第 2 章

Android与MVC设计模式

本章我们将对GeoQuiz应用进行功能升级，让应用能够提供更多的地理知识测试题目，如图2-1所示。

图2-1　更多测试题目

为实现目标，需要为GeoQuiz项目新增一个`TrueFalse`类。该类的一个实例用来封装代表一道题目。

然后再创建一个`TrueFalse`数组对象交由`QuizActivity`管理。

2.1　创建新类

在包浏览器中，右键单击`com.bignerdranch.android.geoquiz`类包，选择New → Class

菜单项，弹出图2-2所示的对话框。类名处填入TrueFalse，保持默认的超类java.lang.Object不变，然后单击Finish按钮。

图2-2 创建TrueFalse类

在TrueFalse.java中，新增两个成员变量和一个构造方法，如代码清单2-1所示。

代码清单2-1 TrueFalse类中的新增代码（TrueFalse.java）

```
public class TrueFalse {
    private int mQuestion;

    private boolean mTrueQuestion;

    public TrueFalse(int question, boolean trueQuestion) {
        mQuestion = question;
        mTrueQuestion = trueQuestion;
    }
}
```

mQuestion为什么是int类型的，而不是String类型的呢？变量mQuestion用来保存地理知识问题字符串的资源ID。资源ID总是int类型，所以这里设置它为int而不是String类型。变量mTrueQuestion用来确定答案正确与否，需要使用到的问题字符串资源稍后会处理。

新增的两个变量需要getter与setter方法。为避免手工输入，可设置由Eclipse自动生成getter与setter方法。

生成getter与setter方法

首先，配置Eclipse识别成员变量的m前缀，并且对于boolean类型的成员变量使用is而不是get前缀。

打开Eclipse首选项对话框（Mac用户选择Eclipse菜单，Windows用户选择Windows → Preferences菜单）。在Java选项下选择Code Style。

在Conventions for variable names:表中，选择Fields行，如图2-3所示。单击右边的Edit按钮，增加m作为fields的前缀。然后增加s作为Static Fields的前缀。（GeoQuiz项目不会用到s前缀，但在之后的项目中会用到。）

确认Use ‘is’ prefix for getters that return boolean选择框被勾选后，单击OK按钮，如图2-3所示。

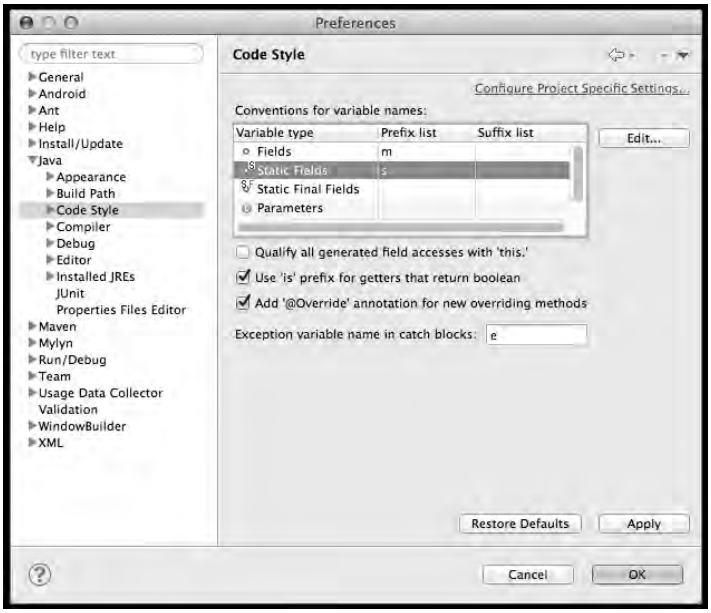

图2-3　设置Java代码风格首选项

刚才设置的前缀有何作用？现在，当要求Eclipse为mQuestion生成getter方法时，它生成的是getQuestion()而不是getMQuestion()方法；而在为mTrueQuestion生成getter方法时，生成的则是isTrueQuestion() 而不是isMTrueQuestion()方法。

回到TrueFalse.java中，右击构造方法后方区域，选择Source → Generate Getters And Setters... 菜单项。点击Select All按钮，为每个变量都生成getter与setter方法。

