



Android应用开发

主讲教师 胡鑫

hithuxin@hit.edu.cn

第5章 Android用户界面

5.1 [用户界面基础](#)

5.2 [界面基本组件](#)

5.3 [界面布局](#)

5.4 [界面菜单](#)




5.5 [事件响应](#)

5.1 用户界面基础

- 用户界面UI（User Interface）是系统和用户之间进行信息交换的媒介。



移动用户界面设计应解决的问题

- 界面设计与程序逻辑完全分离 
 - 这不仅有利于并行开发，而且在后期修改界面时，不用再次修改程序的逻辑代码
- 根据不同型号的屏幕能自动调整界面元素的位置和尺寸 
 - 避免因为屏幕信息的变化而出现显示错误
- 增强用户体验 
 - 能够合理利用较小的屏幕显示空间，构造出符合人机交互规律的用户界面

MVC模式

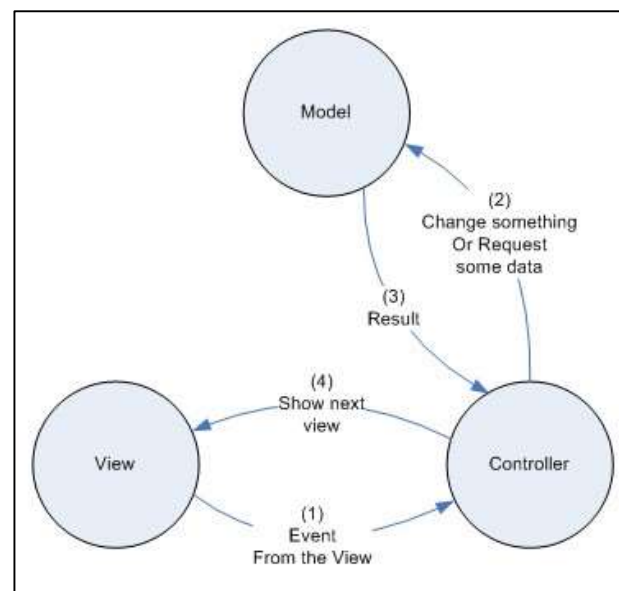
- Android界面框架采用了当前流行的MVC模式：

- MVC (Model-View-Controller)

- 视图V：显示用户界面(XML布局)
- 控制器C：处理用户输入(Activity)
- 模型M：处理业务逻辑(数据/逻辑代码)

- MVC目的：

- M和V的实现代码分离，从而使同一个程序可以使用不同的表现形式，C可确保M和V的同步，一旦M改变，V能同步更新。

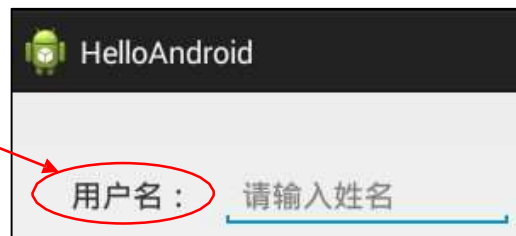


5.2 界面基本组件

- [TextView](#) / [ImageView](#)
- [Button](#) / [ImageButton](#) / [ToggleButton](#)
- [EditText](#)
- [RadioButton](#) / [CheckBox](#) / [Spinner](#)
- [ListView](#)

1. TextView

用于显示信息



■ 基本属性:

```
<TextView
```

```
    android:id="@+id/textView1" ← 组件id
```

@+id表示新建id资源

```
    android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
```

组件的宽度和高度

wrap_content: 根据内部内容自动扩展以适应其大小
match_parent: 将强制性地使组件扩展, 以填充布局单元内尽可能多的空间

```
    android:textColor="#ff0000" ← 颜色
```

```
    android:textSize="20sp" ← 字体大小
```

```
    android:textStyle="normal|bold|italic" ← 字形
```

```
    android:gravity="top|center|..." ← 文字对齐方式
```

```
    android:text="用户名: " /> ← 也使用引用形式: 如@string/**
```

在程序中控制TextView

■ 示例：

类型强制
不能少

====findViewById()方法====
利用View的id值 来获取相应
的View对象

TextView tv=(TextView)findViewById(R.id.textView);

tv.setTextSize(18); //设置文字大小， 单位是sp

tv.setText("username:"); //设置文字内容

String str=tv.getText().toString(); //获取文字内容(转换为字符串)

tv.setTextColor(Color.BLUE); //设置文字颜色或Color.rgb(255, 0, 0)

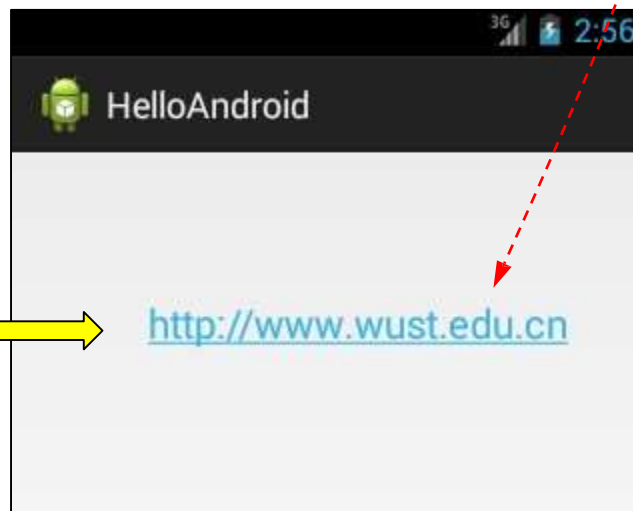
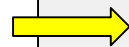
//或者 tv.setTextColor(Color.parseColor("#0000ff"));

tv.setTypeface(null, Typeface.ITALIC); //设置字形

补充： android:autoLink属性

- android:autoLink="web" 自动识别web超链接
- android:autoLink="email" 自动识别email地址
- android:autoLink="phone" 自动识别电话号码
- android:autoLink="all"

```
android:autoLink="web"  
android:text="http://www.wust.edu.cn"
```



2. ImageView

用于显示图片

■ 基本XML描述:

```
<ImageView
```

```
    android:id="@+id/imageView"
```

← 组件id

```
    android:scaleType="缩放类型"
```

```
    android:src="@drawable/ic_launcher" />
```

← 默认fitCenter: 等比缩放填充控件大小, 并居中展示; fitStart: 等比缩放靠左显示

← 要显示的图片

详见: <https://www.jianshu.com/p/32e335d5b842>

在程序中控制 ImageView

■ 示例：

```
ImageView iv=(ImageView)findViewById(R.id.imageView);
```

```
iv.setImageResource(R.drawable.bg320_480);    //根据id值加载图片
```

```
iv.setVisibility( View.INVISIBLE );    //图片不可见
```

可见： View.VISIBLE 注： VISIBLE值为0， INVISIBLE为4

补充：ImageView从相册中选择图片

先在Activity类中定义一个类成员：
ImageView iv;

```
iv = (ImageView) findViewById(R.id.photo);
iv.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(Intent.ACTION_PICK, null);
        intent.setDataAndType(MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
                               "image/*");
        startActivityForResult(intent, 0x1);
    }
});
```

此处代码放在onCreate中

备注：如果相册没有图片会自动打开相机

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == 0x1 && resultCode == RESULT_OK) {
        if (data != null) {
            iv.setImageURI(data.getData());
        }
    }
    super.onActivityResult(requestCode, resultCode, data);
}
```

提醒：ImageView没有getImageURI()

根据图片的URI值加载图片

3. Button / ImageButton / ToggleButton



■ Button基本属性:

```
<Button  
    android:id="@+id/button"  
    android:text="提交" />
```

■ ImageButton基本属性:

```
<ImageButton  
    android:id="@+id/imageButton"  
    android:src="@drawable/login" />
```

Button / ImageButton单击事件响应

```
Button bt=(Button)findViewById(R.id.button);  
bt.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        // 代码  
    }  
});
```

例如：在程序中修改ImageButton图片的代码：

