第9章

嵌入式 Linux 图形设计——Android

9.1 项目导引——手机信息安全卫士系统

当今社会，Android 等智能手机已成为我们生活中不可分割的一部分，手机中的信息与工作生活联系更加紧密。在更换手机或手机丢失时，都会给您工作生活带来不便，手机信息安全卫士将为您排忧解难，它是商务人士不可缺少的工作生活助理。

通过本章“手机信息安全卫士系统”项目练习，使学生了解 Linux 系统的最新 Android 应用平台，熟悉基于 Android 平台的各种开发技术，可以从事简单 Android 应用程序开发。

9.2 项目分析

手机信息安全卫士系统是以 Android 系统平台手机为基础，利用手机技术、通信技术和数据库技术，对 Android 手机信息进行安全管理。

手机信息安全卫士功能如下：

(1)数据备份：定期备份通讯录等信息到指定邮箱，如果手机发生变化，可以立即在其他手机上恢复备份信息。

(2)防盗监控：如果手机 SIM 卡发生变更后报警，得到确认后进入监控模式，将手机新的电话号码发给原手机用户，并删除客户端的相关数据信息。

9.3 技术准备

9.3.1 构建 Android 应用程序

1. Android 简介

Android 是 Google 于 2007 年 11 月推出的基于 Linux 平台的开源手机操作系统的名称，其中文名是安卓(官方)。它采用了软件堆层(software stack，又名软件叠层)的架构，主要分为三部分。底层 Linux 内核只提供基本功能，其他的应用软件则由各公司自行开发，部分程序以 Android 编写。

伴随着第一个 Android SDK 版本(m3-rc20a)的诞生，Google 同时举办了一场 375 万美元的创意挑战赛，这场堪称有史以来奖金最丰厚的创意大赛，全球共有 1800 支队伍参加，让 Android 在短短的三个月成为全球程序员都知道的“机器人”。2008 年 9 月 22 日，经过 Android SDK 的几次改版，美国移动运营商 T-Mobile USA 与 HTC 正式推出第一

东软电子出版社

款 Google 手机——G-phone，代号“G1”，也让 Android 从模拟正式成为机器人。

自 Android 系统首次发布至今，Android 经历了很多的版本更新，表 9-1 列出了 Android 系统的不同版本的发布时间及对应的版本号：

表 9-1 Android 系统的版本信息

Android 版本	发布日期	代号
Android 1.1		
Android 1.5	2009 年 4 月 30 日	Cupcake(纸杯蛋糕)
Android 1.6	2009 年 9 月 15 日	Donut(炸面圈)
Android 2.0/2.1	2009 年 10 月 26 日	Eclair(长松饼)
Android 2.2	2010 年 5 月 20 日	Froyo(冻酸奶)
Android 2.3	2010 年 12 月 6 日	Gingerbread(姜饼)
Android 3.0/3.1/3.2	2011 年 2 月 22 日	Honeycomb(蜂巢)
Android 4.0	2011 年 10 月 19 日	Ice Cream Sandwich(冰淇淋三明治)
Android 4.1	2012 年 6 月 28 日	Jelly Bean(果冻豆)
Android 4.2	2012 年 10 月 8 日	Jelly Bean(果冻豆)
Android 4.3	2013 年 7 月 25 日	Jelly Bean(果冻豆)
Android 4.4	2013 年 11 月 1 日	KitKat(巧克力棒)
Android5.0	2014 年 10 月 16 日	Lollipop(棒棒糖)

移动应用(APPs)于 2008 年 7 月诞生在苹果的应用商店 Store，同年 10 月谷歌上线的 Android Market 仅有 40 多个，截至 2015 年 1 月，前者的 APPs 数量达到 121 万，后者则已突破 143 万。移动应用的快速发展也推动了移动用户的增长。2014 年我国移动端互联网的用户已达 10.6 亿，较 2013 年增长 231.7%。

2. 应用程序的组成部分

Android 是由不同组件组成的，这些组件通过一个项目清单(AndroidManifest.xml)绑定在一起，该清单描述了每一个组件以及它们之间是如何交互的。应用程序一般由以下 6 个组件提供基本的结构模块：

(1)Activity：应用程序的表示层。应用程序的每一个屏幕显示都是对 Activity 类的拓展，Activity 使用 View 来构建显示信息和响应用户动作的图形界面，Activity 相当于其他类似的桌面程序设计中 Form 的概念。

(2)Service：应用程序中不可见的部分。Service 在后台工作，它们可以更新数据源和课件的 Activity、可以触发 Notification，它们被用来执行一些需要持续执行的常规处理。

(3)Content Provider：一个可以共享的数据库。Content Provider 用来管理应用程序数据库，可以通过配置 Content Provider 来允许其他应用程序访问存储的本程序数据，也可以通过其他程序配置的 Content Provider 来访问它们存储的数据库。Android 系统中包含多个本地 Content Provider 来提供有用的数据库供访问，比如联系人信息等。

(4)Intent：一个简单的消息传递框架。使用 Intent，可以在系统范围内向目标 Activity 或 Service 广播消息，以说明希望执行某个动作的意图。最后系统就会确定那些最适合执行动作的目标。

(5)Broadcast Receiver：Intent 广播的消费者。通过创建和注册广播接收器，应用程序可以监听到那些匹配待定的过滤标准的广播。Broadcast Receiver 会自动启动应用程序来响应某个到来的 Intent，这个特点是它成为了事件驱动程序的最佳选择。

(6)Notification：一种用户通知框架。通知允许向用户发送信号，而不会中断他们当前的活动，它是 Service 或者 Broadcast Receiver 引起用户注意的最佳方法。比如当设备接收到一个文本消息或来电的时候，它可以通过闪灯、发出声音、显示图标或者显示对话

东软电子出版社

框的方式来提醒用户。用户也可以在自己的程序中通过 Notification 来触发相同的事件。

在 Android 中，这些组件之间的依赖性是很低的，可以理解为面向对象设计中模块间的低耦合设计，这样做的目的是可以和其他应用程序共享或者单独使用某一部分。

3. 程序的生命周期

前主流的智能手机的操作系统（如 Windows Mobile 和 Symbian）大都支持多任务。多任务有着显而易见的优点，那就是可以同时执行多个应用程序，让用户体验更完美。但是它也有着显而易见的缺点，由于手机内存有限，多一个应用程序就会多消耗一部分系统可用的内存，程序运行的越多内存消耗就越大，系统运行就越慢，导致用户体验越来越糟糕。为了解决这个问题，提高手机内存的利用率，Android 引入了一个全新的机制——应用程序生命周期。

默认的情况下，每一个 Android 应用程序都是通过它们自己的进程运行的，每一个进程都是 Dalvik 的一个单独的实例。每一个应用程序的内存和进程管理都是由运行时专门进行处理的。

应用程序进程创建到结束的过程就是应用程序的生命周期。Android 应用程序的生命周期有一个非常重要、但是又很少见的特性：应用程序进程的生命周期不是由进程自己控制的，而是由 Android 系统决定的。影响应用程序生命周期的主要因素包括：该进程对于用户的重要性，以及当前系统中剩余内存的多少。当然对于开发者而言不正确的使用系统组件可能会导致应用程序在运行过程中被意外终止。

Android 系统主动地管理着它的资源，它会采取任何措施来保证设备保持响应。这就意味着在必要的时候，进程（以及他们的承载应用程序）将会在没有警告的情况下被终止，来为更高优先级的应用程序——通常是当时正直接和用户进行交互的应用程序——释放资源。接下来我们就看一下 Android 系统根据什么来选择被终止的进程。

回收资源的时候，进程被终止的顺序是由它们承载应用程序的优先级所决定的。一个应用程序的优先级等同于它优先级最高的组件的优先级。当两个程序有相同的优先级时，处于较低优先级且运行时间最长的进程将会首先被终止。进程的优先级也会收到进程间依赖性的影响。如果一个应用程序依赖于第二个应用程序所提供的服务或者内容提供器，那么第二个应用程序至少会拥有与它所支持的这个应用程序相同的优先级。

注意：所有的 Android 应用程序都会保持在内存中运行，直到系统需要释放它的资源供其他应用程序使用为止。

图 9-1 显示了用来确定应用程序终止顺序的优先级树。

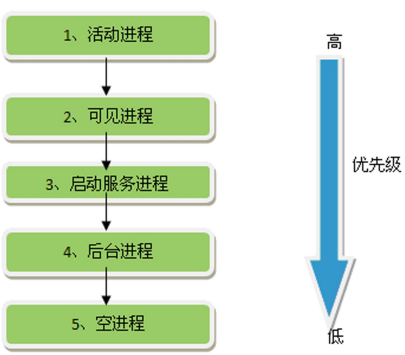

图 9-1　进程优先级树

东软电子出版社

下面我们就对图 9-1 中的各个进程做个解释，并了解组成应用程序的组件是如何确定这个状态的：

(1)活动进程：又可称之为前台进程，主要是指那些有组件正和用户进行交互的承载应用程序的进程。在系统中这样的进程非常少，除非系统可用内存已经少到不够活动进程运行，否则系统不会主动终止活动进程。如果系统不得不终止活动进程，那么意味着系统已经到了非整理内存不可的状态，这种情况下活动前台进程是为了不让用户界面停止响应。

活动进程主要包括：

①处于“活动”状态的 Activity(活动)。也就是说，它们位于前台并对用户事件进行响应。

②正在执行 onReceive 事件处理函数的活动、服务或者广播接收器。

③正在执行 onStart、onCreate、onDestroy 事件处理函数的服务。

(2)可见进程：是用户可见，但是非活动的进程，是指那些承载“可见”活动的进程。顾名思义，可见的活动能被用户看到，但是它们并不在前台运行或者能对用户事件做出反应。比如，一个活动被部分遮挡时(被一个非全屏或者半透明的活动遮挡)就会出现这种情况。可见进程的数量也很少，只有在资源极度匮乏的环境下，为保证活动进程正常运行才终止这些进程。

(3)启动服务进程：或者可以说是服务进程，是指拥有 Service 的进程。服务支持在没有可见界面的情况下仍然能够不间断的处理。尽管用户看不到这些进程，但是它们做的工作对用户来说还是比较重要的，比如，在后台播放 mp3 文件或者在后台上传、下载文件。

(4)后台进程：是指拥有不被用户可见的 Activity 的进程，并且也没有已经启动的任何服务的进程。它们不会直接影响用户的体验。在系统中通常有大量的后台进程，这些进程被放到最近最少使用(Least Recently Used，LRU)列表中。当需要终止进程时，系统确保最近一个被用户看到的进程最后被终止。

(5)空进程：顾名思义，空进程是指没有任何 Activity 的进程。为了提高系统的整体性能，Android 经常在应用程序的生存期结束后仍然把它们保存到内存中。保留这种进程的理由是提供一种缓存机制，以减少应用程序再次启动时所需要的时间。通常这些进程会被定期终止，以平衡进程缓存和底层内核缓存。

Android 系统根据组件的重要性将系统中的进程分为上面的五种，它们的优先级或者重要性如图 9-1 所示。例如一个进程既有一个 Service，又有一个可见的 Activity，那么这个进程会被标记为可见进程而不是服务进程。

4. 构建项目

(1)安装 JDK。

在 windows 上安装 JDK 比较简单，下载 JDK 的安装文件后点击. exe 安装程序，按照提示，一次点击下一步就可完成安装。在完成安装后还需要进行设置。配置步骤如下：

①右键点我的电脑。打开属性，然后选择“高级”选项卡里面的“环境变量”，如图9-2所示。

东软电子出版社

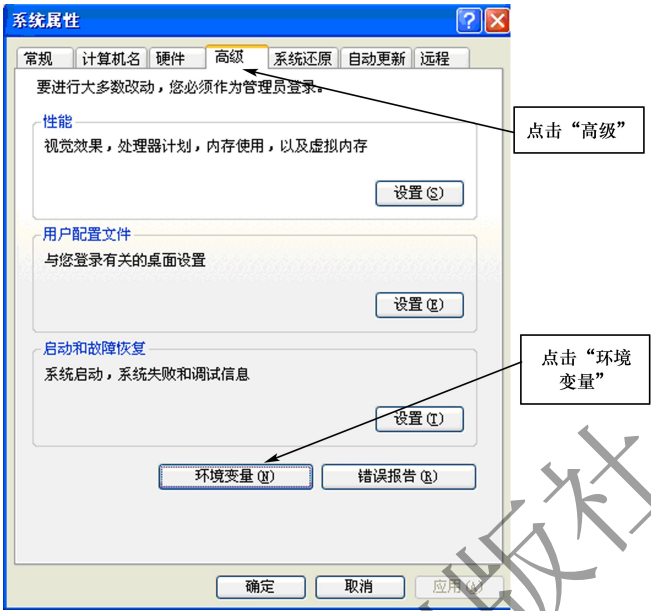

图 9-2　系统环境变量

② 在新打开界面中的系统变量需要设置三个属性“java_home”、“path”和“classpath”，在没安装过 JDK 的环境下，path 属性是本来存在的。而 java_home 和 classpath 是不存在的。图 9-3 中可以看到 classpath 和 java_home 选项。

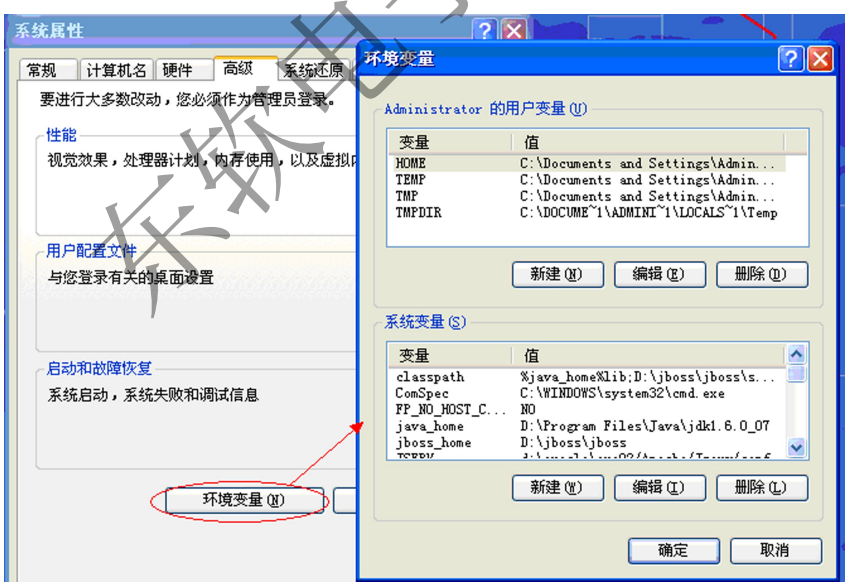

图 9-3　环境变量选项

③接下来就要配置各个选项。在没有配置过 JDK 的情况下，首先点击“新建”按钮，变量名为“java_home”，变量值为刚才 JDK 的安装路径，这里安装路径为：“D:\Program Files\Java\jdk1.6.0_07”，所以 java_home 的配置结果如图 9-4 所示。

东软电子出版社

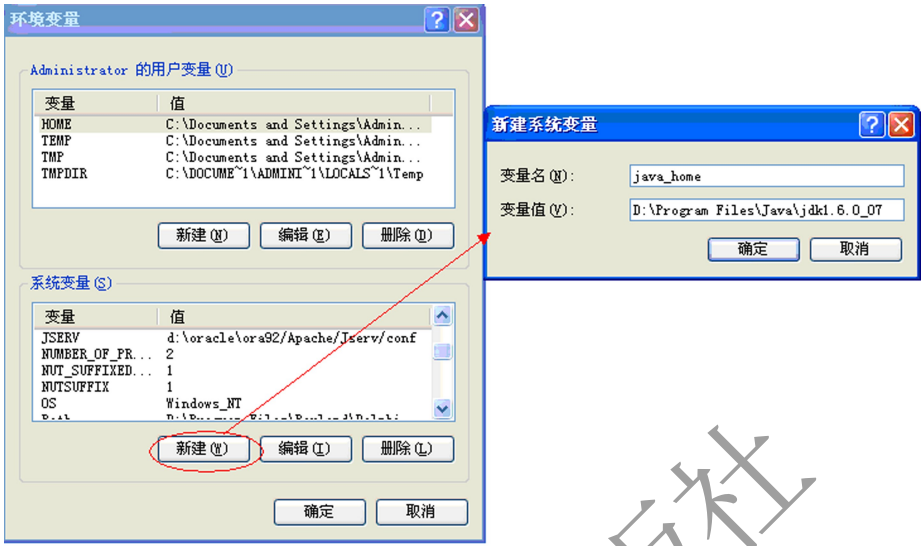

图 9-4　java_home 配置

④配置完“java_home”之后，下面来配置“path”，在系统变量里面找到 path，然后点“编辑”，path 变量的含义就是系统在任何路径下都可以识别 java 命令，其变量值为“%java_home%\bin”，(其中“%java_home%”的意思为刚才设置 java_home 的值)，如图 9-5 所示。

注意：在配置 path 的时候只需要将变量值直接加到原来的变量值后面即可，在添加的时候记得在%java_home%\bin 之前添加一个分号，即“;%java_home%\bin”。

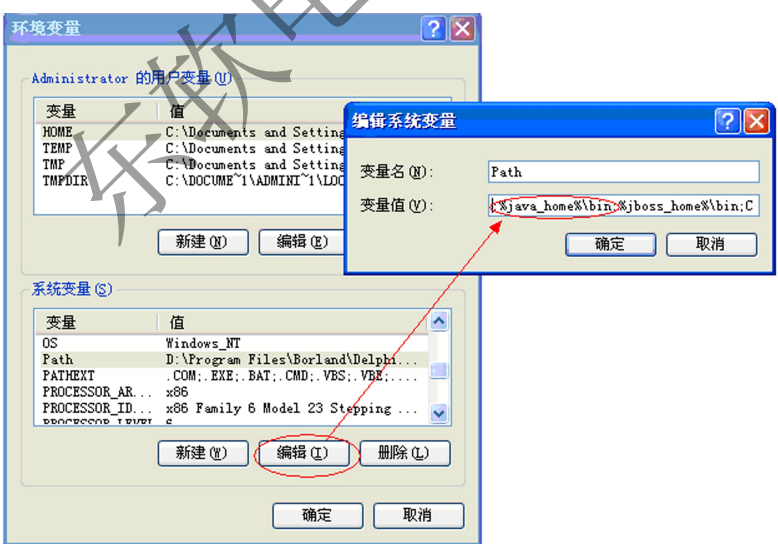

图 9-5　Path 配置

⑤最后来配置“classpath”，同样首先点击“新建”，然后在变量名上写 classpath，该变量的含义是为 java 加载类的路径，只有类在 classpath 中，java 命令才能识别。其值为“.;%java_home%\lib;%java_home%\lib\tools.jar”，如图 9-6 所示。

