第3章 Android用户界面程序设计

3.1 用户界面基础


视频讲解

用户界面(User Interface)是系统和用户间进行信息交换的媒介。Android实行界面设计者和程序开发者独立并行工作的方式,实现了界面设计和程序逻辑完全分离,不仅有利于后期界面修改中避免修改程序的逻辑代码,也有利于针对不同型号手机的屏幕分辨率调整界面尺寸时不影响程序的运行。

为了使界面设计和程序逻辑分离,Android程序将用户界面和资源从逻辑代码中分离出来,使用XML文件描述用户界面,资源文件独立保存在资源文件夹中。Android用户界面框架(Android UI Framework)采用MVC(Model-View-Controller)模型,为用户界面提供处理用户输入的控制器(controller)、显示图像的视图(view)和模型(model)。其中,模型是应用程序的核心,保存数据和代码。控制器、视图和模型的关系如图3.1所示。

MVC中的视图呈现用户界面,使用户在界面上进行输入,控制器能够接收并响应用户的动作,如按键和触摸屏幕等,并将这些动作作为一系列独立事件加入队列中,按照"先进先出"的规则将每个事件分配给对应的事件处理函数进行处理,根据处理结果更新模型。视图根据更新后的模型重新绘制界面并向用户展示,形成一个界面、数据更新的循环。

Android系统的界面元素以一种树形结构组织在一起,称为视图树,如图3.2所示。视图树由View和ViewGroup构成。View是一个重要的基类,所有界面上的可见元素都是View的子类,ViewGroup是能够承载多个View的显示单元,用于承载界面布局和具有原子特性的重构模块。

视图树绘制依据从上至下的原则绘制每个界面元素,且每个元素负责完成自身的绘制,如果元素包含子元素,则该元素通知其下所有子元素进行绘制。

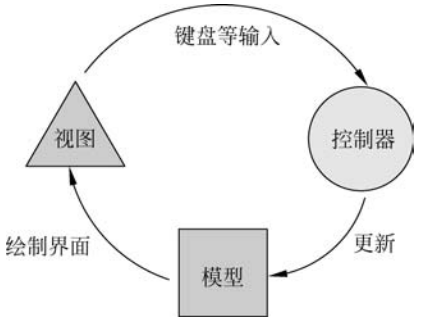

图3.1 MVC模型

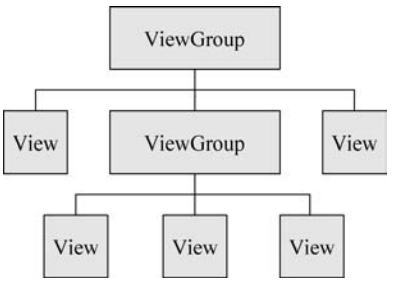

图3.2 视图树

Android 用户界面是单线程用户界面,事件的获取和界面的屏幕绘制使用同一个线程,这样的好处是用户不需要在控制器和视图间进行同步,事件的处理完全按照队列顺序进行;但单线程用户界面的缺点是如果事件函数过于复杂,可能导致用户界面失去响应,因此界面的事件响应函数应尽可能使用简短代码,或者将复杂工作交给后台线程处理。

3.2 界面布局

Android 系统定义了 6 种基本摆放控件的规则,它们都间接或者直接继承 ViewGroup 类,下面介绍这几种布局规则。

视频讲解

3.2.1 框架布局

框架布局(FrameLayout)也叫帧布局,该布局上的控件放置在左上角位置,按放置的前后顺序逐一层叠摆放,后面的控件会遮盖之前的控件。

【例 3-1】 演示框架布局编程方法。

(1) 创建名为 LayoutDemo 的新项目,包名为 edu.cqut.layoutdemo。切换到 Android 视图,右击 res/layout 文件夹,选择 New→XML→Layout XML File,在弹出的对话框的 Layout File Name 栏填入 layout_framelayout,在下方的 Root Tag 栏填入 FrameLayout,创建一个框架布局文件。

(2) 在新创建的布局文件中放置一个 ImageView 和一个 TextView 控件,代码如下。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< FrameLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent" >
    < ImageView
        android:id = "@ + id/mImageView"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:src = "@mipmap/ic_launcher"
    />
    < TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "框架布局"
        android:textSize = "18sp"
    />
</FrameLayout >
```

(3) 在 java/edu.cqut.layoutdemo 文件夹的 MainActivity.java 文件中修改与主 Activity 绑定的布局文件,修改后的代码如下。

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

程序运行结果如图 3.3 所示,界面布局文件中后添加的文本框控件遮挡了之前的图像控件。

3.2.2 线性布局

图 3.3 框架布局效果图


视频讲解

线性布局(LinearLayout)是将控件按照水平(horizontal)或垂直(vertical)两种方式排列,在布局文件中由 android:orientation 属性来控制排列方向。水平方向设置为 android:orientation="horizontal",垂直方向设置为 android:orientation="vertical"。

【例 3-2】 演示线性布局编程方法。

(1) 打开 LayoutDemo 项目,右击 res/layout 文件夹,选择 New→XML→Layout XML File,在弹出的对话框的 Layout File Name 栏填入 layout_linearlayout,在下方的 Root Tag 栏填入 LinearLayout,创建一个线性布局文件。

(2) 将新创建的布局文件的 android:orientation 属性设置为 vertical,然后放置三个 TextView 控件,分别显示第一行、第二行和第三行,代码如下。

```
<?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 = "vertical" >
    < TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "第一行">
    </TextView >
    < TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "第二行">
    </TextView >
    < TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "第三行">
    </TextView >
</LinearLayout >
```

(3) 在 MainActivity.java 代码中修改与主 Activity 绑定的布局文件,修改后的代码如下。

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

图 3.4 线性布局效果图

程序运行结果如图 3.4 所示,控件按垂直方向逐个排列。

3.2.3 相对布局

相对布局(RelativeLayout)是采用相对于

其他控件位置的布局方式,该布局内的控件和其他控件存在相对关系,通常通过指定 id 关联其他控件,以右对齐、上对齐、下对齐或居中对齐等方式来排列控件。

相对布局属性较多,表 3.1 介绍了几种常用属性。

视频讲解

表 3.1 相对布局常用属性

属 性	描 述
android:layout_alignParentTop="true|false"	是否与父控件的顶部平齐
android:layout_alignParentBottom="true|false"	是否与父控件的底部平齐
android:layout_alignParentLeft="true|false"	是否与父控件的左边平齐
android:layout_alignParentRight="true|false"	是否与父控件的右边平齐
android:layout_centerInParent="true|false"	是否在父控件的中间位置
android:layout_centerInHorizontal="true|false"	是否水平方向在父控件的中间
android:layout_centerInVertical="true|false"	是否垂直方向在父控件的中间
android:layout_alignTop="@id/***"	与相应 id 为 *** 控件的顶部平齐
android:layout_alignBottom="@id/***"	与相应 id 为 *** 控件的底部平齐
android:layout_alignLeft="@id/***"	与相应 id 为 *** 控件的左边平齐
android:layout_alignRight="@id/***"	与相应 id 为 *** 控件的右边平齐
android:layout_above="@id/***"	在 id 为 *** 控件的上面,该控件的底部与 *** 顶部平齐
android:layout_blow="@id/***"	在 id 为 *** 控件的下面,该控件的顶部与 *** 底部平齐
android:layout_toRightOf="@id/***"	在 id 为 *** 控件的右边,该控件的左边与 *** 右边平齐
android:layout_toLeftOf="@id/***"	在 id 为 *** 控件的左边,该控件的右边与 *** 左边平齐

【例 3-3】 演示相对布局编程方法。

(1) 打开 LayoutDemo 项目,右击 res/layout 文件夹,选择 New→XML→Layout XML File,在弹出的对话框的 Layout File Name 栏填入 layout_relativelayout,在下方的 Root Tag 栏填入 RelativeLayout,创建一个相对布局文件。

(2) 在该布局中放入三个 TextView 控件,并设置它们之间的相对位置关系,代码如下。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent" >
    < TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_centerHorizontal = "true"
        android:id = "@ + id/textview1"
        android:text = "TextView1(水平方向位于中间)"/>
    < TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
```

```
            android:id = "@ + id/textview2"
            android:layout_below = "@id/textview1"
            android:text = "TextView2(在 TextView1 下方)"/>
        < TextView
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:id = "@ + id/textview3"
            android:layout_below = "@id/textview2"
            android:layout_alignParentRight = "true"
            android:text = "TextView3(在 TextView2 下方且右对齐)"/>
    </RelativeLayout >
```

(3) 在 MainActivity.java 代码中修改与主 Activity 绑定的布局文件,修改后的代码如下。

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

程序运行结果如图 3.5 所示。由运行结果可以很明显地看出相对布局的特点,TextView1 位于水平方向居中,TextView2 位于 TextView1 下方,TextView3 位于 TextView2 下方且右对齐。

图 3.5 相对布局效果图

3.2.4 约束布局

视频讲解

约束布局(ConstrainLayout)是 Android Studio 2.2 中新增的功能之一。与传统的 XML 代码的界面编写方式相反,约束布局使用可视化的方式来编写界面,其结合 Android Studio 的布局编辑器通过拖曳控件完成布局,有利于解决布局嵌套过多的问题,是现在用得比较多的一种布局方式。

图 3.6 约束布局效果图

【例 3-4】 使用约束布局实现如图 3.6 所示的用户登录界面。

(1) 打开 LayoutDemo 项目,打开 res/layout/activity_main.xml 文件。该布局文件是项目创建的默认的布局文件,采用的是约束布局。

(2) 我们切换到 activity_main.xml 文件的 Design 视图,依次将左上 Palette 面板中的 TextView、imageView、Plain Text 和 Button 控件拖到左下 ComPonent Tree 面板中,每拖一个控件就通过中间的布局界面和右边的 Attributes 面板对该控件进行调整和属性设置,最后形成如图 3.7 所示的布局树和布局界面。