```
ImageButton bt=(ImageButton)findViewById(R.id.imageButton1);  
bt.setImageResource(R.drawable.ic_launcher);
```

ToggleButton基本属性



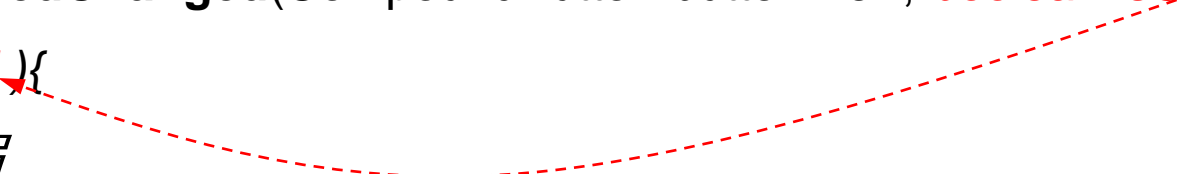
```
<ToggleButton  
    android:id="@+id/toggleButton"  
    android:textOff="关闭" ← 设置关闭时显示的文字(默认为OFF)  
    android:textOn="打开" /> ← 设置开启时显示的文字(默认为ON)
```

提示：Switch控件用法
同ToggleButton



ToggleButton开关切换事件响应

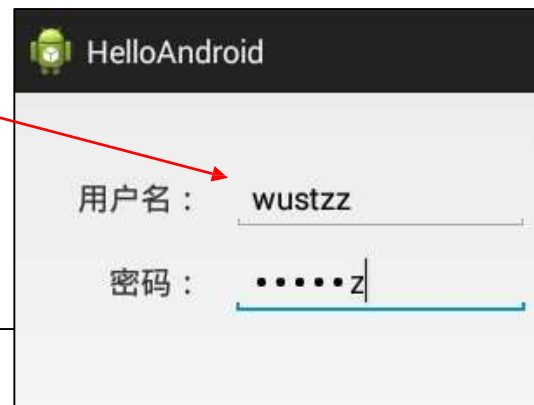
```
ToggleButton tb=(ToggleButton)findViewById(R.id.toggleButton);  
tb.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener(){@Override  
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked)  
        if( isChecked ){  
            // 开关开启  
        }  
        else{  
            // 开关关闭  
        }  
    }  
});
```



4. EditText

用于输入和编辑字符串

基本属性：



<EditText

android:id="@+id/editText"

← 组件id 限制输入或显示

android:maxLength="10"

← 的文本长度 单行文本(默认

android:singleLine="true"

← 可多行)

android:ems="10"

← 设置TextView显示宽度为N个汉字宽度

android:inputType="textPassword"

← 设置为密码框

android:hint="提示信息"

← 设置提示信息(text属性设置为空才显示)

android:text="输入值">

<requestFocus />

← 当前TextView获得焦点

</EditText>

android:inputType的一些取值

- `android:inputType="numberPassword"` 数字密码框
- `android:inputType="textMultiLine"` 多行文本
- `android:inputType="number"` 数字键盘(只能输入数字)
- `android:inputType="phone"` 拨号键盘
- `android:inputType="time"` 时间键盘



android:digits属性

- 该属性用于设置允许输入哪些字符。
- 示例：
 - `android:digits="abcd"` 只能输入abcd这四种字符

在程序中控制EditText

■ 示例：

类型强制
不能少



```
EditText et=(EditText)findViewById(R.id.editText);
```

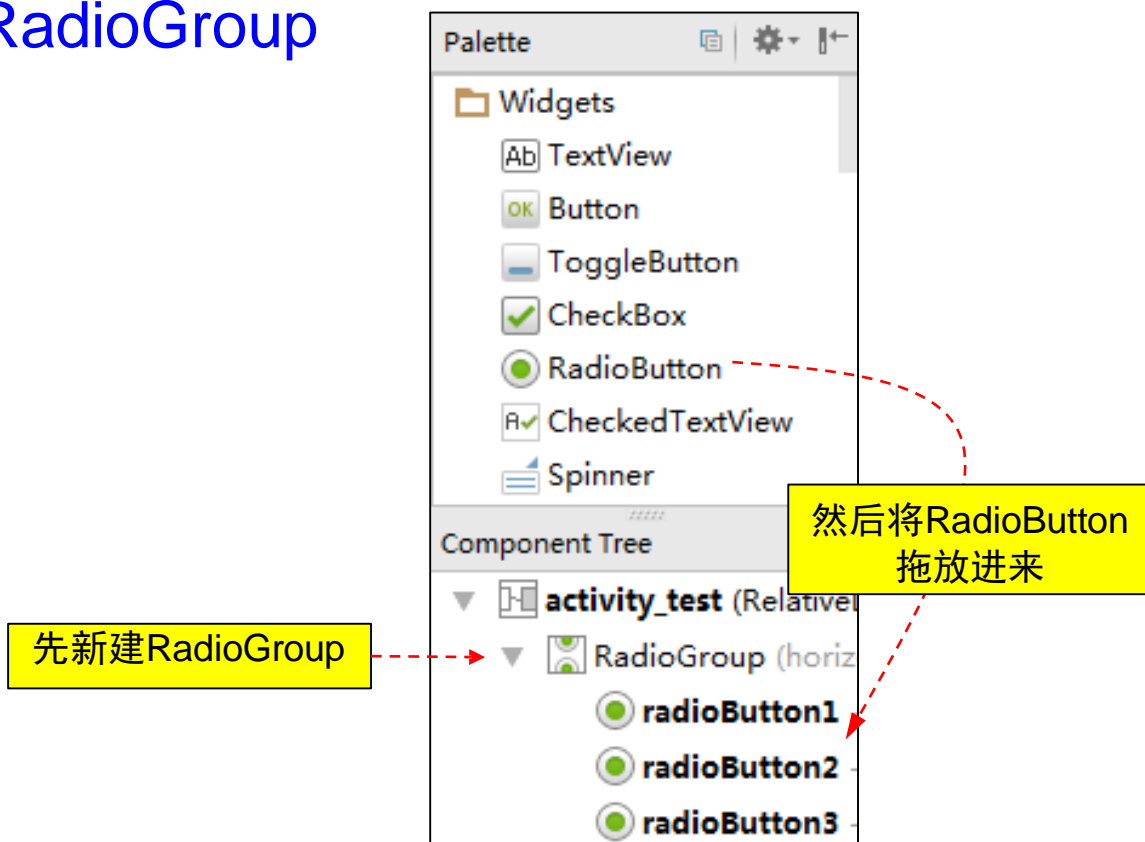
```
et.setText("hello");
```

```
String msg=et.getText().toString(); //toString()转换为字符串
```

5. RadioButton



- 单选按钮须归属一个组，每个组只能有一个选项被选中
- 所以通常使用的是RadioGroup



RadioGroup主要属性

```
<RadioGroup
    android:id="@+id/radioGroup" ← RadioGroup的id
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal" > ← horizontal: 水平排列
                                         vertical: 垂直排列
```



```
<RadioButton
    android:id="@+id/radioButton1" ← RadioButton的id
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="true" ← 设置选中 设置
    android:text="男" /> ← 按钮显示的文字
```

```
<RadioButton
    android:id="@+id/radioButton2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="女" />
```

```
</RadioGroup>
```

如何获取RadioButton选中项的值



```
Button bt=(Button)findViewById(R.id.button);  
bt.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        RadioGroup rg=(RadioGroup)findViewById(R.id.radioGroup);  
  
        RadioButton rb=(RadioButton)findViewById( rg.getCheckedRadioButtonId() );  
  
        Toast.makeText(MainActivity.this, "你选中了"+rb.getText(), Toast.LENGTH_SHORT).show();  
    }  
});
```

先找到RadioGroup

再获得选中项的Id

最后获得选中项的文本值

提醒：最好判断rb一下是否为空

RadioGroup选中项改变事件

宋朝开国皇帝是哪位？

☐ 李渊

☐ 赵匡胤

☒ 朱元璋

☐ 努尔哈赤

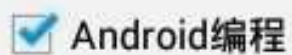
你答错了！