东软电子出版社

注意:在变量值前面要加“.”表示当前路径。

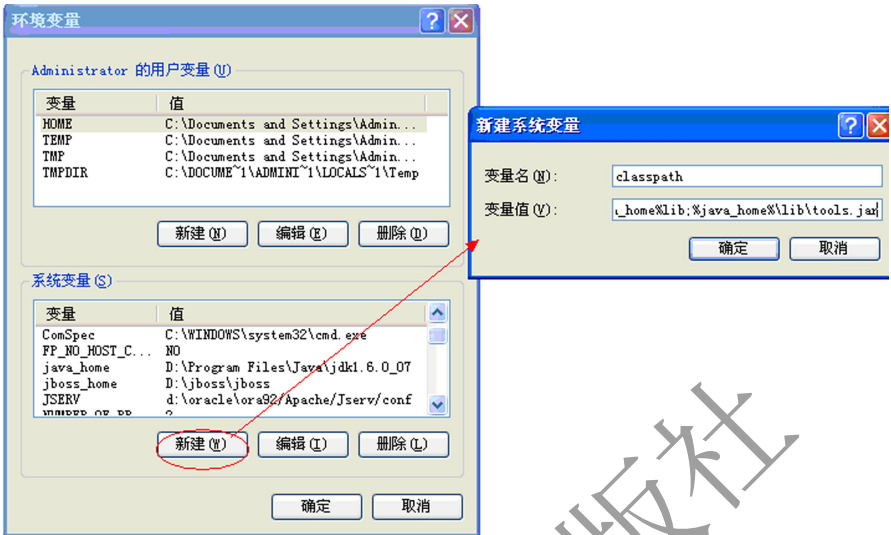

图 9-6　classpath 配置

⑥以上三个变量设置完毕,则按“确定”直至属性窗口消失,下面验证安装是否成功。先打开“开始”→“运行”,打入“cmd”,进入 dos 系统界面。然后输入“java -version”,如果安装成功。系统会显示所安装的 JDK 的版本信息(不同版本号则显示不同),如图9-7所示。

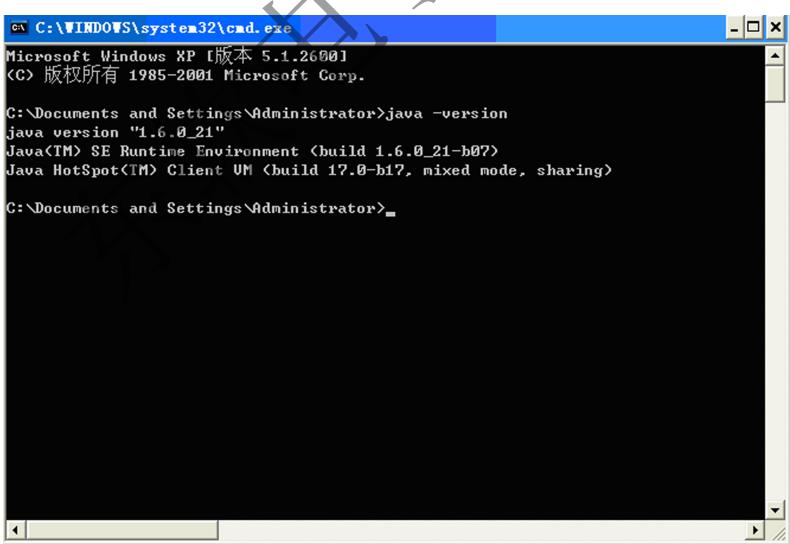

图 9-7　测试 JDK

(2)安装 SDK。

ADT(Android Developer Tools,安卓开发工具),这里我们采用 adt bundle 作为 Android APP 开发工具,adt bundle 由 Google Android 官方提供的集成式 IDE,已经集成了 clipse,是一款方便实用的安卓应用开发软件。解决了大部分入门开发人员通过

电子工业出版社

eclipse 来配置 Android 开发环境的复杂问题。

①ADT Bundle 下载和安装

Adt Bundle 各个平台的版本如表 9-2 所示。

表 9-2　　Adt Bundle 各个平台的版本

平台	安装包名	大小	MD5 校验
Windows 32-bit	adt-bundle-windows-x86-20140702. zip	370612741 bytes	9d2cf3770edbb49461788164af2331f3
Windows 64-bit	adt-bundle-windows-x86_64-20140702. zip	370763706 bytes	bfc3472a12173422ba044182ac466c13
Mac OS X 64-bit	adt-bundle-mac-x86_64-20140702. zip	320593642 bytes	24c51a1ad96c5f6d43821d978bf9866d
Linux 32-bit	adt-bundle-linux-x86-20140702. zip	371950735 bytes	5901c898bae4fe95476463a951b68404
Linux 64-bit	adt-bundle-linux-x86_64-20140702. zip	372259418 bytes	18a7c5778f96c0823349d465f58a0a36

这里采用 adt-bundle-windows-x86_64-20140702. zip 版本，下载地址：http://dl.google. com/android/adt/adt-bundle-windows-x86-20140702. zip。

下载后解压缩文件，这里解压到 D:\adt-bundle-windows-x86_64-20140702 路径，如图 9-8 所示。

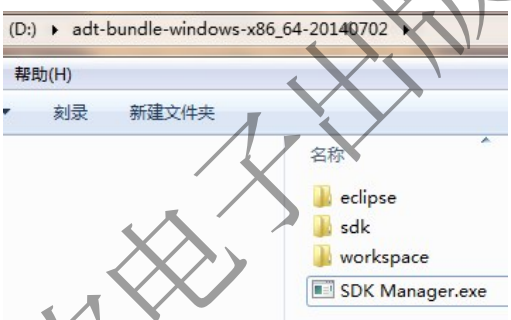

图 9-8　ADT 目录

②安装 Android SDK

打开解压文件夹里面的 eclipse/eclipse. exe，如图 9-9 所示。然后通过点击 Windows - Android SDK Manager 菜单项，如图 9-10 所示。

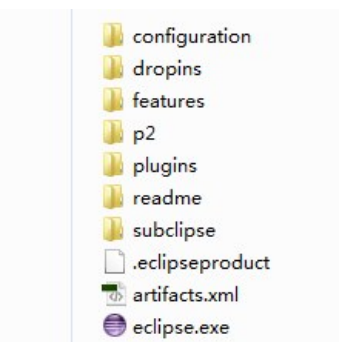

图 9-9　eclipse 开发工具

东软电子出版社

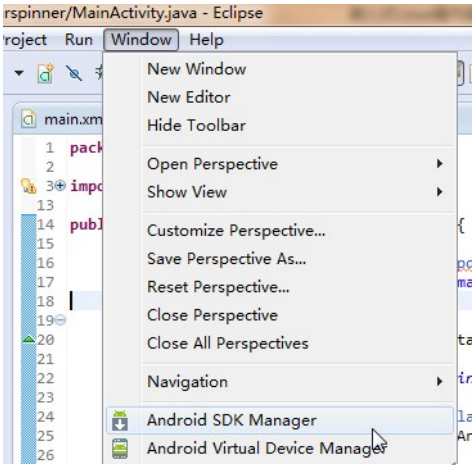

图 9-10　Android SDK Manager 菜单项

更新对应的 SDK 包，可以从官网更新，但是更新速度缓慢，也可以选择国内镜像，这里我们选择国内的东软 Android 镜像站，这样更新速度加快。选择 Tools - Options，如图 9-11 所示。

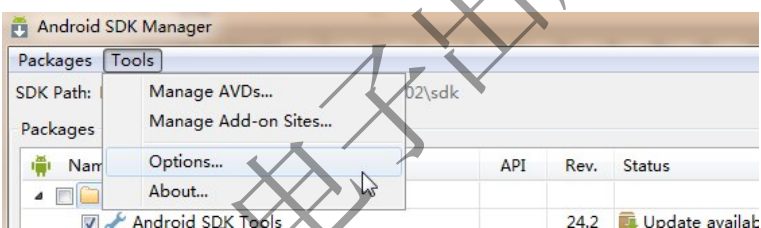

图 9-11　SDK 更新操作

设置代理，并勾选 HTTPS 强制转换 HTTP 选项，如图 9-12 所示。

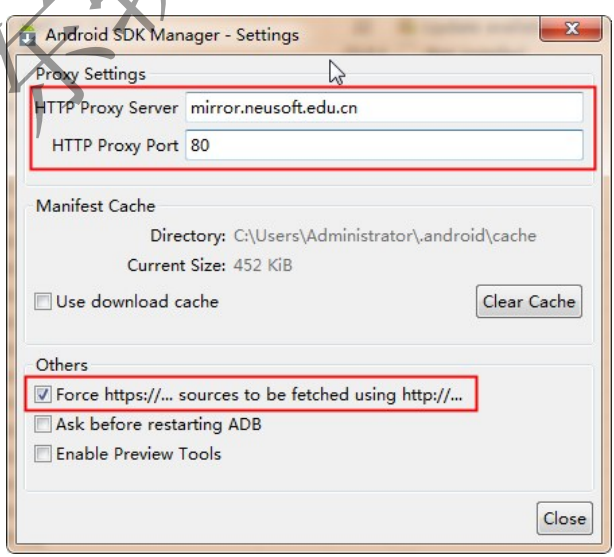

图 9-12　sdk 更新代理设置

东软电子出版社

(3)创建第一个 Android 程序。

①创建 Helloworld APP

启动 Android 开发环境的 eclipse.exe,选择 File - New - Android Application project 选项,如图 9-13 所示。

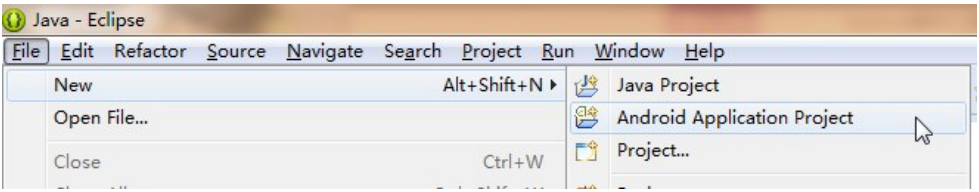

图 9-13 新建项目菜单项

进入新建项目界面,如图 9-14 所示。输入应用名称、工程名称和包名,包名一般为 com.公司名称,也可以随意填写,一般符合大众习惯,之后选择 targetSdk,然后点击 next 按钮,进入项目配置,如图 9-15 所示。

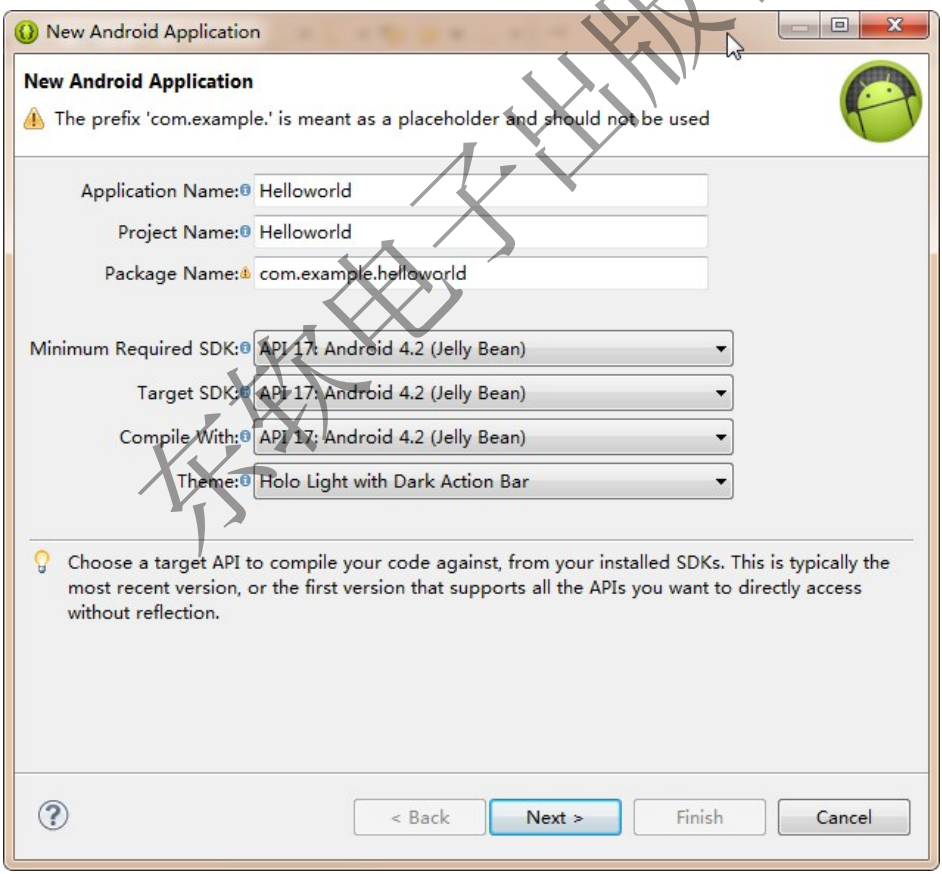

图 9-14 工程名称和 SDK 配置

东软电子出版社

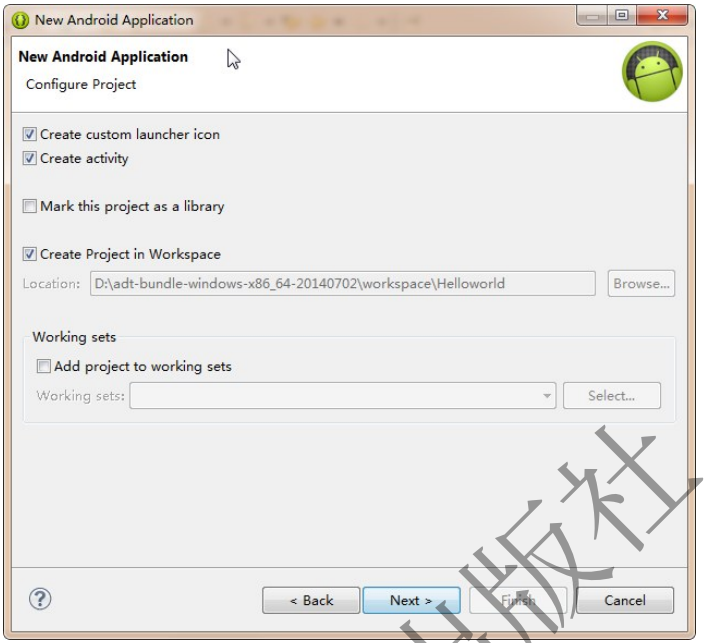

图 9-15　项目配置

设置 APP 图标，如图 9-16 所示，然后点击 next 按钮，进入 Activity 界面，选择默认的创建 BlankActivity，如图 9-17 所示。

