String, Object>> datas=
        new ArrayList<HashMap<String,Object>>();
        for (int i = 1; i < 7; i++) {
            //2.data代表初始化每一项的数据
            HashMap<String, Object> data=new HashMap<String, Object>();
            data.put("图片", R.drawable.a2);
            data.put("文字", "测试"+i);
            datas.add(data);
        }
        //3.datas 代表数据已经有了
        //4.R.layout.lv_item_layout 每一项的布局已经准备好了 这里给的是一个容器
        //5.想对容器里面每一个控件进行数据绑定
        //5.1  找到要绑定值的控件id   new int[]{R.id.iv,R.id.tv}
        //5.2 循环遍历datas列表 找到里面的HashMap 通过new String[]{"图片","文字"} 找到对应的值
        //5.3 将控件与数据进行绑定
        SimpleAdapter adapter=new SimpleAdapter(
                                    this,
                                    datas,
                                    R.layout.lv_item_layout,
                                    new String[]{"图片","文字"},
                                    new int[]{R.id.iv,R.id.tv});
        lv.setAdapter(adapter);
    }

}
```

4. 列表样式&列表监听器

1. 设置listview 的Item之间的横线 颜色和高度 android:divider=“#8E8E8E” android:dividerHeight=“1px”

2. scrollbars属性，作用是隐藏listView的滚动条 android:scrollbars="none“

3. 当ListView 设置背景后 拖动发现出现背景不见了 android:cacheColorHint=”#0000”

    1. 当有背景滑动后出现黑屏 androidScrollingcache="false"
    2. listview的item设置背景后上下滑动 出现上边和下边有黑色的阴影 android:fadingEdge="none“
    3. 设置 ListView 点击子项的时候出现的Item颜色值 android:listSelector=“#0000”

监听器

1. onScrollListener
2. onItemClickListener

创建一个简单列表（模拟数据从网络中来的做法）

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    //1.控件已经装载完毕了
    ListView lv = (ListView) findViewById(R.id.lv);
    adapter=new MyAdapter(this);
    //系统内部自动帮你回调了多次 getCount()   getView()
    lv.setAdapter(adapter);

    //2.数据是从后台服务器发送回来(延迟)
    ArrayList<String> datas=new ArrayList<String>();
    for (int i = 0; i < 20; i++) {
    datas.add("测试"+i);
    }
    //这个地方只是简单的setDatas() 并没有调用getCount()/getView()
    //上面setAdapter导致的多次调用getCount()/getView()可能还没调用完 此时代码执行很快
    //导致了我们setDatas的时候数据进去了 并且刷新的ListView
    adapter.setDatas(datas);
    //调用了该方法 系统就会再次调用getCount()/getView()
    adapter.notifyDataSetChanged();
}
```

Adapter的写法

```
public class MyAdapter extends BaseAdapter {

    private ArrayList<String> mDatas;
    private LayoutInflater mInflater;

    public MyAdapter(Context c) {
        mInflater = LayoutInflater.from(c);
    }

    public void setDatas(ArrayList<String> datas) {
        mDatas=datas;
    }

    @Override
    public int getCount() {
        return mDatas!=null?mDatas.size():0;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        TextView tv=null;
        if (convertView==null) {
        convertView = mInflater.inflate(android.R.layout.simple_list_item_1, null);
        tv=(TextView) convertView.findViewById(android.R.id.text1);
        convertView.setTag(tv);
        }else {
        tv=(TextView) convertView.getTag();
        }
        tv.setText(mDatas.get(position));
        return convertView;
    }

    @Override
    public Object getItem(int position) {
        return mDatas!=null?mDatas.get(position):null;
    }

    @Override
    public long getItemId(int position) {
        return 5;
    }

}
```

监听器代码

```
//lv的滚动监听器
lv.setOnScrollListener(new OnScrollListener() {

    /**
     * 滑动的状态改变的时候调用的
     * scrollState 滚动的状态
     *  SCROLL_STATE_FLING  轻扫
     *  SCROLL_STATE_IDLE   什么都不动的情况下
     *  SCROLL_STATE_TOUCH_SCROLL  拖拽
     * */
    @Override
    public void onScrollStateChanged(AbsListView view, int scrollState) {
        switch (scrollState) {
            case OnScrollListener.SCROLL_STATE_FLING:
                Log.v("520it", "SCROLL_STATE_FLING");
                break;
            case OnScrollListener.SCROLL_STATE_IDLE:
                Log.v("520it", "SCROLL_STATE_IDLE");
                break;
            case OnScrollListener.SCROLL_STATE_TOUCH_SCROLL:
                Log.v("520it", "SCROLL_STATE_TOUCH_SCROLL");
                break;
        }
    }


    /**
     *  滚动的时候被调用
     *  只要列表滚动 就会毁掉该方法
     * */
    @Override
    public void onScroll(AbsListView view, int firstVisibleItem,
        int visibleItemCount, int totalItemCount) {
        //  Log.v("520it", "onScroll  firstItem="+firstVisibleItem
        //  +"   visibleCount="+visibleItemCount
        //  +" totalCount="+totalItemCount);
    }
});

//lv的点击事件
lv.setOnItemClickListener(new OnItemClickListener() {

    /**
     * id 就是BaseAdapter的getItemId 返回的值
     *
     * */
    @Override
    public void onItemClick(AdapterView<?> parent, View view,
    int position, long id) {
        String data = (String) adapter.getItem(position);
        Toast.makeText(MainActivity.this, data, 0).show();
    }
});
```

