Android移动开发

Android Mobile Application Development

第3讲 Android用户界面开发

吴以凡
计算机学院 一教505
yfwu@hdu.edu.cn

Android用户人机界面

用户人机界面

- 用户人机界面 Graphical User Interface, GUI
- Android中GUI可由SDK提供的UI控件组合而成
  - 视图（View）
  - 视图容器（View Group）
  - 布局（Layout）

视图（View）

- Android平台中用户界面的基础元素
  - 文本视图
  - 单选按钮
  - 复选框
- 通过视图View，可实现绘图、焦点变换、滚动条、屏幕区域的按键、用户交互等功能。
- 用户与视图交互是通过事件驱动机制来实现的。

视图（View）

- Android应用的绝大部分UI控件都放在android.widget包和android.view包中

TextView类

```
java.lang.Object
   ↳ android.view.View
         ↳ android.widget.TextView
```

Button类

```
java.lang.Object
   ↳ android.view.View
         ↳ android.widget.TextView
               ↳ android.widget.Button
```

视图（View）

- 所有视图组件View都提供了两种方式来控制组件的行为：
  - 在XML布局文件中通过XML属性进行控制
  - 在JAVA程序代码中通过调用方法进行控制
- View常用的XML属性及对应的Java相关方法

XML属性	方法	说明
android:alpha	setAlpha（float）	设置透明度
android:background	setBackgroundResource(int)	设置背景
android:id	setId(int)	设置组件标识
android:visibility	setVisibility(int)	设置组件是否可见
android:keepScreenOn	setkeepScreenOn(boolean)	设置组件是否会强制手机屏幕一直打开
Android:longClickable	setLongClickable(boolean)	设置是否响应长单击事件
android:scaleX	setScaleX(float)	设置水平方向的缩放比
android:scaleY	setScaleY(float)	设置垂直方向的缩放比

视图容器（View Group）

- 多个视图（View）可以存放在一个视图容器（ViewGroup）中，这种容器既能包含视图组件，也能包含一个已有的视图容器组件
- 一个界面文件必须有且仅有一个容器作为根节点
- 视图容器组件简化了界面的实现方式，Android能够以一个群组的方式管理多个视图组件
- ViewGroup类是android.view.View的子类，其继承关系如下所示：

```
java.lang.Object
  ↳        android.view.View
             ↳        android.view.ViewGroup
```

布局组件（Layout）

- 在UI设计中，除了要清楚控件的作用和接口之外，还需要熟悉控件的布局，布局规定了界面中元素之间的排列方式
- Android提供了许多种布局：
  - 线性布局（LinearLayout）
  - 相对布局（RelativeLayout）
  - 表格布局（TableLayout）
  - 网格布局（GridLayout）
  - 绝对布局（AbsoluteLayout）
  - 帧布局（FrameLayout）
  - 扁平化布局（ConstraintLayout）

页面布局

布局管理器

- 在Android开发当中，界面的设计是通过布局文件实现的，布局文件采用XML的格式，每个应用程序默认会创建一个activity_main.xml布局文件，它是应用启动的界面
- 创建和使用布局文件
  - res/layout文件夹，右键点击New > Layout resource file
  - 添加控件
    - Text: 通过在xml文件中添加
    - Design: 在图形界面上进行拖拉操作，然后再次通过代码进行调整

线性布局（LinearLayout）

- LinearLayout是一种线性排列的布局，在该布局中子元素之间成线性排列即顺序排列。由于布局是显示在二维空间里，其顺序排列是在水平或者垂直方向的顺序排列
- 常用设置参数
  - android:orientation
    - vertical
    - horizontal
  - android:layout_width, android:layout_height
    - fill_parent: 整个屏幕
    - match_content: 与父元素同
    - wrap_content: 随组件本身内容调整
    - 数值

宽高单位

- **px**:代表像素，即在屏幕中可以显示的最小元素单元。分辨率越高的手机，屏幕的像素点就越多。因此，如果使用px来设置控件的大小，在分辨率不同的手机上控件的显示出来的大小也不一样。
- **pt**:代表磅数，一般pt都会作为字体的单位来显示。pt和px的情况相似，在不同分辨率的手机上，用pt作为字体单位显示的大小也不一样。
- **dp**:代表密度无关像素，又称dip，使用dp的好处是在无论屏幕的分辨率如何总能显示相同的大小，一般使用dp作为控件与布局的宽高单位。
- **sp**:代表可伸缩像素，采用与dp相同的设计理念，设置字体大小时使用。

线性布局（LinearLayout）

```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="horizontal">
  <Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button1" />
  <Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button2" />
   <Button ..../>
</LinearLayout>
```