单击OK按钮，Eclipse随即生成了这四个getter与setter方法的代码，如代码清单2-2所示。

代码清单2-2　生成getter与setter方法（TrueFalse.java）

```
public class TrueFalse {
    private int mQuestion;
```

```
    private boolean mTrueQuestion;

    public TrueFalse(int question, boolean trueQuestion) {
        mQuestion = question;
        mTrueQuestion = trueQuestion;
    }

    public int getQuestion() {
        return mQuestion;
    }

    public void setQuestion(int question) {
        mQuestion = question;
    }

    public boolean isTrueQuestion() {
        return mTrueQuestion;
    }

    public void setTrueQuestion(boolean trueQuestion) {
        mTrueQuestion = trueQuestion;
    }

}
```

这样TrueFalse类就完成了。稍后，我们会修改QuizActivity类，以配合TrueFalse类的使用。现在，我们先来整体了解一下GeoQuiz应用，看看各个类是如何一起协同工作的。

我们使用QuizActivity创建TrueFalse数组对象。继而通过与TextView以及三个Button的交互，在屏幕上显示地理知识问题，并根据用户的回答做出适当的反馈，如图2-4所示。

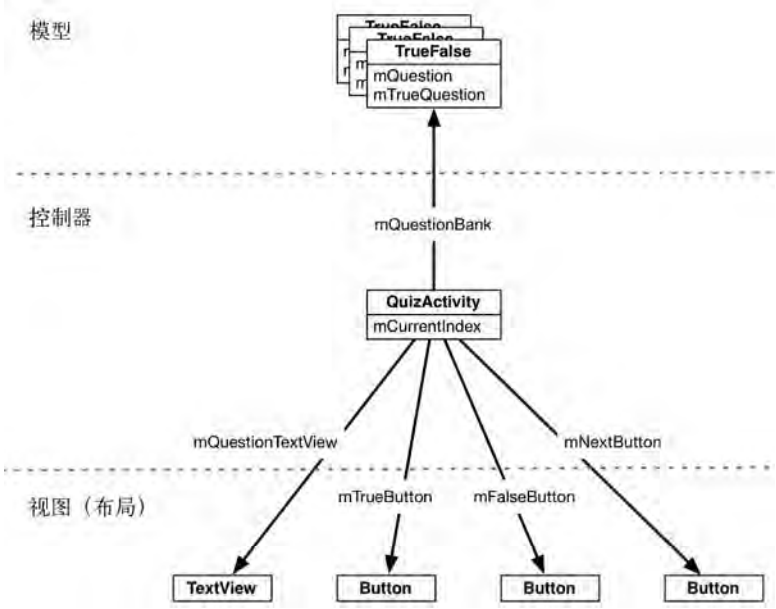

图2-4 GeoQuiz应用对象图解

2.2　Android 与 MVC 设计模式

如图2-4所示，应用的对象按模型、控制器和视图的类别被分为三部分。Android应用是基于模型–控制器–视图（Model-View-Controller，简称MVC）的架构模式进行设计的。MVC设计模式表明，应用的任何对象，归根结底都属于模型对象、视图对象以及控制对象中的一种。

- 模型对象存储着应用的数据和业务逻辑。模型类通常被设计用来映射与应用相关的一些事物，如用户、商店里的商品、服务器上的图片或者一段电视节目。又或是GeoQuiz应用里的地理知识问题。模型对象不关心用户界面，它存在的唯一目的就是存储和管理应用数据。
  Android应用里的模型类通常就是我们创建的定制类。应用的全部模型对象组成了模型层。
  GeoQuiz的模型层由`TrueFalse`类组成。
- 视图对象知道如何在屏幕上绘制自己以及如何响应用户的输入，如用户的触摸等。一个简单的经验法则是，凡是能够在屏幕上看见的对象，就是视图对象。
  Android默认自带了很多可配置的视图类。当然，也可以定制开发自己的视图类。应用的全部视图对象组成了视图层。
  GeoQuiz应用的视图层是由activity_quiz.xml文件中定义的各类组件构成的。
- 控制对象包含了应用的逻辑单元，是视图与模型对象的联系纽带。控制对象被设计用来响应由视图对象触发的各类事件，此外还用来管理模型对象与视图层间的数据流动。
  在Android的世界里，控制器通常是`Activity`、`Fragment`或`Service`的一个子类（第7章和第29章将分别介绍fragment和service的概念）。
  当前，GeoQuiz的控制层仅由`QuizActivity`类组成。

