#4 界面编程（下）

之UI控件

Android基本控件

控件是界面的主要组成元素。它是用户界面功能实现的表现。Android基本控件是开发Android程序必要的工具类。

常用控件如下：

- TextView(文本框)
- Button(按钮)
- CheckBox(复选框)
- EditText(编辑框)
- RadioButton(单选框)
- ImageView(图片控件)

控件概述

Android系统的界面控件分为系统控件和自定义控件

- 系统控件：Android系统提供给用户已经封装的界面控件。提供在应用程序开发过程中常见功能控件。系统控件更有利于帮助用户进行快速开发,同时能够使Android系统中应用程序的界面保持一致性;
- 自定义控件：用户独立开发的控件,或通过继承并修改系统控件后所产生的新控件。能够为用户提供特殊的功能或与众不同的显示需求方式

控件概述

Android的控件,一般是在res/layout下的布局文件中声明。每个控件都有自己的id、显示的宽度(layout_width)和高度(layout_height)。

除此之外,每个控件还有自己特有的属性,进一步的规范了控件在布局中的显示效果。控件就是通过这些属性的设置来实现的。

控件概述

设置控件的属性有两种方法：

- 在布局文件中设置参数
- 在代码中调用对应方法实现

例如:设置TextView的文本内容,一是在布局文件中设置TextView的text属性; 另一种是在代码中,调用TextView.setText()方法。

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/app_name"/>
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/tv"/>
```

```
TextView tv = (TextView)findViewById(R.id.tv);
tv.setText(R.string.app_name);
```

方法和事件的使用

控件在布局文件中声明后,在 Activity中,使用 *super.setContentView(R.layout.某布局文件名)* 加载布局文件。在 Activity 中获取控件的引用,需使用 *super.findViewById(R.id.控件id)* ,接着就可以使用该引用对控件进行操作,例如添加监听,设置内容等。

Activity是Android中最常用的组件,在一个Android应用中,一个 Activity就是一个单独的界面。每一个Activity被给予一个窗口,在上面可以添加任意控件。窗口通常充满屏幕,但也可以小于屏幕而浮于其它窗口之上。

文本类控件

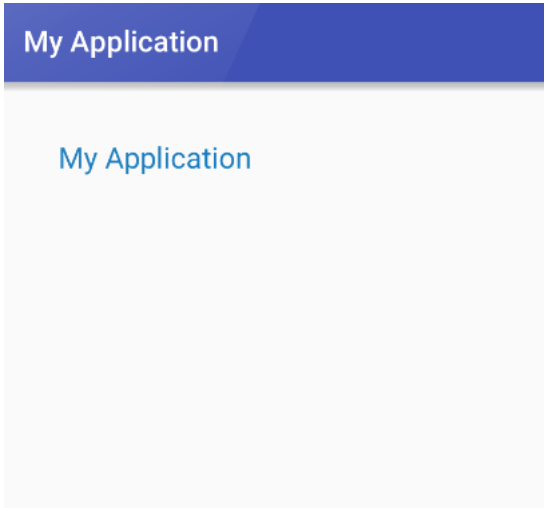
文本类控件是Android程序开发中最常用的控件之一。主要包含两大类———TextView和EditText, 都可以用来显示文本。

文本类控件 TextView

- TextView是一种用于显示字符串的控件。它本身不具备可编辑属性，但可以在代码中通过调用对应方法，编辑文本内容。
- TextView控件中包含很多可以在XML文件中设置的属性，这些属性同样可以在代码中动态声明

文本类控件 TextView

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/tv"
    android:text="@string/app_name"
    android:textSize="20sp"
    android:textColor="@android:color/suggestion_highlight_text"
    />
```



文本类控件 EditText

EditText是用来输入和编辑字符串的控件，它是一个具有编辑功能的 TextView。EditText是TextView的子类，除了TextView的一些属性外，EditText还有一些属性。

- android:hint=“请输入内容” //文本提示内容
- android:inputType=“” //输入类型

文本类控件 EditText

```
<EditText
    android:layout_width="0dp"
    android:layout_weight="1.0"
    android:layout_height="wrap_content"
    android:id="@+id/tv"
    android:hint="@string/input_hint"
    android:textSize="20sp"
    android:textColor="@android:color/darker_gray"
    />
```

Button类控件

在Android程序开发中，Button类控件是较为常用的一类控件。

主要包括: Button、ImageButton、RadioButton和 CheckBox等。

Button类控件 button

- Button是TextView的子类,具有TextView的所有属性。用户可以通过单击 Button来触发一系列事件,然后为Button注册监听器来实现其监听事件。

- 为Button注册监听有两种方法:
  1. 在布局文件中,为Button控件设置OnCilck属性,然后在代码中添加一个public void 参数值{}方法;
     布局文件中,Button属性添加 android:onClick= “click”
     代码文件中,添加 public void click(){.....}
  2. 在代码中绑定匿名监听器,并且重写onClick方法。