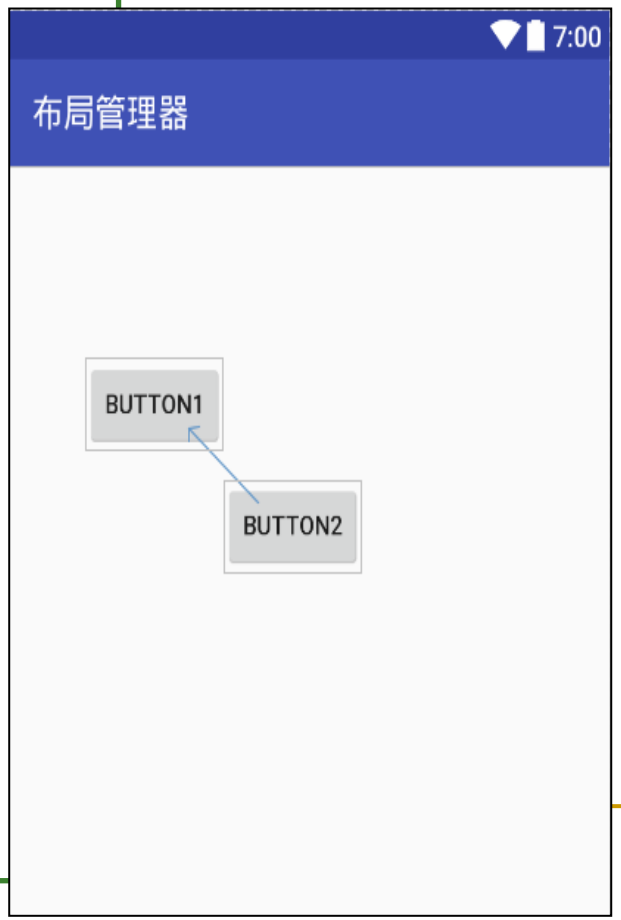
相对布局（RelativeLayout）

- RelativeLayout是一种根据相对位置排列元素的布局，这种方式允许子元素指定它们相对于其它元素或父元素的位置。
- 默认生成的布局

属性声明	功能描述	android:layout_above	在指定控件上边
android:layout_alignParentLeft	是否跟父布局左对齐	android:layout_below	在指定控件下边
android:layout_alignParentRight	是否跟父布局右对齐	android:layout_alignBaseline	与指定控件水平对齐
android:layout_alignParentTop	是否跟父布局顶部对齐	android:layout_alignLeft	与指定控件左对齐
android:layout_alignParentBottom	是否跟父布局底部对齐	android:layout_alignRight	与指定控件右对齐
android:layout_toRightOf	在指定控件右边	android:layout_alignTop	与指定控件顶部对齐
android:layout_toLeftOf	在指定控件左边	android:layout_alignBottom	与指定控件底部对齐

相对布局（RelativeLayout）

```
<RelativeLayout ...
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context="com.jxust.cn.relativelayout.MainActivity">
  <Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_marginTop="100dp"
    android:layout_marginLeft="50dp"
    android:text="Button1" />
  <Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_toRightOf="@id/button"
    android:layout_below="@id/button"
    android:layout_marginTop="15dp"
    android:text="Button2"/>
</RelativeLayout>
```



表格布局（TableLayout）

- 表格布局就是以表格的形式来排列控件，只要将控件放在单元格中，控件就可以整齐地排列
- 表格布局并不是真正意义上的表格，只是按照表格的方式组织元素的布局
- 表格布局不能设置其子元素的宽度属性（layout_width），其属性必须同父元素相同（FILL_PARENT）
- 行数由TableRow对象控制，每行可以放多个控件，列数由最宽的行决定，假设第一行有两个控件，第二行有3个控件，那么这个表格布局就有3列。在控件中使用layout_column属性指定具体的列数，该属性的值从0开始，代表第一列。

表格布局（TableLayout）

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:stretchColumns="1">
  <TableRow
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
      android:id="@+id/button1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_column="0"
      android:text="Button1" />
  </TableRow>
  <TableRow
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
      android:id="@+id/button2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_column="1"/>
  </TableRow>
</TableLayout>
```

表格布局：两行，每行一个按钮。使用android:stretchColumns=“1”属性拉伸第二列，android:layout_column=“0”属性表示显示在第一列中。

TableRow不需要设置宽度和高度，其宽度一定是自动填充容器，高度根据内容改变。

网格布局（ GridLayout ）

- Android4.0新增的布局管理器，需要在4.0之后的版本才能使用
- 类似TableLayout，把整个容器划分为rows×columns个网格，每个网格可以放置一个控件
- 提供了setRowCount(int)和setColumnCount(int)方法来控制该网格的行数量和列数量

网格布局（ GridLayout

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout xmlns:android="http://schemas.android.com/ap
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_gravity="center"
  android:columnCount="4"
  android:rowCount="4"
  android:orientation="horizontal">
  <Button android:text="/"
    android:layout_column="3"/>
  <Button android:text="1"/>
  <Button android:text="2"/>
  <Button android:text="3"/>
  <Button android:text="*"/>
  <Button android:text="4"/>
  <Button android:text="5"/>
  <Button android:text="6"/>
  <Button android:text="-"/>
  <Button android:text="7"/
  <Button android:text="8"/
  <Button android:text="9"/
  <Button android:text="+"
    android:layout_gravity="fill"
    android:layout_rowSpan="2"/>
  <Button android:text="0"/>
  <Button android:text="="
    android:layout_gravity="fill"
    android:layout_columnSpan="2" />
</GridLayout>
```

帧布局（ FrameLayout ）

- 帧布局是Android布局中最简单的一种，帧布局为每个加入其中的控件创建了一块空白区域。采用帧布局的方式设计界面时，只能在屏幕左上角显示一个控件，如果添加多个控件，这些会依次重叠在屏幕左上角显示，且会透明显示之前的文本。

