河北师范大学软件学院
Software College of Hebei Normal University

Android 基础开发

第二章 第四节
通知消息和菜单

教学目标

- 掌握Android的三种提醒类型
- 掌握Android中菜单的使用

目录

Android中的提示信息

- Android中系统经常会向用户反馈一些提示信息。

Android中的提示信息的形式

- Android中系统经常会向用户反馈一些提示信息。
- Android中的提示信息，根据其展现形式可以分为以下几类：
  - 对话框（模式对话框）：以弹出层形式强制用户作出响应；
  - Toast：显示提示信息，显示时间较短；
  - Notification：在状态栏显示通知信息，除非用户查看信息或删除信息，否则一直在状态栏显示。

对话框简介

- 对话框：以弹出层形式显示内容的视图控件，强制用户做出响应，属于模式对话框。

对话框简介

- 对话框：以弹出层形式显示内容的视图控件。
- 对话框的应用也很广泛，很多应用的"新版本"信息、退出时提示、列表项目附加信息等等都是使用对话框形式展现的。
- 在Android中使用AlertDialog类来实现基本对话框的创建。
  - AlertDialog是最常用的对话框类；
  - DatePickerDialog、ProgressDialog、TimePickerDialog是子类，方便创建一些特殊的对话框。

使用AlertDialog对话框

- 在Android中使用对话框有两种方法：
  - 直接在Activity的onCreate( )回调方法中创建对话框；
  - 使用Activity的onCreateDialog( )回调方法创建对话框。
- 使用对话框的基本流程：
  1. 创建AlertDialog.Builder对象（AlertDialog的创建器）；
  2. 调用AlertDialog.Builder对象的方法为对话框设置属性（标题、图标、内容、按钮等）；
  3. 使用AlertDialog.Builder对象的create( )方法创建对话框；
  4. 使用AlertDialog.Builder对象的show( )方法显示对话框。

简单对话框实现

- 实现一个简单的对话框。
  - 创建简单的带有两个按钮的对话框；
  - 在onCreate( )回调方法中创建对话框；
  - 提示：
    - 退出当前应用使用当前Activity的finish( )方法。

Step0：添加按钮并绑定单击事件

- 在Activity中为该按钮绑定单击事件；

```
// 为"退出"按钮绑定单击事件
Button Btn = findViewById(R.id.Btn_Main_button);
Btn.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        // ......
    }
});
```

Step1：创建对话框创建器

- 使用AlertDialog.Builder类创建创建器；

```
AlertDialog.Builder AdBuilder
     = new AlertDialog.Builder(MyActivity.this);
```

Step2：设置对话框属性

- 设置基本属性；

```
AdBuilder.setTitle("温馨提示");
AdBuilder.setMessage("您确定要退出安智市场吗？");
```

- 添加按钮；

```
AdBuilder.setPositiveButton("确定", new OnClickListener() {
    public void onClick(DialogInterface dialog, int which) {
        // 退出当前应用
        MyActivity.this.finish();
    }
});
AdBuilder.setNegativeButton("取消", null);
```

Step3、4：创建、显示对话框

- 创建对话框；

```
AlertDialog ad = AdBuilder.create( );
```

- 添加按钮。

```
ad.show( );
```

使用AlertDialog对话框

- 实现一个自定义内容的对话框
  - 创建自定义内容的对话框；
  - 使用onCreateDialog( )回调方法；
  - 提示：
    - 加载用户自定义布局的XML文件，使用：

```
getLayoutInflater().inflate( )
```

Step0：添加XML布局文件

- 在主布局文件中添加按钮并绑定单击事件；
- 添加res/layout/dialog.xml文件，为对话框内容的布局页面；

Step1、2：创建、设置对话框创建器

- 在Activity的onCreateDialog方法中创建对话框创建器；

```
AlertDialog.Builder AdBuilder
        = new AlertDialog.Builder(this);
```

- 在Activity的onCreateDialog( )方法中设置对话框标题；

```
AdBuilder.setTitle("提高\"我的位置\"精确度");
```

Step2：设置对话框创建器

- 加载对话框内容布局文件；

```
// 加载用户自定义布局文件
View dialogLayout = getLayoutInflater()
        .inflate(R.layout.dialog, null);
AdBuilder.setView(dialogLayout);
// 为用户自定义的布局文件视图控件绑定事件监听器
CheckBox CbAgreen = dialogLayout
        .findViewById(R.id.Cb_Dialog_Agree);
// 为多选框绑定事件监听器
// ......
```