该布局的代码如下。

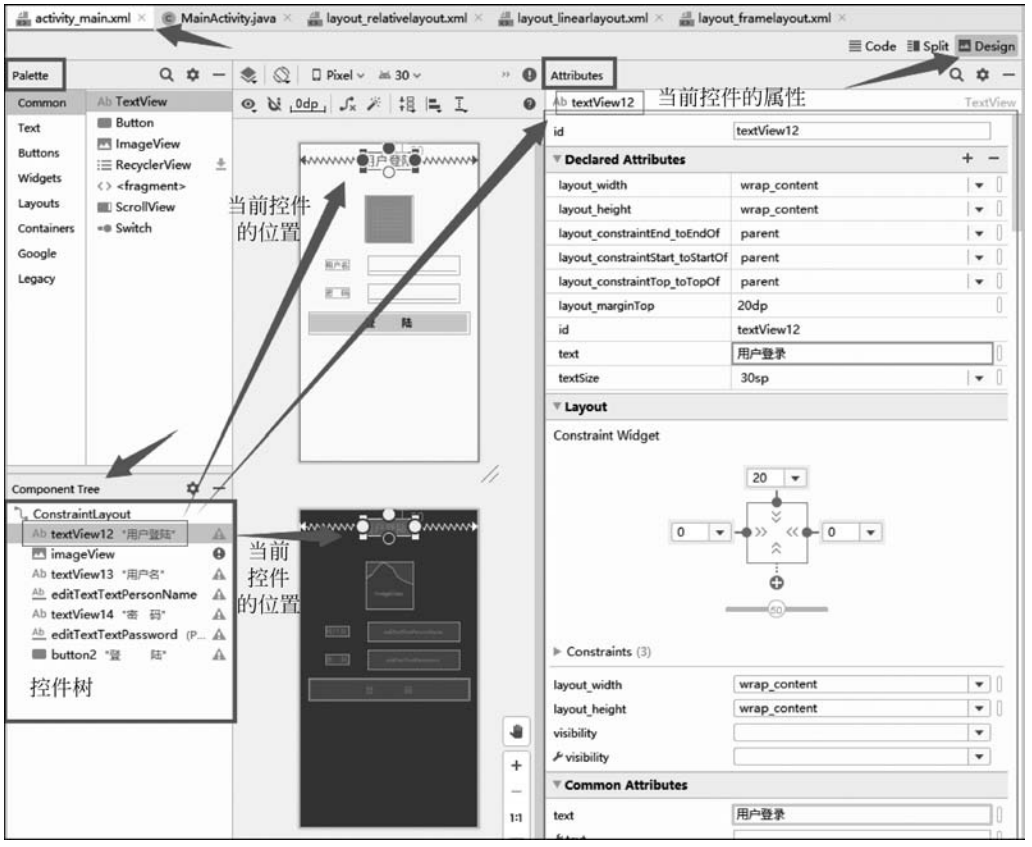

图 3.7 “用户登录”约束布局设计图

```
<?xml version = "1.0" encoding = "utf - 8"?>
< androidx.constraintlayout.widget.ConstraintLayout xmlns:android = "http://schemas.android.
com/apk/res/android"
    xmlns:app = "http://schemas.android.com/apk/res - auto"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    tools:context = ".MainActivity">
    < TextView
        android:id = "@ + id/textView12"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_marginTop = "20dp"
        android:text = "用户登录"
        android:textSize = "30sp"
        app:layout_constraintEnd_toEndOf = "parent"
        app:layout_constraintStart_toStartOf = "parent"
        app:layout_constraintTop_toTopOf = "parent" />
    < ImageView
        android:id = "@ + id/imageView"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_marginTop = "60dp"
        android:contentDescription = "@string/app_name"
```

```
        app:layout_constraintEnd_toEndOf = "parent"
        app:layout_constraintStart_toStartOf = "parent"
        app:layout_constraintTop_toBottomOf = "@id/textView12"
        app:srcCompat = "@drawable/ic_launcher_background"
        tools:ignore = "VectorDrawableCompat" />
    <TextView
        android:id = "@ + id/textView13"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_marginLeft = "60dp"
        android:layout_marginTop = "40dp"
        android:text = "用户名"
        android:textSize = "18sp"
        app:layout_constraintStart_toStartOf = "parent"
        app:layout_constraintTop_toBottomOf = "@id/imageView" />
    <EditText
        android:id = "@ + id/editTextTextPersonName"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:ems = "10"
        android:inputType = "textPersonName"
        app:layout_constraintBaseline_toBaselineOf = "@id/textView13"
        app:layout_constraintEnd_toEndOf = "parent"
        app:layout_constraintStart_toEndOf = "@id/textView13" />
    <TextView
        android:id = "@ + id/textView14"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_marginLeft = "60dp"
        android:layout_marginTop = "40dp"
        android:text = "密          码"
        android:textSize = "18sp"
        app:layout_constraintStart_toStartOf = "parent"
        app:layout_constraintTop_toBottomOf = "@id/textView13" />
    <EditText
        android:id = "@ + id/editTextTextPassword"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:ems = "10"
        android:inputType = "textPassword"
        app:layout_constraintBaseline_toBaselineOf = "@id/textView14"
        app:layout_constraintEnd_toEndOf = "parent"
        app:layout_constraintStart_toEndOf = "@id/textView14" />
    <Button
        android:id = "@ + id/button2"
        android:layout_width = "0dp"
        android:layout_height = "wrap_content"
        android:layout_margin = "20dp"
        android:text = "登          录"
        android:textSize = "24sp"
        android:textStyle = "bold"
        app:layout_constraintEnd_toEndOf = "parent"
        app:layout_constraintStart_toStartOf = "parent"
        app:layout_constraintTop_toBottomOf = "@id/editTextTextPassword" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

上述布局中的 app:layout_constraintXXX_toXXX 属性通过对控件的位置进行约束来实现其设置,这些属性的含义见表 3.2。

表 3.2 约束布局的常用属性及含义

属 性 名	含 义
layout_constraintTop_toTopOf	将所需视图的顶部与另一个视图的顶部对齐
layout_constraintTop_toBottomOf	将所需视图的顶部与另一个视图的底部对齐
layout_constraintBottom_toTopOf	将所需视图的底部与另一个视图的顶部对齐
layout_constraintBottom_toBottomOf	将所需视图的底部与另一个视图的底部对齐
layout_constraintLeft_toTopOf	将所需视图的左侧与另一个视图的顶部对齐
layout_constraintLeft_toBottomOf	将所需视图的左侧与另一个视图的底部对齐
layout_constraintLeft_toLeftOf	将所需视图的左边与另一个视图的左边对齐
layout_constraintLeft_toRightOf	将所需视图的左边与另一个视图的右边对齐
layout_constraintRight_toTopOf	将所需视图右对齐到另一个视图的顶部
layout_constraintRight_toBottomOf	将所需视图右对齐到另一个视图的底部
layout_constraintRight_toLeftOf	将所需视图的右边与另一个视图的左边对齐
layout_constraintRight_toRightOf	将所需视图的右边与另一个视图的右边对齐

(3) 为了给 ImageView 控件添加图像,在 MainActivity.java 文件中引入 ImageView 类。

```
import android.widget.ImageView;
```

(4) 修改与主 Activity 绑定的布局文件,代码如下。

```
setContentView(R.layout.activity_main);
ImageView iv = (ImageView)findViewById(R.id.imageView);
iv.setImageResource(R.mipmap.ic_launcher);
```

上述代码中 findVewById()方法将 id 为 imageView 的控件与 ImageView 类的变量 iv 关联,然后通过 ImageView 类的方法 setImageResource 为 imageView 控件设置图像。

视频讲解

3.2.5 表格布局

表格布局(TableLayout)是将布局页面划分为行、列构成的单元格。用< TableRow > </TableRow >标记表示单元格的一行,单元格的列数等于包含最多控件的 TableRow 的列数。直接在 TableLayout 中加的控件会占据一行。

TableLayout 可设置的属性包括全局属性及单元格属性。

(1) 全局属性也即列属性,有以下 3 个参数。

① android:stretchColumns:设置可伸展的列。该列可以沿行方向伸展,最多可占据一整行。

② android:shrinkColumns:设置可收缩的列。当该列包含的控件的内容太多,已经挤满所在行时,该子控件的内容将沿列方向显示。

③ android:collapseColumns:设置要隐藏的列。

示例:

```
android:stretchColumns = "0"        //第 0 列可伸展
android:shrinkColumns = "1,2"       //第 1,2 列皆可收缩
android:collapseColumns = " * "     // 隐藏所有列
```

说明：列可以同时具备 stretchColumns 及 shrinkColumns 属性，若同时具备，那么当该列的内容很多时，将“多行”显示其内容（这里不是真正的多行，而是系统根据需要自动调节该行的 layout_height）。

（2）单元格属性，有以下 2 个参数。

① android:layout_column：指定该单元格在第几列显示。

② android:layout_span：指定该单元格占据的列数（未指定时为 1）。

示例：

```
android:layout_column = "1"         //该控件显示在第 1 列
android:layout_span = "2"           //该控件占据 2 列
```

说明：一个控件也可以同时具备这两个属性。

【例 3-5】 演示表格布局编程方法。

（1）打开 LayoutDemo 项目，右击 res/layout 文件夹，选择 New→XML→Layout XML File，在弹出的对话框的 Layout File Name 栏填入 layout_tablelayout，在下方的 Root Tag 栏填入 TableLayout，创建一个表格布局文件。

（2）创建 6 个 TextView 控件，分别显示其坐标，代码如下。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< TableLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent" >
    < TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "直接占据一行"/>
    < TableRow >
        < TextView
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "第二行第一列"/>
        < TextView
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "第二行第二列"/>
        < TextView
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "第二行第三列"/>
    </TableRow >
    < TableRow >
        < TextView
            android:layout_width = "wrap_content"
```

```
            android:layout_height = "wrap_content"
            android:text = "第三行第一列"/>
        < TextView
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "第三行第二列"/>
    </TableRow >
</TableLayout >
```