Button类控件 button

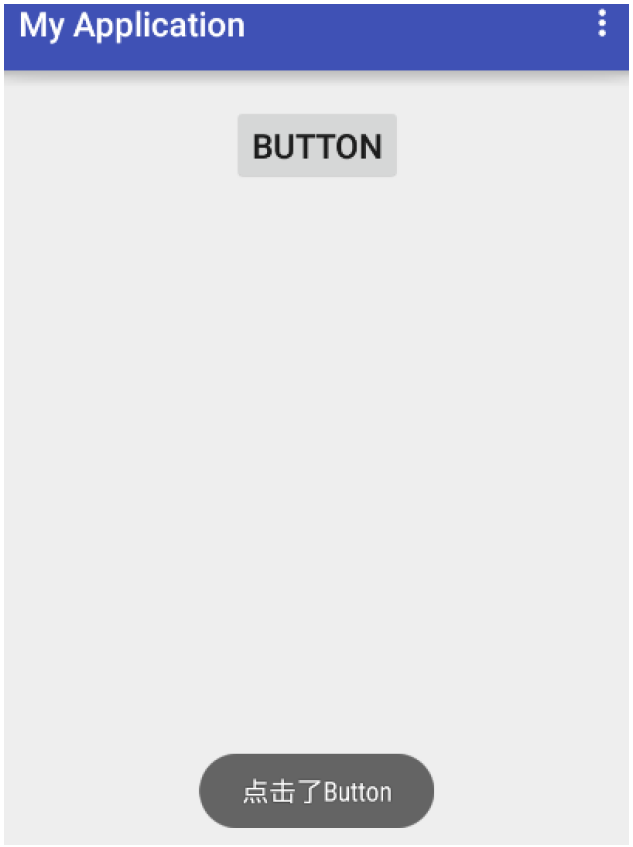
方式一（不推荐）

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:backgroundTint="#ffff9800"
    android:id="@+id/button"
    android:text="@string/btn"
    android:textSize="20sp"
    android:onClick="click"
    />
```

layout.xml

```
public void click(View target) {
    Toast.makeText(getApplication(),R.string.click_msg,
            Toast.LENGTH_LONG).show();
}
```

Activity.java



点击后弹出toast

Button类控件 button

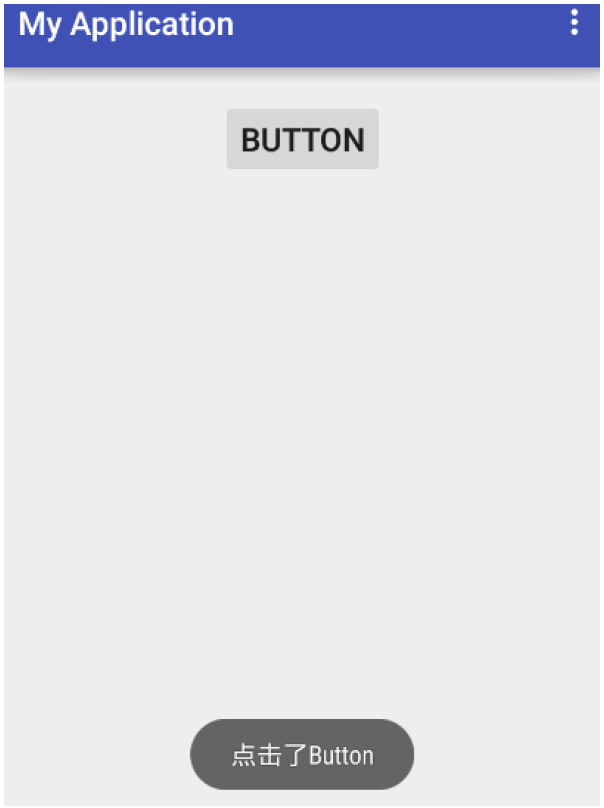
方式二 代码绑定监听器

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

    Button btn = (Button)findViewById(R.id.button);

    btn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Toast.makeText(getApplication(),R.string.click_msg,
                    Toast.LENGTH_LONG).show();
        }
    });
```

Activity.java

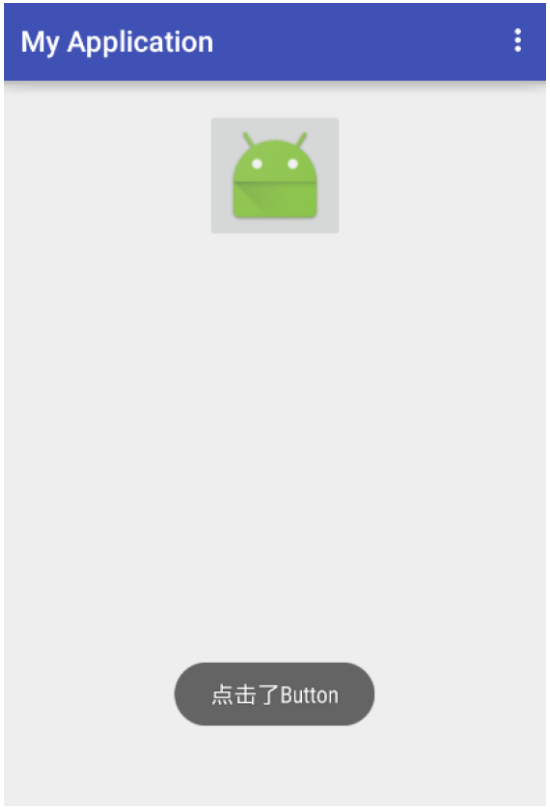


Button类控件 ImageButton

ImageButton控件与Button控件的主要区别是ImageButton中没有text属性,按钮中将显示图片而不是文本。在该控件中,设置按钮显示的图片可以通过布局文件 android:src属性或通过代码setImageResource(int)方法来设置。

Button类控件 ImageButton

```
<ImageButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:src = "@drawable/ic_launcher"
    android:id="@+id/button"
    />
```