图2-5展示了在响应用户单击按钮等事件时，对象间的交互控制数据流。注意，模型对象与视图对象不直接交互。控制器作为它们间的联系纽带，接收来自对象的消息，然后向其他对象发送操作指令。

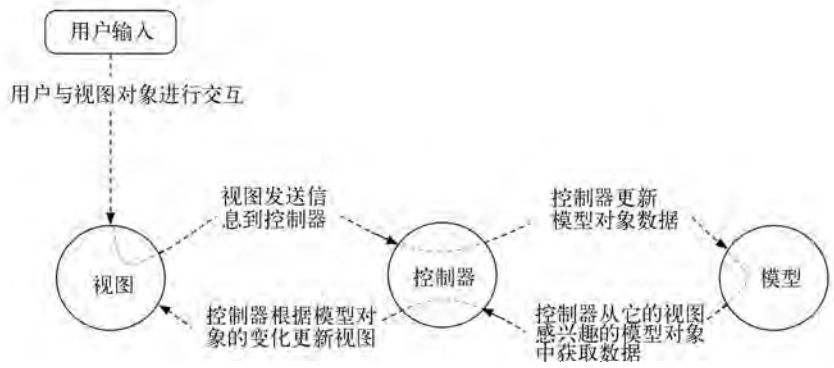

图2-5　MVC 数据控制流与用户交互

使用MVC设计模式的好处

随着应用功能的持续扩展，应用往往会变得过于复杂而让人难以理解。以Java类的方式组织代码有助于我们从整体视角设计和理解应用。这样，我们就可以按类而不是一个个的变量和方法去思考设计开发问题。

同样，把Java类以模型、视图和控制层进行分类组织，也有助于我们设计和理解应用。这样，我们就可以按层而非一个个类来考虑设计开发了。

尽管GeoQuiz不是一个复杂的应用，但以MVC分层模式设计它的好处还是显而易见的。接下来，我们来升级GeoQuiz应用的视图层，并为它添加一个Next按钮。我们会发现，在添加Next按钮的过程中，可完全不用考虑刚才创建的`TrueFalse`类的存在。

使用MVC模式还可以让类的复用更加容易。相比功能多而全的类，有特别功能限定的专用类更加有利于代码的复用。

举例来说，模型类`TrueFalse`与用作显示问题的组件毫无代码逻辑关联。这样，就很容易在应用里按需自由使用`TrueFalse`类。假设现在想显示所有地理知识问题列表，很简单，直接复用`TrueFalse`对象逐条显示就可以了。

2.3 更新视图层

了解了MVC设计模式后，现在我们来更新GeoQuiz应用的视图层，为其添加一个Next按钮。

在Android编程中，视图层对象通常生成自XML布局文件。GeoQuiz应用唯一的布局定义在activity_quiz.xml文件中。布局定义文件需要更新的地方如图2-6所示。（注意，为节约版面，无需变化的组件属性这里就不再列出了。）