图 3.8 表格布局效果图

(3) 在 MainActivity.java 代码中修改与主 Activity 绑定的布局文件,修改后的代码如下。

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

运行结果如图 3.8 所示。由运行结果很容易看出表格布局的特点,各控件分布于一个表格内。


视频讲解

3.2.6 网格布局

网格布局(GridLayout)是 Android 4.0 以上版本出现的,网格布局使用虚细线将布局划分为行、列和单元格,也支持控件在行、列上交错排列。

首先它与 LinearLayout 布局一样,也分为水平和垂直两种方式,默认是水平布局,一个控件挨着一个控件从左到右依次排列,但是通过指定 android:columnCount 属性设置列数后,控件会自动换行进行排列。另一方面,对于 GridLayout 布局中的控件,默认按照 wrap_content 的方式设置其显示。

其次,若要指定某控件显示在固定的行或列,只需设置该控件的 android:layout_row 和 android:layout_column 属性即可。但是需要注意:android:layout_row="0"表示从第一行开始,android:layout_column="0"表示从第一列开始,这与编程语言中一维数组的赋值情况类似。

最后,如果需要设置某控件跨越多行或多列,只需将该控件的 android:layout_rowSpan 或者 layout_columnSpan 属性设置为数值,再设置其 layout_gravity 属性为 fill 即可,前一个设置表明该控件跨越的行数或列数,后一个设置表明该控件填满所跨越的整行或整列。

3.2.7 布局的混合使用

单独使用某一种布局很难做出复杂美观的界面,所以布局往往不是单独使用的,而是恰当的布局嵌套使用,相同的布局可以嵌套,不同的布局也可以嵌套,如 3.4.2 节的主界面设计。


视频讲解

3.3 界面常用控件

3.3.1 TextView 和 EditText

TextView 是用于显示字符的控件,类似于 C# 和 Java 语言中的 Label 控件,但它支持显示多行文本及自动换行。EditText 则是用来输入和编辑字符的控件,具有编辑功能,包

括 Plain Text、Password、Password(Numeric)、E-mail、Phone、Postal Address、Multiline Text、Time、Date、Number、Number(Signed)、Number(Decimal)等控件。这两个控件经常一起使用。

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.
com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:text="@string/textview_user"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <EditText
        android:id="@+id/editTextDemo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:ems="10"
        android:inputType="textPersonName"
        android:hint="@string/edittext_name"
        android:textSize="18sp"
        android:autofillHints=""
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toBottomOf="@id/textView1" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

该布局创建了 TextView 和 EditText 控件,分别声明了 TextView 和 EditText 的 ID,以便于在代码中引用相应的控件对象。“@+id/TextView1”表示所设置的 ID 值,@表示后面的字符串是 ID 资源,加号(+)表示需要建立新资源名称,并添加到 R.java 文件中,但当 R.java 中已经存在同名变量 TextView1 时,该控件会使用这个已存在的值。android:layout_margin="20dp"表示设置控件四周的空隙尺寸为 20 个设备独立像素(device independent pixels)。EditText 中的 android:textSize="18sp"表示文字尺寸大小为 18 个缩放独立像素(scale-independent pixels)。sp 主要用作字体的单位,用此单位设置文字大小,可以在不同像素密度的屏幕上进行同比例的扩大缩小。android:ems 是一种长度单位,表示一个字符的标准长度,适用于动态布局。如果系统的默认字体大小为 1ems,则 android:ems="10"表示该控件可包含 10 个字符的大小,并且,如果窗口变大,该控件也会自动变大。android:hint 属性用于设置 EditText 控件输入框中的提示文字,这里取 strings.xml 文件中 edittext_name 元素的值,而 android:autofillHints 属性用于设置该控件的自动填写文字。如果 EditText 控件缺少这两个属性,则控件在“Design”视图中会出现警告符号 ⚠ 。android:inputType 设置编辑框输入的数值属性,常用数值属性及含义见表 3.3。

表 3.3 EditText 的 android:inputType 属性的值及含义

属性值	含义	属性值	含义
none	/	text	文本
textCapCharacters	字母大写	textCapWords	首字母大写
textCapSentences	仅第一个字母大写	textAutoCorrect	自动完成
textMultiLine	多行输入	textImeMultiLine	输入法多行(如果支持)
textNoSuggestions	不提示	textUri	网址
textEmailAddress	电子邮件地址	textEmailSubject	邮件主题
textShortMessage	短信息	textLongMessage	长信息
textPersonName	人名	textPostalAddress	地址
textPassword	密码	textVisiblePassword	可见密码
textWebEditText	作为网页表单的文本	textFilter	文本筛选过滤
textPhonetic	拼音输入	number	数字
numberSigned	带符号数字格式	numberDecimal	带小数点的浮点格式
phone	拨号键盘	datetime	时间日期
Date	日期键盘	time	时间键盘

为了在代码中引用 activity_main.xml 中设置的控件,首先需要在 MainActivity.java 代码中引入 android.widget 开发包,然后使用 findViewById()函数通过 ID 引用该控件,并把该控件赋值给创建的控件对象。该函数可以引用任何在 XML 文件中定义过 ID 的控件。setText()函数用来设置控件显示的内容。

```
package edu.cqut.commoncontroldemo;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.*;
import android.view.View;
public class MainActivity extends AppCompatActivity {
    EditText editText = null;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        editText = (EditText)findViewById(R.id.editTextDemo);
        editText.setText("请输入");
    }
}
```

程序运行结果如图 3.9 所示。

视频讲解

3.3.2 Button 和 ImageButton

Button 是常用的普通按钮控件,用户能够在该控件上单击,引发相应的响应事件。如果需要在按钮上显示图像,则可以使用 ImageButton 控件。

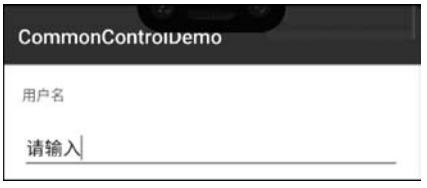

图 3.9 TextView 和 EditView 控件运行效果

【例 3-6】 演示 Button 和 ImageButton 控件编写方法。

(1) 在 CommonControlDemo 项目的 activity_main.xml 中分别添加 Button 和 ImageButton 控件,代码如下。

```
<Button
    android:id="@+id/button_OK"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="12dp"
    android:text="确定"
    app:layout_constraintStart_toStartOf="@id/editTextDemo"
    app:layout_constraintTop_toBottomOf="@id/editTextDemo" />
<ImageButton
    android:id="@+id/imageButton1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="12dp"
    android:layout_marginLeft="12dp"
    android:layout_marginTop="12dp"
    android:background="#FFFFFF"
    android:cropToPadding="true"
    app:layout_constraintStart_toEndOf="@id/button_OK"
    app:layout_constraintTop_toBottomOf="@id/editTextDemo" />
```

(2) Android 支持多种图形格式,如 png、ico、jpg 等,本例使用 jpg 格式。在 Android Studio 的 Project 视图中将 green_bk.jpg 文件复制到 app/src/res/mipmap-hdpi 文件夹中,更新 R.java 文件,选择菜单中的 Build→Rebuild Project 选项进行 R.java 更新。如果 R.java 文件不更新,则无法在代码中使用该资源。

(3) 在 MainActivity.java 代码中引用两个按钮,并让 ImageButton 显示图像 green_bk.jpg 内容。

```
ImageButton imageButton = (ImageButton)findViewById(R.id.imageButton1);
imageButton.setImageResource(R.drawable.green_bk);
Button button = (Button)findViewById(R.id.button_OK);
```

(4) 为了使两个按钮能够响应单击事件,需要在 onCreate()函数中为它们分别添加单击事件监听器,其代码如下。

```
//添加单击 button 事件的监听器
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        editText.setText("你单击了 button 按钮");
    }
});
//添加单击 imageButton 事件的监听器
imageButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
```

```
            editText.setText("你单击了 imageButton 按钮");
        }
    });
```


图 3.10　Button 和 ImageButton 运行效果

按钮对象通过调用 setOnClickeListener()函数,注册单击事件(Click)的监听器 View.OnClickListener(),该监听器接口中仅包含了 onClick()抽象函数。我们实现(Override)该函数。当按钮控件从 Android 界面框架中接收到事件后,首先检查这个事件是否是单击事件,如果是,同时 Button 又注册了监听器,则会调用该监听器中实现的 onClick()函数。程序运行结果如图 3.10 所示。

3.3.3　CheckBox 和 RadioButton

CheckBox 是可以同时选择多个选项的控件,而 RadioButton 则是仅可以选择一个选项的控件。RadioGroup 是 RadioButton 的承载体,程序运行时不可见。在一个 RadioGroup 中,用户仅能选择其中一个 RadioButton。