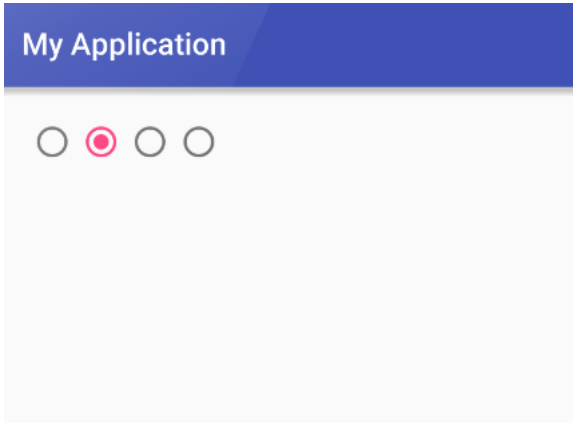


Button类控件 RadioButton

- 一种单个圆形单选框双状态的按钮,可以选择或不选择。在没有被选中时, 用户能够按下或点击来选中它。但在选中后, 通过点击无法变为未选中。
- 由两部分组成: RadioButton和RadioGroup。其中RadioGroup是单选组合框, 用于将RadioButton框起来。在多个RadioButton被RadioGroup包含的情况下,同一时刻仅可以选择一个RadioButton, 并用setOnCheckedChangeListener 来对RadioGroup进行监听。

Button类控件 RadioButton

```
<RadioGroup
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:checked="true"/>
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</RadioGroup>
```

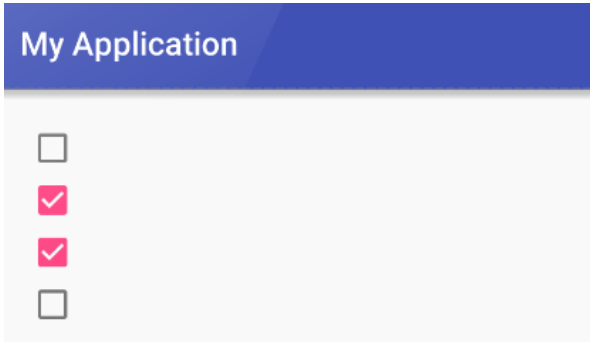


Button类控件 CheckBox

可以进行多选的按钮, 默认以矩形表示。与RadioButton相同,它也有选中或者不选中双状态。可以先在布局文件中定义多选按钮,然后对每一个多选按钮进行事件 监听setOnCheckedChangeListener, 通过isChecked来判断选项是否被选中, 作出相应的事件响应。

Button类控件 CheckBox

```
<CheckBox
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
<CheckBox
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="true"/>
<CheckBox
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="true"/>
<CheckBox
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```



图片控件 ImageView

显示任意图像，例如图标。ImageView类可以加载各种来源的图片，图片的来源可以是系统提供的资源文件,也可以是Drawable对象或Bitmap对象。 提供例如缩放和着色（渲染）各种显示选项。

图片控件 ImageView

ImageView中XML属性src和background的区别：

- background会根据ImageView组件给定的长宽进行拉伸，而src就存放的是原图的大小，不会进行拉伸。src是图片内容（前景），bg是背景，可以同时使用。
- 此外：scaleType只对src起作用；bg可设置透明度，比如在ImageButton中就可以用android:scaleType控制图片的缩放方式

Android 高级控件

Android高级控件,是指具有更高级功能的控件。例如, 自动完成文本、进度条、ListView、Spinner、TabHost、GridView等等。这类控件丰富了界面的多样性,强化了程序的功能,更好的实现了Android应用程序。下面 将详细介绍这类控件。

自动完成文本控件

在Android中提供了两种智能输入框——AutoCompleteTextView和 MultiAutoCompleteTextView。它们的功能大致相同,类似于百度或者 Google在搜索栏输入信息的时候,弹出与输入信息接近的提示信息,然后 用户选择点击需要的信息,自动完成文本输入。

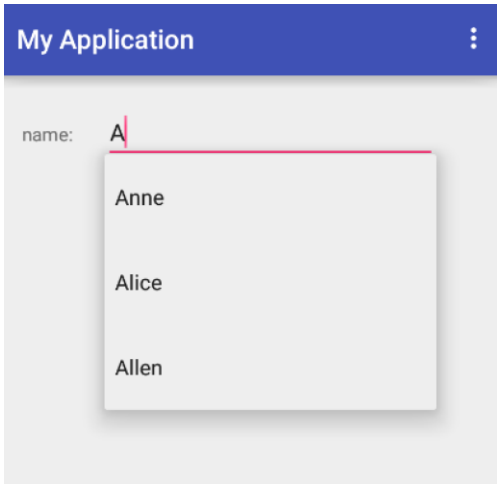
自动完成文本控件AutoCompleteTextView

能够实现动态匹配输入的可编辑的文本视图。当用户输入信息后弹出提示。

提示列表显示在一个下拉菜单中,用户可以从中选择一项,以完成输入。

提示列表是从一个数据适配器获取的数据。

自动完成文本控件AutoCompleteTextView



重要属性：

android:completionThreshold //输入多少字符后出现提示信息

android:dropDownHeight //下拉菜单的高度

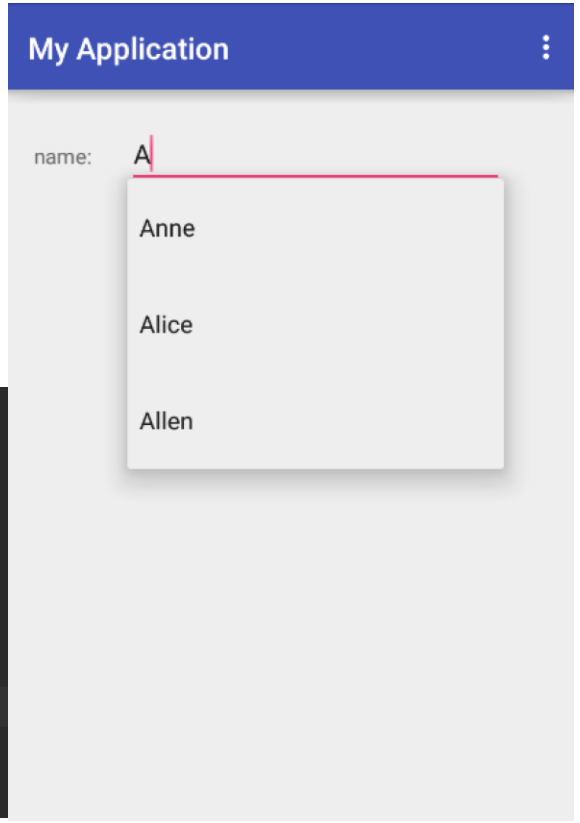
android:dropDownWidth //下拉菜单的宽度

android:popupBackground //下拉菜单的背景

自动完成文本控件AutoCompleteTextView

