#6 界面编程（二）

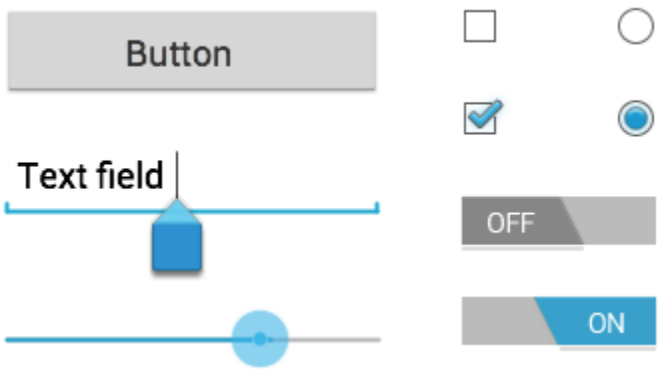
之UI控件

Android输入控件

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。

输入控件：输入控件是应用用户界面中的交互式组件。

如：按钮(Button)、文本字段(EditText)、定位栏(SeekBar)、复选框(CheckBox)、单选按钮(RadioButton)、切换按钮(ToggleButton)等。



控件概述

Android系统的界面控件分为系统控件和自定义控件，每个空间都有相应的属性。

- 系统控件：Android系统提供给用户已经封装的界面控件。
- 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

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);
```

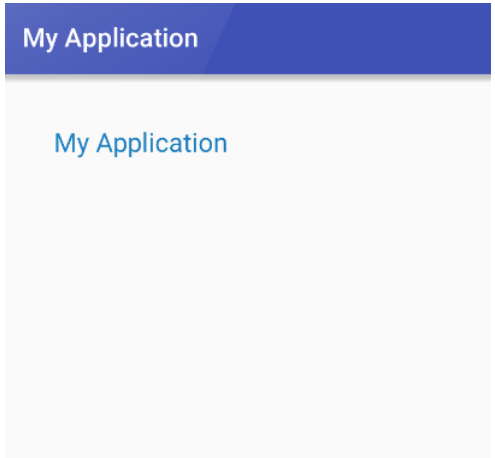
文本类控件

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
    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/holo_blue_light"/>
```



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=“phone” //显示为号码型
- android:inputType=“password”//显示为密码型

```
<EditText
    android:layout_width="match_parent"
    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= “showMessage”
     代码文件中,添加 public void showMessage(){.....}
  2. 在代码中绑定匿名监听器,并且重写onClick方法。

Button类控件 Button

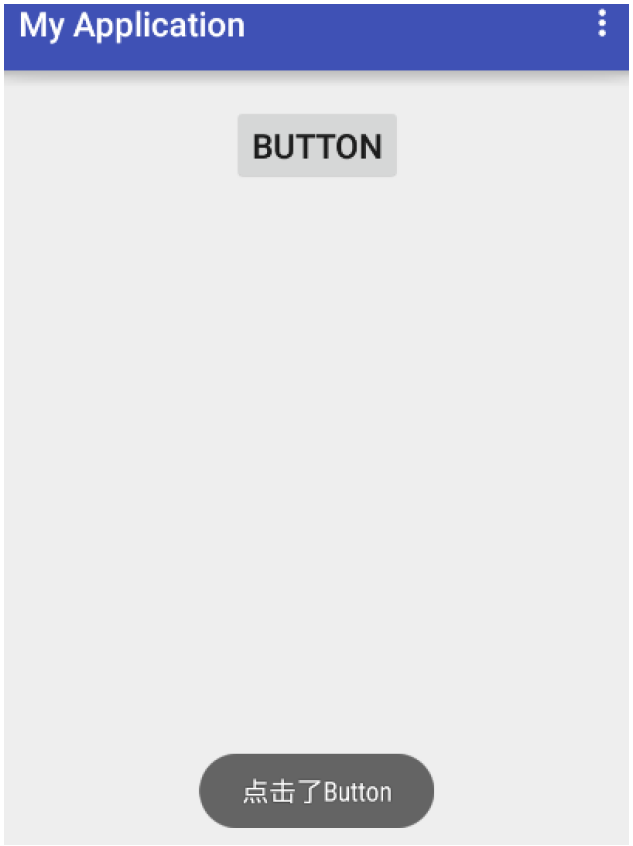
方式一 在xml文件中绑定点击函数

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

layout.xml

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

Activity.java



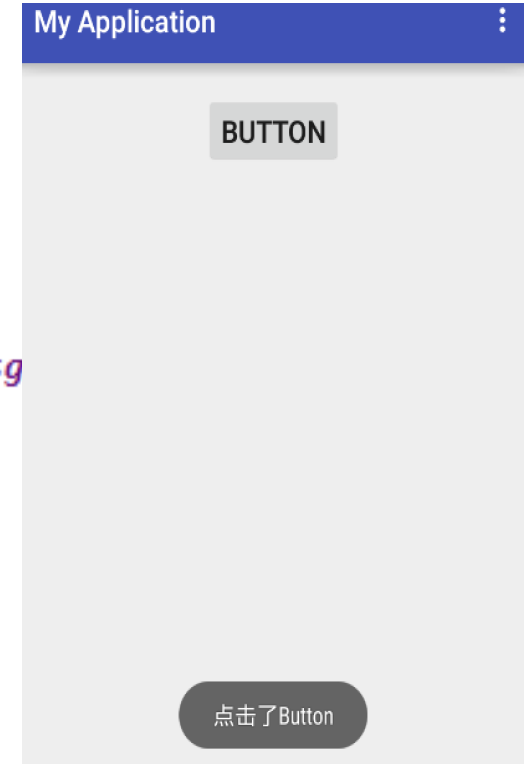
点击后弹出toast

Button类控件 Button

方式二 代码绑定监听器

