

第 1 章

Android应用初体验



本章将介绍编写Android应用需掌握的一些新的概念和UI组件。学完本章，如果没能理解全部内容，也不必担心。后续章节还会有更加详细的讲解，我们将再次温习并理解这些概念。

马上要编写的首个应用名为GeoQuiz，它能测试用户的地理知识。用户通过单击True或False按钮来回答屏幕上的问题，GeoQuiz可即时反馈答案正确与否。

图1-1显示了用户点击False按钮的结果。

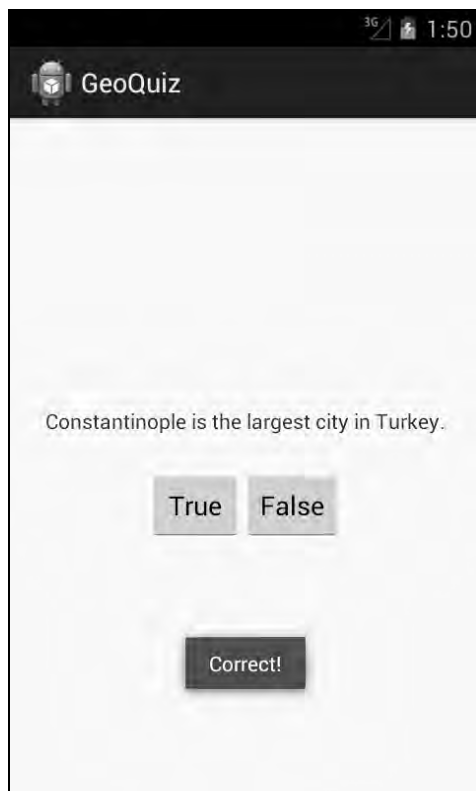


图1-1 正确答案应该是伊斯坦布尔（Istanbul），而不是君士坦丁堡

1.1 应用基础

GeoQuiz应用由一个activity和一个布局（layout）组成。

- activity是Android SDK中Activity类的一个具体实例，负责管理用户与信息屏的交互。应用的功能是通过编写一个个Activity子类来实现的。简单的应用可能只需一个子类，而复杂的应用则会有多个子类。

GeoQuiz是个简单应用，因此它只有一个名为QuizActivity的Activity子类。QuizActivity管理着图1-1所示的用户界面。

- 布局定义了一系列用户界面对象以及它们显示在屏幕上的位置。组成布局的定义保存在XML文件中。每个定义用来创建屏幕上的一个对象，如按钮或文本信息。

GeoQuiz应用包含一个名为activity_quiz.xml的布局文件。该布局文件中的XML标签定义了图1-1所示的用户界面。

QuizActivity与activity_quiz.xml文件的关系如图1-2所示。



图1-2 QuizActivity管理着activity_quiz.xml文件定义的用户界面

学习了这些基本概念后，我们来创建本书第一个应用。

1.2 创建 Android 项目

首先我们来创建一个Android项目。Android项目包含组成一个应用的全部文件。启动Eclipse程序，选择File→New→Android Application Project菜单项，打开新建应用窗口来创建一个新的项目。

在应用名称（Application Name）处输入GeoQuiz，如图1-3所示。此时项目名称（Project Name）会自动更新为GeoQuiz。在包名处（Package Name）输入com.bignerdranch.android.geoquiz。

注意，以上输入的包名遵循了“DNS反转”约定，亦即将企业组织或公司的域名反转后，在尾部加上应用名称。遵循此约定可以保证包名的唯一性，这样，同一设备和Google Play商店的各类应用就可以区分开来。

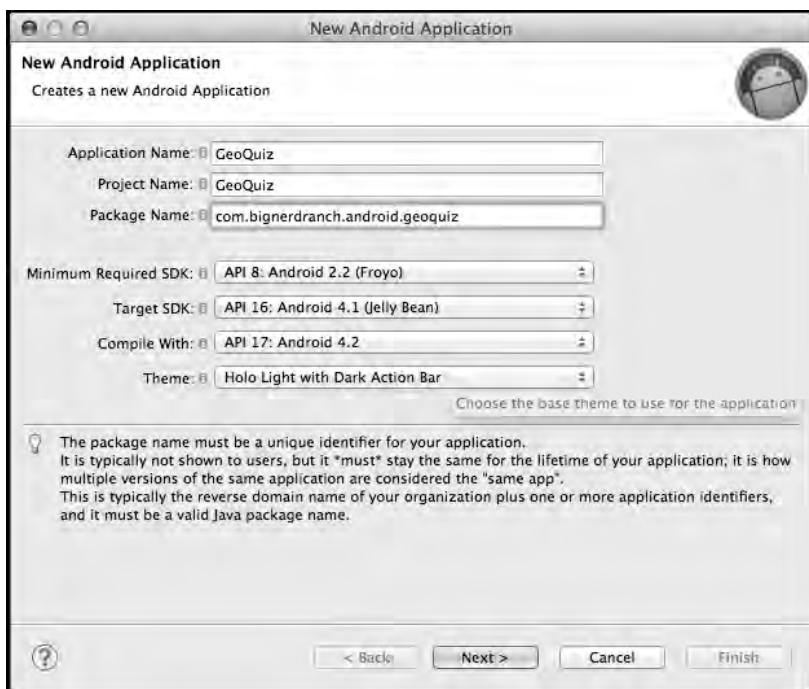


图1-3 创建新应用

接下来的四个选项用来配置应用如何与不同版本的Android设备适配。GeoQuiz应用只需使用默认设置，所以现在可以忽略它们。第6章将介绍Android不同版本的差异。

Android开发工具每年会更新多次，因此当前的向导画面看起来可能会与本书略有不同。这一般不是问题，工具更新后，向导画面的配置选项应该不会有太大差别。

（如果向导画面看起来大有不同，可以肯定开发工具已进行了重大更新。不要担心，请访问本书论坛<http://forums.bignerdranch.com>，学习如何使用最新版本的开发工具。）

现在单击Next按钮。

在第二个窗口中，清除已勾选的创建定制启动图标（Create custom launcher icon）选项，如图1-4所示。GeoQuiz应用只需使用默认的启动图标。最后确认创建活动（Create activity）选项已选中。

单击Next按钮继续。

图1-5所示的窗口询问想要创建的activity类型。选择Blank Activity。



图1-4 配置项目



图1-5 创建新的activity

单击Next按钮。

在应用向导的最后一个窗口，命名activity子类为QuizActivity，如图1-6所示。注意子类名的Activity后缀。尽管不是必需的，但我们建议遵循这一好的命名约定。

