

目 录

Android 基础篇

第 1 章 初识 Android	2
1.1 什么是 Android	2
1.1.1 Android 的发展	2
1.1.2 Android 的架构及特点	2
1.2 Android 开发平台的搭建	3
1.2.1 下载 JDK	3
1.2.2 安装 JDK	4
1.2.3 配置 Java 环境变量	4
1.2.4 安装 Eclipse	5
1.2.5 安装与配置 Android SDK	6
1.2.6 安装 Android ADT	6
1.2.7 虚拟设备的创建与模拟器的运行	8
1.3 HelloAndroid——我的第一个 Android 程序	9
1.3.1 创建第一个 Android 程序——HelloAndroid	9
1.3.2 基本文件及 Android 框架	11
1.4 小结	11
1.5 习题	12
第 2 章 Android 程序界面布局设计	16
2.1 布局概述	16
2.1.1 什么是布局	16
2.1.2 布局的类型	17
2.1.3 布局文件的常用概念	18
2.2 相对布局	19
2.2.1 相对容器布局	19
2.2.2 相对控件布局	21
2.3 线性布局	23
2.3.1 什么是线性布局	23

2.3.2	线性布局的语法	24
2.3.3	创建线性布局	24
2.4	表格布局	26
2.4.1	什么是表格布局	26
2.4.2	表格布局的语法	26
2.4.3	创建表格布局	27
2.5	帧布局	28
2.5.1	什么是帧布局	28
2.5.2	帧布局的语法	28
2.5.3	创建帧布局	29
2.6	网格布局和布局控件	30
2.6.1	什么是网格布局	30
2.6.2	网格布局的语法	30
2.6.3	创建网格布局	31
2.6.4	什么是布局控件	32
2.7	小结	34
2.8	习题	34
第3章	基本控件	40
3.1	控件概述	40
3.1.1	控件的构成	40
3.1.2	属性的使用	40
3.1.3	方法和事件的使用	41
3.2	文本类控件	42
3.2.1	文本框	42
3.2.2	编辑框	44
3.3	按钮类控件	45
3.3.1	按钮	45
3.3.2	图片按钮	47
3.3.3	开关按钮	48
3.3.4	单选按钮	49
3.3.5	复选按钮	50
3.4	图片控件	52
3.5	动画播放技术	53
3.5.1	补间动画	54
3.5.2	帧动画	56
3.6	时钟控件	58
3.7	日期与时间控件	60
3.7.1	日期选择控件	60

3.7.2 时间选择控件·····	60
3.8 小结·····	61
3.9 习题·····	62
第4章 高级控件·····	71
4.1 自动完成文本类控件·····	71
4.1.1 自动完成文本控件·····	71
4.1.2 多文本自动完成输入控件·····	73
4.2 进度条与拖动条·····	75
4.2.1 进度条·····	75
4.2.2 拖动条·····	76
4.3 评分条·····	78
4.4 滚动视图·····	80
4.5 列表视图·····	81
4.6 下拉列表·····	83
4.7 选项卡·····	85
4.8 页面滑动切换控件·····	87
4.9 图片切换控件·····	89
4.10 网格视图·····	92
4.11 小结·····	94
4.12 习题·····	94

Android 技术篇

第5章 消息提示·····	110
5.1 菜单·····	110
5.1.1 选项菜单和子菜单·····	110
5.1.2 上下文菜单·····	113
5.2 对话框·····	115
5.2.1 普通对话框·····	115
5.2.2 提示对话框·····	116
5.2.3 进度对话框·····	119
5.2.4 日期选择对话框·····	120
5.2.5 时间选择对话框·····	121
5.3 消息提示框·····	122
5.4 通知提示框·····	123
5.5 小结·····	125
5.6 习题·····	125

第 6 章 深入解析 Activity	131
6.1 从一个单界面程序看 Activity	131
6.1.1 启动单界面程序	131
6.1.2 了解 Activity 的状态变化	132
6.1.3 结束 Activity	133
6.2 在两个 Activity 之间跳转	135
6.2.1 启动第一个 Activity——主 Activity	135
6.2.2 创建第二个 Activity	136
6.2.3 启动第二个 Activity	138
6.2.4 跳转回主 Activity	139
6.2.5 “BACK”到第二个 Activity	140
6.3 在两个 Activity 之间传递数据	142
6.3.1 传递数据到目标 Activity	143
6.3.2 返回数据到主 Activity	145
6.4 Intent 和 IntentFilter	148
6.4.1 Intent	148
6.4.2 IntentFilter	154
6.5 小结	155
6.6 习题	156
第 7 章 服务与消息广播	158
7.1 Service 简介	158
7.1.1 开发 Service	158
7.1.2 Service 的生命周期	159
7.2 操作 Service	160
7.2.1 调用 context.startService() 方法启动 Service	160
7.2.2 调用 context.bindService() 方法启动 Service	163
7.3 Service 通信	165
7.3.1 本地服务通信	165
7.3.2 远程服务通信	168
7.4 系统提供的 Service	171
7.4.1 电话管理器	171
7.4.2 短信管理器	175
7.4.3 音频管理器	177
7.4.4 振动器	180
7.5 广播接收者	182
7.5.1 开发广播接收者组件	182
7.5.2 接收系统广播	186
7.6 小结	188

7.7 习题	188
第 8 章 数据存储	192
8.1 SharedPreferences 轻量级存储	192
8.1.1 SharedPreferences 和 Editor	192
8.1.2 使用 SharedPreferences 存储数据	193
8.2 文件存储	195
8.2.1 读写文件中的数据	195
8.2.2 读写 SD 卡中的数据	197
8.3 SQLite 数据库存储	201
8.3.1 SQLiteDatabase 和 SQLiteOpenHelper	201
8.3.2 数据库的基本操作	205
8.4 内容提供者	209
8.4.1 ContentProvider 简介	209
8.4.2 自定义 ContentProvider	212
8.5 小结	216
8.6 习题	216

Android 应用篇

第 9 章 网络应用	220
9.1 Socket 通信	220
9.1.1 Socket 的工作机制	220
9.1.2 Socket 服务器端开发	221
9.1.3 Socket 客户端开发	222
9.1.4 运行程序	224
9.2 HTTP 通信	224
9.2.1 HTTP 通信方式	224
9.2.2 使用 HttpURLConnection 接口进行开发	225
9.2.3 使用 HttpClient 接口进行开发	226
9.3 URL 通信	232
9.3.1 URL 简介	232
9.3.2 URL 通信开发	233
9.4 WebView	235
9.4.1 WebView 简介	236
9.4.2 WebView 开发	236
9.5 小结	239
9.6 习题	239

第 10 章 地理位置应用	241
10.1 GPS 定位服务	241
10.1.1 GPS 的相关类	241
10.1.2 获取 GPS 信息	242
10.2 Google Maps	245
10.2.1 获取 Map API Key	245
10.2.2 创建模拟器	247
10.2.3 Google Maps 的相关类	248
10.2.4 Google Maps 地图查询应用	249
10.3 Google StreetView	253
10.3.1 Google StreetView 服务的原理	253
10.3.2 Google StreetView 程序的开发	253
10.4 小结	255
10.5 习题	255
第 11 章 音频和视频应用	258
11.1 音频应用	258
11.1.1 MediaPlayer 类简介	258
11.1.2 使用 MediaPlayer 播放本地音频文件	259
11.1.3 使用 MediaPlayer 播放标准音频文件	263
11.2 视频应用	265
11.2.1 视频相关类简介	265
11.2.2 使用 VideoView 播放视频文件	266
11.3 音频和视频的录制	267
11.3.1 录制音频	267
11.3.2 录制视频	271
11.4 小结	274
11.5 习题	274
第 12 章 传感器应用	277
12.1 传感器开发	277
12.1.1 系统传感器	277
12.1.2 传感器的开发过程	277
12.1.3 真机测试	278
12.2 常用传感器	280
12.2.1 加速度传感器	280
12.2.2 磁场传感器	282
12.2.3 方向传感器	283
12.2.4 重力传感器	285

12.2.5 亮度传感器	287
12.3 小结	288
12.4 习题	288
第 13 章 手势应用	292
13.1 输入法手势识别	292
13.1.1 Gesture 相关类简介	292
13.1.2 输入法手势程序开发	292
13.2 触摸屏手势识别	295
13.2.1 GestureDetector 简介	295
13.2.2 触摸屏手势程序的开发	296
13.3 小结	298
13.4 习题	299
第 14 章 图形应用	302
14.1 位图	302
14.1.1 AssetManager 类	302
14.1.2 Bitmap 对象和 BitmapFactory 类	302
14.1.3 对 assets 文件夹的访问	303
14.2 Canvas 绘图	305
14.2.1 Canvas 类和 Paint 类	305
14.2.2 绘制基本图形	306
14.3 小结	307
14.4 习题	307

Android 开发篇

第 15 章 Android 程序开发——音乐播放器	312
15.1 程序简介	312
15.1.1 功能概述	312
15.1.2 开发环境及目标平台	312
15.2 程序架构	313
15.3 登录界面的设计与实现	313
15.4 主界面的设计与实现	314
15.4.1 主界面布局	314
15.4.2 音乐播放列表	315
15.4.3 音乐播放控制	316
15.4.4 音乐音量调节	320
15.4.5 音乐播放进度控制	321

15.4.6	主界面菜单.....	322
15.5	录音界面的设计与实现.....	326
15.5.1	录音界面.....	326
15.5.2	实现录音功能.....	326
15.5.3	播放录音文件.....	328
15.6	设置界面的设计与实现.....	328
15.6.1	设置来电铃声.....	328
15.6.2	设置闹铃铃声.....	329
15.6.3	设置通知铃声.....	330
15.7	小结.....	331

Android 基础篇

- 第 1 章 初识 Android
- 第 2 章 Android 程序界面布局设计
- 第 3 章 基本控件
- 第 4 章 高级控件

第 1 章 初识 Android

Android 是 Google 于 2007 年 11 月 5 日发布的基于 Linux 内核的移动开发平台，主要用于便携设备。该平台由操作系统、中间件、用户界面和应用软件组成，是一个真正开放的移动开发平台。

本章首先介绍 Android 的发展、特点、应用程序框架以及开发环境的搭建，使读者对 Android 平台有一定的了解；然后将开发第一个 Android 程序——HelloAndroid，并通过对该程序进行简单的分析，带领读者步入 Android 开发的大门。

1.1 什么是 Android

Android 操作系统最初主要支持手机，现在它的应用逐渐扩展到平板电脑及其他领域，主要竞争对手是苹果的 iOS 以及 RIM 的 Blackberry OS。那么，Android 是如何发展的，又有哪些优点使它成长得如此迅速呢？

1.1.1 Android 的发展

Andy Rubin 创建了 Android 公司，并开发了 Android 平台。2005 年，Google 收购了成立仅 22 个月的高科技企业 Android，其以后的发展如图 1.1 所示。

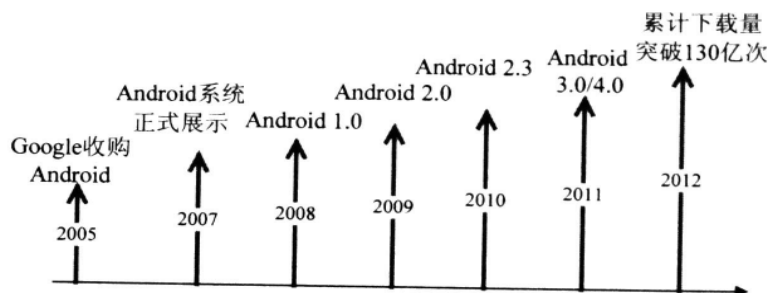


图 1.1 Android 的发展

目前，Android 系统已经升级到 Android 4.2，研发代号与 Android 4.1 相同，仍是 Jelly Bean（果冻豆），而 Android 4.1 则第一次与一个 Platform Developer Kit（平台开发工具包）捆绑。

1.1.2 Android 的架构及特点

Android 系统采用分层架构，从高层到低层分为 4 层，分别是应用程序层、应用程序框架层、系统运行库层和 Linux 核心层，如图 1.2 所示。与其他手机操作系统相比，Android 的优点



如图 1.3 所示。

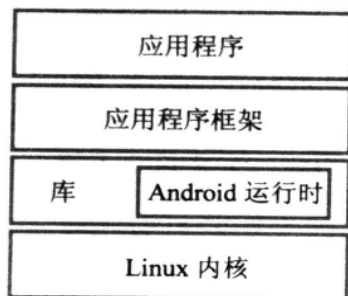


图 1.2 Android 系统架构

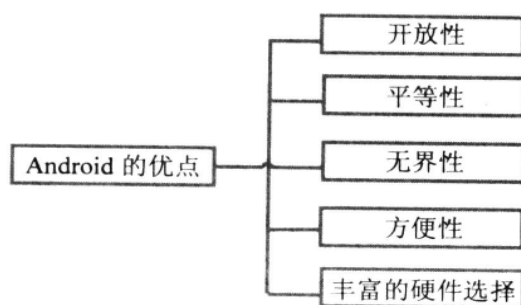


图 1.3 Android 的优点



1.2 Android 开发平台的搭建

Android 开发环境在 Eclipse 中构建, 需要安装开发工具 JDK、SDK 和 ADT, 并配置 JDK 和 SDK 的环境变量。下面将依次讲解它们的下载与安装方法。

1.2.1 下载 JDK

在浏览器的地址栏中输入 Oracle 官方网站地址“<http://www.oracle.com>”, 在网站主页找到相关链接, 下载 JDK 安装包, 操作过程如图 1.4 所示。

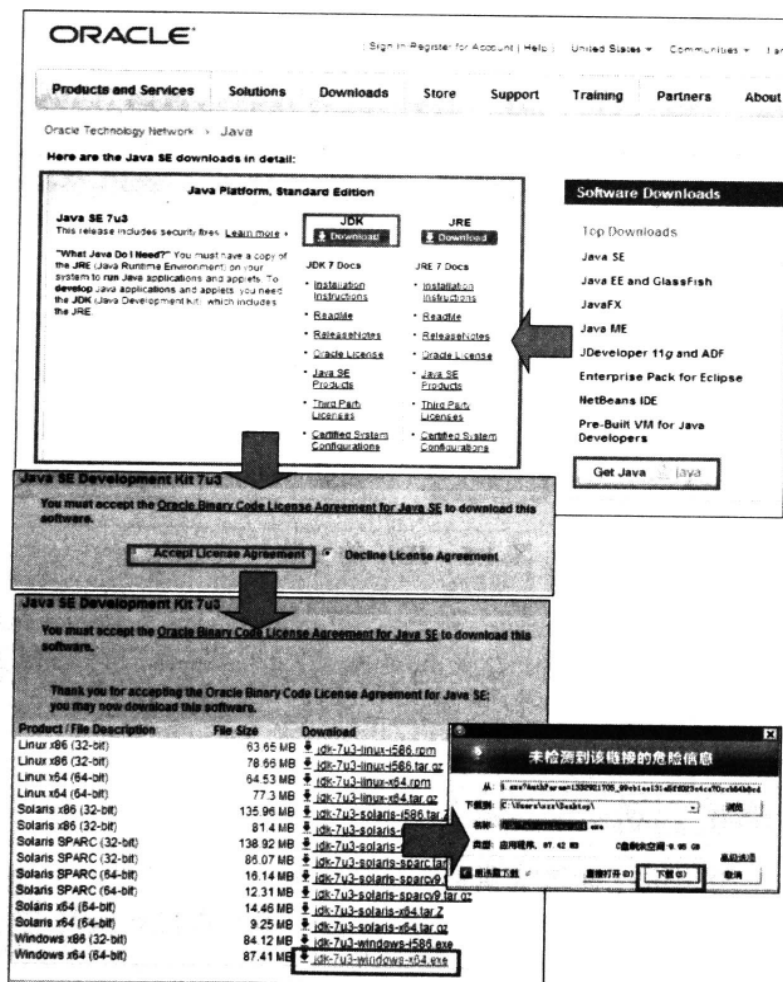


图 1.4 JDK 安装包的下载



注意：在 Java SE Development Kit 7u3 表格中有适用于各操作系统的 JDK 版本，请读者根据自己使用的操作系统进行选择。其中适合 Windows 操作系统的有 Windows x86 (32bit) 和 Windows x64 (64bit) 两种：如果操作系统是 32 位的，请选择第一种；如果操作系统是 64 位的，请选择第二种。笔者使用的是 32 位 Windows 7 操作系统，所以下载 jdk-7u3-windows-x64.exe。

1.2.2 安装 JDK

JDK 安装包下载完成后，即可双击安装程序图标，使用安装向导进行安装，操作过程如图 1.5 所示。

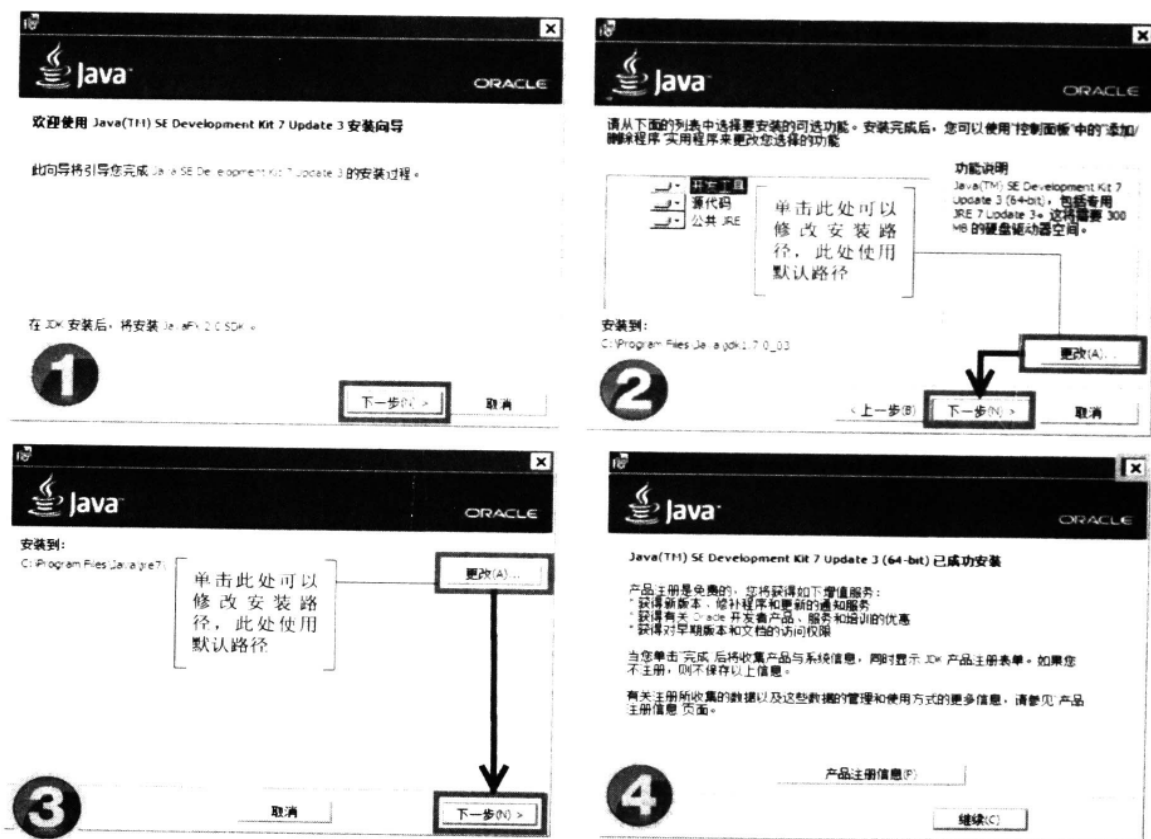


图 1.5 JDK 的安装

注意：在图 1.5 的 4 号对话框中，如果单击“继续”按钮，将进行 JavaFX SDK 的安装，而这一步在此是不需要的，所以单击右上角的“关闭”按钮即可。

1.2.3 配置 Java 环境变量

JDK 安装完成后，需要进行环境变量的配置。这是因为程序执行时不知道某些 Java 组件安装在哪里，而如果在环境变量里设置了，程序就会到那里查找要执行的方法的路径。JDK 环境变量的配置过程如图 1.6 所示。

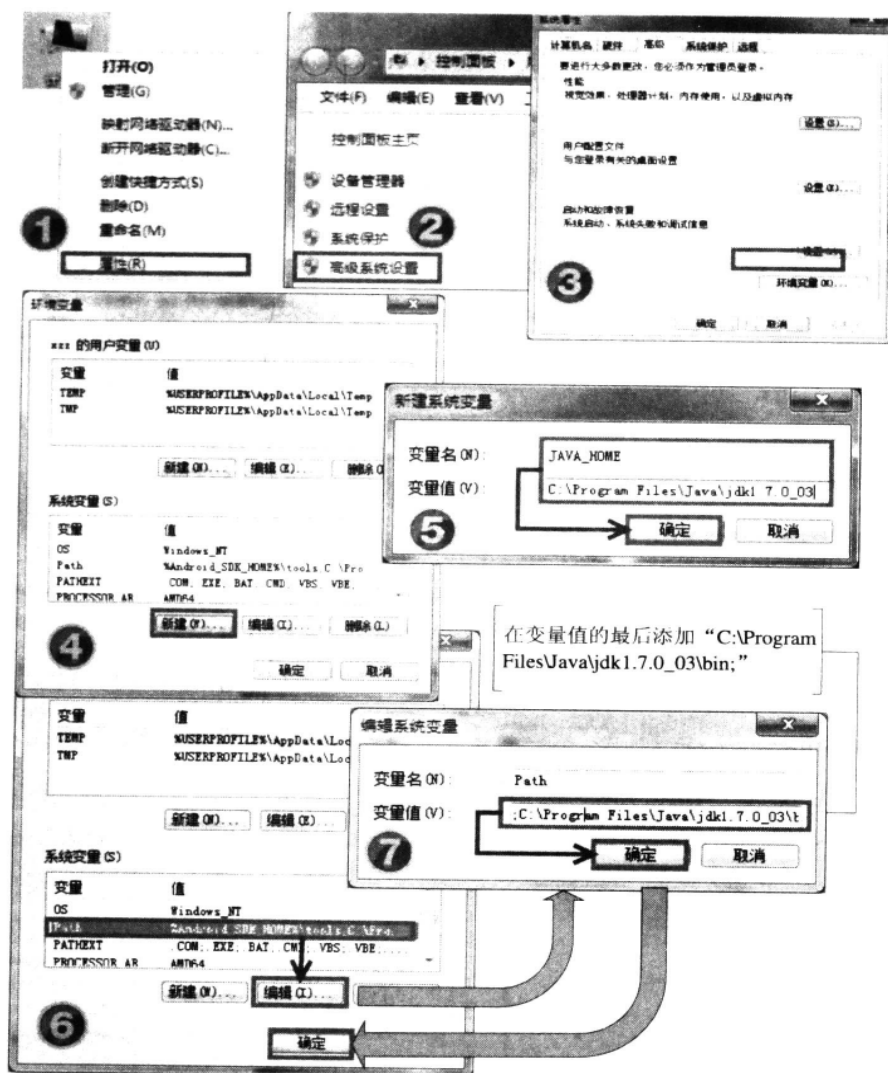


图 1.6 JDK 环境变量的配置

1.2.4 安装 Eclipse

在浏览器的地址栏中输入“<http://www.eclipse.org/downloads/>”，进入 Eclipse 安装包的下载页面，操作过程如图 1.7 所示。

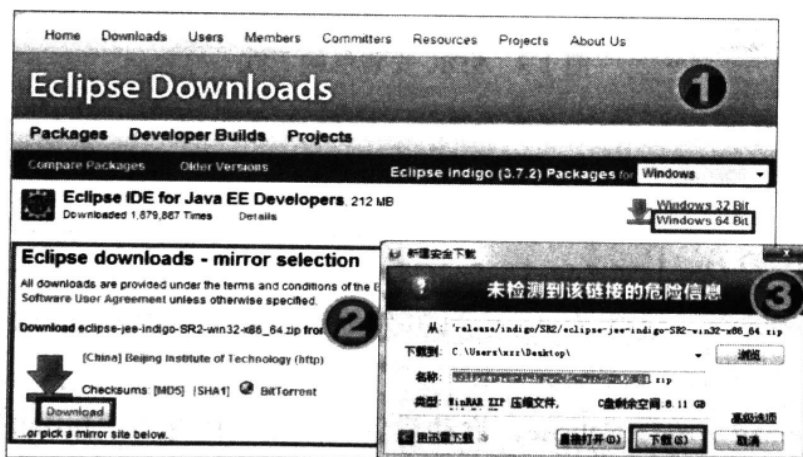


图 1.7 Eclipse 安装包的下载



将下载的 Eclipse 安装包解压到 C 盘的 Android 文件夹中。

1.2.5 安装与配置 Android SDK

SDK 为开发者提供了库文件以及其他开发工具，它们是在整体开发中使用的工具包。SDK 的安装与配置步骤如下。

(1) 在浏览器的地址栏中输入“<http://developer.android.com>”，进入“Android SDK | Android developers”页面，下载 SDK 安装包，操作过程如图 1.8 所示。

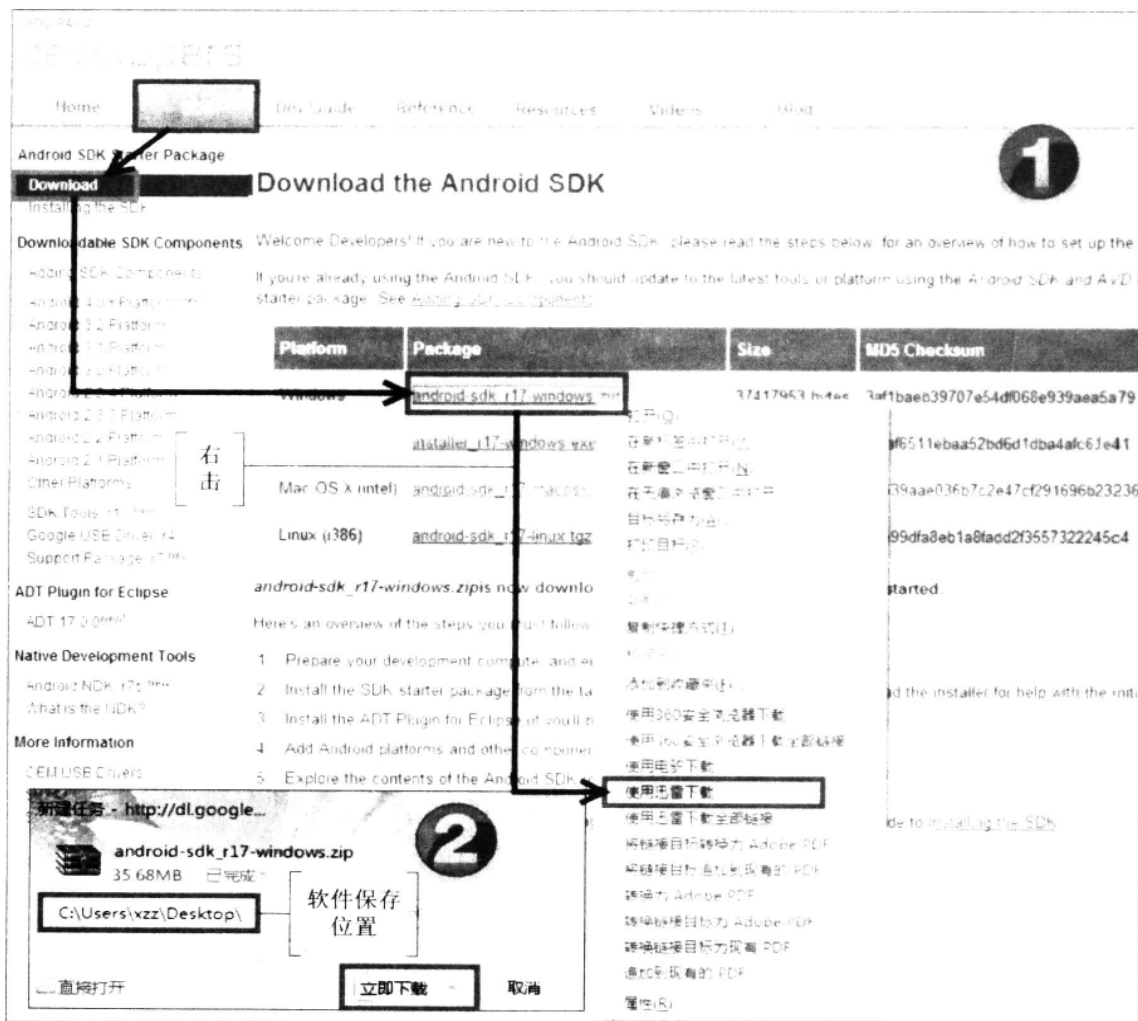


图 1.8 SDK 安装包的下载

(2) 将下载的 SDK 安装包解压到 C 盘的 Android 文件夹中。

(3) 配置 SDK 环境变量，过程如图 1.9 所示。

1.2.6 安装 Android ADT

在 Eclipse 编译环境中，ADT 为 Android 开发提供开发工具的升级和更新，其安装与配置步骤如下。

(1) 双击 eclipse.exe，运行 Eclipse。启动后依次单击“Help”、“Install”、“New Software”选项，打开“Install”对话框。ADT 的安装过程如图 1.10 所示。