```
<AutoCompleteTextView
    android:id="@+id/auto_text"
    android:layout_width="250dp"
    android:layout_height="wrap_content"
    android:completionThreshold="1" />
```

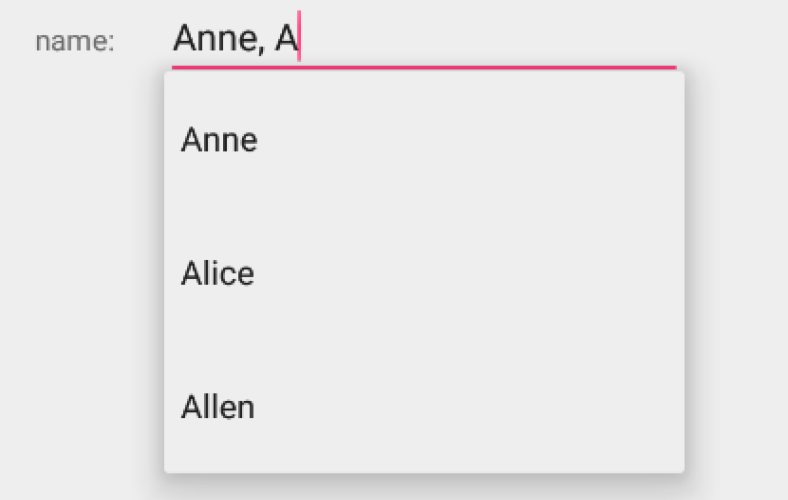
```
String[] names = {"Lily", "Lucy", "Cathy", "Anne" ,
        "Alice", "Allen","Mark"};
AutoCompleteTextView nameText =
        (AutoCompleteTextView)this.findViewById(R.id.auto_text);
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_dropdown_item_1line, names);
nameText.setAdapter(adapter);
```



多文本自动完成输入MultiAutoCompleteTextView

- 能够对用户键入的文本进行有效地扩充提示的可编辑文本视图。用户必须提供一个MultiAutoCompleteTextView.Tokenizer用来区分不同的子串。
- 与AutoCompleteTextView不同的是, MultiAutoCompleteTextView可以在输入框一直增加选择值。
- 对自动完成文本框的设置可以在XML文件中使用属性进行设置,也可以在Java代码中通过方法进行设置

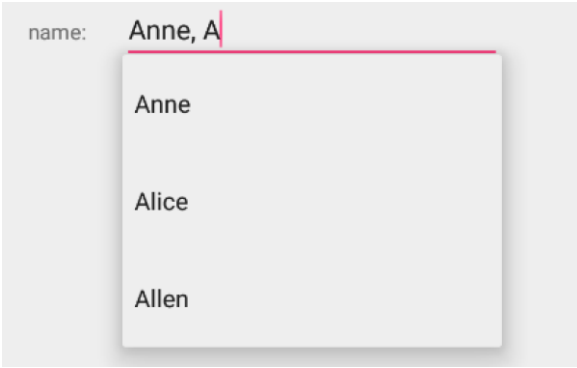
多文本自动完成输入MultiAutoCompleteTextView



当输入完一个字符串后,在该字符串后面输入一个逗号(,),在逗号前后可以有任意多个空格,然后再输入 一个字符串,仍然会显示自动提示列表。使用MultiAutoCompleteTextView时,需要为它的setTokenizer方法指定 MultiAutoCompleteTextView.CommaTokenizer类对象实例,该对象表示采用逗号作为输入多个字符串的分隔符。

多文本自动完成输入MultiAutoCompleteTextView

```
<MultiAutoCompleteTextView
    android:layout_width="250dp"
    android:layout_height="wrap_content"
    android:completionThreshold="1"
    android:id="@+id/auto_text"/>
```