帧布局（ FrameLayout ）

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent">
   <Button
      android:id="@+id/button10"
      android:layout_width="314dp"
      android:layout_height="315dp"
      android:text="Button1" />
   <Button
      android:id="@+id/button11"
      android:layout_width="216dp"
      android:layout_height="140dp"
      android:text="Button2" />
   <Button
      android:id="@+id/button12"
      android:layout_width="103dp"
      android:layout_height="wrap_content"
      android:text="Button3" />
</FrameLayout>
```

绝对布局（ AbsoluteLayout ）

- AbsoluteLayout（绝对布局）与相对布局相反，绝对布局不需要指定其参照物，绝对布局使用整个手机界面作为坐标系（屏幕坐标系），通过两个偏移量（水平偏移量和垂直偏移量）来唯一指定其位置。
- 屏幕坐标系：左上角为（0，0），往屏幕下方为y正半轴，右方为x正半轴
- 对于绝对布局标签，使用的两个参数是：
  - android:layout_x指定x坐标的位置
  - android:layout_y 指定y坐标的位置

绝对布局（ AbsoluteLayout ）

```
<AbsoluteLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent">
   <Button
      android:id="@+id/button3"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_x="71dp"
      android:layout_y="25dp"
      android:text="Button" />
</AbsoluteLayout>
```

扁平化布局（ ConstraintLayout ）

- ConstraintLayout是Android Studio2.2中主要的新增功能之一，为了解决布局嵌套层次太多
- 实现扁平化布局——几乎不需要任何嵌套
- ConstraintLayout使可视化方式来编写界面更容易
- 多种定位方法
  - 相对定位
  - 角度定位
  - 居中和偏移
  - 边距
  - 尺寸约束

约束布局（ ConstraintLayout ）

- 从 Android Studio 2.3 起，官方的模板默认使用 ConstraintLayout

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:id="@+id/lay_root"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <TextView
    android:id="@+id/text1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="test" />
</android.support.constraint.ConstraintLayout>
```

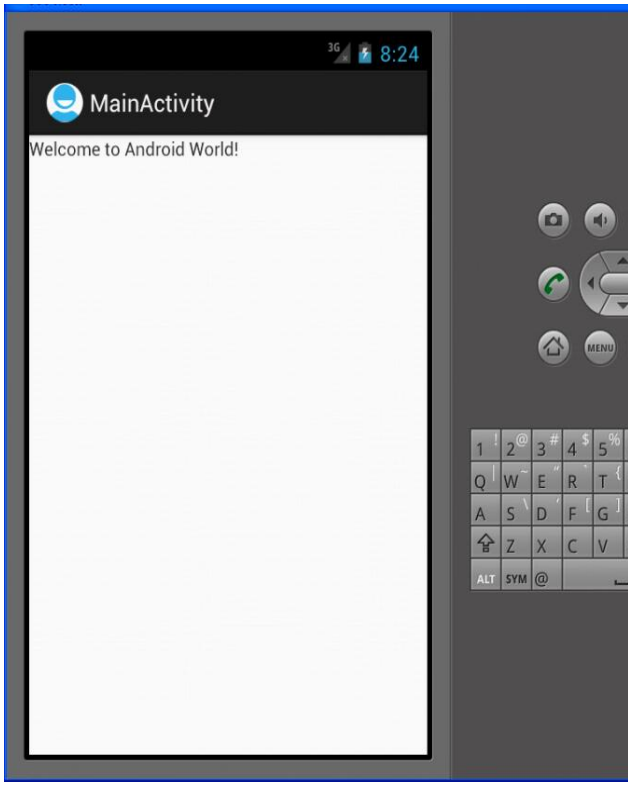
Widget控件

文本框（TextView)

- TextView是一个不可编辑的文本框,用来在屏幕中显示静态字符串
- 例如

```
<TextView ...
   android:id = "@+id/myTextViewID"
   android:textSize = "16sp"
   android:text = "Welcome to Android World!" />
```

```
myTextView=(TextView)
this.findViewById(R.id.myTextViewID);
myTextView.setText("Welcome to
                    Android World!");
```



编辑框（EditText）

- EditText同TextView功能基本类似，它们之间主要区别是EditText提供了可编辑的文本框
- 重要属性：
  - android:inputType，即输入类型，如手机号、密码、日期等
  - android:hint 输入提示
- 自定义样式

```
<shape
  xmlns:android=http://schemas.android.com/apk/res/android>
  <solid android:color="#FFFFFF" />
  <corners android:radius="3dip"/>
  <stroke
      android:width="2dip"
      android:color="#3F51B5" />
</shape> <!-- 通过background属性调用 -->
```

按钮（Button）

- Button类提供了控制按钮的功能，Button类是android.widget.TextView的子类
- 点击事件：onClick
- 样式
  - 默认采用当前android版本的系统默认样式
  - 自定义：android:textColor, android:background，drawable

图片（ImageView）

- ImageView继承自View组件
- 与TextView功能基本类似，主要区别是显示的资源不同。ImageView可显示图像资源，而TextView只能显示文本资源。
- ImageView可通过两种方式设置资源
  - 第一种是通过setImageBitmap()方法设置图片资源
  - 第二种是通过<ImageView> XML元素的android:src属性或setImageResource(int)方法指定ImageView的图片。
- 可以通过Bitmap和BitmapFactory实现ImageView图片的放大、缩小、左转和右转
- ImageView还派生了ImageButton、ZoomButton等组件

图片（ImageView）

```
<ImageView
   android:id="@+id/imageView"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_gravity="center"
   android:src="@mipmap/ic_launcher" />
```