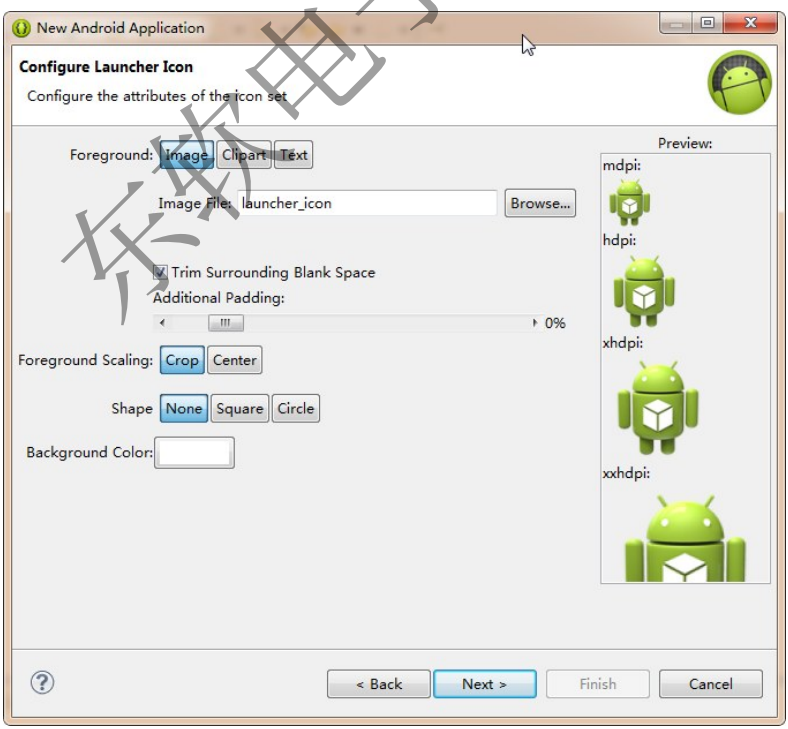

图 9-16　APP 图标

东软电子出版社

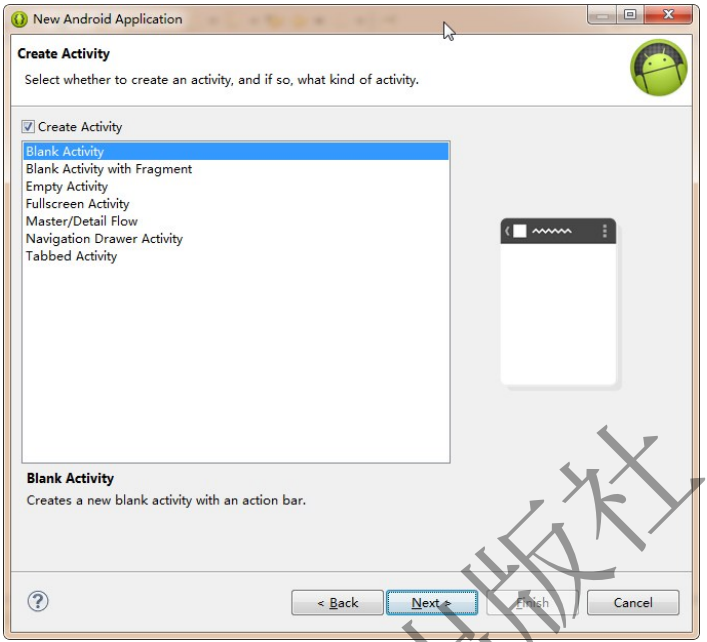

图 9-17　Activity 界面

新建一个 Activity，这里使用默认设置，如图 9-18 所示，点击 Finish 按钮，进入 Helloworld 项目开发界面，如图 9-19 所示。

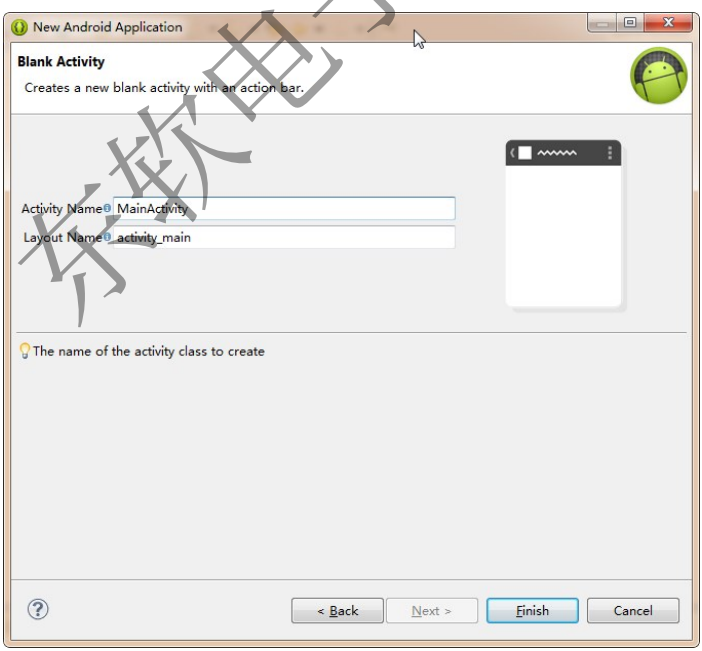

图 9-18　工程配置完成

东软电子出版社

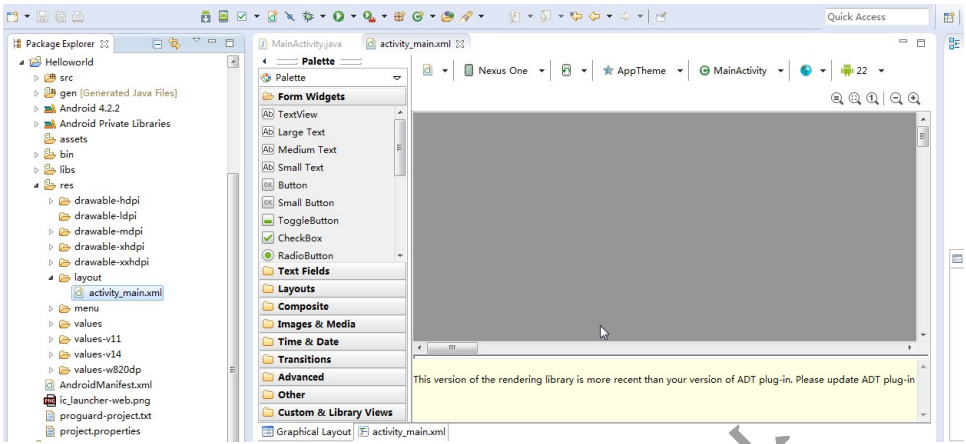

图 9-19　Helloworld 项目开发界面

②Android 项目目录结构

通过 Helloworld 项目来介绍 Android 项目的目录结构，Helloworld 项目(基于 Android 4.2.2)在 eclipse 的左侧展开 Helloworld 项目，可以看到如图 9-20 所示的目录结构。

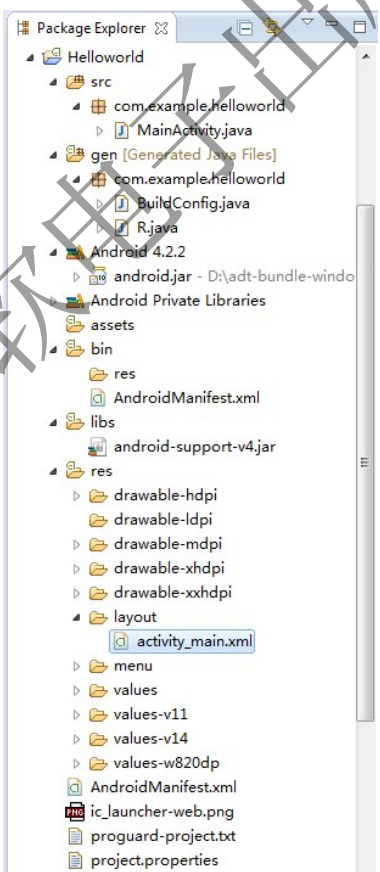

图 9-20　Helloworld 项目目录结构

东软电子出版社

主要包括以下几个文件和目录：

• src 文件夹：存放项目的源代码。

• gen 文件夹：该文件夹下面有个 R.java 文件，R.java 是在建立项目时自动生成的，这个文件是只读模式的，不能更改。

• Android 4.2.2 文件夹：该文件夹下包含 android.jar 文件，这是一个 Java 归档文件，其中包含构建应用程序所需的所有的 Android SDK 库（如 Views、Controls）和 APIs。

• Assets：包含应用系统需要使用到的例如 mp3、视频类的文件，此文件夹中的数据只允许读，不能写。

• res 文件夹：资源目录，包含你项目中的资源文件并将编译进应用程序。

• AndroidManifest.xml：项目的总配置文件，记录应用中所使用的各种组件。

③Android 虚拟机

点击图 9-21 中 eclipse 工具栏的虚拟机图标或者通过菜单 window - Android Virtual Device Manager 进入 AVD 安卓虚拟机管理对话框，如图 9-22 所示。

图 9-21 启动 Android Virtual Device Manager

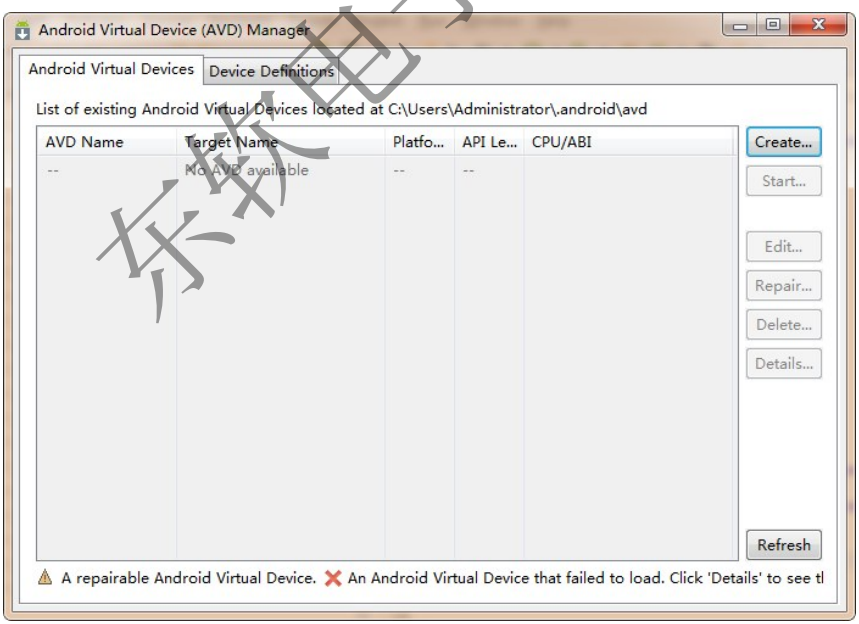

图 9-22 Android Virtual Device Manager

在 Android Virtual Device Manage 对话框中单击 Create 按钮，弹出创建新 AVD 对话框，如图 9-23 所示。

东软电子出版社

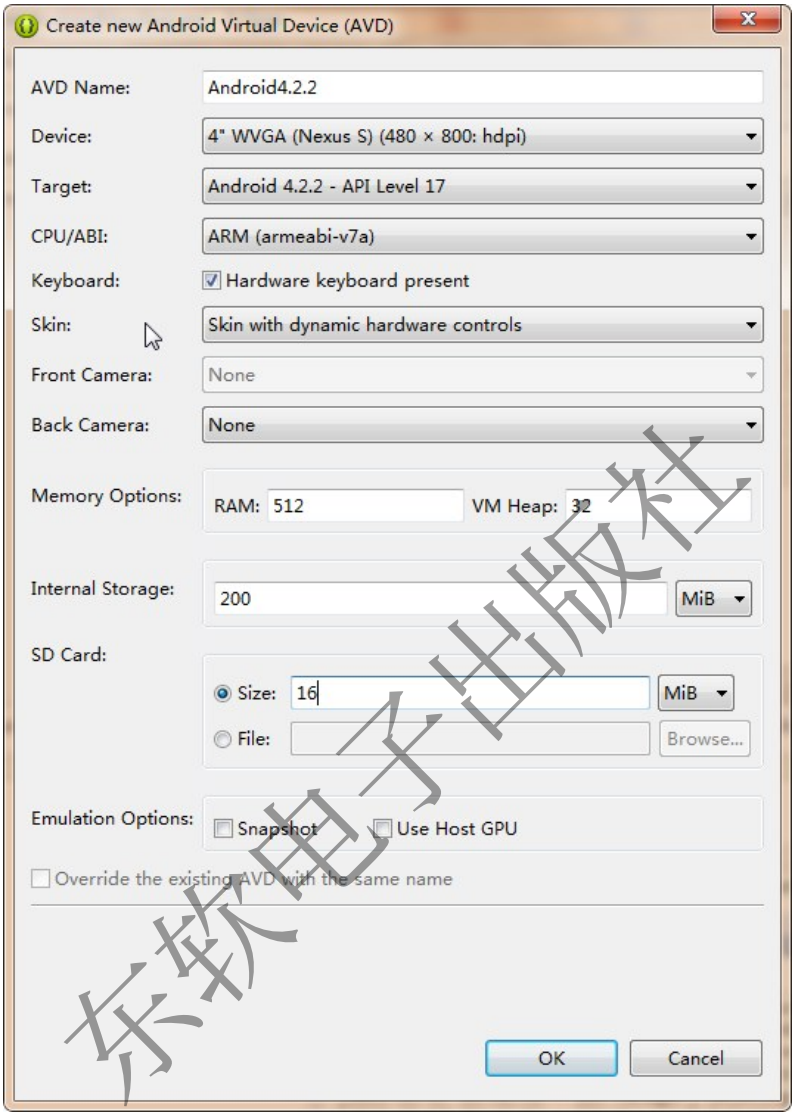

图 9-23　新建 Android Virtual Device

在 Create New Android Virtual Device(AVD)对话框中,配置虚拟安卓系统的参数,配置好后点击 OK 按钮返回 Android Virtual Device Manage 对话框。可以根据自己计算机的配置进行虚拟机设置。

注意:RAM 不可设得太大,否则启动虚拟机将非常慢。

④运行 Android 应用程序

利用 eclipse 编译运行 Helloworld 应用程序,在左侧资源管理器中选择 Helloworld 工程,点击右键,选择 Run As - Android Application,如图 9-24 所示,或者点击工具栏上的运行图标,如图 9-25 所示,这个程序就会自动下载安装到我们刚才建好的虚拟机中并自动运行。

东软电子出版社

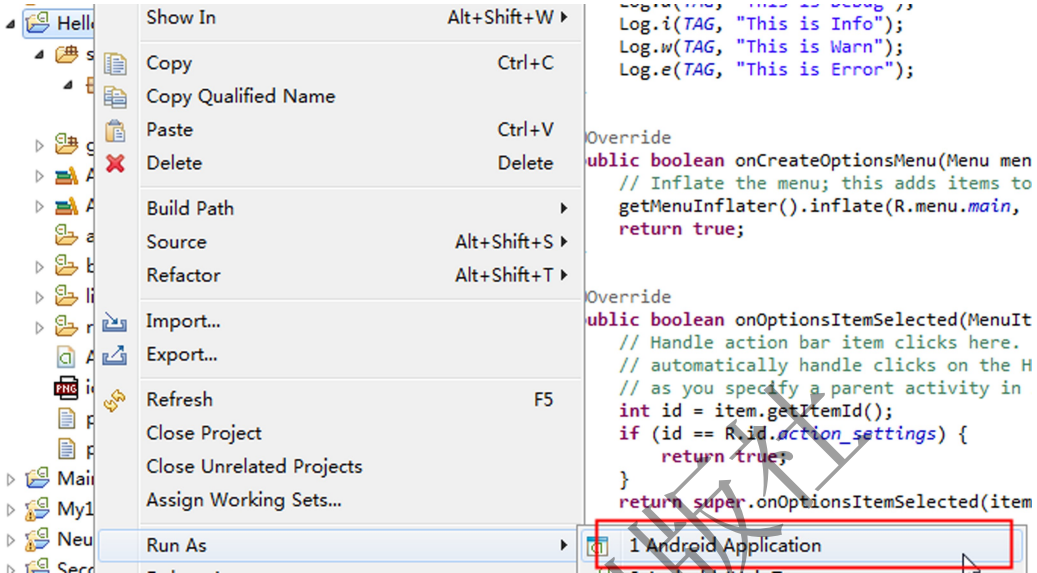

图 9-24 右键方式编译运行

运行结果如图 9-26 所示。

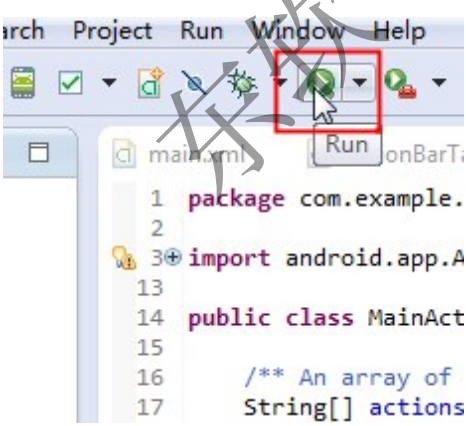

图 9-25 图标方式编译运行

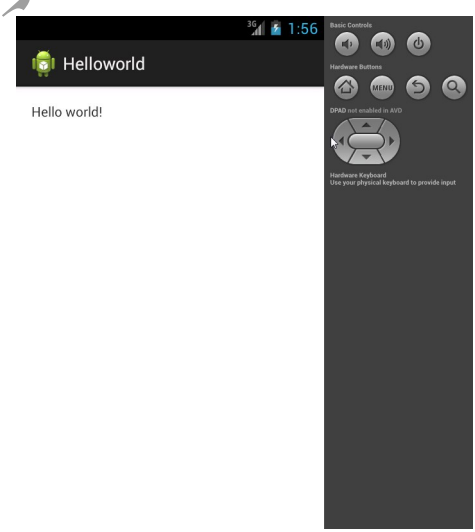

图 9-26 运行结果

可以按下 Ctrl+F12 来切换布局，切换后运行结果如图 9-27 所示。

东华电子出版社

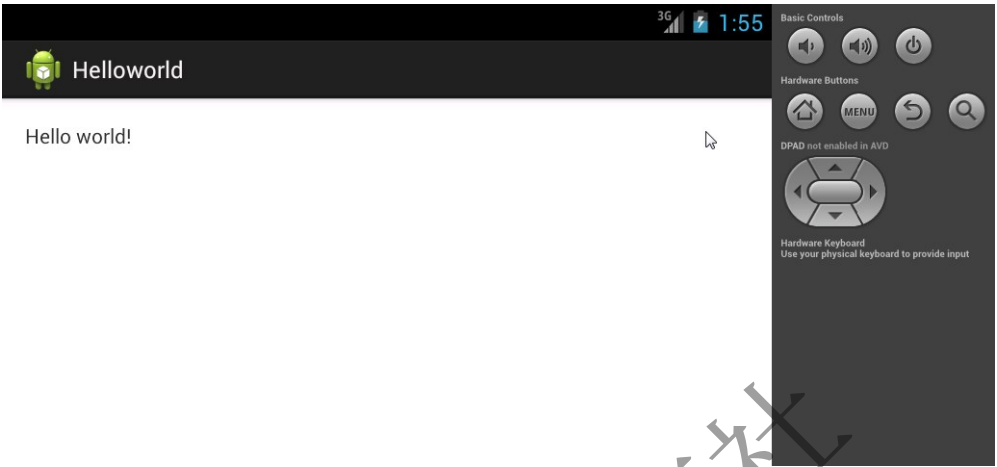

图 9-27　按下 Ctrl＋F12 后效果图

如果要退出程序的话可以点击模拟器上的退格键，返回虚拟机的主界面，如图 9-28 所示。

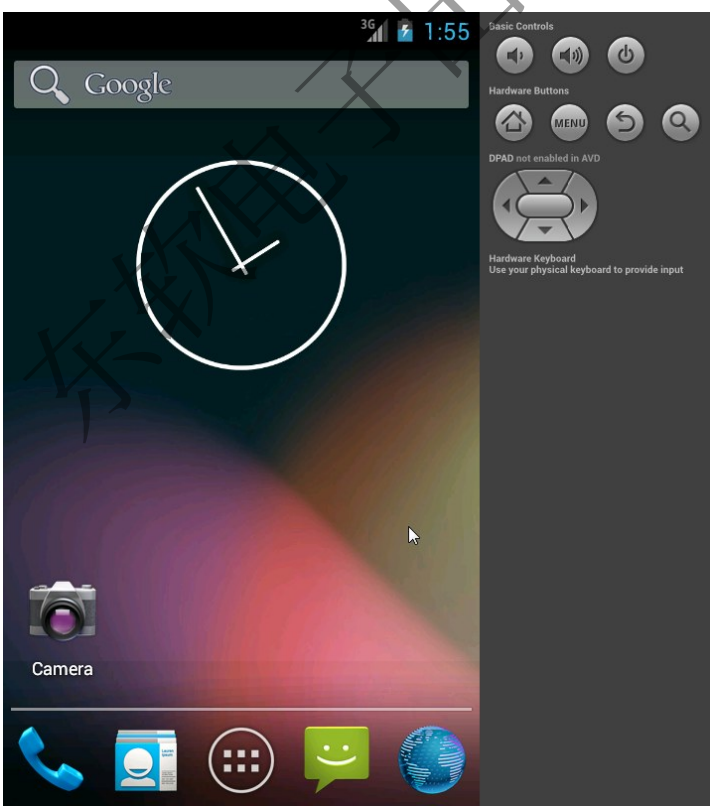

图 9-28　虚拟机主界面

虽然我们在程序中没有写任何代码，但是向导已经帮我们写好了程序的进入点、布局

文件、字符串常数、应用程序访问权限等，除了作为应用模板之外，也是 Android 手机程序的最佳学习范例。

9.3.2 程序界面设计

为应用程序创建用户界面是至关重要的。一个良好的用户的必备条件是：内容清楚、指示明白、屏幕美观和具有亲切感。应用程序界面的设计是对控件进行恰当的取舍以及功能的选择和处理的过程。

1. 构成 Android UI 的基本元素

Android 用户界面的开发包括两个方面的知识：用户界面的设计和事件处理。在 Android 中用户界面是由 View 和 ViewGroup 对象构建的。View 与 ViewGroup 都有很多种类，而它们都是 View 类的子类。事件处理则包括按钮事件、触屏事件以及一些高级控件的事件监听。