```
String[] names = {"Lily", "Lucy", "Cathy", "Anne" ,
        "Alice", "Allen","Mark"};
MultiAutoCompleteTextView nameText =
        (MultiAutoCompleteTextView) this.findViewById(R.id.auto_text);
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_dropdown_item_1line, names);
nameText.setAdapter(adapter);
//设置分隔符
nameText.setTokenizer(new MultiAutoCompleteTextView.CommaTokenizer());
```

进度条与拖动条

拖动条主要是完成于用户的简单交互,而进度条则是需要长时间加载某些资源时,为用户显示加载的进度的控件。

进度条ProgressBar

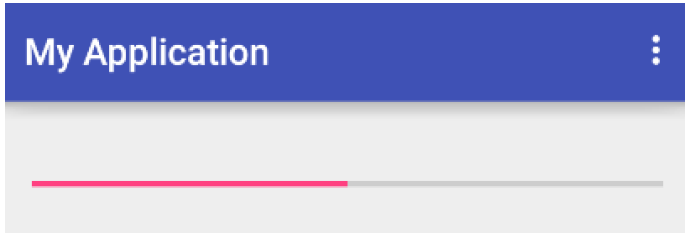
- 是某些操作的进度可视指示器,为用户呈现操作的进度。它还有一个次要的进度条,用来显示中间进度, 如在流媒体播放的缓冲区的进度。也可不确定其进度。在不确定模式下,进度条显示循环动画,常用于任务长度是未知的情况。
- 进度条默认为圆圈形式,要显示水平进度条,可以在xml文件中设置进度条的 style属性。

除了有水平进度条,还有圆形进度条,包括*Large*、*Norma*l和*Smal*l三种规格

进度条ProgressBar

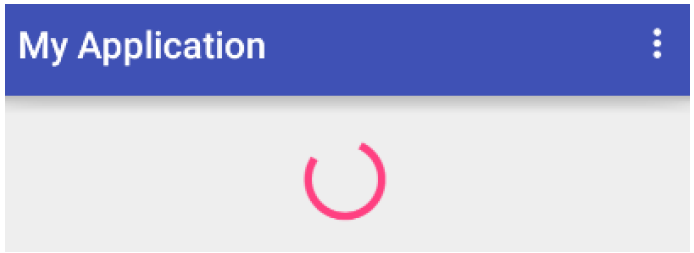
```
<ProgressBar
    android:layout_width="fill_parent"
    android:layout_height="50dp"
    style="?android:attr/progressBarStyleHorizontal"
    android:id="@+id/download_bar"
    android:max="100"/>
```

```
//设置当前进度，默认最大值是100
progressBar.setProgress(50);
```



进度为50的水平进度条

style=“?android:attr/progressBarStyleHorizontal“
水平进度条
style=“?android:attr/progressBarStyleLarge“
较大的圆圈进度条
style=“?android:attr/progressBarStyleSmallTitle
标题大小的圆圈进度条



较大的圆圈进度条

拖动条SeekBar

添加了滑块的进度条, 用户可以通过拖动滑块, 来调节当前进度。例如可以拖动滑块, 调节调节音量的大小。为了让程序能响应拖动条滑块位置的改变, 可以为它绑定一个OnSeekBarChangeListener监听器。

拖动条SeekBar

```
<SeekBar
    android:layout_width="fill_parent"
    android:layout_height="50dp"
    android:id="@+id/seek_bar"
    android:progress="30"
    android:max="100"/>
```

Android:thumb//设置滑块的样式

Android:max//设置拖动条进度的最大值

Android:progress//设置拖动条当前的进度值

拖动条SeekBar

```
seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
        //进度发生变化
        textView.setText("当前进度为：" + Integer.toString(progress));
    }

    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {
        //用户开始拖动SeekBar
    }

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {
        //用户结束拖动SeekBar
    }
});
```

评分条RatingBar

用星型来显示等级评定。默认显示5颗星,用户可以通过触屏点击或者轨迹球左右移动来进行星型等级评定 。

主要属性：