【例 3-7】 演示 CheckBox 和 RadioButton 控件编写方法。

(1) 在 CommonControlDemo 项目的 activity_main.xml 中分别添加 CheckBox 和 RadioButton 控件的代码如下。

```
<CheckBox
    android:id="@+id/checkBox1"
    android:layout_width="86dp"
    android:layout_height="29dp"
    android:layout_marginTop="16dp"
    android:text="多选框 1"
    app:layout_constraintStart_toStartOf="@id/button_OK"
    app:layout_constraintTop_toBottomOf="@id/imageButton1" />
<CheckBox
    android:id="@+id/checkBox2"
    android:layout_width="87dp"
    android:layout_height="28dp"
    android:layout_marginStart="12dp"
    android:layout_marginLeft="12dp"
    android:text="多选框 2"
    app:layout_constraintBaseline_toBaselineOf="@id/checkBox1"
    app:layout_constraintStart_toEndOf="@id/checkBox1" />
<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/checkBox1"
    android:layout_marginTop="16dp"
    android:text="请选择单选按钮"
    app:layout_constraintStart_toStartOf="@id/checkBox1"
```

```
    app:layout_constraintTop_toBottomOf = "@id/checkBox1" />
<RadioGroup
    android:id = "@ + id/RadioGroup01"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:layout_below = "@id/textView2"
    android:layout_marginTop = "16dp"
    android:orientation = "horizontal"
    app:layout_constraintStart_toStartOf = "@id/textView2"
    app:layout_constraintTop_toBottomOf = "@id/textView2">
    <RadioButton
        android:id = "@ + id/radioButton1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "选择项 1"
        android:layout_margin = "5dp"/>
    <RadioButton
        android:id = "@ + id/radioButton2"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "选择项 2"
        android:layout_margin = "5dp"/>
</RadioGroup>
```

（2）在 MainActivity.java 代码中引用创建的 CheckBox 和 RadioButton 控件，并在 onCreate()函数中为它们添加单击事件监听器，代码如下。

```
public class MainActivity extends AppCompatActivity
{
    …
    CheckBox checkBox1 = null;
    CheckBox checkBox2 = null;
    RadioButton radioButton1 = null;
    RadioButton radioButton2 = null;
@Override
    protected void onCreate(Bundle savedInstanceState) {
        …
        checkBox1 = (CheckBox)findViewById(R.id.checkBox1);
        checkBox2 = (CheckBox)findViewById(R.id.checkBox2);
        radioButton1 = (RadioButton)findViewById(R.id.radioButton1);
        radioButton2 = (RadioButton)findViewById(R.id.radioButton2);

        //将多个 CheckBox 控件注册到一个选择单击事件的监听器上
        CheckBox.OnClickListener checkboxListener = new CheckBox.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (checkBox1.isChecked() && checkBox2.isChecked())
                    editText.setText("你选择了多选框 1 和多选框 2");
                else if (checkBox1.isChecked())
                    editText.setText("你选择了多选框 1");
                else if (checkBox2.isChecked())
```

```
                editText.setText("你选择了多选框 2");
            else
                editText.setText("");
        }
    };
    checkBox1.setOnClickListener(checkboxListener);
    checkBox2.setOnClickListener(checkboxListener);

    //将多个 RadioButton 控件注册到一个单击事件的监听器上
    RadioButton.OnClickListener radioButtonListener = new RadioButton.OnClickListener()
    {
        @Override
        public void onClick(View v){
            if (radioButton1.isChecked() && radioButton2.isChecked())
                editText.setText("你选择了单选框 1 和单选框 2");
            else if (radioButton1.isChecked())
                editText.setText("你选择了单选框 1");
            else if (radioButton2.isChecked())
                editText.setText("你选择了单选框 2");
            else
                editText.setText("");
        }
    };
    radioButton1.setOnClickListener(radioButtonListener);
    radioButton2.setOnClickListener(radioButtonListener);
  }
}
```

视频讲解

图 3.11　CheckBox 和 RadioButton 运行效果

CheckBox 和 RadioButton 控件运行效果如图 3.11 所示。

3.3.4　Spinner 和 ListView

Spinner 是从多个选项中选择一个选项的控件,类似于桌面程序的组合框(ComboBox),但没有组合框的下拉菜单,而是使用浮动菜单为用户提供选择。ListView 是用于垂直显示的列表控件,如果显示内容过多,则会出现垂直滚动条。

这两个控件在界面设计中经常使用,其原因是它们能够通过适配器将数据和显示控件绑定,且支持单击事件,可以用少量代码实现复杂的选项功能。Spinner 和 ListView 控件效果如图 3.12 所示。

Spinner 和 ListView 的直接父类是 ViewGroup,其中定义了子 View 的排列规则。Spinner 及 ListView 和所要展示的内容(即数据源)之间需要 Adapter(适配器)来实现。Adapter 是一个桥梁,如图 3.13 所示,对 ListView 和 Spinner 的数据进行管理。

图 3.12　Spinner 和 ListView 控件

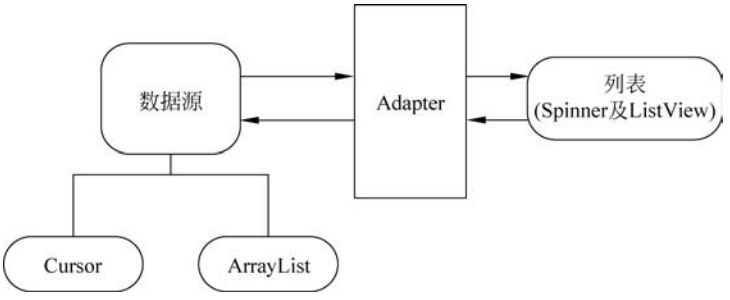

图 3.13　数据源、Adapter 和列表间的关系图

Adapter 是一个接口，图 3.14 列出了 Android 中与 Adapter 有关的所有接口、类的完整层级图，比较常用的有 BaseAdapter、SimpleAdapter、ArrayAdapter、SimpleCursorAdapter 等。其中 BaseAdapter 是一个抽象类，继承它需要较多的方法，所以具有较高的灵活性。ArrayAdapter 支持泛型操作，最为简单，只能展示一行文本。SimpleAdapter 有最好的扩充性，可以自定义各种效果。

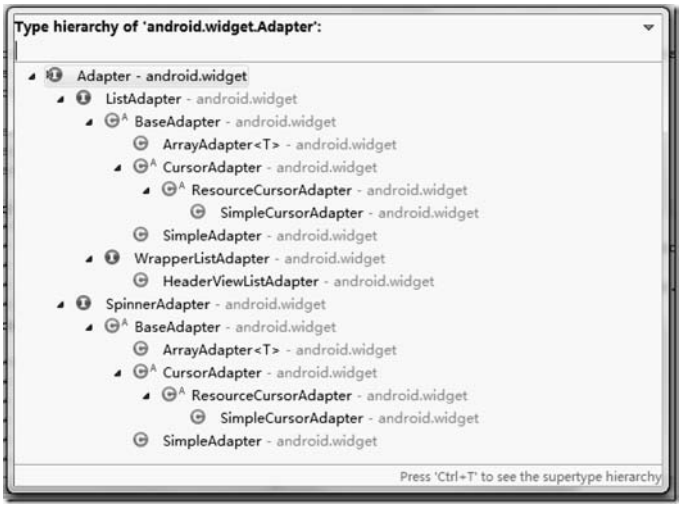

图 3.14　Android 中所有的 Adapter 一览

Spinner 和 ListView 显示前要使用 setAdapter()方法，ListView 本身继承自 ViewGroup，只设定它里面的 View 的排列规则，不设定其是什么样的，而 View 是什么样的需要靠 ListAdapter 里面的 getView 方法来确定，只要设置不同的 ListAdapter 实例对象，就会生成不一样的 ListView。

【例 3-8】 使用 ArrayAdapter 演示 Spinner 和 ListView 控件编程方法。

(1) 在 CommonControlDemo 项目的 activity_main.xml 中分别添加 Spinner 和 ListView 控件的代码如下。

```
<Spinner
    android:id="@+id/spinner1"
    android:layout_width="wrap_content"
```

```
        android:layout_height = "wrap_content"
        android:layout_marginTop = "16dp"
        app:layout_constraintStart_toStartOf = "@id/RadioGroup01"
        app:layout_constraintTop_toBottomOf = "@id/RadioGroup01" />
    <ListView
        android:id = "@ + id/listView1"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        app:layout_constraintTop_toBottomOf = "@id/spinner1"
        app:layout_constraintStart_toStartOf = "@id/spinner1"
        android:layout_marginTop = "23dp"/>
```

(2) 在 MainActivity.java 代码中引用创建的 Spinner 控件,并在 onCreate()函数中添加单击子项选中事件监听器,代码如下。

```
spinner = (Spinner)findViewById(R.id.spinner1);
List<String> listspinner = new ArrayList<String>();
listspinner.add("Spinner 子项 1");
listspinner.add("Spinner 子项 2");
//使用 ArrayAdapter 数组适配器将界面控件和底层数据绑定在一起,即 Spinner 和 ArrayList 绑定
ArrayAdapter<String> adapter1 = new ArrayAdapter<String>(this,
        android.R.layout.simple_spinner_item, listspinner);
//设置 Spinner 浮动菜单显示方式
adapter1.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
spinner.setAdapter(adapter1);     //完成绑定
//添加单击 spinner 选项的事件监听器
spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id)
    {
        editText.setText(((TextView)view).getText());
    }
    @Override
    public void onNothingSelected(AdapterView<?> arg0)
    {
        editText.setText("");
    }
});
```

上面代码中 android.R.layout.simple_spinner_dropdown_item 为 Spinner 浮动菜单显示的方式之一,效果如图 3.15 所示。另一种浮动菜单是 android.R.layout.simple_ spinner_item,显示效果如图 3.16 所示。

图 3.15 Spinner 的 dropdown 菜单

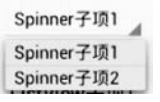

图 3.16 Spinner 的 item 菜单

AdapterView.OnItemSelectedListener()是 Spinner 子项选中事件监听器,需要实现 onItemSelected()和 onNothingSelected()两个函数。其中,onItemSelected()有 4 个参数,参数 parent 表示控件适配器,这里就是 Spinner;参数 view 表示适配器内部被选中的控件,即 Spinner 中的子项;参数 position 表示选中的子项的位置;参数 id 表示选中的子项的行号。