```
RadioGroup rg=(RadioGroup)findViewById(R.id.radioGroup);
rg.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(RadioGroup group, int checkedId) {
        RadioButton rb=(RadioButton)findViewById(checkedId); if("赵匡胤
        ".equals(rb.getText()))
            Toast.makeText(TestActivity.this, "你答对了", Toast.LENGTH_SHORT).show();
        else
            Toast.makeText(TestActivity.this, "你答错了", Toast.LENGTH_SHORT).show();
    }
});
```

↑ 这里使用RadioGroup类，不同于ToggleButton

选中项的Id

6. CheckBox



■ CheckBox基本属性：

```
<CheckBox
```

```
    android:id="@+id/checkBox"
```

```
    android:text="Android编程" />
```

← 设置按钮显示的文字

如何获得多个CheckBox的选中值

```
Button bt=(Button)findViewById(R.id.button);  
bt.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        String str="选中了： ";  
        CheckBox cb1= (CheckBox)findViewById(R.id.checkBox1);  
        CheckBox cb2= (CheckBox)findViewById(R.id.checkBox2);  
        if(cb1.isChecked())  
            str+=cb1.getText()+" ";  
        if(cb2.isChecked())  
            str+=cb2.getText()+" ";  
        Toast.makeText(MainActivity.this, str, Toast.LENGTH_SHORT).show();  
    }  
});
```

显示选中项值

☒ Java编程

☒ Android编程

选中了：Java编程 Android编程

CheckBox选中状态改变事件响应

TextView



```
CheckBox cb=(CheckBox)findViewById(R.id.checkBox);
```

```
cb.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
```

```
@Override
```

这里使用CompoundButton类，与ToggleButton相同

```
public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
```

```
    TextView tv=(TextView)findViewById(R.id.textView2);
```

是否选中

```
    if( isChecked )
```

```
        tv.setTextColor(Color.RED);
```

```
    else
```

```
        tv.setTextColor(Color.BLACK);    }
```

```
});
```

7. Spinner

下拉列表

■ Spinner基本属性

```
<Spinner
```

```
    android:id="@+id/spinner"
```

```
    android:spinnerMode="dropdown"
```

```
    android:prompt="提示信息"
```

```
    android:popupBackground="背景(dialog模式下无效)"
```

```
    android:entries="内容"/>
```

设置Spinner的模式
dropdown: 默认
dialog: 弹出对话框



Spinner数据填充: ✓

(1) 打开res/values/strings.xml文件, 添加:

```
<string name="major">专业</string>
```

以后作为Spinner
dialog模式时的标题

(2) 新建res/values/arrays.xml资源文件:

```
<string-array name="major_array">
```

```
<item>计算机科学</item>
```

```
<item>软件工程</item>
```

```
</string-array>
```



Spinner数据填充：

(3) 在Spinner对象中添加属性：

```
<Spinner
```

```
    android:id="@+id/spinner"
```

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
```

```
    android:spinnerMode="dialog" ← 设置Spinner模式为弹出对话框
```

```
    android:prompt="@string/major" ← 设置对话框模式时的标题
```

```
    android:entries="@array/major_array" />
```

← 设置Spinner数据来源(string-array)



如何获取Spinner选中项值

```
Button bt=(Button)findViewById(R.id.button);
bt.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Spinner sp=(Spinner)findViewById(R.id.spinner);
        String str=sp.getSelectedItem().toString(); 获得选中项内容
        Toast.makeText(MainActivity.this,"选中了:"+str,Toast.LENGTH_SHORT).show();
    }
});
```

Spinner选中项事件响应

```
sp.setOnItemSelectedListener( new AdapterView.OnItemSelectedListener() {  
    @Override  
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {  
        // 选中某项  
    }  
    @Override  
    public void onNothingSelected(AdapterView<?> parent) {  
        // 失去焦点时，此处一般不写代码  
    }  
});
```

↑ 这里使用AdapterView类

↑ Spinner对象

↑ 选中项对象
这里是一个TextView

↑ 选中项的序号值

↑ 选中项的Id值

Spinner选中项事件示例

```
sp.setOnItemSelectedListener( new AdapterView.OnItemSelectedListener() {  
    @Override  
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {  
        Toast.makeText(MainActivity.this,  
            "选中了: "+ parent.getItemAtPosition(position).toString(),  
            Toast.LENGTH_SHORT).show();  
    }  
    @Override  
    public void onNothingSelected(AdapterView<?> arg0) {  
        //失去焦点时, 此处一般不写代码  
    }  
});
```

此例也可直接用
((TextView)view).getText()

示例：Spinner级联操作

(1) **res/values/strings.xml**中添加：

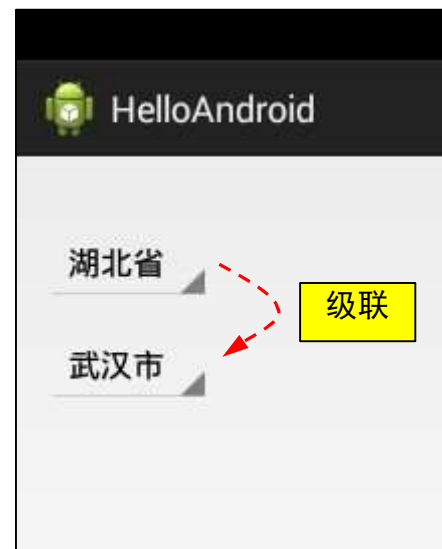
```
<string name="province">请选择省份</string>  
<string name="city">请选择城市</string>
```

(2) **res/values/arrays.xml**中添加：

```
<string-array name="province_array">  
    <item>湖北省</item>  
    <item>湖南省</item>  
</string-array>
```

```
<string-array name="HB_city_array">  
    <item>武汉市</item>  
    <item>宜昌市</item>  
</string-array>
```

```
<string-array name="HN_city_array">  
    <item>长沙市</item>  
    <item>湘潭市</item>  
</string-array>
```



示例：Spinner级联操作

(3) 添加两个Spinner，基本描述：

```
<Spinner
    android:id="@+id/spinner1"
    android:spinnerMode="dialog"
    android:prompt="@string/province"
    android:entries="@array/province_array" />
```

```
<Spinner
    android:id="@+id/spinner2"
    android:spinnerMode="dialog"
    android:prompt="@string/city" />
```

第一个Spinner填充数据

(4) MainActivity主要代码:

```

public class MainActivity extends Activity {
    private enum ProvinceList {湖北省, 湖南省}; //定义一个枚举类型, 以后给switch用 ✓
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
        Spinner sp=(Spinner)findViewById(R.id.spinner1);
        sp.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
                String province=parent.getItemAtPosition(position).toString(); //得到省份
                ArrayAdapter<CharSequence> adapter=null;
                Spinner sp2=(Spinner)findViewById(R.id.spinner2);
                switch(ProvinceList.valueOf(province)){
                    case 湖北省:
                        adapter = ArrayAdapter.createFromResource(
                            MainActivity.this,
                            R.array.HB_city_array, //用对应的string-array填充spinner2
                            android.R.layout.simple_spinner_item ); //设置显示模式
                        adapter.setDropDownViewResource(
                            android.R.layout.simple_spinner_dropdown_item); //设置下拉样式
                        sp2.setAdapter(adapter); //填充
                        break;
                    case 湖南省: ...
                    default:
                }
            }
        });
    }
}

```

使用ArrayAdapter
适配器来填充
Spinner

8. ListView

列表框

Java编程

Android编程

ASP.NET编程

- ListView通常使用适配器来填充数据和设置显示样式。
- ListView主要属性：

```
<ListView
```

```
    android:id="@+id/listView" ← ListView的id
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
```

```
    android:layout_below="@+id/btn_test"
```

```
    android:layout_marginTop="20dp"
```

```
    android:entries="@array/major_array"/> ← 填充的静态数据(见后)
```

android:entries数据填充（静态）：

新建res/values/array.xml文件：

```
<string-array name="major_array">  
    <item>计算机科学</item>  
    <item>软件工程</item>  
</string-array>
```