Step2：设置对话框创建器

- 为对话框添加按钮；

```
// 加载用户自定义布局文件
AdBuilder.setPositiveButton("确定", new OnClickListener() {
    public void onClick(DialogInterface dialog, int which) {
        // 执行操作
    }
});
AdBuilder.setNegativeButton("取消", null);
```

Step3、4：创建、显示对话框

- 创建对话框；

```
return AdBuilder.create();
```

- 在Activity中按钮的监听器方法中调用，显示对话框。

```
showDialog( 对话框标识id );
```

PopupWindow

- PopupWindow是用来实现一个弹出框的，可以使用任意布局的View作为其内容，这个弹出框是悬浮在当前activity之上的。
- PopupWindow与AlertDialog的主要区别是：
  - AlertDialog只能默认显示在屏幕最中间（当然也可以通过设置WindowManager参数来改变位置）；
  - PopupWindow可以指定显示位置的，随便哪个位置都可以，更加灵活。

PopupWindow示例

对话框小结

- 对话框：在应用的退出、更新、提示等地方使用。
  - AlertDialog
  - PopupWindow
- 对话框使用方法小结：
  - 宏观上，对话框创建有两种方法。
    - 在事件监听器中直接创建对话框；
    - 在onCreateDialog( )回调方法中创建对话框。
  - 对话框创建及使用流程：
    - 创建AlertDialog.Builder对象；
    - 设置对话框属性（标题、图标、内容、按钮、……）；
    - 创建、显示对话框。

目录

Toast简介

- Toast：没有焦点（即不干扰用户其它操作），且显示时间较短，会自动消失。

Toast简介

- Toast：没有焦点（即不干扰用户其它操作），且显示时间较短，会自动消失。
- Toast一般使用在用户信息合法性校验、关闭应用时的提示等场合。
- 同对话框一样，Toast一般在事件监听器中使用（即当特定事件触发时，显示Toast消息）。

使用Toast

- Toast使用的基本流程：
  - 创建Toast；
  - 设置Toast基本属性；
    - 可以设置Toast消息内容来自于XML布局文件，使用流程基本同对话框，方法为Toast.setView( )。
  - 显示Toast。

使用Toast

- 实现简单的用户注册信息校验
  - 当点击“注册”按钮时，校验密码
  - 弹出Toast

使用Toast

- Step0：在XML文件中添加视图控件并绑定单击事件（用以触发Toast）；
  - 在res/layout/main.xml文件中添加按钮（略）；
  - 在Activity中为"注册"按钮绑定单击事件（略）。

Step1：创建Toast

- 创建Toast；
  - 在"注册"按钮的监听器函数中。

```
Toast toastTip
    = Toast.makeText(MyActivity.this,
                     "提示字符串",
                     Toast.LENGTH_LONG);
```

  - 第1个参数：表示上下文环境。
  - 第2个参数：提示文本。
  - 第3个参数：提示信息显示时间。

Step2：设置Toast

- 设置Toast基本属性；

```
toastTip.setGravity(Gravity.CENTER, 0, 0);
```

  - 设置Toast信息的显示位置。
  - 其它常用属性参考：

http://developer.android.com/reference/android/widget/Toast.html

Step3：使用Toast

- 显示Toast。

```
toastTip.show( );
```

- 练习：点击两次返回按钮退出系统。

Toast小结

- Toast：多用在用户数据校验、退出应用前提示等场合。
- Toast使用流程：
  1. 创建Toast；
  2. 设置Toast基本属性（可能设置Toast内容由XML文件加载）；
  3. 显示Toast。

目录

Notification简介

- Notification：在状态栏显示提示信息，除非用户查看或关闭信息，状态栏才取消显示。

Notification简介

- Notification在状态栏显示提示信息，除非用户查看或关闭信息，状态栏才取消显示。
- Notification一般使用在收到短信后、收到应用的推送消息后、收到未接电话等场合。
- Notification使用时，需要借助NotificationManager（通知管理器）来实现。

使用Notification

- NotificationManager ：通知管理器类，它是一个系统服务，调用 notify() 方法可以向系统发送通知。
- Notification.Builer ：通知构造器，使用建造者模式构建 Notification 对象。
- Notification ：通知类，保存通知相关的数据
- NotificationChannel ：通知渠道，Android API 26引入的新特性。

使用Notification

- Notification使用的基本流程：
  1. 获取通知管理器服务（NotificationManager）；
  2. 创建通知构造器（Notification.Builer）；
  3. 通过通知构造器创建通知（Notification）；
  4. 调用通知管理器服务发送通知。