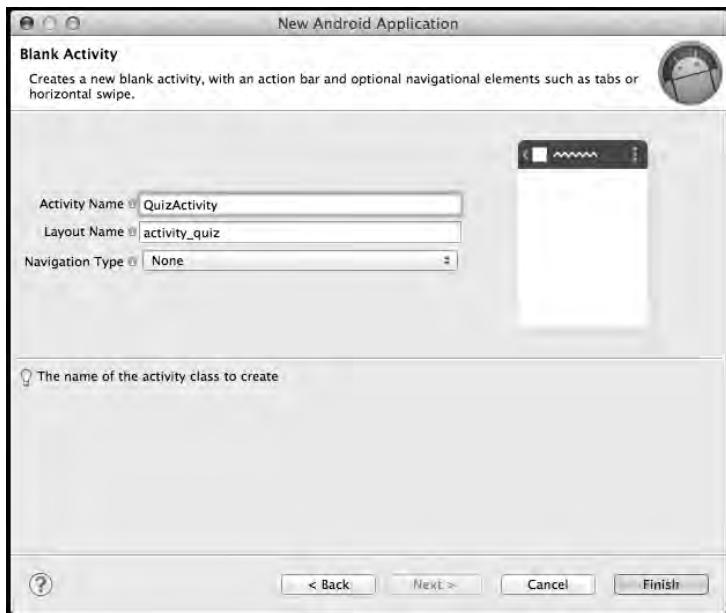


图1-6 配置新建的activity

为体现布局与activity间的对应关系，布局名称（Layout Name）会自动更新为activity_quiz。布局的命名规则是：将activity名称的单词顺序颠倒过来并全部转换为小写字母，然后在单词间添加下划线。对于后续章节中的所有布局以及将要学习的其他资源，建议统一采用这种命名风格。

保持导航类型（Navigation Type）的None选项不变，单击Finish按钮。Eclipse完成创建并打开新的项目。

1.3 Eclipse 工作区导航

如图1-7所示，Eclipse已在工作区窗口（workbench window）里打开新建项目。（注意，如是安装后初次使用Eclipse，则需关闭初始的欢迎窗口，才能看到如图所示的工作区窗口。）

整个工作区窗口分为不同的区域，这里统称为视图。

最左边是包浏览器（package explorer）视图，通过它可以管理所有项目相关的文件。

中间部分是代码编辑区（editor）视图。为便于开发，Eclipse默认在代码编辑区打开了activity_quiz.xml文件。

在工作区的右边以及底部还有一些其他视图。通过点击视图名称旁边的x关闭标志，可关闭右边的各种视图，如图1-7所示。底部的视图以分组面板（tab group）形式显示。可通过右上角的

控制功能最小化整个分组面板，而不是全部关闭它们。

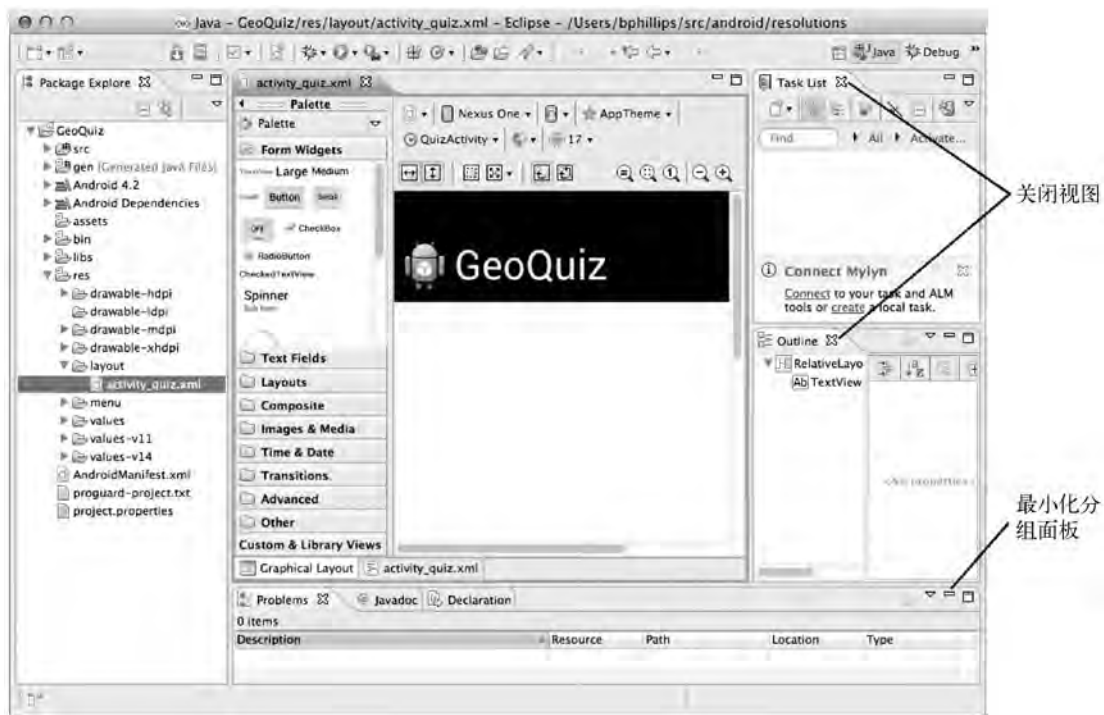


图1-7 调整安排工作区窗口

视图被最小化后，聚缩到了Eclipse工作区边缘区域的工具栏上。移动鼠标到工具栏的任意图标上，即可看到对应视图的名字，点击图标可恢复对应视图。

1.4 用户界面设计

如前所述，Eclipse已默认打开activity_quiz.xml布局文件，并在Android图形布局工具里显示了预览界面。虽然图形化布局工具非常好用，但为更好地理解布局的内部原理，我们还是先学习如何使用XML代码来定义布局。

在代码编辑区的底部选择标为activity_quiz.xml的标签页，从预览界面切换到XML代码界面。

当前，activity_quiz.xml文件定义了默认的activity布局。应用的默认布局经常改变，但其XML布局文件却总是与代码清单1-1文件相似。

代码清单1-1 默认的activity布局（activity_quiz.xml）

```
<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"
```

```
tools:context=".QuizActivity" >

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    android:text="@string/hello_world" />

</RelativeLayout>
```

首先，我们注意到activity_quiz.xml文件不再包含指定版本声明与文件编码的如下代码：

```
<?xml version="1.0" encoding="utf-8"?>
```

ADT21开发版本以后，Android布局文件已不再需要该行代码。不过，在很多情况下，可能还是会看到它。

应用activity的布局默认定义了两个组件（widget）：**RelativeLayout**和**TextView**。

组件是组成用户界面的构造模块。组件可以显示文字或图像、与用户交互，甚至是布置屏幕上的其他组件。按钮、文本输入控件和选择框等都是组件。

Android SDK内置了多种组件，通过配置各种组件可获得所需的用户界面及行为。每一个组件是**View**类或其子类（如**TextView**或**Button**）的一个具体实例。