引用： android:entries="@array/major_array"

↑
注意单数形式

ListView数据动态填充方法



Java编程

Android编程

ASP.NET编程

J2EE编程

```
ArrayList<String> list = new ArrayList<String>();
```

```
list.add("Java编程"); list.add("Android编
```

```
程"); list.add("ASP.NET编程");
```

```
list.add("J2EE编程");
```

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(
```

ArrayAdapter适配器

MainActivity.this,

上下文

android.R.layout.simple_list_item_1,

填充的数据 List<T>

list);

```
ListView li=(ListView)findViewById(R.id.listView1);
```

```
li.setAdapter(adapter);
```

设置适配器

系统提供的子项布局资源id,
也可自定义布局

系统提供的一些其他布局

- `android.R.layout.simple_list_item_single_choice` (radio样式)
- `android.R.layout.simple_list_item_multiple_choice` (checkbox样式)
- `android.R.layout.simple_list_item_checked`

Java编程	<input type="radio"/>
Android编程	<input checked="" type="radio"/>
ASP.NET编程	<input type="radio"/>
J2EE编程	<input type="radio"/>

Java编程	<input checked="" type="checkbox"/>
Android编程	<input checked="" type="checkbox"/>
ASP.NET编程	<input type="checkbox"/>
J2EE编程	<input type="checkbox"/>

Java编程	<input checked="" type="checkbox"/>
Android编程	<input checked="" type="checkbox"/>
ASP.NET编程	<input type="checkbox"/>
J2EE编程	<input type="checkbox"/>

注意： 上述这些布局

单选需要配合：`li.setChoiceMode(ListView.CHOICE_MODE_SINGLE);`

多选需要配合：`li.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE);`

ListView选项单击事件响应

```
ListView li=(ListView)findViewById(R.id.listView);
```

```
li.setOnItemClickListener( new AdapterView.OnItemClickListener() {
```

```
    @Override
```

↑ 这里使用AdapterView类

```
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
```

```
        Toast.makeText(getApplicationContext(),
```

```
            "选中了: "+((TextView)view).getText().toString()+" ,id="+id,
```

```
            Toast.LENGTH_SHORT).show();
```

```
    }
```

```
});
```

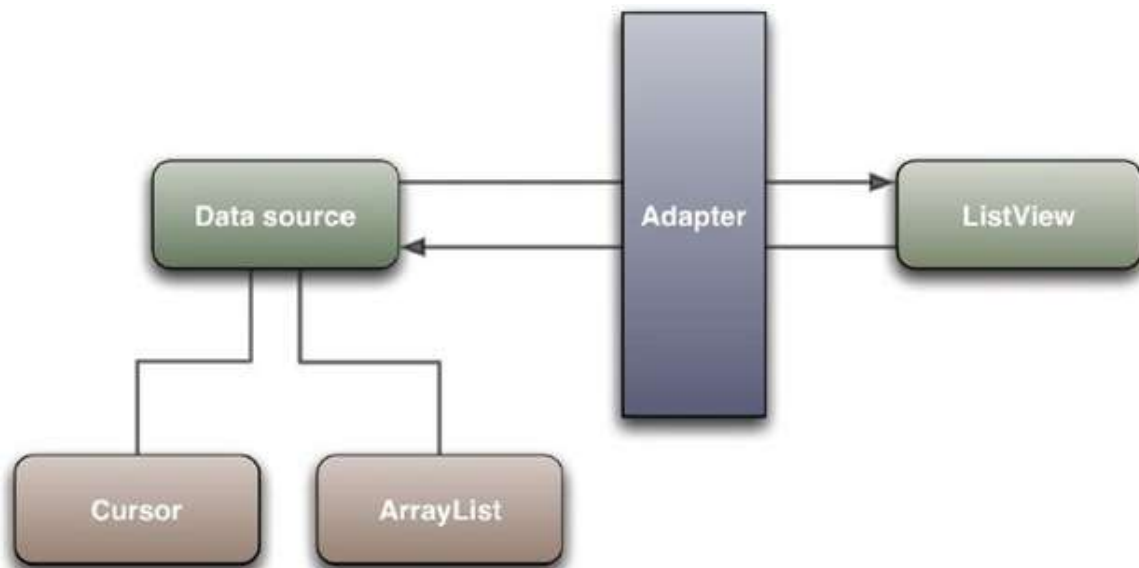
四个参数含义

parent: 指ListView对象
view: 击中项的视图对象
position: 击中项的位置值
id: 击中项的行id值(从0开始)
一般情况position和id值一样

特别注意: ListView不要使用OnItemSelectedListener(Spinner才用)

Adapter

- ✓ Adapter是连接后端数据和前端显示的适配器接口，是数据和UI（View）之间一个重要的纽带。
- ✓ 在常见的View(ListView,GridView)等地方都需要用到Adapter。



Adapter

Type hierarchy of 'android.widget.Adapter':



Adapter

- ✓ BaseAdapter是一个抽象类，继承它需要实现较多的方法，所以也就具有较高的灵活性。
- ✓ ArrayAdapter支持泛型操作，最为简单，只能展示一行字。
- ✓ SimpleAdapter有最好的扩充性，可以自定义出各种效果。
- ✓ SimpleCursorAdapter可以适用于简单的纯文字型ListView，它需要Cursor的字段和UI的id对应起来。
- ✓ 如需要实现更复杂的UI也可以重写其他方法。可以认为是SimpleAdapter对数据库的简单结合，可以方便地把数据库的内容以列表的形式展示出来。

如何获取选中的多选值

```
String str="";
```

获得选项总数

```
ListView li=(ListView)findViewById(R.id.listView1);
```

```
for(int i=0;i<li.getCount();i++){
```

获得所有选中项位置值

```
    if( li.getCheckedItemPositions().get(i))
```

查询 i 对应的boolean值

```
    {str+=li.getItemAtPosition(i).toString()+" ";
```

根据位置值获得选中项

```
    }
```

```
}
```

获得选中个数(API 11)

```
Toast.makeText(MainActivity.this,"选中了"+li.getCheckedItemCount()  
+"项:"+str,Toast.LENGTH_SHORT).show();
```

备注:

getCheckedItemPositions()返回值是SparseBooleanArray，其内部存放的元素是(序号,布尔值)对。

get(int key)方法可以获得key关键字对应存放的boolean值，如果为true，则表明该key对应的选项被选中。

ListView示例：简单的音乐播放器（真机测试）



关键问题：

1. android权限问题：如添加SD卡访问权限
2. java文件操作：如何获取SD卡指定位置.mp3音乐文件
3. ListView适配器的数据填充
4. MediaPlayer基本用法：播放、暂停、继续和停止

ListView自定义布局



主要步骤

1. 新建Fruit类
2. 自定义ListView布局
3. 自定义Adapter
4. 使用Adapter



```
public class Fruit {
    private String name; //水果名称
    private int imageld; //水果图片id值

    public Fruit(String name, int imageld) {
        this.name = name;
        this.imageld = imageld;
    }