Android 用户界面有两种方式可以生成，一种是通过 xml 布局文件来生成，另一种是用代码直接生成。对于 xml 生成的布局文件可以通过 ADT 提供的 UI 预览功能来预览创建的用户界面。在写好布局的 xml 文件后只要打开项目中的“/res/layout/main.xml”并右键单击，依次选择“open with”→“Android layout Editor”菜单命令，也可以直接双击打开布局文件，然后切换到 UI 设计界面，如图 9-29 所示。

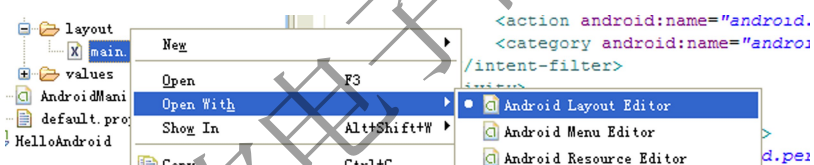

图 9-29 使用 Android Layout Editor 打开布局文件

打开后显示如图 9-30 所示界面，左边的 Layouts 标签的内容是一些线性布局，可以使用它来完成对布局的排版，如横向或者纵向。Views 标签则是 UI 控件，可以将这些控件直接拖到右边的窗口中进行编辑。

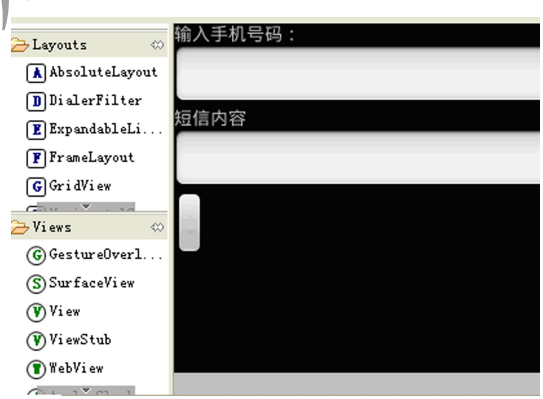

图 9-30 打开布局文件

可以通过 xml 编辑器和上面的功能配合使用来完成对布局的设计和开发。这里推

东软电子出版社

荐一款开源的软件 DroidDraw。它是一款公开了源码的 UI 设计器，可以根据自己的需要来进行修改。软件的功能比较强大，可以直接拖动控件到窗口，然后设置属性、参数等。设置好参数以后可以点击“Generate”来生成相应的布局文件。当然可以点击“Load”来载入已经编辑好的布局文件，用户界面如图 9-31 所示。

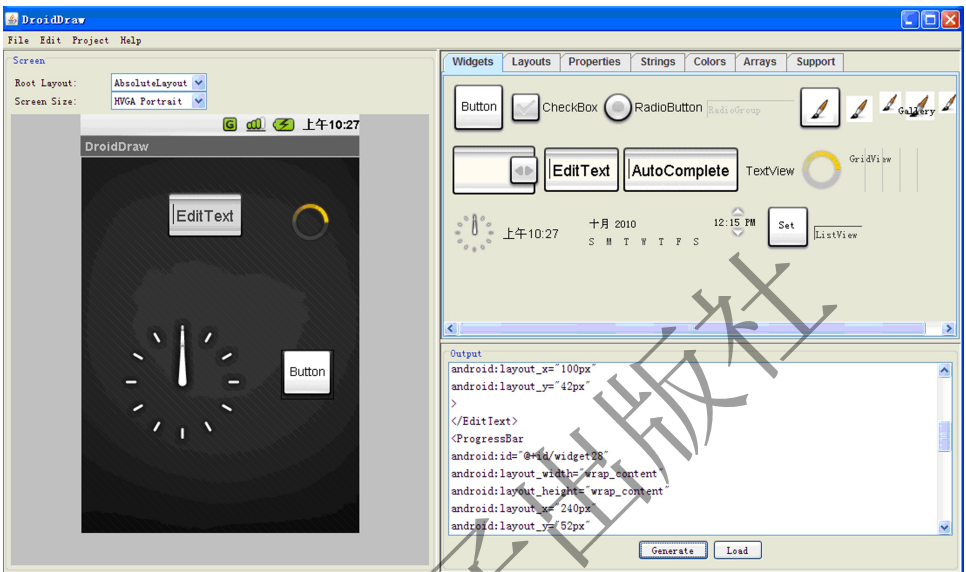

图 9-31　DroidDraw 用户界面

(1)View 简介。

View 对象是 Android 平台中用户界面体现的基础单位。任何一个 View 对象都将继承 android.view.View 类。它是一个存储有屏幕上特定的一个矩形布局和内容属性的数据结构。一个 View 对象可以处理测距、布局、绘图、焦点变换、滚动条，以及屏幕区域自己表现的按键和手势。作为一个基类，View 类为 Widget 服务，Widget 则是一组用于绘制交互屏幕元素的完全实现子类。Widget 处理自己的测距和绘图，所以可以快速地用它们去构建 UI 常用到的 Widget，包括 Text、EditText、Button、RadioButton、Checkbox 和 ScrollView 等。

(2)ViewGroup 简介。

ViewGroup 是一个 android.view.ViewGroup 类的对象。顾名思义，ViewGroup 是一个特殊的 View 对象，它的功能是装载和管理一组下层的 View 和其他 ViewGroup。ViewGroup 可以为 UI 增加结构，并且将复杂的屏幕元素组成一个独立的实体。作为一个基类，ViewGroup 为 Layout 服务。Layout 则是一组提供屏幕界面通用类型的完全实现子类。Layout 可以为一组 View 构建一个结构。

图 9-32 是一个由 View 和 ViewGroup 布局的 Activity 界面。从图 9-32 中可以看到一个 Activity 的界面可以包含多个 ViewGroup 和 View，通过两者的组合使用能够更好地完成更复杂的界面的设计。

东软电子出版社

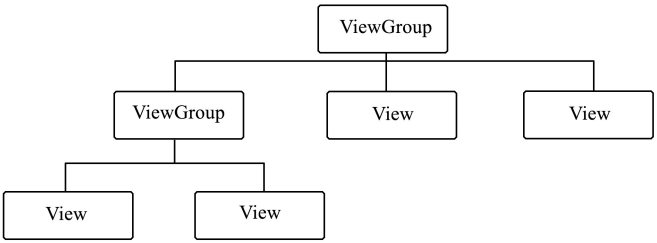

图 9-32 View 与 ViewGroup 组合使用布局的 Activity 界面

一个新的 Activity 被创建的时候是一个空白屏幕，可以把自己的用户界面放到上面。要设置用户界面，可以调用 setContentView()方法，并传入需要显示的 View 实例(通常是一个布局)。由于空白屏幕并不是我们所想要的，所以在创建一个新的 Activity 的时候在 onCreate()处理程序中总是调用 setContentView()方法来设置需要显示的用户界面。

2. 常用 UI 控件

应用程序的人机交互界面由许多 Android 控件组成，在前面已经用到了一些常用的控件，如 EditText、TextView 和 Button 等。这些控件是直接与用户交互的对象，掌握好这些控件对以后的开发会起到至关重要的作用。

(1)按钮(Button)。

按钮是在开发中最常用到的控件。在 Android 平台中，按钮是通过 Button 来实现的。Button 的属性与前面讲解过的 TextView 和 EditView 相似，大家可以参照前面的知识来掌握 Button 的属性。

为了能够响应按钮被按下后的事件，需要对按钮设置 setOnClickListener()事件监听。本节通过按钮来改变 TextView 中文字的颜色。程序运行后效果如图 9-33 和图9-34 所示。

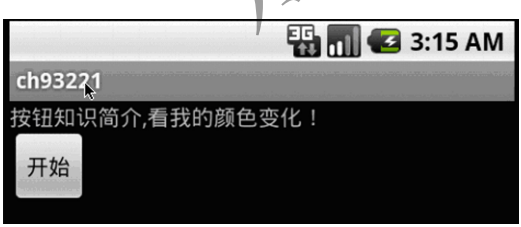

图 9-33 程序初始状态

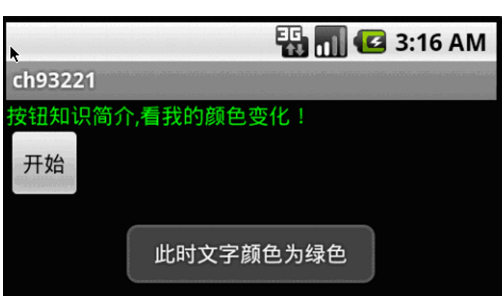

图 9-34 点击“开始”

程序的布局很简单，这里不再介绍布局文件的源代码。直接来看程序的代码实现：

```
public class CH93221 extends Activity {
    /*
     * 声明 TextView 和 Button 对象
     */
```

东软电子出版社

```
    private Button btnStart;
    private TextView tv;
    private int[] colorArray;
    private int colorNum;
    private String strColor[];
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        /*
         * 获取通过 findViewById()获得对象
         */
        btnStart = (Button)findViewById(R.id.btnStart);
        tv = (TextView)findViewById(R.id.texview1);
          //设置颜色数组
          colorArray = new int[]{Color.YELLOW,Color.BLUE,Color.GREEN,Color.DKGRAY,Color.
CYAN,Color.MAGENTA};
          strColor = new String[]{"黄色","蓝色","绿色"," 深灰色"," 青绿色"," 紫红色"};
          colorNum = 0;
          /*
           * 设置按钮的监听函数
           */
          btnStart.setOnClickListener(new Button.OnClickListener(){
            public void onClick(View v){
            if(colorNum<colorArray.length){
              tv.setTextColor(colorArray[colorNum]);
               Toast toast = Toast.makeText(CH93221.this, "此时文字颜色为" + strColor
[colorNum], Toast.LENGTH_SHORT);
              //设置 Toast 显示的位置
              toast.setGravity(Gravity.TOP, 0, 150);
              toast.show();
              colorNum + + ;
            }
            else colorNum = 0;
          }
          });
      }
  }
```

东软电子出版社

在程序中定义了一个 TextView 对象和 Button 对象；两个颜色数组 colorArray 和 strColor：前者用来在程序中设定 TextView 中文字的颜色，后者在 Toast 的提示信息中使用；通过使用 colorNum 来控制数组的下标从而更新字体颜色。

为了能更新 TextView 中的字体颜色设置 setOnClickListener()来监听每个按钮事件，通过 setTextColor()方法将颜色数组 colorArray 中的颜色设置成 TextView 中字体的颜色；在 Toast 的提示信息中显示字符串数组中的颜色信息。其中注意 toast.setGravity(Gravity.TOP, 0, 150)，用它来设定提示信息的显示位置，默认的情况下是在屏幕的下端，现在让它显示在屏幕的上端，其中最后两个参数分别是在 x 轴和 y 轴上的位移。

(2)文本框(TextView)与编辑框(EditText)。

①文本框(TextView)。

在前面的学习中已经知道 TextView 是用来显示文本信息的控件。可以在布局文件中设置 TextView 的内容，当然也可以在程序中动态的设置 TextView 的内容。下面的例子通过定义格式化的定型对象来对 TextView 中的内容进行样式化。相当于 CSS 样式的方法，可以用来指定颜色、大小等。程序运行如图 9-35 所示。

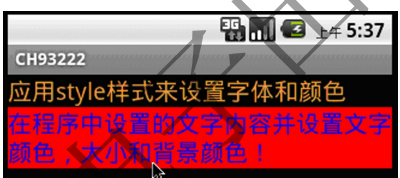

图 9-35 TextView 使用

在程序中使用了两个 TextView，其中第一个 TextView 通过 style 来设置 TextView 文本的字体大小和字体颜色信息。文本内容在布局文件 main.xml 通过 android:text=""来设置。第二个 TextView 的文字内容和字体颜色以及背景颜色在程序中来设置。下面来看一下布局文件 main.xml 文件的代码清单：

```
<? xml version = "1.0" encoding = "utf-8"? >
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    >
<TextView
    style = "@style/style1"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "@string/teststyle"
    />
<TextView
```

东软电子出版社

```
    android:id = "@ + id/textview2"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    />
</LinearLayout>
```

在上面的代码中需要注意：在第一个 TextView 中 style＝“@style/style1”这一行通过 style 来设置 TextView 的属性。其中 style.xml 是事先定义好的样式，位于 res\values 下。Style.xml 的源代码如下所示：

```
<? xml version = "1.0" encoding = "UTF-8"? >
<resources>
    <style name = "style1">
    <item name = "android:textSize">20sp</item>
    <item name = "android:textColor"># EC9237</item>
    </style>
</resources>
```

主程序的代码如下所示：

```
//包的引入语句省略
public class CH93222 extends Activity {
    /* * Called when the activity is first created. * /
    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        TextView tv2 = (TextView)findViewById(R.id.textview2);
        tv2.setBackgroundColor(Color.RED);
        tv2.setTextSize(20);
        tv2.setTextColor(Color.BLUE);
        tv2.setText("在程序中设置的文字内容并设置文字颜色,大小和背景颜色!");
    }
}
```

在上面的代码中需要注意：首先通过 findViewById(R.id.textview2)获得布局文件中 id 为 textview2 的 TextView。然后通过 setBackgroundColor(Color.RED)来将 TextView 的背景颜色设置为红色；通过 setTextSize(20)将文字的字体设置为 20。通过 setTextColor(Color.BLUE)将文字颜色设置为蓝色；最后通过 setText()来设置 TextView 的内容。上面的例子使用了两种不同的方式了设置 TextView 的属性，在实际的开发过程中可以根据不同的情况来使用不同的方法。

上面的例子使用了 Color.BLUE 将字体颜色设置成为蓝色，在 Android 中有 12 种不同的颜色：

东软电子出版社

Color.BLACK(黑色),Color.BLUE(蓝色),Color.CYAN(青绿色),Color.DKGRAY(深灰色),Color.GRAY(灰色),Color.GREEN(绿色),Color.LTGRAY(浅灰色),Color.MAGENTA(紫红色),Color.RED(红色),Color.TRANSPARENT(透明的),Color.WHITE(白色),Color.YELLOW(黄色)。

这些颜色常数定义在 Android.graphics.Color 中,如表 9-3 所示。

表 9-3 颜色常数

类型	常数	值	色码
int	BLACK	−16777216	0xff000000
int	BLUE	−16776961	0xff0000ff
int	CYAN	−16711681	0xff00ffff
int	DKGRAY	−12303292	0xff444444
int	GRAY	−7829368	0xff888888
int	GREEN	−16711936	0xff00ff00
int	LTGRAY	−3355444	0xffcccccc
int	MAGENTA	−65281	0xffff00ff
int	RED	−65536	0xffff0000
int	TRANSPARENT	0	0x00000000
int	WHITE	−1	0xffffffff
int	YELLOW	−256	0xffffff00

②编辑框(EditText)。

EditText 也是我们经常用到的控件。顾名思义,EditText 就是能够编辑 Text 的文本框,可以在里面输入内容。它可以接受多行输入并能自动换行。我们同样通过一个实例来向大家演示一下 EditText 的基本用法。先来看一下程序的运行结果,如图 9-36 和图 9-37 所示,在用户没有输入姓名和密码时则会提示用户输入姓名和密码。当用户输入姓名和密码时会在下面的 TextView 中显示出用户的输入。

图 9-36 EditText 实例

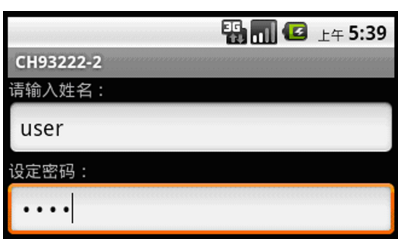

图 9-37 获取 EditText 中的内容

在这个例子中,我们在布局文件 main.xml 中定义了两个 EditText 来作为用户输入的文本框,三个 TextView 用来输出提示信息。其中在第二个 EditText 控件上我们设置了事件监听方法:setOnKeyListener(),并实现了 onKey()方法,当用户在密码框中按键输入时便会触发这个事件,从而可以通过 getText()方法来取得用户输入的内容。

东软电子出版社

下面先来看一下布局文件的代码清单：

```
<? xml version = "1.0" encoding = "utf-8"? >
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    >
<TextView
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:text = "@string/name"
    />
<EditText
    android:id = "@ + id/edtName"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    android:maxLength = "10"
    android:hint = "请输入用户名"
    android:textColorHint = "@drawable/red"
    />
<TextView
  android:layout_width = "fill_parent"
  android:layout_height = "wrap_content"
  android:text = "@string/password"/>
<EditText
  android:id = "@ + id/PWD"
  android:layout_width = "fill_parent"
  android:layout_height = "wrap_content"
  android:password = "true"
  android:maxLength = "10"/>
<TextView
  android:id = "@ + id/message"
  android:layout_width = "wrap_content"
  android:layout_height = "wrap_content"
  />
</LinearLayout>
```

在上面的代码中需要注意一下下面的代码：

```
android:maxLength = "10"
    android:hint = "请输入用户名"
    android:textColorHint = "@drawable/red"
```

东软电子出版社

通过 android:maxLength="10"来设定用户名的长度最大为 10 个字符;android:hint="请输入用户名"的作用是当用户名的编辑框中没有输入内容就会显示提示信息,在密码的编辑框中我们是在程序中通过 setHint()来实现同样的效果;android:textColorHint="@drawable/red"用来设置提示显示信息的颜色。这里的颜色文件是位于 CH93222_2\res\values 下的 color.xml 文件。color 文件的源代码如下所示,在代码中我们只设定了一个颜色标签。

```
<? xml version = "1.0" encoding = "UTF-8"? >
<resources>
    <drawable name = "red">#FFFF0000</drawable>
</resources>
```

布局文件完成后下面我们来看一下具体的代码实现。如下代码清单所示:

```
public class CH93222_2 extends Activity {
    //声明 TextView、EditText 对象
    private TextView tv;
    private EditText edtName;
    private EditText edtPWD;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //获得 TextView 和 EditView 对象
        tv = (TextView)findViewById(R.id.message);
        edtName = (EditText)findViewById(R.id.edtName);
        edtPWD = (EditText)findViewById(R.id.PWD);
          /*设置提示信息,当密码为空时提示。
         * 用户名为空的提示是在 xml 中通过:android:hint = "请输入用户名"实现
         *
         */
    edtPWD.setHint("请输入密码!");
        edtPWD.setOnKeyListener(new EditText.OnKeyListener(){
          @Override
          public boolean onKey(View v, int keyCode, KeyEvent event){
              tv.setText("您的用户名为:" + edtName.getText().toString() + "密码为:" +
edtPWD.getText().toString());
          return false;
        }
      });
    }
}
```

东软电子出版社

在上面的代码中使用 setOnKeyListener()来监听用户在编辑框中的输入。我们重写了 onKey()方法，当用户输入内容时将 EditText 中的文本信息通过 getText()方法获得，并显示在 TextView 中。