(3) 在 MainActivity.java 代码中引用创建的 ListView 控件,并在 onCreate()函数中添加单击子项选中事件监听器,代码如下。

```
listview = (ListView)findViewById(R.id.listView1);
List<String> list = new ArrayList<String>();
for (int i = 1; i<10; i++)
    list.add("ListView 子项" + i);
//使用 ArrayAdapter 数组适配器将界面控件和底层数据绑定在一起,即 ListView 和 ArrayList 绑定
ArrayAdapter<String> adapter2 = new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1, list);
listview.setAdapter(adapter2);          //完成绑定
//添加单击 ListView 选项的事件监听器
listview.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id)
    {
        editText.setText(((TextView)view).getText());
    }
});
```

为 ListView 添加了 10 个子项,这样当屏幕显示不下 ListView 控件列表时,会出现垂直滑块,通过上下滑动,可以显示其余子项。代码中 onItemClick()函数的参数含义与之前 onItemSelected()函数参数含义相同。图 3.17 是 CommonControlDemo 项目的效果图。

3.3.5 自定义列表


视频讲解

对于不同的适配器类型,ArrayAdapter 是最简单的一种,就如例 3-8 演示的一样,只能显示一行文字,而 SimpleAdapter 的扩展性最好,可以定义各种各样的布局,也可以放上 ImageView,还可以放上 Button 和 CheckBox 等,因此可以用它来生成自定义列表。下面的例子演示了如何生成一个带图片的菜单列表。

【例 3-9】 使用 SimpleAdapter 演示自定义列表编程方法。

图 3.18 是例 3-9 的运行效果,下面来实现该列表。

创建名为 CustomListViewDemo 的新项目,包名为 edu.cqut.customlistviewdemo。切换到 Project 视图。

(1) 进入项目的 app\src\main\res 目录,右击 res 文件夹,在弹出菜单中选择 New→Directory,创建 raw 文件夹,在其中存放 4 张菜品图片。

(2) 在 res\layout 文件夹中创建名为 listitem.xml 的布局文件。该布局文件采用混合线性布局,用于定义 ListView 中每行显示的控件,依次为菜品图片、菜品名称和菜品简介。代码如下。

图 3.17　CommonControlDemo 效果图

图 3.18　自定义列表运行效果

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "horizontal"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:id = "@ + id/listitem">
    < ImageView
        android:id = "@ + id/img"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_margin = "5dp"/>
    < LinearLayout
        android:orientation = "vertical"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content">
        < TextView android:id = "@ + id/title"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:textColor = "#00000000"
            android:textIsSelectable = "false"
            android:textSize = "20sp" />
        < TextView
            android:id = "@ + id/info"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
```

```
            android:textIsSelectable = "false"
            android:textSize = "15sp"/>
    </LinearLayout >
</LinearLayout >
```

(3) 修改 layout 文件夹中的 activity_main. xml 的布局文件,代码如下。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:id = "@ + id/caipinlistlayout"
    android:orientation = "vertical" >
    < TableRow
        android:id = "@ + id/DishHead"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content">
        < TextView
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:textColor = "#000000"
            android:textSize = "20sp"
            android:text = " **** 菜     单 **** "/>
    </TableRow >
    < ListView
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:id = "@ + id/ListViewDemo" />
</LinearLayout >
```

该布局文件显示菜单页面,第一行为 TableRow 布局,其中 TextView 控件用于显示标题,TableRow 布局的下一行为 ListView 控件,该控件显示各个具体的菜品,而这个 ListView 控件的布局将使用 listitem. xml 文件。

(4) 在项目 app\src\main\java\edu. cqut. customlistviewdemo 文件夹中添加名为 Dish. java 的 Java 文件,定义用于菜品 Dish 类。

```
public class Dish
{
    public String mName;            //菜名
    public int mImage;              //菜品图像
    public String mInfo;            //介绍
}
```

(5) 在 MainActivity. java 文件的 MainActivity 类中创建适配器和数据源。

```
static List < Map < String, Object >> mfoodinfo;     //菜品数据源列表,由 HashMap 表构成
public ListView mlistview;
static SimpleAdapter mlistItemAdapter;
public ArrayList < Dish > mDishes = new ArrayList < Dish >();      //菜品列表
```

编写一个函数用于将具体的菜单数据填入 mDishes。

```
private void FillDishesList()
{
        Dish theDish = new Dish();
        //添加菜品
        theDish.mName = "宫保鸡丁";
        theDish.mInfo = "北京宫廷菜,入口鲜辣香脆";
        theDish.mImage = (R.raw.food01gongbaojiding);
        mDishes.add(theDish);

        theDish = new Dish();
        theDish.mName = "椒盐玉米";
        theDish.mInfo = "色香味俱全,浙江菜";
        theDish.mImage = (R.raw.food02jiaoyanyumi);
        mDishes.add(theDish);

        theDish = new Dish();
        theDish.mName = "清蒸武昌鱼";
        theDish.mInfo = "湖北鄂州传统名菜";
        theDish.mImage = (R.raw.food03qingzhengwuchangyu);
        mDishes.add(theDish);

        theDish = new Dish();
        theDish.mName = "鱼香肉丝";
        theDish.mInfo = "经典汉族传统川菜";
        theDish.mImage = (R.raw.food04yuxiangrousi);
        mDishes.add(theDish);
}
```

SimpleAdapter 适配器的数据源是 HashMap 列表的数据结构,函数 getFoodData()负责将 ArrayList < Dish >的数据结构转换成适用于 SimpleAdapter 的 List < Map < String, Object >>数据结构。

```
Private ArrayList < Map < String, Object >> getFoodData()
{
    ArrayList < Map < String, Object >> fooddata = new ArrayList < Map < String,Object >>();
    //将菜品信息填充进 foodinfo 列表
    int s = mDishes.size();                 //得到菜品数量
    for (int i = 0; i < s; i++) {
        Dish theDish = mDishes.get(i);      //得到当前菜品
        Map < String, Object > map = new HashMap < String, Object >();
        map.put("image", theDish.mImage);
        map.put("title", theDish.mName);
        map.put("info", theDish.mInfo);
        fooddata.add(map);
    }
    return fooddata;
}
```

(6) 修改 onCreate()函数如下。

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    //生成菜单信息列表
    FillDishesList();
    mlistview = (ListView) findViewById(R.id.ListViewDemo);
    mfoodinfo = getFoodData();
    //构造 SimpleAdapter 适配器,将它和 ListView 自定义的布局文件、List 数据源关联
    mlistItemAdapter = new SimpleAdapter(this,mfoodinfo,       //数据源,列表的每一节对应
                                                               //ListView 的一行
            R.layout.listitem,     //ListItem 的 XML 实现动态数组与 listitem 对应的子项,必
                                   //须与 mfoodinfo 中的各资源名字一致
            new String[] {"image", "title", "info"},
            //ListItem 的 XML 文件里面的 1 个 ImageView,3 个 TextView
            new int[]{ R.id.img, R.id.title, R.id.info});
    mlistItemAdapter.notifyDataSetChanged();
    mlistview.setAdapter(mlistItemAdapter);
    //设置 ListView 选项单击监听器
    this.mlistview.setOnItemClickListener(new OnItemClickListener(){
        @Override
        public void onItemClick(AdapterView<?> arg0,  //选项所属的 ListView
                                View arg1,            //被选中的控件,即 ListView 中被选中
                                                      //的子项
                                int arg2,             //被选中子项在 ListView 中的位置
                                long arg3)            //被选中子项的行号
        {
             ListView templist = (ListView)arg0;
             View mView = templist.getChildAt(arg2);
             final TextView tvTitle = (TextView)mView.findViewById(R.id.title);
            Toast.makeText(MainActivity.this, tvTitle.getText().toString(), Toast.LENGTH_
LONG).show();
        }
    });
}
```

3.4 “移动点餐系统”用户界面

视频讲解

3.4.1 实体模型类设计

在设计用户界面之前,先进行“移动点餐系统”的实体模型类设计。该项目实体主要有菜品、菜单、订单、订单细目、用户及购物车。

(1) 设计菜品实体模型类,其代码如下。

```
public class Dish
{
    public int mId = -1;              //菜品 ID
    public String mName;              //菜名
    public int mImage;                //菜品图像
    public float mPrice;              //价格
}
```

(2) 设计菜单实体模型类,其代码如下。

```
import java.util.ArrayList;
public class Dishes
{
    public ArrayList<Dish> mDishes;        //菜品列表
    public int GetDishQuantity(){
        return mDishes.size();
    }
    public Dish GetDishbyIndex(int i){
        return mDishes.get(i);
    }
    public Dish GetDishbyName(String dishName){
        int s = mDishes.size();
        for (int i=0; i<s; i++) {
            Dish theDish = mDishes.get(i);
            if (dishName.equals(theDish.mName)) {
                return theDish;
            }
        }
        return null;
    }
}
```

(3) 设计订单细目实体模型类,该类用于购物车类中,其代码如下。

```
public class OrderItem
{
    public Dish mOneDish;            //该订购细目中的一个菜品
    public int mQuantity=0;          //该菜品的数量
    OrderItem(Dish theDish, int quantity)
    {
        mOneDish = theDish;
        mQuantity = quantity;
    }
    public float GetItemTotalPrice()
    {
        return mOneDish.mPrice * mQuantity;
    }
}
```

(4) 设计购物车实体模型类,其代码如下。

```
import java.util.ArrayList;
public class ShoppingCart
{
    public String mUserName;                          //购物车所属用户的用户名
    private ArrayList<OrderItem> mOrderItems;         //存放已点菜品的链表
    ShoppingCart(String userName){
        mUserName = userName;
        mOrderItems = new ArrayList<OrderItem>();
    }
```