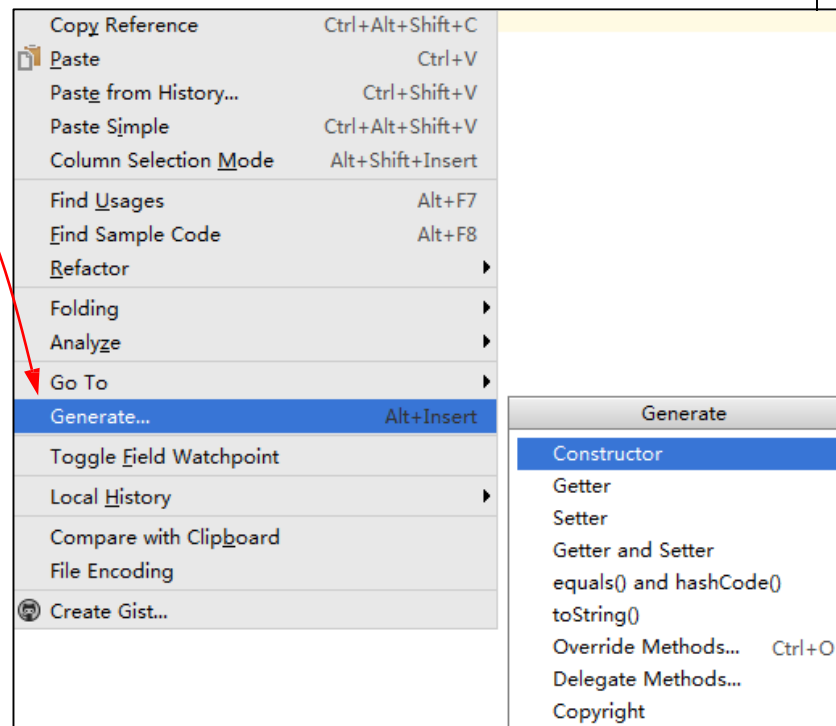
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getImageld() {
        return imageld;
    }

    public void setImageld(int imageld) {
        this.imageld = imageld;
    }
}
```

代码全部自动生成




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:id="@+id/fruit_image"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

    <TextView
        android:id="@+id/fruit_name"
        android:text="TextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:layout_gravity="center"/>

</LinearLayout>
```

左边留点空，文字居中对齐

3.新建FruitAdapter类

```
public class FruitAdapter extends ArrayAdapter<Fruit> {  
    private int resourceId; //添加一个成员  
    public FruitAdapter(Context context, int resource, List<Fruit> objects) {  
        super(context, resource, objects); //3个参数:上下文、ListView子项布局id、数据  
        resourceId=resource; //添加语句  
    }  
    @Override   
    public View getView(int position, View convertView, ViewGroup parent) {  
        Fruit fruit = getItem(position); // 获取当前选中的Fruit实例，将其内容填充到各个位置  
        //填充ListView 布局  
        View view = LayoutInflater.from(getContext()).inflate(resourceId, null);  
        ImageView fruitImage = (ImageView) view.findViewById(R.id.fruit_image);  
        TextView fruitName = (TextView) view.findViewById(R.id.fruit_name);  
        fruitImage.setImageResource(fruit.getImageId()); //根据图片id值加载图片  
        fruitName.setText(fruit.getName());  
        return view;  
    }  
}
```



4.使用自定义Adpater

```
final ArrayList<Fruit> list = new ArrayList<Fruit>();
```

```
Fruit apple = new Fruit("Apple", R.drawable.apple);
```

```
Fruit banana = new Fruit("Banana", R.drawable.banana);
```

```
list.add(apple);
```

```
list.add(banana);
```

```
FruitAdapter adapter = new FruitAdapter(  
    MainActivity.this,
```

```
    R.layout.fruit_item, ← 自定义的布局
```

```
    list );
```

```
ListView li=(ListView)findViewById(R.id.listView);
```

```
li.setAdapter(adapter);
```

4.使用自定义Adapter(续)

```
li.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
    @Override  
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
        //方法1:  
        TextView tv=(TextView)view.findViewById(R.id.fruit_name);  
        Toast.makeText(getApplicationContext(), "选中了: "+tv.getText(),Toast.LENGTH_SHORT).show();  
        //方法2:  
        Fruit f=list.get(position); //从list中获取对象(list要加final声明)  
        Toast.makeText(getApplicationContext(), "选中了: "+f.getName(), Toast.LENGTH_SHORT).show();  
    }  
});
```

5.3 界面布局

将程序的表现层和控制层分离 调整
用户界面时，无需更改程序的源代码

■ 6种界面布局

- 相对布局 RelativeLayout
- 线性布局 LinearLayout
- 框架布局 FrameLayout
- 表格布局 TableLayout
- 网格布局 GridLayout
- 绝对布局 AbsoluteLayout

RelativeLayout

- 相对布局是一种非常灵活的布局方式；
- 通过指定界面元素与其他元素的相对位置关系，来确定界面中所有元素的布局位置；
- 优点：能够最大程度保证在各种屏幕尺寸的手机上正确显示界面布局。

LinearLayout ★★★

- 线性布局是常用的一种布局方式；
- 通过设置android:orientation来确定是垂直还是水平布局：
 - `android:orientation="vertical"` 垂直布局
 - `android:orientation="horizontal"` 水平布局（默认）
- 垂直布局时：每行仅包含一个界面元素
- 水平布局时：所有界面元素都在一行

垂直布局



嵌入一个水平布局
右对齐 `android:gravity="right"`

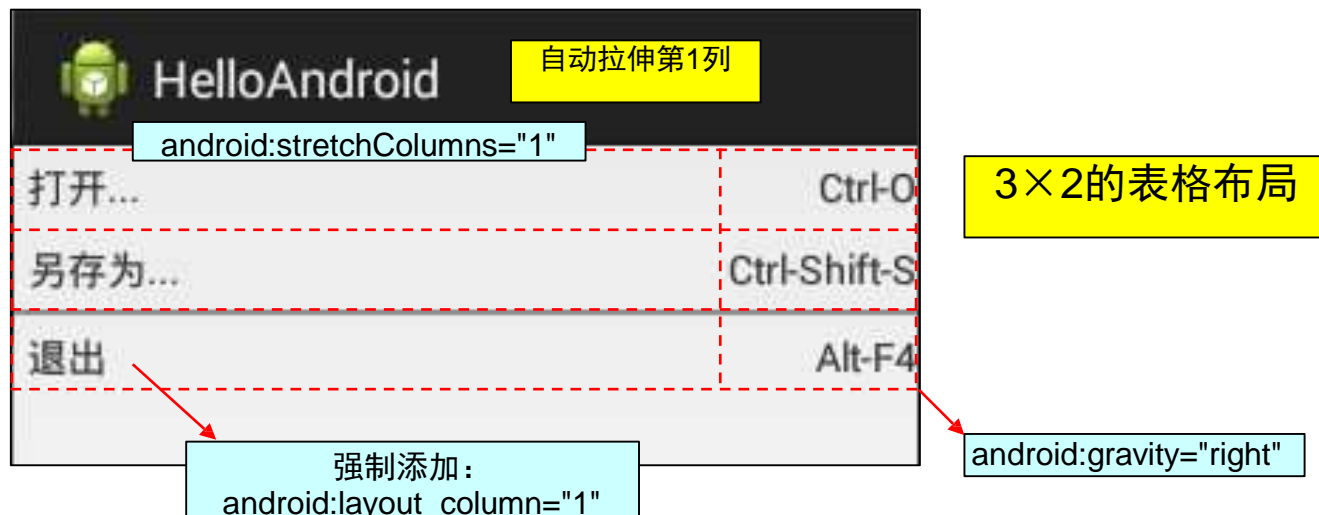
FrameLayout

- FrameLayout中所有界面对象都是从屏幕的左上角(0,0)开始布局，不能指定位置；
- 多个组件层叠排列，上一层的会覆盖下一层的控件；
- 第一个添加的组件放到最底层，最后添加的显示在最上面。



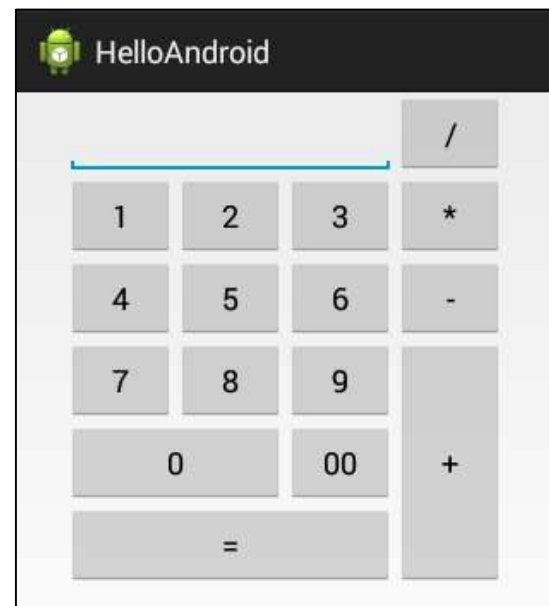
TableLayout

- 表格布局以行列的形式管理子元素，每一行是一个 TableRow 布局对象，当然也可以是普通的 View 对象；
- TableRow 表现为一个水平 LinearLayout，里面的元素会水平排列。