图片按钮（ImageButton）

- ImageButton跟Button功能基本类似，主要区别是ImageButton可通过图像表示按钮的外观
- 可通过<ImageButton> XML元素的android:src属性或setImageResource(int)方法指定ImageButton的图片
- 点击事件：onTouch、onClick

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

单选按钮（RadioButton）

- 继承自Button类
- 该控件表明一个特定的状态是勾选 (on，值为1) 还是不勾选 (off，值为0) ，android:checked属性可指定RadioButton初始时是否被选中
- 为实现一组RadioButton只能选中其中一个，RadioButton通常要和RadioGroup一起使用，用于定义一组单选按钮。

单选按钮（RadioButton）

```
<RadioGroup
  android:layout_width=“match_parent”
  android:layout_height="wrap_content">
    <RadioButton
      android:id="@+id/radioButton2"
      android:layout_width="wrap_content"
      android:layout_height=“wrap_content”
      android:checked="true"
      android:text="男" />
    <RadioButton
      android:id="@+id/radioButton"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="女" />
</RadioGroup>
```

多选框（CheckBox）

- 与单选按钮类似
  - 继承自Button类
  - 有两种状态：选中（on,1）或者不选中状态（off,0）
  - android:checked属性可指定初始时是否被选中

```
<CheckBox
  android:id=“@+id/checkBox”
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:text="读书" />
<CheckBox
  android:id="@+id/checkBox2"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:checked=“true”
  android:text="看电影" />
  ……
```

进度条（ProgressBar）

- 用于提示应用正在处理中或处理的进度

```
android.widget.View
        ↳ android.widget.ProgressBar
```

- 还派生了两个常用的组件：SeekBar和RatingBar

进度条（ProgressBar）

XML属性	说明
android:max	设置该进度条的最大值
android:progress	设置该进度条的已完成进度值
android:progressDrawable	设置该进度条的轨道对应的Drawable对象
android:indeterminate	设置进度条是否精确显示进度
android: indeterminateDrawable	设置不显示进度条的Drawable对象
android: indeterminateDuration	设置不精确显示进度的持续事件

Android支持多种风格的进度条，通过style属性可以为ProgressBar指定风格：

属性值	说明
Widget.ProgressBar.Horizontal	水平进度条
Widget.ProgressBar.Inverse	普通大小的环形进度条
Widget.ProgressBar.Large	大环形进度条
Widget.ProgressBar.Large.Inverse	大环形进度条
Widget.ProgressBar.Small	小环形进度条
Widget.ProgressBar.Small.Inverse	小环形进度条

进度条（ProgressBar）

```
<ProgressBar
    android:id="@+id/progressBar5"
    style="?android:attr/progressBarStyle"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```


```
<ProgressBar
    android:id=“@+id/progressBar4”
    style="?android:attr/progressBarStyleHorizontal"
    android:layout_width="145dp"
    android:layout_height="25dp"
    android:layout_gravity="center"
    android:layout_marginTop=“30dp”
    android:background="@android:color/holo_green_light" />
```

拖动条（SeekBar）

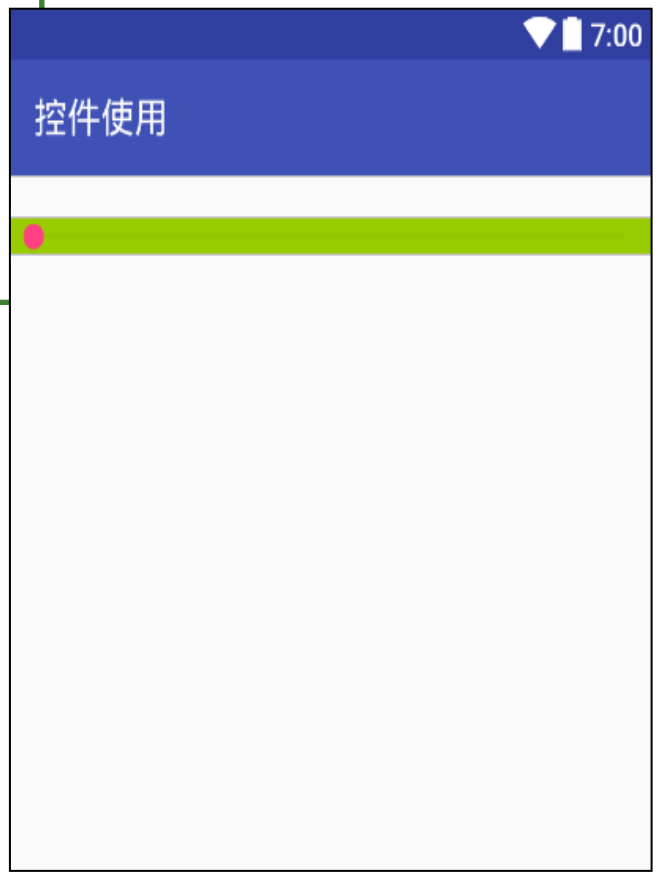
- SeekBar可实现拖动进度条的功能，通常用于对系统的某种数值进行调节，如拖动视频、拖动音量等。

```
android.widget.View
        ↳ android.widget.ProgressBar
            ↳ android.widget.SeekBar
```

- 拖动条类只定义了setOnSeekBarChangeListener方法和thumb属性。
  - thumb 用于指定拖动条滑块的外观（一个drawable对象）
  - setOnSeekBarChangeListener方法用于注册拖动条的监听器OnSeekBarChangeListener。该监听器能够监听三种事件：
    - StartTrackingTouch（拖动开始）
    - ProgressChanged（拖动中）
    - StopTrackingTouch（拖动结束）

拖动条（SeekBar）