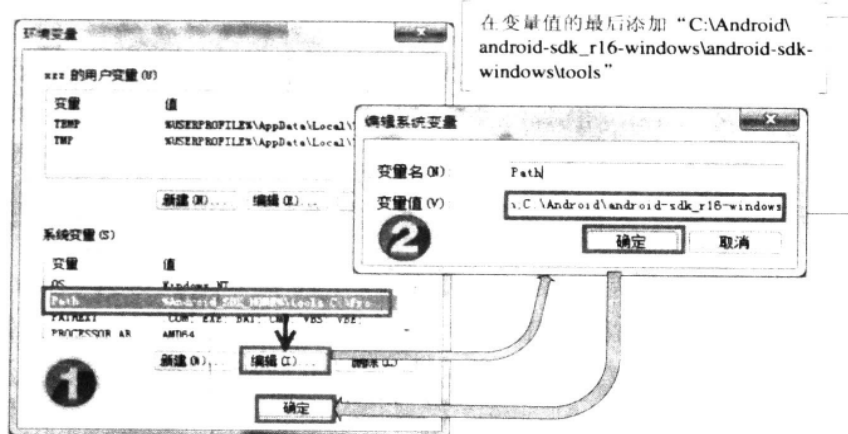


图 1.9 SDK 环境变量的配置

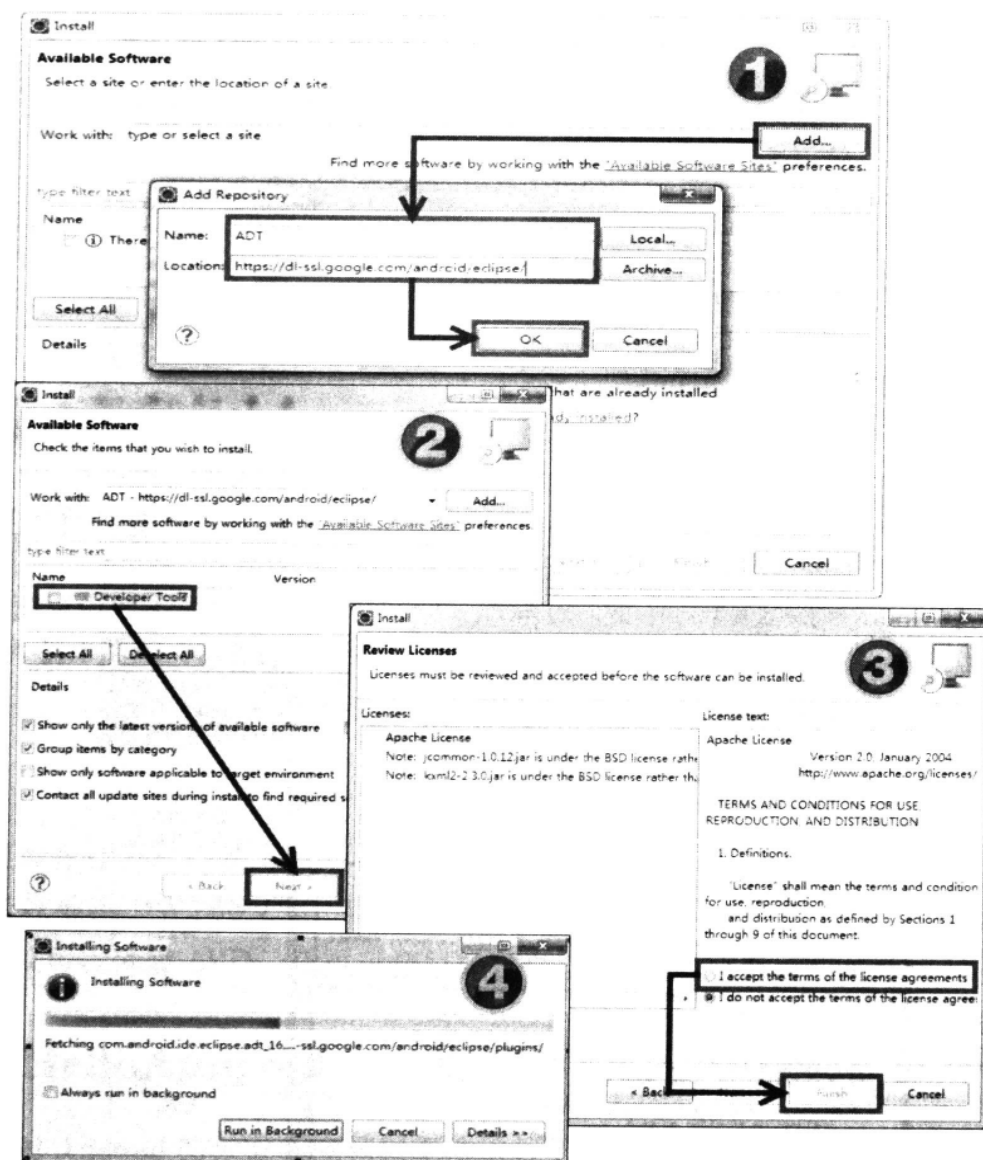


图 1.10 ADT 的安装

在图 1.10 的 1 号对话框中，在 “Name” 文本框中可以输入任意的名字，在 “Location” 文本框中应输入 ADT 的下载地址 “https://dl-ssl.google.com/android/eclipse/”。



提示: 如果输入的地址不能正常下载 ADT, 读者可以尝试将“https://dl-ssl.google.com/android/eclipse/”改为“http://dl-ssl.google.com/android/eclipse/”。如果出现下载的 ADT 与已经安装的 SDK 的版本不兼容的情况, 读者可以手动下载 ADT 安装包, 然后单击“Archive...”按钮, 选择手动下载的 ADT 安装包。

(2) ADT 安装完成后会重启 Eclipse。重启 Eclipse 后, 依次选择“Window”、“Preferences”选项, 打开“Preferences”对话框, 指定 SDK 的位置, 操作过程如图 1.11 所示。

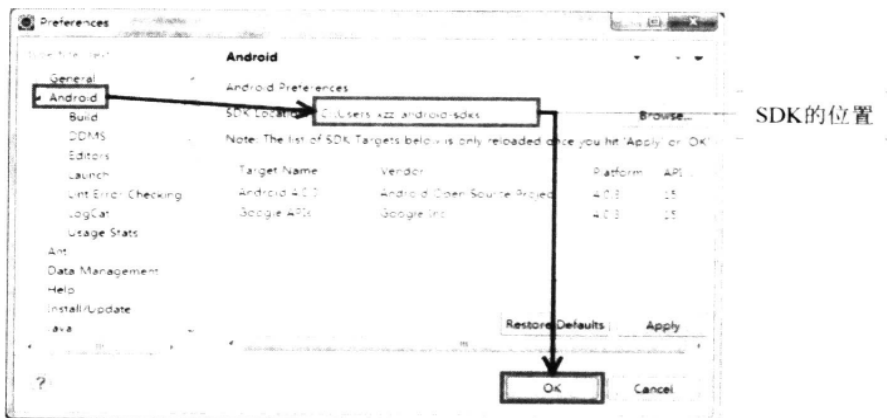


图 1.11 指定 SDK 的位置

1.2.7 虚拟设备的创建与模拟器的运行

完成 Eclipse 集成环境的搭建工作后, 在运行模拟器之前, 还需要创建 AVD (Android Virtual Device, Android 虚拟设备)。AVD 实际上是用来描述模拟器属性的工具。有了 AVD, 开发人员可以对自己的模拟器进行必要的设置, 如设置屏幕尺寸、内存等。本节将运用之前搭建的 Eclipse 环境创建虚拟设备并运行模拟器, 具体步骤如下。

(1) 按下【Windows】+【R】组合键, 打开“运行”对话框, 在“打开”设置框中输入“cmd”命令, 单击“确定”按钮, 打开命令行窗口。在命令行窗口输入相关命令, 如图 1.12 和图 1.13 所示。

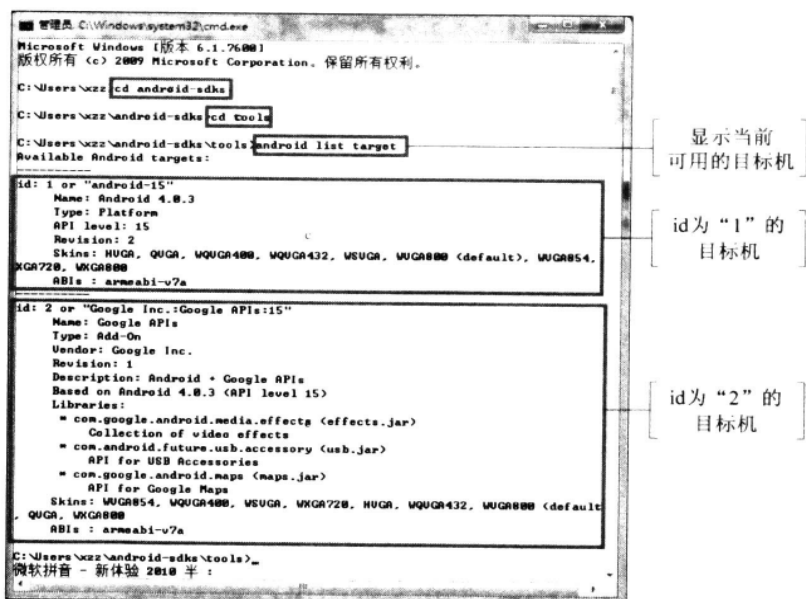


图 1.12 显示当前可用的目标机

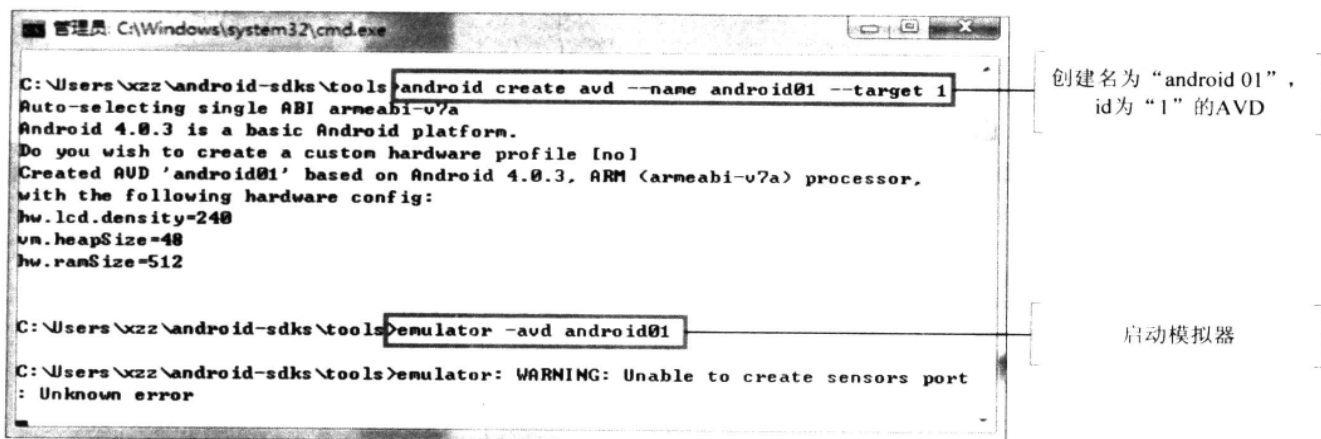


图 1.13 创建并启动虚拟设备 android01

(2) 如果安装和配置正确，将出现如图 1.14 所示的模拟器界面。

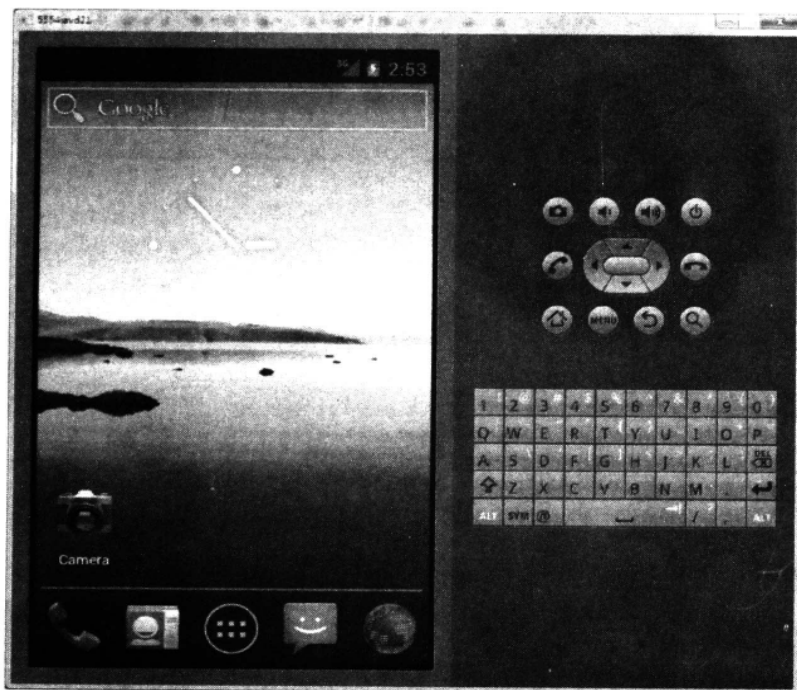


图 1.14 android01 模拟器界面

除了使用命令行方式创建 AVD，还可以使用 Eclipse 自带的 AVD Manager 创建 ADV。在 Eclipse 界面依次单击“Window”、“AVD Manager”选项，可以打开 AVD 管理界面。在 AVD 管理界面中不仅可以创建 AVD，还可以对已经存在的 AVD 进行管理，如图 1.15 所示。



1.3 HelloAndroid——我的第一个 Android 程序

前面已经对 Android 的开发环境和模拟器进行了配置，本节将构建第一个 Android 应用程序——HelloAndroid，并对该程序进行简单的讲解。

1.3.1 创建第一个 Android 程序——HelloAndroid

下面介绍创建和运行第一个 Android 应用程序——HelloAndroid 的基本操作。

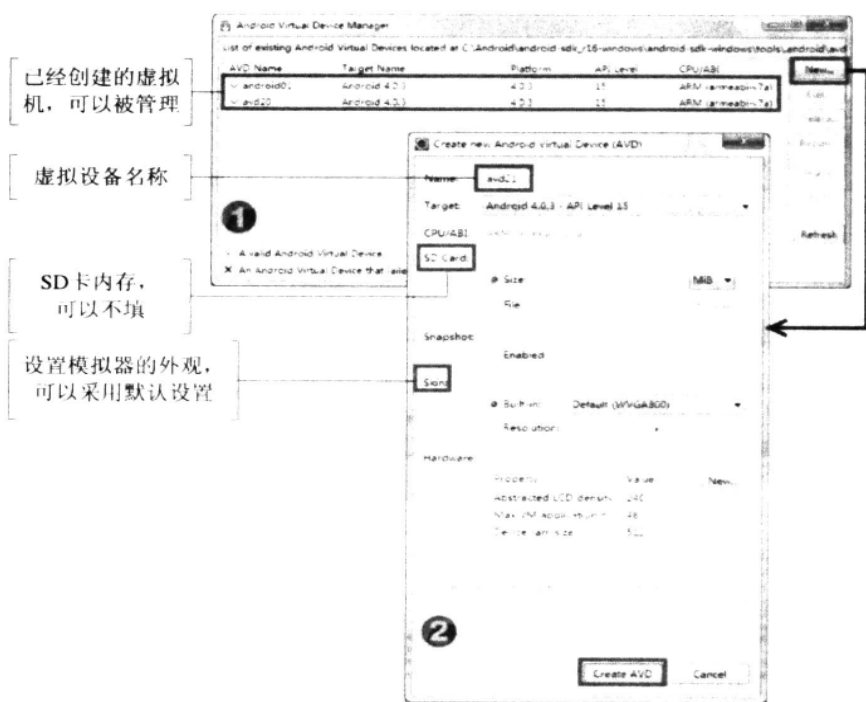


图 1.15 AVD 的创建与管理

1. 创建第一个 Android 应用程序

启动 Eclipse，依次单击“File”、“New”、“Other”选项，打开“New”对话框。创建一个 Android 应用程序的过程如图 1.16 所示。

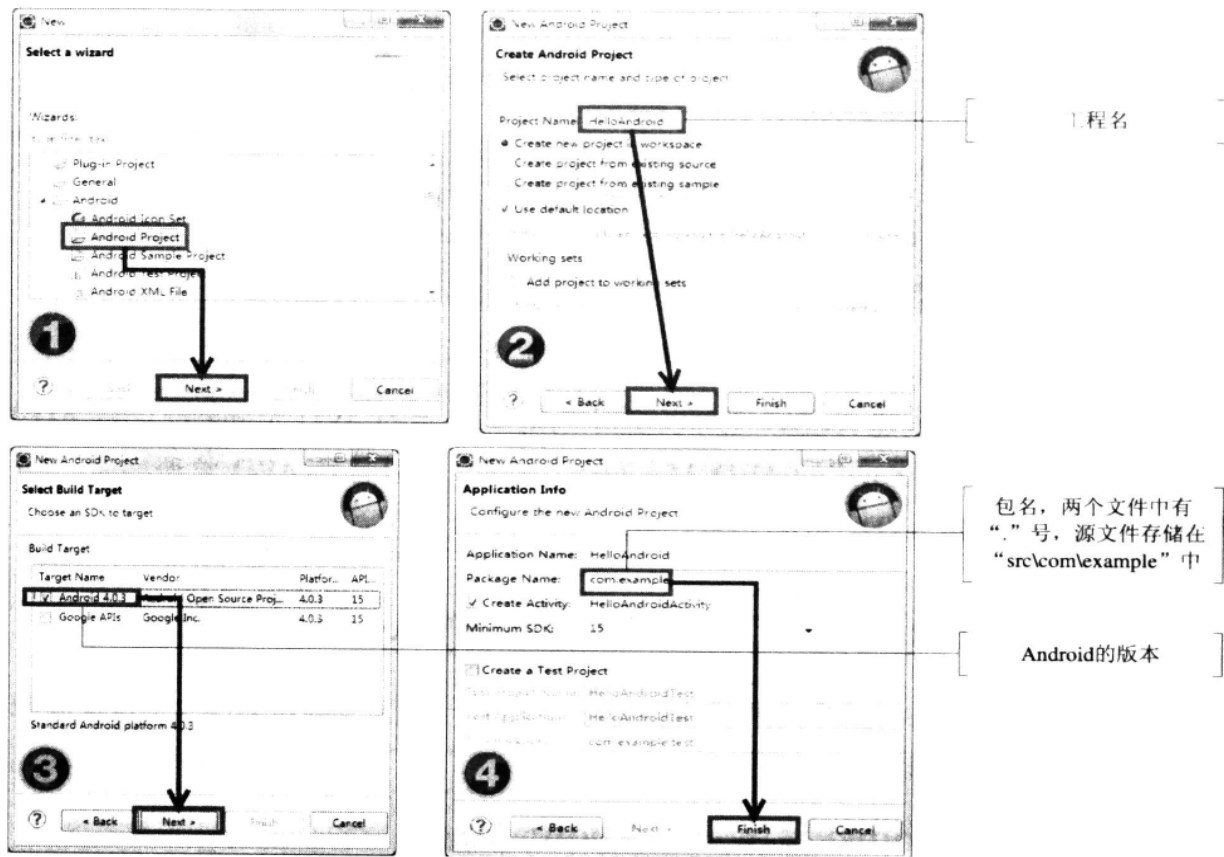


图 1.16 创建第一个 Android 程序



2. 运行创建的 Android 程序

程序创建后,可在“Package Explorer”窗口看到项目的目录结构。右击项目名,在弹出的快捷菜单中依次单击“Run As”、“Android Application”选项,启动 Android 模拟器,其运行效果如图 1.17 所示。

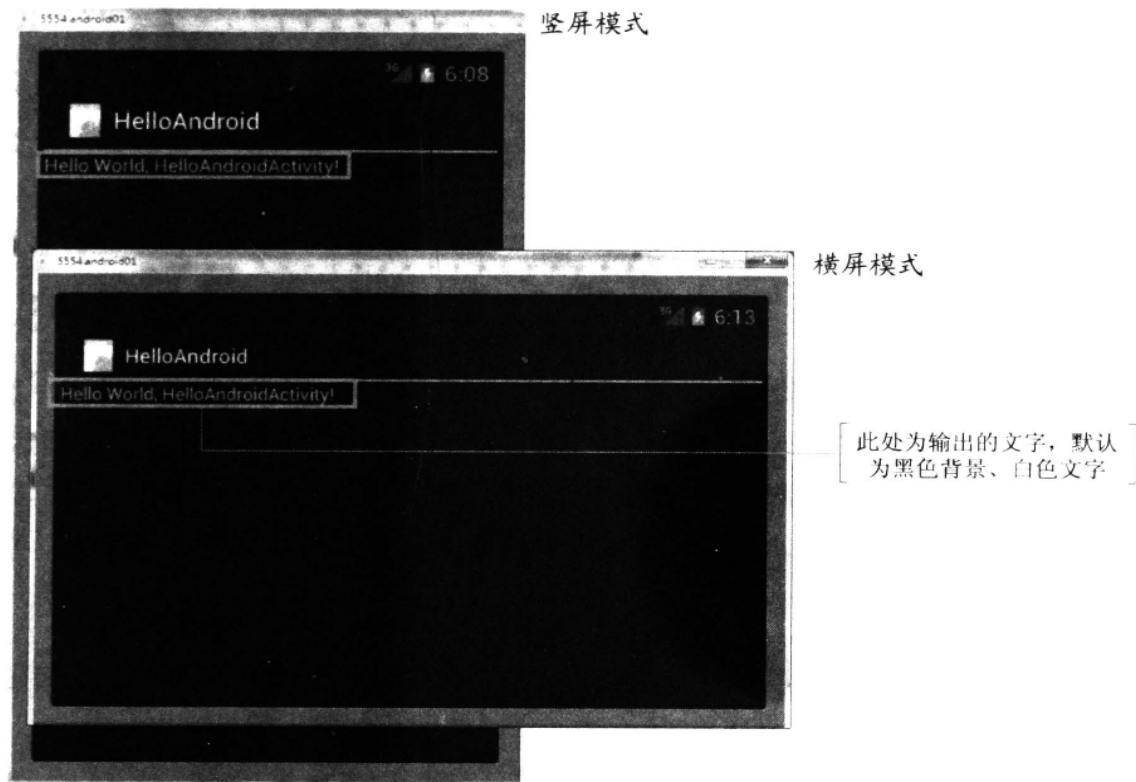


图 1.17 HelloAndroid 程序的运行效果

提示: 因为很多程序和游戏是横屏模式的,所以在程序调试过程中可能需要将模拟器切换成横屏模式。读者可以使用【Ctrl】+【F12】组合键来切换模拟器的横/竖屏模式。

1.3.2 基本文件及 Android 框架

虽然已经在第 1.3.1 节中创建并运行了一个简单的 Android 程序,但读者可能还是不了解 Android。本节将通过 HelloAndroid 程序在 Eclipse 环境中的目录文件与磁盘文件的对应关系来说明各个文件的作用,如图 1.18 所示。

1.4 小结

本章介绍了 Android 系统的发展过程与主要特点,详细讲解了 Android 开发平台的搭建步骤,开发了第一个 Android 程序——HelloAndroid,并简单说明了其中各个目录和文件的作用。通过本章的学习,读者应重点掌握环境变量的配置及 AVD 的创建,并对 Android 平台应用程序的开发步骤有初步的了解。

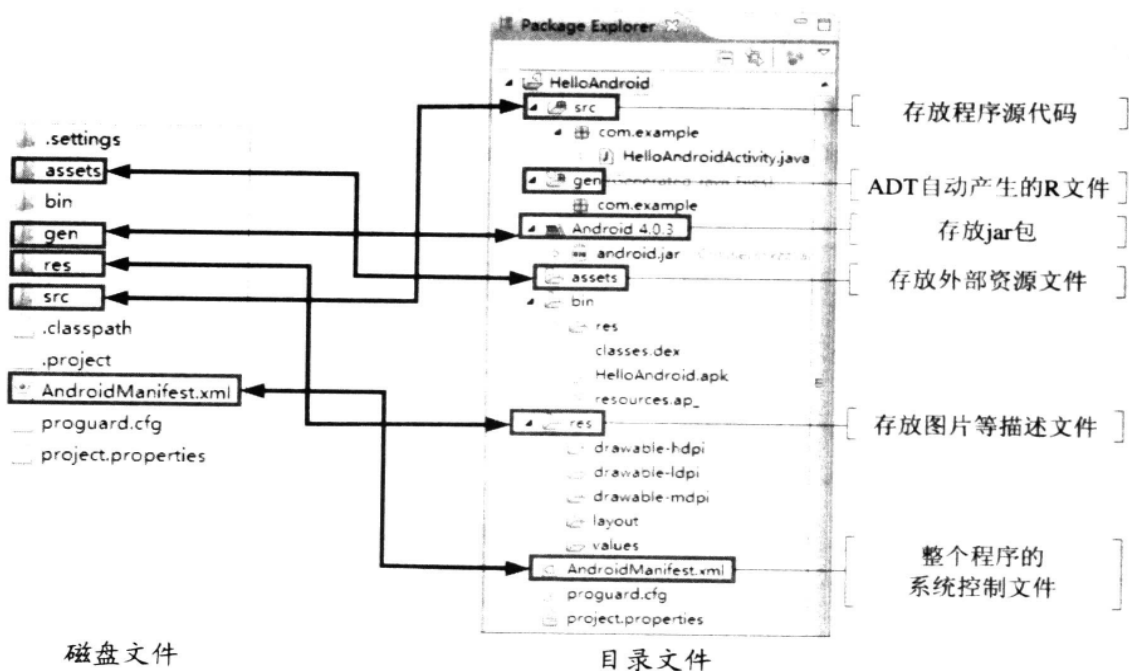


图 1.18 Android 程序框架



1.5 习题

1. 安装 SDK 并配置环境变量，在 DOS 窗口执行“adb”命令，检查安装是否成功完成。若执行效果如图 1.19 所示，则说明 SDK 安装成功，且环境变量配置正确。

```
C:\Windows\system32\cmd.exe
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation. 保留所有权利。

C:\Users\yztx5>adb
Android Debug Bridge version 1.0.29

-d                - directs command to the only connected USB device
                  returns an error if more than one USB device is
                  present.
-e                - directs command to the only running emulator.
                  returns an error if more than one emulator is r
                  unning.
-s <serial number> - directs command to the USB device or emulator w
                  ith
                  the given serial number. Overrides ANDROID_SERI
                  AL
                  environment variable.
-p <product name or path> - simple product name like 'sooner', or
                  a relative/absolute path to a product
                  out directory like 'out/target/product/sooner'.
                  If -p is not specified, the ANDROID_PRODUCT_OUT
                  environment variable is used, which must
```

图 1.19 检查 SDK 的安装情况

【分析】本题考查读者对 Android 环境搭建过程的掌握情况。首先安装 SDK，然后配置环境变量。如果“adb”命令无法执行，则很有可能是环境变量配置不正确。此时要确保 adb.exe 已经配置到“path”路径中，路径之间要用分号隔开。

