第 6 章

Android SDK版本与兼容

通过GeoQuiz应用，大家已经有了初步的开发体验。本章我们来纵览一下不同Android版本的背景知识。在本书后续学习及相对复杂的实际应用开发过程中，就会明白掌握本章内容是多么的重要。

6.1 Android SDK 版本

表6-1显示了各SDK版本、相应的Android固件版本及截至2013年3月使用各版本的设备比例。

表6-1 Android API级别、固件版本以及使用设备比例

API级别	代　　号	设备固件版本	使用的设备比例
17	Jelly Bean	4.2	1.6
16		4.1	14.9
15	Ice Cream Sandwich（ICS）	4.0.3、4.0.4	28.6
13	Honeycomb（只面向平板电脑）	3.2	0.9
12		3.1.x	0.3
10	Gingerbread	2.3.3 ~ 2.3.7	43.9
9		2.3.2、2.3.1、2.3	0.2
8	Froyo	2.2.x	7.5
7	Eclair	2.1.x	1.9

每一个具有发布代号的版本随后都会有对应的增量发布版本。例如，Ice Cream Sandwich最初的发布版本为Android 4.0（API 14级）。但没过多久，它就被Android 4.0.3及4.0.4（API 15级）的增量发行版本取代。

当然，表6-1中的比例会不断变化，但我们可从这些比例中看出一种重要趋势，即新版本发布后，运行老版本的Android设备是不会立即得到升级或者被取代的。截至2013年3月，半数的设备仍然运行着代号为Froyo或Gingerbread的SDK版本。Android 2.3.7（Gingerbread最后的升级版本）发布于2011年9月。而2012年11月发布的Android 4.2版本的运行设备，所占比例只有1.6%。

（感兴趣的话，可去http://developer.android.com/resources/dashboard/platform-versions.html查看表6-1数据的动态更新。也可访问该网址获取最新的趋势数据。）

为什么仍有这么多设备运行着Android老版本系统？主要是由于Android设备生产商和运营商之间的激烈竞争。运营商希望拥有其他运营商所没有的具有特色功能的手机。设备生产商也有同样的压力——所有手机都基于相同的操作系统，而他们又希望在竞争中脱颖而出。最终，在市场和运营商的双重压力下，各种专属的、无法升级的定制版Android设备涌向市场，令人眼花缭乱、目不暇接。

具有专属定制版本的Android设备不能运行Google发布的新版本Android系统。因此，用户只能寄希望于兼容的专属版本升级。 然而，即便可以获得这种升级，通常也是Google新版本发布后数月的事情了。生产商往往更愿意投入资源推出新设备，而不是保持旧设备的更新升级。此外，老设备的硬件有时无法满足运行Android新版本也是一个主要因素。

6.2 Android 编程与兼容性问题

各种设备迟缓的版本升级再加上Google定期的新版本发布，给Android编程带来了重大的兼容性问题。为取得更广阔的市场，对于运行Froyo、Gingerbread、Honeycomb、Ice Cream Sandwich和Jelly Bean这些版本的Android设备，以及各种款式尺寸的设备，Android开发人员必须保证应用兼容它们并运行良好。

应用开发时，不同尺寸设备的处理要比想象中的简单。手机屏幕尺寸虽然繁多，但Android布局系统为编程适配做了很好的工作。平板设备处理起来会复杂一些，但使用配置修饰符可帮我们完成屏幕适配的任务（第22章会介绍相关知识）。不过，对于同样运行着Android系统的Google TV，由于UI差异太大，因此通常需要针对它开发单独的应用。

不同版本的兼容就是另一回事了。如发布的是增量版本，向下兼容通常问题不大。然而，如果发布的是重大全新版本，这才是真正的大问题。

6.2.1 全新的系统版本——Honeycomb

全新Honeycomb版本发布的前后间是Android兼容性这一重大问题出现的转折点。Honeycomb版本的发布是Android世界的一个重大转变分支，同时该版本还引入了全新的UI和构造组件。Honeycomb专为平板设备和Google TV而开发（未被广泛采用），所以直到Ice Cream Sandwich的发布，它才开发完成并正式发布给终端用户使用。随后又经历了几次增量版本升级。