GridLayout

- Android4.0新支持的布局；
- 网格布局将用户界面划分为网格，界面元素可随意摆放在网格中，网格布局比表格布局更灵活；
- 在网格布局中界面元素可以占用多个网格。



AbsoluteLayout

- 绝对布局能通过指定界面元素的坐标位置，来确定用户界面的整体布局。**已经不推荐使用。**
- 绝对布局不能够根据不同屏幕对界面元素进行位置调整。

5.4 界面菜单

- [选项菜单OptionsMenu](#)
- [上下文菜单ContextMenu](#)
- [子菜单](#)

1. 选项菜单OptionsMenu

- Android手机上有个Menu键，当Menu按下时，在屏幕底部弹出一个菜单，这个菜单就叫OptionsMenu。
- 每个Activity有且只有一个OptionsMenu，它为整个Activity服务。



OptionsMenu基本用法

(1) 新建菜单资源文件并添加菜单项：

- 新建res/menu/main.xml

- 添加菜单项描述：

```
<item android:id="@+id/id名称" android:title="菜单名称"></item>
```

菜单资源示例

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">  
    <item android:id="@+id/menu1" android:title="添加"></item>  
    <item android:id="@+id/menu2" android:title="编辑"></item>  
</menu>
```

OptionsMenu基本用法

(2) 创建选项菜单：

点击手机Menu时，系统会自动调用当前Activity的onCreateOptionsMenu方法，并传一个实现了一个Menu接口的menu对象供你使用。

在Activity中覆盖onCreateOptionsMenu()方法

```
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    getMenuInflater().inflate(R.menu.main, menu);  
    return true;  
}
```


这里使用的是
main.xml菜单资源文件

OptionsMenu基本用法

(3) 添加选择菜单项事件：

在Activity中覆盖onOptionsItemSelected()方法

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch ( item.getItemId() ) {
        case R.id.某个菜单id:
            //do something
            return true;
        case R.id.某个菜单id:
            //do something
            return true;
        default:
            return false;
    }
}
```



根据菜单项id
进行分支

【[返回](#)】

2. 上下文菜单ContextMenu

- 上下文菜单类似Windows中的鼠标右键弹出菜单；
- 上下文菜单一般是通过长按（大约2秒）屏幕弹出。



上下文菜单基本用法

- (1) 新建菜单资源并添加菜单项描述
- (2) 创建上下文菜单：

在Activity中覆盖onCreateContextMenu()方法

```
@Override  
public void onCreateContextMenu(ContextMenu menu, View v,  
                                ContextMenuInfo menuInfo) {  
    menu.setHeaderTitle("标题");  
    getMenuInflater().inflate(R.menu.main, menu);  
}
```


上下文菜单弹出时可以加个标题栏

这里假定仍然使用
main.xml菜单资源文件

(3) 添加选择菜单项事件：

在Activity中覆盖onContextItemSelected()方法

```
@Override
public boolean onContextItemSelected(Menuitem item) {
    switch ( item.getItemId() ) {
        case R.id.某个菜单id:
            //do something
            return true;
        case R.id.某个菜单id:
            //do something
            return true;
        default:
            return false;
    }
}
```



根据菜单项id
进行分支

(4) 将上下文菜单注册到某个View组件上的： 

示例：在Activity的onCreate方法中：

```
EditText et1=(EditText)findViewById(R.id.editText1);
```

```
registerForContextMenu(et1);
```

功能：将上下文菜单
注册到TextView组件上

unregisterForContextMenu：取消上下文菜单

3. 子菜单

- Android系统使用浮动窗体的形式显示菜单子项，可以更好地适应小屏幕的显示方式。



创建子菜单方法

- 子菜单：在菜单项<menu></menu>中嵌套<item>即可

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >  
  <item android:id="@+id/menu1" android:title="添加"> ← 一级菜单  
    <menu>  
      <item android:id="@+id/menu1_1" android:title="添加书籍"></item>  
      <item android:id="@+id/menu1_2" android:title="添加出版社"></item>  
    </menu>  
  </item>  
  <item android:id="@+id/menu2" android:title="编辑"></item> ← 一级菜单  
</menu>
```

由于子菜单总是依赖选项菜单或上下文菜单出现，所以没有特殊编程，只是在 switch(item.getItemId()) case 时注意用子菜单的id

5.5 事件响应

- 以Button按钮单击事件总结一下Android事件响应方式。

方式1：匿名接口实现（常规）

```
Button bt=(Button)findViewById(R.id.button1);  
bt.setOnClickListener( new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        // 代码  
    }  
});
```

方式2：内部类实现接口

```
public class MainActivity extends Activity {
```

外部类

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
```

```
        Button bt=(Button)findViewById(R.id.button1);
```

```
        bt.setOnClickListener( new mybuttonlistener() );
```

```
    }
```

内部类

```
private class mybuttonlistener implements View.OnClickListener{
```

```
    @Override
```

```
    public void onClick(View v) {
```

```
        // 事件代码
```

```
    }
```

```
}
```

```
}
```

内部类不要
写到其他方
法里面了

内部类声明为private
表明该内部类只能被其
外部类中的方法操作

内部类要实现监听接口的

方式3：绑定监听器（也较常用）

//先定义一个监听器

```
View.OnClickListener buttonListener = new View.OnClickListener(){  
    @Override  
    public void onClick(View v) {  
        switch( v.getId() ){  
            case R.id.button1: ...  
            case R.id.button2: ...  
        }  
    };  
};
```

//然后将监听器绑定给对象

```
button1.setOnClickListener(buttonListener);  
button2.setOnClickListener(buttonListener);
```

优势： 这种方式允许多个按钮注册 到同一个事件监听器上，实现代码重用

示例：多个按钮注册到同一个事件的监听器上

添加

删除

修改

查询

```
View.OnClickListener buttonListener = new View.OnClickListener(){
    @Override
    public void onClick(View v) {
        switch( v.getId() ){ ←
            case R.id.button1:
                //添加记录
                break;
            case R.id.button2:
                //删除记录
                break;
            case R.id.button3:
                //修改记录
                break;
            case R.id.button4:
                //查询操作
                break;
            default:
                //其他操作
        }
    }
};
```

根据组件的id来判断哪个按钮被按下

```
Button bt1=(Button)findViewById(R.id.button1);
Button bt2=(Button)findViewById(R.id.button2);
Button bt3=(Button)findViewById(R.id.button3);
Button bt4=(Button)findViewById(R.id.button4);

bt1.setOnClickListener(buttonListener);
bt2.setOnClickListener(buttonListener);
bt3.setOnClickListener(buttonListener);
bt4.setOnClickListener(buttonListener);
```

方式4: activity实现接口

```
public class MainActivity extends Activity implements View.OnClickListener{  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        ...  
        Button bt=(Button)findViewById(R.id.button1);  
        bt.setOnClickListener( this );  
    }  
  
    @Override  
    public void onClick(View v) {  
        switch( v.getId() ){  
            case R.id.button1: ...  
            case R.id.button2: ...  
        }  
    }  
}
```

方式5：在组件的属性中绑定事件(紧耦合不推荐)

```
<Button  
    android:id="@+id/button1"  
    android:text="提交"  
    android:onClick="bt1Click"/>
```

```
public class MainActivity extends Activity {  
    public void bt1Click(View view){    //xml中绑定的事件  
        // 处理事件代码  
    }  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        ...  
    }  
}
```