- android:isIndicator //是否是一个指示器(值为“true”时用户无法进行更改)
- android:numStars //显示的星型数量,必须为整数
- android:rating //默认的评分,必须是浮点型
- android:stepSize //评分的步长,一次增加或减少的星型数量是这个数字的整数倍,必须是浮点型

评分条RatingBar

```
<RatingBar
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    style="@style/Widget.AppCompat.RatingBar.Small"
    android:numStars="4"
    android:rating="1.5"/>
<RatingBar
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    style="@style/Base.Widget.AppCompat.RatingBar.Indicator"
    android:layout_marginTop="10dp"
    android:numStars="5"
    android:rating="3"/>
<RatingBar
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    style="@style/Widget.AppCompat.RatingBar"
    android:layout_marginTop="10dp"
    android:numStars="6"
    android:rating="5"/>
```

风格：

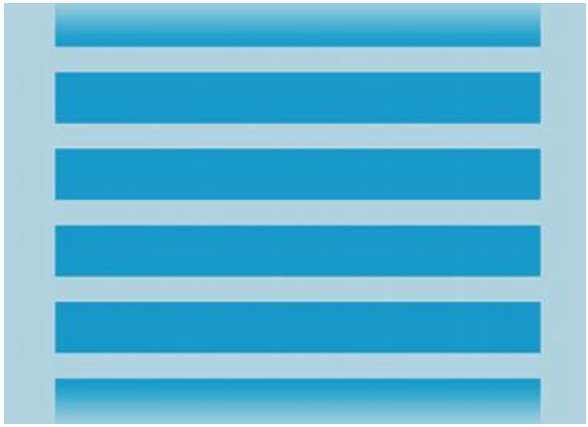
RatingBarStyleSmall

RatingBarStyleIdicator

RatingBarStyle

列表ListView

是将数据显示在一个垂直且可滚动的列表中的一种控件。数据来源于与 ListView绑定的Adapter,包含图片,文本等内容。每一行数据为 一条item。



列表ListView

一个**ListView**要显示其相关内容,需要满足三个条件

1. 需要ListView显示的数据;
2. 与ListView相关联的适配器(Adapter);
3. 一个ListView对象。

ListView

列表ListView

适配器是一个连接数据和AdapterView（ListView就是一个典型的AdapterView）的桥梁，通过它能有效地实现数据与AdapterView的分离设置，使AdapterView与数据的绑定更加简便，修改更加方便。

ListView可用的适配器：

- ArrayAdapter用来绑定一个数组，支持泛型操作
- SimpleAdapter用来绑定在xml中定义的控件对应的数据
- SimpleCursorAdapter用来绑定游标得到的数据
- BaseAdapter通用的基础适配器

列表ListView

1. ArrayAdapter：是ListView中最简单的一种适配器,将一个数组和ListView之间 建立连接,可将数组里定义的内容对应显示在ListView中,每项一般只有一个TextView,即每行只显示一个数组Item调用toString()方法生成的一行字符串;

2. SimpleAdapter：可以让ListView中的每项内容可以自定义出各种效果,可以将ListView中某一项的布局信息直接写在一个单独的xml文件中,通过 R.layout.layout_name来获得该布局(layout_name是xml布局文件的名字);

3. SimpleCursorAdapter：是SimpleAdapter与数据库的简单结合,通过该适配器,ListView能方便显示对应数据库中的内容。

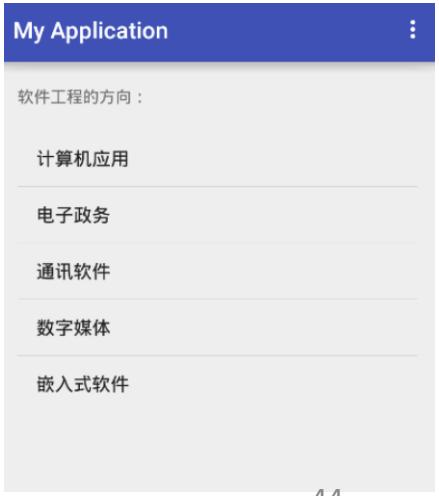
列表ListView

实例： arrayAdapter

（1）定义一个数组来存放ListView中item的内容。
（2）通过实现ArrayAdapter的构造函数来创建一个ArrayAdapter的对象。
（3）通过ListView的setAdapter()方法绑定ArrayAdapter。

ArrayAdapter有多个构造函数，例子中实现的是最常用的一种。第一个参数为上下文，第二个参数为一个包含TextView，用来填充ListView的每一行的布局资源ID。第三个参数为ListView的内容。其中第二个参数可以自定义一个layout，但是这个layout必须要有TextView控件。