```
<SeekBar
    android:id="@+id/seekBar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:background="@android:color/holo_green_light"
    android:layout_weight="1" />
```



评分组件（RatingBar）

- RatingBar 是基于 SeekBar 和 ProgressBar 的扩展，用星型来显示等级评定。
- Android定义了3种类型的评分条：
  - ratingBarDefault（默认评分条）
  - ratingBarStyleIndicator（指示功能评分条）：不能交互
  - ratingBarStyleSmall（小评分条）：不能交互
- RatingBar的监听器为OnRatingBarChangeListener，当RatingBar控件状态发生变化时，该监听器的onRatingChanged方法会被触发。

自动完成文本（AutoCompleteTextView）

- 想象这样一个场景：上网搜索时，只要输入几个文字，搜索框就会提示相近的关键字。Android提供了这种功能的控件-AutoCompleteTextView。
- AutoCompleteTextView是一个文本框组件，它提供了自动完成文本功能。用户只需输入一部分内容，剩下部分系统就会给予提示。
- 使用AutoCompleteTextView时，必须提供一个MultiAutoCompleteTextView.Tokenizer对象以用来区分不同的子串。

```
android.widget.TextView
        ↳ android.widget.EditText
            ↳ AutoCompleteTextView
```

日期选择器（DatePicker）

- DatePicker是一个选择日期的布局视图，它提供了日期选择器的功能。

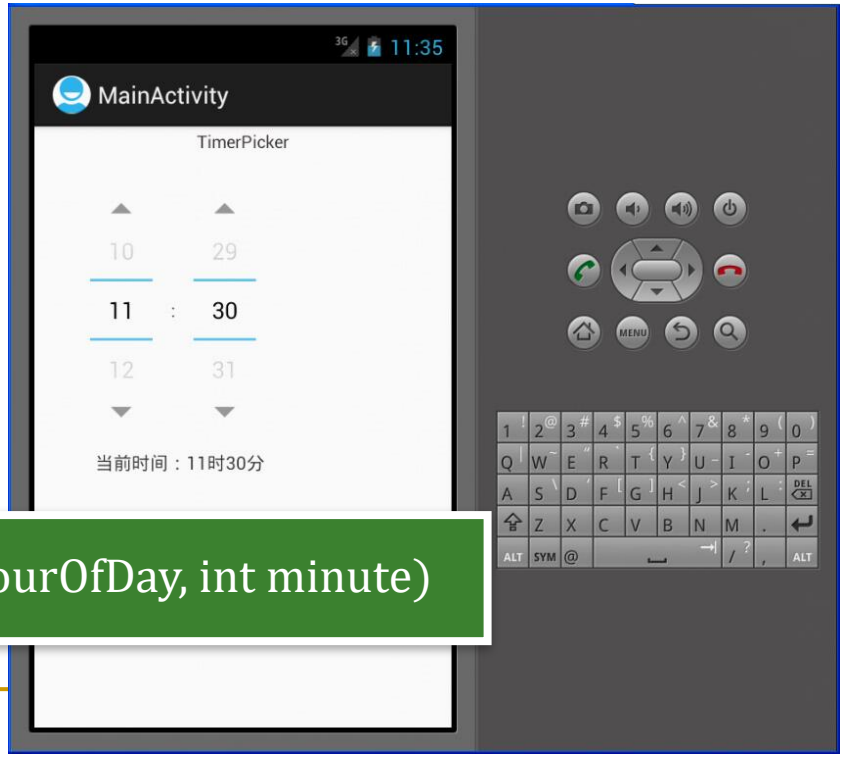
```
java.lang.Object
    ↳ android.view.View
        ↳ android.view.ViewGroup
            ↳ android.widget.FrameLayout
                ↳ android.widget.DatePicker
```

- DatePicker，常用的监听器就是OnDateChangedListener。当日期发生改变时，OnDateChangedListener的onDateChanged方法会被触发，系统会将发生变化后的日期值以传递参数的形式传给onDateChanged方法。

```
onDateChanged (DatePicker view, int year, int monthOfYear, int dayOfMonth)
```

时间选择器（TimePicker）

- TimePicker支持 24小时及上午/下午模式。小时，分钟及上午/下午都可以用垂直滚动条来调整。
- TimePicker常用的监听器是OnTimeChangedListener，当时间发生改变时，系统会触发 onTimeChanged方法。



onTimeChanged (TimePicker view, int hourOfDay, int minute)

AdapterView及其子类

AdapterView

- AdapterView是一组重要的组件，AdapterView本身是一个抽象基类，它派生的子类在用法上十分相似，只是显示界面上有一定的区别
- AdapterView继承了ViewGroup，它的本质是容器
- AdapterView可以包括多个“列表项”，并以合适的方式显示出来
- AdapterView显示的多个“列表项”由Adapter提供。调用AdapterView的setAdapter(Adapter)方法设置Adapter即可

ListView和ListActivity

- ListView是AdapterView的子类之一，它以垂直列表的形式显示所有的列表项。生成列表视图有如下两种方式：
  - 直接使用ListView进行创建
  - 创建一个继承ListActivity的Activity（相当于该Activity显示的组件为ListView）
- 在程序中获得了ListView后，需要为ListView设置它要显示的列表项了。
  - 采用其所具有的AdapterView的特征：通过setAdapter(Adapter)方法为之提供Adapter，并由Adapter提供列表项

ListView常用XML属性

XML属性	说明
android:divider	设置分割条样式（颜色或者Drawable对象）
android:dividerHeight	设置分割条高度
android:entries	指定一个数组资源，用来填充ListView项
android:footerDividersEnabled	设置为false，则不在footer view之前绘制分割条
android:headerDividersEnabled	设置为false，则不在header view之后绘制分割条
android:scrollbars	设置是否显示滚动条
android:fadingEdge	设置是否去除ListView滑到顶部和底部时边缘的黑色阴影
android:listSelector	设置是否去除点击颜色
android:cacheColorHint	设置ListView去除滑动颜色

ListView