这个例子中的实时输入实时显示的效果可以扩展到许多手机应用中，比如文字过滤效果：例如当用户输入不雅文字时可以不接受部分关键字，如用户输入 shit 时在 TextView 中显示 sh * t。用兴趣的读者可以自己尝试一下。

此外，不仅仅是 Widget 才具备 setOnKeyListener 方法的重写的功能，事实上，在 View 里也有 View. setOnKeyListener()，也就是捕捉 User 单击键盘时的事件处理，但需要注意，只有 View 取得用户焦点才能触发 onKeyDown 事件。

(3)菜单(Menu)。

Android 系统里面有 3 种类型的菜单：options menu，context menu，sub menu。

①options menu。

按 Menu 键就会显示，用于当前的 Activity。它包括两种菜单项：

• 因为 options menu 在屏幕底部最多只能显示 6 个菜单项，这些菜单项称为 icon menu。icon menu 只支持文字(title)以及 icon，可以设置快捷键，不支持 checkbox 以及 radio 控件，所以不能设置 checkable 选项。

• 而多于 6 的菜单项会以“more” icon menu 来调出，称为 expanded menu。它不支持 icon，其他的特性都和 icon menu 一样。

在 Activity 里面，一般通过以下函数来使用 options menu：

• Activity::onCreateOptionsMenu (Menu menu)：创建 options menu，这个函数只会在 menu 第一次显示时调用。

• Activity::onPrepareOptionsMenu (Menu menu)：更新改变 options menu 的内容，这个函数会在 menu 每次显示时调用。

• Activity::onOptionsItemSelected (MenuItem item)：处理选中的菜单项。

②context menu。

要在相应的 view 上按几秒后才显示，用于 view，跟某个具体的 view 绑定在一起。这类型的菜单不支持 icon 和快捷键。在 Activity 里面，一般通过以下函数来使用 context menu：

• Activity::registerForContextMenu(View view)：为某个 view 注册 context menu，一般在 Activity::onCreate 里面调用。

• Activity::onCreateContextMenu(ContextMenu menu, View v, ContextMenu. ContextMenuInfo menuInfo)：创建 context menu，和 options menu 不同，context meun 每次显示时都会调用这个函数。

• Activity::onContextItemSelected(MenuItem item)：处理选中的菜单项。

③sub menu。

options menu 和 coutext menu 都可以加入子菜单，但子菜单不能嵌套子菜单，这意味着在 Android 系统，菜单只有两层，设计时需要注意。同时子菜单不支持 icon。

上述的三种类型的 menu 都能够定义为 xml 资源，但需要手动地使用 MenuInflater

东软电子出版社

来得到 Menu 对象的引用。

一个菜单，对应一个 xml 文件，因为要求只能有一个根节点<menu>。xml 文件保存为 res/menu/some_file. xml。Java 代码引用资源：R. menu. some_file。

接下来介绍相关的节点和属性(所有的属性都定义为 android 空间内，例如 android:icon="@drawable/icon")：

<menu> 根节点，没有属性。

<group> 表示在它里面的<item>在同一 group。相关属性包括：

• id:group id

• menuCategory:对应 常量 Menu CATEGORY_* — 定义了一组的优先权，有效值:container,system,secondary,和 alternative。

• orderInCategory:定义这组菜单在菜单中的默认次序,int 值。

• checkableBehavior:这组菜单项是否 checkable。有效值:none,all(单选/单选按钮 radio button),single(非单选/复选类型 checkboxes)。

• visible:这组菜单是否可见,true or false。

• enabled:这组菜单是否可用,true or false。

<item>菜单项，可以嵌入<menu>作为子菜单。相关属性包括：

• id:item id。

• menuCategory：用来定义 menu 类别。

• orderInCategory：用来定义次序，与一个组在一起(Used to define the order of the item, within a group)。

• title：标题。

• titleCondensed:标题摘要，当原标题太长的时候，需要用简短的字符串来代替 title。

• icon：icon 图标。

• alphabeticShortcut：字母快捷键。

• numericShortcut:数学快捷键。

• checkable:是否为 checkbox，true or false 。

• checked:是否设置为 checked 状态,true or false。

• visible：是否可见，true or false。

• enabled:是否可用,true or false。

菜单样式如图 9-38 所示。

图 9-38 菜单样式

东软电子出版社

由于这是 contextMenu，所以可以看到即使 xml 定义里面的 item1. seticon 了，但还是没有显示出来的，即那语句是无效的。

另外，要明确的是，要显示 radio，需要用 group，而 group 里面的 item 设置了 checked = true 即选中。而 checkable 和 checked 的区别在于，checkable = true 表示这个 item 是 checkbox，checked 则表示是否选中。所以对于 checkbox item，最好先写 checkable = "true"，然后再写 checked。

(4)对话框(Dialog)。

对话框也是 Android 提供的一种可见的组件。Android 提供不同类型的对话框，前面介绍了 DatePickerDialog 和 TimePickerDialog 这两种对话框，下面介绍 Dialog，它就相当于我们在 javascript 中的 Alert 一样，用来显示提示信息；还有一种是 AlertDialog，它是一个简单的可以交互的对话框，跟 JavaScript 中的 confirm 对话框是一样的效果，能够允许用户选择 yes 或者 no，单选按钮，复选按钮或者简单的文本输入。当然还有一种对话框就是 ProcessDialog。对于 ProgressDialog，在 Windows 窗口程序或者 Flash 程序中经常见到，诸如“加载中…”等对话框。在 Android 系统中，我们则是通过 ProgressDialog 来实现，这个类封装在 Android. app. ProgressDialog 里，但是需要注意一点：在 Android 中，ProgressDialog 必须要在后台程序运行完毕前，以 dismiss() 方法来关闭取得焦点，否则程序就会陷入无法终止的无穷循环中；在线程里面不可有任何更改 Context 或 parent View 的任何状态、文字输出等事件，因为线程中的 Context 和 View 并不属于 parent，两者之间没有关联。

下面通过一个例子来看一下关于 Android 中对话框的类型和使用方法。我们模拟一个论坛发表评论时经常出现一个场景：当我们要发表评论时首先需要登录，登录后才可以发表评论。

在这个例子中，首先要为 Activity 设置布局，在这里使用的布局就是 main. xml 文件，只是添加了一个 Button、一个 EditText 和一个 TextView。这里不详细列出布局文件的代码，只是将布局后的效果展示出来，如图 9-39 所示。

图 9-39　Main. xml 布局文件效果图

另外，在这个例子中还需要对自定义的对话框的布局进行设置。图 9-40 所示是自己

东软电子出版社

定义的登录框。其中包括两个 TextView 和两个 EditText。

图 9-40　自定义登录界面

其布局文件如下代码所示：

```
<? xml version = "1.0" encoding = "utf-8"? >
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "wrap_content"
    >
<TextView
    android:layout_height = "wrap_content"
    android:layout_width = "wrap_content"
    android:layout_marginLeft = "20dip"
    android:layout_marginRight = "20dip"
    android:text = "账号"
    android:gravity = "left"
    android:textAppearance = "? android:attr/textAppearanceMedium"/>
<EditText
    android:id = "@ + id/userName"
    android:layout_height = "wrap_content"
    android:layout_width = "fill_parent"
    android:layout_marginLeft = "20dip"
    android:layout_marginRight = "20dip"
    android:scrollHorizontally = "true"
    android:autoText = "false"
    android:capitalize = "none"
    android:gravity = "fill_horizontal"
    android:textAppearance = "? android:attr/textAppearanceMedium"/>
```

东软电子出版社

```
    <TextView
    android:layout_height = "wrap_content"
    android:layout_width = "wrap_content"
    android:layout_marginLeft = "20dip"
    android:layout_marginRight = "20dip"
    android:text = "密码"
    android:gravity = "left"
    android:textAppearance = "? android:attr/textAppearanceMedium"/>
<EditText
    android:id = "@ + id/userName"
    android:layout_height = "wrap_content"
    android:layout_width = "fill_parent"
    android:layout_marginLeft = "20dip"
    android:layout_marginRight = "20dip"
    android:scrollHorizontally = "true"
    android:autoText = "false"
    android:capitalize = "none"
    android:gravity = "fill_horizontal"
    android:password = "true"
    android:textAppearance = "? android:attr/textAppearanceMedium"/>
</LinearLayout>
```

在上面的代码中需要注意几个属性的应用：android:gravity="left"表示对齐方向；android:textAppearance="? android:attr/textAppearanceMedium"表示文字的显示外观，其中有四个外观：

```
android:textAppearance = "? android:attr/textAppearanceSmall"
android:textAppearance = "? android:attr/textAppearanceMedium"
android:textAppearance = "? android:attr/textAppearanceLarge"
android:textAppearance = "? android:attr/textAppearanceLarge"
```

android:scrollHorizontally="true"表示当文字超出一行时是否出现滚动条；android:layout_marginLeft 和 android:layout_marginRight 表示左右边空白的大小。

在设置好上面的布局后来看一下主程序中的源代码。

代码清单\CH93224\src\com\study\chapter9

```
//包的引入省略
public class CH93224 extends Activity {
  ProgressDialog logining;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button btnCommont = (Button)findViewById(R.id.btnCom);
        btnCommont.setOnClickListener(new OnClickListener(){
@Override
```

东软电子出版社

```
public void onClick(View v){
  // TODO Auto-generated method stub
  Dialog dialog = new AlertDialog.Builder(CH93224.this)
    .setTitle("登录提示")//设置登录提示
    .setMessage("您还没有登录,请先登录")//设置提示信息
    .setPositiveButton("确定",  //设置确定按钮
      new DialogInterface.OnClickListener(){

        @Override
        public void onClick(DialogInterface dialog,
            int which){
          // TODO Auto-generated method stub
          //跳转到登录框
          LayoutInflater factory = LayoutInflater.from(CH93224.this);
            //用来将 xml 文件转成 View
          final View DialogView = factory.inflate(R.layout.login, null);
          AlertDialog dlg = new AlertDialog.Builder(CH93224.this)
            .setTitle("登陆框")
            .setView(DialogView)
            .setPositiveButton("确定", //设置确定按钮
              new DialogInterface.OnClickListener(){

                @Override
                public void onClick(
                  DialogInterface dialog,
                  int which){
                  // TODO Auto-generated method stub
                logining = ProgressDialog.show(CH93224.this, "登录中", "正在登录,
请稍后……",true);

                new Thread(){
                  public void run(){
                    try{
                      sleep(3000);
                    }catch(Exception e){
                      e.printStackTrace();
                    }
                    finally{
                      logining.dismiss();
                    }
                  }
                  }.start();
                }

              })
```

东软电子出版社

```
                .setNegativeButton("取消",//设置取消按钮事件
                  new DialogInterface.OnClickListener(){

                    @Override
                    public void onClick(DialogInterface dialog,
                        int which){
                      // TODO Auto-generated method stub
                      CH93224.this.finish();
                    }

                  }).create();
                  dlg.show();



              }

          }).setNeutralButton("取消",
            new DialogInterface.OnClickListener(){
          public void onClick(DialogInterface dialog,int which){
          CH93224.this.finish();
          }
          }).create();//创建
              dialog.show();//显示
          }});
      }
  }
```

在例子中点击“评论”按钮时将会提示登录对话框，点击确定时就会跳转到登录框中，如图 9-41 所示。

图 9-41　登录示意图

在上面的代码中需要注意：

```
factory = LayoutInflater.from(CH93224.this);//用来将 xml 文件转成 View
final View DialogView = factory.inflate(R.layout.login, null);
```

东软电子出版社

上面两行代码将设置的布局文件 login.xml 用 LayoutInflater 对象的 inflate 方法转换成 View。

从 CH93224 例子中可以看到用 AlertDialog 来创建对话框是非常方便的，使用 AlertDialog.Builder 可以很方便地创建指定内容及样式的对话框。

AlertDialog.Builder 还提供了多种方法来定制对话框，具体如下：

- setTitle(CharSequence title)，setTitle(int titleId)，设置标题字符串。
- setSingleChoiceItems，设置为单选项对话框。
- setMultiChoiceItems，设置为多选选项对话款。
- setItems，设置为选项对话框，不区分多选单选。
- setPositiveButton(int textId，DialogInterface.OnClickListener listener)。
- setPositiveButton(CharSequence text，DialogInterface.OnClickListener listener)。
- setNegativeButton(CharSequence text，DialogInterface.OnClickListener listener)。
- setNegativeButton(int textId，DialogInterface.OnClickListener listener)。
- setNeutralButton(CharSequence text，DialogInterface.OnClickListener listener)。
- setNeutralButton(int textId，DialogInterface.OnClickListener listener)。

以上方法均是为对话框设置按钮，最多可以设置三个按钮。文档所谓的 Positive，negative 和 neutral 只是为了便于区分，没有特殊的含义。

(5)通知(Notification & Toast)。

Toast 是 Android 平台下用来向用户显示信息的一种机制。Toast 是没用焦点的，当 Toast 出现的时候用户仍然可以在原来的 Activity 上进行输入，在显示一定的时间后自动消失。前面的例子中已经用到了好多次 Toast 来提示信息，下面结合实例来深入了解 Toast 的工作机制。在这个例子中需要实现如下功能：我们在收到短信时会用 Toast 来提示用户短信的内容。

Toast 的使用很简单，只有短短的一行代码：Toast.makeText(Context context，CharSequence text，int duration)。其中 context 是用来显示 Toast 的上下文，text 是 Toast 中显示的内容，duration 是 Toast 显示停留的时间长度，有 Toast.LENGTH_SHORT 和 Toast.LENGTH_LONG 两种时间。下面看一下这个例子的主要代码：

```
public class CH93225 extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button btnTest = (Button)findViewById(R.id.test);
        btnTest.setOnClickListener(new Button.OnClickListener(){
        public void onClick(View v){
           Toast toast = Toast.makeText(CH93225.this,"信息在这里显示",Toast.LENGTH_
SHORT);
          toast.setGravity(Gravity.TOP, 0, 200);
          toast.show();
        }
```

东软电子出版社

```
        });
    }
}
```

上面的代码中使用了一个按钮来查看短信提示信息显示的位置，通过 setOnClickListener()来监听按钮事件。在监听事件中使用了 Toast 来显示信息，以告诉用户 Toast 显示的位置。程序运行如图 9-42 所示。

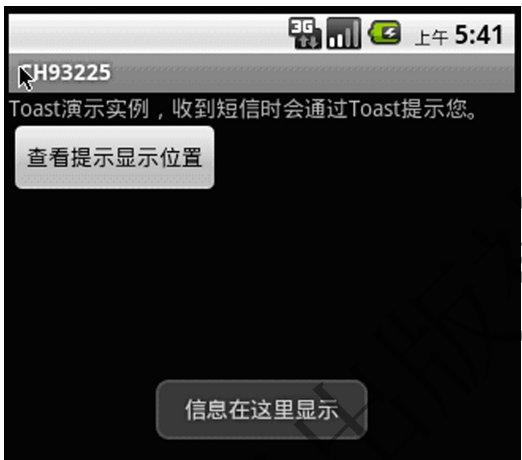

图 9-42　Toast 显示位置

为了能够显示短信信息，还需要创建一个接受短信的类，继承自 BroadcastReceiver，这里我们命名为 SMSReceiver。我们需要重写里面的 onReceive()方法，在 onReceive()中使用 Toast，当有短信到达时就用 Toast 显示短信内容。代码如下所示：

```
public class SMSReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context arg0, Intent arg1){
        // TODO Auto-generated method stub
        Bundle bundle = arg1.getExtras();
        Object message[] = (Object[])bundle.get("pdus");
        SmsMessage smsMessage[] = new SmsMessage[message.length];
        for (int n = 0;n<message.length;n++){
        smsMessage[n] = SmsMessage.createFromPdu((byte[])message[n]);
        }
        Toast toast = Toast.makeText(arg0, "短信内容:" + smsMessage[0].getMessageBody(),
Toast.LENGTH_LONG);
        toast.setGravity(Gravity.TOP, 0, 200);
        toast.show();
    }
}
```

由于这个实例中使用了短信接口，所以要在 AndroidManifest.xml 中声明其权限。代码如下所示：

```
<? xml version = "1.0" encoding = "utf-8"? >
<manifest xmlns:android = "http://schemas.android.com/apk/res/android"
      package = "com.study.chapter9"
      android:versionCode = "1"
      android:versionName = "1.0">
    <application android:icon = "@drawable/icon" android:label = "@string/app_name">
        <activity android:name = ".CH93225"
                  android:label = "@string/app_name">
            <intent-filter>
                <action android:name = "android.intent.action.MAIN" />
                <category android:name = "android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver android:name = ".SMSReceiver" android:enabled = "true">
        <intent-filter>
        <action android:name = "android.provider.Telephony.SMS_RECEIVED"/>
        </intent-filter>
        </receiver>
    </application>
    <uses-sdk android:minSdkVersion = "8" />
<uses-permission android:name = "android.permission.RECEIVE_SMS">
</uses-permission>
</manifest>
```

完成上面的设置后运行程序，此时还需要再启动一个模拟器来发送短信，当然我们可以切换到 DDMS 环境下来给模拟器 1 发送短信。这里用 DDMS 来向模拟器发送短信。如图 9-43 所示，通过 Emulator Control 来向模拟器 5554 来发送短信，短信内容：Toast，testing.