使用Notification

- 简单的状态栏提示信息展示。
  - 当点击“发送”按钮时，在状态栏显示。

Step0：添加控件并绑定事件

- 在XML文件中添加视图控件并绑定单击事件（以触发Notification）；
  - 在res/layout/avtivity_main.xml文件中添加按钮（略）。
  - 在Activity中为"发送短信"按钮绑定单击事件（略）。

Step1：获取通知管理器服务

- 获取通知管理器服务：

```
NotificationManager manager
    = (NotificationManager)getApplicationContext()
        .getSystemService(Context.NOTIFICATION_SERVICE);
```

Step2：创建通知构造器

- 创建通知构造器：

```
Notification.Builder builder
    = new Notification.Builder(getApplicationContext())
                .setContentTitle("最简单的Notification")
                .setContentText("只有小图标、标题、内容")
                .setSmallIcon(R.drawable.header_xiaoxin)
                .setLargeIcon(BitmapFactory
                    .decodeResource(getResources(),
                            R.drawable.header_xiaoxin));
```

Step3、4：创建通知并发送

- 通过通知构造器创建通知：

```
Notification noti = builder.build();
```

- 通过通知管理器发送通知：

```
manager.notify(0, noti);
```

目录

菜单简介

- 菜单：显示一个应用程序的主界面中不是直接可见的额外选项的视图组件。

菜单简介

- 菜单：用来显示一个应用程序的主用户界面中不是直接可见的额外选项。
- 菜单的应用十分广泛，应用的设置窗口、选项窗口、快捷操作等等都有菜单的身影。
- 在Android中支持3种菜单形式：
  - 选项菜单；
  - 子菜单；
  - 上下文菜单。

菜单简介

- 选项菜单：当用户按下"Menu"键时，弹出的菜单。
  - 如Android主窗口点击"Menu"弹出的菜单。
- 子菜单：当用户点击"选项菜单"中的某一项时，弹出的附加菜单。
  - 如在选项菜单中点击某一个选项时，弹出。
- 上下文菜单：当用户长按某个视图元素时，弹出的菜单（相当于电脑中的右键菜单）。
  - 如文本元素长按时，会出现"复制"类菜单。

使用选项菜单和子菜单

- 选项菜单/子菜单使用的基本流程：
  1. 创建菜单项XML布局文件（或由Java代码生成）；
  2. 在Activity中创建菜单（onCreateOptionsMenu方法）；
  3. 绑定菜单项选择事件（onOptionsItemSelected方法）。

使用选项菜单和子菜单

- 实现简单的选项菜单，要求如下：
  - 建立一个简单的选项菜单；
  - 使用XML布局文件建立菜单；
  - 该菜单中包含子菜单；
  - 为菜单项绑定选择事件。

Step1:创建菜单资源

- Step1：使用XML文件创建菜单资源；
  - 在res/menu/menu_options.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/menu_item1" android:title="选项一">
        <menu>
            <item android:id="@+id/menu_item1_item1"
                android:title="子菜单选项一" />
            <item android:id="@+id/menu_item1_item2"
                android:title="子菜单选项二" />
        </menu>
    </item>
    <item android:id="@+id/menu_item2" android:title="选项二" />
</menu>
```

Step1:创建菜单资源

- 该文件中根节点必须是<menu>元素。
- 子元素可以是 <item> 或 <group> 元素。
- 具体节点属性参考：

http://developer.android.com/guide/topics/resources/menu-resource.html

Step2：加载菜单资源

- 在Activity中加载菜单资源；

```
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_options, menu);
    return super.onCreateOptionsMenu(menu);
}
```

  - 加载XML文件方式的菜单项；
  - 若使用Java代码方式加载菜单项，需要调用menu对象的add方法为其依次添加菜单项；
  - 具体查看：

http://developer.android.com/guide/topics/ui/menus.html#options-menu

Step3：绑定菜单项选择事件

- 绑定菜单项选择事件。

```
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.menu_item2: // TODO
            return true;
        case R.id.menu_item1_item1: // TODO
            return true;
        ......
    }
    return super.onOptionsItemSelected(item);
}
```

– 其它菜单事件监听器，参考：

http://developer.android.com/reference/android/app/Activity.html#pubmethods

使用上下文菜单

- 上下文菜单使用的基本流程：
  1. 创建菜单项XML布局文件（或由Java代码生成）；
  2. 在Activity中创建上下文菜单；
  3. 为视图元素绑定上下文菜单；
  4. 绑定菜单项选择事件。

使用上下文菜单

- 实现简单的上下文菜单，要求如下：
  - 模拟文本的复制和粘贴功能；
  - 使用XML布局文件建立菜单；
  - 该菜单中包含子菜单；
  - 为菜单项绑定选择事件。

Step1：使用XML文件创建菜单资源

- 在res/menu/menu_options.xml中。
  - 该文件的使用方法，同选项菜单。

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/copy"
        android:title="复制" />
    <item android:id="@+id/paste"
        android:title="粘贴" />
</menu>
```