事实上，超过半数的设备仍然运行着Gingerbread甚至更老的版本。开发者无法彻底放弃老版本。尽管老版本设备最终会逐渐退出，但退出过程可能超乎想象的缓慢。

因此Android开发者必须花费时间保证向后兼容，架起Gingerbread（API 10级）和Honeycomb（API 11级）以及更高版本开发间的桥梁。尽管Android以及第三方库提供了相应的兼容性编程支持。但兼容性问题已实实在在地增加了Android编程学习的复杂性。

同时，这也意味着我们常常需要学习完成同一件事的两种方法，以及如何将这两种方法进行整合。而有时虽然只学习一种方法，但学习起来却异常地复杂，因为我们要努力实现至少两套开发需求。

如果你的Android编程学习计划可以推迟，建议你再等等，等Gingerbread设备基本退出市场了再开始学习。等不到那个时候？那么我们希望你能明白Android编程某些复杂问题究竟是怎么一回事。

新建GeoQuiz项目时，在新建应用向导界面，如图6-1所示，有三处SDK版本需要设置。（注意Android的“SDK版本”和“API级别”代表同一意思，可以交替使用。）

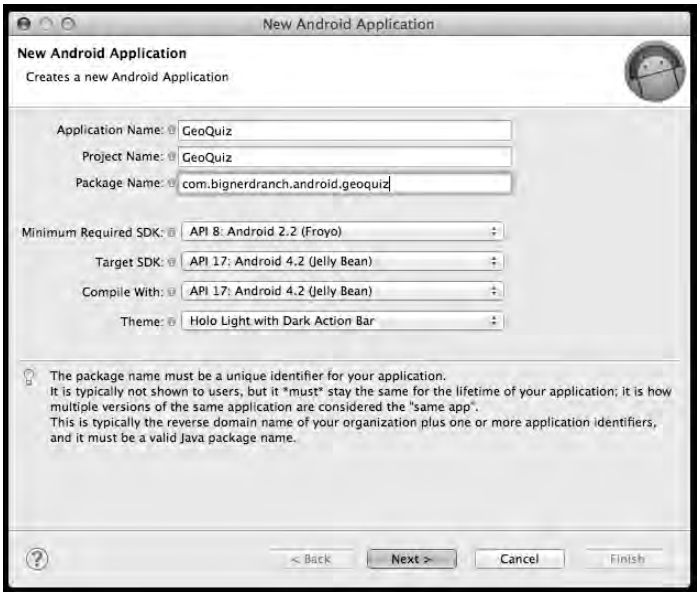

图6-1　创建新项目向导，还有印象吗

我们来看看项目中的这些设置都位于哪里，然后解释默认设置并搞清楚要如何更改它们。

SDK最低需求版本及SDK目标版本都设置在manifest配置文件里。在包浏览器中，重新打开AndroidManifest.xml配置文件。在`uses-sdk`元素节点下，查看android:minSdkVersion和android:targetSdkVersion的属性值。在manifest配置文件中寻找SDK最低版本的做法如代码清单6-1所示。

代码清单6-1　在配置文件中寻找minSdkVersion（AndroidManifest.xml）

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.bignerdranch.android.geoquiz"
  android:versionCode="1"
  android:versionName="1.0" >

  <uses-sdk
    android:minSdkVersion="8"
    android:targetSdkVersion="17" />

  ...