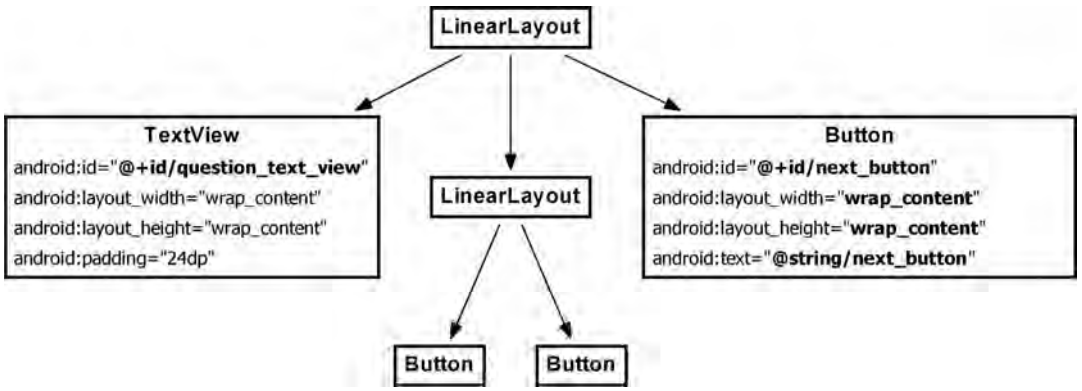

图2-6 新增的按钮

应用视图层所需的变动操作如下：

- 删除`TextView`的`android:text`属性定义。这里不再需要硬编码问题。

- 为TextView新增android:id属性。TextView组件需要一个资源ID，以便在QuizActivity代码中为它设置要显示的文字。
- 以根LinearLayout为父组件，新增一个Button组件。

回到activity_quiz.xml文件中，参照代码清单2-3完成XML文件的相应修改。

代码清单2-3　新增按钮以及文本视图的调整（activity_quiz.xml）

```
<LinearLayout
  ... >

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

  <LinearLayout
    ... >

    ...

  </LinearLayout>

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

</LinearLayout>
```

保存activity_quiz.xml文件。这时，可能会得到一个熟悉的错误弹框提示，提醒我们缺少字符串资源。

返回到res/values/strings.xml文件中。删除硬编码的问题字符串，添加新按钮所需的字符串资源定义，如代码清单2-4所示。

代码清单2-4　更新字符串资源定义（strings.xml）

```
...

  <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="next_button">Next</string>
  <string name="correct_toast">Correct!</string>

  ...
```

保持strings.xml文件处于打开状态，添加向用户显示的一系列地理知识问题的字符串，如代码清单2-5所示。

代码清单2-5 新增问题字符串（strings.xml）

```
...
    <string name="incorrect_toast">Incorrect!</string>
    <string name="menu_settings">Settings</string>
    <string name="question_oceans">The Pacific Ocean is larger than
      the Atlantic Ocean.</string>
    <string name="question_mideast">The Suez Canal connects the Red Sea
      and the Indian Ocean.</string>
    <string name="question_africa">The source of the Nile River is in Egypt.</string>
    <string name="question_americas">The Amazon River is the longest river
      in the Americas.</string>
    <string name="question_asia">Lake Baikal is the world\'s oldest and deepest
      freshwater lake.</string>
    ...
```

注意最后一行字符串定义中的“\”，这里，我们使用了转义字符对符号“'”进行了处理。在字符串资源定义中，也可使用其他常见的转义字符，比如\n新行符。

保存修改过的文件。然后回到activity_quiz.xml文件中，在图形布局工具里预览确认修改后的布局文件。

至此，GeoQuiz应用视图层的操作就全部完成了。接下来，我们对控制层的QuizActivity类进行代码编写与资源引用，从而最终完成QeoQuiz应用。

2.4 更新控制层

在上一章，GeoQuiz应用控制层的QuizActivity类的处理逻辑很简单：显示定义在activity_quiz.xml文件中的布局对象，通过在两个按钮上设置监听器，响应用户点击事件并创建提示消息。

既然现在有了更多的地理知识问题可以检索与展示，那么QuizActivity类将需要更多的处理逻辑来关联GeoQuiz应用的模型层与视图层。

打开QuizActivity.java文件，添加TextView和新的Button变量。另外，再创建一个TrueFalse对象数组以及一个该数组的索引变量，如代码清单2-6所示。

代码清单2-6 增加按钮变量及TrueFalse对象数组（QuizActivity.java）

```
public class QuizActivity extends Activity {

    private Button mTrueButton;
    private Button mFalseButton;
    private Button mNextButton;
    private TextView mQuestionTextView;

    private TrueFalse[] mQuestionBank = new TrueFalse[] {
        new TrueFalse(R.string.question_oceans, true),
        new TrueFalse(R.string.question_mideast, false),
        new TrueFalse(R.string.question_africa, false),
        new TrueFalse(R.string.question_americas, true),
        new TrueFalse(R.string.question_asia, true),
    };

    private int mCurrentIndex = 0;
    ...
```