2. 安装并配置 ADT，在 Eclipse 环境中打开 Windows 环境下的 Android SDK Manager，安装最新版本的 API，如图 1.20 所示。

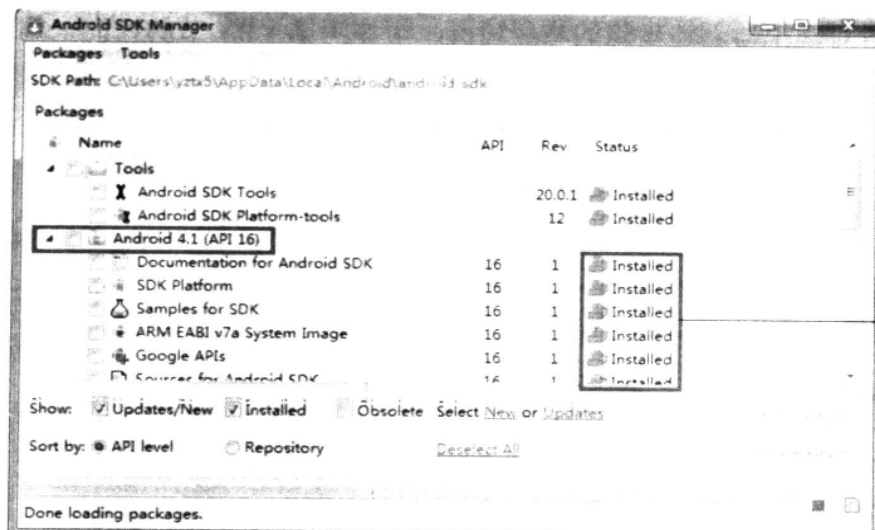


图 1.20 安装 API

【分析】第一次安装 ADT 时要安装其最新版本，以防因出现 Bug 而无法开发 Android 程序，之后可以根据需要安装其他版本。

3. 在 Eclipse 中创建 AVD，将其命名为“AVD4.1”，设置“Skin”为“QVGA”，“SDCard”为“5G”。创建完成后，查看该 AVD 的属性信息，如图 1.21 所示。启动 AVD4.1，如图 1.22 所示。

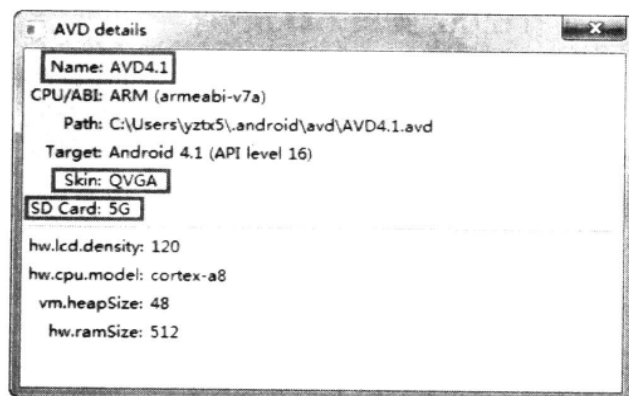


图 1.21 AVD4.1 的属性



图 1.22 成功启动 AVD4.1



【分析】本题考查读者对模拟器创建步骤的掌握情况，可参考第 1.2.7 节的内容来编写程序。

4. 在 DOS 环境下创建 AVD，将其命名为“mAVD”，设置“Skin”为“QVGA”，“SDCard”为“1G”。创建完成后，在 DOS 窗口中启动 mAVD，如图 1.23 所示。



图 1.23 在 DOS 窗口中启动 mAVD

【分析】本题考查读者对模拟器创建步骤的掌握情况，可参考第 1.2.7 节的内容来编写程序。注意，在 DOS 环境下创建 AVD，必须指定 AVD 的 target 属性，否则创建将会失败。

5. 开发一个 Android 程序，将其命名为“FirstAndroid”，修改布局文件的内容，显示文本“我的第一个 Android 程序”。运行程序，如图 1.24 所示。

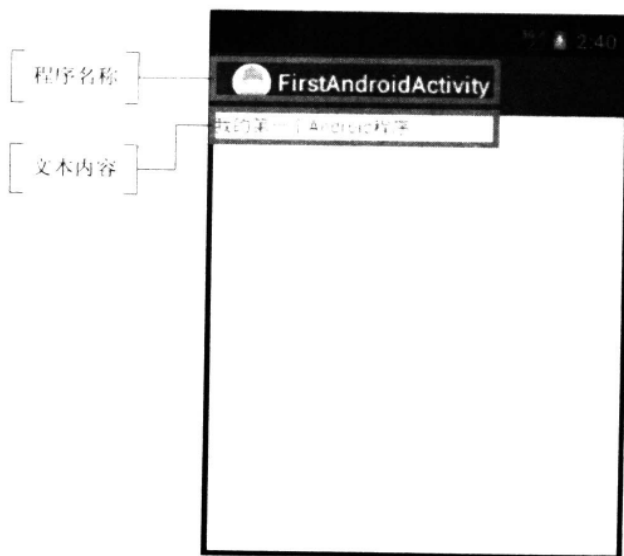


图 1.24 FirstAndroidActivity 界面

【分析】本题主要考查读者对 Android 程序开发过程的掌握情况，并检验 Android 环境是否搭建成功，可参考第 1.3.1 节的内容来编写程序。

【核心代码】本题的关键代码是布局文件中 TextView 控件的引用，具体如下。

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <TextView
```



```

android:id="@+id/textView1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentLeft="true"
android:layout_alignParentTop="true"
android:text="我的第一个 Android 程序" />

```

```
</RelativeLayout>
```

6. 在 Eclipse 环境的“Package Explorer”面板中打开“FirstAndroid”项目的目录，然后打开磁盘中的对应目录，查看项目中的各个文件与磁盘中文件的对应关系，如图 1.25 所示。

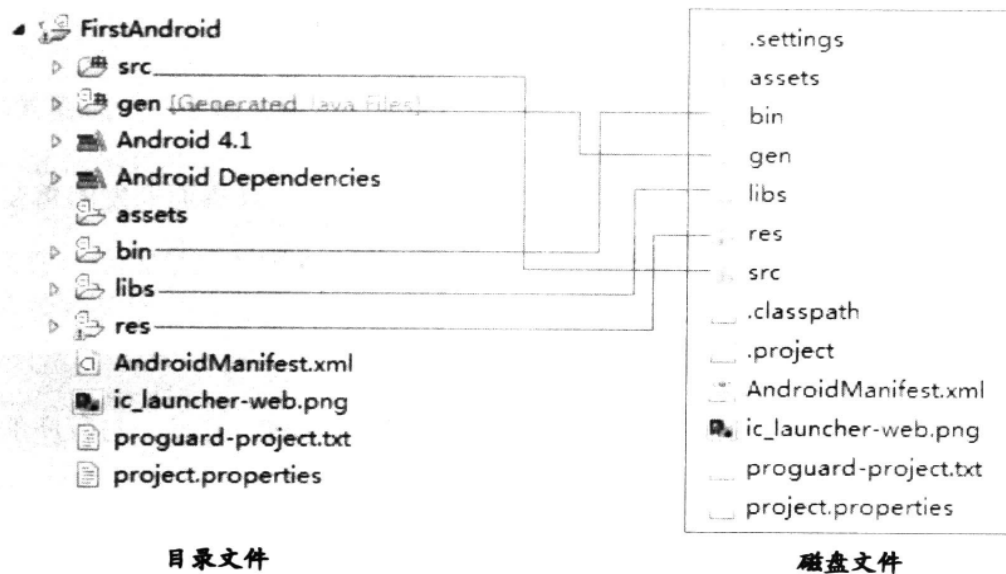


图 1.25 程序文件与磁盘文件的对应关系

【分析】本题主要考查读者对 Android 程序架构的掌握情况，要求读者了解每一个文件在项目中的作用，可参考第 1.3.2 节的内容来帮助理解。

第 2 章 Android 程序界面布局设计

本章主要介绍 Android 程序开发中的布局文件，它是 Android 界面开发的重要一环，是用户对程序的第一印象。界面开发的优劣，直接影响用户对程序认可与否。

2.1 布局概述

布局文件主要用于规范和设计用户界面（UI）。它采用 XML 开发，与程序的逻辑代码分离，使程序的开发变得清晰。

2.1.1 什么是布局

Android 的工作界面通常由容器和控件构成。为了规范控件在容器中的显示，设计人员通常需要规定控件在界面中的显示方式，这就是布局。

1. 布局的作用

在布局中，可以通过设置控件或者容器的属性来规定控件的显示方式，如图 2.1 所示。

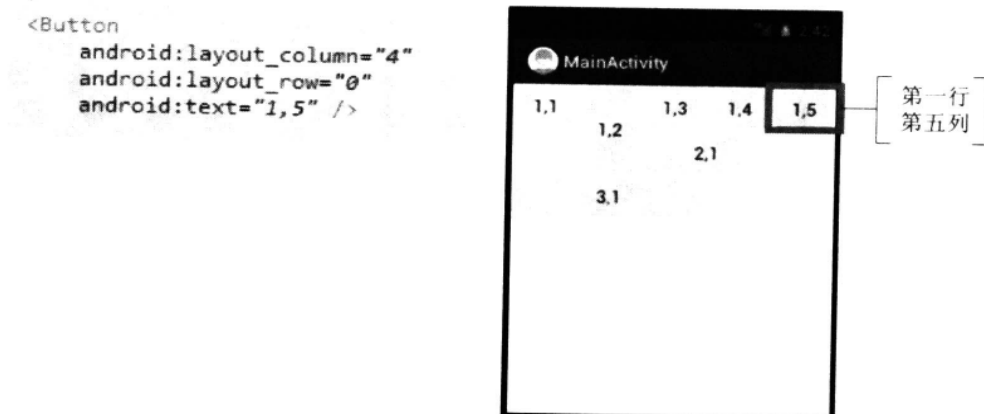


图 2.1 布局的作用 I

在如图 2.1 所示的代码中，`layout_column` 和 `layout_row` 指定了 `Button` 控件显示在第一行第五列，`text` 指定显示文本为“1,5”。

与图 2.1 中的代码相比，图 2.2 中的代码添加了 `layout_rowSpan` 属性和 `layout_gravity` 属性，指定按钮跨越两行并垂直填充。

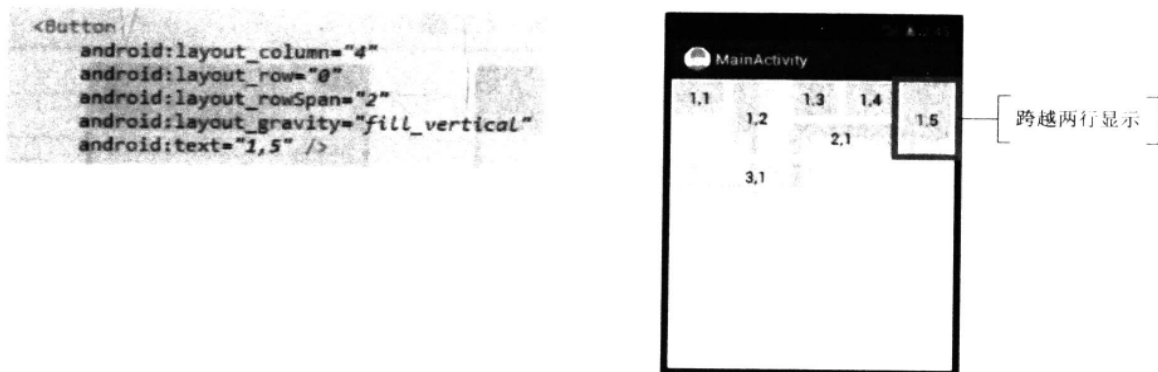


图 2.2 布局的作用 II

2. 布局的实现——布局文件

在 Android 应用程序中，界面通常是通过布局文件设定的。该文件采用 XML 格式。每个应用程序默认包含一个主界面布局文件，该文件位于项目文件“res”目录的“layout”子目录中。打开该文件，首先可以看到界面设计面板。切换至“activity_main.xml”选项卡，可以查看该界面所对应的布局文件，如图 2.3 所示。

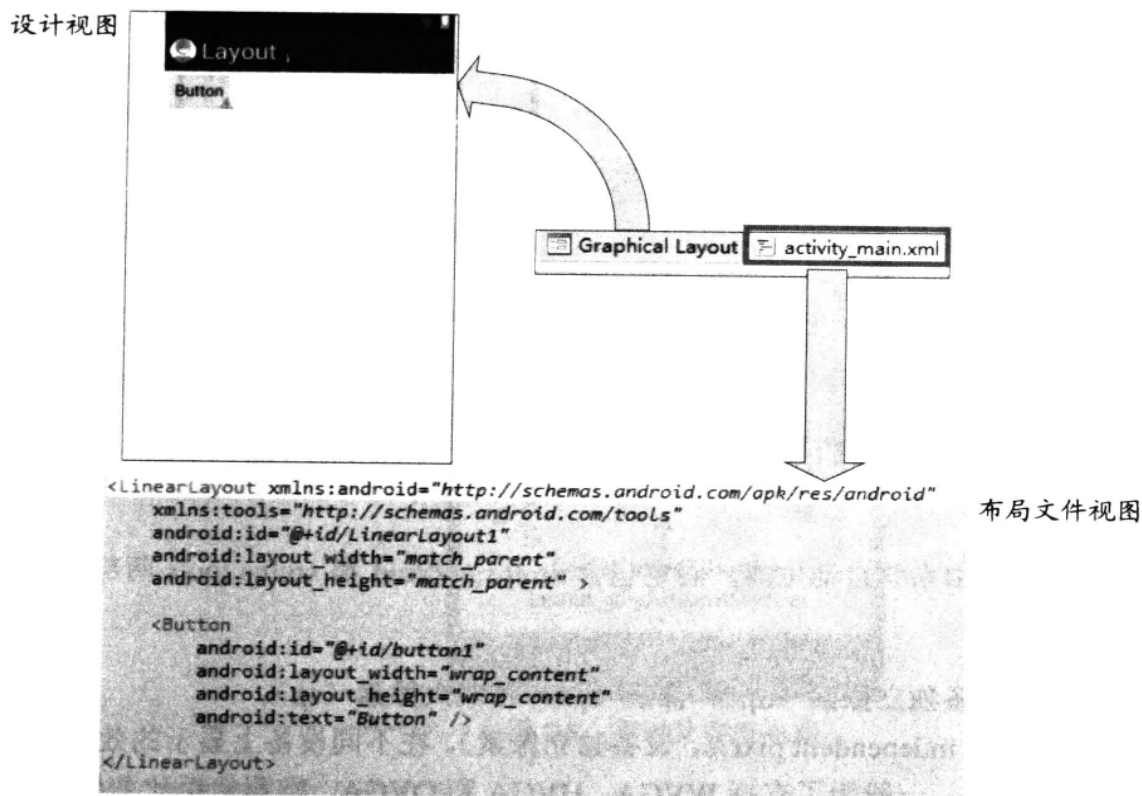


图 2.3 查看布局文件

新建 Android 应用程序的默认布局为相对布局。因此，开发时如果要使用其他布局，应对布局进行修改。

2.1.2 布局的类型

Android 布局分为相对布局、线性布局、表格布局、帧布局和网格布局 5 种，如图 2.4 所示。因为 Android 设备多种多样，分辨率不统一，在不同的机型上使用绝对布局，显示效果往



往差别很大，所以不推荐用绝对布局来设计 UI。



图 2.4 各种布局

2.1.3 布局文件的常用概念

布局文件是 Android 程序开发的必要文件，以下是布局文件的共有属性。

1. 命名空间

布局文件的命名空间由系统自动生成，固定包含 `xmlns:android` 和 `xmlns:tools` 两部分。

2. 距离单位

布局的距离单位有设备独立像素 (dp)、像素 (px) 和放大像素 (sp)。

- **dp**: 即 dip (device independent pixels, 设备独立像素)，在不同设备上显示的效果不同。它与设备硬件有关。一般为了支持 WVGA、HVGA 和 QVGA，推荐使用此单位，因为它不依赖像素。
- **px**: pixels (像素)，在不同的设备上显示效果相同。一般用 HVGA 代表 320×480 像素，此单位使用较多。
- **sp**: scaled pixels (放大像素)，主要用于字体的显示。

表示长度、高度等属性时可以使用 dp，设置字体大小时需要使用 sp。

3. 常用属性

布局的常用属性及含义如表 2.1 所示。



表 2.1 布局的常用属性

属性名称	属性说明
id	唯一标识布局文件
layout-width	布局的宽度
layout-height	布局的高度



2.2 相对布局

在 Eclipse 环境中开发程序时，默认采用相对布局（RelativeLayout）。相对布局包括相对容器布局和相对控件布局，下面一一介绍。

2.2.1 相对容器布局

子控件相对于父容器的相对布局，主要是相对于父容器的边框而言，如图 2.5 所示。
`layout_alignParentLeft` 表示子控件以父容器的左边缘为参照标准。`layout_marginLeft` 表示子控件左边缘与父容器左边缘之间的距离，单位为 dp。如果控件与左右边缘距离相等，可以直接使用 `layout_centerHorizontal` 设置水平居中。同理，如果控件与上下边缘距离相等，可以直接使用 `layout_centerVertical` 设置垂直居中。如果控件与上下边缘距离相等，且与左右边缘距离也相等，就可以使用 `layout_centerInParent` 设置该控件在整个父容器中居中。

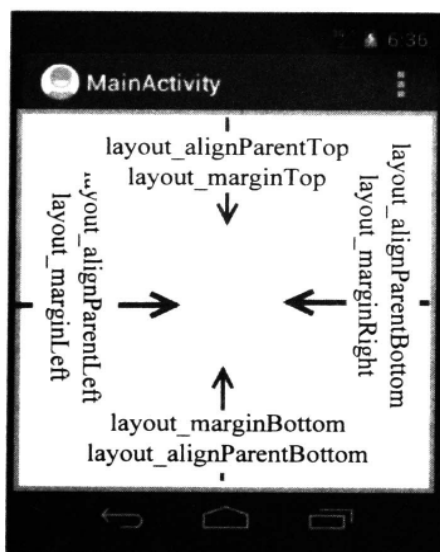


图 2.5 相对父容器布局

相对父容器布局的语法如下。

```

<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    <Widgets>
        .....
        .....
        android:layout_centerHorizontal="true"
    </Widgets>
</RelativeLayout>

```

命名空间

布局的宽度和高度

任意控件

在父容器中的居中方式



```
android:layout_alignParentLeft=" " [与父容器对齐的方式]
android:layout_marginLeft=" " [该控件左侧与父容器
                              之间的距离]
.....
.....
/>
</RelativeLayout>
```

相对父容器布局的语法包括以下部分，语法属性如表 2.2 所示。

- `xmlns:android` 和 `xmlns:tools` 为命名空间，由系统自动产生。
- `layout-width` 和 `layout-height` 分别为布局的宽度和高度，参数有 `fill-content`、`match-content` 和 `wrap-content`。
- `layout_centerHorizontal` 表示控件在父容器中水平居中，`layout_centerVertical` 表示控件在父容器中垂直居中，`layout_centerInParent` 表示控件位于父容器中央。

表 2.2 相对父容器布局的语法属性

属性名称	属性说明
<code>layout_alignParentLeft</code>	以父容器的左边缘为参照标准
<code>Layout_marginLeft</code>	控件左边缘与父容器左边缘的距离
<code>layout_alignParentRight</code>	以父容器的右边缘为参照标准
<code>Layout_marginRight</code>	控件右边缘与父容器右边缘的距离
<code>layout_alignParentTop</code>	以父容器的上边缘边为参照标准
<code>Layout_marginTop</code>	控件上边缘与父容器上边缘的距离
<code>layout_alignParentBottom</code>	与父容器的下边缘对齐
<code>Layout_marginBottom</code>	控件下边缘与父容器下边缘的距离

【示例 2-1】下面演示相对父容器布局的使用。其中，`Button1` 控件以父容器的上边缘为参照标准，距上边缘 189dp，在父容器中水平居中，代码如下。

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/
apk/res/android"
    xmlns:tools="http://schemas.android.com/
tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

        android:layout_alignParentTop="true"
        android:layout_marginTop="189dp" [该按钮上边缘距
                                        父容器上边缘189dp]

        android:layout_centerHorizontal="true" [该按钮在父容器中
                                                水平居中]
        android:text="Button1" />
</RelativeLayout>
```



在以上代码中，`layout_alignParentTop` 和 `Layout_marginTop` 指定了 `Button1` 控件在竖直方向上的位置，`layout_centerHorizontal` 指定了 `Button1` 控件在水平方向上的位置，这样就可以固定 `Button1` 控件的位置。



注意：如果不设置 `Layout_marginTop` 属性，则使用默认设置时，显示效果为与父容器上边缘对齐，如图 2.6 所示。

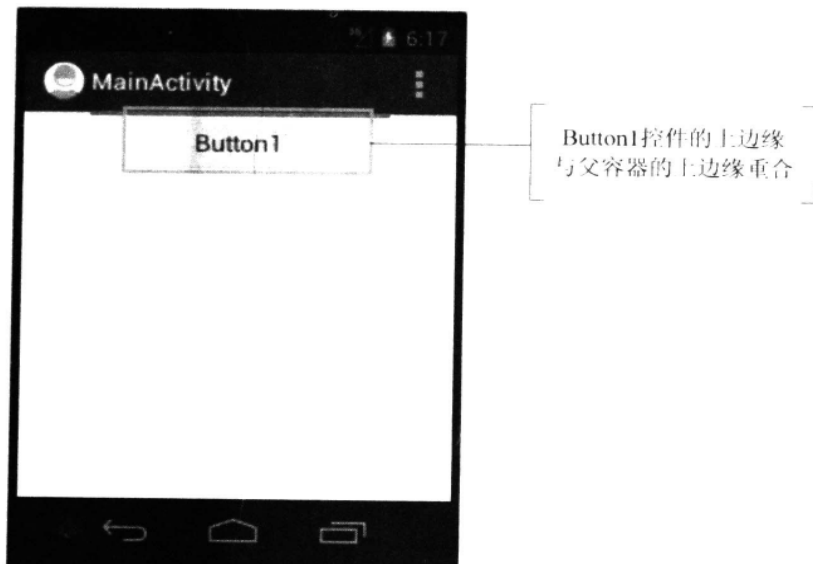


图 2.6 `Layout_marginTop` 取默认值

【示例 2-2】下面看一个控件居中的示例。`Button1` 控件在整个父容器中居中，即位于屏幕的中心，代码如下。

```
<RelativeLayout
    xmlns:android="http://
schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/
tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button1"

        android:layout_centerInParent="true"
    />
</RelativeLayout>
```



在整个父容器中居中

2.2.2 相对控件布局

相对控件布局是指一个控件以另一个控件为参照物布局。首先要确定该控件的位置，通过 `id` 唯一标识具体控件。如图 2.7 所示，`layout_above` 表示其他控件相对位于该控件的上方，`layout_alignTop` 表示其他控件以该控件上边缘为参照标准，`layout_marginTop` 表示当前控件上边缘与已知控件之间的距离，单位为 `dp`。

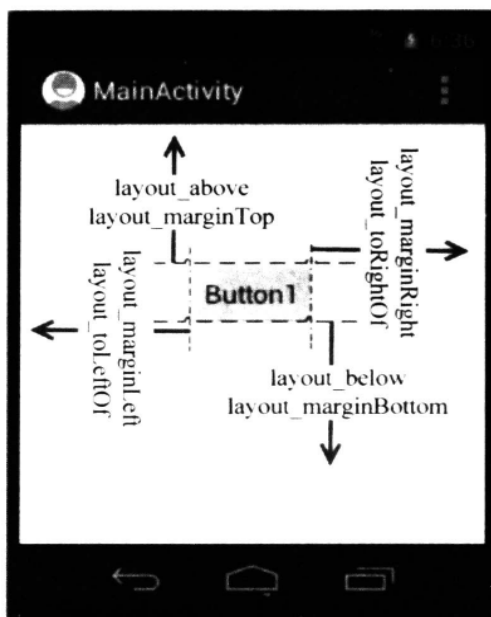


图 2.7 相对控件布局

相对控件布局的语法如下。

<pre><RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android" xmlns:tools="http://schemas.android.com/tools"</pre>	命名空间
<pre> android:layout_width=" " android:layout_height=" " ></pre>	布局的高度和宽度
<pre> <Widgets android:id="@+id/Widget1" /></pre>	已知控件的id
<pre> <Widgets android:id="@+id/Widget2"</pre>	未知控件的id
<pre> android:layout_alignBottom="@+id/Widget1"</pre>	与已知控件的对齐方式
<pre> android:layout_marginBottom=" "</pre>	该控件底部与已知控件之间的距离
<pre> android:layout_toRightOf="@+id/Widget1"/></pre>	该控件居于已知控件的右侧
<pre></RelativeLayout></pre>	

相对控件布局的语法包括以下部分，语法属性如表 2.3 所示。

- `xmlns:android` 和 `xmlns:tools` 为命名空间，由系统自动产生。
- `layout-width` 和 `layout-height` 分别为布局的宽度和高度，参数有 `fill-content`、`match-content` 和 `wrap-content`。
- `android:id` 唯一标识某具体控件。
- `layout_toRightOf` 表示控件位于已知控件右侧，`layout_toLeftOf` 表示控件位于已知控件左侧，`layout_above` 表示控件位于已知控件上方，`layout_below` 表示控件位于已知控件下方，参数为已知控件的 `id`。



表 2.3 相对控件布局的语法属性

属性名称	属性说明
layout_alignLeft	以已知控件的左边缘为参照标准
Layout_marginLeft	控件左边缘与已知控件之间的距离
layout_alignRight	以已知控件的右边缘为参照标准
Layout_marginRight	控件右边缘与已知控件之间的距离
layout_alignTop	以已知控件的上边缘为参照标准
Layout_marginTop	控件上边缘与已知控件之间的距离
layout_alignBottom	以已知控件的下边缘为参照标准
Layout_marginBottom	控件下边缘与已知控件之间的距离
layout_alignBaseLine	以已知控件的 BaseLine 为参照标准

【示例 2-3】下面演示相对控件布局的使用。Button1 为已知控件；Button2 控件相对于 Button1 控件，位于其右侧，下边缘距 Button1 控件 85dp，代码如下。

```

<RelativeLayout
    xmlns:android="http://schemas.android.com/
    apk/res/android"
    xmlns:tools="http://schemas.android.com/
    tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="189dp"
        android:text="Button1" />
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/button1"
        android:layout_marginBottom="85dp"

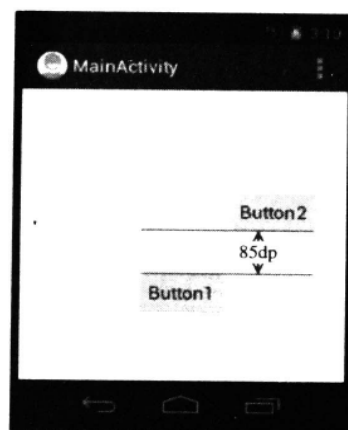
        android:layout_toRightOf="@+id/button1"
        android:text="Button2" />
</RelativeLayout>