```
<!--直接使用数组资源给list view添加列表项-->
<!--设置分割条的颜色-->
<!--设置分割条的高度-->
  <ListView
    android:id="@+id/listview1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:divider="#C4C4C4"
    android:entries="@array/teacher_name"
    android:dividerHeight="1dp">
  </ListView>
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<!--添加数组元素-->
  <string-array name="teacher_name">
    <item>张三</item>
    <item>李四</item>
    <item>王五</item>
    <item>赵六</item>
  </string-array>
</resources>
```

ListView点击事件

- 点击某一项的事件响应可利用
  - AdapterView.OnItemClickListener

```
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        ...
        // position为被点击的项在adapter中的位置
    }
});
```

Adapter接口

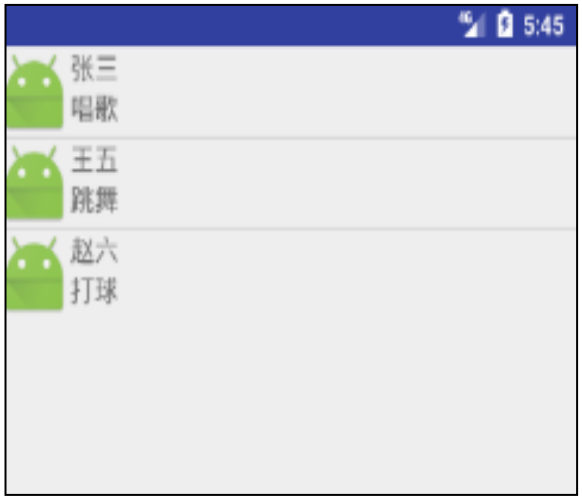
- Adapter本身只是一个接口，它派生了两个子接口
  - ListAdapter：为AbsListView提供列表项
  - SpinnerAdapter：为AbsSpinner提供列表项
- Adapter常用的实现类包括：
  - ArrayAdapter：支持泛型操作，最为简单，只能展示一行字
  - SimpleAdapter：有最好的扩充性，可以自定义出各种效果
  - BaseAdapter：是一个抽象类，继承它需要实现较多的方法，所以也就具有较高的灵活性

ArrayAdapter

```
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.listview_layout);

      ListView  listView = (ListView)findViewById(R.id.listview1);

      //定义一个数组，用来填充listView
      String[] arr={"章节1","章节2","章节3"};
      ArrayAdapter<String> adapter=new ArrayAdapter<String>
(this,android.R.layout.simple_expandable_list_item_1,arr);

      //为listView设置adapter
      listView.setAdapter(adapter);
   }
}
```

SimpleAdapter

```
<LinearLayout ...>
  <!--定义一个ImageView组件，用来显示头像-->
  <ImageView
    android:id="@+id/icon"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
      <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <!--定义一个TextView组件，用来显示名字-->
        <TextView
          android:id="@+id/name"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:textSize="16sp"/>
        <!--定义一个TextView组件，用来显示人物的描述-->
        <TextView
          android:id="@+id/desc"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:textSize="16sp"/>
      </LinearLayout>
</LinearLayout>
```

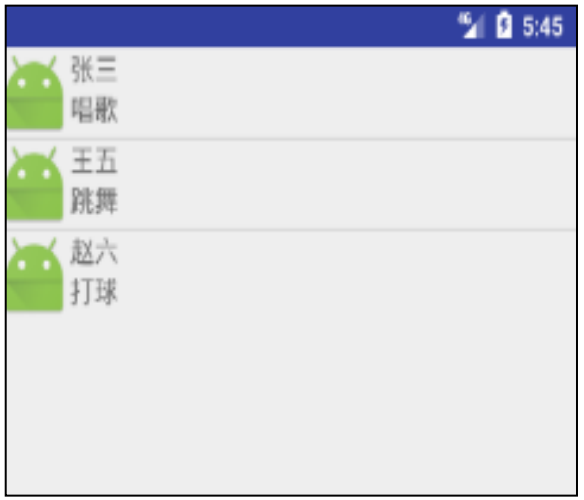


SimpleAdapter

```
public class MainActivity extends Activity {
   //定义名字数组
   private String[] name={"张三","王五","赵六"};
   //定义描述数组
   private String[] desc={"唱歌","跳舞","打球"};
   //定义头像数组
   Private int[] icon=new int[] {R.mipmap.ic_launcher,R.mipmap.ic_launcher,R.mipmap.ic_launcher};
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.listview_layout);
      ListView listView = (ListView)findViewById(R.id.listview1);
      //创建一个list集合，list集合的元素是MAP
      List<Map<String,Object>> list = new ArrayList<Map<String,Object>>();
      for(int i=0;i<name.length;i++){
         Map<String, Object> listitem = new HashMap<String, Object>();
         listitem.put("icon",icon[i]);
         listitem.put("name",name[i]);
         listitem.put("desc",desc[i]);
         list.add(listitem);
      }
```

SimpleAdapter