图1-8展示了代码清单1-1中定义的**RelativeLayout**和**TextView**是如何在屏幕上显示的。

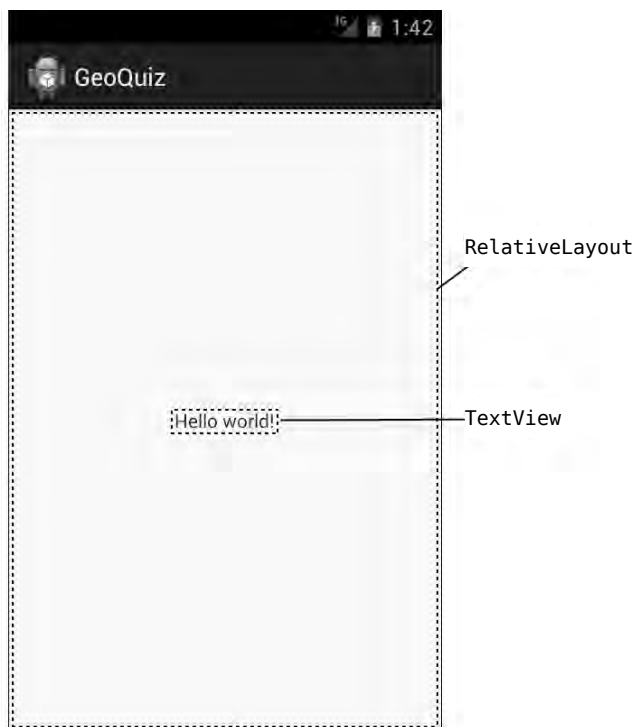


图1-8 显示在屏幕上的默认组件

不过,图1-8所示的默认组件并不是我们需要的,QuizActivity的用户界面需要下列五个组件:

- 一个垂直LinearLayout组件;
- 一个TextView组件;
- 一个水平LinearLayout组件;
- 两个Button组件。

图1-9展示了以上组件是如何构成QuizActivity活动用户界面的。

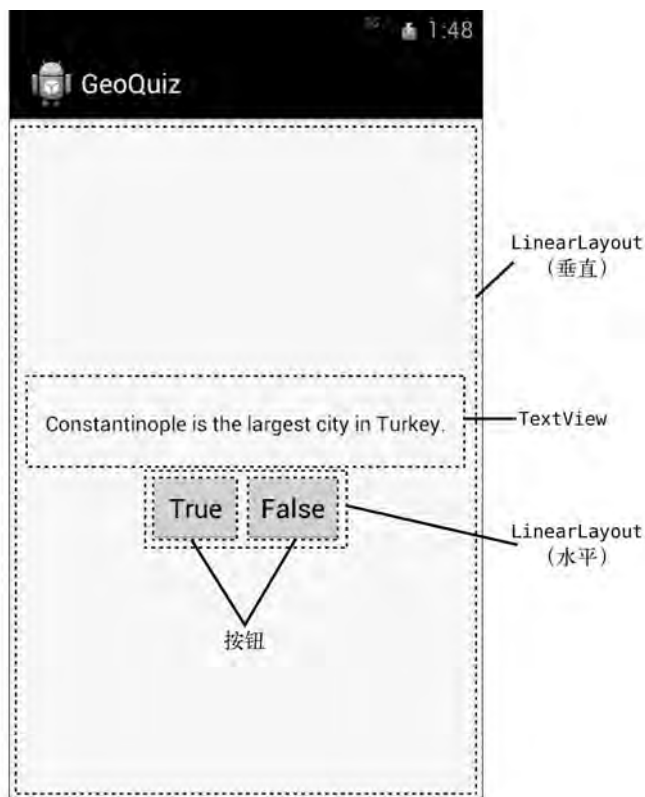


图1-9 布置并显示在屏幕上的组件

下面我们在activity_quiz.xml文件中定义这些组件。

如代码清单1-2所示,修改activity_quiz.xml文件。注意,需删除的XML已打上删除线,需添加的XML以粗体显示。本书统一使用这样的版式约定。

代码清单1-2 在XML文件(activity_quiz.xml)中定义组件

```
<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"  
tools:context=".QuizActivity">
```



```

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    android:text="@string/hello_world" />

</RelativeLayout>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="24dp"
        android:text="@string/question_text" />

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/true_button" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/false_button" />

    </LinearLayout>

</LinearLayout>

```

参照代码清单直接输入代码, 就算不理解这些代码也没关系, 你会在后续的学习中弄明白的。需要特别注意的是, 开发工具无法校验布局XML内容, 请避免输入或拼写错误。

根据所使用的工具版本不同, 可能会得到三行以`android:text`开头的代码有误。先暂时忽略它们, 以后再去解决这一问题。

将XML文件与图1-9所示的用户界面进行对照, 可以看出组件与XML元素一一对应。元素的名称就是组件的类型。

各元素均有一组XML属性。属性可以看作是如何配置组件的指令。

以层次等级视角来研究布局, 有助于我们更方便地理解元素与属性的运作方式。

1.4.1 视图层级结构

组件包含在视图对象的层级结构, 即视图层级结构 (view hierarchy) 中。图1-10展示了代码清单1-2所示XML布局对应的视图层级结构。

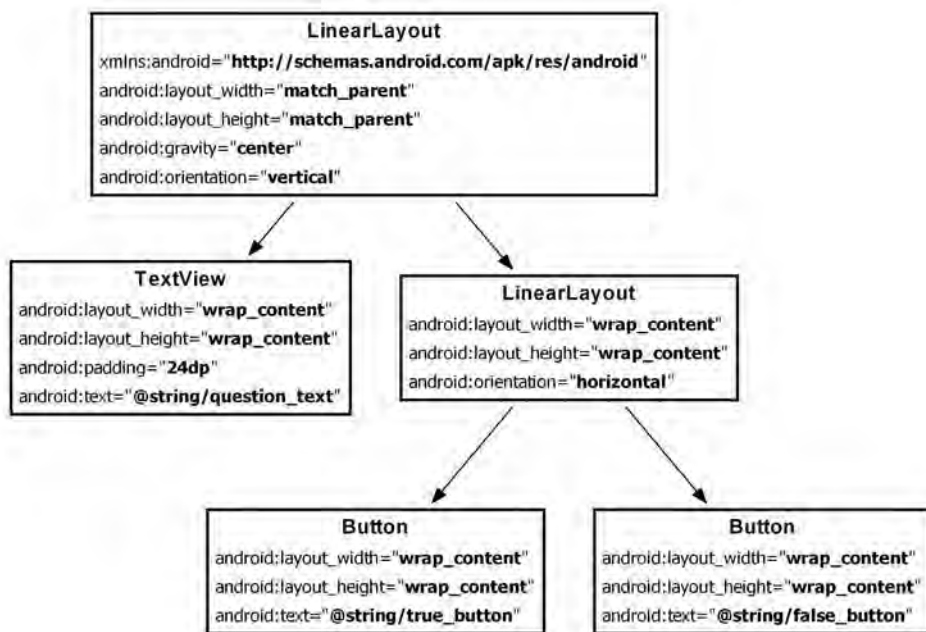


图1-10 布局中组件及属性的层级结构

从布局的视图层级结构可以看到,其根元素是一个LinearLayout组件。作为根元素,LinearLayout组件必须指定Android XML资源文件的命名空间属性为`http://schemas.android.com/apk/res/android`。