Step2：在Activity中加载菜单资源

- 加载XML文件方式的菜单项。
- 若使用Java代码方式加载菜单项，需要调用menu对象的add方法为其依次添加菜单项。

```
public void onCreateContextMenu(ContextMenu menu, View v,
                                ContextMenu.ContextMenuInfo menuInfo) {
    getMenuInflater().inflate(R.menu.menu_options, menu);
    super.onCreateContextMenu(menu, v, menuInfo);
}
```

- 具体查看：

http://developer.android.com/guide/topics/ui/menus.html#context-menu

Step3：绑定菜单项选择事件

```
public boolean onContextItemSelected(MenuItem item) {
    Context context = getApplicationContext();
    ClipboardManager cm = (ClipboardManager)
            getSystemService(Context.CLIPBOARD_SERVICE); // 获取剪贴板管理服务
    switch (item.getItemId()) {
    case R.id.copy:
        EditText editText = findViewById(R.id.source_text);
        cm.setPrimaryClip(ClipData.newPlainText("test",
                editText.getText().toString())); // 获取文本并放入剪贴板中
        return true;
    case R.id.paste:
        TextView textView = findViewById(R.id.copy_text);
        textView.setText(cm.getPrimaryClip().getItemAt(0).getText().toString());
        return true;
    }
    return super.onOptionsItemSelected(item);
}
```

Step4：为TextView注册上下文菜单

- 在onCreate( )方法中为TextView控件注册上下文菜单。

```
TextView textView = findViewById(R.id.copy_text);
registerForContextMenu(textView);
```

- ContextMenu必须通过Activity的registerForContextMenu(View)来进行注册，而OptionsMenu不用。

PopupMenu

- PopupMenu可以非常方便的在指定view的下面显示一个弹出菜单，类似于ActionBar溢出菜单的效果。

Step1：创建菜单文件

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/item1" android:title="测试一"/>
    <item android:id="@+id/item2" android:title="测试二"/>
    <item android:id="@+id/item3" android:title="测试三"/>
    <item android:id="@+id/item4" android:title="测试四"/>
</menu>
```

Step2：给Button添加点击监听

- 在Activity的onCreate( )方法中为Button添加点击监听。

```
final Button btn = findViewById(R.id.button);
btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Step3、Step4
    }
});
```

Step3：创建弹出菜单并填充

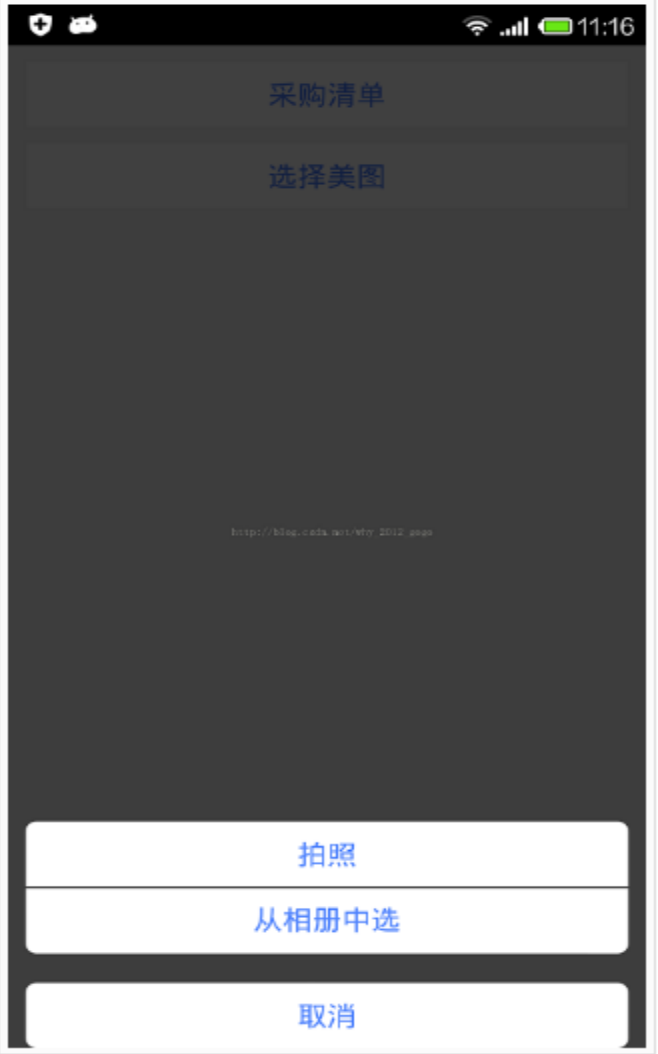
- 给指定Button创建弹出弹出菜单对象；
- 通过XML文件对创建的弹出菜单对象进行填充。