```

Button1控件的id

Button2控件下边缘距Button1控件85dp

Button2控件位于Button1控件右侧



2.3 线性布局

线性布局（LinearLayout）包括两种，分别是水平线性布局和垂直线性布局，下面一一介绍。

2.3.1 什么是线性布局

线性布局是 Android 布局设计中较为常用的布局方式，它主要以水平或者垂直的方式来显示界面中的控件，如图 2.8 所示。

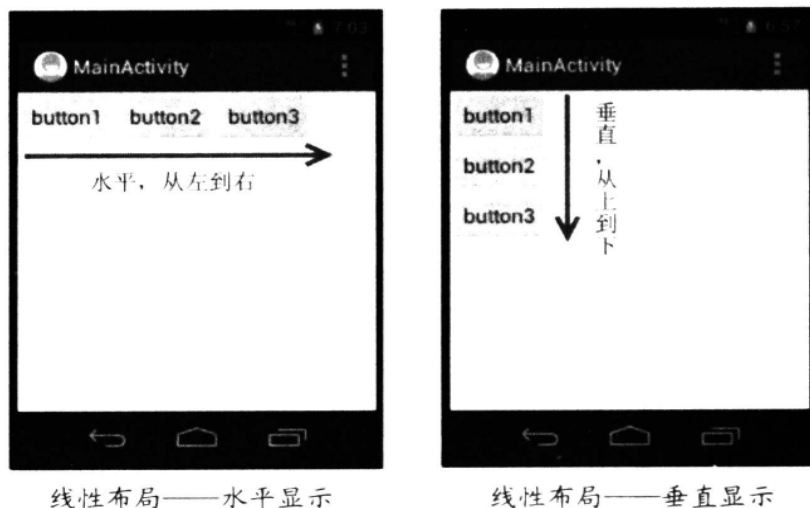
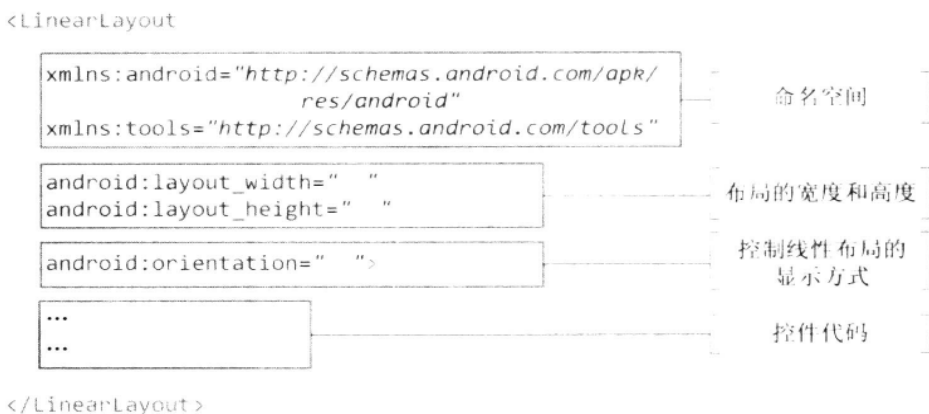


图 2.8 线性布局

其中，水平显示顺序为从左到右，垂直显示顺序为从上到下。

2.3.2 线性布局的语法

Android 为线性布局提供了一个标签——`<LinearLayout>`，代码如下。



线性布局的语法包括以下部分。

- `xmlns:android` 和 `xmlns:tools` 为命名空间，由系统自动产生。
- `layout-width` 和 `layout-height` 分别为布局的宽度和高度，参数有 `fill-content`、`match-content` 和 `wrap-content`。
- `orientation` 属性可以控制线性布局的显示方式，有 `vertical`（垂直）和 `horizontal`（水平）两种，默认为 `horizontal` 方式。
- 控件代码由开发者编写。

2.3.3 创建线性布局

使用 Eclipse 创建 Android 项目，默认使用相对布局。如果要使用线性布局，就需要改变项目的布局方式，操作方法为：打开 `activity_main.xml` 布局文件的设计视图，在“Outline”面板中右击“Relative-Layout”分支，从弹出的快捷菜单中选择“Change Layout”命令修改布局，如图 2.9 所示。

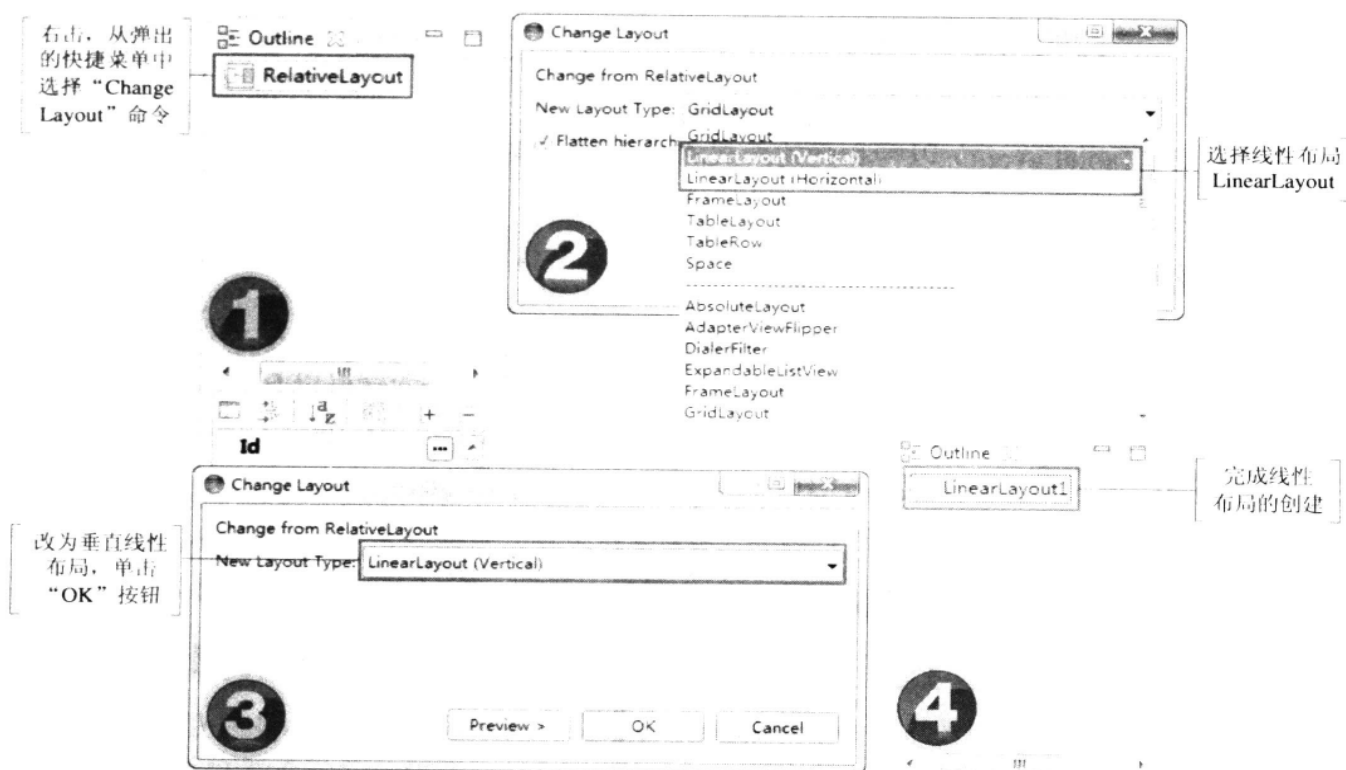


图 2.9 创建线性布局

在图 2.9 的 2 号对话框中可以看到，线性布局有 vertical 和 horizontal 两种，开发者可以根据设计需要自行选择。当选择了一种线性布局方式后，界面布局就改为线性布局。

【示例 2-4】下面演示线性布局的使用。打开 activity_main.xml 布局文件的设计视图，从“Form Widgets”面板中拖拽 3 个 Button 控件到主界面，切换到布局文件视图，就可以看到线性布局的代码了，具体如下。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <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
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="button3"/>
</LinearLayout>
```



在以上代码中，orientation 属性设置为“vertical”，表示采用垂直显示方式，3 个 <Button> 标签代表界面上添加的 3 个按钮控件。读者也可以使用其他控件并查看显示效果。



2.4 表格布局

表格布局 (TableLayout) 以行和列的形式将界面划分成多个单元格，每个单元格中都可以添加控件。

2.4.1 什么是表格布局

表格布局是指以表格的形式来显示界面中添加的控件。表格的每一行为一个 TableRow，每当一个控件添加到 TableRow 中，就生成一个单元格。表格的每一行可以有 0 或多个单元格，一个单元格可以跨越多列。如图 2.10 所示是一个三行二列的表格布局，第一行和第二行中各有 3 个 TextView 控件，第三行中有 1 个 TextView 控件。

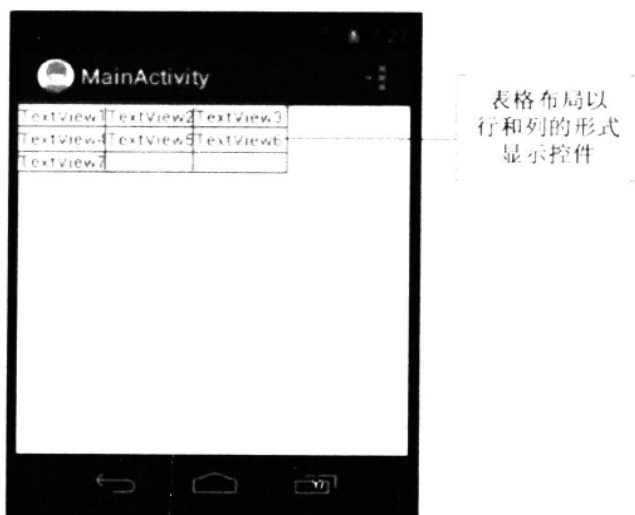
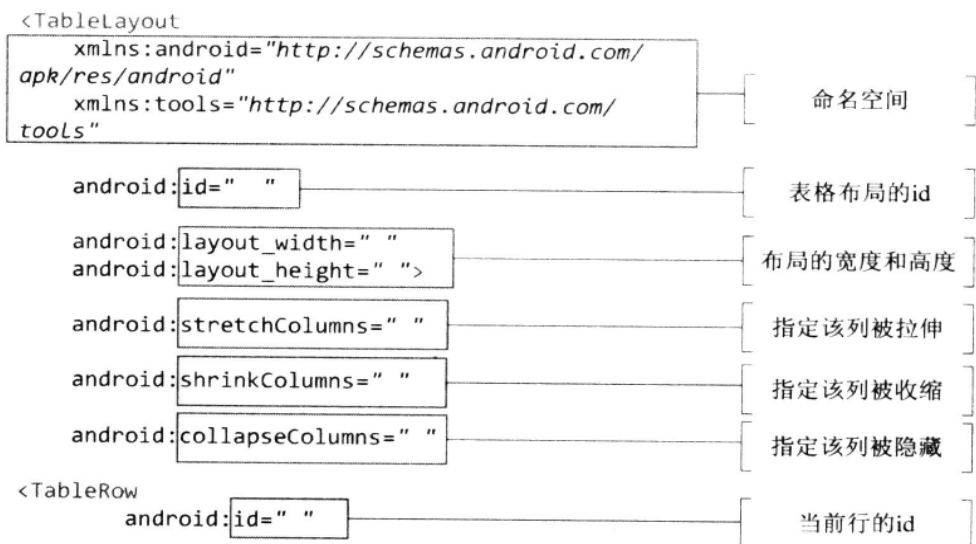
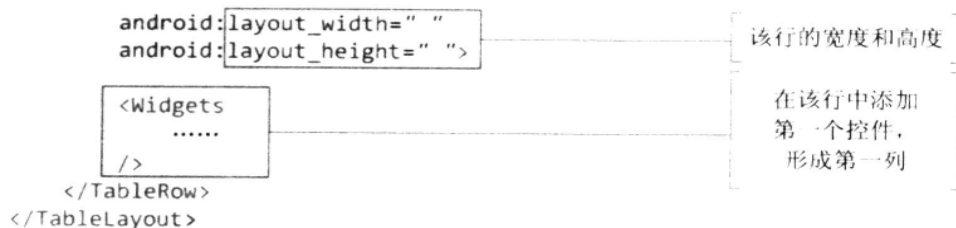


图 2.10 表格布局

2.4.2 表格布局的语法

表格布局针对列的设置提供了 3 个特有属性，分别是 stretchColumns、shrinkColumns 和 collapseColumns，具体如下。





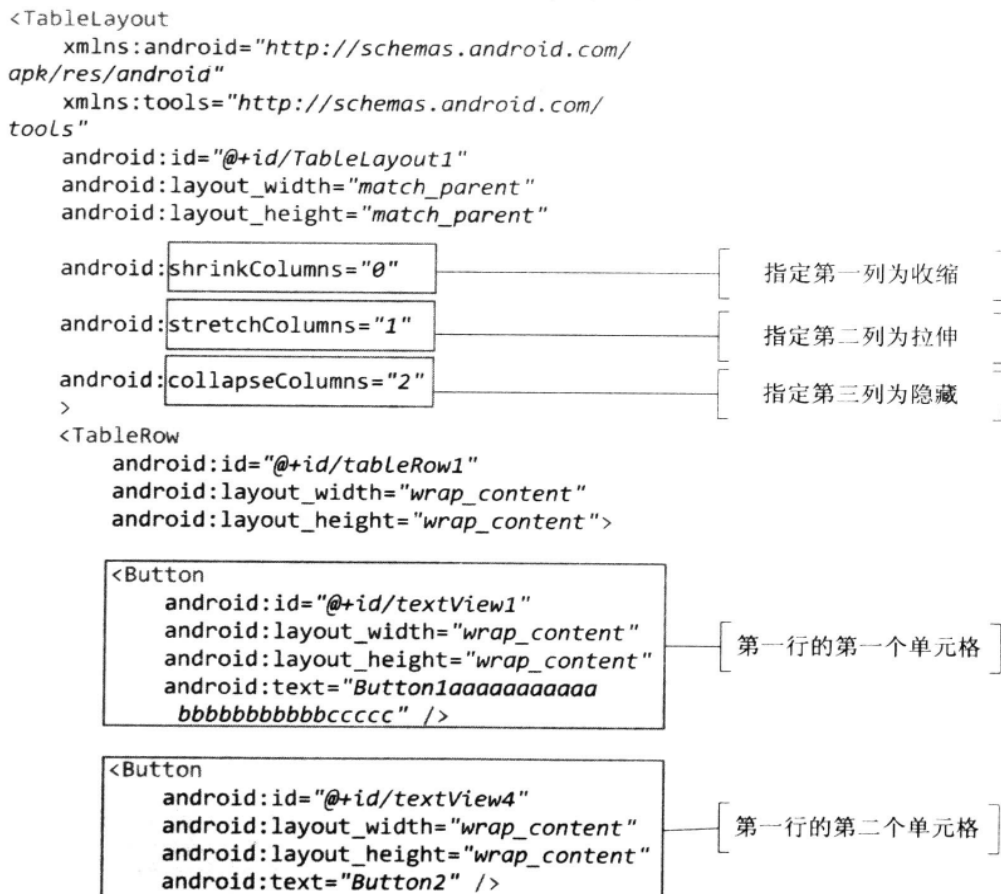
表格布局的语法包括以下部分。

- `xmlns:android` 和 `xmlns:tools` 为命名空间，由系统自动产生。
- `layout-width` 和 `layout-height` 分别为布局的宽度和高度，参数有 `fill-content`、`match-content` 和 `wrap-content`。
- `stretchColumns` 属性指定该列被拉伸，列号从 0 开始。
- `shrinkColumns` 属性指定该列被收缩，列号从 0 开始。
- `collapseColumns` 属性指定该列被隐藏，列号从 0 开始。
- `<TableRow>` 和 `</TableRow>` 这一对标签代表表格布局中的行。

2.4.3 创建表格布局

由于使用 Eclipse 创建 Android 项目时默认使用相对布局，所以，要使用表格布局，就要改变项目的布局方式，具体操作方法可参考线性布局的修改。

【示例 2-5】下面演示表格布局的使用。这是一个三行二列的表格布局，第一列设置为收缩，第二列设置为拉伸，第三列设置为隐藏，代码如下。





```
<Button
    android:id="@+id/textView5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button3" />
</TableRow>
.....
</TableLayout>
```

第一行的第三个单元格

从以上代码中可以看出，该表格布局的第一列被收缩，第二列被拉伸，第三列被隐藏，因此，第一行第一个单元格的内容折回，第二列拉伸至屏幕右边，第三列没有显示，如图 2.11 所示。



2.5 帧布局

帧布局 (FrameLayout) 是 Android 中最简单的布局方式，在日常程序开发中很少使用。

2.5.1 什么是帧布局

帧布局为每个加入其中的控件创建一个空白区域（称为一帧，每个控件占据一帧）。采用帧布局方式设计界面时，只能在屏幕左上角显示一个控件。如果添加了多个控件，这些控件会按顺序在屏幕左上角重叠显示，且会透明显示之前控件的文本。如图 2.12 所示，界面上添加了 3 个 Button 控件，Button1 是首先添加的大按钮，Button2 是接着添加的较小按钮，Button3 是最后添加的小按钮，三者叠加在屏幕左上角，但它们的文本内容都能显示出来。



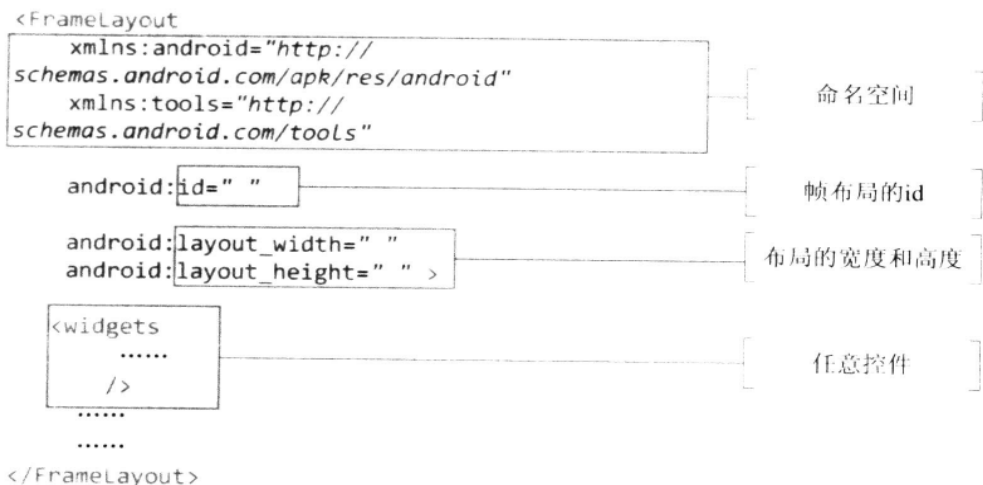
图 2.11 表格布局示例



图 2.12 帧布局

2.5.2 帧布局的语法

帧布局的语法非常简单，只要在布局框架中添加控件并将其显示出来即可，具体如下。



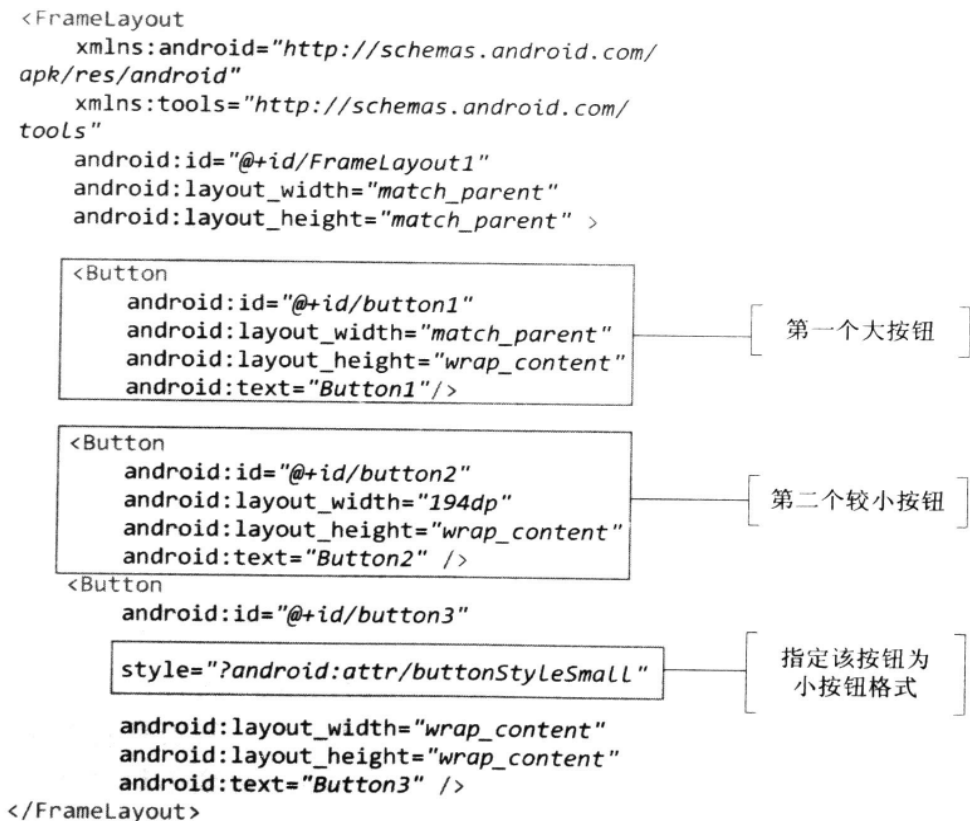
帧布局的语法包括以下部分。

- xmlns:android 和 xmlns:tools 为命名空间，由系统自动产生。
- layout-width 和 layout-height 分别为布局的宽度和高度，参数有 fill-content、match-content 和 wrap-content。
- id 可唯一标识该帧布局。
- <widgets> 标签代表任意可添加到界面的控件。

2.5.3 创建帧布局

由于使用 Eclipse 创建 Android 项目时默认采用相对布局，所以，要使用帧布局，就要改变项目的布局方式，具体操作方法可参考线性布局的修改。

【示例 2-6】下面演示帧布局的使用。以下是如图 2.12 所示界面的布局文件代码。



在这段代码中，引用的 Button3 控件不同于之前的 Button1 控件和 Button2 控件，它用专属



的 style 属性来指定该按钮为小按钮格式。



2.6 网格布局和布局控件

网格布局（GridLayout）是 Android 4.0 新增的布局。它实现了控件的交错显示，能够避免使用之前的布局嵌套，对设备性能的影响大大减小，更有利于自由编辑布局的开发。

2.6.1 什么是网格布局

网格布局用一组无限细的直线将绘图区域分割成行、列和单元，并指定控件的显示区域和控件在该区域的显示方式，形式如图 2.13 所示。

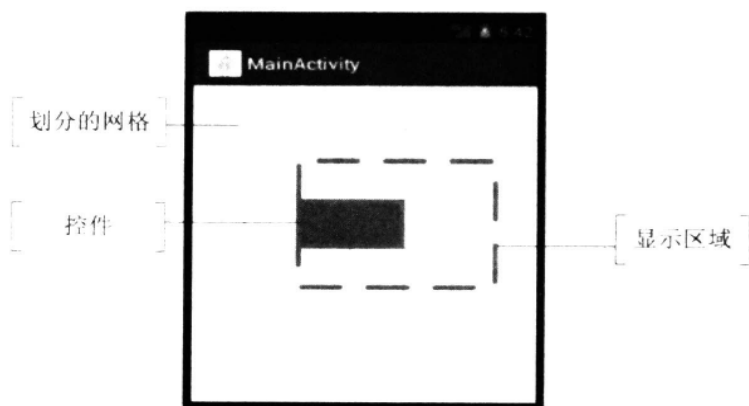
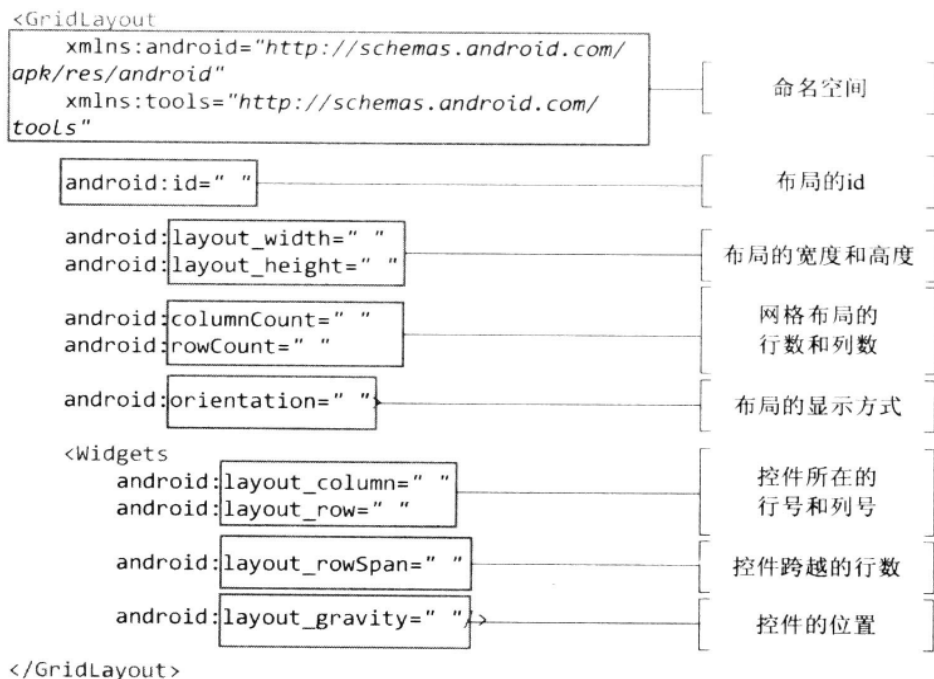


图 2.13 网格布局

2.6.2 网格布局的语法

网格布局是一种功能比较强大的布局方式，相关属性较多，具体如下。



网格布局的语法包括以下部分，如图 2.14 所示。

- xmlns:android 和 xmlns:tools 为命名空间，由系统自动产生。



- layout-width 和 layout-height 分别为布局的宽度和高度, 参数有 fill-content、match-content 和 wrap-content。
- columnCount 和 rowCount 指定网格布局的行数和列数。
- orientation 属性控制网格布局的显示方式, 有 vertical 和 horizontal 两种, 默认为 horizontal 方式。
- <widgets> 标签代表任意可添加到界面的控件。
- layout_raw 和 layout_column 表示控件处于网格中的行和列 (即某个单元格中)。
- layout_columnSpan 表示控件跨越的列数, layout_rowSpan 表示控件跨越的行数。
- layout_gravity 指定控件所在区域的位置。假设控件的显示区域为九宫格, 则使用不同的参数, 其显示效果不同, 如图 2.15 所示。

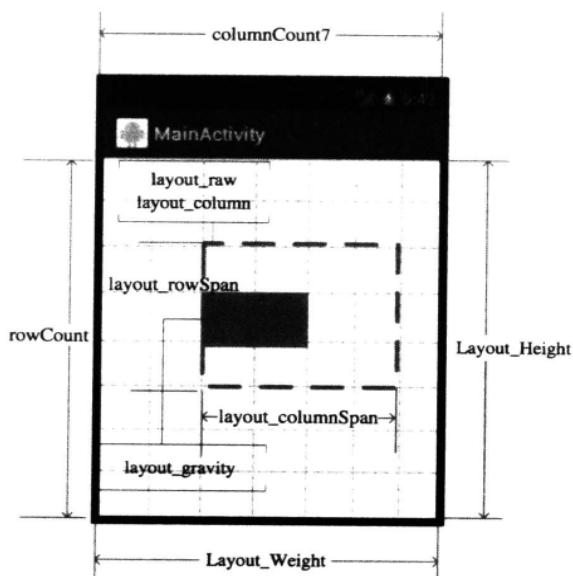


图 2.14 GridLayout 的语法

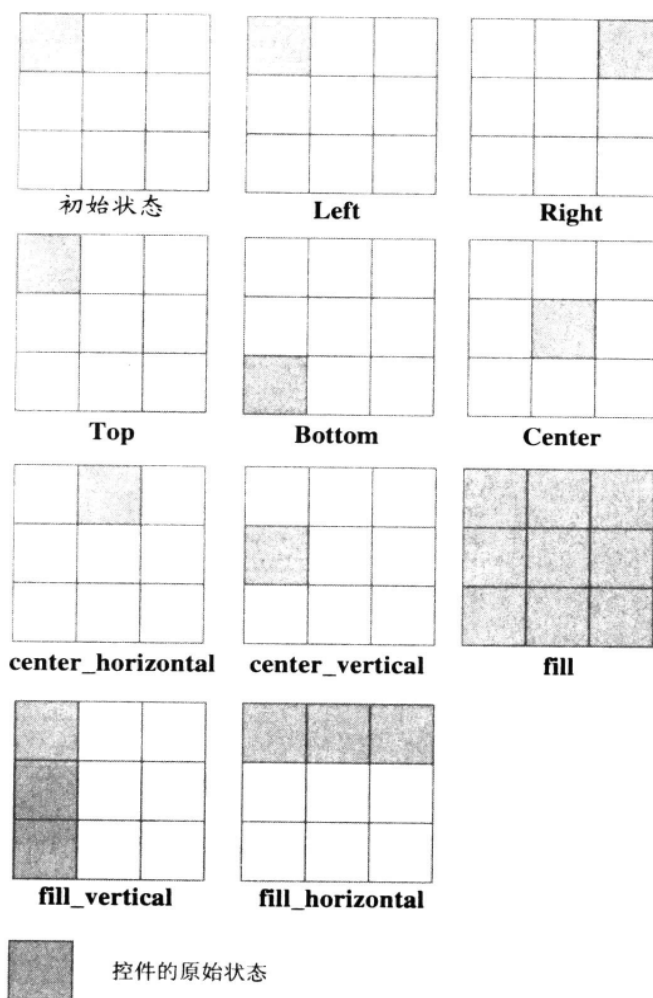


图 2.15 layout_gravity 的效果

2.6.3 创建网格布局

由于使用 Eclipse 创建 Android 项目时默认采用相对布局, 所以, 要使用网格布局, 就要改变项目的布局方式, 具体操作方法可参考线性布局的修改。

【示例 2-7】下面演示网格布局的使用, 以实现布局交错。在一个三行五列的布局中, Button 控件跨行跨列交错显示, 代码如下, 效果如图 2.16 所示。