LinearLayout组件继承自View子类的ViewGroup组件。ViewGroup组件是一个包含并配置其他组件的特殊组件。如需以一系列或一排的样式布置组件,使用LinearLayout组件就可以了。其他ViewGroup子类还包括FrameLayout、TableLayout和RelativeLayout。

若某个组件包含在一个ViewGroup中,该组件与ViewGroup即构成父子关系。根LinearLayout有两个子组件:TextView和LinearLayout。作为子组件的LinearLayout本身还有两个Button子组件。

1.4.2 组件属性

下面我们一起来看看配置组件的一些常用属性。

1. android:layout_width和android:layout_height属性

几乎每类组件都需要android:layout_width和android:layout_height属性。它们通常被设置为以下两种属性值之一。

❑ `match_parent`: 视图与其父视图大小相同。

❑ `wrap_content`: 视图将根据其内容自动调整大小。

(以前还有一个`fill_parent`属性值,等同于`match_parent`,目前已废弃不用。)

根LinearLayout组件的高度与宽度属性值均为match_parent。LinearLayout虽然是根元素，但它也有父视图（View）——Android提供该父视图来容纳应用的整个视图层级结构。

其他包含在界面布局中的组件，其高度与宽度属性值均被设置为wrap_content。请参照图1-9理解该属性值定义尺寸大小的作用。

TextView组件比其包含的文字内容区域稍大一些，这主要是android:padding="24dp"属性的作用。该属性告诉组件在决定大小时，除内容本身外，还需增加额外指定量的空间。这样屏幕上显示的问题与按钮之间便会留有一定的空间，使整体显得更为美观。（不理解dp的意思？dp即density-independent pixel，指与设备无关的像素，第8章将介绍有关它的概念。）

2. android:orientation属性

android:orientation属性是两个LinearLayout组件都具有的属性，决定了二者的子组件是水平放置的还是垂直放置的。根LinearLayout是垂直的，子LinearLayout是水平的。

LinearLayout子组件的定义顺序决定着其在屏幕上显示的顺序。在竖直的LinearLayout中，第一个定义的子组件出现在屏幕的最上端。而在水平的LinearLayout中，第一个定义的子组件出现在屏幕的最左端。（如果设备语言为从右至左显示，如Arabic或者Hebrew，第一个定义的子组件则出现在屏幕的最右端。）

3. android:text属性

TextView与Button组件具有android:text属性。该属性指定组件显示的文字内容。

请注意，android:text属性值不是字符串字面值，而是对字符串资源（string resources）的引用。

字符串资源包含在一个独立的名为strings的XML文件中，虽然可以硬编码设置组件的文本属性，如android:text="True"，但这通常不是个好方法。将文字内容放置在独立的字符串资源XML文件中，然后引用它们才是好方法。在第15章中，我们将学习如何使用字符串资源轻松实现本地化。

需要在activity_quiz.xml文件中引用的字符串资源目前还不存在。现在我们来添加这些资源。

1.4.3 创建字符串资源

每个项目都包含一个名为strings.xml的默认字符串文件。

在包浏览器中，找到res/values目录，点击小三角显示目录内容，然后打开strings.xml文件。忽略图形界面，在编辑器底部选择strings.xml标签页，切换到代码界面。

可以看到，项目模版已经默认添加了一些字符串资源。删除不需要的hello_world部分，添加应用布局需要的三个新的字符串，如代码清单1-3所示。

代码清单1-3 添加字符串资源（strings.xml）

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">GeoQuiz</string>
    <string name="hello_world">Hello, world!</string>
    <string name="question_text">Constantinople is the largest city in Turkey.</string>
    <string name="true_button">True</string>
    <string name="false_button">False</string>
```

```
<string name="menu_settings">Settings</string>
```

```
</resources>
```

（项目已默认配置好应用菜单，请勿删除menu_settings字符串设置，否则将导致与应用菜单相关的其他文件发生版式错误。）

现在，在GeoQuiz项目的任何XML文件中，只要引用到@string/false_button，应用运行时，就会得到文本“False”。

保存strings.xml文件。这时，activity_quiz.xml布局曾经提示缺少字符串资源的信息应该不会再出现了。（如仍有错误信息，那么检查一下这两个文件，确认是否存在输入或拼写错误。）

字符串文件默认被命名为strings.xml，当然也可以按个人喜好任意取名。一个项目也可以有多个字符串文件。只要这些文件都放置在res/values/目录下，并且含有一个resources根元素，以及多个string子元素，字符串定义即可被应用找到并得到正确使用。

1.4.4 预览界面布局

至此，应用的界面布局已经完成，现在我们使用图形布局工具来进行实时预览。首先，确认保存了所有相关文件并且无错误发生，然后回到activity_quiz.xml文件，在编辑区底部选择图形布局标签页进行界面布局预览，如图1-11所示。



图1-11 在图形布局工具中预览界面布局（activity_quiz.xml）

1.5 从布局 XML 到视图对象

想知道 activity_quiz.xml 中的 XML 元素是如何转换为视图对象的吗？答案就在于 Quiz-Activity 类。

在创建 GeoQuiz 项目的同时，也创建了一个名为 QuizActivity 的 Activity 子类。QuizActivity 类文件存放在项目的 src 目录下。目录 src 是项目全部 Java 源代码的存放处。

在包浏览器中，依次展开 src 目录与 com.bignerdranch.android.geoquiz 包，显示其中的内容。然后打开 QuizActivity.java 文件，逐行查看其中的代码，如代码清单 1-4 所示。

代码清单 1-4 QuizActivity 活动的默认类文件（QuizActivity.java）

```
package com.bignerdranch.android.geoquiz;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;

public class QuizActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_quiz);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_quiz, menu);
        return true;
    }

}
```

（如果无法看到全部类包导入语句，请单击第一行导入语句左边的 ⊕ 符号，从而显示全部导入语句。）

该 Java 类文件包含两个 Activity 方法：onCreate(Bundle) 和 onCreateOptionsMenu(Menu)。

暂不用理会 onCreateOptionsMenu(Menu) 方法，第 16 章会详细介绍它。

activity 子类的实例创建后，onCreate(Bundle) 方法将会被调用。activity 创建后，它需要获取并管理属于自己的用户界面。获取 activity 的用户界面，可调用以下 Activity 方法：