【完】

附录1：动态添加菜单（以上下文菜单为例）

```
TextView tv1=(TextView)findViewById(R.id.textView1);    registerForContextMenu(tv1);
TextView tv2=(TextView)findViewById(R.id.textView2);    registerForContextMenu(tv2);
public void onCreateContextMenu(ContextMenu menu, View v, ContextMenuInfo menuInfo) {
    switch( v.getId() ){
        case R.id.textView1:
            menu.setHeaderTitle("TextView1");
            menu.add(0, 1, "菜单1");
            menu.add(0, 2, "菜单2");
            break;
        case R.id.textView2:
            menu.setHeaderTitle("TextView2");
            menu.add(1, 3, "菜单3");
            menu.add(1, 4, "菜单4");
            break;
    }
}
```

Menu.add(int groupId, int itemId, int order, CharSequence title)

自定义的菜单id值

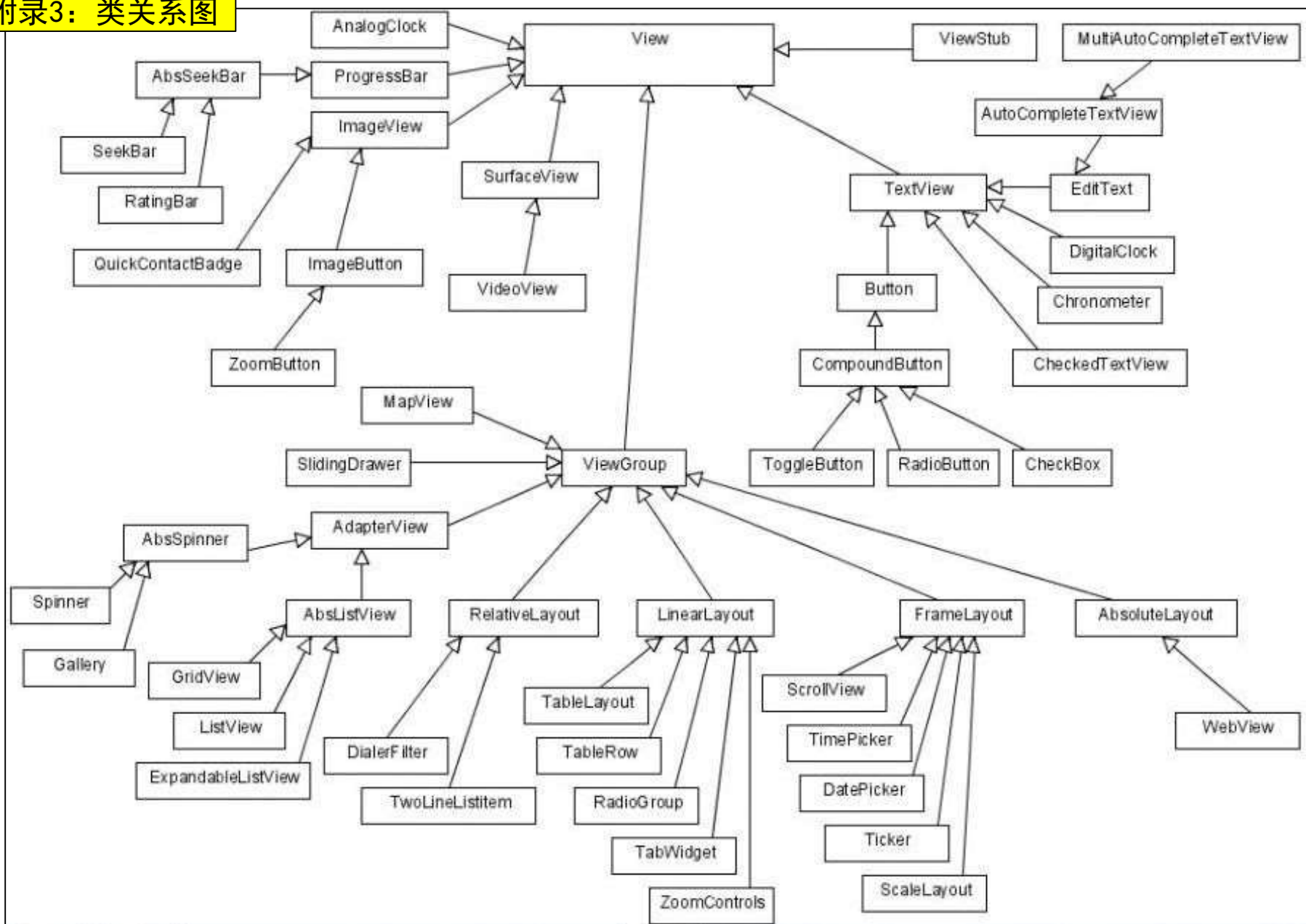
动态添加菜单（以上下文菜单为例）

```
public boolean onContextItemSelected(MenuItem item) {  
    switch( item.getItemId() ){  
        case 1: //菜单id=1  
            //菜单1操作  
            return true;  
        case 3: //菜单id=3  
            //菜单3操作  
            return true;  
    }  
    return false;  
}
```


附录2：Android剪贴板用法

```
public void copy(String content, Context context) {  
    ClipboardManager cmb = (ClipboardManager)  
        context.getSystemService(Context.CLIPBOARD_SERVICE);  
    cmb.setPrimaryClip( ClipData.newPlainText( null, content.trim() ) );  
}  
  
public String paste(Context context) {  
    ClipboardManager cmb = (ClipboardManager)  
        context.getSystemService(Context.CLIPBOARD_SERVICE);  
    if ( cmb.hasPrimaryClip() ){  
        return cmb.getPrimaryClip().getItemAt(0).getText().toString().trim();  
    }  
    return "";  
}
```

附录3：类关系图



附录4：AlertDialog用法详解

- 确定取消对话框
- 多个按钮信息框
- 单选列表框对话框
- 复选列表框对话框
- 自定义布局的对话框



确定取消对话框

```
new AlertDialog.Builder(AlertDialogActivity.this)
    .setIcon(R.drawable.ic_launcher)
    .setTitle("提示")
    .setMessage("确定退出? ")
    .setPositiveButton("确定",new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            finish();
        }
    })
    .setNegativeButton("取消", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            Toast.makeText(getApplicationContext(), "真的不想退出呀!", Toast.LENGTH_SHORT ).show();
        }
    })
    .show();
```



多个按钮信息框

```
new AlertDialog.Builder(AlertDialogActivity.this)
    .setTitle("提示")
    .setMessage("确定退出? ")
    .setPositiveButton("足球", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            Toast.makeText(getApplicationContext(), "你选择了足球", Toast.LENGTH_SHORT).show();
        }
    })
    .setNegativeButton("篮球", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            Toast.makeText(getApplicationContext(), "你选择了篮球", Toast.LENGTH_SHORT).show();
        }
    })
    .setNegativeButton("羽毛球", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            Toast.makeText(getApplicationContext(), "你选择了羽毛球", Toast.LENGTH_SHORT).show();
        }
    })
    ).show();
```



单选列表框对话框



```
final String[] items = {"刘德华", "张家辉", "郭富城", "文章", "章子怡", "赵薇", "孙俪", "杨幂"};
```

```
new AlertDialog.Builder(AlertDialogActivity.this)
    .setTitle("请选择你最喜欢的影星")
    .setIcon(android.R.drawable.ic_dialog_info)
    .setSingleChoiceItems(items, -1,
        new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(getApplicationContext(), "你选择了" + items[which], Toast.LENGTH_SHORT).show();
                //dialog.dismiss();
            }
        })
    .setNegativeButton("取消", null)
    .show();
```