```

<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/GridLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

    android:columnCount="5"
    android:rowCount="3"

    android:orientation="horizontal">

    <Button
        android:layout_column="0"
        android:layout_row="0"
        android:text="1,1" />
    <Button
        android:layout_column="1"
        android:layout_row="0"
        android:layout_rowSpan="2"

        android:layout_gravity="fill_vertical"
        android:text="1,2" />
    .....
    <Button
        android:layout_column="2"
        android:layout_row="1"
        android:layout_columnSpan="2"

        android:layout_gravity="fill_horizontal"
        android:text="2,1" />
    <Button
        android:layout_column="0"
        android:layout_row="2"
        android:layout_columnSpan="3"

        android:layout_gravity="fill_horizontal"
        android:text="3,1" />
</GridLayout>

```

三行五列的网格布局

水平显示

该Button控件位于第一行第一列

该Button控件位于第一行第二列

该Button控件跨越两行

垂直填充

该Button控件位于第二行第三列

该Button控件跨越两列

水平填充

该Button控件位于第三行第一列

该Button控件跨越三列

水平填充



图 2.16 交错的网格布局

2.6.4 什么是布局控件

布局控件（Space）是一个轻量级视图子类，用于分隔不同的控件，在其中形成一个空白



区域,以创建通用布局中组件之间的距离,代码如下。如图 2.17 所示是布局控件在网格布局中的使用。

<code><Space</code>	
<code>android:layout_column=" "</code>	布局控件所在的行号和列号
<code>android:layout_row=" "</code>	
<code>android:layout_gravity=" "</code>	布局控件的填充方式
<code>android:minHeight=" "</code>	布局控件的最小高度
<code>android:layout_columnSpan=" "/></code>	布局控件的跨越的列数

布局控件的语法包括以下部分。

- `layout_column` 和 `layout_row` 分别表示布局控件所在的行号和列号。
- `layout_gravity` 指定布局控件的位置。
- `minHeight` 表示布局控件的最小高度, `minWidth` 表示布局控件的最小宽度。
- `layout_columnSpan` 表示布局控件跨越的列数, `layout_rowSpan` 表示布局控件跨越的行数。

【示例 2-8】下面演示布局控件的使用。以下是如图 2.17 所示界面的布局文件代码。

<code><GridLayout</code>	
<code>xmlns:android="http://schemas.android.com/apk/res/android"</code>	
<code>xmlns:tools="http://schemas.android.com/tools"</code>	
<code>android:id="@+id/GridLayout1"</code>	
<code>android:layout_width="match_parent"</code>	
<code>android:layout_height="match_parent"</code>	
<code>android:columnCount="3"</code>	四行三列的网格布局
<code>android:rowCount="4"</code>	
<code>android:orientation="horizontal"</code>	水平显示
<code><Button</code>	
<code>android:layout_column="0"</code>	该Button控件位于第一行第一列
<code>android:layout_row="0"</code>	
<code>android:text="1,1" /></code>	
<code><Space</code>	
<code>android:layout_column="1"</code>	在第一行第二列插入布局控件
<code>android:layout_row="0"</code>	
<code>android:layout_gravity="fill_horizontal"/></code>	水平填充所在单元格
<code><Button</code>	
<code>android:layout_column="2"</code>	该Button控件位于第一行第三列
<code>android:layout_row="0"</code>	
<code>android:text="1,3" /></code>	
<code><Space</code>	
<code>android:layout_column="0"</code>	在第二行第一列插入布局控件
<code>android:layout_row="1"</code>	
<code>android:layout_columnSpan="3"</code>	跨越三列
<code>android:layout_gravity="fill_vertical"</code>	垂直填充所在单元格
<code>android:minHeight="100dp"/></code>	布局控件的最小高度为100dp
<code><Button</code>	
<code>android:layout_column="0"</code>	该Button控件位于第三行第一列
<code>android:layout_gravity="right"</code>	
<code>android:layout_row="2"</code>	
<code>android:text="3,1" /></code>	
<code></GridLayout></code>	

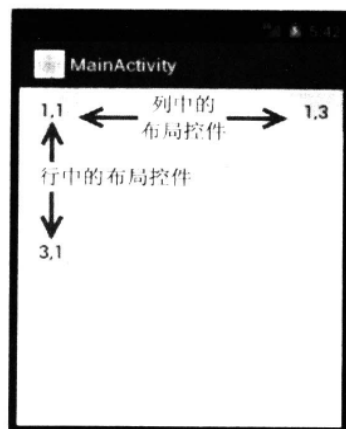


图 2.17 布局控件



2.7 小结

本章主要介绍了 Android 的常用布局，包括相对布局（RelativeLayout）、线性布局（LinearLayout）、表格布局（TableLayout）和帧布局（FrameLayout），以及 Android 4.0 中新增的网格布局（GridLayout）和布局控件（Space）。其中，读者需要重点掌握的是相对布局、线性布局和表格布局。本章的难点是网格布局和布局控件，需要读者多多练习掌握。

2.8 习题

1. 采用相对布局新建项目。在布局中添加一个按钮 Button1，使其相对父容器水平居中，上边缘距父容器上边缘 64dp，如图 2.18 所示。

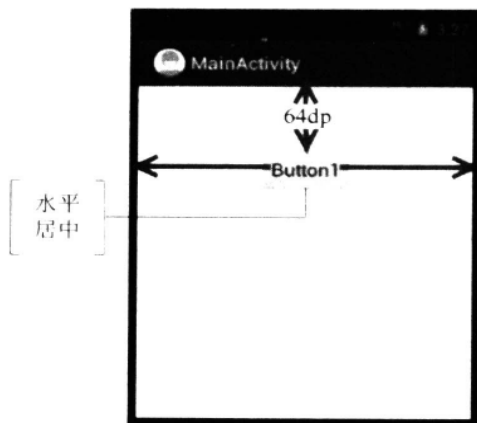


图 2.18 相对父容器布局示例

【分析】本题主要考查读者对相对父容器布局的掌握情况，可参考第 2.2.1 节的内容来开发程序。

【核心代码】本题的关键代码如下。

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="64dp"
        android:text="Button1" />

</RelativeLayout>
```

2. 修改第 1 题中的布局文件，添加一个按钮 Button2，使其位于 Button1 的右下方，且上边缘与 Button1 的下边缘相距 34dp，如图 2.19 所示。

【分析】本题主要考查读者对相对控件布局的掌握情况，可参考第 2.2.2 节的内容来开发程序。

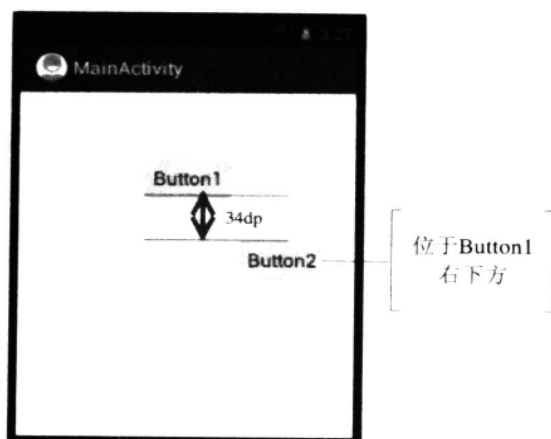


图 2.19 相对控件布局示例

【核心代码】本题的关键代码如下。

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="64dp"
        android:text="Button1" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/button1"
        android:layout_marginTop="34dp"
        android:layout_toRightOf="@+id/button1"
        android:text="Button2" />

</RelativeLayout>
```

3. 新建项目，先后采用垂直线性布局和水平线性布局在布局中添加两个按钮，如图 2.20 所示。



图 2.20 线性布局示例



【分析】本题主要考查读者对线性布局的掌握情况，可参考第 2.3 节的内容来开发程序。

【核心代码】本题的关键代码如下。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    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"/>

</LinearLayout>
```

4. 采用表格布局新建项目。在布局中构建一个二行三列的表格，添加 6 个按钮，Button1、Button2 和 Button3 位于第一行，Button4、Button5 和 Button6 位于第二行，设置第一列为拉伸，第二列为收缩，第三列为隐藏，如图 2.21 所示。

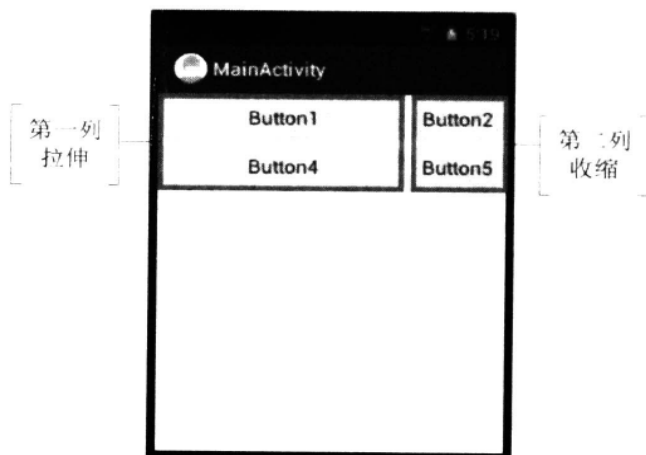


图 2.21 表格布局示例

【分析】本题主要考查读者对表格布局的掌握情况，可参考第 2.4 节的内容来开发程序。

【核心代码】本题的关键代码如下。

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/TableLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:shrinkColumns="1"
    android:stretchColumns="0"
    android:collapseColumns="2"
>
```



```
<TableRow
    android:id="@+id/tableRow1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >

    <Button
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button1" />

    <Button
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button2" />

    <Button
        android:id="@+id/textView5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button3" />

</TableRow>

<TableRow
    android:id="@+id/tableRow2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >

    <Button
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button4" />

    <Button
        android:id="@+id/textView6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button5" />

    <Button
        android:id="@+id/textView7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button6" />

</TableRow>

</TableLayout>
```

5. 采用帧布局新建项目，并在布局中添加一张系统提供的图片，如图 2.22 所示。

【分析】本题主要考查读者对帧布局的掌握情况，可参考第 2.5 节的内容来开发程序。开发程序时要注意系统图片的引用方法。



图 2.22 帧布局示例

【核心代码】本题的关键代码如下。

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/FrameLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@android:drawable/sym_def_app_icon" />

</FrameLayout>
```

6. 采用网格布局新建项目。在布局中添加 3 个按钮，并在这 3 个按钮之间添加布局控件，如图 2.23 所示。

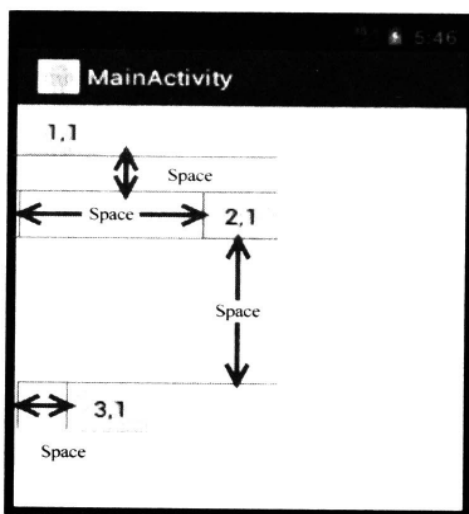


图 2.23 网格布局示例

【分析】本题主要考查读者对网格布局的掌握情况，可参考第 2.6 节的内容来开发程序。

【核心代码】本题的关键代码如下。

```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
```



```
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/GridLayout1"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:columnCount="2"
android:orientation="horizontal"
android:rowCount="5" >

<Button
    android:layout_column="0"
    android:layout_row="0"
    android:text="1,1" />

<Space
    android:layout_width="64dp"
    android:layout_column="1"
    android:layout_gravity="fill_horizontal"
    android:layout_row="0" />

<Button
    android:layout_column="0"
    android:layout_columnSpan="2"
    android:layout_gravity="center_horizontal"
    android:layout_row="2"
    android:text="2,1" />

<Space
    android:layout_width="1dp"
    android:layout_height="24dp"
    android:layout_column="0"
    android:layout_row="1" />

<Button
    android:layout_column="0"
    android:layout_gravity="right"
    android:layout_row="3"
    android:text="3,1" />

<Space
    android:layout_width="1dp"
    android:layout_height="168dp"
    android:layout_column="0"
    android:layout_row="2" />

<Space
    android:layout_height="172dp"
    android:layout_column="0"
    android:layout_columnSpan="2"
    android:layout_gravity="fill_vertical"
    android:layout_row="3"
    android:minHeight="100dp" />

<Space
    android:layout_width="100dp"
    android:layout_height="47dp"
    android:layout_row="4" />

</GridLayout>
```

第 3 章 基本控件

控件是界面的主要组成元素，是用户界面功能的表现。Android 的基本控件是开发 Android 程序的必要工具，包括 Button 类控件、文本类控件、图片控件和日期时间控件等。本章将详细介绍这些控件。



3.1 控件概述

Android 界面控件分为定制控件和系统控件两种。

- 定制控件是指用户独立开发的控件，或者通过继承并修改系统控件而产生的新控件，它能够为用户提供特殊的功能和与众不同的显示方式。
- 系统控件是 Android 系统提供给用户的已经封装的界面控件，包括应用程序开发过程中常用的功能控件。系统控件可以帮助用户进行快速开发，并能够使 Android 系统应用程序的界面保持一致。

3.1.1 控件的构成

Android 控件一般在“res/layout”目录下的布局文件中声明。每个控件都有自己的 id 以及在布局中显示的宽度（layout_width）和高度（layout_height）。除此之外，每个控件还有自己特有的属性。这些属性进一步规范了控件在布局中的显示效果，而控件就是通过对这些属性的设置实现的。

3.1.2 属性的使用

控件的属性也是在布局文件中设置的。设置控件的属性有两种方法，一种是在布局文件中设置参数，另一种是在代码中调用对应的方法。

对比如下两段代码：要想设置 TextView 控件的文本内容，一是在布局文件中设置 TextView 控件的 text 属性，二是在代码中调用 TextView.setText() 方法，运行结果均如图 3.1 所示，文本内容为“http://www.hao123.com/”。

```
<TextView
    android:id="@+id/tv2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/tv1"
```



```

android:textSize="30sp"
android:text="http://www.hao123.com/"
android:autoLink="web"
/>

```

通过设置text属性指定
文本内容

```

public class MainActivity extends Activity {
    TextView textView;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        textView = (TextView)findViewById(R.id.tv2);
        textView.setText("http://www.hao123.com/");
    }
}

```

调用TextView.setText()
方法设置文本内容



图 3.1 控件属性设置

3.1.3 方法和事件的使用

Activity 是 Android 的常用组件。在一个 Android 应用中，一个 Activity 就是一个单独的界面，每一个 Activity 被赋予一个窗口，开发人员可以在其中添加任意控件。窗口通常充满屏幕，但也可以小于屏幕而浮于其他窗口之上。

在布局文件中声明控件后，需要在 Activity 中通过使用 `super.setContentView(R.layout.布局文件名)` 方法来加载布局文件。在 Activity 中获取控件的引用需要先调用 `super.findViewById(R.id.控件的id)` 方法，再使用这个引用对控件进行操作，例如添加监听、设置内容等，代码如下，运行结果如图 3.2 所示。

```

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <TextView
        android:id="@+id/tv1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

        android:text="hello_world"
        android:textSize="25sp"/>

</RelativeLayout>

```

控件的id

文本内容



```

package com.example.day7_11;

import android.os.Bundle;

public class MainActivity extends Activity {
    TextView textView;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);
        textView = (TextView)findViewById(R.id.tv1);
        textView.setText("hello_android");
    }
}

```

声明一个 TextView 控件

加载布局文件

通过id获取控件

重置文本框内容

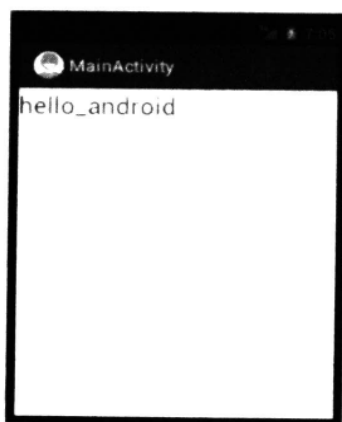


图 3.2 TextView 的文本重置

从图 3.2 中可以看出，在布局文件中，文本内容设置为“hello_world”，可是调用 `TextView.setText()` 方法后，重置文本内容为“hello_android”，文本内容更改成功。



3.2 文本类控件

文本类控件是 Android 程序开发中的常用控件，主要有 `TextView` 和 `EditText` 两大类，它们都可以用来显示文本。

3.2.1 文本框

文本框（`TextView`）是一种用于显示字符串的控件，它本身不具备可编辑属性，但在代码中通过调用对应方法来编辑文本内容，语法如下。

```

<TextView
    android:layout_width=" "
    android:layout_height=" "

    android:width=" "
    android:height=" "

    android:text=" "

    android:textSize=" "

    android:textColor=" "

    android:textStyle=" "

```

TextView 控件边框包围的内容

TextView 控件的准确宽度和高度

文本内容

文本字号

文本颜色

文本字体



android:gravity=" "	文本的显示位置
android:autoLink=" "	是否转换为可点击的超链接形式
android:singleLine=" "	是否只在一行内显示全部内容
android:ellipsize=" "/>	内容的省略显示方式

TextView 控件包含很多可以在 XML 文件中设置的属性，这些属性同样可以在代码中动态声明，如表 3.1 所示。图 3.3 是 TextView 控件的语法示意图。

表 3.1 TextView 控件的常用属性及对应方法说明

属性名称	对应方法	说 明
android:autoLink	setAutoLinkMask(int)	设置是否将指定格式的文本转换为可点击的超链接形式,传入的参数值可取 ALL、EMAIL_ADDRESSES、MAP_ADDRESSES、PHONE_NUMBERS 和 WEB_URLS
android:height	setHeight(int)	以像素为单位定义 TextView 控件的高度
android:width	setWidth(int)	以像素为单位定义 TextView 控件的宽度
android:singleLine	setTransformationMethod(TransformationMethod)	设置文本内容只在一行内显示
android:text	setText(CharSequence)	为 TextView 控件设置显示的文本内容
android:textColor	setTextColor(ColorStateList)	设置 TextView 控件的文本颜色
android:textSize	setTextSize(float)	设置 TextView 控件的文本字号
android:textStyle	setTypeface(Typeface)	设置 TextView 控件的文本字体
android:ellipsize	setEllipsize(TextUtils.TruncateAt)	如果设置了该属性,当要显示的内容超过 TextView 控件的长度时,会对内容进行省略,可取的值得有 start、middle、end 和 marquee

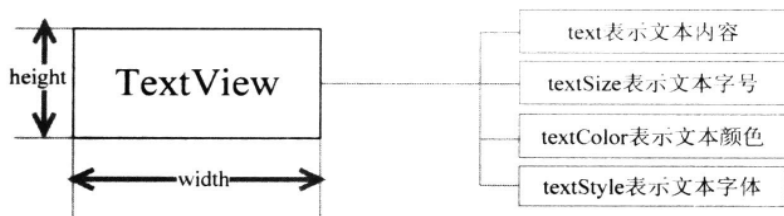


图 3.3 TextView 控件的语法

【示例 3-1】对 TextView 控件进行不同设置的代码如下，显示效果如图 3.4 所示。

<TextView		
android:layout_width="wrap_content"		TextView 控件边框
android:layout_height="wrap_content"		包围的内容
android:width="200dp"		TextView 控件的
android:height="50dp"		准确宽度和高度
android:text="TextView"		文本内容
android:textSize="30sp"		文本字号为 30sp
android:textColor="#FF0000"		文本颜色为红色
android:textStyle="italic"		文本字体为斜体
android:gravity="center_horizontal"/>		文本水平居中



图 3.4 TextView 控件

从图 3.4 中可以看出：第一个 TextView 控件中的内容显示效果为基本设置效果；第二个 TextView 控件中的内容显示效果为可点击的超链接形式；第三个 TextView 控件中的内容超过了显示长度的限制，所以中间部分被省略。

3.2.2 编辑框

编辑框（EditText）是用来输入和编辑字符串的控件，是一种具有编辑功能的 TextView 控件。EditText 是 TextView 的子类，它除了具有 TextView 的部分属性外，还有一些自己的属性，如表 3.2 所示。图 3.5 是 EditText 控件的语法示意图。

表 3.2 EditText控件的常用属性及对应方法说明

属性名称	对应方法	说 明
android:lines	setLines(int)	通过设置固定的行数来决定 EditText 控件的高度
android:maxLines	setMaxLines(int)	设置最大行数
android:minLines	setMinLines(int)	设置最小行数
android:password	setTransformationMethod(TransformationMethod)	设置文本框中的内容是否显示为密码
android:phoneNumber	setKeyListener(KeyListener)	设置文本框中的内容只能是电话号码
android:scrollHorizontally	setHorizontallyScrolling(boolean)	设置文本框是否可以水平滚动
android:capitalize	setKeyListener(KeyListener)	如果设置，将自动转换用户输入的内容为大写字母
android: hint	setHint(int)	文本为空时显示提示信息
android: numeric	setKeyListener(KeyListener)	如果设置，则输入的内容只能为数字
android:maxLength	setFilters(InputFilter)	设置最大显示长度

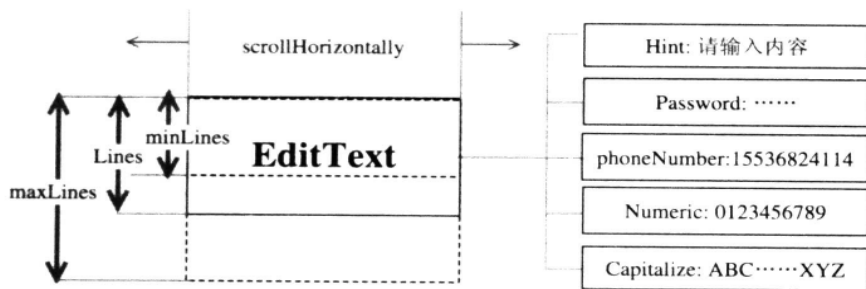
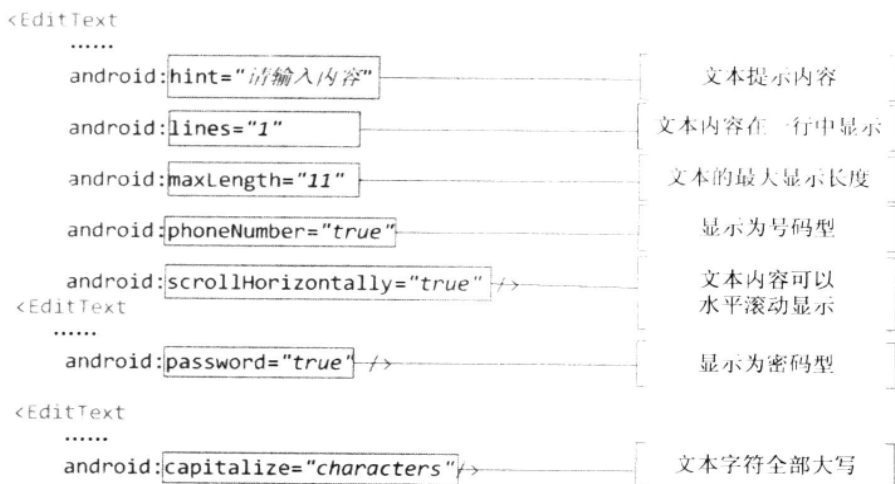


图 3.5 EditText 控件的语法

【示例 3-2】 EditText 控件的使用方法示例如下。



如图 3.6 所示：在第一个 EditText 控件中输入内容前，显示提示信息“请输入内容”，输入内容显示为号码型；第二个 EditText 控件中的内容显示为密码型；第三个 EditText 控件中的字符全部为大写。



3.3 按钮类控件

按钮（Button）在 Android 程序开发中也是较为常用的一类控件，主要包括 Button、ImageButton、ToggleButton、RadioButton 和 CheckBox 等。

3.3.1 按钮

Button 是 TextView 的子类，具有 TextView 的所有属性。用户可以通过单击 Button 控件来触发一系列事件，然后为 Button 控件注册监听器，以实现 Button 控件的监听事件。

为 Button 控件注册监听有两种方法：一种方法是在布局文件中为 Button 控件设置 OnClick 属性，然后在代码中添加一个 public void 参数值；另一种方法是在代码中绑定匿名监听器并重写 onClick() 方法。代码如下，详细过程如图 3.7 所示。



运行初始效果



输入文本内容

图 3.6 EditText 控件示例



```
<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:layout_alignParentLeft="true"
    android:layout_marginTop="60dp"
    android:onClick="click"
    android:text="button2" />

public class MainActivity extends Activity {
    Button button1,button2;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        button1 = (Button)findViewById(R.id.button1);
        button2 = (Button)findViewById(R.id.button2);

        button1.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                // TODO: Auto-generated method stub
                button1.setText("点击");
            }
        });

        public void click(View v) {
            button2.setText("注册成功");
        }
    }
}
```

设置OnClick属性为
"click"

绑定单击监听事件

重写OnClick()方法

添加click()方法，即
OnClick属性的参数值



运行初始效果

触发监听事件

图 3.7 Button 控件的监听注册

Button1 首先绑定匿名监听，然后重写 `onClick()` 方法，最后在监听器代码中调用 `Button.setText()` 方法来触发修改文本监听事件；而 Button2 首先在布局文件中设置 `onClick` 属性绑定监听，然后在代码中添加对应方法来触发修改文本监听事件。



注意：Button 控件的 `onClick` 属性，其参数值为在代码中添加的对应方法名，因此在设置该参数值时需注意命名的规范性。



【示例 3-3】下面通过一个示例来说明如何将图片作为 Button 控件的背景，代码如下。

```
button3.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        // TODO Auto-generated method stub
        button3 . setBackground(getResources().getDrawable(R.drawable.android));
    }
});
```

在以上代码中，首先为 Button 控件注册监听，用户单击按钮即触发监听事件。此时将调用 `Button.setBackground()` 方法，并通过图片的引用获取图片资源，把图片作为 Button 控件的背景来显示，运行效果如图 3.8 所示。



图 3.8 更换 Button 控件的背景图片

另外，也可以在布局文件中设置 Button 控件的 `background` 属性为图片的引用，将图片显示为 Button 控件的背景，而不需要编写代码——将以上代码替换为 “`android:background="@drawable/android"`”，便可实现相同的显示效果。

3.3.2 图片按钮

图片按钮(`ImageButton`)控件与 Button 控件的主要区别是 `ImageButton` 控件没有 `text` 属性，即按钮中显示的是图片而不是文本，如图 3.9 所示。在 `ImageButton` 控件中，要设置按钮显示的图片，可以通过 `android:src` 属性来实现，也可以通过 `setImageResource(int)` 方法来实现，代码如下。



图 3.9 ImageButton 控件



```

<ImageButton
    android:id="@+id/imageButton1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:layout_marginLeft="38dp"
    android:layout_marginTop="130dp"
    android:src="@drawable/ic_launcher" />

```

通过引用设置背景图片

【示例 3-4】下面给出一个单击 ImageButton 控件并改变其背景图片的示例。

(1) 在 “res/drawable-mdpi” 目录下新建一个 myselector.xml 文件，在其中输入如下代码。

```

<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="false"
        android:drawable="@drawable/ic_launcher" />
    <item android:state_pressed="true"
        android:drawable="@drawable/android" />
</selector>