```
Button btn = (Button) findViewById(R.id.button);
btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Toast.makeText(getApplication(), R.string.click_msg
                Toast.LENGTH_SHORT).show();
    }
});
```

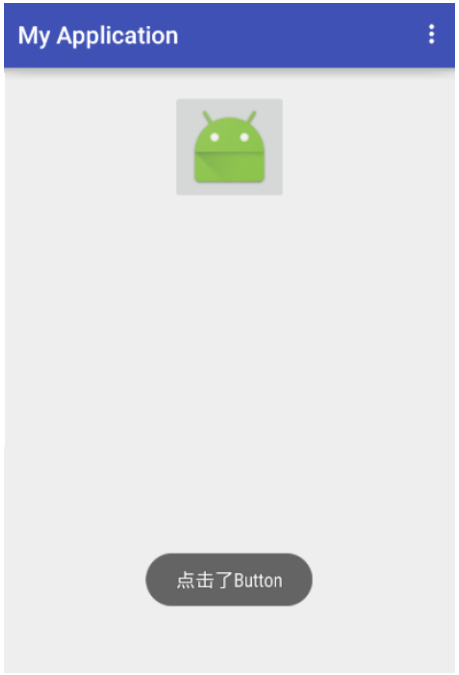
Activity.java



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。

```
<ImageButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@mipmap/ic_launcher"
    android:id="@+id/imgButton"/>
```

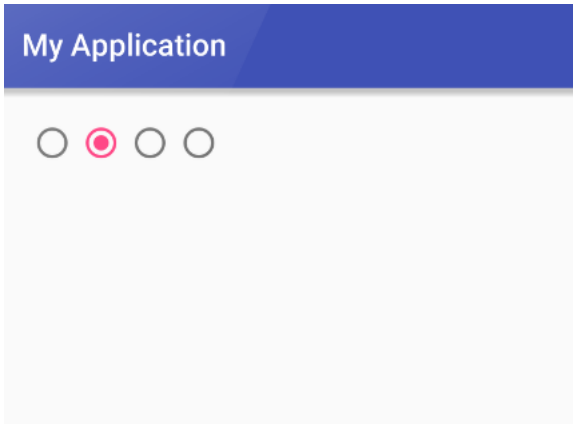


Button类控件 RadioButton

- 一种单个圆形单选框双状态的按钮,可以选择。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，呈被包含和包含的关系。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" />
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</RadioGroup>
```