这里，我们通过多次调用TrueFalse类的构造方法，创建了一个TrueFalse对象数组。

（在更为复杂的项目里，这类数组的创建和存储我们会单独处理。在本书后续应用开发中，将会介绍更好的模型数据存储管理方式。现在，简单起见，我们选择在控制层代码中创建数组。）

通过使用mQuestionBank数组、mCurrentIndex变量以及TrueFalse对象的存取方法，从而把一系列问题显示在屏幕上。

首先，引用TextView，并将其文本内容设置为当前数组索引所指向的问题，如代码清单2-7所示。

代码清单2-7　使用TextView（QuizActivity.java）

```
public class QuizActivity extends Activity {

    ...

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

        mQuestionTextView = (TextView)findViewById(R.id.question_text_view);
        int question = mQuestionBank[mCurrentIndex].getQuestion();
        mQuestionTextView.setText(question);

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

保存所有文件，确保没有错误发生。然后运行GeoQuiz应用。可看到数组存储的第一个问题显示在TextView上了。

现在我们来处理Next按钮。首先引用Next按钮，然后为其设置监听器View.OnClick-Listener。该监听器的作用是：递增数组索引并相应更新显示TextView的文本内容。如代码清单2-8所示。

代码清单2-8　使用新增按钮（QuizActivity.java）

```
public class QuizActivity extends Activity {

    ...

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

        mQuestionTextView = (TextView)findViewById(R.id.question_text_view);
        int question = mQuestionBank[mCurrentIndex].getQuestion();
        mQuestionTextView.setText(question);

        ...
```

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

        mNextButton = (Button)findViewById(R.id.next_button);
        mNextButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mCurrentIndex = (mCurrentIndex + 1) % mQuestionBank.length;
                int question = mQuestionBank[mCurrentIndex].getQuestion();
                mQuestionTextView.setText(question);
            }
        });
    }
}
```

我们发现，用来更新mQuestionTextView变量的相同代码分布在了两个不同的地方。参照代码清单2-9，花点时间把公共代码放在单独的私有方法里。然后在mNextButton监听器里以及onCreate(Bundle)方法的末尾分别调用该方法，从而初步设置activity视图中的文本。

代码清单2-9 使用updateQuestion()封装公共代码（QuizActivity.java）

```
public class QuizActivity extends Activity {

    ...

    private void updateQuestion() {
        int question = mQuestionBank[mCurrentIndex].getQuestion();
        mQuestionTextView.setText(question);
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...

        mQuestionTextView = (TextView)findViewById(R.id.question_text_view);
        int question = mQuestionBank[mCurrentIndex].getQuestion();
        mQuestionTextView.setText(question);

        mNextButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mCurrentIndex = (mCurrentIndex + 1) % mQuestionBank.length;
                int question = mQuestionBank[mCurrentIndex].getQuestion();
                mQuestionTextView.setText(question);
                updateQuestion();
            }
        });

        updateQuestion();
    }
}
```

现在，运行GeoQuiz应用验证新增的Next按钮。

一切正常的话，问题应该已经完美显示出来了。下面我们开始处理问题答案的显示。同样地，为避免将相同的代码写在两处，我们将它们封装在一个私有方法里以供调用。

要添加到QuizActivity类的方法如下：

```
private void checkAnswer(boolean userPressedTrue)
```

该方法接受传入的boolean类型的变量参数，可用于判别用户单击了True还是False按钮。

然后，将用户的答案同当前TrueFalse对象中的答案作比较。最后，根据答案的正确与否，生成一个Toast向用户提示反馈消息。

在QuizActivity.java文件中，添加checkAnswer(boolean)方法的实现代码，如代码清单2-10所示。

代码清单2-10　增加方法checkAnswer(boolean)（QuizActivity.java）

```
public class QuizActivity extends Activity {

    ...

    private void updateQuestion() {
        int question = mQuestionBank[mCurrentIndex].getQuestion();
        mQuestionTextView.setText(question);
    }

    private void checkAnswer(boolean userPressedTrue) {
        boolean answerIsTrue = mQuestionBank[mCurrentIndex].isTrueQuestion();

        int messageResId = 0;

        if (userPressedTrue == answerIsTrue) {
            messageResId = R.string.correct_toast;
        } else {
            messageResId = R.string.incorrect_toast;
        }

        Toast.makeText(this, messageResId, Toast.LENGTH_SHORT)
            .show();
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...

    }
}
```