```

(2) 设置布局文件中 ImageButton 控件的 src 属性参数为 myselector.xml 的引用，代码如下。

```

android:src="@drawable/myselector"

```

注意：由于本示例并不需要在 Activity 部分进行额外的代码开发，故不需要对创建项目时自动生成的代码进行修改。

完成上述步骤之后，运行程序。

当按下 ImageButton 控件时，背景图片显示为 ic_launcher.png，当按住 ImageButton 控件时，背景图片显示为 android.png，当松开 ImageButton 控件时，背景图片又恢复为 ic_launcher.png，如图 3.10 所示。



图 3.10 ImageButton 控件示例

3.3.3 开关按钮

开关按钮 (ToggleButton) 是 Android 系统中比较简单的一个控件，它带有亮度指示，具有选中 and 未选中两种状态，不同的状态需要设置不同的显示文本。ToggleButton 控件的常用属性及说明如表 3.3 所示。

表 3.3 ToggleButton控件的常用属性及说明

属性名称	属性说明
android:textoff	未选中时按钮显示的文本
android:texton	选中时按钮显示的文本

【示例 3-5】下面通过一个示例来介绍 ToggleButton 控件，代码如下，运行效果如图 3.11 所示。

```
<ToggleButton
    android:id="@+id/toggleButton1"
    android:layout_width="150dp"
    android:layout_height="80dp"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:textOn="开"
    android:textOff="关"/>
```

选中时显示的文本

未选中时显示的文本



图 3.11 ToggleButton 控件示例

未选中 ToggleButton 控件时，显示文本为“关”；选中 ToggleButton 控件时，显示文本为“开”，并伴有亮度指示。

3.3.4 单选按钮

单选按钮（RadioButton）控件在 Android 平台上的应用也比较广泛，例如设置一些选项时会用到单选按钮。它是一种单个圆形单选框按钮，有选中和未选中两种状态。在 RadioButton 控件没有被选中时，用户能够通过按下或单击来选中它，但是在选中它后，通过单击无法将它变为未选中状态。

单选按钮由 RadioButton 和 RadioGroup 两部分组成。RadioGroup 是单选组合框，用于将 RadioButton 组合起来。在多个 RadioButton 被 RadioGroup 包含的情况下，同一时刻用户只能选中一个 RadioButton，并用 setOnCheckedChangeListener() 方法对 RadioGroup 进行监听。

【示例 3-6】下面给出一个 RadioButton 控件的使用示例。

(1) 创建一个项目，在布局文件中添加一对 <RadioGroup> 和 </RadioGroup> 标签来表示一个单选组合框，并在其中添加两个 RadioButton 控件。RadioGroup 的 orientation 属性用于控制选项的显示方式，vertical 为垂直显示，horizontal 为水平显示，本示例采用垂直显示方式，代码如下。



```

<RadioGroup
    android:id="@+id/rg1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
</RadioGroup>

<RadioButton
    android:id="@+id/rb1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="30sp"
    android:text="舍"
</RadioButton>

<RadioButton
    android:id="@+id/rb2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="30sp"
    android:text="得"
</RadioButton>

```

选项以垂直方式显示

选项的文本内容

选项的文本内容

(2) 编写逻辑代码，在代码中为 RadioGroup 添加监听，通过检查 id 判断当前被选中的选项，然后作出相应的事件响应，具体如下。

```

radioGroup.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
    public void onCheckedChanged(RadioGroup group, int checkedId) {
        // Auto-generated method stub
        if (checkedId == R.id.rb1) {
            textView.setText("您选择的是：" + radioButton1.getText());
        } else {
            textView.setText("您选择的是：" + radioButton2.getText());
        }
    }
});

```

绑定选择监听事件

通过检查id来判断当前被选中的选项

(3) 运行程序并查看效果，如图 3.12 所示。可以看到，两个选项垂直显示，当选择不同的选项时，文本就变为所对应的选项内容。



图 3.12 RadioButton 控件示例

3.3.5 复选按钮

复选按钮 (CheckBox)，顾名思义，是一种可以进行多项选择的按钮，默认以矩形复选框表示。与 RadioButton 控件一样，CheckBox 控件也有选中和未选中两种状态。可以先在布局文



件中定义 `CheckBox` 控件，然后对每一个 `CheckBox` 控件通过 `setOnCheckedChangeListener()` 方法进行事件监听，并通过 `isChecked()` 方法来判断选项是否被选中，进而作出相应的事件响应。

【示例 3-7】下面是一个演示 `CheckBox` 监听事件响应的示例。

(1) 创建一个 Android 项目，在布局文件中添加 3 个 `CheckBox` 控件，并设置相应属性，代码如下。

```
<CheckBox
    android:id="@+id/checkBox1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="25sp"
    android:text="Andeoid" /> 选项内容

<CheckBox
    android:id="@+id/checkBox2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="25sp"
    android:text="Java" /> 选项内容

<CheckBox
    android:id="@+id/checkBox3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="25sp"
    android:text="C++" /> 选项内容
```

(2) 编写逻辑代码，在代码中为每一个 `CheckBox` 控件添加监听，并通过 `isChecked()` 方法判断当前选项是否被选中，然后作出相应的事件响应，代码如下。

```
checkBox1.setOnCheckedChangeListener(new OnCheckedChangeListener() {
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if (isChecked == true) {
            textView.append(checkBox1.getText() + ",");
        }
    }
});

checkBox2.setOnCheckedChangeListener(new OnCheckedChangeListener() {
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if (isChecked == true) {
            textView.append(checkBox2.getText() + ",");
        }
    }
});

checkBox3.setOnCheckedChangeListener(new OnCheckedChangeListener() {
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if (isChecked == true) {
            textView.append(checkBox3.getText() + ",");
        }
    }
});
```



(3) 运行程序并查看运行效果, 如图 3.13 所示。可以看到, 3 个选项被同时选中, 文本显示为所对应的选择内容。



图 3.13 CheckBox 控件示例



3.4 图片控件

ImageView 控件负责显示图片。图片可以是系统提供的资源文件, 也可以是 Drawable 对象或 Bitmap 对象。ImageView 控件的常用属性如表 3.4 所示。

表 3.4 ImageView 控件的常用属性及对应方法说明

属性名称	对应方法	说 明
android:adjustViewBounds	setAdjustViewBounds(boolean)	设置是否需要让 ImageView 控件调整自己的边界来保证所显示图片的比例
android:maxHeight	setMaxHeight(int)	ImageView 控件的最大高度, 可选
android:maxLength	setMaxLength(int)	ImageView 控件的最大宽度, 可选
android:scaleType	setScaleType(ImageView.ScaleType)	控制图片应如何调整或移动来适应 ImageView 控件的尺寸
android:src	setImageResource(int)	设置 ImageView 控件要显示的图片

【示例 3-8】下面通过一个示例来演示如何在界面上显示图片, 代码如下。

```
<ImageView
    android:id="@+id/imageView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

    android:adjustViewBounds="true"
    android:maxLength="300dp"
    android:maxLength="300dp"
    android:scaleType="fitXY"
    android:src="@drawable/picture" />
```

调整边界以保持图片比例

设置图片的最大高度和最大宽度分别为300dp

将原图横向拉伸后绘制

加载图片

(1) 显示 Drawable 对象, 如图 3.14 所示。



图 3.14 显示 Drawable 对象

(2) 显示系统提供的资源图片，如图 3.15 所示。在这里，只需设置 src 属性的参数值为资源图片的引用即可，代码如下。

```
<ImageView
    android:id="@+id/imageView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

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

加载图片



图 3.15 显示系统资源图片

注意：在设置 src 属性时，若加载的是 Drawable 对象，参数值为“@drawable/对象名”，若加载的是系统提供的资源图片，参数值则为“@android:drawable/图片名”，要注意二者的区别。



3.5 动画播放技术

本节要介绍的是 Android 平台的动画播放技术。动画播放技术主要有两种，分别是补间动画和帧动画。补间动画主要包括对位置、角度、尺寸、透明度等属性的变换，而帧动画则是通



过若干帧图片的轮流显示实现的。

3.5.1 补间动画

补间动画 (Tween Animation) 是指对场景里的对象不断进行图像变换来产生动画效果, 可以实现对象的旋转、平移、缩放和渐变等。

补间动画的 XML 文件位于程序的 “res/anim” 目录下, 在该 XML 文件中可以指定进行何种变换、何时进行变换以及变换将持续多长时间。当需要在 XML 文件中定义多个变换时, 应将多个变换放置在一组 `<set>` 和 `</set>` 标签中。表 3.5 列出了补间动画的变换标签及属性值说明。

表 3.5 Tween Animation 的标签及属性值说明

标签名称	属 性 值	说 明
<code><set></code>	<ul style="list-style-type: none">• <code>shareInterpolator</code>: 是否在子元素中共享插入器	可以包含其他动画变换的容器, 同时也可以包含 <code><set></code> 标签
<code><alpha></code>	<ul style="list-style-type: none">• <code>fromAlpha</code>: 变换的起始透明度• <code>toAlpha</code>: 变换的终止透明度, 取值为 0.0~1.0, 其中 0.0 代表全透明	实现透明度变换效果
<code><scale></code>	<ul style="list-style-type: none">• <code>fromXScale</code>: 起始时 X 坐标的尺寸• <code>toXScale</code>: 终止时 X 坐标的尺寸• <code>fromYScale</code>: 起始时 Y 坐标的尺寸• <code>toYScale</code>: 终止时 Y 坐标的尺寸, 其中 1.0 代表原始大小• <code>pivotX</code>: 进行尺寸变换的中心 X 坐标• <code>pivotY</code>: 进行尺寸变换的中心 Y 坐标	实现尺寸变换效果, 可以指定一个变换中心, 例如指定 <code>pivotX</code> 和 <code>pivotY</code> 为 “(0, 0)”, 则尺寸的拉伸或收缩均从左上角开始
<code><translate></code>	<ul style="list-style-type: none">• <code>fromXDelta</code>: 起始 X 坐标的位置• <code>toXDelta</code>: 终止 X 坐标的位置• <code>fromYDelta</code>: 起始 Y 坐标的位置• <code>toYDelta</code>: 终止 Y 坐标的位置	实现水平或竖直方向上的移动效果。如果属性值以 “%” 结尾, 代表相对于自身的比例; 如果属性值以 “%p” 结尾, 代表相对于父控件的比例; 如果属性值不以任何后缀结尾, 代表绝对值
<code><rotate></code>	<ul style="list-style-type: none">• <code>fromDegree</code>: 开始旋转的位置• <code>toDegree</code>: 结束旋转的位置, 以角度为单位• <code>pivotX</code>: 旋转中心点的 X 坐标• <code>pivotY</code>: 旋转中心点的 Y 坐标	实现旋转效果, 可以指定旋转定位点

下面介绍标签的一些共有属性, 如表 3.6 所示。

表 3.6 Tween Animation 标签的共有属性值说明

属 性 值	说 明
<code>duration</code>	变换持续的时间, 以毫秒为单位
<code>startOffset</code>	变换开始的时间, 以毫秒为单位
<code>repeatCount</code>	动画的重复次数
<code>interpolator</code>	为每个子标记变换设置插入器。系统已经设置了一些插入器, 可以在 <code>R.anim</code> 包中找到

【示例 3-9】下面通过一个示例来说明补间动画的使用方法。

(1) 创建一个 Android 项目, 复制程序所需图片到 “res/drawable” 目录下。



(2) 在 Eclipse 环境中开发一个定义了补间动画的 XML 文件, 本项目为 tween.xml, 位于 “res/anim” 目录下 (该目录需要手动创建)。补间动画的透明度、大小、位置和旋转变化设置如下。

- 设置透明度变化的代码如下, 效果如图 3.16 所示。

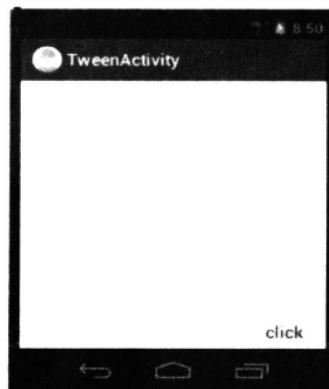
```
<?xml version="1.0" encoding="UTF-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <alpha
        android:fromAlpha="1.0" android:toAlpha="0.0"
        android:duration="10000"
    />
</set>
```



初始效果



逐渐透明



完全透明

图 3.16 透明度变化

- 设置大小变化的代码如下, 效果如图 3.17 所示。

```
<?xml version="1.0" encoding="UTF-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <scale
        android:fromXScale="1.0" android:toXScale="0.0"
        android:fromYScale="1.0" android:toYScale="0.0"
        android:pivotX="50%" android:pivotY="50%"
        android:duration="9000"
    />
</set>
```



初始效果



逐渐变小



完全消失

图 3.17 大小变化



- 设置位置变化的代码如下，效果如图 3.18 所示。

```
<?xml version="1.0" encoding="UTF-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">

    <translate

        android:fromXDelta="30" android:toXDelta="0"

        android:fromYDelta="30" android:toYDelta="0"

        android:duration="10000"

    />

</set>
```



图 3.18 位置变化

- 设置旋转变化的代码如下，效果如图 3.19 所示。

```
<?xml version="1.0" encoding="UTF-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">

    <rotate

        android:fromDegrees="0" android:toDegrees="+360"

        android:pivotX="50%" android:pivotY="50%"

        android:duration="10000"

    />

</set>
```

- (3) 编写逻辑代码，通过触发“click”按钮的监听事件来加载图片并播放动画。

```
button.setOnClickListener(new OnClickListener() {

    public void onClick(View v) {
        // TODO Auto-generated method stub

        Animation animation = AnimationUtils.loadAnimation(TweenActivity.this, R.anim.tween);
        imageView.startAnimation(animation);
    }
});
```

3.5.2 帧动画

帧动画 (Frame Animation) 就如同电影一样，通过顺序播放一系列事先加载的静态图片产生动画效果。

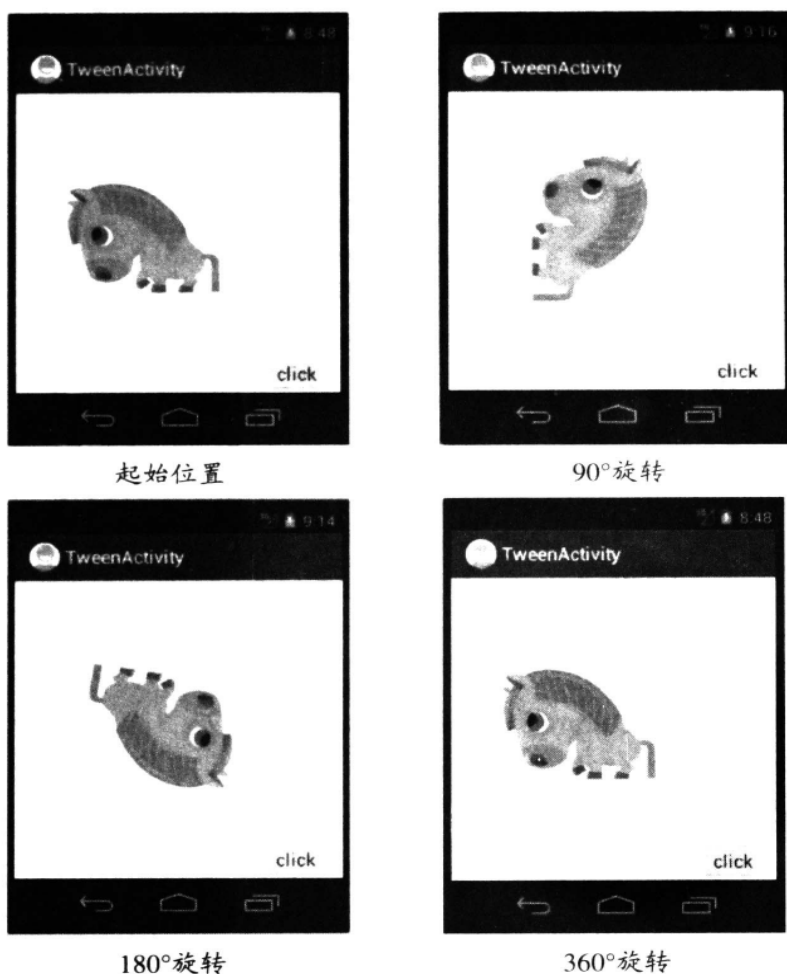


图 3.19 旋转变化

定义帧动画的 XML 文件存储在项目的“res/anim”目录下。这个 XML 文件中指定了图片帧出现的顺序以及每个帧的持续时间，而这些属性都保存在 animation-list 根节点的 item 子节点中。帧动画的 XML 文件中主要包括其需要使用的标签及属性，如表 3.7 所示。

表 3.7 Frame Animation的标签及属性值说明

标签名称	属 性 值	说 明
<animation-list>	<ul style="list-style-type: none"> • android:oneshot: 如果设置为“true”，则该动画只播放一次，并停止在最后一帧 	Frame Animation 的根标记中包含若干 <item> 标记
<item>	<ul style="list-style-type: none"> • android:drawable: 图片帧的引用。 • android:duration: 图片帧的停留时间。 • android:visible: 图片帧是否可见 	每个 <item> 标记定义了一个图片帧，其中包含图片资源的引用等属性

【示例 3-10】下面通过一个示例来讲解帧动画。

(1) 创建一个 Android 项目，复制程序所需图片到“res/drawable”目录下。

(2) 在 Eclipse 环境中开发一个定义了帧动画的 XML 文件，本项目为 frame.xml，位于“res/anim”目录下（该目录需要手动创建），其代码如下。



```
<?xml version="1.0" encoding="UTF-8"?>
<animation-list
    xmlns:android="http://schemas.android.com/apk/res/android"

    android:oneshot="false" >
    <item android:duration="500" android:drawable="@drawable/a"/>
    <item android:duration="500" android:drawable="@drawable/b"/>
    <item android:duration="500" android:drawable="@drawable/c"/>
    <item android:duration="500" android:drawable="@drawable/d"/>
    <item android:duration="500" android:drawable="@drawable/e"/>
    <item android:duration="500" android:drawable="@drawable/f"/>
    <item android:duration="500" android:drawable="@drawable/g"/>
    <item android:duration="500" android:drawable="@drawable/h"/>
    <item android:duration="500" android:drawable="@drawable/i"/>
    <item android:duration="500" android:drawable="@drawable/j"/>
    <item android:duration="500" android:drawable="@drawable/k"/>
    <item android:duration="500" android:drawable="@drawable/l"/>
    <item android:duration="500" android:drawable="@drawable/m"/>
    <item android:duration="500" android:drawable="@drawable/n"/>
</animation-list>
```

设置动画循环播放

(3) 编写布局代码，在布局中添加一个 `ImageView` 控件和一个 `Button` 控件。将 `ImageView` 控件的 `src` 属性设置为 “@anim/frame”，即加载 `frame.xml` 文件中的一系列图片，代码如下。

```
<ImageView
    android:id="@+id/imageView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="86dp"
    android:src="@anim/frame" />
```

加载一系列图片

(4) 编写如下逻辑代码，通过触发 “click” 按钮的监听事件来加载图片并播放动画。

```
button.setOnClickListener(new OnClickListener() {

    public void onClick(View v) {
        // TODO Auto-generated method stub
        imageView.setBackgroundResource(R.anim.frame);
        aDrawable =(AnimationDrawable)imageView.getBackground();
        aDrawable.start();
    }
});
```

(5) 运行程序，顺序播放一系列图片，如图 3.20 所示。



3.6 时钟控件

时钟控件是 Android 用户界面上一个比较简单的控件。时钟控件分为 `AnalogClock` 和 `DigitalClock` 两种，它们都负责显示时间。不同的是，`AnalogClock` 控件是一个模拟时钟，只显示时针和分针；而 `DigitalClock` 控件是一个数字时钟，可精确到秒。将二者结合使用，能够准确地表示时间。



图 3.20 帧动画示例

【示例 3-11】使用时钟控件时，只需在布局文件中添加 AnalogClock 控件和 DigitalClock 控件即可，代码如下，效果如图 3.21 所示。

```
<AnalogClock
    android:id="@+id/analogClock1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="104dp" />

<DigitalClock
    android:id="@+id/digitalClock1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/analogClock1"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="17dp" />
```



图 3.21 时钟控件示例



3.7 日期与时间控件

Android 平台提供了显示日期与时间的控件——DatePicker 和 TimePicker，用户可以根据自己的需要对其进行修改。

3.7.1 日期选择控件

日期选择控件（DatePicker）的主要功能是提供包含年、月、日的日期数据，并允许用户对其进行选择。

【示例 3-12】在布局文件中添加 DatePicker 控件，即可在界面中显示系统的当前日期，代码如下，效果如图 3.22 所示。

```
<DatePicker
    android:id="@+id/datePicker1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true" />
```



图 3.22 DatePicker 控件示例

如果用户需要选择日期，可以单击界面左半部分的上下日期滚动修改，也可以单击界面右半部分日历上的日期进行修改，两者的显示内容会同步变化。

3.7.2 时间选择控件

时间选择控件（TimePicker）用于显示一天中的时间，可以为 24 小时制，也可以为 AM/PM 制，允许用户自行设置。

【示例 3-13】下面介绍 TimePicker 控件的两种时间显示方式。

TimePicker 控件默认以 AM/PM 制显示时间。在布局文件中添加 TimePicker 控件，即可显示系统的当前时间，用户可以通过单击上下时间来修改显示的时间，如图 3.23 所示。



图 3.23 TimePicker 控件的 AM/PM 制显示

要想以 24 小时制显示时间，需要在逻辑代码中添加声明，还需要设置显示的时间为 0 点 0 分，具体如下，效果如图 3.24 所示。

```
public class TimePickerActivity extends Activity {
    private TimePicker timePicker;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_time_picker);
        timePicker = (TimePicker)findViewById(R.id.timePicker1);

        timePicker.setIs24HourView(true);
        timePicker.setCurrentHour(0);
        timePicker.setCurrentMinute(0);
    }
}
```

以24小时制显示时间

设置当前时间为0点0分

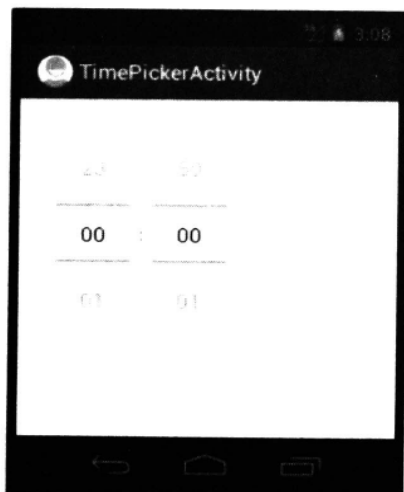


图 3.24 TimePicker 控件的 24 小时制显示



3.8 小结

本章主要介绍了 Android 平台常用的一些比较简单的控件，其中 Animation 为本章的难点，



需要读者认真学习掌握。熟悉这些控件的用法，并结合第 2 章介绍的布局知识，就能够开发出各种各样的 Android 用户界面了。



3.9 习题

1. 新建项目“TextView”。在布局中添加一个 TextView 控件，设置其字号为 30sp，颜色为蓝色，字体格式为斜体，文本内容为“<http://www.google.com.hk>”，显示为超链接样式，全部内容在一行显示，超过 TextView 控件的长度限制时省略末尾部分，如图 3.25 所示。



图 3.25 TextView 控件示例

【分析】 本题主要考查读者对 TextView 控件的掌握情况，可参考第 3.2.1 节的内容来开发程序。

【核心代码】 本题的关键代码如下。

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <TextView
        android:id="@+id/tv1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

        android:text="http://www.google.com.hk"

        android:textSize="30sp"

        android:textColor="#0000FF"

        android:textStyle="italic"

        android:autoLink="web"

        android:singleLine="true"

        android:ellipsize="end"/>

</RelativeLayout>
```



2. 新建项目“EditText”。在布局中添加3个 EditText 控件，如图 3.26 所示：第一个显示为密码格式，当用户输入密码时显示文本“请输入密码”；第二个用于显示电话号码；第三个用于将显示内容自动转换为大写形式。



图 3.26 EditText 控件示例

【分析】本题主要考查读者对 EditText 控件的掌握情况，可参考第 3.2.2 节的内容来开发程序。

【核心代码】本题的关键代码如下。

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <EditText
        android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:ems="10"
        android:inputType="textPassword"
        android:hint="请输入密码">
    </EditText>

    <EditText
        android:id="@+id/editText2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/editText1"
        android:layout_marginTop="26dp"
        android:ems="10"
        android:inputType="phone" />

    <EditText
        android:id="@+id/editText3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/editText2"
        android:layout_marginTop="44dp"
```



```
android:ems="10"  
android:capitalize="characters" />
```

```
</RelativeLayout>
```

3. 新建项目“Button”。在布局中添加两个按钮，如图 3.27 所示：Button1 在代码中绑定监听，修改标题内容为“Button1 注册成功”；Button2 在布局中通过 onClick 属性绑定监听，修改标题内容为“Button2 注册成功”。



图 3.27 Button 注册监听示例

【分析】本题主要考查读者对 Button 控件的两种注册监听方式的掌握情况，可参考第 3.3.1 节的内容来开发程序。

【核心代码】本题的关键代码如下。

- 布局文件代码如下。

```
<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:layout_alignParentLeft="true"  
    android:layout_marginTop="60dp"  
    android:onClick="click"  
    android:text="button2" />
```

- 逻辑代码如下。

```
Button button1,button2;  
@Override  
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    button1 = (Button)findViewById(R.id.button1);  
    button2 = (Button)findViewById(R.id.button2);  
  
    button1.setOnClickListener(new OnClickListener() {  
        public void onClick(View v) {
```



```
// TODO Auto-generated method stub
setTitle("Button1 注册成功");
}
});
}

public void click(View v) {
    setTitle("Button2 注册成功");
}
```

4. 新建项目“ImageButton”。复制一幅图片到项目的“res/drawable-hdpi”目录下，在布局文件中添加 ImageButton 控件，引用“res/drawable-hdpi”目录下刚刚添加的图片，如图 3.28 所示。



图 3.28 ImageButton 控件示例

【分析】本题主要考查读者对 ImageButton 控件的掌握情况，可参考第 3.3.2 节的内容来开发程序。读者应注意引用 Drawable 对象的方法。

【核心代码】本题的关键代码如下。

```
<ImageButton
    android:id="@+id/imageButton1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/p1102" />
```

5. 新建项目“RadioButton”。在布局文件中添加一对 <RadioGroup> 和 </RadioGroup> 标签，将其设置为水平显示。在 RadioGroup 控件中添加两个 RadioButton 控件，分别代表“音乐”和“舞蹈”；在 RadioButtonActivity 中为 RadioGroup 控件绑定监听，使用 TextView 控件显示用户选中的选项，如图 3.29 所示。

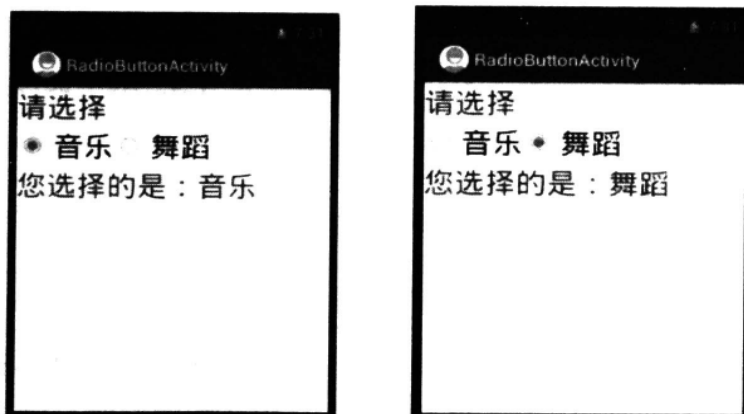


图 3.29 RadioButton 控件示例



【分析】本题主要考查读者对 `RadioButton` 控件、`RadioGroup` 控件及其监听事件的掌握情况，可参考第 3.3.4 节的内容来开发程序。

【核心代码】本题的关键代码如下。

- 布局代码如下。