```
// 给Button创建弹出菜单
PopupMenu popupMenu
    = new PopupMenu(getApplicationContext(), btn);

// 通过XML文件对菜单进行填充
popupMenu.getMenuInflater()
    .inflate(R.menu.popup_menu, popupMenu.getMenu());
```

Step4：菜单添加点击监听并显示

```
// 给弹出菜单添加点击监听
popupMenu.setOnMenuItemClickListener(new PopupMenu
        .OnMenuItemClickListener() {
    @Override
    public boolean onMenuItemClick(MenuItem item) {
        // TODO
        return false;
    }
});
// 显示弹出菜单
popupMenu.show();
```

PopupWindow实现菜单



- Step1：自定义弹出菜单的样式（XML文件）
- Stept2：自定义一个继承自PopupWindow的类，实现PopupWindow与XML样式的整合
- Stept3：在Activity中给弹出菜单绑定组件（即测试弹出菜单）

PopupWindow实现菜单示例

- Step1：

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:gravity="center_horizontal">
    <LinearLayout
        android:id="@+id/pop_layout"
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal"
        android:layout_alignParentBottom="true">
        <Button
            android:id="@+id/btn_take_photo"
            android:layout_marginLeft="20dip"
            android:layout_marginRight="20dip"
            android:layout_marginTop="20dip"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="拍照"
            android:textStyle="bold"/>
        <Button
            android:id="@+id/btn_pick_photo"
            android:layout_marginLeft="20dip"
            android:layout_marginRight="20dip"
            android:layout_marginTop="5dip"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="从相册选择"
            android:textStyle="bold"/>
        <Button
            android:id="@+id/btn_cancel"
            android:layout_marginLeft="20dip"
            android:layout_marginRight="20dip"
            android:layout_marginTop="15dip"
            android:layout_marginBottom="15dip"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="取消"
            android:textColor="#ffffff"
            android:textStyle="bold"/>
    </LinearLayout>
</RelativeLayout>
```

PopupWindow实现菜单示例

- Stept2：

```
public class PopupDialog extends PopupWindow {
    private Button btn_take_photo, btn_pick_photo, btn_cancel;
    private View mMenuView;
    public PopupDialog(Activity context, View.OnClickListener itemsOnClick){...}
}
```

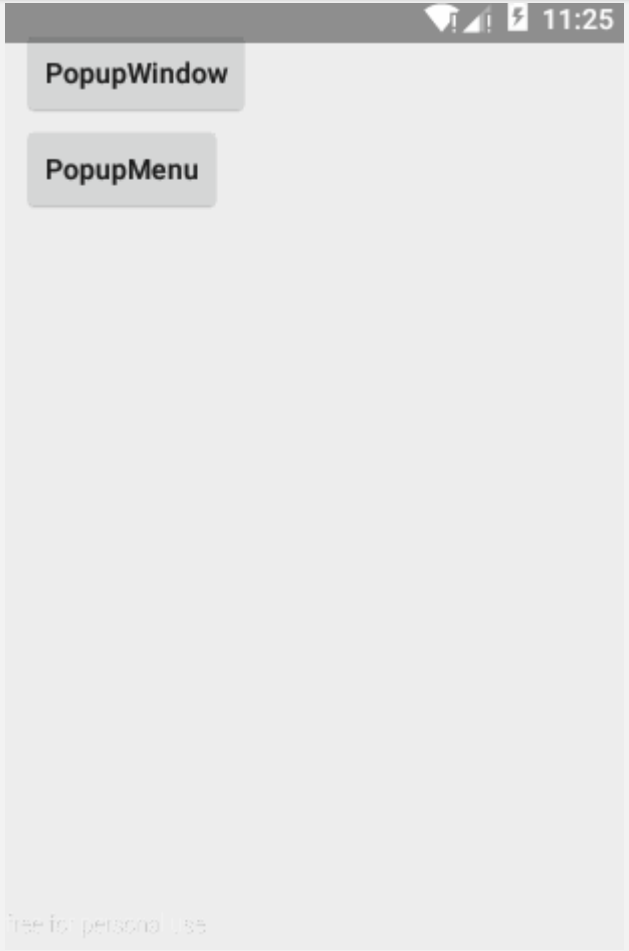
```
//设置PopupDialog的View
this.setContentView(mMenuView);
//设置PopupDialog弹出窗体的宽
this.setWidth(ActionBar.LayoutParams.MATCH_PARENT);
//设置PopupDialog弹出窗体的高
this.setHeight(ActionBar.LayoutParams.WRAP_CONTENT);
//设置PopupDialog弹出窗体可点击
this.setFocusable(true);
//设置PopupDialog弹出窗体动画效果
this.setAnimationStyle(R.style.AnimBottom);
//实例化一个ColorDrawable颜色为半透明
ColorDrawable dw = new ColorDrawable(0xb0000000);
//设置PopupDialog弹出窗体的背景
this.setBackgroundDrawable(dw);
//mMenuView添加OnTouchListener监听判断获取触屏位置如果在选择框外面则销毁弹出框
mMenuView.setOnTouchListener(new View.OnTouchListener() {
```