```
        //创建一个SimpleAdapter
        SimpleAdapter adapter = new SimpleAdapter(
          this,
          list,
          R.layout.list_item_layout,
          new String[]{"name","icon","desc"},
          new int[]{R.id.name,R.id.icon,R.id.desc}
        );
        listView.setAdapter(adapter);
    }
}
```



使用SimpleAdapter最重要的是它的5个参数：

- 第二个参数是List<? Extends Map<String,?>>类型的集合对象，该集合中每个Map<String,?>对象生成一行
- 第三个参数是指定一个界面布局的ID，这里引用了一个自定义的布局list_item_layout.xml文件
- 第四个参数是String[]类型的参数，该参数决定提取哪些内容显示在listview的每一行
- 最后一个是int[]类型的参数，决定显示哪些组件

BaseAdapter

- 在使用SimpleAdapter时，用户可以在布局当中定义按钮，但是当用户点击时，由于点击操作被ListView的Item所覆盖，导致按钮无法获取到焦点，此时应使用最灵活的适配器BaseAdapter
- BaseAdapter是最基础的Adapter，具有全能性，不会像ArrayAdapter等的封装好的类有那么多局限性，但使用难度更高
- 需要继承BaseAdapter类，并重写getCount、getItem、getItemId、getView四个方法

BaseAdapter


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="horizontal" >
      <LinearLayout
        android:layout_width="200dip"
        android:layout_height="match_parent"
        android:orientation="horizontal">
        <ImageView
          android:id="@+id/imageview"
          android:layout_width="50dip"
          android:layout_height="50dip" />
        <TextView
          android:id="@+id/textview"
          android:layout_width="wrap_content"
          android:layout_height="match_parent"
          android:paddingTop="8dip"
          android:textSize="20sp" />
      </LinearLayout>
      <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

（1）自定义布局文件list_item_layout.xml作为每一行的布局样式

BaseAdapter

```
public class MyAdapter extends BaseAdapter {
    private List<Map<String, Object>> datas;
    private Context mContext;
    public MyAdapter(List<Map<String, Object>> datas, Context mContext) {
      this.datas = datas;
      this.mContext = mContext;
   }
   public int getCount() {
      // 返回数据的总数
      return datas.size();
   }
   public Object getItem(int position) {
      // 返回在list中指定位置的数据的内容
      return datas.get(position);
   }
   public long getItemId(int position) {
      // 返回数据在list中所在的位置
      return position;
   }
```

（2）自定义一个MyAdapter类继承自BaseAdapter，重写4个方法方法

BaseAdapter

```
public View getView(int position, View convertView, ViewGroup parent) {
    final ViewHolder holder;
    if (convertView == null) {
        // 使用自定义的布局文件作为Layout
        convertView = LayoutInflater.from(mContext).inflate( R.layout.list_item_layout,
null);
        // 减少findView的次数
        holder = new ViewHolder();
        // 初始化布局中的元素
        holder.mImageView = (ImageView)
        convertView.findViewById(R.id.imageview);
        holder.mTextView = (TextView) convertView.findViewById(R.id.textview);
        holder.mButton = (Button) convertView.findViewById(R.id.button);
        holder.mButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View view) {
                Toast.makeText(mContext,"你点了我！哈哈", Toast.LENGTH_SHORT).show();
            }
        });
        convertView.setTag(holder);
```

BaseAdapter

```
        } else {
            holder = (ViewHolder) convertView.getTag();
          }
              // 从传入的数据中提取数据并绑定到指定的view中
        holder.mImageView.setImageResource((Integer) datas.get(position).get("img"));
        holder.mTextView.setText(datas.get(position).get("title").toString());
         holder.mButton.setText(datas.get(position).get("button").toString());
            return convertView;
    }

    static class ViewHolder {
        ImageView mImageView;
        TextView mTextView;
        Button mButton;
    }
}
```

BaseAdapter

（3）MainActivity中添加数据以及为ListView添加上文自定义的Adapter

```
public class MainActivity extends Activity {
   private ListView mListView;
   private MyAdapter myAdapter;
   private List<Map<String, Object>> list =
            new ArrayList<Map<String,
Object>>();
    protected void onCreate(Bundle
savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      initData();
      mListView=(ListView)
findViewById(R.id.listview);
      myAdapter = new MyAdapter(list, this);
      mListView.setAdapter(myAdapter);
   }
```

```
 private void initData() {
      Map<String, Object> map = new
HashMap<String, Object>();
      map.put("img", R.drawable.android);
      map.put("title", "Android");
      map.put("button", "学习");
      list.add(map);
      map = new HashMap<String, Object>();
      map.put("img", R.drawable.java1);
      map.put("title", "JAVA");
      map.put("button", "学习");
      list.add(map);
      map = new HashMap<String, Object>();
      map.put("img", R.drawable.html5);
      map.put("title", "HTML5");
      map.put("button", "学习");
      list.add(map);
    ...
```

对话框

AlertDialog对话框

- 使用AlertDialog对话框步骤：
  - 创建AlertDialog.Builder对象
  - 调用AlertDialog.Builder.setTitle()或setCustomTitle()方法设置标题
  - 调用AlertDialog.Builder.setIcon设置对话框图标
  - 调用AlertDialog.Builder.setPositiveButton等添加按钮
  - 调用AlertDialog.Builder的create方法创建AlertDialog对象
  - 调用AlertDialog的show方法把对话框显示出来

AlertDialog对话框

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent">
   <Button
      android:id="@+id/dialog"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_weight="1"
      android:layout_marginTop="200dp"
      android:text="显示对话框" />
</LinearLayout>
```