```
public void setContentView(int layoutResID)
```

通过传入布局的资源 ID 参数，该方法生成指定布局的视图并将其放置在屏幕上。布局视图生成后，布局文件包含的组件也随之以各自的属性定义完成实例化。

资源与资源 ID

布局是一种资源。资源是应用非代码形式的内容，比如图像文件、音频文件以及 XML 文件等。项目的所有资源文件都存放在目录 res 的子目录下。通过包浏览器可以看到，布局 activity_

quiz.xml资源文件存放在res/layout/目录下。包含字符串资源的strings文件存放在res/values/目录下。

可使用资源ID在代码中获取相应的资源。activity_quiz.xml文件定义的布局资源ID为R.layout.activity_quiz。

在包浏览器展开目录gen，找到并打开R.java文件，即可看到GeoQuiz应用当前所有的资源ID。R.java文件在Android项目编译过程中自动生成，遵照该文件头部的警示，请不要尝试修改该文件的内容，如代码清单1-5所示。

代码清单1-5 GeoQuiz应用当前的资源ID（R.java）

```
/* AUTO-GENERATED FILE. DO NOT MODIFY.
...
*/

package com.bignerdranch.android.geoquiz;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int ic_launcher=0x7f020000;
    }
    public static final class id {
        public static final int menu_settings=0x7f070003;
    }
    public static final class layout {
        public static final int activity_quiz=0x7f030000;
    }
    public static final class menu {
        public static final int activity_quiz=0x7f060000;
    }
    public static final class string {
        public static final int app_name=0x7f040000;
        public static final int false_button=0x7f040003;
        public static final int menu_settings=0x7f040006;
        public static final int question_text=0x7f040001;
        public static final int true_button=0x7f040002;
    }
    ...
}
```

可以看到R.layout.activity_quiz即来自该文件。activity_quiz是R的内部类layout里的一个整型常量名。

它们定义的字符串同样具有资源ID。目前为止，我们还未在代码中引用过字符串，但如果需要，则应该使用以下方法：

```
setTitle(R.string.app_name);
```

Android为整个布局文件以及各个字符串生成资源ID，但activity_quiz.xml布局文件中的组件除外，因为不是所有的组件都需要资源ID。在本章中，我们只用到两个按钮，因此只需为这两个按钮生成相应的资源ID即可。

要为组件生成资源ID，请在定义组件时为其添加上android:id属性。在activity_quiz.xml文

件中，分别为两个按钮添加上`android:id`属性，如代码清单1-6所示。

代码清单1-6 为按钮添加资源ID（`activity_quiz.xml`）

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

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="24dp"
    android:text="@string/question_text" />

<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <Button
        android:id="@+id/true_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/true_button" />

    <Button
        android:id="@+id/false_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/false_button" />

</LinearLayout>

</LinearLayout>
```

请注意`android:id`属性值前面有一个`+`标志，而`android:text`属性值则没有，这是因为我们将要创建资源ID，而对字符串资源只是做了引用。

保存`activity_quiz.xml`文件，重新查看`R.java`文件，确认`R.id`内部类中生成了两个新的资源ID，如代码清单1-7所示。

代码清单1-7 新的资源ID（`R.java`）

```
public final class R {
...
    public static final class id {
        public static final int false_button=0x7f070001;
        public static final int menu_settings=0x7f070002;
        public static final int true_button=0x7f070000;
    }
...
}
```

1.6 组件的实际应用

既然按钮有了资源ID，就可以在`QuizActivity`中直接获取它们。首先，在`QuizActivity.java`文件中增加两个成员变量。

在QuizActivity.java文件中输入代码清单1-8所示代码。（请勿使用代码自动补全功能。）

代码清单1-8 添加成员变量（QuizActivity.java）

```
public class QuizActivity extends Activity {

    private Button mTrueButton;
    private Button mFalseButton;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_quiz);
    }

    ...
}
```

文件保存后，可看到两个错误提示。没关系，这点错误马上就可以搞定。请注意新增的两个成员（实例）变量名称的m前缀。该前缀是Android编程所遵循的命名约定，本书将始终遵循该约定。

现在，请将鼠标移至代码左边的错误提示处，可看到两条同样的错误：Button cannot be resolved to a type。

该错误提示告诉我们需要在QuizActivity.java文件中导入android.widget.Button类包。可在文件头部手动输入以下代码：

```
import android.widget.Button;
```

或者采用下面介绍的便捷方式自动导入。

1.6.1 类包组织导入

使用类包组织导入，就是让Eclipse依据代码来决定应该导入哪些Java或Android SDK类包。如果之前导入的类包不再需要了，Eclipse也会自动删除它们。

通过以下组合键命令，进行类包组织导入：

❑ Command+Shift+O（Mac系统）；

❑ Ctrl+Shift+O（Windows和Linux系统）。

类包导入完成后，刚才的错误提示应该就会消失了。（如果错误提示仍然存在，请检查Java代码以及XML文件，确认是否存在输入或拼写错误。）

接下来，我们来编码使用按钮组件，这需要以下两个步骤：

❑ 引用生成的视图对象；

❑ 为对象设置监听器，以响应用户操作。

1.6.2 引用组件

在activity中，可通过以下Activity方法引用已生成的组件：


```
public View findViewById(int id)
```

该方法接受组件的资源ID作为参数，返回一个视图对象。

在QuizActivity.java文件中，使用按钮的资源ID获取生成的对象后，赋值给对应的成员变量，如代码清单1-9所示。注意，赋值前，必须先将返回的View转型（cast）为Button。

代码清单1-9 引用组件（QuizActivity.java）

```
public class QuizActivity extends Activity {

    private Button mTrueButton;
    private Button mFalseButton;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_quiz);

        mTrueButton = (Button)findViewById(R.id.true_button);
        mFalseButton = (Button)findViewById(R.id.false_button);
    }

    ...
}
```

1.6.3 设置监听器

Android应用属于典型的事件驱动类型。不同于命令行或脚本程序，事件驱动型应用启动后，即开始等待行为事件的发生，如用户单击某个按钮。（事件也可以由操作系统或其他应用触发，但用户触发的事件更显而易见。）

应用等待某个特定事件的发生，也可以说该应用正在“监听”特定事件。为响应某个事件而创建的对象叫做监听器（listener）。监听器是实现特定监听器接口的对象，用来监听某类事件的发生。

无需自己编写，Android SDK已经为各种事件内置开发了很多监听器接口。当前应用需要监听用户的按钮“单击”事件，因此监听器需实现View.OnClickListener接口。