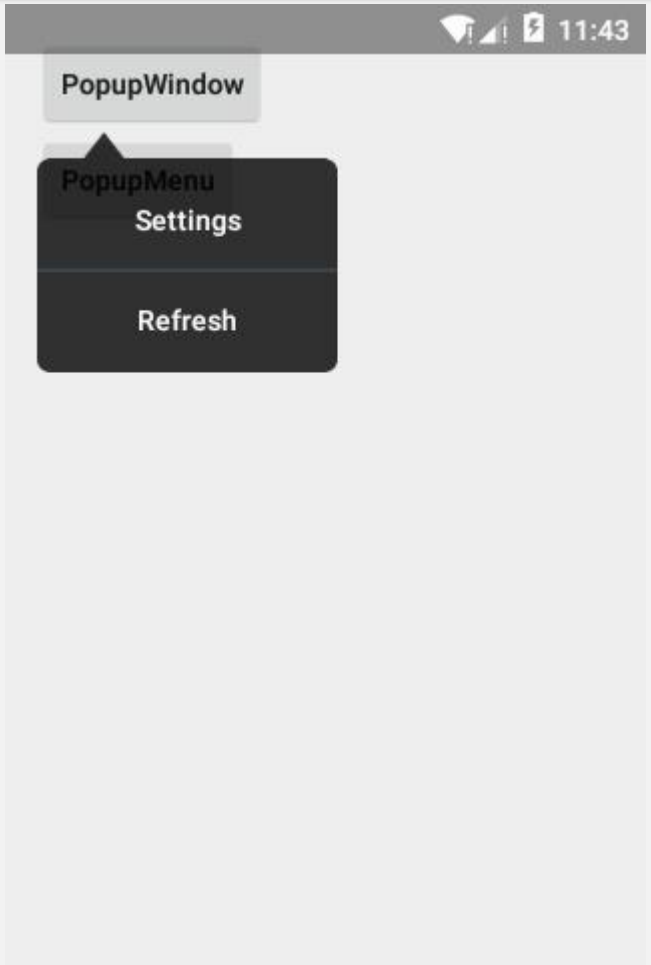
PopupWindow实现菜单示例

- Stept3：

```
public class PopupWindowMenuActivity extends AppCompatActivity {
    private TextView textView;
    private PopupDialog menuWindow;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_popup_window_menu);
        textView=(TextView)findViewById(R.id.caidantextview);
        //把文字控件添加监听，点击弹出自定义窗口
        textView.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                //实例化SelectPicPopupWindow
                menuWindow = new PopupDialog(PopupWindowMenuActivity.this,
                        itemsOnClick);
                //显示窗口
                menuWindow.showAtLocation(PopupWindowMenuActivity.this
                        .findViewById(R.id.main),
                        Gravity.BOTTOM|Gravity.CENTER_HORIZONTAL, 0, 0);
                //设置layout在PopupWindow中显示的位置

            }

        });
    }
    //为弹出窗口实现监听类
    private View.OnClickListener itemsOnClick = new View.OnClickListener(){

        public void onClick(View v) {
            menuWindow.dismiss();
            Toast.makeText(PopupWindowMenuActivity.this,v.getId()+"",
                    Toast.LENGTH_LONG).show();
        }
    };
}
```