```
<TextView
    android:id="@+id/tv1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="30sp"
    android:text="请选择" />

<RadioGroup
    android:id="@+id/rg1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal" >

    <RadioButton
        android:id="@+id/rb1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="30sp"
        android:text="音乐" />

    <RadioButton
        android:id="@+id/rb2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="30sp"
        android:text="舞蹈" />
</RadioGroup>

<TextView
    android:id="@+id/tv2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="30sp"
    android:text="您选择的是: " />
```

- 逻辑代码如下。

```
radioGroup.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {

    public void onCheckedChanged(RadioGroup group, int checkedId) {
        // TODO Auto-generated method stub
        if (checkedId == R.id.rb1) {

            textView.setText("您选择的是: " + radioButton1.getText());

        }else {

            textView.setText("您选择的是: " + radioButton2.getText());

        }
    }
});
```

6. 新建项目“`CheckBox`”。在布局中添加 3 个 `CheckBox` 控件，分别代表“音乐”、“舞蹈”和“旅行”；在 `CheckBoxActivity` 中为各个 `CheckBox` 控件绑定监听，使用 `TextView` 控件显示用户选中的选项，如图 3.30 所示。

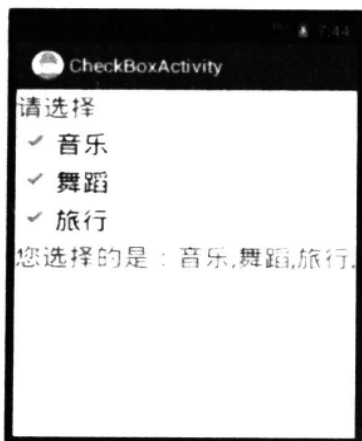


图 3.30 CheckBox 控件示例

【分析】本题主要考查读者对 CheckBox 控件及其监听事件的掌握情况，可参考第 3.3.5 节的内容来开发程序。

【核心代码】本题的关键代码如下。

- 布局代码如下。

```
<TextView
    android:id="@+id/textView1"
    android:layout_width="121dp"
    android:layout_height="wrap_content"
    android:textSize="25sp"
    android:text="请选择" />

<CheckBox
    android:id="@+id/checkbox1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="25sp"
    android:text="音乐" />

<CheckBox
    android:id="@+id/checkbox2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="25sp"
    android:text="舞蹈" />

<CheckBox
    android:id="@+id/checkbox3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="25sp"
    android:text="旅行" />

<TextView
    android:id="@+id/textview2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="25sp"
    android:text="您选择的是：" />
```

- 逻辑代码如下。

```
checkbox1.setOnCheckedChangeListener(new OnCheckedChangeListener() {
```



```

        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
            if (isChecked == true) {
                textView.append(checkBox1 .getText() + ",");
            }
        }
    });

    checkBox2.setOnCheckedChangeListener(new OnCheckedChangeListener() {

        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
            if (isChecked == true) {
                textView.append(checkBox2 .getText() + ",");
            }
        }
    });

    checkBox3.setOnCheckedChangeListener(new OnCheckedChangeListener() {

        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
            if (isChecked == true) {
                textView.append(checkBox3 .getText() + ",");
            }
        }
    });
}

```

7. 新建项目“Tween”，实现图片的旋转缩小效果，如图 3.31 所示。

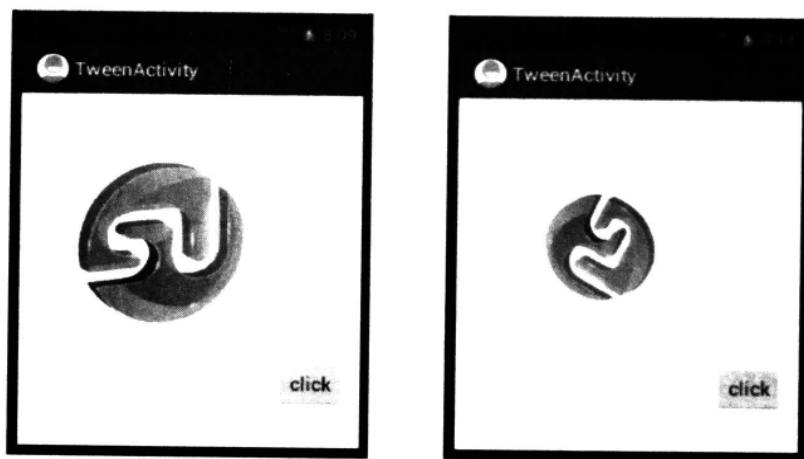


图 3.31 补间动画示例

【分析】本题主要考查读者对补间动画的掌握情况，可参考第 3.5.1 节的内容来开发程序。

【核心代码】本题的关键代码如下。

- 布局代码如下。

```

<ImageView
    android:id="@+id/imageView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:layout_marginLeft="48dp"
    android:layout_marginTop="64dp"
    android:src="@drawable/stumbleupon"/>

```



```
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:layout_below="@+id/imageView1"
    android:layout_marginRight="14dp"
    android:layout_marginTop="30dp"
    android:text="Click" />
```

- 动画代码如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">

    <rotate
        android:fromDegrees="0" android:toDegrees="+360"
        android:pivotX="50%" android:pivotY="50%"
        android:duration="10000"
    />

    <scale
        android:fromXScale="1.0" android:toXScale="0.0"
        android:fromYScale="1.0" android:toYScale="0.0"
        android:pivotX="50%" android:pivotY="50%"
        android:duration="10000"
    />

</set>
```

- 逻辑代码如下。

```
button.setOnClickListener(new OnClickListener() {
```

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

```
        // TODO Auto-generated method stub
```

```
        Animation animation = AnimationUtils.loadAnimation(TweenActivity.this,
```

```
R.anim.tween);
```

```
        imageView.startAnimation(animation);
```

```
    }
```

```
});
```

8. 新建项目“Frame”，实现多张图片连续播放的效果，如图 3.32 所示。



图 3.32 帧动画示例



【分析】本题主要考查读者对帧动画的掌握情况，可参考第 3.5.2 节的内容来开发程序。

【核心代码】本题的关键代码如下。

- 布局代码如下。

```
<ImageView
    android:id="@+id/imageView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="86dp"
    android:src="@anim/frame" />

<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_marginTop="26dp"
    android:layout_toRightOf="@+id/imageView1"
    android:text="click" />
```

- 动画代码如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<animation-list
    xmlns:android="http://schemas.android.com/apk/res/android"

    android:oneshot="false" >

    <item android:duration="500" android:drawable="@drawable/empty_sketchy"/>
    <item android:duration="500" android:drawable="@drawable/application"/>
    <item android:duration="500" android:drawable="@drawable/comment_bubble"/>
    <item android:duration="500" android:drawable="@drawable/favorite"/>

</animation-list>
```

- 逻辑代码如下。

```
button.setOnClickListener(new OnClickListener() {

    public void onClick(View v) {
        // TODO Auto-generated method stub
        imageView.setAnimation(animation);

        aDrawable.start();
    }
});
```

第4章 高级控件

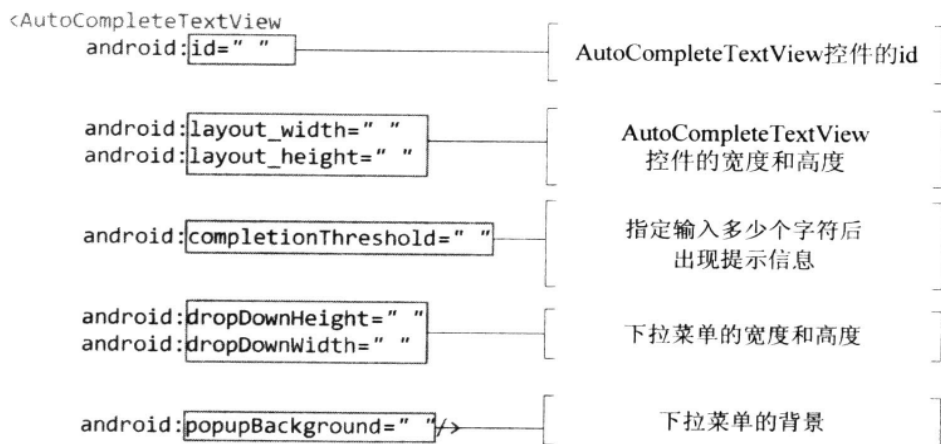
Android 的高级控件是指具有高级功能的控件，如自动完成文本类控件、进度条、列表视图、下拉列表、选项卡、网格视图等。这些控件丰富了界面的样式，强化了程序的功能，更好地实现了 Android 应用程序。本章将详细介绍这些控件。

4.1 自动完成文本类控件

Android 平台提供了两种智能输入框，分别是 `AutoCompleteTextView` 和 `MultiAutoCompleteTextView`。它们的功能大致相同，类似于在百度或者 Google 的搜索栏中输入信息时，弹出与输入信息接近的提示信息，用户单击选择所需信息自动完成文本输入的机制。

4.1.1 自动完成文本控件

自动完成文本控件（`AutoCompleteTextView`）是一个可编辑的文本视图，它能够动态匹配输入的内容，当用户输入信息时在一个下拉列表中显示提示信息，用户可以选择一项来完成输入，下拉列表中的内容是从一个数据适配器中获取的，示例如下。



对 `AutoCompleteTextView` 控件的设置，可以在 XML 文件中使用属性进行，也可以在 Java 代码中通过调用对应的方法进行。`AutoCompleteTextView` 控件常用属性与方法的对照如表 4.1 所示。



表 4.1 AutoCompleteTextView 控件的属性与方法

属性名称	对应方法	属性说明
android:completionThreshold	setThreshold(int)	定义需要用户输入的字符数
android:dropDownHeight	setDropDownHeight(int)	设置下拉菜单的高度
android:dropDownWidth	setDropDownWidth(int)	设置下拉菜单的宽度
android:popupBackground	setDropDownBackgroundResource(int)	设置下拉菜单的背景

图 4.1 是 AutoCompleteTextView 控件的语法示意图。

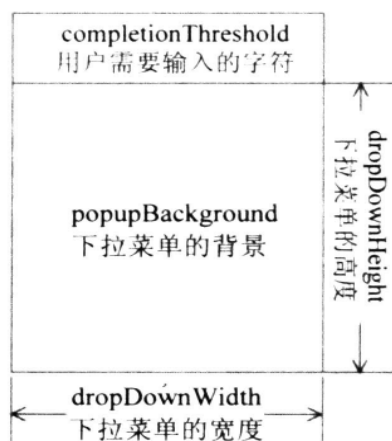


图 4.1 AutoCompleteTextView 控件的语法

【示例 4-1】下面通过一个示例来介绍 AutoCompleteTextView 控件的使用方法。

(1) 新建一个项目 “AutoCompleteTextView”，在布局文件 activity_auto_complete_text_view.xml 中添加 AutoCompleteTextView 控件，并设置其属性，代码如下。

```
<AutoCompleteTextView
    android:id="@+id/autoCompleteTextView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

    android:completionThreshold="2"
    android:dropDownHeight="100dp"
    android:dropDownWidth="200dp"
    android:popupBackground="@drawable/ic_launcher"/>
```

输入2个字符后出现
下拉菜单提示信息

下拉菜单的高度和宽度

下拉菜单的背景图片

(2) 在 AutoCompleteTextViewActivity 中声明一个数组和一个数组适配器，该适配器负责把数组内容提供给下拉菜单作为提示信息显示，代码如下。

```
private String[] str = new String[]{"aaa", "aab", "aaf", "bbb", "bbr", "ccc"};
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
    android.R.layout.simple_list_item_1, str);
```

(3) 为 AutoCompleteTextView 控件绑定适配器，代码如下。

```
autoCompleteTextView.setAdapter(adapter);
```

(4) 运行程序，效果如图 4.2 所示。



图 4.2 AutoCompleteTextView 控件示例

当用户连续输入两个“a”之后，会出现提示信息下拉菜单，用户可以根据需要进行选择，以完成文本的自动输入。

4.1.2 多文本自动完成输入控件

多文本自动完成输入控件（MultiAutoCompleteTextView）也是一个可编辑的文本视图，能够对用户输入的文本进行有效的扩展提示，而不需要用户输入完整的内容，但用户必须提供一个 MultiAutoCompleteTextView.Tokenizer API 来区分不同的子串，示例如下。与 AutoCompleteTextView 控件不同的是，可以向 MultiAutoCompleteTextView 控件的输入框中增加选择值。

<MultiAutoCompleteTextView android:id="@+id/..."	MultiAutoCompleteTextView 控件的id
android:layout_width="wrap_content" android:layout_height="wrap_content"	MultiAutoCompleteTextView 控件的宽度和高度
android:completionThreshold="1"	指定输入多少个字符后 出现提示信息
android:dropDownHeight="100dp" android:dropDownWidth="100dp"	下拉菜单的高度和宽度
android:popupBackground="@drawable/..."	下拉菜单的背景

对 MultiAutoCompleteTextView 控件的设置可以在 XML 文件中使用属性进行，也可以在 Java 代码中通过方法进行，其常用属性与方法的对照如表 4.2 所示。

表 4.2 MultiAutoCompleteTextView 控件的属性与方法

属性名称	对应方法	属性说明
android:completionThreshold	setThreshold(int)	定义需要用户输入的字符数
android:dropDownHeight	setDropDownHeight(int)	设置下拉菜单的高度
android:dropDownWidth	setDropDownWidth(int)	设置下拉菜单的宽度
android:popupBackground	setDropDownBackgroundResource(int)	设置下拉菜单的背景

图 4.3 是 MultiAutoComplete TextView 控件的语法示意图。

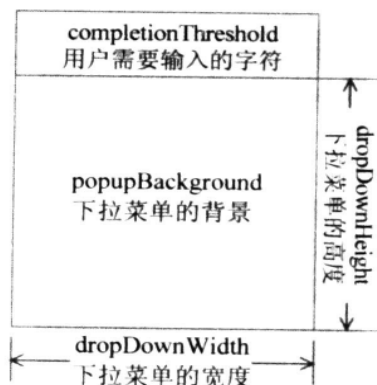


图 4.3 MultiAutoCompleteTextView 控件的语法

【示例 4-2】下面来看看 MultiAutoCompleteTextView 控件与 AutoCompleteTextView 控件的区别。

(1) 新建项目 “MultiAutoCompleteTextView”，在布局文件 activity_multi_auto_complete_text_view.xml 中添加 MultiAutoCompleteTextView 控件，并设置相应属性，代码如下。

```
<MultiAutoCompleteTextView
    android:id="@+id/multiAutoCompleteTextView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:dropDownHeight="100dp"
    android:dropDownWidth="200dp"/>
```

下拉菜单的高度和宽度

(2) 在 MultiAutoCompleteTextViewAcitivity 中声明一个数组和一个数组适配器，为下拉菜单提供提示信息，代码如下。

```
private String[] str = new String[]{"aaa", "aab", "aaf", "bbb", "bbr", "ccc"};
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
    android.R.layout.simple_list_item_1, str);
```

(3) 为 MultiAutoCompleteTextView 控件绑定适配器，并设置 Tokenizer 来区分不同的子串，代码如下。

```
mAutoCompleteTextView.setAdapter(adapter);
mAutoCompleteTextView.setTokenizer(new MultiAutoCompleteTextView.CommaTokenizer());
```

(4) 运行程序，如图 4.4 所示，MultiAutoCompleteTextView 控件在输入框中一直增加选择值，以多次进行文本自动输入，而 AutoCompleteTextView 控件只显示一次提示。



图 4.4 MultiAutoCompleteTextView 控件示例



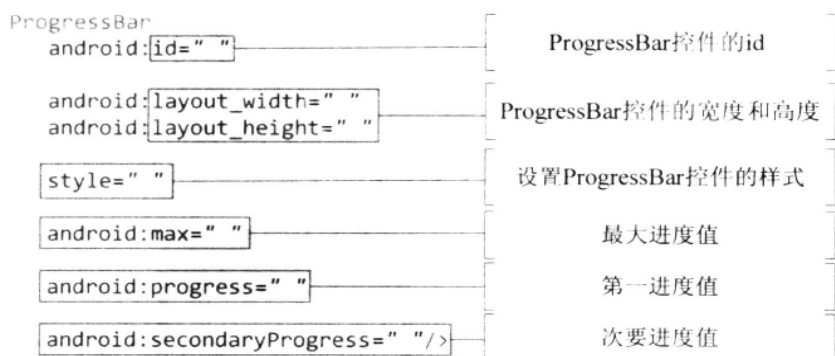
4.2 进度条与拖动条

本节将对 Android 中的进度条和拖动条进行介绍。拖动条主要用于完成与用户的简单交互，而进度条则用于在需要长时间加载某些资源时为用户显示加载的进度。

4.2.1 进度条

进度条（ProgressBar）是某些操作的进度可视指示器，用于呈现操作的进度。还有一个次要进度条用来显示中间进度，如流媒体播放缓冲区的进度。

一个进度条也可不确定其进度。在不确定模式下，进度条显示循环动画。这种模式常用于任务长度未知的情况，代码如下。



ProgressBar 控件的语法如图 4.5 所示。

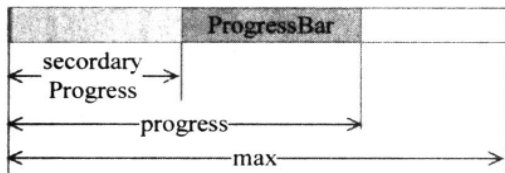
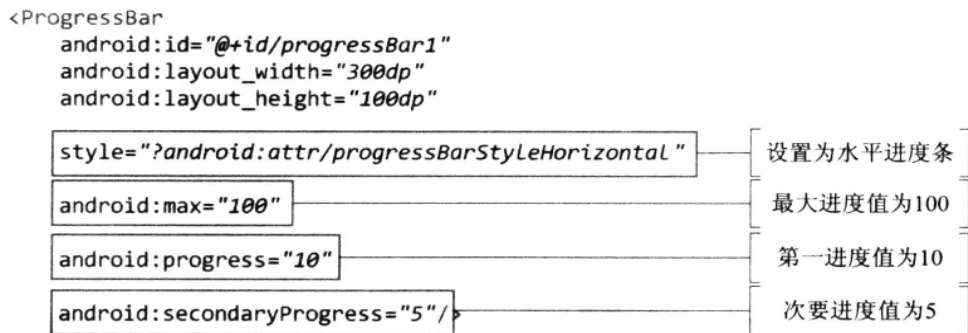


图 4.5 ProgressBar 控件的语法

【示例 4-3】下面介绍 ProgressBar 控件的使用方法。

(1) 新建一个项目“ProgressBar”，在布局文件 activity_progress_bar.xml 中添加 ProgressBar 控件，并设置其相应属性，代码如下。



注意：除了有水平进度条，还有圆形进度条。Android 为用户提供了 Large、Normal 和 Small 共 3 种规格的圆形进度条。



(2) 在 `ProgressBarActivity` 中启动一个线程。该线程定时修改进度值，每隔 1 秒，进度值增加 10，直至增加到进度条的最大值 100 为止，代码如下。

```
new Thread(new Runnable() {
    public void run() {
        // TODO Auto-generated method stub
        while (mProgressStatus < 100) {
            handler.post(new Runnable() {
                public void run() {
                    progressBar.setProgress(mProgressStatus);
                }
            });
            mProgressStatus += 10;
            try {
                new Thread().sleep(1000);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
}).start();
```

(3) 运行程序，如图 4.6 所示。

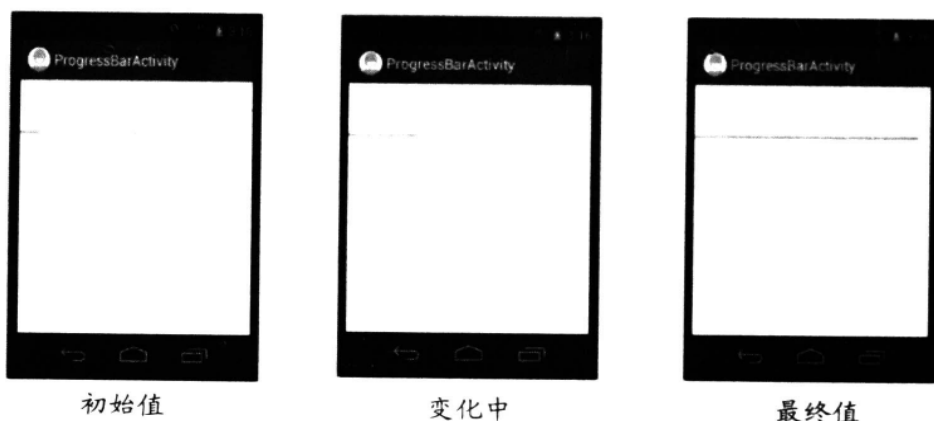


图 4.6 ProgressBar 控件示例

4.2.2 拖动条

拖动条 (SeekBar) 就是添加了滑块的进度条。用户可以通过拖动滑块来调节当前进度，例如调节电影的播放进度或者音量的大小。为了让程序能响应拖动条滑块位置的改变，可以考虑为它绑定一个 `OnSeekBarChangeListener` 监听器。

<code><SeekBar</code>	
<code>android:id="@+id/seek_bar"</code>	SeekBar 控件的 id
<code>android:layout_width="match_parent"</code>	SeekBar 控件的宽度和高度
<code>android:layout_height="wrap_content"</code>	
<code>android:thumb="@drawable/seek_bar_thumb"</code>	滑块样式
<code>android:max="100"</code>	SeekBar 控件的最大值
<code>android:progress="0"</code>	SeekBar 控件的当前进度值
<code>></code>	

对拖动条的设置可以在 XML 文件中使用属性进行，常用属性如表 4.3 所示。



表 4.3 SeekBar控件的常用属性

属性名称	属性说明
android:thumb	设置滑块样式
android:max	设置拖动条进度的最大值
android:progress	设置拖动条的当前进度值

【示例 4-4】下面给出一个示例，用 TextView 控件显示拖动条的进度变化，演示拖动条的使用。

(1) 新建一个项目“SeekBar”。在布局文件 activity_seek_bar.xml 中添加一个 SeekBar 控件和一个 TextView 控件，并设置其属性，代码如下。

```
<SeekBar
    android:id="@+id/seekBar1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"

    android:thumb="@android:drawable/btn_star_big_on"
    android:max="100"
    android:progress="10"
/>
<TextView..... />
```

滑块显示为亮着的星形

拖动条最大值为100

拖动条当前进度值为10

注意：改变滑块的外观，实际上就是将滑块显示为 ImageView，其参数设置方法与 ImageView 控件的 src 属性设置方法完全相同。

(2) 在 SeekBarActivity 中为 SeekBar 控件添加监听 OnSeekBarChangeListener。向左拖动滑块时，进度值变小，最小值为 0；向右拖动滑块时，进度值变大，最大值为 100。TextView 控件用于显示当前进度值，代码如下。

```
seekBar.setOnSeekBarChangeListener(new OnSeekBarChangeListener() {

    public void onStopTrackingTouch(SeekBar seekBar) {
        // TODO Auto-generated method stub
    }

    public void onStartTrackingTouch(SeekBar seekBar) {
        // TODO Auto-generated method stub
    }

    public void onProgressChanged(SeekBar seekBar, int progress,
        boolean fromUser) {
        // TODO Auto-generated method stub
        textView.setText("当前进度值: " + progress);
    }
});
```

(3) 运行程序，如图 4.7 所示。



图 4.7 SeekBar 控件示例



4.3 评分条

评分条 (RatingBar) 是拖动条和进度条的扩展。它用星形图标来显示等级评定，默认显示 5 颗星，用户可以通过点击触屏或者左右移动轨迹球来进行等级评定。

RatingBar 控件有 3 种风格，分别是 RatingBarStyle (默认风格)、RatingBarStyleSmall (小风格) 和 RatingBarStyleIndicator (大风格)。其中，默认风格的 RatingBar 是用户通常使用的可以交互的评分条，而其他两种风格的评分条不能进行交互，只能作为指示牌。

设置 RatingBar 样式的方法是在布局文件的 RatingBar 控件内设置风格，代码如下。

```
style="?android:attr/ratingBarStyle"  
style="?android:attr/ratingBarStyleSmall"  
style="?android:attr/ratingBarStyleIndicator"
```

RatingBar 控件的效果如图 4.8 所示。



图 4.8 RatingBar 控件



对评分条的设置，可以在布局文件中使用属性进行，代码如下。

<code><RatingBar</code>	
<code>android:id=" " "</code>	RatingBar控件的id
<code>style=" " "</code>	RatingBar控件的样式
<code>android:layout_width=" " "</code> <code>android:layout_height=" " "</code>	RatingBar控件的宽度和高度
<code>android:numStars=" " "</code>	显示的星形数量
<code>android:isIndicator=" " "</code>	是否显示为指示器
<code>android:rating=" " "</code>	默认评分
<code>android:stepSize=" " "/></code>	评分步长

RatingBar 控件的常用属性如表 4.4 所示。

表 4.4 RatingBar控件的常用属性

属性名称	属性说明
android:isIndicator	RatingBar 控件是否是一个指示器（值为“true”时，用户无法进行更改）
android:numStars	显示的星形数量，必须是整型
android:rating	默认的评分，必须是浮点型
android:stepSize	评分的步长，即一次增加或者减少的星形数量是这个数字的整数倍，必须是浮点型

【示例 4-5】下面看一个具体示例。用 TextView 控件显示评分变化，演示 RatingBar 控件的使用。

(1) 新建一个项目“RatingBar”。在布局文件 activity_rating_bar.xml 中添加一个 RatingBar 控件和一个 TextView 控件，并设置其相应属性，代码如下。

<code><RatingBar</code>	
<code>android:id="@+id/ratingBar1"</code>	RatingBar控件的id
<code>style="?android:attr/ratingBarStyle"</code>	设置为默认风格
<code>android:layout_width="wrap_content"</code> <code>android:layout_height="wrap_content"</code>	RatingBar控件的宽度和高度
<code>android:numStars="5"</code>	星形的数量为5
<code>android:isIndicator="false"</code>	不显示为指示器
<code>android:rating="4.5"</code>	默认评分为4.5
<code>android:stepSize="0.5"/></code>	评分步长为0.5
<code><TextView...../></code>	

(2) 在 RatingBarActivity 中为 RatingBar 控件添加 OnRatingBarChangeListener() 方法。向左拖动时，评分值减小，最小值为 0；向右拖动时，评分值增大，最大值为 5，TextView 控件用来显示当前分数，代码如下。

```
ratingBar.setOnRatingBarChangeListener(new OnRatingBarChangeListener() {
    public void onRatingChanged(RatingBar ratingBar, float rating,
        boolean fromUser) {
        // TODO Auto-generated method stub
    }
})
```



```
        textView.setText("受欢迎度为: " + rating + "颗星");  
    }  
});
```

(3) 运行程序，效果如图 4.9 所示。



图 4.9 RatingBar 控件示例



4.4 滚动视图

滚动视图 (ScrollView) 是在一屏不能完全显示所有需要显示的信息时使用的控件。它支持垂直滚动，当一屏无法显示其中所包含的所有控件或信息时，便会自动添加滚动功能以显示更多内容。

ScrollView 控件的使用非常简单。由于 ScrollView 实质上是一种帧布局，因此它的使用方法与布局的使用方法完全一致，示例如下。但是，因为 ScrollView 只能拥有一个直接子类，所以在使用 ScrollView 控件时，需要将其他布局嵌套在 ScrollView 之内，常用的子元素是垂直方向的 LinearLayout。

```
<ScrollView  
    xmlns:android="http://schemas.android.com/  
apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
  
    android:id="@+id/scrollView"   
  
    android:layout_width="match_parent"  
    android:layout_height="match_parent">  
</ScrollView>
```

命名空间

ScrollView控件的id

ScrollView控件的宽度和高度

【示例 4-6】下面看一个 ScrollView 控件的应用示例。

(1) 新建一个项目“ScrollView”，将默认布局 RelativeLayout 修改为 LinearLayout (vertical)，并将 ScrollView 放在垂直线性布局之外，代码如下。