首先处理True按钮，在QuizActivity.java文件中，在变量赋值语句后输入下列代码到onCreate(...)方法内，如代码清单1-10所示。

代码清单1-10 为True按钮设置监听器（QuizActivity.java）

```
...

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_quiz);

    mTrueButton = (Button)findViewById(R.id.true_button);
    mTrueButton.setOnClickListener(new View.OnClickListener() {
        @Override
```

```

        public void onClick(View v) {
            // Does nothing yet, but soon!
        }
    });

    mFalseButton = (Button)findViewById(R.id.false_button);
}

```

（如果遇到View cannot be resolved to a type的错误提示，请使用Mac的Command+Shift+O或Windows的Ctrl+Shift+O快捷键导入View类。）

在代码清单1-10中，我们设置了一个监听器。当按钮mTrueButton被点击后，监听器会立即通知我们。setOnClickListener(OnClickListener)方法以监听器作为参数被调用。在特殊情况下，该方法以一个实现了OnClickListener接口的对象作为参数被调用。

1. 使用匿名内部类

setOnClickListener(OnClickListener)方法传入的监听器参数是一个匿名内部类(anonymous inner class)实现，语法看上去稍显复杂，不过，只需记住最外层括号内的全部实现代码是作为整体参数传入setOnClickListener(OnClickListener)方法内的即可。该传入的参数就是新建的一个匿名内部类的实现代码。

```

mTrueButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Does nothing yet, but soon!
    }
});

```

本书所有的监听器都作为匿名内部类来实现。这样做的好处有二。其一，在大量代码块中，监听器方法的实现一目了然；其二，匿名内部类的使用只出现在一个地方，因此可以减少一些命名类的使用。

匿名内部类实现了OnClickListener接口，因此它也必须实现该接口唯一的onClick(View)方法。onClick(View)方法的代码暂时是一个空结构。实现监听器接口需要实现onClick(View)方法，但具体如何实现由使用者决定，因此即使是空的实现方法，编译器也可以编译通过。

（如果匿名内部类、监听器、接口等概念你已忘得差不多了，现在就去复习复习Java语言的基础知识，或者手边放一本参考书备查。）

参照代码清单1-11为False按钮设置类似的事件监听器。

代码清单1-11 为False按钮设置监听器（QuizActivity.java）

```

...

mTrueButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Does nothing yet, but soon!
    }
});

mFalseButton = (Button)findViewById(R.id.false_button);

```

```

mFalseButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Does nothing yet, but soon!
    }
});
}

```

2. 创建提示消息

现在让我们把按钮全副武装起来，使其具有可操作性吧。接下来要实现的就是，分别单击两个按钮，弹出我们称为toast的提示消息。Android的toast指用来通知用户的简短弹出消息，但无需用户输入或做出任何操作。这里，我们要做的就是使用toast来告知用户其答案正确与否，如图1-12所示

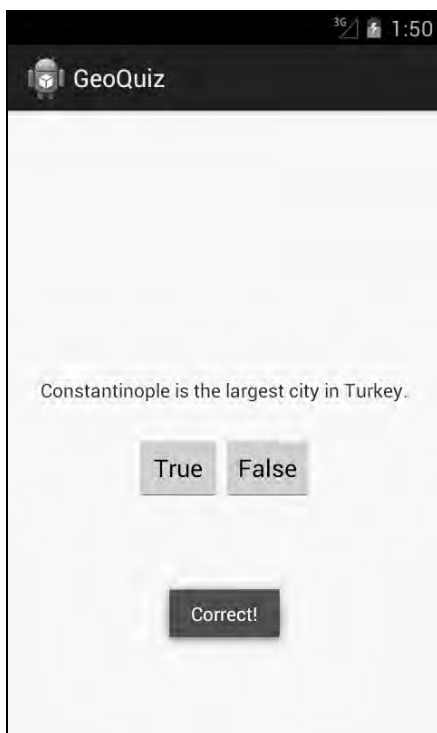


图1-12 toast反馈消息提示

首先回到strings.xml文件，如代码清单1-12所示，为toast添加消息显示用的字符串资源。

代码清单1-12 增加toast字符串（strings.xml）

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">GeoQuiz</string>
    <string name="question_text">Constantinople is the largest city in Turkey.</string>
    <string name="true_button">True</string>
    <string name="false_button">False</string>
    <string name="correct_toast">Correct!</string>

```

```

<string name="incorrect_toast">Incorrect!</string>
<string name="menu_settings">Settings</string>
</resources>

```

通过调用来自Toast类的以下方法，可创建一个toast：

```
public static Toast makeText(Context context, int resId, int duration)
```

该方法的Context参数通常是Activity的一个实例（Activity本身就是Context的子类）。第二个参数是toast待显示字符串消息的资源ID。Toast类必须利用context才能找到并使用字符串的资源ID。第三个参数通常是两个Toast常量中的一个，用来指定toast消息显示的持续时间。

创建Toast后，可通过调用Toast.show()方法使toast消息显示在屏幕上。

在QuizActivity代码里，分别对两个按钮的监听器调用makeText(...)方法，如代码清单1-13所示。在添加makeText(...)时，可利用Eclipse的代码自动补全功能，让代码输入工作更加轻松。

3. 使用代码自动补全

代码自动补全功能可以节约大量开发时间，越早掌握受益越多。

参照代码清单1-13，依次输入代码。当输入到Toast类后的点号时，Eclipse会弹出一个窗口，窗口内显示了建议使用的Toast类的常量与方法。

为便于选择所需的建议方法，可按Tab键移焦至自动补全弹出窗口上。（如果想忽略Eclipse的代码自动补全功能，请不要按Tab键或使用鼠标点击弹出窗口，只管继续输入代码直至完成。）

在列表建议清单里，选择makeText(Context, int, int)方法，代码自动补全功能会自动添加完成方法调用，包括参数的占位符值。

第一个占位符号默认加亮，直接输入实际参数值QuizActivity.this。然后按Tab键转至下一个占位符，输入实际参数值，依次类推，直至参照代码清单1-13完成全部参数的输入。

代码清单1-13 创建提示消息（QuizActivity.java）