AlertDialog对话框


```
public class MainActivity extends Activity {
  @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.dialog_layout);
       Button button=(Button)findViewById(R.id.dialog);
       button.setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View view) {
            new AlertDialog.Builder(MainActivity.this).setTitle("系统提示")
              .setMessage("请确认所有数据都保存后再推出系统！")
              .setPositiveButton("确定", new DialogInterface.OnClickListener() {
                 public void onClick(DialogInterface dialog, int which) {
                    finish(); }
              }).setNegativeButton("返回", new DialogInterface.OnClickListener() {
                  @Override
                  public void onClick(DialogInterface dialog, int which) { }
              }).show();
          }
      });
}
```

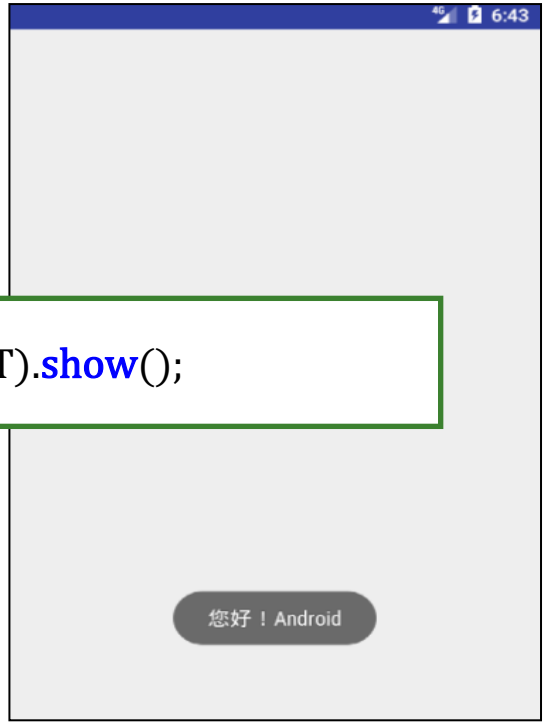
Toast提示消息

- Toast会显示一个消息在屏幕上告诉用户一些信息，并且在短暂的时间后会自动消息
- 主要使用Toast的两个方法：
  - makeText()方法
  - show()方法
- 可在Android的实际开发过程当中，利用Toast作为调试工具，显示传递的变量值等，观察是否跟预想情况一样

Toast提示消息

- Toast显示时间长短
  - Toast.LENGTH_SHORT
  - Toast.LENGTH_LONG

```
Toast.makeText(this,“您好！Android” , Toast.LENGTH_SHORT).show();
```



菜单

上下文菜单（Context Menu）

- 使用Android上下文菜单时，需要长时间按住才能显示其子菜单。如下图所示。
- 创建一个上下文菜单，需要重写onCreateContextMenu()方法，然后可通过registerForContextMenu()将视图添加到上下文菜单中。

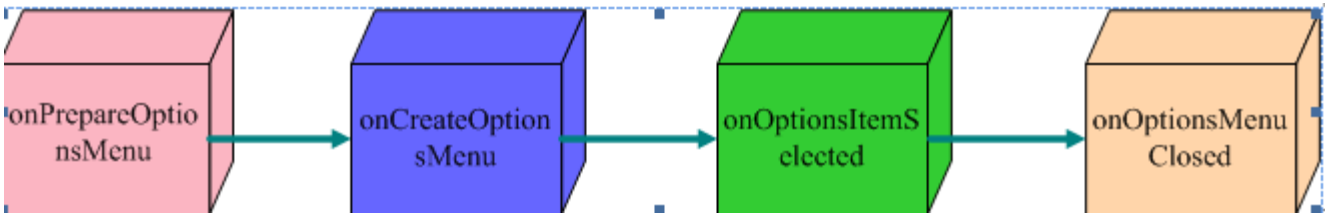
onCreateContextMenu(ContextMenu menu, View view, ContextMenuInfo info)

选项菜单（Options Menu）

- 选项菜单提供了一种特殊的菜单显示方式，这种菜单从视图底部弹出选项供用户选择
- 这种菜单不同于上下文菜单，选项菜单没有对应的视图即用户无法通过点击屏幕上的视图来加载选项菜单

选项菜单（Options Menu）

- 选项菜单生命周期
  - 实现一个选项菜单的过程比较简单，需要重写OptionsMenu的
    - onPrepareOptionsMenu
    - onCreateOptionsMenu
    - onOptionsItemSelected
    - onOptionsMenuClosed
  - 其中这些方法在一个选项菜单的生命周期中的执行顺序如下所示：



基于XML的菜单结构

- 实现菜单结构的方式
  - Java代码
  - XML
    - res\menu

练习

- 用户注册


```
Spinner spinner=(Spinner)findViewById(R.id.spinner);
String[] city=new String[]{"北京","上海","武汉","南京","南昌
"};
ArrayAdapter<String> adapter=new
ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1,city);
spinner.setAdapter(adapter);
```