```

<ScrollView
    xmlns:android="http://schemas.android.com/
    apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/scollView1"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        xmlns:android="http://schemas.android.com/
        apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        android:id="@+id/LinearLayout1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        .....
    </LinearLayout>
</ScrollView>

```

命名空间

垂直线性布局

(2) 由于本示例不涉及逻辑代码，故可直接运行程序以查看效果，如图 4.10 所示。



图 4.10 ScollView 控件示例

从图 4.10 中可以看出，布局中添加了 10 个 Button 控件，运行的初始效果是在屏幕上只显示 6 个 Button 控件，拖动滚动条将屏幕向上滚动，便可显示剩余 4 个 Button 控件。



4.5 列表视图

列表视图 (ListView) 是将数据显示在一个垂直且可滚动的列表中的一种控件，数据来源于与其绑定的 ListAdapter，包括图片、文本等，每一行数据为一条 item，示例如下。

```

<ListView
    android:id="@+id/"
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:divider="@android:drawable/divider_horizontal_textview"
    android:dividerHeight="10dp"
/>

```

ListView控件的id

ListView控件的高度和宽度

分割线的样式

分割线的高度

ListView 控件的常用属性如表 4.5 所示。



表 4.5 ListView控件的常用属性

属性名称	属性说明
android:divider	每条 item 之间的分割线，参数值可以是一个 Drawable 对象，也可以是一个色值
android:dividerHeight	分割线的高度

【示例 4-7】模拟一个 QQ 通讯录，每一条 item 显示一张图片、一个用户名和对应的 QQ 号。

(1) 新建一个项目“ListView”，在布局文件 activity_list_view.xml 中添加一个 ListView 控件，并设置相应属性，代码如下。

```
<ListView
    android:id="@+id/users"
    android:layout_height="wrap_content"
    android:layout_width="fill_parent"
    android:divider="#87CEFF"
    android:dividerHeight="2dp"
/>
```

Diagram annotations for the ListView code:

- `android:id="@+id/users"` points to "ListView控件的id"
- `android:layout_height="wrap_content"` and `android:layout_width="fill_parent"` point to "ListView控件的宽度填充父容器高度并环绕内容"
- `android:divider="#87CEFF"` points to "分割线样式为蓝色"
- `android:dividerHeight="2dp"` points to "分割线高度为2dp"

(2) 在“res/layout”目录下新建一个布局文件 another_layout.xml。将该布局文件作为 ListAdapter 的参数，指定每一条 item 的显示样式，代码如下。

```
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="fill_parent"
    android:layout_width="fill_parent"
    android:orientation="vertical"
    android:stretchColumns="1"
>

<TableRow>
    <ImageView
        android:layout_width="wrap_content"
        android:id="@+id/imageView1"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_launcher"
    </ImageView>

    <TextView
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:id="@+id/name"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
    </TextView>

    <TextView
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:id="@+id/qq"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
    </TextView>
</TableRow>
</TableLayout>
```

Diagram annotations for the TableRow code:

- The `<ImageView>` block points to "item中的图片"
- The `<TextView>` block with `android:id="@+id/name"` points to "item中的用户名"
- The `<TextView>` block with `android:id="@+id/qq"` points to "item中用户的QQ号"

由代码可知，每一个 item 为一个一行三列的表格，第一列是用户图片，第二列是用户名，第三列是用户 QQ 号。

(3) 在 ListViewActivity 中声明一个 Map 类型的 List 以及一个包含用户名和用户 QQ 号的数组，代码如下。

```
private List<Map<String, Object>> sList = new ArrayList<Map<String, Object>>();
private String name[] = {
    "Ricky",
    "Daisy",
    "Seven",
    "Fever",
}
```



```

        "Mike",
        "Rose",
        "Jack",
        "Tom"
    };
    private String qq[] = {
        "1553518517",
        "1881044874",
        "132946223",
        "155360824",
        "1553682340",
        "1372890765",
        "151368842",
        "1873462872",
    };
};

```

(4) 利用 for 循环，将 item 的内容加载到 List 中，代码如下。

```

for (int i = 0; i < name.length; i++) {
    Map<String, Object> map = new HashMap<String, Object>();
    map.put("userPic", R.drawable.ic_launcher);
    map.put("userName", name[i]);
    map.put("userQicq", qq[i]);
    sList.add(map);
}

```

(5) 声明适配器，并将其绑定到 ListView 控件，代码如下。

```

ListAdapter listAdapter = new SimpleAdapter(this, sList, R.layout.another_layout,
    new String [] {"userPic", "userName", "userQicq"},
    new int [] {R.id.imageView1, R.id.name, R.id.qq}
);

((ListView)findViewById(R.id.users)).setAdapter(listAdapter);
}

```

(6) 运行程序以查看效果，如图 4.11 所示。

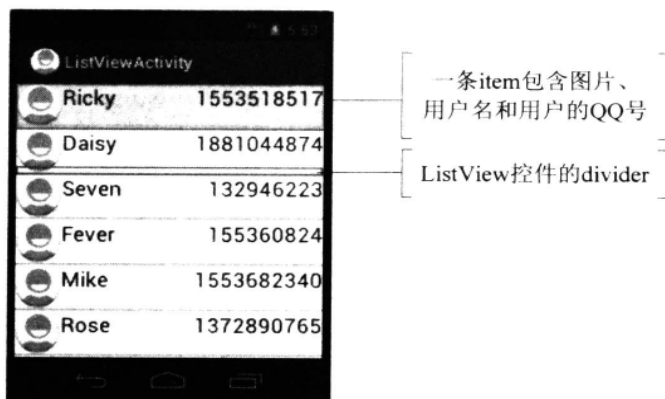


图 4.11 ListView 控件示例



4.6 下拉列表

下拉列表 (Spinner) 每次只显示用户选中的元素，当用户再次点击时会弹出选择列表供用户选择，而选择列表中的元素同样来自适配器，示例如下。



<code><Spinner</code>		
<code>android:id="@+id/"</code>	<code>" "</code>	Spinner控件的id
<code>android:layout_width=" "</code>	<code>" "</code>	Spinner控件的宽度和高度
<code>android:layout_height=" "</code>	<code>" "</code>	
<code>android:spinnerMode=" "</code>	<code>" "</code>	Spinner控件的样式
<code>android:dropDownWidth=" "</code>	<code>" "</code>	Spinner控件的宽度
<code>android:dropDownVerticalOffset=" "</code>	<code>" "</code>	Spinner控件的垂直偏移
<code>android:dropDownHorizontalOffset=" "</code>	<code>" "</code>	Spinner控件的水平偏移
<code>android:popupBackground=" "</code>	<code>" "</code>	Spinner控件的背景

对 Spinner 控件的设置可以在 XML 文件中使用属性进行,也可以在 Java 代码中通过方法进行。Spinner 控件的常用属性与方法如表 4.6 所示。

表 4.6 Spinner 控件的常用属性及对应方法说明

属性名称	对应方法	属性说明
android:spinnerMode		设置 Spinner 控件的样式,有 dropdown 和 dialog 两种
android:dropDownVerticalOffset	setDropDownVerticalOffset(int)	设置 Spinner 控件的水平偏移
android:dropDownHorizontalOffset	setDropDownHorizontalOffset(int)	设置 Spinner 控件的垂直偏移
android:dropDownWidth	setDropDownWidth(int)	设置 Spinner 控件的宽度
android:popupBackground	setPopupBackgroundResource()	设置 Spinner 控件的背景

【示例 4-8】下面给出一个示例,演示学生信息下拉列表。

(1) 新建一个项目“Spinner”,在布局文件 activity_spinner.xml 中添加 Spinner 控件并设置相应属性,代码如下。

<code><Spinner</code>		
<code>android:id="@+id/spinner1"</code>		Spinner控件的id
<code>android:layout_width="fill_parent"</code>		用宽度填充父容器的高度并环绕内容
<code>android:layout_height="wrap_content"</code>		
<code>android:spinnerMode="dropdown"</code>		设置为下拉样式
<code>android:dropDownWidth="wrap_content"</code>		Spinner控件的宽度环绕内容
<code>android:dropDownVerticalOffset="20dp"</code>		Spinner控件垂直偏移20dp
<code>android:dropDownHorizontalOffset="10dp"</code>		Spinner控件水平偏移10dp
<code>android:popupBackground="@drawable/ic_action_search"/></code>		Spinner控件的背景为 ic_action_search.png

(2) 在 SpinnerActivity 中声明一个学生数组和一个 List,代码如下。

```
private String students[]={
    "Ricky",
    "Daisy",
    "Seven",
    "Bug",
    "Fever"};

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



(3) 利用 for 循环，将下拉菜单的显示内容加载到 List 中，代码如下。

```
for (int i = 0; i < students.length; i++) {
    list.add(students[i]);
}
```

(4) 声明一个 ArrayAdapter，并将其绑定到 Spinner 控件，代码如下。

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>
(this, android.R.layout.simple_spinner_dropdown_item, list);

spinner.setAdapter(adapter);
```

(5) 运行程序以查看效果，如图 4.12 所示。

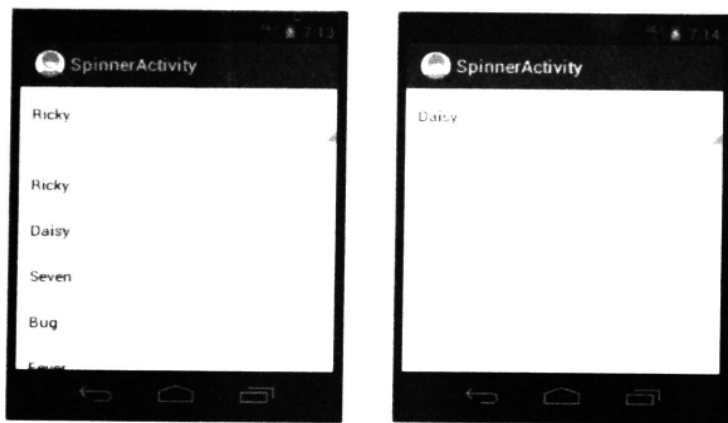


图 4.12 Spinner 控件示例

4.7 选项卡

选项卡（TabHost）控件可以实现多个标签页的效果，用户单击每个选项卡即可打开其对应的标签页。

TabHost 控件是整个 Tab 的容器，包括 TabWidget 和 FrameLayout 两部分。TabWidget 是 Tab 的标签，FrameLayout 则是 Tab 的内容，示例如下。

```
<TabHost
    xmlns:android="http://
schemas.android.com/apk/res/android"
    android:layout_width=" "
    android:layout_height=" "
    android:id="@android:id/tabhost"
>

<TabWidget
    android:id="@android:id/tabs"
    android:layout_width=" "
    android:layout_height=" " >
</TabWidget>

<FrameLayout
    ..... >
</FrameLayout>

</TabHost>
```

命名空间

TabHost控件的宽度和高度

TabHost控件的固定id

Tab的固定id

Tab的内容



【示例 4-9】下面通过一个示例来讲解 TabHost 控件的使用。

(1) 新建一个项目“TabHost”。首先将默认布局 RelativeLayout 修改为 TabHost，然后并列添加 TabWidget 标签和 FrameLayout 布局，TabWidget 用于显示 Tab 的标签，FrameLayout 用于显示 Tab 的内容，代码如下。

```
<TabHost
    xmlns:android="http://schemas.android.com/
    apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:id="@android:id/tabhost"
>

<TabWidget
    android:id="@android:id/tabs"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" >
</TabWidget>
<FrameLayout
    android:id="@android:id/tabcontent"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_weight="1" >

    <TextView
        android:id="@+id/tv11"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="chicken"
        android:textSize="11pt" />

    <TextView
        android:id="@+id/tv22"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="fish "
        android:textSize="11pt" />
</FrameLayout>

</TabHost>
```

(2) 在 TabHostActivity 中调用 Tabhost.setup() 方法。首先添加 Tab 上显示的图片，然后设置当前显示的选项卡，代码如下。

```
TabHost tabhost = (TabHost)findViewById(android.R.id.tabhost);
tabhost.setup();

tabhost.addTab(tabhost.newTabSpec("tab1")

    .setIndicator(null , getResources().getDrawable(R.drawable.xiaoji))
    .setContent(R.id.tv11));

tabhost.addTab(tabhost.newTabSpec("tab2")

    .setIndicator(null , getResources().getDrawable(R.drawable.xiaoyu))
    .setContent(R.id.tv22));

tabhost.setCurrentTab(1);
```



(3) 运行程序以查看效果, 如图 4.13 所示。

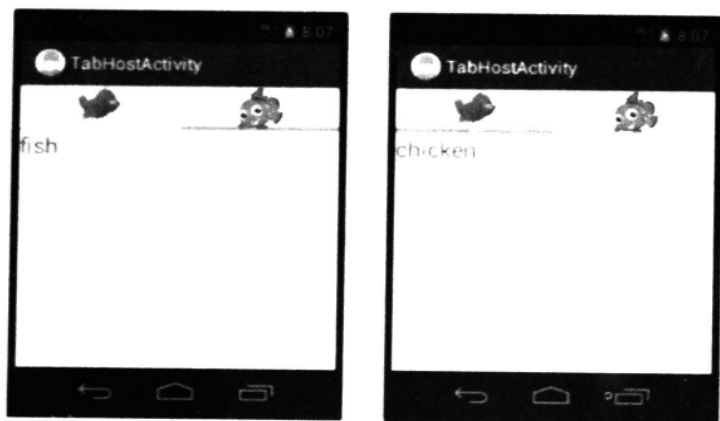


图 4.13 TabHost 控件示例

4.8 页面滑动切换控件

Android 的左右滑动功能在实际编程中经常使用, 如查看多张图片、左右切换 Tab 等。早期的通用做法是使用 ViewFlipper 控件, Android 3.0 之后的 SDK 提供了 android-support-v4.jar 包以实现版本兼容, 使在旧版本系统下开发的应用可以通过加入 jar 包实现扩展, 其中就有一个用于实现左右滑动的类——ViewPager。

ViewPager 是 android-support-v4.jar 包中的一个系统控件, 它继承自 ViewGroup 类, 专门用于实现左右滑动切换, 示例如下。只要右击项目, 在弹出的快捷菜单中依次单击“Android Tools”、“Add Support Library”选项, 加入这个 jar 包, 就可以使用 ViewPager 类了。

```
<android.support.v4.view.ViewPager
    android:id="@+id/vp"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center" />
<android.support.v4.view.PagerTitleStrip
    android:id="@+id/ptstrip"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="bottom" />
</android.support.v4.view.ViewPager>
```

ViewPager 控件的 id
ViewPager 控件的宽度和高度
ViewPager 控件的位置
PagerTitleStrip 的 id
PagerTitleStrip 的宽度和高度
PagerTitleStrip 的位置

- `<android.support.v4.view.ViewPager>.....</android.support.v4.view.ViewPager>`: 表示引入 android-support-v4.jar 包中的多页显示控件 ViewPager。
- `<android.support.v4.view.PagerTitleStrip>.....</>`: 表示引入 android-support-v4.jar 包中的 PagerTitleStrip 类, 用于显示当前页面的标题。

【示例 4-10】下面演示如何使用 ViewPager 控件实现页面的左右滑动切换。

(1) 新建一个项目“ViewPager”, 在布局文件 activity_view_pager.xml 中添加 ViewPager



控件，并设置其相应属性，代码如下。

```

<android.support.v4.view.ViewPager
    android:id="@+id/viewpager1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center"
<android.support.v4.view.PagerTitleStrip
    android:id="@+id/pagertitle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="top"
</android.support.v4.view.ViewPager>

```

ViewPager控件的id

宽度和高度分别与父容器匹配

ViewPager控件居中

PagerTitleStrip的id

宽度和高度分别与父容器匹配

PagerTitleStrip 居于顶部

(2) 在“res/layout”目录下新建两个布局文件 view1.xml 和 view3.xml，在每个文件中添加一个 ImageView 控件，并设置图片居中，用于显示不同界面的不同图片，代码如下。

```

<ImageView
    android:id="@+id/imageView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:src="@drawable/a" />
<ImageView
    android:id="@+id/imageView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:src="@drawable/b" />

```

(3) 在 ViewPagerActivity 中声明一个布局解析器 LayoutInflater，解析以上创建的两个布局文件，并将其加载到对应的界面中，代码如下。

```

LayoutInflater mLi = LayoutInflater.from(this);
View view1 = mLi.inflate(R.layout.view1, null);
View view2 = mLi.inflate(R.layout.view2, null);

final ArrayList<View> views = new ArrayList<View>();
views.add(view1);
views.add(view2);

```

(4) 声明 ViewPager 的适配器 PagerAdapter，并将其绑定到 ViewPager 控件，代码如下。

```

PagerAdapter mPagerAdapter = new PagerAdapter() {

    public boolean isViewFromObject(View arg0, Object arg1) {
        return arg0 == arg1;
    }

    public int getCount() {
        return views.size();
    }

    public void destroyItem(View container, int position, Object object) {

```



```

        ((ViewPager)container).removeView(views.get(position));
    }

    public CharSequence getPageTitle(int position) {
        return titles.get(position);
    }

    public Object instantiateItem(View container, int position) {
        ((ViewPager)container).addView(views.get(position));
        return views.get(position);
    }
};

viewPager.setAdapter(mPagerAdapter);

```

(5) 运行程序以查看效果, 如图 4.14 所示。

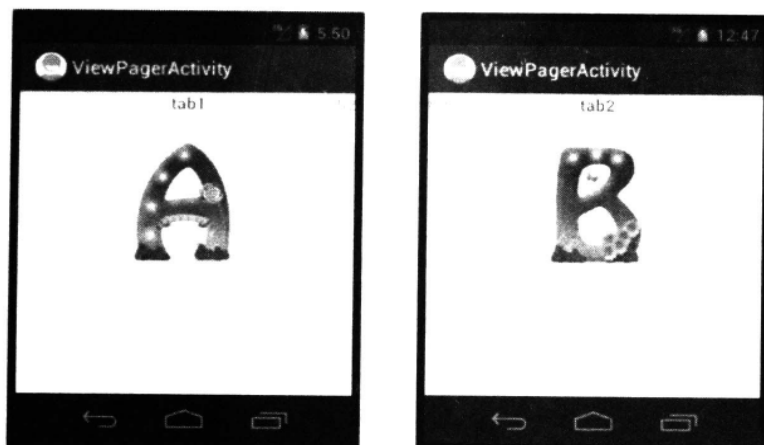


图 4.14 ViewPager 控件示例



4.9 图片切换控件

要想在 Windows 平台上查看多张图片, 最简单的方法就是通过“Windows 图片和传真查看器”在“下一张”和“上一张”之间切换。在 Android 平台上, 可以通过 ImageSwitcher 类来实现这一效果, 示例如下。

<ImageSwitcher		
android:id="@+id/..."	ImageSwitcher控件的id	
android:layout_width="wrap_content"	ImageSwitcher 控件的宽度和高度	
android:layout_height="wrap_content"		
android:animateFirstView="true"	首次显示时是否对当前视图应用动画	
android:inAnimation="@anim/..."	标识显示视图时使用的动画	
android:outAnimation="@anim/..."	标识隐藏视图时使用的动画	
</ImageSwitcher>		



对 `ImageSwitcher` 类，必须设置一个 `ViewFactory` 将显示的图片 and 父窗口区分开来，因此需要实现 `ViewSwitcher.ViewFactory` 接口。通过 `makeView()` 方法显示图片会返回一个 `ImageView` 对象。`setImageResource()` 方法用于指定图片资源。`ImageSwitcher` 控件的常用属性如表 4.7 所示。

表 4.7 `ImageSwitcher` 控件的常用属性说明

属性名称	属性说明
<code>android:animateFirstView</code>	定义 <code>ViewAnimation</code> 首次显示时是否对当前视图应用动画
<code>android:inAnimation</code>	标识显示视图时使用的动画
<code>android:outAnimation</code>	标识隐藏视图时使用的动画

【示例 4-11】下面用 `ImageSwitcher` 类来模拟实现一个图片查看器。

(1) 新建一个项目 “`ImageSwitcher`”，在布局文件 `activity_image_switcher.xml` 中添加一个 `ImageSwitcher` 控件和两个 `Button` 控件，并设置相应属性，代码如下。

```
<ImageSwitcher
    android:id="@+id/imageSwitcher1"
    android:layout_width="200px"
    android:layout_height="200px"
    android:animateFirstView="true"
    android:inAnimation="@android:anim/fade_in"
    android:outAnimation="@android:anim/fade_out">
</ImageSwitcher>
<Button
    .....
    android:text="下一张" />
<Button
    .....
    android:text="上一张" />
```

ImageSwitcher 控件的 id
ImageSwitcher 控件的宽度和高度均为 200px
首次显示时对当前视图应用动画
动画淡入效果
动画淡出效果
单击显示下一张图片
单击显示上一张图片

(2) 在 `ImageSwitcherActivity` 中声明一个存放图片 id 的 `int` 型一维数组和一个表示数组下标的变量，代码如下。

```
private int[] imageIds = {
    R.drawable.baiyang, R.drawable.chunv, R.drawable.jinniu,
    R.drawable.juxie, R.drawable.mojie, R.drawable.sheshou,
    R.drawable.shizi, R.drawable.shuangyu, R.drawable.shuangzi,
    R.drawable.shuiping, R.drawable.tiancheng, R.drawable.tianxie,};
private int index=0;
```

(3) 设置一个 `ViewFactory` 以实现 `ViewSwitcher.ViewFactory` 接口，调用 `View.makeView()` 方法设置显示图片并返回该 `ImageView`，`setImageResource` 用于指定图片资源，代码如下。

```
imageSwitcher.setFactory(new ViewFactory() {

    public View makeView() {
        ImageView imageView = new ImageView(ImageSwitcherActivity.this);
        imageView.setBackgroundColor(0xff0000);
```



```

        imageView.setScaleType(ImageView.ScaleType.FIT_CENTER);
        imageView.setLayoutParams(new ImageSwitcher.LayoutParams(200,200));
        return imageView;
    }
});
imageSwitcher.setBackgroundResource(imageIds[index]);

```

(4) 分别为“上一张”和“下一张” Button 控件绑定 Click 监听，单击 Button 控件触发监听事件，通过数组下标的变化实现图片的切换，代码如下。

```

buttonNext.setOnClickListener(new OnClickListener() {

    public void onClick(View v) {
        // TODO Auto-generated method stub
        if(index>=0&&index<imageIds.length-1)
        {
            index++;
            imageSwitcher.setBackgroundResource(imageIds[index]);
        }else
        {
            index=imageIds.length-1;
        }
    }

});

buttonPre.setOnClickListener(new OnClickListener() {

    public void onClick(View v) {
        // TODO Auto-generated method stub
        if(index>0&&index<imageIds.length)
        {
            index--;
            imageSwitcher.setBackgroundResource(imageIds[index]);
        }else
        {
            index=imageIds.length-1;
        }
    }

});

```

(5) 运行程序以查看效果，如图 4.15 所示。

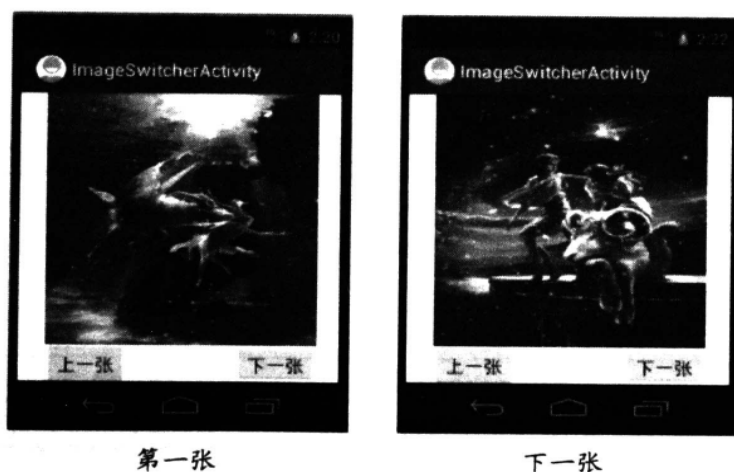


图 4.15 ImageSwitcher 控件示例

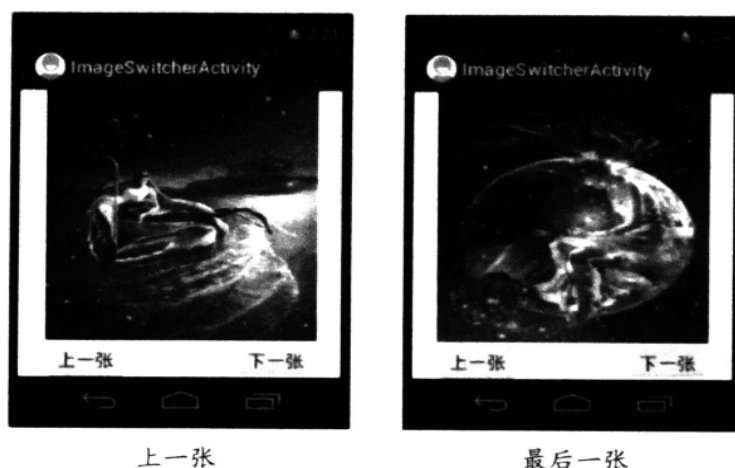


图 4.15 ImageSwitcher 控件示例（续）



4.10 网格视图

网格视图（GridView）是 Android 中比较常用的多控件视图。网格视图将其他多个控件以二维格式显示在界面表格中，这些控件都来自 ListAdapter，示例如下。

<code><GridView</code>	
<code>android:id=" "</code>	GridView控件的id
<code>android:layout_width=" "</code> <code>android:layout_height=" "</code>	GridView控件的宽度和高度
<code>android:numColumns=" "</code>	GridView控件的列数
<code>android:columnWidth=" "</code>	GridView控件每列的宽度
<code>android:gravity=" "</code>	GridView控件的对齐方式
<code>android:verticalSpacing=" "</code>	各控件间的垂直距离
<code>android:horizontalSpacing=" "</code> <code></GridView></code>	各控件间的水平距离

GridView 类的属性同样有两种配置方式，即 XML 属性配置和 Java 代码配置。表 4.8 中列出了 GridView 类常用的属性和方法。

表 4.8 GridView类的常用属性与方法

属性名称	对应方法	属性说明
android:columnWidth	setColumnWidth(int)	设置列的宽度
android:gravity	setGravity(int)	设置对齐方式
android:horizontalSpacing	setHorizontalSpacing(int)	设置各个元素之间的水平距离
android:numColumns	setNumColumns(int)	设置列数
android:verticalSpacing	setVerticalSpacing(int)	设置各个元素之间的垂直距离

【示例 4-12】下面演示如何在 GridView 控件中添加 9 张图片和对应的文本。



(1) 新建一个项目“GridView”，在布局文件 activity_grid_view.xml 中添加一个 GridView 控件和一个 ImageView 控件，并设置相应属性，代码如下。

<GridView		
android:id="@+id/gridView1"	—	GridView控件的id
android:layout_width="match_parent"	—	GridView控件的宽度匹配 父容器高度环绕内容
android:layout_height="wrap_content"	—	
android:numColumns="3"	—	GridView为3列
android:columnWidth="80dp"	—	列宽80dp
android:gravity="center_vertical"	—	位于父容器水平居中的位置
android:verticalSpacing="5dp"	—	各控件间垂直距离为5dp
android:horizontalSpacing="10dp">	—	各控件间水平距离为10dp
</GridView>		

(2) 在“res/layout”目录下新建一个布局文件 another.xml，并添加一个 ImageView 控件和一个 TextView 控件，ImageView 控件用于显示网格视图中各元素的图片，TextView 控件用于显示对应图片的文本内容，代码如下。

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/RelativeLayout01"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <ImageView android:id="@+id/image"
        android:layout_width="80dp"
        android:layout_height="80dp">
    </ImageView>

    <TextView android:id="@+id/name"
        android:layout_below="@+id/image"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:gravity="center">
    </TextView>
</RelativeLayout>
```

(3) 在 GridViewActivity 中声明一个 List、一个存放图片 id 的 int 型数组和一个存放文本内容的 String 型数组，代码如下。

```
private List<Map<String, Object>> list;
private String[] name = {"calculator","cart","phone","gift","mail","notes",
"basket","truck","wallet"};
private int[] imgId = {
    R.drawable.calculator,R.drawable.cart,R.drawable.phone,
    R.drawable.gift,R.drawable.mail,R.drawable.notes,
    R.drawable.basket,R.drawable.truck,R.drawable.wallet
};
```

(4) 利用 for 循环，将图片和文本添加到 List 中，代码如下。