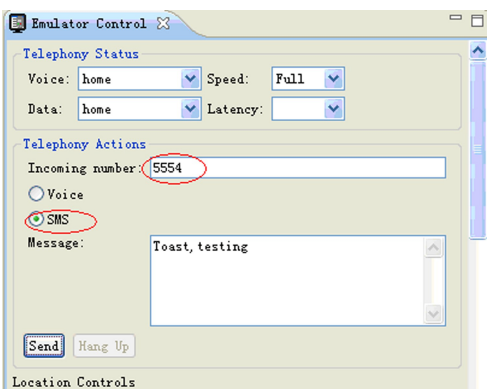

图 9-43　通过 Emulator Control 发送短信

东软电子出版社

点击“Send”发送短信，这时候在模拟器查看一下程序运行情况，如图 9-44 所示。

图 9-44　模拟器收到短信后提示界面

(6)下拉列表(Spinner)。

提到 Spinner 大家可能没有听过，但是如果说下拉列表大家肯定再熟悉不过了。熟悉 J2ee 的读者可能会想到 swing 的 combo box，熟悉网页设计的人自然会想到 html 中的＜select＞。Spinner 的主要功能就是方便用户，能够让用户不必填写一些琐碎的信息，如血型，而是通过 Spinner 来进行选择。我们通过下面实例演示关于 Spinner 的用法。程序运行效果如图 9-45 所示。

图 9-45　Spinner 下拉菜单

在布局文件中需要使用标签＜Spinner＞来将 Spinner 组件添加到布局文件中。在本例中还需要一个 TextView 来显示所选择的选项，需要一个 EditText 来输入我们所要添加的信息，然后需要两个按钮分别来处理添加和删除事件。布局文件代码如下所示：

```
＜? xml version = "1.0" encoding = "utf-8"? ＞
＜LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
```

东软电子出版社

```
        android:layout_height = "fill_parent"
        >
    <TextView
        android:id = "@ + id/cityName"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        />
    <EditText
        android:id = "@ + id/etCity"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        />
    <Button
        android:id = "@ + id/btnAdd"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "添加"/>
    <Button
        android:id = "@ + id/btnDel"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "删除"/>
    <Spinner
        android:id = "@ + id/mySpinner"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        />
    </LinearLayout>
```

在编码实现时，首先需要在布局中引入 Spinner 组件，然后将可选内容通过 ArrayAdapter 和下拉列表连接起来，设置按钮监听来监听添加和删除事件。更重要的是学习设计事件监听 setOnItemSelectedListener 并实现 onItemSelected 来获得用户所选择的内容。需要注意的是要通过 ArrayList 来将预先声明的静态数组添加到 ArrayList 对象中。代码如下所示：

```
//包引入语句省略
public class CH93226 extends Activity {
    private static final String[] cities = {"北京","新德里","马德里","渥太华"};
    private TextView tvCity;
    private EditText etCity;
    private Button btnAdd;
    private Button btnDel;
    private Spinner citySpinner;
```

东软电子出版社

```
    private ArrayAdapter <String> adapter;
    private ArrayList<String> cityAll;
    /* * Called when the activity is first created. * /
    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    //获取我们定义的对象
        tvCity = (TextView)findViewById(R.id.cityName);
        etCity = (EditText)findViewById(R.id.etCity);
        btnAdd = (Button)findViewById(R.id.btnAdd);
        btnDel = (Button)findViewById(R.id.btnDel);
        citySpinner = (Spinner)findViewById(R.id.mySpinner);
      //将数组中的信息放到 List 列表中
      cityAll = new ArrayList<String>();
      for(int i = 0;i<cities.length;i + + ){
      cityAll.add(cities[i]);
      }
      //新建 ArrayAdapter 对象,并传入 cityAll 列表
         adapter = new ArrayAdapter<String>(this,android.R.layout.simple_spinner_
item,cityAll);
      //设置下来菜单的样式
         adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_
item);
      //将我们定义好的 adapter 添加到 Spinner 对象中
        citySpinner.setAdapter(adapter);
      //对添加按钮设置按钮监听事件
        btnAdd.setOnClickListener(new Button.OnClickListener(){

          @Override
          public void onClick(View v){
            String newCity = etCity.getText().toString();//取得编辑框中输入的内容
            /*判断输入内容是否已经存在*/
            for(int i = 0;i<adapter.getCount();i + + ){
              if(newCity.equals(adapter.getItem(i))){
                Toast.makeText(CH93226.this,"该城市已经存在",Toast.LENGTH_LONG).
show();

                return;}
            }
            if(newCity.equals("")){
              Toast.makeText(CH93226.this,"城市不能为空",Toast.LENGTH_LONG).show();
              return;
```

东软电子出版社

```
            }else{//若输入内容不为空
              adapter.add(newCity);//将我们输入的内容添加到 adapter 对象中
              int position = adapter.getPosition(newCity);//取得添加内容的位置
              citySpinner.setSelection(position);//在 spinner 中选择新加入的内容
              etCity.setText("");
            }
          }});
        //对删除按钮设置按钮监听事件
        btnDel.setOnClickListener(new Button.OnClickListener(){

          @Override
          public void onClick(View v){
            if(citySpinner.getSelectedItem()! = null){
    adapter.remove(citySpinner.getSelectedItem().toString());//从 adapter 中删除选择的项
              etCity.setText("");
              if(adapter.getCount() = = 0){
                tvCity.setText("");
              }
            }
          }});
        /*
          * 对 Spinner 对象设置 onItemSelectedListener()监听
          * 将我们选择的项显示到顶部的 TextView 中
          */
        citySpinner.setOnItemSelectedListener(new Spinner.OnItemSelectedListener(){

          @Override
          public void onItemSelected(AdapterView<? > arg0, View arg1,
              int arg2, long arg3){
            tvCity.setText(arg0.getSelectedItem().toString());
          }

          @Override
          public void onNothingSelected(AdapterView<? > arg0){

          }});
      }
  }
```

在上面的代码中需要解释以下几点：

①首先我们通过 adapter = new ArrayAdapter<String>(this, android.R.layout.simple_spinner_item, cityAll)来将 ArrayList 对象 cityAll 添加到新建的 ArrayAdapter 中，并指定其 textViewResourceId。

东软电子出版社

②adapter. setDropDownViewResource(android. R. layout. simple_spinner_dropdown_item)；通过 setDropDownViewResource 函数来设置下拉列表的样式，当然可以自己设计样式，并放到 layout 文件夹下。Android 提供了两个基本的样式：

• android. R. layout. simple_spinner_item：TextView 的下拉菜单。

• android. R. layout. simple_spinner_dropdown_item：右边带有 radio 的下拉菜单。

③citySpinner. setAdapter(adapter)；在设置好了 adapter 后需要为 Spinner 对象来指定 ArrayAdapter 对象。

在添加和删除按钮的监听事件中我们是对 adapter 对象来动态添加和删除下拉菜单的选项。同时我们还要注意在添加时判断是否已经存在，我们用循环通过 equals(adapter. getItem(i))来比较下拉列表中的所有选项与我们所要添加的选项是否相同。

3. 界面布局管理

在 Android SDK 中主要包含以下几种布局：

(1)LinearLayout：线性布局，是我们最常用的一种布局方式。分为水平线性布局和垂直线性布局，在线性布局里面我们可以放多个控件，但是一行或者一列只能放一个控件。

(2)RelativeLayout：相对布局，与线性布局一样，在里面我们可以放多个控件，但是每个控件的位置都是相对的。我们可以定义每一个子 View 与其他子 View 之间以及与屏幕边界之间的相对位置。

(3)TableLayout：表格布局，在表格布局中可以使用多行多列的表格来布局，View 可以跨越多行和多列，而且列可以设置为收缩或者增大的。

(4)FrameLayout：单帧布局，是最简单的布局管理器，它只是把控件放置在 View 的左上角，当我们添加一个新的 View 子类时，它会把每一个新的子 View 放到最上层。

(5)AbsoluteLayout：绝对布局或者坐标布局，顾名思义，在这个方式下的子 View 的位置都是绝对的。使用这个类的好处是我们可以使我们的布局更加精确，但是却丧失了它的自适应的能力。

(6)TabWidget：切换卡，这是个特殊的布局模式，主要功能是实现标签切换，类似于 Android 系统“联系人”和“通话记录”的样式。

9.3.3　数据持久化存储

1. 文件存储

Android 中可以在设备本身的存储设备或者外接的存储设备中创建用于保存数据的文件。在默认的情况下，文件是不能在不同的程序间共享的。由于 Android 是以 Linux 为内核的，所以其 File 的形式也是 Linux 下的形式。用文件来存储数据可以通过 openFileOutput 方法打开一个文件，然后通过 Load 方法来获取文件中的数据，通过 deleteFile 方法可以删除一个指定的文件。

在我们进入 Files 的实例演示之前，我们需要了解一个概念——Properties(属性)，我们可以把 Properties 继承自 Hashtable，理解成一个 Hashtable ，不过唯一不同的是，Properties 对应的“键-值”必须是字符串形式的数据类型。Files 数据存储主要是使用 Properties 配合 FileInputStream 或者 FileOutputStream 对文件写入操作。表 9-4 列举

东软电子出版社

出了一些 Properties 常用的方法。

表 9-4 Properties 常用方法

方法	说明
getProperty(String name, String defaultValue)	通过指定的“name”即 Key,搜索属性,参数二为默认值,即通过 Key 找不到文件中的属性时,要返回的默认值。返回值为 String
list(PrintStream out)	通过 PrintStream 列出可读的属性列表,无返回值
list(PrintWriter writer)	通过 PrintStream 列出可写的属性列表,无返回值
save(OutputStream out, String comment)	注意,这种保存方法已经过时,Google 不推荐使用此种写法,这种方法忽略任何 IO 异常,所以在实际操作过程中,可能会发生不必要的异常。
setProperty(String name, String value)	设置属性,保存一个”键-值“对的属性
store(OutputStream out, String comment)	通过 FileOutputStream 打开对应的程序文件,然后通过 Store 保存之前 Properties 打包好的数据。这里备注可以为空
storeToXML(OutputStream os, String comment)	通过 FileOutputStream 打开对应的程序文件,将打包好的数据写入到 XML 文件
storeToXML(OutputStream os, String comment, String encoding)	通过 FileOutputStream 打开对应的程序文件,将打包好的数据写入到 XML 文件,第三个参数可以指定编码

下面来看一下 Files 在 Android 数据存储中的应用。其中我们在布局上做了修改,但是所实现的功能是相同的。可以对比一下 SharedPreferences 和 Files 在实现上的区别。首先看一下效果图,如图 9-46 和图 9-47 所示。

图 9-46 程序启动初始界面

图 9-47 程序再次启动时默认账号和密码

东华电子出版社

下面看一下布局文件的代码：

```
<? xml version = "1.0" encoding = "utf-8"? >
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
android:layout_width = "fill_parent" android:layout_height = "fill_parent"
android:gravity = "right" android:layout_gravity = "right"
android:background = "@drawable/bg" android:orientation = "vertical">
<TableLayout android:layout_width = "fill_parent"
  android:layout_height = "wrap_content" android:stretchColumns = "1">
  <TableRow android:gravity = "center" android:layout_gravity = "center">
  <ImageView android:layout_width = "fill_parent"
    android:layout_height = "wrap_content" android:id = "@+id/ivlogo"
    >
  </ImageView>
  </TableRow>
</TableLayout>
<TableLayout android:layout_width = "fill_parent"
  android:layout_height = "wrap_content" android:stretchColumns = "1">
  <TableRow android:layout_marginTop = "100dip">
  <TextView android:layout_width = "wrap_content"
    android:layout_marginLeft = "20dip" android:gravity = "center_vertical"
    android:layout_height = "wrap_content" android:id = "@+id/tvaccount"
    android:text = "帐号:" android:textSize = "20sp">
  </TextView>
  <EditText android:layout_width = "70px" android:layout_height = "wrap_content"
    android:id = "@+id/etaccount" android:layout_marginRight = "20dip"
    android:maxLength = "20"></EditText>
  </TableRow>
  <TableRow android:layout_marginTop = "10dip">
  <TextView android:layout_width = "wrap_content"
    android:layout_height = "wrap_content" android:id = "@+id/tvpw"
    android:layout_marginLeft = "20dip" android:gravity = "center_vertical"
    android:text = "密码:" android:textSize = "20sp">
  </TextView>
  <EditText android:layout_width = "70px" android:layout_height = "wrap_content"
    android:layout_marginRight = "20dip" android:id = "@+id/etpw"
    android:inputType = "textPassword"></EditText>
  </TableRow>
</TableLayout>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
  android:layout_width = "wrap_content" android:layout_height = "wrap_content"
  android:orientation = "horizontal" android:layout_marginTop = "5dip" android:layout_
marginRight = "20dip">

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

东软电子出版社

```
    android:text = "清除 Cookies" android:textColor = "#aa0000" android:textSize = "12px"
></TextView>

  </LinearLayout>
  <TableLayout android:layout_width = "fill_parent"
    android:layout_height = "wrap_content" android:layout_marginTop = "20dip">
    <TableRow android:gravity = "center" android:layout_width = "fill_parent">
    <Button android:layout_width = "100px" android:layout_height = "wrap_content"
      android:id = "@+id/btnlogin" android:layout_gravity = "center"
      android:text = "登录"></Button>
    <Button android:layout_width = "100px" android:layout_height = "wrap_content"
      android:id = "@+id/btnexit" android:layout_gravity = "center"
      android:text = "退出"></Button>
    </TableRow>
  </TableLayout>
  <LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "wrap_content" android:layout_height = "wrap_content"
    android:orientation = "horizontal" android:layout_marginTop = "25dip">
    <CheckBox android:layout_width = "wrap_content"
    android:layout_height = "wrap_content" android:id = "@+id/remeberPWD"
    android:text = "记住密码" android:textSize = "12px"></CheckBox>
    <CheckBox android:layout_width = "wrap_content"
    android:layout_height = "wrap_content" android:id = "@+id/autoLogin"
    android:text = "自动登录" android:textSize = "12px"></CheckBox>
  </LinearLayout>
  </LinearLayout>
```

上面的布局文件是多个样式的整合，正如前面说过的，可以借助工具来进行布局。下面了解下主程序的代码：

```
public class CH9331 extends Activity {
    private CheckBox remeberPWD;
    private CheckBox autoLogin;
    private EditText extUser;
    private EditText extPassword;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        remeberPWD = (CheckBox)findViewById(R.id.remeberPWD);
        autoLogin = (CheckBox)findViewById(R.id.autoLogin);
        extUser = (EditText)findViewById(R.id.etaccount);
        extPassword = (EditText)findViewById(R.id.etpw);

        load();
```

东软电子出版社

```
            remeberPWD
              .setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener(){
              @Override
              public void onCheckedChanged(CompoundButton buttonView,
                boolean isChecked){
            if(remeberPWD.isChecked()){
              save();
            }}
              });
            autoLogin
              .setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener(){
              @Override
              public void onCheckedChanged(CompoundButton buttonView,
                boolean isChecked){

              }
            });
        }
        //初始化操作
        private void load(){
          // TODO Auto-generated method stub
          Properties properties = new Properties();
        try{
          FileInputStream stream = this.openFileInput("user.cfg");
          properties.load(stream);
        }catch(FileNotFoundException e){
          return;
        }catch(IOException e){
          return;
        }
        //取得数据
      //Toast.makeText(this,properties.getProperty("account").toString(), Toast.LENGTH_
LONG).show();
       extUser.setText(properties.getProperty("account").toString());
      extPassword.setText(properties.getProperty("password").toString());
       }

       private boolean save(){
         Properties properties = new Properties();
         //将数据打包成 Properties
         properties.put("account",extUser.getText().toString());
         properties.put("password",extPassword.getText().toString());
         try{
           FileOutputStream stream = this.openFileOutput("user.cfg", Context.MODE_WORLD_
WRITEABLE);
           properties.store(stream,"");
```

东软电子出版社

```
        }catch(FileNotFoundException e){
          return false;
        }catch(IOException e){
          return false;
        }
        return true;
    }
}
```

若文件不存在或不能以可写的方式打开，则会抛出 FileNotFoundException 异常。若文件不存在也可以在打开文件 OutputStream 的同时创建文件，即使用 Context 对象调用 FileOutputStream openFileOutput(String name,int mode). 参数，name 为目录下文件名为 name 的 OutputStream，mode 为 Context 类中定义的一个常量，表示用什么模式打开或创建文件，当需要用到多个模式打开时用“|”隔开。其中 mode 的值为：

(1)MODE_APPEND，以在文件末尾写入数据这种模式打开文件。

(2)MODE_PRIVATE，以仅仅只有应用程序自己可读可写的模式创建文件。

(3)MODE_WORLD_READABLE，以其他应用程序可读的模式创建文件。

(4)MODE_WORLD_WRITEABLE，以其他应用程序对文件可写的模式创建文件。

在大多数情况下，我们可以通过 Context 的对象调用 openFileOutput()和 openFileInput()来直接打开或创建位于私有目录的文件。另外，Context 对象还可以通过调用 fileList()方法来获得私有文件目录下所有文件的文件名组成的字符串数组。

与 SharedPreferences 的存储位置类似，我们用 Files 来存储的数据也是放在/data/data/＜包名＞/files/下面，如图 9-48 所示。

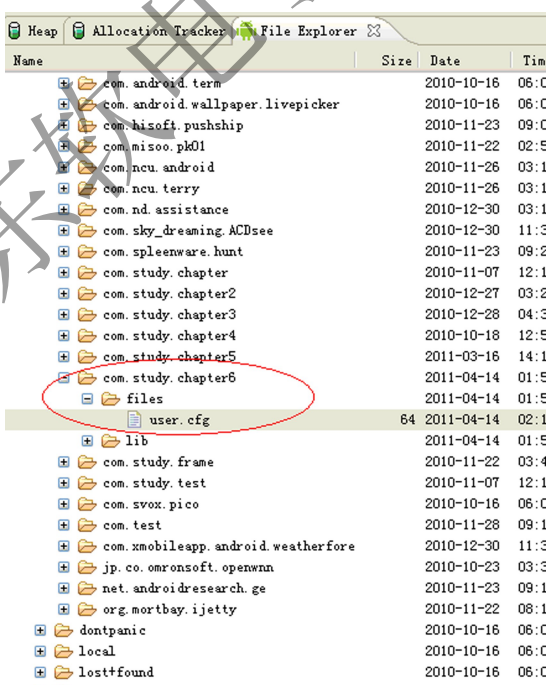

图 9-48　files 存放位置

东软电子出版社

2. SharePreference

Shared Preferences 类似于我们常用的 ini 文件，用来保存我们在应用程序中的一些属性设置，在 Android 平台常用于存储简单的参数设置。我们可以用来保存上一次用户所做的修改或者自定义参数设定，当再次启动程序时仍然保持原有的设置。下面通过一个实例来看一下 SharedPreferences 的具体使用方法。在这个实例中我们定义了两个 TextView 和两个 EditText，让用户输入用户名和密码。然后通过 Shared Preferences，在下次启动程序时自动默认的是用户设置过的用户名和密码。实例初始运行状态图如图 9-49 所示，然后在文本框中分别输入 user 和 password，按返回键，当再次进入程序时，界面如图 9-50 所示。

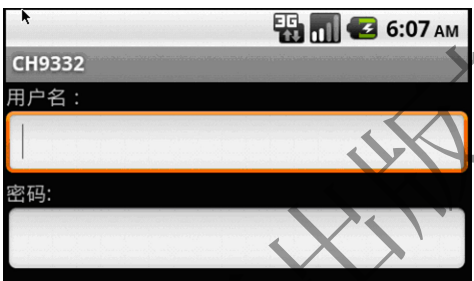

图 9-49　程序初始运行截图

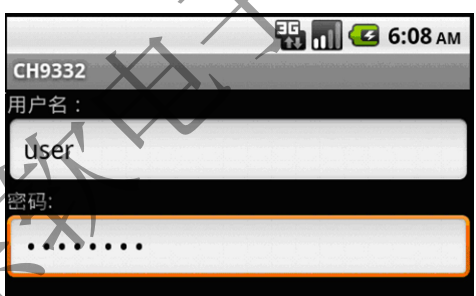

图 9-50　再次启动程序截图

使用 SharedPreferences 保存 key-value 对的步骤如下：

(1)使用 Activity 类的 getSharedPreferences 方法获得 SharedPreferences 对象，其中存储 key-value 的文件的名称由 getSharedPreferences 方法的第一个参数指定。

(2)使用 SharedPreferences 接口的 edit 获得 SharedPreferences. Editor 对象。

(3)通过 SharedPreferences. Editor 接口的 put 方法保存 key-value 对。其中 Xxx 表示不同的数据类型。例如：字符串类型的 value 需要用 putString 方法。

(4)通过 SharedPreferences. Editor 接口的 commit 方法保存 key-value 对。commit 方法相当于数据库事务中的提交(commit)操作。

表 9-5 中列出了两个获得 SharedPreferences 的方法。

东软电子出版社

表 9-5 获取 SharedPreferences 的方法

返回值	函数	备注
SharedPreferences	Context. getSharedPreferences (String name,int mode)	name 为本组件的配置文件名(如果想要与本应用程序的其他组件共享此配置文件,可以用这个名字来检索到这个配置文件)。 mode 为操作模式,默认的模式为 0 或 MODE_PRIVATE,还可以使用 MODE_WORLD_READABLE 和 MODE_WORLD_WRITEABLE
SharedPreferences	Activity. getPreferences (int mode)	配置文件仅可以被调用的 Activity 使用。mode 为操作模式,默认的模式为 0 或 MODE_PRIVATE,还可以使用 MODE_WORLD_READABLE 和 MODE_WORLD_WRITEABLE

除此之外,我们还将介绍几个重要的方法:

(1)public abstract boolean contains (String key):检查是否已存在该文件,其中 key 是 xml 的文件名。

(2)edit():为 preferences 创建一个编辑器 Editor,通过创建的 Editor 可以修改 preferences 里面的数据,但必须执行 commit()方法。

(3)getAll():返回 preferences 里面的多有数据。

(4)getBoolean(String key, boolean defValue):获取 Boolean 型数据。

(5)getFloat(String key, float defValue):获取 Float 型数据。

(6)getInt(String key, int defValue):获取 Int 型数据。

(7)getLong(String key, long defValue):获取 Long 型数据。

(8)getString(String key, String defValue):获取 String 型数据。

(9) registerOnSharedPreferenceChangeListener (SharedPreferences. OnSharedPreferenceChangeListener listener):注册一个当 preference 发生改变时被调用的回调函数。

(10) unregisterOnSharedPreferenceChangeListener(SharedPreferences. OnSharedPreferenceChangeListener listener):删除当前回调函数。

在理解了 SharedPreferences 的基本知识后来看一下程序的实现方法。由于布局文件比较简单,这里就不再列出布局文件的代码清单。程序中主程序源代码如下所示:

```
public class CH9332 extends Activity {
public String userName,passWord;
public EditText extName,PWD;
  /* * Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState){
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      extName = (EditText)findViewById(R.id.extUser);
      PWD = (EditText)findViewById(R.id.extPWD);
      SharedPreferences  user = getPreferences(Activity.MODE_PRIVATE);
```

东软电子出版社

```
            userName = user.getString("user_info", "");
            passWord = user.getString("password", "");
            extName.setText(userName);
            PWD.setText(passWord);
        }
    //当按下返回键时我们将我们在文本框中输入的内容保存到 Preferences 中。
    public boolean onKeyDown(int keyCode,KeyEvent event){
        if(keyCode = = KeyEvent.KEYCODE_BACK){
            SharedPreferences user = getPreferences(0);
            SharedPreferences.Editor editor = user.edit();
            editor.putString("user_info", extName.getText().toString());
            editor.putString("password",PWD.getText().toString());
            editor.commit();
        this.finish();
    return true;
        }
        return super.onKeyDown(keyCode, event);
    }
}
```

下面看一下数据的保存。Eclipse 下切换到 DDMS 视图，选择 File Explorer 标签。找到/data/data 目录中对应的项目文件夹下的 Shared_prefs 文件夹。所看到的 xml 文件就是保存数据的地方了，如图 9-51 所示。

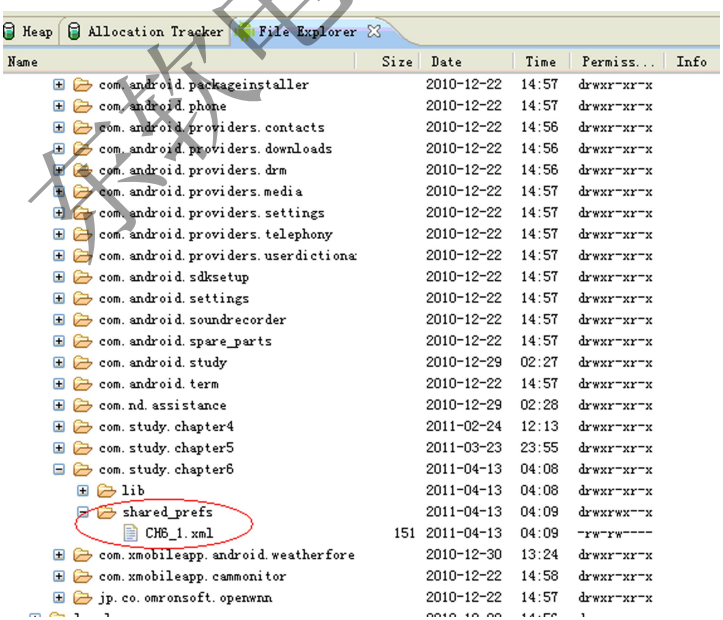

图 9-51　Preferences 数据存储目录

东软电子出版社

9.3.4 网络通讯

1. HTTP 通信

HTTP(Hyper Text Transfer Protocol,超文本传输协议)用于传送 WWW 方式的数据。HTTP 协议采用了请求/响应模型。客户端向服务器发送一个请求,请求头包含了请求的方法、URI、协议版本,以及包含请求修饰符、客户信息和内容类似于 MIME 的消息结构。服务器以一个状态行作为响应,响应的内容包括消息协议的版本、成功或者错误编码,还包括服务器信息、实体元信息以及可能的实体内容。

Google 以网络搜索引擎著称,自然而然也会使 Android SDK 拥有强大的 HTTP 访问能力。在 Android SDK 中,Google 集成了 Apache 的 HttpClient 模块。要注意的是,这里的 Apache HttpClient 模块是 HttpClient4.0(org.apache.http.*),而不是 Jakarta Commons HttpClient 3.x(org.apache.commons.httpclient.*)。

HTTP 通信中使用最多的就是 Get 和 Post。Get 请求方式中,参数直接放在 URL 字串后面,传递给服务器。格式如下:

```
HttpGet method = new HttpGet("http://www.baidu.com? admin=Get");
HttpResponse response = client.execute(method);
```

而 Post 请求方式中,参数必须采用 NameValuePair[]数组的传送方式。格式如下:

```
HttpPost method = new HttpPost("http://www.baidu.com");
List<NameValuePair> params = new ArrayList<NameValuePair>();
params.add(new BasicNameValuePair("admin", "Get"));
method.setEntity(new UrlEncodedFormEntity(params));
HttpResponse response = client.execute(method);
```

在这两种通信方式中,一般情况下,两种方式实现的效果一样。但也有特殊情况,可能服务器只支持 GET 的请求方式,而不支持 POST 的请求方式,所以导致 POST 请求方式获取不到需要的数据;也可能服务器只支持 POST 的请求方式,不支持 GET 的请求方式。于是,我们需要查看服务器返回的状态码,如果是 200 则证明连接成功,否则连接失败。状态码的取得方式可以通过抓包观察,也可以直接用代码获取。用代码获得服务器返回的状态码具体参照为:

```
HttpResponse httpResponse = new DefaultHttpClient().execute(method);
If(httpResponse.getStatusLine().getStatusCode() == 200)
{
    // TODO: get data from URL
}
else
{
    // TODO: show connection false
}
```

这里需要注意的是,由于 Android 的很多操作都涉及到权限的问题,Android 尝试连续网络时,是需要权限的。所以加入网络连接权限:

东软电子出版社

在 AndroidManifest. xml 中添加

```
<uses-permission android:name = "android. permission. INTERNET" />
```

2. HttpURLConnection

在 Android 的 SDK 中，Google 同时也继承了网络连接中标准的 Java 接口，使得最基本的一些连接方式得以继续沿用。注意，URLConnection 与 HttpURLConnection 都是抽象类，无法直接实例化对象。其对象主要通过 URL 的 openConnection 方法获得。

标准的 Java 接口格式如下：

```
URL url = new URL("http://www.baidu.com");
HttpURLConnection http = (HttpURLConnection)url.openConnection();
int response = http.getResponseCode();
if(200 = = response)
{
  //TODO: get data from URL
}
else
{
// TODO: show connection false
}
```

在上述方法中，如果所连接的网址不存在，会报出 java. net. UnknownHostException 异常，所以连接需要 try catch。

3. HttpClient

在 Android 开发中经常会用到网络连接功能与服务器进行数据的交互，为此 Android 的 SDK 提供了 Apache 的 HttpClient 来方便我们使用各种 Http 服务。你可以把 HttpClient 想象成一个浏览器，通过它的 API 我们可以很方便地发出 GET，POST 请求。

只需以下几行代码就能发出一个简单的 GET 请求并打印响应结果：

```
try {
      // 创建一个默认的 HttpClient
      HttpClient httpclient = new DefaultHttpClient();
      // 创建一个 GET 请求
      HttpGet request = new HttpGet("www.google.com");
      // 发送 GET 请求，并将响应内容转换成字符串
      String response = httpclient.execute(request, new BasicResponseHandler());
      Log.v("response text", response);
} catch (ClientProtocolException e){
      e.printStackTrace();
} catch (IOException e){
      e.printStackTrace();
}
```

这只是一段演示代码，实际的项目中的请求与响应处理会复杂一些，并且还要考虑到代码的容错性，但是这并不是本篇的重点。注意代码的第三行：

```
HttpClient httpclient = new DefaultHttpClient();
```

东软电子出版社

在发出 HTTP 请求前，先创建了一个 HttpClient 对象。在实际项目中，很可能在多处需要进行 HTTP 通信，这时候不需要为每个请求都创建一个新的 HttpClient。因为之前已经提到，HttpClient 就像一个小型的浏览器，对于整个应用，只需要一个 HttpClient 对象，具体如下所示。

```
public class CustomerHttpClient {
    private static HttpClient customerHttpClient;
    private CustomerHttpClient(){
    }

    public static HttpClient getHttpClient(){
        if(null = = customerHttpClient){
            customerHttpClient = new DefaultHttpClient();
        }
        return customerHttpClient;
    }
}
```

应用程序使用同一个 HttpClient 来管理所有的 HTTP 请求，一旦出现并发请求，会出现多线程的问题。HttpClient 提供了创建线程安全对象的 API，解决了多线程安全问题。

解决多线程问题：

```
public class CustomerHttpClient {
    private static final String CHARSET = HTTP.UTF_8;
    private static HttpClient customerHttpClient;
      private CustomerHttpClient(){
    }
      public static synchronized HttpClient getHttpClient(){
       if (null = = customerHttpClient){
            HttpParams params = new BasicHttpParams();
            // 设置一些基本参数
                    HttpProtocolParams.setVersion(params, HttpVersion.HTTP_1_1);
            HttpProtocolParams.setContentCharset(params,
                CHARSET);
            HttpProtocolParams.setUseExpectContinue(params, true);
            HttpProtocolParams
                .setUserAgent(
                    params,
                            "Mozilla/5.0(Linux;U;Android 2.2.1;en-us;Nexus One Build.
                            FRG83)" + "AppleWebKit/553.1(KHTML,like Gecko)Version/4.0
```

东软电子出版社

```
                    Mobile Safari/533.1");
            // 超时设置
            /* 从连接池中取连接的超时时间 */
            ConnManagerParams.setTimeout(params, 1000);
            /* 连接超时 */
            HttpConnectionParams.setConnectionTimeout(params, 2000);
            /* 请求超时 */
            HttpConnectionParams.setSoTimeout(params, 4000);

            // 设置我们的 HttpClient 支持 HTTP 和 HTTPS 两种模式
            SchemeRegistry schReg = new SchemeRegistry();
            schReg.register(new Scheme("http", PlainSocketFactory
                    .getSocketFactory(), 80));
            schReg.register(new Scheme("https", SSLSocketFactory
                    .getSocketFactory(), 443));
                // 使用线程安全的连接管理来创建 HttpClient
                ClientConnectionManager conMgr = new ThreadSafeClientConnManager(
                params, schReg);
            customerHttpClient = new DefaultHttpClient(conMgr, params);
        }
        return customerHttpClient;
    }
}
```

在上面的 getHttpClient()方法中，我们为 HttpClient 配置了一些基本参数和超时设置，然后使用 ThreadSafeClientConnManager 来创建线程安全的 HttpClient。

4. SAX(XML)

SAX，全称 Simple API for XML，既是指一种接口，也是指一个软件包。SAX 最初是由 David Megginson 采用 Java 语言开发，之后 SAX 很快在 Java 开发者中流行起来。San 现在负责管理其原始 API 的开发工作，这是一种公开的、开放源代码软件。不同于其他大多数 XML 标准的是，SAX 没有语言开发商必须遵守的标准 SAX 参考版本。因此，SAX 的不同实现可能采用区别很大的接口。

作为接口，SAX 是事件驱动型 XML 解析的一个标准接口(standard interface)，不会改变，已被 OASIS(Organization for the Advancement of Structured Information Standards)所采纳。作为软件包，SAX 最早的开发始于 1997 年 12 月，由一些在互联网上分散的程序员合作进行。后来，参与开发的程序员越来越多，组成了互联网上的 XML-DEV 社区。5 个月以后，1998 年 5 月，SAX 1.0 版由 XML-DEV 正式发布。目前，最新的版本是 SAX 2.0。2.0 版本在多处与 1.0 版本不兼容，包括一些类和方法的名字。

东软电子出版社

由前面介绍可知，SAX 是一种事件驱动的接口，它的基本原理是由接口的用户提供符合定义的处理器，XML 分析时遇到特定的事件，就去调用处理器中特定事件的处理函数。一般 SAX 接口都是用 Java 实现的，但事实上 C++也可以用于实现 SAX 接口，只是 C++的分析器比较少。之所以叫做“简单”应用程序接口，是因为这个接口确实非常简单，绝大多数事情分析器都没有做，需要应用程序自己去实现，因而开发者的任务也相应重一些。

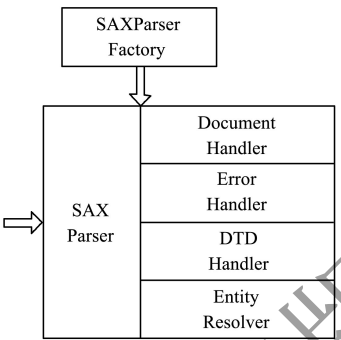

图 9-52　SAX 解析器框架

图 9-52 中的 SAXParserFactory 用来生成一个分析器实例。XML 文档是从左侧箭头所示处读入，当分析器对文档进行分析时，就会触发在 DocumentHandler，ErrorHandler，DTDHandler 以及 EntityResolver 接口中定义的回调方法。

下面对 SAX 分析器中的几个主要 API 接口做简单的介绍。

(1)ContentHandler 接口。

ContentHandler 是 Java 类包中一个特殊的 SAX 接口，位于 org. xml. sax 包中。该接口封装了一些对事件处理的方法，当 XML 解析器开始解析 XML 输入文档时，它会遇到某些特殊的事件，比如文档的开头和结束、元素开头和结束、以及元素中的字符数据等事件。当遇到这些事件时，XML 解析器会调用 ContentHandler 接口中相应的方法来响应该事件。ContentHandler 接口的方法有以下几种：

①void startDocument()。

②void endDocument()。

③ void startElement (String uri, String localName, String qName, Attributes atts)。

④void endElement(String uri, String localName, String qName)。

⑤void characters(char[ ] ch, int start, int length)。

(2)DTDHandler 接口。

DTDHandler 用于接收基本的 DTD 相关事件的通知。该接口位于 org. xml. sax 包中。此接口仅包括 DTD 事件的注释和未解析的实体声明部分。SAX 解析器可按任何顺序报告这些事件，而不管声明注释和未解析实体时所采用的顺序；但是，必须在文档处理程序的 startDocument()事件之后，在第一个 startElement()事件之前报告所有的 DTD 事件。

东软电子出版社

DTDHandler 接口包括以下两个方法：

① void startDocumevoid notationDecl (String name, String publicId, String systemId)nt()。

②void unparsedEntityDecl(String name, String publicId, String systemId, String notationName)。

EntityResolver 接口

EntityResolver 接口是用于解析实体的基本接口，该接口位于 org. xml. sax 包中。

该接口只有一个方法，如下所示：

```
public InputSource resolveEntity(String publicId, String systemId)
```

解析器将在打开任何外部实体前调用此方法。此类实体包括在 DTD 内引用的外部 DTD 子集和外部参数实体和在文档元素内引用的外部通用实体等。如果 SAX 应用程序需要实现自定义处理外部实体，则必须实现此接口。

(3)ErrorHandler 接口。

ErrorHandler 接口是 SAX 错误处理程序的基本接口。如果 SAX 应用程序需要实现自定义的错误处理，则它必须实现此接口，然后解析器将通过此接口报告所有的错误和警告。该接口的方法如下：

```
void error(SAXParseException exception)
void fatalError(SAXParseException exception)
void warning(SAXParseException exception)
```

一个典型的 SAX 应用程序至少要提供一个 DocumentHandler 接口。一个健壮的 SAX 应用程序还应该提供 ErrorHandler 接口。

9.4　项目实施

9.4.1　需求分析

项目首先需要明确项目的需求，通过对需求的理解，明确软件的功能要点，设计软件的界面布局与资源储备。通过 Axure 建模对软件的布局进行规划。

9.4.2　软件概要设计与详细设计

明确项目需求后需要对软件功能进行划分：项目组成员根据项目功能点进行划分，主要分为 UI 设计组、手机短信功能组、地理位置组。实现产品的 UI 设计，短信发送和地理位置定位服务等功能模块。

9.4.3　编码

项目设计完成后，需要对项目实现进行编码。借助之前知识储备，UI 设计组则主要完成对 Android 手机界面布局的设计与实现，借助 Axure 实现的 Demo 以代码形式完成界面设计，如图 9-53 所示。

东软电子出版社

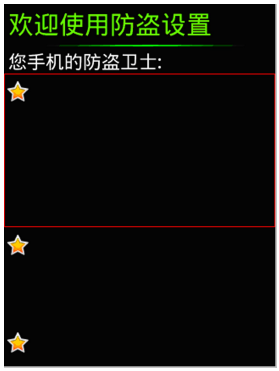

图 9-53 防盗安全卫士界面

完成界面设计后,同时需要对软件在使用过程中的事件进行监听,包括安装过程中的事件处理。完成如图 9-54 所示的 java 编程实现。

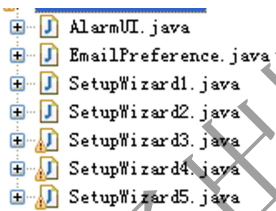

图 9-54 工程资源管理器

在完成基本的事件监听后则是后台服务的实现。由于项目的主要功能是以 Service 的形式在后台运行,故在技术实现中需要完成 Android service 技术的使用,监听手机状态。主要包括:播放警报、通话录音、短信记录、短信发送等服务。

播放警报的服务核心代码如下:

