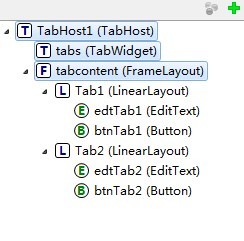
开发course笔记

1. 图放的位置不一样，有放到drawable-hdpi中的，有放到drawable-mdpi里面，但是在xml里面引用的时候都是放android:background=*"@drawable/share\_button"*里。
2. 用要实现线程功能必须要用handler来发送一个信息。
3. 加了<include layout=*"@layout/main\_topbar"* />自定义view控件以后，就不能预览了。
4. Android利用ViewFlipper实现屏幕切换动画效果
5. GridView(网格视图)是按照行列的方式来显示内容的，一般用于显示图片，图片等内容，比如实现九宫格图，用GridView是首选，也是最简单的
6. 多了一个大括号，可能引起变量不被识别或者其他语法错误，多了几个还不容易变辨别。
7. [TabHost,TabWidget](http://blog.csdn.net/hellogv/article/details/5958565)

Android的Tab控件，Tab控件可以达到分页的效果，让一个屏幕的内容尽量丰富，当然也会增加开发的复杂程度，在有必要的时候再使用。Android的Tab控件使用起来有点奇怪，必须包含和按照以下的顺序：



TabHost控件->TabWidget(必须命名为tabs)->FrameLayout(必须命名为tabcontent)。

1. 更新UI线程有三种方法

方法一：用Handler

1、主线程中定义Handler：

Handler mHandler = new Handler() {

@Override

public void handleMessage(Message msg) {

super.handleMessage(msg);

switch (msg.what) {

case 0:

//完成主界面更新,拿到数据

String data = (String)msg.obj;

updateWeather();

textView.setText(data);

break;

default:

break;

}

}

};

2、子线程发消息，通知Handler完成UI更新：

private void updateWeather() {

new Thread(new Runnable(){

@Override

public void run() {

//耗时操作，完成之后发送消息给Handler，完成UI更新；

mHandler.sendEmptyMessage(0);

//需要数据传递，用下面方法；

Message msg =new Message();

msg.obj = "数据";//可以是基本类型，可以是对象，可以是List、map等；

mHandler.sendMessage(msg);

}

}).start();

}

方法一的Handler对象必须定义在主线程中，如是多个类直接互相调用，就不是很方便，需要传递content对象或通过接口调用；

在子线程中通过runOnUiThread()方法更新UI：

方法二：用runOnUiThread更新

new Thread() {

public void run() {

//这儿是耗时操作，完成之后更新UI；

runOnUiThread(new Runnable(){

@Override

public void run() {

//更新UI

imageView.setImageBitmap(bitmap);

}

});

}

}.start();

如果在非上下文类中（Activity），可以通过传递上下文实现调用；

Activity activity = (Activity) imageView.getContext();

activity.runOnUiThread(new Runnable() {

@Override

public void run() {

imageView.setImageBitmap(bitmap);

}

});

这种方法使用比较灵活，但如果Thread定义在其他地方，需要传递Activity对象；

方法三：View.post(Runnable r)

imageView.post(new Runnable(){

@Override

public void run() {

imageView.setImageBitmap(bitmap);

}

});

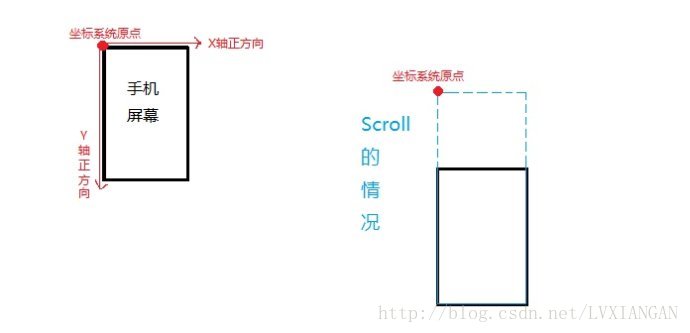
这种方法更简单，但需要传递要更新的View过去；

1. 手势动作做出了定义

* 按下（onDown）： 刚刚手指接触到触摸屏的那一刹那，就是触的那一下。
* 抛掷（onFling）： 手指在触摸屏上迅速移动，并松开的动作。
* 长按（onLongPress）： 手指按在持续一段时间，并且没有松开。
* 滚动（onScroll）： 手指在触摸屏上滑动。
* 按住（onShowPress）： 手指按在触摸屏上，它的时间范围在按下起效，在长按之前。
* 抬起（onSingleTapUp）：手指离开触摸屏的那一刹那。

除了这些定义之外，鄙人也总结了一点算是经验的经验吧，在这里和大家分享一下。

* 任何手势动作都会先执行一次按下（onDown）动作。
* 长按（onLongPress）动作前一定会执行一次按住（onShowPress）动作。
* 按住（onShowPress）动作和按下（onDown）动作之后都会执行一次抬起（onSingleTapUp）动作。
* 长按（onLongPress）、滚动（onScroll）和抛掷（onFling）动作之后都不会执行抬起（onSingleTapUp）动作。

1. 

（一）首先明确一下 android 中的坐标系统 ：

     屏幕的左上角是**坐标系统原点**（0,0）

     原点向右延伸是X轴正方向，原点向下延伸是Y轴正方向

（二）关于Scroll： 屏幕显示的内容很多时，会有超出一屏的情况，于是就产生了Scroll的概念。

      在View类中有个方法：

**getScrollY**()  英文原文描述是：

      Return the scrolled top position of this view. This is the top edge of the displayed part of your view....

      其实理解起来仍然就是：就是这个view相对于“坐标系统原点”(见上图)在Y轴上的偏移量.(getScrollX同理)

     【哇哈，有了图就是好说明啊~ 省了多少语言描述啊，而且还不一定能说清楚~】

android系统中的每个View的子类都具有下面三个和TouchEvent处理密切相关的方法：

1）public boolean dispatchTouchEvent(MotionEvent ev)  这个方法用来分发TouchEvent

2）public boolean onInterceptTouchEvent(MotionEvent ev) 这个方法用来拦截TouchEvent

3）public boolean onTouchEvent(MotionEvent ev) 这个方法用来处理TouchEvent

1. android系统中的每个View的子类都具有下面三个和TouchEvent处理密切相关的方法

1）public boolean dispatchTouchEvent(MotionEvent ev)  这个方法用来分发TouchEvent

2）public boolean onInterceptTouchEvent(MotionEvent ev) 这个方法用来拦截TouchEvent

3）public boolean onTouchEvent(MotionEvent ev) 这个方法用来处理TouchEvent

12.