</manifest>
```

6.2.2 SDK最低版本

之前讲过，manifest是操作系统用来与应用交互的元数据。以最低版本设置值为标准，操作系统会拒绝将应用安装在系统版本低于标准的设备上。

例如，设置版本为API 8级(Froyo)，便赋予了系统在运行Froyo及以上版本的设备上安装GeoQuiz应用的权限。显然，在运行Eclair版本的设备上，系统会拒绝安装GeoQuiz应用。

再看表6-1，我们就会明白为什么Froyo作为SDK最低版本比较合适，因为有95%的在用设备支持安装此应用。

6.2.3 SDK目标版本

目标版本的设定值可告知Android：应用是设计给哪个API级别去运行的。大多数情况下，目标版本即最新发布的Android版本。

什么时候需要降低SDK目标版本呢？新发布的SDK版本会改变应用在设备上的显示方式，甚至连后台操作系统运行也会受到影响。如果应用已开发完成，需确认它在新版本上能否如预期那样正常运行。查看网址http://developer.android.com/reference/android/os/Build.VERSION_CODES.html上的文档，检查可能出现问题的地方。根据分析结果，要么修改应用去适应新版本系统，要么降低SDK目标版本。降低SDK目标版本可以保证的是，即便在高于目标版本的设备上，应用仍然可以正常运行，且运行行为仍和目标版本保持一致。这是因为新发布版本中的变化已被忽略。

6.2.4 SDK编译版本

图6-1中，最后一项标为Compile With的是SDK编译版本设置。该设置不会出现在manifest配置文件里。SDK最低版本和目标版本会通知给操作系统，而SDK编译版本是我们和编译器之间的私有信息。

Android的特色功能是通过SDK中的类和方法展现的。在编译代码时，SDK编译版本或编译目标指定具体要使用的系统版本。Eclipse在寻找类包导入语句中的类和方法时，编译目标确定具体的基准系统版本。

编译目标的最佳选择为最新的API 级别（当前级别为17，代号为Jelly Bean）。当然，需要的话，也可以改变应用的编译目标。例如，Android新版本发布时，可能就需要更新编译目标。

要改变编译目标，可在包浏览器中，右键单击GeoQuiz项目并选择Properties菜单。在弹出对话框的左边，选择Android以查看所有不同编译目标的选项，如图6-2所示。

知道Google API与Android开源项目编译目标之间的区别吗？Google API包括Android API以及Google附加API（即支持使用Google地图服务的重要API）。

GeoQuiz项目的编译目标无需变动，单击Cancel按钮，继续我们的学习。

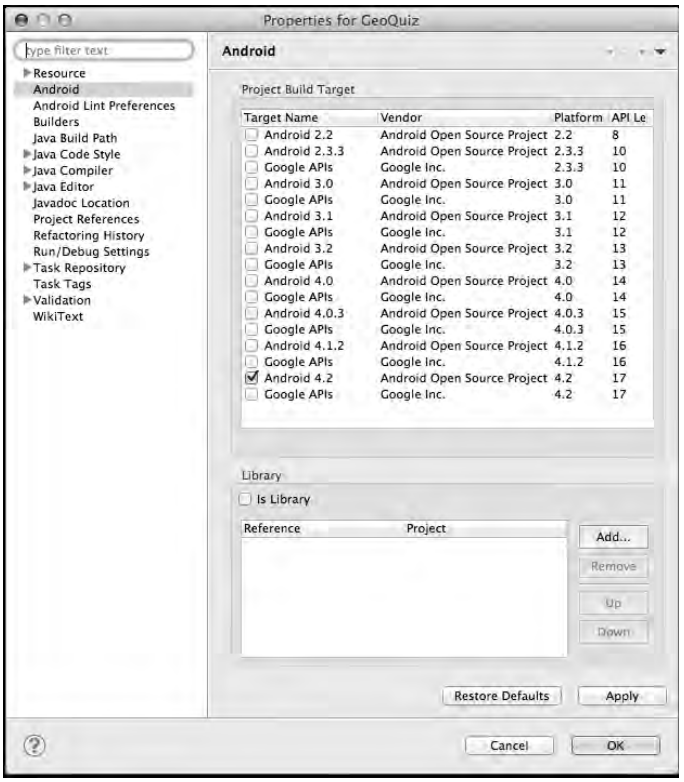

图6-2　更改编译目标

6.2.5　安全添加新版本API中的代码

GeoQuiz应用的SDK最低版本和编译版本间的差异带来的兼容问题需要我们来处理。例如，在GeoQuiz应用中，如果调用了Froyo（API 8级）以后的SDK版本中的代码会怎么样呢？结果显示，当在Froyo设备上安装并运行应用时，应用会发生崩溃。

该问题可以说是曾经的测试噩梦。然而，受益于Android Lint的不断改进，最终，当新版本API代码在老版本系统上运行时，可能存在的问题在运行时就被捕获了。如果使用了高版本系统API中的代码，Android Lint会提示编译错误。

目前GeoQuiz应用中的简单代码都来自于API 8级或更早版本。现在，我们来增加API 11级的代码，看看会发生什么。

打开QuizActivity.java文件，在`onCreate(Bundle)`方法中，添加代码清单6-2所示代码，在操作栏显示子标题，用来指定测试问题属于哪一地理知识领域。

代码清单6-2　添加操作栏代码（QuizActivity.java）

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
```

```
Log.d(TAG, "onCreate() called");
setContentView(R.layout.activity_quiz);

ActionBar actionBar = getActionBar();
actionBar.setSubtitle("Bodies of Water");

mIsCheater = false;
```