PopupWindow和PopupMenu





PopupWindow仿PopupMenu

ActionBar

- ActionBar：活动条，位于传统标题栏的位置，即屏幕的顶部。
- ActionBar的主要功能：
  - 显示选项菜单的菜单项；
  - 使用程序图标作为返回Home或者向上的导航操作；
  - 提供交互式的View作为Action View；
  - 提供基于Tab的导航方式，可用于切换多个Fragment；
  - 提供基于下拉的导航方式。

ActionBar

- Android SDK高于11版本的默认会启用ActionBar组件。
- ActionBar的显示和隐藏可以通过程序来控制。
  - show()：显示ActionBar。
  - hide()：隐藏ActionBar。

使用ActionBar显示选项菜单项

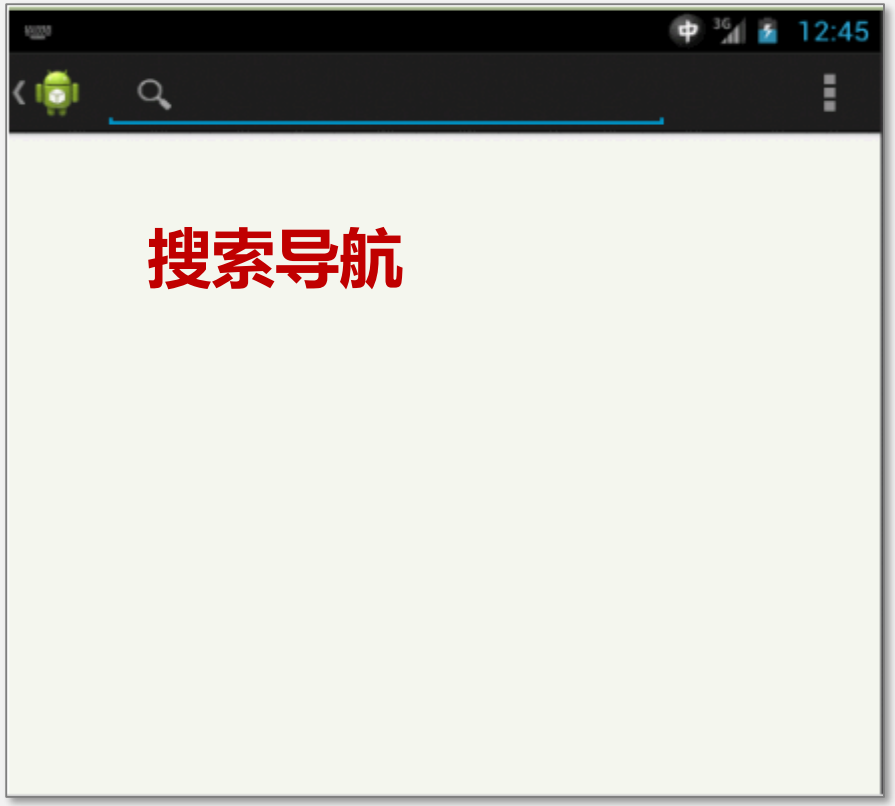
- 解决没有Menu按键的问题，通过ActionBar可以将选项菜单显示成Action Item。
  - setShowAsAction(int actionEnum)，设置选项菜单是否显示在ActionBar上。
    - SHOW_AS_ACTION_ALWAYS：总是显示在ActionBar上。
    - SHOW_AS_ACTION_COLLAPSE_ACTION_VIEW：将ActionView折叠成普通的菜单项。
    - SHOW_AS_ACTION_IF_ROOM：当ActionBar位置足够时才显示。
    - SHOW_AS_ACTION_NEVER：不将MenuItem显示在ActionBar上。
    - SHOW_AS_ACTION_WITH_TEXT：将MenuItem显示在ActionBar上，并显示该菜单项的文本。