```
ShoppingCart(String userName, ArrayList<OrderItem> orderitems){
    mUserName = userName;
    mOrderItems = orderitems;
}
public int GetOrderItemsQuantity() {
    int s = mOrderItems.size();
    return s;
}
public OrderItem GetItembyIndex(int i){
    return mOrderItems.get(i);
}
public boolean DeleteItemByIndex(int i) {
    int s = mOrderItems.size();
    if (i>=0 && i<s) {
        mOrderItems.remove(i);
        return true;
    }
    return false;
}
//计算购物车中菜品总价
public float GetCartTotalPrice() {
    float totalPrice = 0;
    if (!mOrderItems.isEmpty()){
        int s = mOrderItems.size();
        for (int i=0; i<s; i++)
            totalPrice += ((OrderItem)mOrderItems.get(i)).GetItemTotalPrice();
    }
    return totalPrice;
}
//根据菜品信息将菜品插入已点菜品链表中,返回插入菜品在链表中的索引
public int AddOneOrderItem(Dish dish, int num){
    int index = GetDishIndex(dish.mName);       //查询该菜是否已点
    if (index == -1) {                          //该菜没点
        if (num>0) {                            //将其插入链表末尾
            OrderItem theItem = new OrderItem(dish, num);
            mOrderItems.add(theItem);
            return mOrderItems.size()-1;
        }
        else
            return -1;
    }
    else {                                      //该菜已点
        if (num<=0 ) {                          //如果点餐数量小于等于0表示用户要删
                                                //除该菜品
            DeleteOneOrderItem(dish.mName);
            return -1;
        }
        else {                                  // 只需修改链表中相应菜的数量
            OrderItem theItem = new OrderItem(dish, num);
            mOrderItems.set(index, theItem);
```

```
                return index;
            }
        }
    }
    //根据菜名从已点菜品链表中将该菜品删除
    public void DeleteOneOrderItem(String dishName) {
        if (!mOrderItems.isEmpty()) {
            int s = mOrderItems.size();
            for (int i = 0; i < s; i++) {
                String theName = ((OrderItem)mOrderItems.get(i)).mOneDish.mName;
                if (dishName.equals(theName)) {
                    mOrderItems.remove(i);
                    break;
                }
            }
        }
    }
    //根据菜名在已点菜品链表中修改该菜数量,返回修改菜品在购物车中的位置,当菜品在购物
    //车中不存在时返回 - 1
    public int ModifyOneOrderItem(String dishName, int num) {
        if (!mOrderItems.isEmpty()) {
            int s = mOrderItems.size();
            for (int i = 0; i < s; i++) {
                OrderItem theItem = (OrderItem)mOrderItems.get(i);
                if (dishName.equals(theItem.mOneDish.mName)) {
                    theItem.mQuantity = num;
                    mOrderItems.set(i, theItem);
                    return i;
                }
            }
        }
        return - 1;
    }
    //根据菜品名在已点菜品链表中查询该菜是否已点,返回已点菜品在链表中的位置,若没有返
    //回 - 1
    private int GetDishIndex(String dishName)
    {
        if (!mOrderItems.isEmpty()) {
            int s = mOrderItems.size();
            for (int i = 0; i < s; i++) {
                OrderItem theItem = (OrderItem)mOrderItems.get(i);
                if (dishName.equals(theItem.mOneDish.mName)) {
                    return i;
                }
            }
        }
        return - 1;
    }
}
```

(5) 设计订单实体模型类,其代码如下。

```
public class Order
{
   public long mId = -1;                    //订单号
   public ShoppingCart mOrderItems;         //存放已点菜品的链表(已点菜品由购物车生成)
   public String mOrderTime;                //订单生效时间
   public String mSeatName;                 //餐厅就餐的餐位号
   public Boolean mIsFinished;              //订单是否配送完成
   public Order(String userid) { mOrderItems = new ShoppingCart(userid); }
   public Order(long orderId, ShoppingCart cart, String time, String seat, Boolean fin)
   {
      mId = orderId;
      this.mOrderItems = cart;
      mOrderTime = time;
      mSeatName = seat;
      mIsFinished = fin;
   }
}
```

(6) 设计用户实体模型类,其代码如下。

```
public class MyUser
{
    public String mUserid = "x";            //用户名
    public String mSeatname = "";           //桌名或房间号
    public String mPassword = "0";          //用户密码
    public String mUserphone = "";          //用户手机号
    public String mUseraddress = "";        //用户地址
    public Boolean mIslogined = false;      //用户登录状态
}
```

(7) 使用 Android Application 传递共享数据。

Android 的 Application 同 Activity 和 Service 一样,都是 Android 框架的组成部分。它通常在 APP 启动时就自动创建,在 APP 中是一个单实例模式,且是整个程序生命周期最长的对象。所有的 Activity 和 Service 都共用一个 Application,所以常用来进行共享数据、数据缓存和数据传递。

为了使用户、购物车等对象在整个程序生命周期中都被访问到,设计一个派生自 Application 的 MyApplication 类,存放这些全局变量。

```
import java.util.ArrayList;
import android.content.Context;
import android.app.Application;
public class MyApplication extends Application        //该类用于保存全局变量
{
    MyUser g_user;                                    //用户
    ShoppingCart g_cart;                              //与登录用户相关联的购物车
    ArrayList<Order> g_orders;                        //与登录用户相关联的订单
    Dishes g_dishes;                                  //菜品列表
    public String g_ip = "";                          //TCP 通信时店面服务器 IP 地址
    public String g_http_ip = "";                     //用 HTTP 通信时的店面 IP 地址
```

```
    public int g_communiMode = 1;          //通信模式,1 为 TCP 通信,2 为 HTTP 通信
    public int g_objPort = 35885;          //店面服务器监听端口号
    Context g_context;
}
```

在 AndroidManifest.xml 文件中的<application>标签中添加 android:name 属性,设置定义的 Application 派生类。

```
<application
    android:name = ".MyApplication"
    ...
    <activity android:name = ".MainActivity">
        <intent-filter>
            <action android:name = "android.intent.action.MAIN" />
            <category android:name = "android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
```

3.4.2 主界面设计

该系统用户主界面如图 3.19 所示,分为 5 个部分,图 3.19(a)为初始界面。其操作流程如下。

(1) 用户登录及注册:用户单击“登录”按钮后弹出“登录及注册”对话框,进行登录或注册操作。只有登录用户才可以进行“个人中心”“点餐”“外卖”“我的订单”的操作,非登录用户系统会提示需进行登录才能进行下一步操作。用户登录后“登录”按钮会切换成“注销”按钮,如图 3.19(b)所示。

(a) 非登录用户界面　　(b) 登录用户界面

图 3.19　移动点餐系统主界面

(2) 个人信息查询和修改:登录用户单击“个人中心”按钮切换到“用户信息”页面,进行信息查询和修改操作。

(3) 点餐:用户单击“点餐”按钮后,首先弹出一个对话框让其输入餐桌号或包间号,输入完毕后切换到“菜品”页面供用户点餐。

(4) 外卖:外卖用户单击“外卖”按钮后会直接进入“菜品”页面进行点餐操作。

(5) 订单查询:登录用户单击“我的订单”按钮后进入“我的订单”页面,进行订单查询操作。

下面来设计主界面。

(1) 将主界面所用图片根据图像分辨率大小复制到 MobileOrderFood 项目的 res\mipmap-*** 文件夹中。

(2) 修改 layout 目录中的 activity_main.xml 布局文件如下。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< androidx.constraintlayout.widget.ConstraintLayout xmlns:android = "http://schemas.android.
com/apk/res/android"
    xmlns:app = "http://schemas.android.com/apk/res - auto"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:layout_margin = "5dp">
    < ImageView
        android:id = "@ + id/homeImageView"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:adjustViewBounds = "true"
        android:cropToPadding = "true"
        android:scaleType = "centerCrop"
        app:layout_constraintEnd_toEndOf = "parent"
        app:layout_constraintStart_toStartOf = "parent"
        app:layout_constraintTop_toTopOf = "parent"
        app:srcCompat = "@mipmap/diancanlogo" />
    < LinearLayout
        android:id = "@ + id/linearLayout1"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:orientation = "horizontal"
        android:layout_marginTop = "5dp"
        app:layout_constraintEnd_toEndOf = "parent"
        app:layout_constraintStart_toStartOf = "parent"
        app:layout_constraintTop_toBottomOf = "@id/homeImageView">
        < ImageButton
            android:id = "@ + id/imgBtnRest"
            style = "@android:style/Widget.ImageButton"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_marginRight = "3dp"
            android:layout_weight = "1"
            android:adjustViewBounds = "true"
            android:background = "#FFFFFF"
            android:cropToPadding = "true"
            android:scaleType = "centerCrop"
            app:srcCompat = "@mipmap/diancan" />
        < ImageButton
            android:id = "@ + id/imgBtnTakeout"
            style = "@android:style/Widget.ImageButton"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_marginLeft = "3dp"
            android:layout_weight = "1"
            android:adjustViewBounds = "true"
            android:background = "#FFFFFF"
            android:cropToPadding = "true"
            android:scaleType = "centerCrop"
            app:srcCompat = "@mipmap/waimai" />
    </LinearLayout >
    < LinearLayout
        android:id = "@ + id/linearLayout2"
        android:layout_width = "match_parent"
```

```
            android:layout_height = "wrap_content"
            android:orientation = "horizontal"
            android:layout_marginTop = "5dp"
            app:layout_constraintEnd_toEndOf = "parent"
            app:layout_constraintStart_toStartOf = "parent"
            app:layout_constraintTop_toBottomOf = "@id/linearLayout1">
            < ImageButton
                android:id = "@ + id/imgBtnUserInfo"
                style = "@android:style/Widget. ImageButton"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:layout_marginRight = "3dp"
                android:layout_weight = "1"
                android:adjustViewBounds = "true"
                android:background = "#FFFFFF"
                android:cropToPadding = "true"
                android:scaleType = "centerCrop"
                app:srcCompat = "@mipmap/gerenzhongxin" />
            < ImageButton
                android:id = "@ + id/imgBtnLogin"
                style = "@android:style/Widget. ImageButton"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:layout_marginLeft = "3dp"
                android:layout_weight = "1"
                android:adjustViewBounds = "true"
                android:background = "#FFFFFF"
                android:cropToPadding = "true"
                android:scaleType = "centerCrop"
                android:visibility = "visible"
                app:srcCompat = "@mipmap/denglu" />
            < ImageButton
                android:id = "@ + id/imgBtnLogout"
                style = "@android:style/Widget. ImageButton"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:layout_marginLeft = "3dp"
                android:layout_weight = "1"
                android:background = "#FFFFFF"
                android:adjustViewBounds = "true"
                android:cropToPadding = "true"
                android:scaleType = "centerCrop"
                android:visibility = "gone"
                app:srcCompat = "@mipmap/zhuxiao" />
        </LinearLayout >
        < ImageButton
            android:id = "@ + id/imgBtnMyOrder"
            android:layout_height = "wrap_content"
            android:layout_width = "match_parent"
            android:layout_marginTop = "5dp"
            android:background = "@mipmap/wodedingdan"
            app:layout_constraintStart_toStartOf = "parent"
            app:layout_constraintTop_toBottomOf = "@id/linearLayout2"/>
    </androidx. constraintlayout. widget. ConstraintLayout >
```