```
ListView listView = (ListView)findViewById(R.id.list_view);
//定义数据源
String[] data = {"计算机应用","电子政务", "通讯软件",
        "数字媒体", "嵌入式软件"};
//新建ArrayAdapter, 并绑定到ListView
listView.setAdapter(new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1, data));
```



下拉列表Spinner

每次只显示用户选中的元素,当用户再次点击时,会弹出选择列表供用户选择,而选择列表中的元素来自适配器。

对Spinner的设置可以在XML文件中使用属性进行设置,也可以在Java代码中通过方法进行设置

Spinner.getItemAtPosition(Spinner.getSelectedItemPosition());获取下拉列表框的值

调用setOnItemSelectedListener()方法,处理下拉列表框被选择事件,把AdapterView.OnItemSelectedListener实 例作为参数传入

下拉列表Spinner

```
Spinner spinner = (Spinner)findViewById(R.id.spinner);
//定义数据源
String[] data = {"计算机应用","电子政务", "通讯软件",
        "数字媒体", "嵌入式软件"};
//新建ArrayAdapter，并绑定到Spinner
spinner.setAdapter(new ArrayAdapter<String>(this,
        android.R.layout.simple_spinner_dropdown_item, data));
//设置DropDown的位置偏移量
spinner.setDropDownVerticalOffset(100);
spinner.setDropDownHorizontalOffset(40);
```

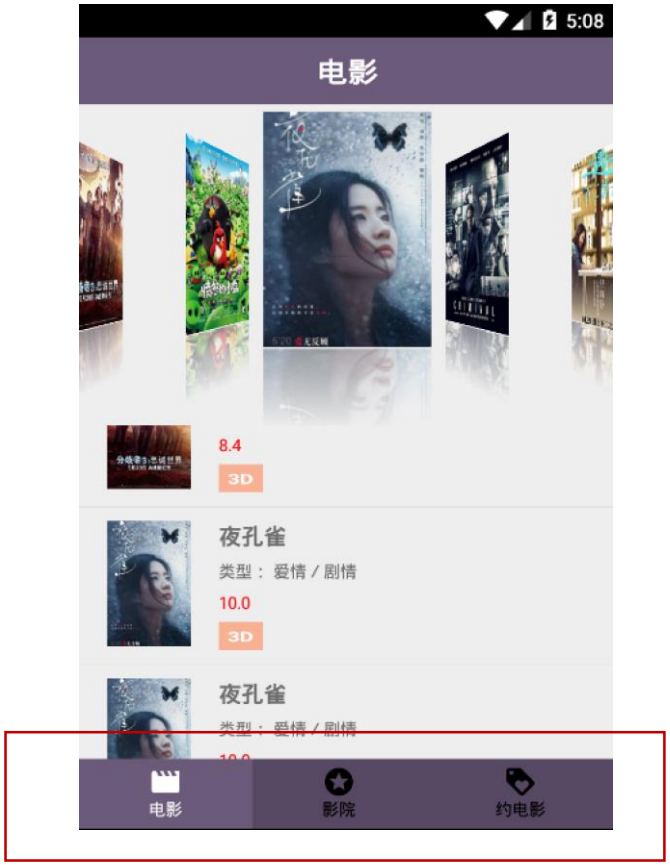
滚动视图ScollView

当一个屏幕不能完全显示所有需要显示的信息的情况下使用。支持垂直 滚动,当一个屏幕显示不下其中所包含的所有控件或信息时,便会自动添加滚动功能,来显示更多内容。

使用非常简单。由于ScollView实质上是一种帧布局,因此它的使用与布局的使用完全一致。然而ScollView只能拥有一个直接子类,所以在使用ScollView时,需要将其他布局嵌套在ScollView之内。

选项卡TabHost

选项卡(TabHost)控件可以实现多个标签样式的效果。单击每个选项卡,打开其对应的内容界面。TabHost是整个Tab的容器,包括两部分,TabWidget和FrameLayout。TabWidget就是每个tab的标签,FrameLayout则是 tab内容。



TabHost实现的选项卡

页面滑动切换控件ViewPager

Android的左右滑动在实际编程经常能用到,比如查看多张图片,左右切换 tab页。早期通用做法是使用ViewFlipper,自Android 3.0之后的SDK中提供了android-support-v4包用以实现版本兼容, 其中有一个可以实现左右滑动的类ViewPager。

ViewPager是android-support-v4.jar包中的一个系统控件,继承自ViewGroup,专门用以实现左右滑动切换View的效果。

TabLayout+ViewPager实现的Page效果

图片切换控件ImageSwitcher

在Windows 平台上要查看多张图片,最简单的办法就是通过“Window 图片和传真查看器”在“下一张”和“上一张”之间切换。Android平台上可以通过 ImageSwitcher 类来实现这一效果。ImageSwitcher 类必须设置一个ViewFactory,用来将显示的图片和父窗口区分开来,因此需要实现ViewSwitcher.ViewFactory接口。通过makeView()方法来显示图片,这里会返回一个ImageView 对象, 而方法 setImageResource用来指定图片资源。

窗体设置(Window)

应用程序界面始终限制在Android系统默认的框体之内,外部还有系统状态栏 和标题栏,为了能够更全面的操控Android的UI,可以设置应用程序窗体显示状态。 例如全屏显示、隐藏标题栏自定义标题等等。

主要使用requestWindowFeature(featureId)方法来设置,该方法的功能就是启动窗体的扩展特性,其参数值是Window类中定义的枚举常量。另外还可以通过Window类的setFlags()方法来隐藏系统的状态栏。当Activity设置了同时隐藏标题栏和状态栏时,就是全屏显示的状态了。

窗体设置(Window)

部分常用的常量如下：

DEFAULT_FEATURES:系统默认的状态;

FEATURE_CONTEXT_MENU:启动ContextMenu,默认启动该项;

FEATURE_XUSTOM_TITLE:当需要自定义标题的时候指定;

FEATURE_INDETERMINATE_PROGRESS:在标题栏上进度;

FEATURE_LEFT_ICON:标题栏左侧显示图标;

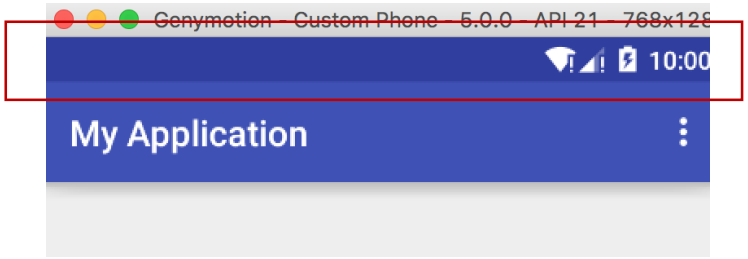
FEATURE_NO_TITLE:无标题栏;

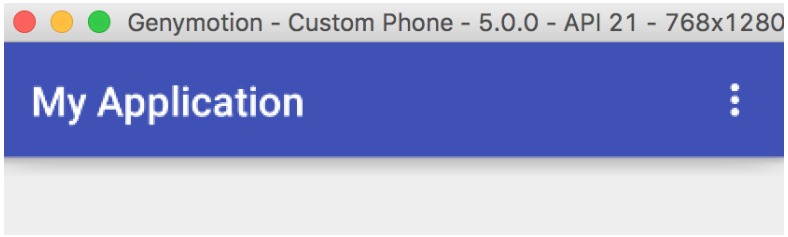
FEATURE_OPTION_PANEL:启动选项面板功能,默认启动;

FEATURE_PROGRESS:进度指示器功能;

FEATURE_RIGHT_ICON:标题栏右侧显示图标;

窗体设置(Window)