```

...
mTrueButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Toast.makeText(QuizActivity.this,
                       R.string.incorrect_toast,
                       Toast.LENGTH_SHORT).show();
    }
});

mFalseButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Toast.makeText(QuizActivity.this,
                       R.string.correct_toast,
                       Toast.LENGTH_SHORT).show();
    }
});

```

在makeText(...)里，传入QuizActivity实例作为Context的参数值。注意此处应输入的参数是QuizActivit.this，不要想当然地直接输入this作为参数。因为匿名类的使用，这里的this指的是监听器View.OnClickListener。

使用代码自动补全功能，Eclipse会自动导入所需的类。因此，无需使用类包组织导入来导入Toast类了。

1.7 使用模拟器运行应用

要运行Android应用，需使用硬件设备或者虚拟设备（virtual device）。包含在开发工具中的Android设备模拟器可提供多种虚拟设备。

要想创建Android虚拟设备（AVD），在Eclipse中，选择Window → Android Virtual Device Manager菜单项，当AVD管理器窗口弹出时，点击窗口右边的New...按钮。

在随后弹出的对话框中，可以看到有很多配置虚拟设备的选项。对于首个虚拟设备，我们选择模拟运行Google APIs - API Level 17的Galaxy Nexus设备，如图1-13所示。注意，如果使用的是Windows系统，需要将内存选项值从1024改为512，这样虚拟设备才能正常运行。配置完成后，点击OK确认。

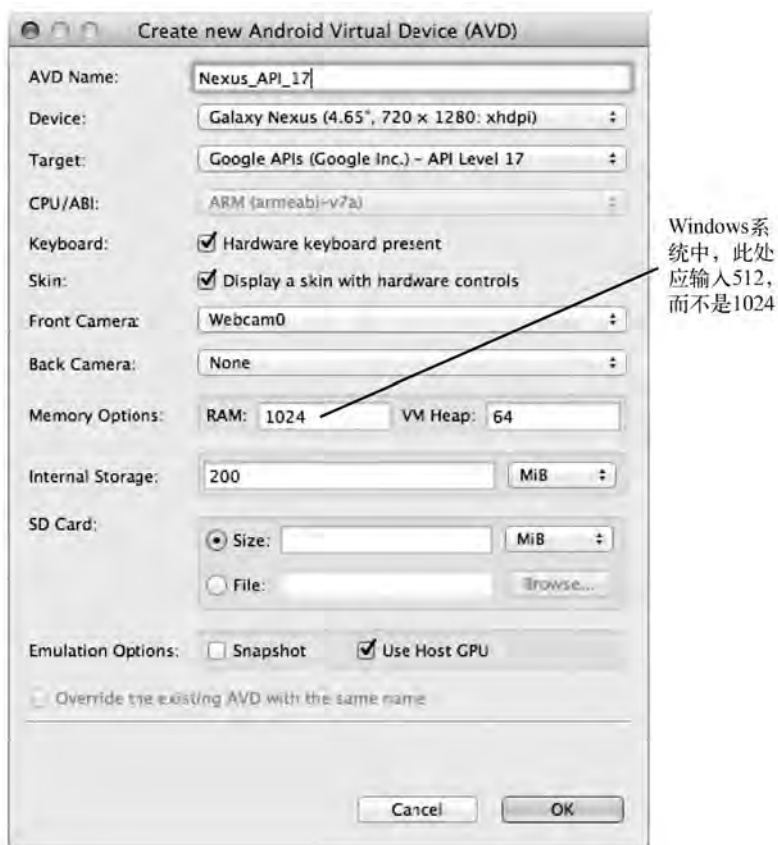
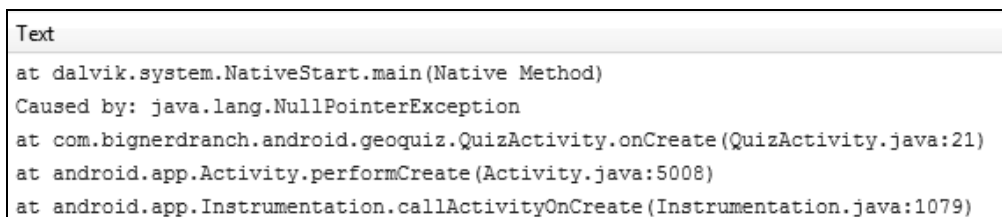


图1-13 创建新的AVD

AVD创建成功后，我们用它运行GeoQuiz应用。在包浏览器中，右击GeoQuiz项目文件夹。在弹出的右键菜单中，选择Run As → Android Application菜单项。Eclipse会自动找到新建的虚拟设备，安装应用包（APK），然后启动并运行应用。在此过程中，如果Eclipse询问是否使用LogCat自动监控，选择“Yes”。

启动虚拟机可能比较耗时，请耐心等待。设备启动完成，应用运行后，就可以在应用界面点击按钮，让toast告诉我们答案。（注意，如果应用启动运行后，我们凑巧不在电脑旁，回来时，就可能需要解锁AVD。如同一台真实设备，AVD闲置一段时间会自动锁上。）

假如GeoQuiz应用启动时或在我们点击按钮时发生崩溃，LogCat会出现在Eclipse工作区的底部。查看日志，可看到抢眼的红色异常信息，如图1-14所示。日志中的Text列可看到异常的名字以及发生问题的具体位置。