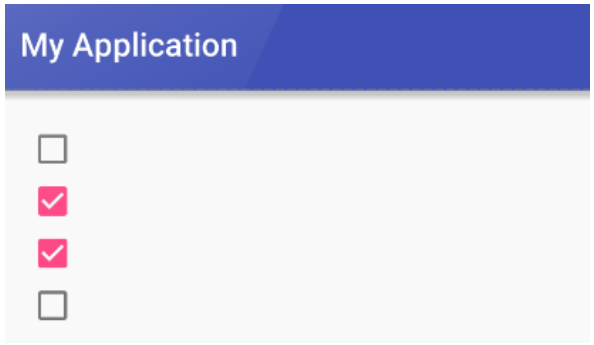


Button类控件 CheckBox

可以进行多选的按钮, 默认以矩形表示。有选中或者不选中双状态。

可以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。

```
<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。

Android 高级控件

Android高级控件,是指具有更高级功能的控件。例如, ListView、对话框等等。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。

自动完成文本控件

在Android中提供了两种智能输入框——AutoCompleteTextView和 MultiAutoCompleteTextView。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。

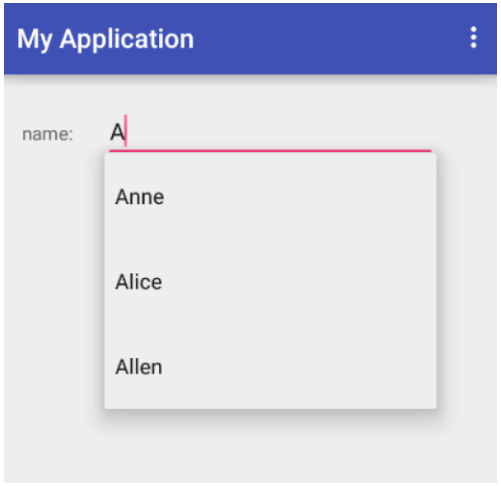
重要属性：

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

android:dropDownHeight //下拉菜单的高度

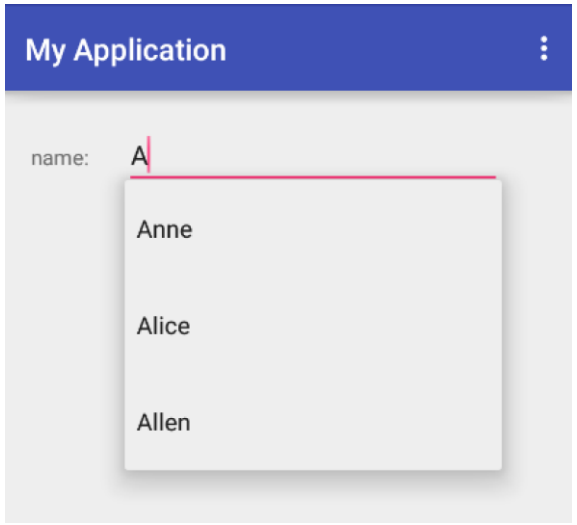
android:dropDownWidth //下拉菜单的宽度

android:popupBackground //下拉菜单的背景



自动完成文本控件 AutoCompleteTextView

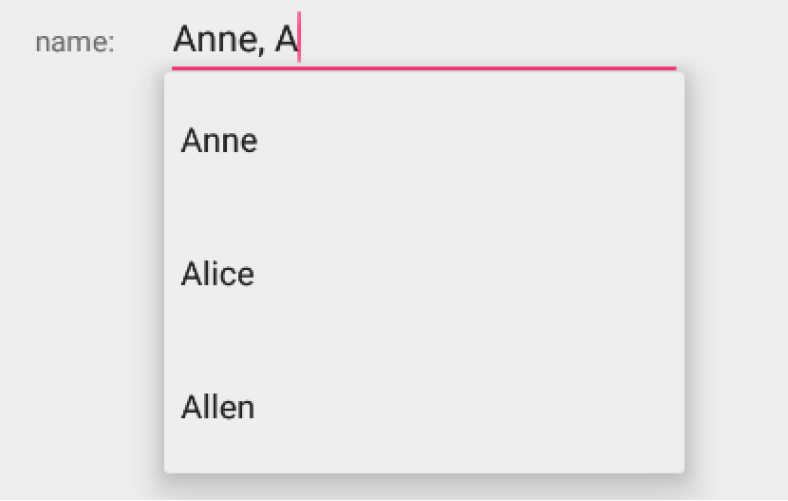
```
<AutoCompleteTextView
    android:layout_width="250dp"
    android:layout_height="wrap_content"
    android:id="@+id/auto_text"
    android:completionThreshold="1"/>
```