5. GridView与Spinner布局

```
<GridView
        android:id="@+id/gv"
        android:numColumns="4"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:listSelector="#0000"  />
```

```
<Spinner
        android:id="@+id/sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:spinnerMode="dialog"  />
```

6. ExpandableListView可折叠的列表

6.1创建适配器 实现4个方法：getGroupCount()/ getGroupView() getChildCount()/ getChildView()

```
public class MyAdapter extends BaseExpandableListAdapter {

    private ArrayList<String> mGroupNames;
    private HashMap<String, ArrayList<String>> mChildNames;

    public MyAdapter(ArrayList<String> groupNames,
    HashMap<String, ArrayList<String>> childNames) {
        mGroupNames=groupNames;
        mChildNames=childNames;
    }

    @Override
    public int getGroupCount() {
        return mGroupNames.size();
    }

    @Override
    public View getGroupView(int groupPosition, boolean isExpanded,
    View convertView, ViewGroup parent) {
        LayoutInflater inflater = LayoutInflater.from(parent.getContext());
        View itemView = inflater.inflate(android.R.layout.simple_list_item_1, null);
        TextView tv = (TextView) itemView.findViewById(android.R.id.text1);
        tv.setText(mGroupNames.get(groupPosition));
        return itemView;
    }

    @Override
    public int getChildrenCount(int groupPosition) {
        //1.获取组
        String groupName = mGroupNames.get(groupPosition);
        //2.根据组名获取子列表
        ArrayList<String> childNames = mChildNames.get(groupName);
        return childNames.size();
    }

    @Override
    public View getChildView(int groupPosition, int childPosition,
    boolean isLastChild, View convertView, ViewGroup parent) {
        LayoutInflater inflater = LayoutInflater.from(parent.getContext());
        View itemView = inflater.inflate(android.R.layout.simple_list_item_1, null);
        //1.控件已经被获取了
        TextView tv = (TextView) itemView.findViewById(android.R.id.text1);
        //2.获取子视图的数据
        //3.获取组
        String groupName = mGroupNames.get(groupPosition);
        //4.根据组名获取子列表
        ArrayList<String> childNames = mChildNames.get(groupName);
        //5.从子列表里面获取某个Item
        String text = childNames.get(childPosition);
        tv.setText(text);
        return itemView;
    }


}
```

6.2在Activity的onCreate中创建数据 并将数据绑定到列表中

6.3设置列表监听器 （组折叠/展开/点击监听器 子视图点击的监听器）

```
public class MainActivity extends Activity {

    private ExpandableListView mExpLv;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mExpLv=(ExpandableListView) findViewById(R.id.explv);

        //1.组的数据
        ArrayList<String> groupNames=new ArrayList<String>();
        for (int i = 0; i < 3; i++) {
            groupNames.add("组 "+i);
        }
        //2.成员的数据
        HashMap<String, ArrayList<String>> childNames=
        new HashMap<String, ArrayList<String>>();
        for (int i = 0; i < groupNames.size(); i++) {
            ArrayList<String> children=new ArrayList<String>();
            for (int j = 0; j < 5; j++) {
                children.add("成员"+j);
            }
            //遍历下组名 key就是组名 值就时一个成员队列
            childNames.put(groupNames.get(i), children);
        }

        MyAdapter adapter=new MyAdapter(groupNames,childNames);
        mExpLv.setAdapter(adapter);


        //为ExpandableListView的组设置折叠监听器
        //  mExpLv.setOnGroupCollapseListener(onGroupCollapseListener)
        //  mExpLv.setOnGroupExpandListener(onGroupExpandListener)
        //  mExpLv.setOnGroupClickListener(onGroupClickListener)
        //  mExpLv.setOnChildClickListener(onChildClickListener)

        mExpLv.setOnGroupExpandListener(new OnGroupExpandListener() {

        @Override
        public void onGroupExpand(int groupPosition) {
                Toast.makeText(MainActivity.this, groupPosition+" 被展开了", 0).show();
            }
        });


    }
}
```