```
Text
at dalvik.system.NativeStart.main(Native Method)
Caused by: java.lang.NullPointerException
at com.bignerdranch.android.geoquiz.QuizActivity.onCreate(QuizActivity.java:21)
at android.app.Activity.performCreate(Activity.java:5008)
at android.app.Instrumentation.callActivityOnCreate(Instrumentation.java:1079)
```

图1-14 第21行代码发生了NullPointerException异常

将输入的代码与书中的代码作一下比较，找出错误并修改后，再尝试重新运行应用。

建议保持模拟器一直运行，这样就不必在反复运行调试应用时，痛苦地等待AVD启动了。单击回退按钮（即AVD模拟器上的U型箭头按钮）可以停止应用。需要调试变更时，再通过Eclipse重新运行应用。

虽然模拟器非常有用，但在真实设备上测试应用能够获得更准确的结果。在第2章中，我们将在真实硬件设备上运行GeoQuiz应用，并且为GeoQuiz应用添加更多地理知识问题，以供用户回答。

1.8 Android 编译过程

学习到这里，你可能对Android编译过程是如何工作的充满疑惑。我们已经知道在项目文件发生变化时，无需使用命令行工具，Eclipse便会自动进行编译。在整个编译过程中，Android开发工具将资源文件、代码以及AndroidManifest.xml文件（包含应用的元数据）编译生成.apk文件，如图1-15所示。为了让.apk应用能够在模拟器上运行，.apk文件必须以debug key签名。（分发.apk应用给用户时，应用必须以release key签名。如需了解更多有关编译过程的信息，可参考Android开发文档<http://developer.android.com/tools/publishing/preparing.html>。）

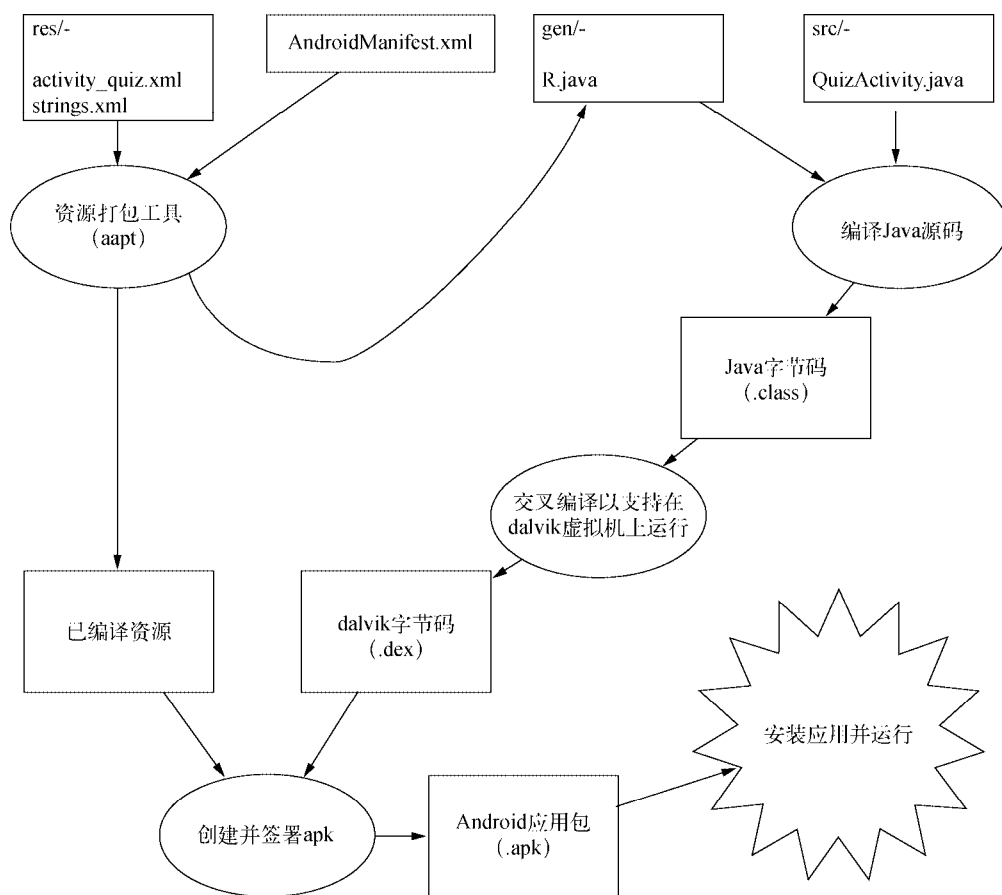


图1-15 编译GeoQuiz应用

那么，应用的`activity_quiz.xml`布局文件的内容该如何转变为View对象呢？作为编译过程的一部分，aapt（Android Asset Packaging Tool）将布局文件资源编译压缩紧凑后，打包到.apk文件中。当QuizActivity类的`onCreate(...)`方法调用`setContentView(...)`方法时，QuizActivity使用LayoutInflater类实例化定义在布局文件中的每一个View对象，如图1-16所示。

（除了在XML文件中定义视图的方式外，也可以在activity里使用代码的方式创建视图类。但应用展现层与逻辑层分离有很多好处，其中最主要的优点是可以利用SDK内置的设备配置改变，有关这一点将在第3章中详细讲解。）

有关XML不同属性的工作原理以及视图是如何显示在屏幕上的等更多信息，请参见第8章。

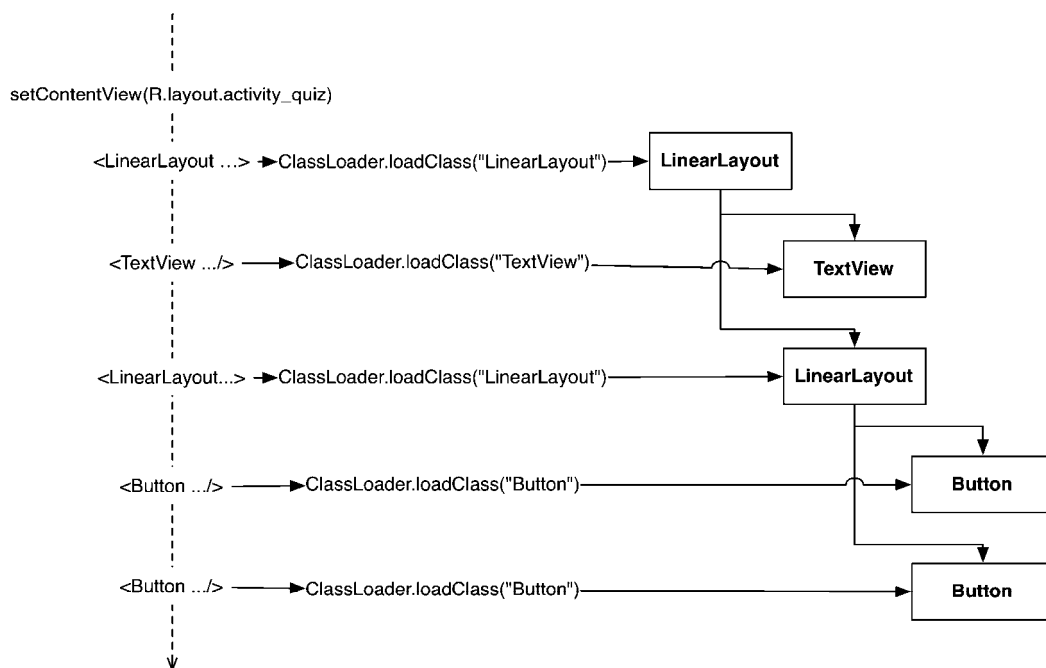


图1-16 activity_quiz.xml中的视图实例化

Android编译工具

截至目前，我们所看到的项目编译都是在Eclipse里执行的。编译功能已整合到正在使用的ADT开发插件中，插件调用了aapt等Android的标准编译工具，但编译过程本身是由Eclipse来管理的。

有时，出于种种原因，可能需要脱离Eclipse进行代码编译。最简单的方法是使用命令行编译工具。当前最流行的两大工具是maven和ant。使用ant的人相对较少，但使用起来相对比较容易。使用ant前应完成如下准备工作：

- ❑ 确保ant已安装并可以正常运行；
- ❑ 确保Android SDK的tools/和platform-tools/目录包含在可执行文件的搜索路径中。

现在切换到项目目录并执行以下命令：

```
$ android update project -p .
```

Eclipse项目生成器模板不包含ant可用的build.xml。以上命令将生成build.xml文件。该命令只需要运行这一次即可。

接下来就可以编译项目了。如果需要编译并签名为debug的.apk，请在同一目录下执行如下命令：


```
$ ant debug
```

该命令执行后即开始进行实际的项目编译，编译完成后在bin/*your-project-name*-debug.apk目录下生成相应的.apk文件。再通过以下安装命令安装.apk文件：

```
$ adb install bin/your-project-name-debug.apk
```

以上命令将把apk应用安装到当前连接的设备上，但不会运行它。要运行应用，需要在设备上手动启动安装的应用。

1