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

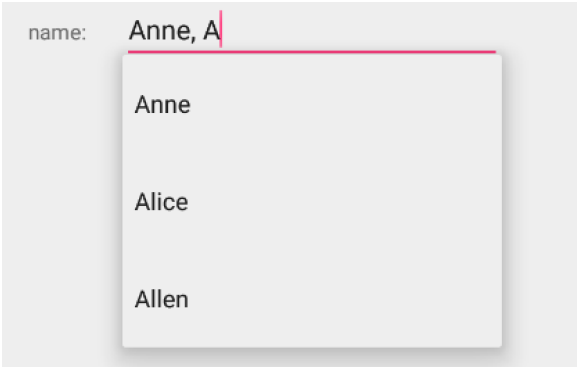
自动完成文本控件 MultiAutoCompleteTextView

- 与AutoCompleteTextView不同的是, MultiAutoCompleteTextView可以在输入框一直增加选择值。
- 当输入完一个字符串后,在该字符串后面输入一个逗号(,)，再输入 一个字符串,会显示自动提示列表。 使用时,需要为它的setTokenizer方法指定MultiAutoCompleteTextView.CommaTokenizer类对象实例, 表示采用逗号作为输入多个字符串的分隔符。



自动完成文本控件 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) 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；

进度条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

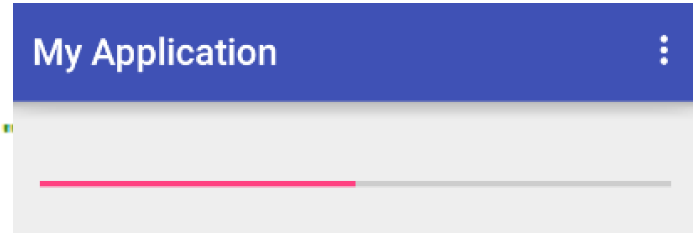
- 是某些操作的进度可视指示器,为用户呈现操作的进度。在不确定模式下,进度条显示循环动画,常用于任务长度是未知的情况。
- 进度条默认为圆圈形式,要显示水平进度条,可以在xml文件中设置进度条的 style属性。

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

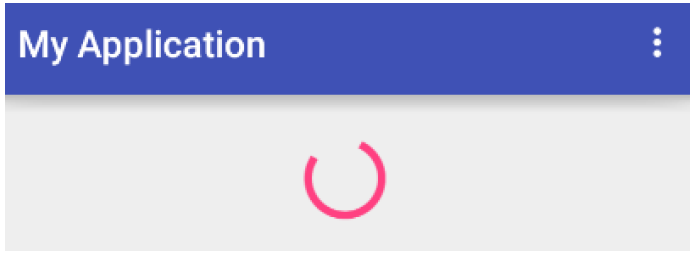
```
<ProgressBar
    android:layout_width="match_parent"
    android:layout_height="50dp"
    style="?android:attr/progressBarStyleHorizontal"
    android:id="@+id/download_bar"
    android:max="100"/>
```