默认选择项 id, -1 表示不选中

复选列表框对话框



```
final String[] items = {"刘德华", "张家辉", "郭富城", "文章", "章子怡", "赵薇", "孙俪", "杨幂"};
ArrayList<Integer> MultiChoiceID = new ArrayList<Integer>();

MultiChoiceID.clear();
new AlertDialog.Builder(AlertDialogActivity.this)
    .setTitle("请选择你最喜欢的影星")
    .setIcon(android.R.drawable.ic_dialog_info)
    .setMultiChoiceItems(items, null, new DialogInterface.OnMultiChoiceClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which, boolean isChecked) {
            if (isChecked) {
                MultiChoiceID.add(new Integer(which));
            } else {
                MultiChoiceID.remove(new Integer(which));
            }
        }
    })
    .setPositiveButton("确定", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            String str = "";
            int size = MultiChoiceID.size();
            for (int i = 0; i < size; i++) { str += items[MultiChoiceID.get(i)] + ", "; }
            Toast.makeText(getApplicationContext(), "你选择了" + str, Toast.LENGTH_SHORT).show();
        }
    })
    .setNegativeButton("取消", null)
    .show();
```

自定义布局的对话框



加载自定义的布局文件方法:

LayoutInflater factory =

`LayoutInflater.from(AlertDialogActivity.this);`

final View alertDialogView =

`factory.inflate(R.layout.testview, null);`

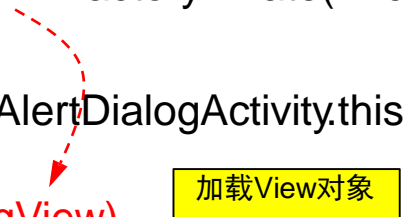
自定义的布局testview.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:orientation="horizontal"
    android:id="@+id/dialog">
    <LinearLayout
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:orientation="horizontal"
        android:id="@+id/dialogname">
        <TextView android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:id="@+id/tvUserName" android:text="姓
            名: " />
        <EditText android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:id="@+id/etUserName"
            android:minWidth="200dip"/>
        </LinearLayout>
        <LinearLayout
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:orientation="horizontal"
            android:id="@+id/dialognum"
            android:layout_below="@+id/dialogname"
            >
            <TextView android:layout_height="wrap_content"
                android:layout_width="wrap_content"
                android:id="@+id/tvPassWord"
                android:text="密码: " />
            <EditText android:layout_height="wrap_content"
                android:layout_width="wrap_content"
                android:id="@+id/etPassWord"
                android:minWidth="200dip"/>
            </LinearLayout>
        </RelativeLayout>
```


自定义布局的对话框

```
LayoutInflater factory = LayoutInflater.from(AlertDialogActivity.this);
final View alertDialogView = factory.inflate(R.layout.testview, null);

new AlertDialog.Builder(AlertDialogActivity.this)
    .setTitle("请输入")
    .setView(alertDialogView)
    .setPositiveButton("确定", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            EditText userName = (EditText) alertDialogView.findViewById(R.id.etUserName);
            EditText password = (EditText) alertDialogView.findViewById(R.id.etPassWord);
            Toast.makeText(getApplicationContext(), "姓名 : " + userName.getText().toString+ "密码: "
                + password.getText().toString(), Toast.LENGTH_SHORT).show();
        }
    })
    .setNegativeButton("取消", null)
    .show();
```



附录5: AutoCompleteTextView用法

■ 基本XML描述



<AutoCompleteTextView

`android:id="@+id/autoCompleteTextView1"` ← id

`android:hint="请输入省份"` ← 设置提示文本

`android:completionThreshold="1"` ← 设置输入几个以上字符时进行匹配

`android:text="AutoCompleteTextView" />`

← 一般不要text值，可删除

AutoCompleteTextView示例

(1) 在res/values/**arrays.xml**中定义:

```
<string-array name="province_array">  
    <item>北京</item>  
    <item>上海</item>  
    <item>湖北省</item>  
    <item>湖南省</item>  
    <item>山东省</item>  
    <item>山西省</item>  
    <item>广东省</item>  
    <item>广西省</item>  
</string-array>
```



AutoCompleteTextView示例

(2) 在MainActivity中添加代码：

使用ArrayAdapter填充数据的另一种方法

```
ArrayAdapter<CharSequence> adapter =  
    ArrayAdapter.createFromResource( ← 使用已有资源来创建适配器  
        MainActivity.this, R.array.province_array, ← 资源  
        android.R.layout.simple_spinner_dropdown_item); ← 样式  
AutoCompleteTextView ac=(AutoCompleteTextView)  
    findViewById(R.id.autoCompleteTextView1);  
ac.setAdapter(adapter);
```

附录6: DatePicker用法



宽屏全显示(默认)

基本XML描述:

<DatePicker

android:id="@+id/datePicker1"

android:calendarViewShown="false"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_alignParentLeft="true"

android:layout_alignParentTop="true" />

不显示右侧日历视图

DatePicker基本用法

```
Button bt=(Button)findViewById(R.id.button1);
bt.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View arg0) {
        DatePicker dp=(DatePicker)findViewById(R.id.datePicker1);
        Toast.makeText(MainActivity.this, "设置的日期: " + dp.getYear() + "
            年" + (dp.getMonth()+1) + "月" + dp.getDayOfMonth() + "日",
            Toast.LENGTH_SHORT).show();
    }
});
```



DatePicker init()方法

```
Calendar ca = Calendar.getInstance(); //获得当前系统日期
```

```
int year = ca.get(Calendar.YEAR);
```

```
int month = ca.get(Calendar.MONTH);
```

```
int day = ca.get(Calendar.DAY_OF_MONTH);
```

获取年、月、日

```
DatePicker dp=(DatePicker)findViewById(R.id.datePicker1);
```

```
dp.setCalendarViewShown(false);
```

不显示右边的日历(API11)

```
✓ dp.init ( year, month, day, new DatePicker.OnDateChangedListener() {
```

```
@Override
```

```
public void onDateChanged(DatePicker view, int year, int monthOfYear,int dayOfMonth) {
```

```
    Toast.makeText(MainActivity.this, "设置的日期: " + year +  
        "年" + (monthOfYear+1) + "月" + dayOfMonth + "日",
```

```
        Toast.LENGTH_SHORT).show();
```

```
}
```

```
});
```

第四个参数一般
可设置为null



DatePickerDialog用法

- (1) 在界面上放置一个TextView和Button Button弹出
DatePickerDialog来设置日期 TextView用于显示日期



(2) 添加如下代码:

```

public class MainActivity extends Activity {
    static final int DATE_DIALOG_ID = 1;
    //用来保存年月日:
    private int year;    private int month;    private int day;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Calendar currentDate = Calendar.getInstance();
        year = currentDate.get(Calendar.YEAR);
        month = currentDate.get(Calendar.MONTH);
        day = currentDate.get(Calendar.DAY_OF_MONTH);

        //单击按钮打开一个Dialog
        Button bt=(Button)findViewById(R.id.button1);
        bt.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showDialog(DATE_DIALOG_ID);
            }
        });
    }
}
//end onCreate

```

声明一个标识常量，来作为要显示DatePicker的Dialog的ID

获取当前的年、月、日
后面用于创建DatePickerDialog

调用Activity类的showDialog方法来显示Dialog
同时Activity类会调用onCreateDialog(int)来请求
一个Dialog (见后页)

(2) 添加如下代码：续

@Override

protected Dialog onCreateDialog(int id) {

switch (id) {

case DATE_DIALOG_ID:

return new DatePickerDialog(MainActivity.this, myDateSetListener, year, month, day);

}

return null;

} //end onCreateDialog

当Activity调用showDialog函数时会触发该函数的调用

创建一个DatePickerDialog对话框

定义弹出的DatePicker对话框的事件监听器

private DatePickerDialog.OnDateSetListener myDateSetListener=new DatePickerDialog.OnDateSetListener() {

@Override

public void onDateSet(DatePicker view, int year, int monthOfYear,int dayOfMonth) {

MainActivity.this.year=year;

MainActivity.this.month=monthOfYear;

MainActivity.this.day=dayOfMonth;

保存最新设置的年月日

TextView tv=(TextView)findViewById(R.id.textView1);

tv.setText(year+"年"+(monthOfYear+1)+"月"+dayOfMonth+"日");

让TextView显示

}

};

} //end activity