（简单起见，我们使用了固定字符串。如果真的需要显示子标题，或者根据不同的问题类别显示不同的子标题，可新增字符串资源并引用它们。）

通过类包组织导入功能自动导入ActionBar类。ActionBar类来自于API 11级，所以在低于这个版本的设备上运行代码会发生崩溃。我们会在第16章学习更多有关ActionBar的知识。这里仅用它作为Froyo不常用的代码示例。

组织导入ActionBar类后，在包浏览器中，选择项目GeoQuiz，然后选择Android Tools → Run Lint: Check for Common Errors菜单项。因为SDK编译版本为API 17级，所以编译器本身编译代码没有问题。然而，Android Lint知道项目SDK最低版本的信息，因此会抛出兼容性问题的错误信息。

错误信息显示为Class requires API level 11（current min is 8）。基本上，除非兼容性问题得到解决，否则Android Lint是不会让我们进行编译的。

该怎么消除这些错误信息呢？一种办法是提升SDK最低版本到11。然而，提升SDK最低版本只是回避了兼容性问题。如果应用不能安装在Gingerbread和老版本设备上，那么也就不存在新老系统的兼容性问题了。因此，实际上这并没有真正地解决兼容性问题。

比较好的方法是将ActionBar代码置于检查Android设备版本的条件语句中，如代码清单6-3所示。

代码清单6-3 首先检查设备的编译版本

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Log.d(TAG, "onCreate() called");
    setContentView(R.layout.activity_quiz);

    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB) {
        ActionBar actionBar = getActionBar();
        actionBar.setSubtitle("Bodies of Water");
    }
}
```

Build.VERSION.SDK_INT常量代表了Android设备的版本号。可将该常量同代表Honeycomb版本的常量进行比较。（版本号清单可参考网页http://developer.android.com/reference/android/os/Build.VERSION_CODES.html。）

现在ActionBar代码只有在Honeycomb或更高版本的设备上运行应用才会被调用。应用代码在Froyo设备上终于安全了，Android Lint**应该**也满意了吧。然而，如尝试再次运行应用，错误依然如故。

禁止Lint提示兼容性问题

很不幸，尽管我们已经处理了兼容性问题，但Android Lint却无从知晓，所以必须明令禁止其再提示兼容性问题。如代码清单6-4所示，在onCreate(Bundle)实现方法前添加如下注解。

代码清单6-4　使用注解向Android Lint声明版本信息

```
@TargetApi(11)
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Log.d(TAG, "onCreate() called");
    setContentView(R.layout.activity_quiz);