```
//设置隐藏状态
this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
        WindowManager.LayoutParams.FLAG_FULLSCREEN);
```

窗体设置(Window)

窗体设置(**Window**)之**xml**方法设置除了在java代码中实现之外,还可以直接在AndroidManifest.xml文件中的application元素中通过设置属性的方式来设置一部分窗体风格。

例如在application标签中加入

android:theme=“@android:style/Theme.NoTitleBar”隐藏标题栏。

在application标签中加入

android:theme="@android:style/Theme.NoTitleBar.Fullscreen "隐藏标题栏和状态栏,实现全屏显示的效果

对话框Dialog

在用户界面中,除了经常用到的菜单之外,对话框也是程序与用户进行 交互的主要途径之一。一个对话框一般是一个出现在当前Activity之上的一个小窗口,处于下面的Activity失去焦点。对话框接受所有的用户交互。对话框一般用于提示信息和与当前应用程序直接相关的小功能。

Android平台下的对话框,主要包括普通对话框、提示对话框、单选和 复选对话框、列表对话框、进度对话框、日期与时间对话框等。

对话框Dialog

1. Android dialog实现的方法有两种

a) 通过Dialog.Builder 初始化dialog 然后再showDialog

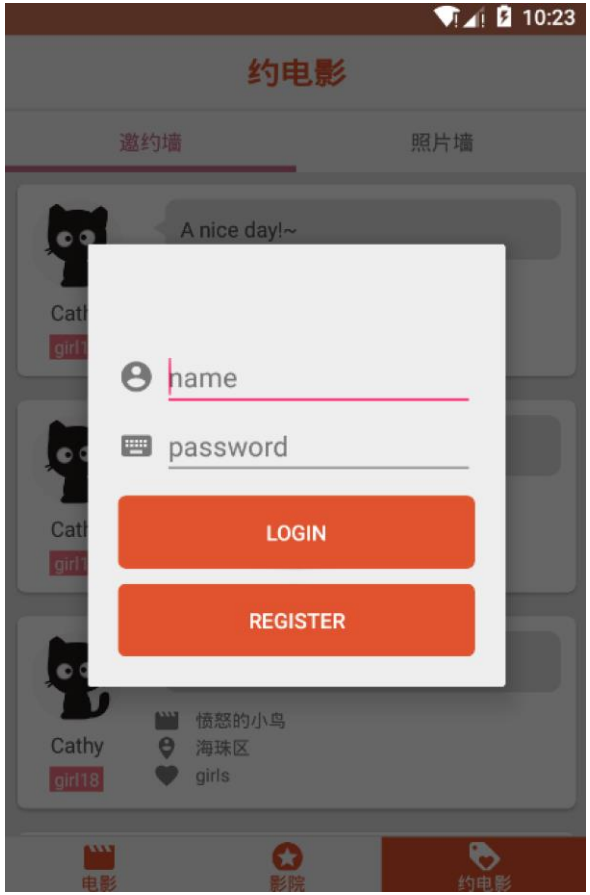
b) 通过将androidManifest.xml中的activity的属性设为android:theme= “@android:style/Theme.Dialog,伪装为dialog

2. showDialog的线程问题

Android dialog的显示不会阻塞ui线程

对话框Dialog

对话框通常是覆盖在当前Activity上面的小窗口,当对话框出现后,当前的 Activity将失去焦点,用户在此情况下只能与对话框进行交互。Android对话框能够 十分方便的进行创建、保存、回复和管理。



对话框Dialog AlertDialog与ProgressDialog

AlertDialog允许在对话窗口中添加最多3个按钮(positive, neutral,negative),还 可以包含一个提供了可选项的列表(如CheckBoxes或者RadioButtons等),它是实现Android中对话框的Dialog类的直接子类之一,在使用时AlertDialog对象通常不是通过构造 函数来新建,而是通过其内部静态类AlertDialog.Builder来进行构造的。

ProgressDialog是一个AlertDialog扩展类,是在AlertDialog的基础上加入表示进度的功能,通常表现为包含有进度条的对话框,因为其扩展于AlertDialog,AlertDialog的基本功能和用法在ProgressDialog上也基本一样。默认创建圈形进度条。

对话框Dialog AlertDialog

```
final AlertDialog.Builder alertDialog = new AlertDialog.Builder(this);
alertDialog.setTitle("标题").setMessage("消息").setPositiveButton("确认",
        new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(getApplicationContext(),
                        "你点击了确认",Toast.LENGTH_SHORT).show();
            }
        }).setNegativeButton("取消",
        new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(getApplicationContext(),
                        "你点击了取消",Toast.LENGTH_SHORT).show();
            }
        }).create();

Button btn = (Button) findViewById(R.id.btn);
if (btn != null)
    btn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            alertDialog.show();
        }
    });
```

上面代码采用的是个链式调用,像*setTitle()*、*setMessage()*这些方法,他们的返回值都是当前对话框对象

对话框Dialog AlertDialog

//设置【确定】按钮的方法两种重载形式

Public Builder setPositiveButton(CharSequence text,final OnClickListener listener)

Public Builder setPositiveButton(int textId,final OnClickListener listener)

//设置【取消】按钮的方法两种重载形式

Public Builder setNegativeButton(CharSequence text,final OnClickListener listener)

Public Builder setNegativeButton(int textId,final OnClickListener listener)

//设置【忽略】按钮的方法两种重载形式

Public Builder setNeutralButton(CharSequence text,final OnClickListener listener)

Public Builder setNeutralButton(int textId,final OnClickListener listener)

其中textId为文本资源的ID,可以使用资源ID,也可以直接使用文本text 第二个参数为实现DialogInterface.OnClickListener接口的对象实例

对话框Dialog ProgressDialog

```
Button btn = (Button) findViewById(R.id.btn);
if (btn != null)
    btn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            ProgressDialog.show(MainActivity.this,
                    "请稍等","正在加载...",true);
        }
    });
```

构造参数如下：

```
Context context, CharSequence title, CharSequence message
Context context, CharSequence title, CharSequence message, boolean indeterminate
Context context, CharSequence title, CharSequence message, boolean indeterminate, boolean cancelable
Context context, CharSequence title, CharSequence message, boolean indeterminate, boolean cancelable, OnCancelListener cancelListener
```

对话框Dialog 自定义布局

通过xml布局文件自定义一个布局,用于填充对话框;

LayoutInflater这个类的作用与findViewById(int ID)相似,不同的是LayoutInflater是用来寻找layout文件夹下的xml布局文件, 并且将这个布局实例化,而 findViewById()的作用是寻找项目中的某个xml文件下的具体widget控件(如:Button,TextView等)。

对话框Dialog 自定义布局

```
LayoutInflater inflater=getLayoutInflater();
View layout=inflater.inflate(R.layout.login_dialog,
        (ViewGroup)findViewById(R.id.dialog));

final AlertDialog.Builder alertDialog = new AlertDialog.Builder(this);

alertDialog.setTitle("登录").setView(layout).setPositiveButton("确认",
        new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(getApplicationContext(),
                        "你点击了确认",Toast.LENGTH_SHORT).show();
            }
        }).setNegativeButton("取消",
        new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(getApplicationContext(),
                        "你点击了取消",Toast.LENGTH_SHORT).show();
            }
        }).create();
```

QUESTIONS?