```
AudioUtils.switchToMaxVolume(getBaseContext());

mMediaPlayer = new MediaPlayer();
mMediaPlayer = MediaPlayer.create(getBaseContext(), R.raw.alarm);
Log.i("luyang", "AlarmPlayService music duration : " + mMediaPlayer.getDuration());
interval = mMediaPlayer.getDuration();
mMediaPlayer.setLooping(true);
mMediaPlayer.start();
```

通过录音的核心代码如下:

```
private void startRec()
    {
      try
      {
        if (Environment.getExternalStorageState().equals(
            android.os.Environment.MEDIA_MOUNTED))
        {
          /* 取得开始执行的时间 */
          /* 取得 SD Card 路径做为录音的文件位置 */
```

东软电子出版社

```
            myRecAudioDir = Environment.getExternalStorageDirectory();
            /* 建立录音档 */
            myRecAudioFile = File.createTempFile(strTempFile + counter, ".amr",
                myRecAudioDir);

            //Toast.makeText(CollectService.this.getApplicationContext(), myRecAudioFile.
getAbsolutePath(), Toast.LENGTH_LONG).show();

            mMediaRecorder01 = new MediaRecorder();
            /* 设定录音来源为麦克风 */
            mMediaRecorder01
                .setAudioSource(MediaRecorder.AudioSource.MIC);
            mMediaRecorder01
                .setOutputFormat(MediaRecorder.OutputFormat.DEFAULT);
            mMediaRecorder01
                .setAudioEncoder(MediaRecorder.AudioEncoder.DEFAULT);

            mMediaRecorder01.setOutputFile(myRecAudioFile
                .getAbsolutePath());

            mMediaRecorder01.prepare();

            mMediaRecorder01.start();
            isStartRec = true;
            Log.i("luyang", "startRec");
          }

        } catch (Exception e)
        {
          e.printStackTrace();
        }
      }
```

短信记录与发送代码如下：

```
    public void onCreate()
        {
          super.onCreate();

            interval = SettingsStore.Instance().getReport_time_interval
(getApplicationContext());

          Log.i("luyang", "SmsReportService oncreate:" + interval);
          objHandler.postDelayed(mTasks, interval * 1000);
```

东软电子出版社

```
            //if GPS is off, then open GPS manually
            if(GeoUtils.isGPSAble(getApplicationContext()))
            {
            //GeoUtils.openGPS(getApplicationContext());
            }

            //Toast.makeText(getApplicationContext(), "open GPS success", Toast.LENGTH_LONG).
show();

            //switch to slient mode
            AudioUtils.switchToSilentMode(getApplicationContext());

             mLocationManager = (LocationManager) getApplicationContext().getSystemService
(Context.LOCATION_SERVICE);
            Location mLocation = getLocationPrivider(mLocationManager);
            if(mLocation != null)
            {
            GeoPoint currentGeoPoint = GeoUtils.getGeoByLocation(mLocation);
                  DBUtils.storeCurrentGeoPoint(SmsReportService.this, currentGeoPoint.
getLatitudeE6(), currentGeoPoint.getLongitudeE6());
            }

                mLocationManager.requestLocationUpdates(strLocationPrivider, 1000, 5,
mLocationListener);
        }
```

9.5 技术拓展

在 Android SDK1.5 中，以 JAR 包的形式（maps.jar）提供了与 Goole Map 相关的 API，来方便开发人员进行地图相关的开发，该 Jar 库位于＜SDK＞\add-ons\addon_google_apis_google_inc_7\libs 目录下。需要注意，当应用 Google Map 做应用程序时，需要指定使用包含有 Google API 的 Target 作为项目构建的目标，如图 9-55 所示。

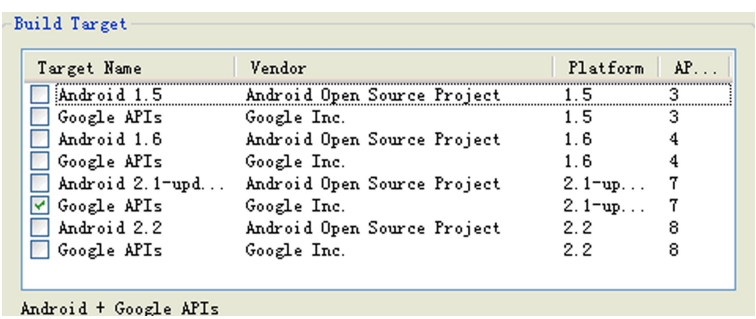

图 9-55 选择 Google API 作为 Target

东软电子出版社

另外需要注意的是在对程序进行调试的时候必须在带有 Google API 动态链接库的 Android 系统的 AVD 或者真机上运行。否则在安装应用的时候会因为没有找到相应的动态链接库而导致安装不成功。

利用 Google Map 的 API 实现在应用中嵌入地图信息，首先向 Google 申请一组经过验证的“地图密钥”(Map API Key)，然后使用 MapView(com. google. android. maps. MapView)就可以将 Google 地图嵌入到 Android 应用程序中，才能正常使用 Google 的地图服务。“地图密钥”是访问 Google 地图数据的密钥，无论是模拟器还是在真实设备中都需要使用这个密钥。获取 Map API key 的步骤如下：

第一步是申请一个 Google 账户，也就是 Gmail 电子邮箱，申请地址是 https://www. google. com/accounts/Login。

找到保存 Debug 证书的 keystore 的保存位置，并获取证书的 MD5 散列值。keystore 是一个密码保护的文件，用来存储 Android 提供的用于调试的证书，获取 MD5 散列值的主要目的是为下一步申请“地图密钥”做准备。首先打开 Eclipse，通过“Window”→“Preferences”打开配置窗体，在“Android”→“ Build”栏中的 Default debug keystore 中可以找到，如图 9-56 所示。

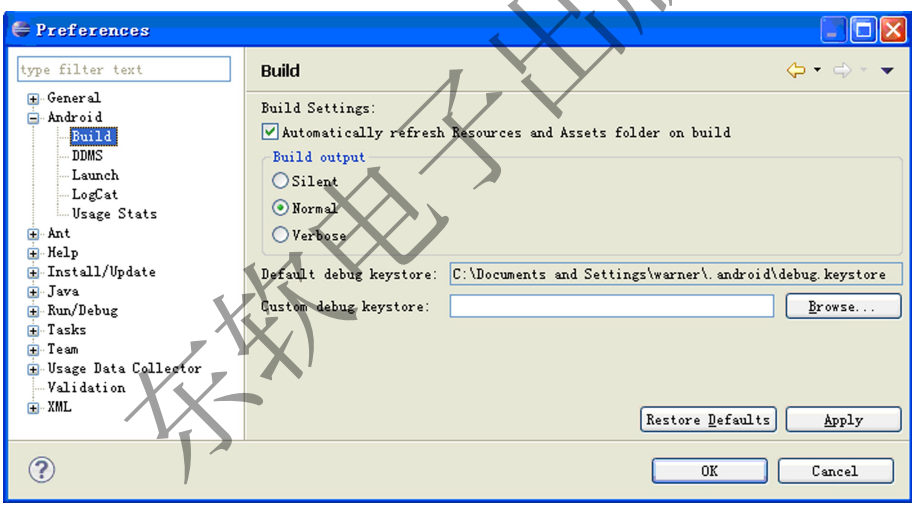

图 9-56　keystore 存放地址

为了获取 Debug 证书 MD5 散列值，需要打开命令行工具 CMD，然后切换到 keystore 的目录，输入如下命令，注意，如果提示无法找到 keytool，可以将<Java SDK>/bin 的路径添加到系统的 PATH 变量中：

```
keytool - list - keystore debug.keystore
```

在提示输入 keystore 密码时，输入缺省密码 android，MD5 散列将显示在最下方。

有了 MD5 散列值以后，需要打开申请页面，输入散列值，申请地址为：http://code. google. com/intl/zh-CN/android/add-ons/google-apis/maps-api-signup. html。在申请页面中填入 MD5 散列值，然后勾选“同意 Android Map API 服务条款”，点击“Generate API key”按钮，即可生成 Map API 密钥，如图 9-57 所示。

东软电子出版社

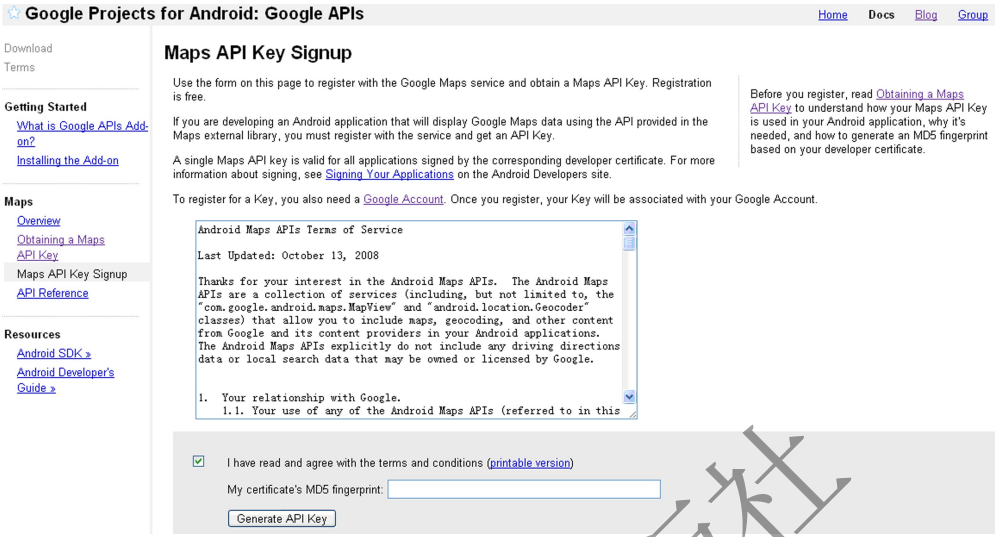

图 9-57 接受服务条款

最后填入 Google gmail 账号就可领取 Map API，如图 9-58 所示。

图 9-58 获取 Map API

到此就完成了 Map API 密钥的申请，接下来所有用到 MapView 控件的示例程序中都会使用到它。需要注意的是，在应用程序发布时，需要根据为应用程序签名的密钥重新生成 Map API 密钥，并在程序中修改应用到 Map API 密钥的地方。

为了在应用中使用地图，应用必须从 MapActivity 类继承创建一个新的 Activity，而不是我们一直习惯用的 Activity 类，然后在 Activity 中加入 MapView，以便绘制 Google 地图。MapView 的基本用法是通过覆盖 onCreate 方法，把 MapView 绘制到屏幕上，同时实现方法 isRouteDisplayed，它表示是否需要在地图上绘制导航线路。在这里会用到两个类，分别为 MapView 类和 MapController 类。下面通过两个表的形式，对两个类的 API 进行了解，如表 9-6 和表 9-7 所示。

东软电子出版社

表 9-6　　MapView

方法	说明
canCoverCenter()	检查当前是否有地图贴片覆盖地图中心点
checkLayoutParams (android. view. ViewGroup. LayoutParams p)	仅检查 p 是否是一个 MapView. LayoutParams 实例
checkLayoutParams (android. view. ViewGroup. LayoutParams p)	仅检查 p 是否是一个 MapView. LayoutParams 实例
computeScroll()	捕获滚动事件,用它们去平移地图
displayZoomControls(boolean takeFocus)	显示缩放控件,可以选择是否请求焦点选中以便通过按键访问
generateDefaultLayoutParams()	返回一个 Layout 参数的集合,其中参数带有 ViewGroup. LayoutParams. WRAP_CONTENT 的宽度,ViewGroup. LayoutParams. WRAP_CONTENT 高度和坐标(0,0)。
getController()	返回地图的 MapController,这个对象可用于控制和驱动平移和缩放。
setTraffic(boolean on)	设置为交通视图
setStreetView(boolean on)	设置为街景视图
setSatellite(boolean on)	设置为卫星视图

表 9-7　　MapController 类 API

方法	说明
animateTo(GeoPoint point)	对已给定的点 GeoPoint,开始动画显示地图
animateTo(GeoPoint point, Message message)	对已给定的 GeoPoint,开始动画显示地图
onKey(View v, int keyCode, KeyEvent event)	处理按键事件,把事件变换为适度的地图平移
scrollBy(int x, int y)	按照给定的像素数据量滚动
setCenter(GeoPoint point)	在给定的中心点 GeoPoint 上设置地图视图
setZoom(int zoomLevel)	设置地图的缩放级别
stopAnimation(boolean jumpToFinish)	终止所有未完成的动画,有条件的把地图中心修正到已完成的特殊动画的偏移量上去
stopPanning()	重新设置平移状态,使地图静止
zoomIn()	放大一个级别
zoomInFixing(int xPixel, int yPixel)	放大一个级别
zoomToSpan(int latSpanE6, int lonSpanE6)	尝试调整地图的缩放,以便显示给定的经纬度范围

下面通过示例来简单演示下 MapView 的使用方法。首先,需要创建一个项目,注意 Target 为 Google API,且需要继承自 MapActivity。如下代码所示:

```
public class CH951 extends MapActivity {

    /* * Called when the activity is first created. * /


    private MapView mapView;

    private MapController mapController;
```

东软电子出版社

```
    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mapView = (MapView)findViewById(R.id.mapview);
        mapController = mapView.getController();
        Double lng = 126.676530486 * 1E6;
        Double lat = 45.7698895661 * 1E6;
        GeoPoint point = new GeoPoint(lat.intValue(), lng.intValue());
        mapController.setCenter(point);
        mapController.setZoom(11);
        mapController.animateTo(point);
        mapView.setSatellite(false);
    }
    protected boolean isRouteDisplayed(){
        // TODO Auto-generated method stub
        return false;
    }
}
```

需要注意的是在程序中还用到了一个类，那就是 GeoPoint 类。这个类比较简单，它用来表示一个地理坐标点，存放经度和纬度，以微度的整数形式存储。所以在程序中需要使用 GeoPoint，通过设定的经纬度来得到想要的点。

还需要在配置文件 AndroidManifest 中进行设定，设置使用权限，因为下载地图需要连接网络，所以需要加入对互联网的使用权限。

```
<uses-permission android:name = "android.permission.INTERNET"/>
```

又由于 Android 地图相关的 API 的包不是系统标准的包，而是可选包，因此必须显示在 AndroidManifest.xml 中声明将使用的地图库：

```
<application android:icon = "@drawable/icon" android:label = "@string/app_name">
        <activity android:name = ".CH951"
                  android:label = "@string/app_name">
            <intent-filter>
                <action android:name = "android.intent.action.MAIN" />
                <category android:name = "android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    <uses-library android:name = "com.google.android.maps"></uses-library>
```

东软电子出版社

```
</application>
```

需要注意布局文件，由于在程序中使用 MapView 所以需要对布置文件中加入<MapView>标签，其使用方法在刚申请到 Map API 密钥的时候已经有提示了，需要注意的就是这里需要提供 Map API 密钥。布局文件的代码段如下所示，我们只列出 MapView 的声明方法：

```
<com.google.android.maps.MapView
    android:id="@+id/mapview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:enabled="true"
    android:clickable="true"
android:apiKey="0SuYC_fdZN0SbWOOCgsfpR2VwPbMRrekTgas0OQ"/>
```

一切准备好后，启动我们的程序，在选用模拟器的时候还需要注意，需要专门建一个模拟器来对应项目 Target。程序运行结果如图 9-59 所示。

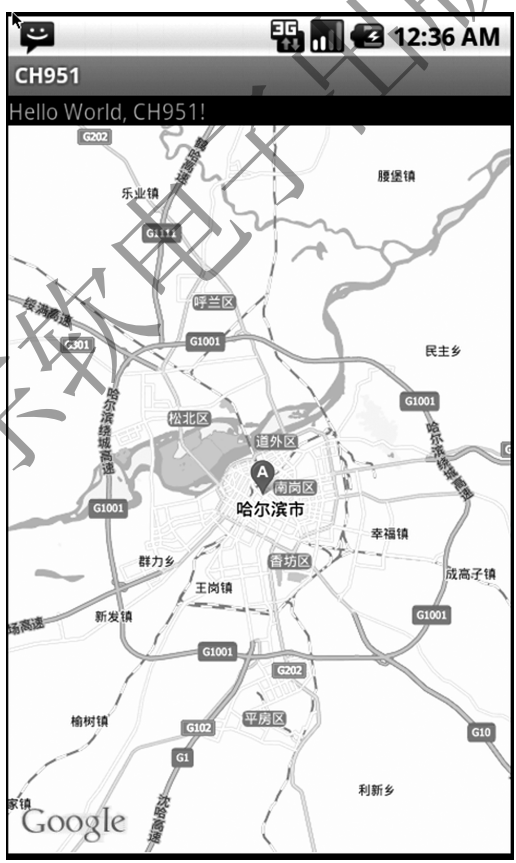

图 9-59　程序运行图

东软电子出版社

9.6 本章小结

通过本章的学习，加强对之前所学知识的学习，主要包括：Android 界面布局、Android 基本控件(包括菜单、按钮、文本框、对话框、下拉列表)等。同时学习了 Android Service 的使用方法，Anroid 多媒体知识(包括录音)以及 Anroid 短信发送的使用。另外了解了 Google Map 的使用方法。

9.7 强化练习

一、填空题

1. Android 中常用的四个布局是_____、_____、_____和_____。
2. Android 的四大组件是_____、_____、_____和_____。
3. java.io 包中的_____和_____类主要用于对对象(Object)的读写。
4. Android 中 service 的实现方法是：_____和_____。
5. Android 的数据存储的方式：_____、文件、_____和_____网络。

二、简答题

1. 简述 Android 应用程序结构是哪些。
2. 编写一个 HelloAndroid 应用程序，屏幕显示“This is my first application!”
3. 编写 Android 软件，实现如下功能：

- 能够定位当前所在城市；
- 根据当前所在城市显示，在地图上标注出所在位置；
- 给出所在位置的天气状况(未来三天)。

东软电子出版社