    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB) {
        ActionBar actionBar = getActionBar();
        actionBar.setSubtitle("TFFTT");
    }
}
```

已经有了if语句的判断处理，为什么还需要添加以上注解呢？把Android编程想象成一处海滩。海滩附近的海水里有一群鲨鱼——在旧版本设备上使用新版本方法或类时抛出的运行异常。Android Lint则是海滩上巡逻的救生员。一旦有鲨鱼靠近的危险，就立即跳入水中解救我们。

代码清单6-4中的新增代码做了两件事：使用驱鲨剂以及婉拒救生员的救助。`if`语句就是驱鲨剂。`getActionBar()`方法置于`if`语句中，只有在语句存在时才会被调用，这样鲨鱼就无法攻击我们了。注解`@TargetApi(11)`则向救生员（Android Lint）的救助进行婉拒，表明不要担心鲨鱼——我已经控制住了局面。这样，救生员也就无需下水进行救助了。

所以，在使用`@TargetApi`注解告诉救生员无需救助时，请确认已使用了`SDK_INT`防鲨剂，否则将被运行异常的鲨鱼吃掉。

在Honeycomb或更高版本的设备上运行GeoQuiz，确认子标题显示正常，如图6-3所示。

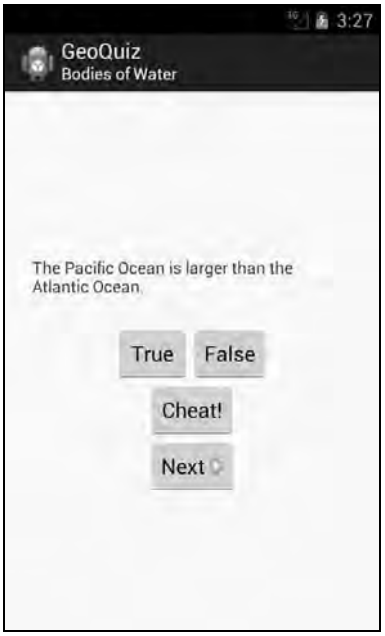

图6-3　显示子标题的操作栏

也可以在Froyo或Gingerbread设备（虚拟或实体）上运行GeoQuiz应用。当然，用户界面不会显示操作栏或子标题，但可验证应用是否仍能正常运行。

6.3 使用 Android 开发者文档

Android Lint错误信息可告知不兼容代码所属的API级别。也可在Android开发者文档里查看各API级别特有的类和方法。

最好现在就开始熟悉使用开发者文档。我们不可能记住Android SDK中的海量信息，而且新版本系统也会定期发布，因此，只需学会查阅SDK文档，不断学习新的东西并掌握它们即可。

Android开发者文档是优秀而丰富的信息来源。文档的主页是 http://developer.android.com/。文档分为三大部分，即*设计*、*开发*和*发布*。*设计*部分的文档包括应用UI设计的模式和原则。*开发*部分包括SDK文档和培训资料。*发布*部分告知我们如何在Google Play商店上或通过开放发布模式准备并发布应用。有机会的话，一定要仔细研读这些资料。

*开发*部分可细分为四大块内容：

- Android培训，初级和中级开发者的培训模块，包括可下载的示例代码；