```
//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="match_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

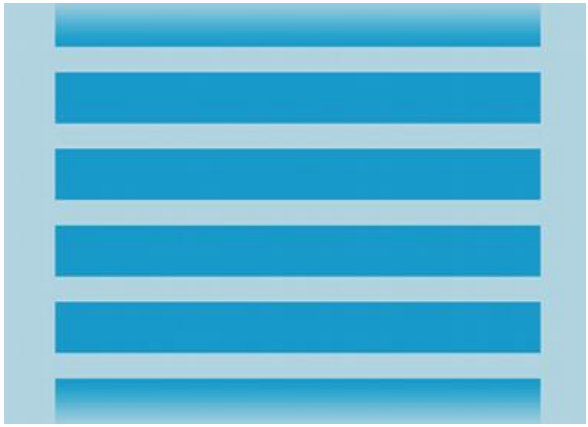
```
seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(SeekBar seekBar, int i, boolean b) {
        //进度发生变化
        textView.setText("当前进度为：" + Integer.toString(i));
    }

    @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
    }
});
```

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

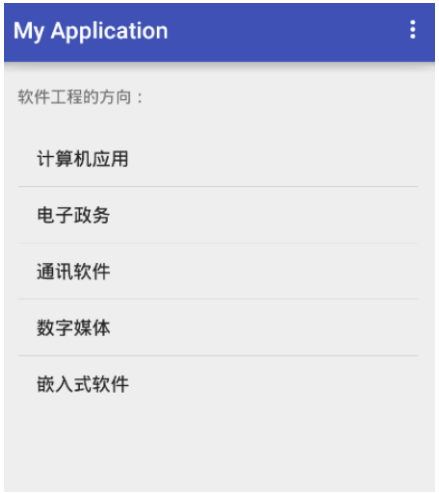
列表ListView

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的数据。其中第二个参数可以自定义一个layout。

```
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。

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ScrollView

- 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。
- 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。

```
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.small.myapplication.MainActivity">

    <android.support.constraint.ConstraintLayout...>
</ScrollView>
```

网页视图WebView

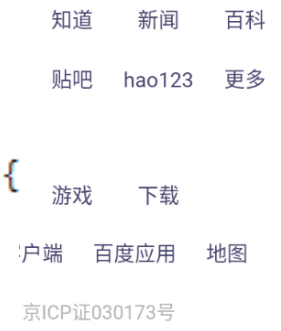
WebView是一个基于webkit引擎、展现web页面的控件。

WebView控件功能强大，除了具有一般View的属性和设置外，还可以对url请求、页面加载、渲染、页面交互进行强大的处理。

网页视图WebView

```
<WebView
    android:id="@+id/webView"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```


```
webView = (WebView) findViewById(R.id.webView);
webView.loadUrl("https://www.baidu.com/");
webView.setWebViewClient(new WebViewClient() {
    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String url) {
        view.loadUrl(url);
        return true;
    }
});
```



WebView加载百度网页

对话框Dialog

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。对话框不会填充屏幕，通常用于需要用户采取行动才能继续执行的模式事件。

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

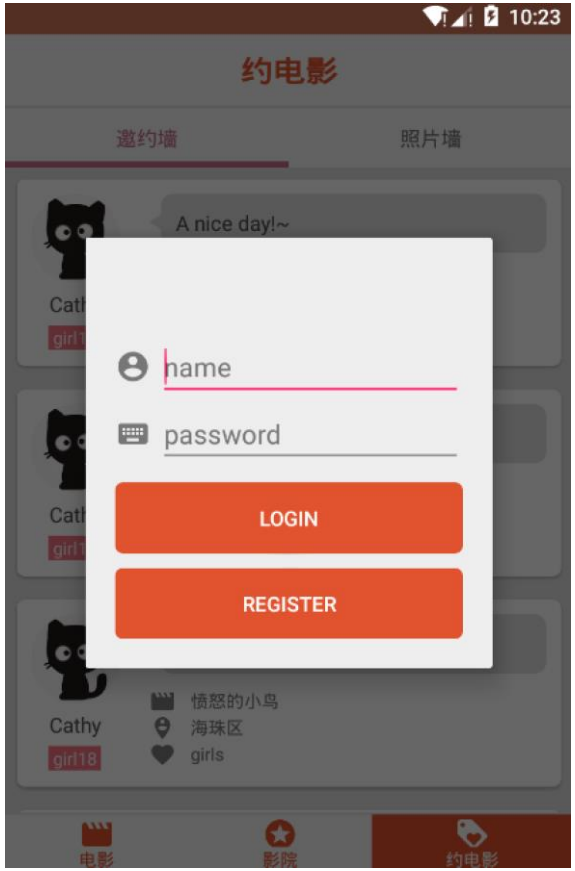
b) 通过将androidManifest.xml中的activity的属性设为android:theme=“@android:style/Theme.Dialog”,伪装为dialog

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。



对话框Dialog

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对象通常是通过其内部静态类AlertDialog.Builder来进行构造的。

对话框Dialog

```
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 dialogInterface, int i) {
                Toast.makeText(getApplicationContext(),
                        "你点击了确认", Toast.LENGTH_SHORT).show();
            }
        }).setNegativeButton("取消",
        new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                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 view) {
            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

//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 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 dialogInterface, int i) {
                Toast.makeText(getApplicationContext(),
                        "你点击了确认", Toast.LENGTH_SHORT).show();
            }
        }).setNegativeButton("取消",
        new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                Toast.makeText(getApplicationContext(),
                        "你点击了取消", Toast.LENGTH_SHORT).show();
            }
        }).create();
```


Questions?