从上面布局文件中可以看到，主界面采用的是约束布局中嵌套线性布局的混合布局形式。约束布局的顶部包含一个 ImageView 控件作为软件的 LOGO，该控件加载“diancanlogo.jpg”图像资源；其下是 2 个水平线性布局，每个水平布局中嵌套 2 个 ImageButton 控件，最后是 1 个 ImageButton 控件，将这些控件的背景设置为相应图片。

（3）在项目的 res 目录下创建 raw 文件夹，在其中存放 4 张菜品图片。

（4）在项目的 MainActivity.java 文件中创建相应按钮的对象，添加按钮监听事件。

```
package edu.cqut.MobileOrderFood;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.*;
import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {
    public static MyApplication mAppInstance;                    //用来访问程序全局变量
    public ImageButton mImgBtnLogin, mImgBtnLogout;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mAppInstance = (MyApplication)getApplication();          //获得全局变量对象
        mAppInstance.g_context = getApplicationContext();
        mAppInstance.g_user = new MyUser();                      //创建用户
        mAppInstance.g_orders = new ArrayList<Order>();          //创建订单列表
        mAppInstance.g_dishes = new Dishes();
        mAppInstance.g_dishes.mDishes = FillDishesList();        //填充菜品列表

        ImageButton imgBtnRest = (ImageButton)findViewById(R.id.imgBtnRest);
        ImageButton imgBtnTakeout = (ImageButton)findViewById(R.id.imgBtnTakeout);
        ImageButton imgBtnUserInfo = (ImageButton)findViewById(R.id.imgBtnUserInfo);
        ImageButton imgBtnMyOrder = (ImageButton)findViewById(R.id.imgBtnMyOrder);
        mImgBtnLogin = (ImageButton)findViewById(R.id.imgBtnLogin);
        mImgBtnLogout = (ImageButton)findViewById(R.id.imgBtnLogout);
        //将各图像按钮注册到 myImageButton 单击事件监听器
        imgBtnRest.setOnClickListener(new myImageButtonListener());
        imgBtnTakeout.setOnClickListener(new myImageButtonListener());
        imgBtnUserInfo.setOnClickListener(new myImageButtonListener());
        imgBtnRest.setOnClickListener(new myImageButtonListener());
        mImgBtnLogin.setOnClickListener(new myImageButtonListener());
        mImgBtnLogout.setOnClickListener(new myImageButtonListener());
    }
    private ArrayList<Dish> FillDishesList()
    {
        ArrayList<Dish> theDishesList = new ArrayList<Dish>();
        Dish theDish = new Dish();
        //添加菜品
        theDish.mId = 1001;
        theDish.mName = "宫保鸡丁";
        theDish.mPrice = (float) 20.0;
        theDish.mImage = (R.raw.food01gongbaojiding);
        theDishesList.add(theDish);
        theDish = new Dish();
        theDish.mId = 1002;
```

```
            theDish.mName = "椒盐玉米";
            theDish.mPrice = (float) 24.0;
            theDish.mImage = (R.raw.food02jiaoyanyumi);
            theDishesList.add(theDish);
            theDish = new Dish();
            theDish.mId = 1003;
            theDish.mName = "清蒸武昌鱼";
            theDish.mPrice = (float) 48.0;
            theDish.mImage = (R.raw.food03qingzhengwuchangyu);
            theDishesList.add(theDish);
            theDish = new Dish();
            theDish.mId = 1004;
            theDish.mName = "鱼香肉丝";
            theDish.mPrice = (float) 20.0;
            theDish.mImage = (R.raw.food04yuxiangrousi);
            theDishesList.add(theDish);
            return theDishesList;
        }
        public class myImageButtonListener implements View.OnClickListener
        {
            @Override
            public void onClick(View v) {
                switch (v.getId())
                {
                    case R.id.imgBtnRest:
                        Toast.makeText(MainActivity.this, "单击了点餐按钮!", Toast.LENGTH_
    LONG).show();
                        return;
                    case R.id.imgBtnTakeout:
                        Toast.makeText(MainActivity.this, "单击了外卖按钮!", Toast.LENGTH_
    LONG).show();
                        return;
                    case R.id.imgBtnLogin:          //用户未登录时该按钮才会出现
                        Toast.makeText(MainActivity.this, "单击了登录按钮!", Toast.LENGTH_
    LONG).show();
                        //隐藏"登录"按钮,显示"注销"按钮
                        mImgBtnLogin.setVisibility(Button.GONE);
                        mImgBtnLogout.setVisibility(Button.VISIBLE);
                        return;
                    case R.id.imgBtnUserInfo:
                        Toast.makeText(MainActivity.this, "单击了用户信息按钮!", Toast.
    LENGTH_LONG).show();
                        return;
                    case R.id.imgBtnLogout:         //用户登录后该按钮才会出现
                        Toast.makeText(MainActivity.this, "单击了注销按钮!", Toast.LENGTH_
    LONG).show();
                        //隐藏"注销"按钮,显示"登录"按钮
                        mImgBtnLogout.setVisibility(Button.GONE);
                        mImgBtnLogin.setVisibility(Button.VISIBLE);
                    case R.id.imgBtnMyOrder:
                        Toast.makeText(MainActivity.this, "单击了我的订单按钮!", Toast.
    LENGTH_LONG).show();
                        return;
                }
            }
        }
    }
```

3.4.3 用户注册界面设计

“用户注册”界面的布局效果如图 3.20 所示。

图 3.20 “用户注册”界面

(1) 在项目的 layout 文件夹中建立 activity_register.xml 布局文件,在布局文件中添加代码如下。

```
<?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 = "vertical" >
    < TextView
        android:id = "@ + id/textView1"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_gravity = "center_horizontal"
        android:layout_marginTop = "30dp"
        android:text = "用户注册"
        android:textSize = "25sp" />
    < LinearLayout
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:layout_margin = "10dp"
        android:orientation = "horizontal">
        < TextView
            android:id = "@ + id/textView2"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_marginLeft = "30dp"
            android:text = "用 户 名: " />
        < EditText
            android:id = "@ + id/etRegisterUserId"
```

```
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_marginRight = "30dp"
            android:ems = "30" >
            < requestFocus />
        </EditText >
    </LinearLayout >
    < LinearLayout
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:layout_margin = "10dp"
        android:orientation = "horizontal">
        < TextView
            android:id = "@ + id/textView3"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_marginLeft = "30dp"
            android:text = "用户密码: " />
        < EditText
            android:id = "@ + id/etRegisterUserPsword"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_marginRight = "30dp"
            android:ems = "30"
            android:inputType = "textPassword" >
            < requestFocus />
        </EditText >
    </LinearLayout >
    < LinearLayout
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:layout_margin = "10dp"
        android:orientation = "horizontal">
        < TextView
            android:id = "@ + id/textView4"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_marginLeft = "30dp"
            android:text = "确认密码: " />
        < EditText
            android:id = "@ + id/etRegisterUserAffirmPsword"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_marginRight = "30dp"
            android:ems = "30"
            android:inputType = "textPassword" >
            < requestFocus />
        </EditText >
    </LinearLayout >
    < LinearLayout
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:layout_margin = "10dp"
        android:orientation = "horizontal">
```

```
        <TextView
            android:id="@+id/textView5"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="30dp"
            android:text="电话号码:" />
        <EditText
            android:id="@+id/etRegisterUserMobilePhone"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginRight="30dp"
            android:ems="30"
            android:inputType="phone" >
            <requestFocus />
        </EditText>
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:orientation="horizontal">
        <TextView
            android:id="@+id/textView6"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="30dp"
            android:text="送餐地址:" />
        <EditText
            android:id="@+id/etRegisterUserAddress"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginRight="30dp"
            android:ems="30" >
            <requestFocus />
        </EditText>
    </LinearLayout>
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:layout_gravity="center"
        android:orientation="horizontal">
        <Button
            android:id="@+id/btnRegister"
            android:layout_width="100dp"
            android:layout_height="wrap_content"
            android:text="注 册" />
        <Button
            android:id="@+id/btnCancel"
            android:layout_width="100dp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="30dp"
            android:text="取 消" />
    </LinearLayout>
</LinearLayout>
```