- API使用指导，基于主题的应用组件、特色功能详述以及它们的最佳实践；
- 参考文档，SDK中类、方法、接口、属性常量等可搜索、交叉链接的参考文档；
- 开发工具，开发工具的描述及下载链接。

无需联网也可查看文档。浏览下载SDK的文件系统，会发现有一个docs目录，该目录包含了全部的Android开发者文档内容。

开发时，为确定`getActionBar()`方法所属的API级别，使用文档浏览器右上角的搜索框搜索该方法。第一条搜索结果是有关操作栏的API使用指导。但我们想要的结果位于参考文档部分。很简单，点击左边的Reference过滤搜索结果即可。

选择第一条结果，进入`Activity`类的参考文档页面，如图6-4所示。该页面顶部的链接可以链接到不同的部分。点击Methods链接可以查看`Activity`方法列表。

向下滚动，找到并点击`getActionBar()`方法名查看具体的方法描述。从该方法名的右边可以看到，`getActionBar()`方法最早被引入的API级别是API 11级。

如想查看`Activity`类的哪些方法可调用于API 8级，可按API级别过滤引用，如图6-5所示。在页面左边按包索引的类列表上方，找到API级别过滤框。点击展示下拉表单，然后选择数字8。我们会发现，所有API 8级以后引入的方法都自动变为灰色被过滤掉了。

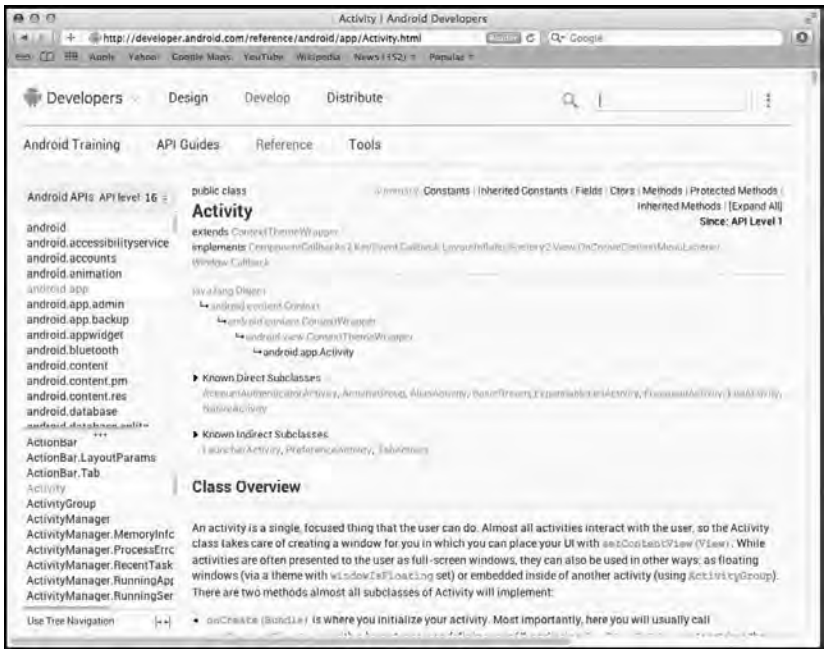

图6-4　Activity参考文档页面

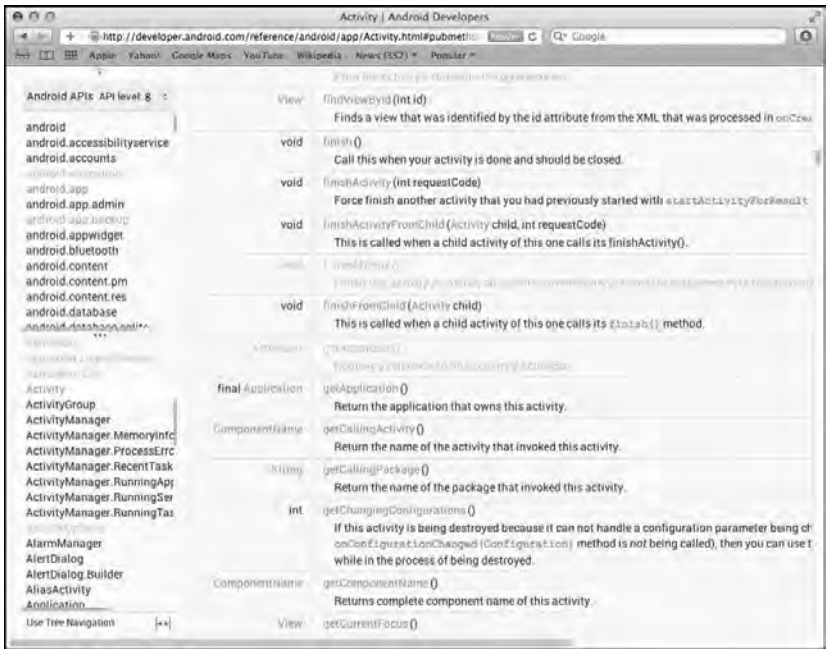

图6-5　以API 8级过滤Activity类方法

在后续章节的学习过程中，记得经常查阅开发者文档。解决章末的挑战练习，探究某些类、方法或其他主题时，同样需要查阅相关的文档资料。Android文档也一直在更新和改进，新知识新概念也不断涌现，因此大家需要更加努力地学习。

6.4 挑战练习：报告编译版本

在GeoQuiz应用页面布局上增加一个TextView组件，向用户报告设备运行系统的API级别，如图6-6所示。

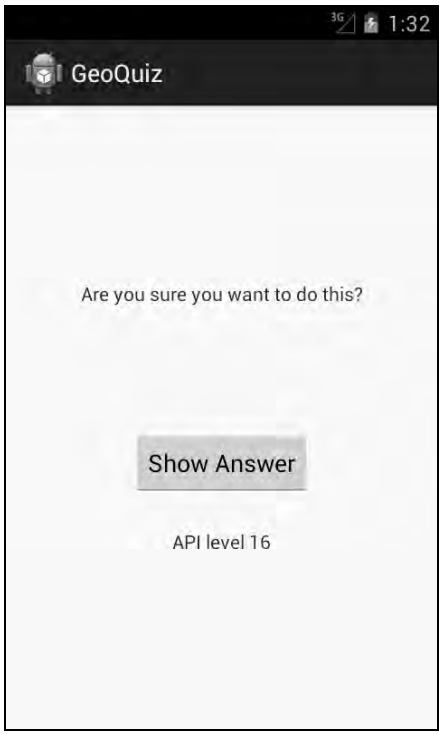

图6-6 完成后的用户界面

只有在应用运行时才能知道设备的编译版本，所以我们不能直接在布局上设置TextView的值。打开Android文档中的TextView参考页，查找TextView的文本赋值方法。寻找可以接受字符串或CharSequence的单参数方法。

另外，可运用TextView参考手册里列出的其他XML属性来调整文字的尺寸或样式。