ActionBar的其他功能

- 启动程序图标导航。
- 实现下拉式导航。

ActionBar的其他功能

- 实现搜索导航。
- 实现Tab导航。



ActionBar

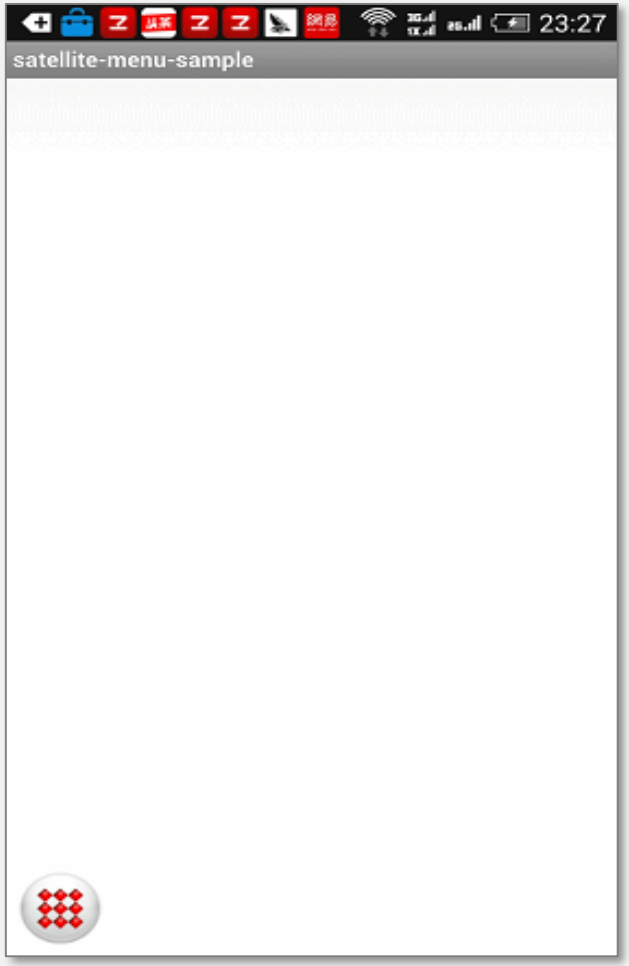

13:14
全部
全部
热门信息
朋友
家人
认识的人
关注对象
附近信息
照片
签到
心情
文字

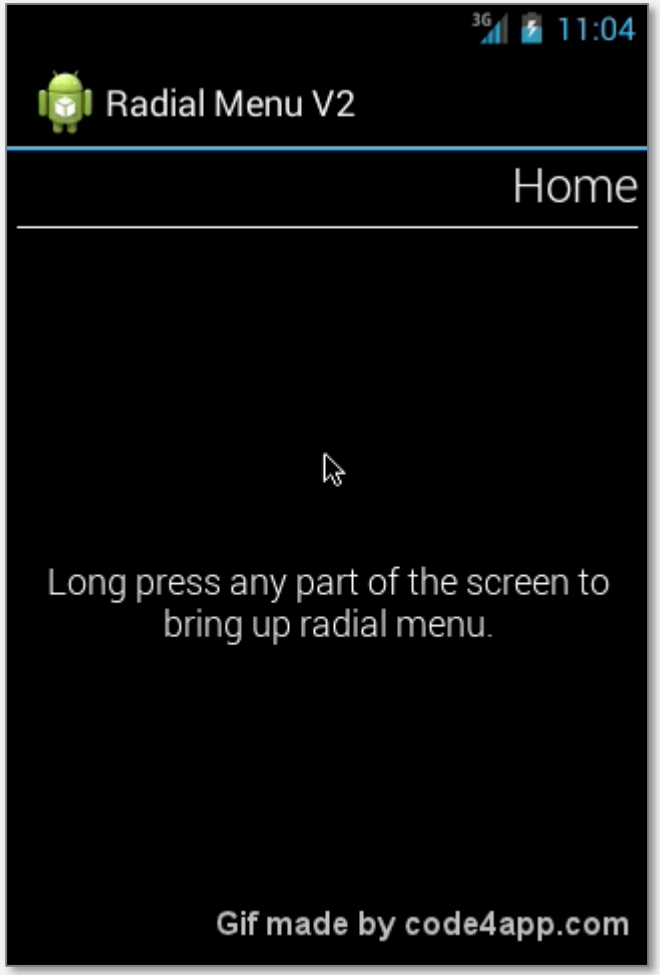

16:32
Demo
全部
全部
热门信息
朋友
家人
认识的人
关注对象
附近信息

菜单小结

- 菜单分类：
  - 选项菜单；
  - 子菜单；
  - 上下文菜单。
- 菜单使用的基本流程：
  1. 建立菜单布局视图（使用XML或Java代码方式）；
  2. 在Activity中加载菜单视图；
  3. 绑定元素的上下文菜单（仅限上下文菜单）；
  4. 为菜单绑定事件监听器。

课外练习


23:27
satellite-menu-sample


11:04
Radial Menu V2
Home
Long press any part of the screen to bring up radial menu.
Gif made by code4app.com


11:05
Semi Circular Radial Menu
Gif made by code4app.com
Open

内容回顾

1. 对话框的使用
2. Toast的使用
3. Notification的使用
4. 菜单的使用

Thank you!