(2) 右击项目 src 文件的 edu.cqut.MobileOrderFood 包,在弹出菜单中选择 New→Java Class,在弹出的对话框中输入文件名 RegisterActivity,然后,修改 RegisterActivity.java 文件的代码如下。

```
import android.os.Bundle;
import android.app.Activity;
import android.view.View;
import android.widget.*;

public class RegisterActivity extends Activity {
   public EditText metId, metPsword, metAffirmPsword, metPhone, metAddress;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_register);
      metId = (EditText)findViewById(R.id.etRegisterUserId);
      metPsword = (EditText)findViewById(R.id.etRegisterUserPsword);
      metAffirmPsword = (EditText)findViewById(R.id.etRegisterUserAffirmPsword);
      metPhone = (EditText)findViewById(R.id.etRegisterUserMobilePhone);
      metAddress = (EditText)findViewById(R.id.etRegisterUserAddress);
      Button btnOK = (Button)findViewById(R.id.btnRegister);
      Button btnCancel = (Button)findViewById(R.id.btnCancel);
      Button.OnClickListener mybtnListener = new Button.OnClickListener()
      {
         @Override
         public void onClick(View v) {
            switch (v.getId())
            {
            case R.id.btnCancel:
               finish();
               break;
            case R.id.btnRegister:
                Toast.makeText(RegisterActivity.this, "单击了注册按钮!", Toast.LENGTH_
LONG).show();
            }
         }
      };
      btnOK.setOnClickListener(mybtnListener);
      btnCancel.setOnClickListener(mybtnListener);
   }
}
```

(3) 在 AndroidManifest.xml 文件中注册新建的 RegisterActivity 页面,代码如下。

```
<application
    ...
    <activity
        android:name=".MainActivity"
        android:label="@string/app_name">
    ...
    </activity>
    <activity
        android:name=".RegisterActivity"
        android:label="@string/app_name">
    </activity>
</application>
```

将 AndroidManifest.xml 文件中原来位于 MainActivity 标签下的< intent-filter >标签移到< RegisterActivity >标签中，将启动页面改为 RegisterActivity，即可看到注册页面的运行效果，具体修改代码如下。

```
<application
    ...
    <activity android:name = ".MainActivity">
    </activity>
    <activity
        android:name = ".RegisterActivity"
        android:label = "@string/app_name" >
        <intent-filter>
            <action android:name = "android.intent.action.MAIN" />
            <category android:name = "android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
```

3.4.4 点餐菜单界面设计

点餐菜单界面采用自定义列表形式，图 3.21 是菜单界面，下面来实现该界面。

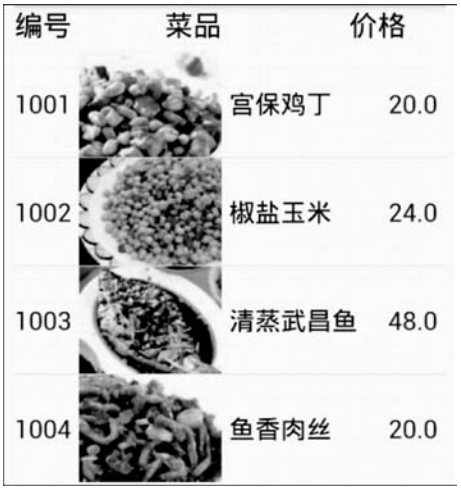

图 3.21 自定义列表运行效果

(1) 在项目的 res 目录下创建 raw 文件夹，在其中存放 4 张菜品图片。

(2) 在 layout 文件夹中创建名为 listitem.xml 的布局文件，代码如下。

```
<?xml version = "1.0" encoding = "utf-8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "horizontal"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:id = "@ + id/listitem">
    <TextView android:id = "@ + id/dishid"
```

```
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:textIsSelectable = "false"
            android:layout_gravity = "center"
            android:textSize = "18sp" />"
        < ImageView
            android:id = "@ + id/img"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_gravity = "center"
            android:layout_marginLeft = "5dp"/>
        < TextView android:id = "@ + id/title"
            android:layout_width = "100dp"
            android:layout_height = "wrap_content"
            android:textColor = "#000000"
            android:textIsSelectable = "false"
            android:gravity = "left"
            android:layout_gravity = "center"
            android:layout_marginLeft = "5dp"
            android:textSize = "18sp" />
        < TextView
            android:id = "@ + id/price"
            android:layout_width = "0dip"
            android:layout_height = "wrap_content"
            android:textIsSelectable = "false"
            android:layout_gravity = "center"
            android:layout_marginLeft = "5dp"
            android:textSize = "18sp"
            android:layout_weight = "1"
            android:gravity = "center"/>
    </LinearLayout >
```

该布局文件采用水平线性布局,用于定义 ListView 中各列属性,依次为菜品 ID、菜品图片、菜品名称和单价。

(3) 在 layout 文件夹中创建名为 activity_caipin_list.xml 的布局文件,添加代码如下。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:id = "@ + id/caipinlistlayout"
    android:orientation = "vertical" >
    < TableRow
        android:id = "@ + id/DishHead"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content">
        < TextView
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:textColor = "#000000"
            android:textSize = "20sp"
```

```
            android:gravity = "center"
            android:text = "编号"/>
        < TextView
            android:layout_height = "wrap_content"
            android:layout_width = "wrap_content"
            android:textColor = "＃000000"
            android:textSize = "20sp"
            android:gravity = "center"
            android:layout_marginBottom = "4dp"
            android:layout_marginLeft = "10dp"
            android:layout_weight = "2"
            android:text = "菜品"/>
        < TextView
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:textColor = "＃000000"
            android:textSize = "20sp"
            android:text = "价格"
            android:layout_marginLeft = "5dp"
            android:gravity = "center"
            android:layout_weight = "1"/>
    </TableRow >
    < ListView
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:id = "@ + id/ListViewCainpin" />
</LinearLayout >
```

该布局文件显示菜单页面，第一行为 TableRow 布局，用于菜单标题，该布局含三个 TextView 控件，依次显示编号、菜品、价格。TableRow 布局的下一行为 ListView 控件，该控件显示各个具体的菜品，而这个 ListView 控件的布局将使用 listitem. xml 文件。

(4) 在 src 文件夹中添加 CaipinActivity. java 文件，在 CaipinActivity 类中创建适配器和数据源。

```
static List < Map < String, Object >> mfoodinfo;  //菜品数据源列表，由 HashMap 表构成
public ListView mlistview;
static SimpleAdapter mlistItemAdapter;
```

(5) 在 onCreate()函数中添加菜单的代码如下。

```
@Override
protected void onCreate(Bundle savedInstanceState)
{
    super. onCreate(savedInstanceState);
    setContentView(R. layout. activity_caipin_list);
    mlistview = (ListView) findViewById(R. id. ListViewCainpin);
    mfoodinfo = getFoodData();
    //构造 SimpleAdapter 适配器，将它和自定义的布局文件、List 数据源关联
    mlistItemAdapter = new SimpleAdapter(this, mfoodinfo,     //数据源
                R. layout. listitem,                          //ListItem 的 XML 实现
```

```
                //动态数组与 ImageItem 对应的子项
                new String[] {"dishid", "image","title", "price", "order"},
                //ImageItem 的 XML 文件里面的 1 个 ImageView,3 个 TextView ID
                new int[] {R.id.dishid, R.id.img, R.id.title, R.id.price});
    mlistItemAdapter.notifyDataSetChanged();
    mlistview.setAdapter(mlistItemAdapter);
    //设置 ListView 选项单击监听器
    this.mlistview.setOnItemClickListener(new OnItemClickListener(){
        @Override
        public void onItemClick(AdapterView<?> arg0,  //选项所属的 ListView
                                View arg1,          //被选中的控件,即 ListView 中被选中的子项
                                int arg2,           //被选中子项在 ListView 中的位置
                                long arg3)          //被选中子项的行号
        {
            ListView templist = (ListView)arg0;
            View mView = templist.getChildAt(arg2);
            final TextView tvTitle = (TextView)mView.findViewById(R.id.title);
            Toast.makeText(MainActivity.this, tvTitle.getText().toString(), Toast.LENGTH_
LONG).show();
        }
    });
}
```

SimpleAdapter 适配器的数据源是 HashMap 列表的数据结构,函数 getFoodData()负责将 ArrayList<Dish>的数据结构转换成适用于 SimpleAdapter 的 List<Map<String, Object>>数据结构。

```
private ArrayList<Map<String, Object>> getFoodData()
{
    ArrayList<Map<String, Object>> fooddata = new ArrayList<Map<String,Object>>();
    //将菜品信息填充进 foodinfo 列表
    int s = mDishes.size();                 //得到菜品数量
    for (int i = 0; i < s; i++) {
        Dish theDish = mDishes.get(i);      //得到当前菜品
        Map<String, Object> map = new HashMap<String, Object>();
        map.put("dishid", theDish.mId);
        map.put("image", theDish.mImage);
        map.put("title", theDish.mName);
        map.put("price", theDish.mPrice);
        fooddata.add(map);
    }
    return fooddata;
}
```

(6) 在 AndroidManifest.xml 文件中注册 CaipinActivity 页面,代码如下。

```
<application
    ...
    <activity
        android:name=".RegisterActivity"
        android:label="@string/app_name">
```

```
    </activity>
    <activity
        android:name=".CaipinActivity"
        android:label="@string/app_name">
    </activity>
</application>
```

为了测试点餐菜单界面，我们用 Intent 将 MainActivity 页面切换到 CaipinActivity 页面，在 MainActivity.java 文件中添加以下粗体代码实现两者的切换，代码具体含义我们将在第 4 章中介绍。

```
import android.content.Intent;
…
public class MainActivity extends AppCompatActivity {
    …
    public class myImageButtonListener implements View.OnClickListener
    {
        @Override
        public void onClick(View v) {
            switch (v.getId())
            {
                case R.id.imgBtnRest:
                     Toast.makeText(MainActivity.this, "单击了点餐按钮!", Toast.LENGTH_
LONG).show();
                    Intent intent = new Intent(MainActivity.this, CaipinActivity.class);
                    startActivity(intent);
                    return;
                }
            …
        }
    }
}
```