在按钮的监听器里，调用checkAnswer(boolean)方法，如代码清单2-11所示。

代码清单2-11　调用方法checkAnswer(boolean)（QuizActivity.java）

```
public class QuizActivity extends Activity {

    ...
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...

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

            @Override
            public void onClick(View v) {
                Toast.makeText(QuizActivity.this,
                               R.string.incorrect_toast,
                               Toast.LENGTH_SHORT).show();
                checkAnswer(true);
            }
        });

        mFalseButton = (Button)findViewById(R.id.false_button);
        mFalseButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(QuizActivity.this,
                               R.string.correct_toast,
                               Toast.LENGTH_SHORT).show();
                checkAnswer(false);
            }
        });

        mNextButton = (Button)findViewById(R.id.next_button);
        ...

    }
}
```

GeoQuiz应用已经为再次运行做好准备了，接下来让我们在真实设备上运行一下吧。

2.5 在设备上运行应用

本节，我们将学习系统设备以及应用的设置方法，从而实现在硬件设备上运行GeoQuiz应用。

2.5.1 连接设备

首先，将设备连接到系统上。如果是在Mac系统上开发，系统应该会立即识别出所用设备。如果是Windows系统，则可能需要安装adb（Android Debug Bridger）驱动。如果Windows系统自身无法找到adb驱动，请到设备的制造商网站上去下载一个。

可打开Devices视图来确认设备是否得到了识别。单击Eclipse工作区右上角的DDMS按钮打开DDMS透视图，是打开Devices视图的最快方式。Devices视图应该出现在工作区的左手边。AVD以及硬件设备应该已经列在了Devices视图里。

若想回到代码编辑区以及其他Eclipse视图，请单击工作区右上角的Java按钮。

如果遇到设备无法识别的问题，首先尝试重置adb。在Devices视图里，单击该视图右上方向

下的箭头以显示一个菜单。选择底部的Reset adb菜单选项，稍等片刻，设备可能就会出现在列表中。

如果重置adb不起作用，请访问Android开发网站http://developer.android.com/tools/device.html寻求帮助信息。也可访问本书论坛http://forums.bignerdranch.com寻求帮助。

2.5.2　配置设备用于应用开发

要在设备上运行应用，首先应设置设备允许其运行非Google Play商店应用：

- Android 4.1或更早版本的设备，选择“设定 → 应用项”，找到并勾选“未知来源”选项。
- Android 4.2版本的设备，选择“设定 → 安全”项，找到并勾选“未知来源”选项。

其次，还需启用设备的USB调试模式。

- Android 4.0版本以前的设备，选择“设定 → 应用项→ 开发”项，找到并勾选“USB调试”选项。
- Android 4.0或4.1版本的设备，选择“设定 → 开发”项，找到并勾选“USB调试”选项。
- Android 4.2版本的设备，开发选项默认不可见。先选择“设定 → 关于平板/手机”项，通过点击版本号（BuildNumber）7次启用它，然后回到“设定”项，选择“开发”项，找到并勾选“USB调试”选项。

从以上操作中我们可以看出，不同版本设备的设置差异较大。如在设置过程中遇到问题，请访问http://developer.android.com/tools/device.html寻找帮助信息。

再次运行GeoQuiz应用，Eclipse会询问是在虚拟设备上还是在硬件设备上运行应用，选择硬件设备并继续。GeoQuiz应用应该已经在设备上开始运行了。（如果Eclipse没有提供选择，应用依然在虚拟设备上运行了，请按以上步骤重新检查设备设置，并确保设备与系统已正确连接。）

2.6　添加图标资源

GeoQuiz应用现在已经能正常运行了。假如Next按钮上能够显示向右的图标，用户界面看起来应该会更美。

我们在本书随书代码文件中提供了这样的一个箭头图标（http://www.bignerdranch.com/solutions/AndroidProgramming.zip）。随书代码文件是一个Eclipse项目文件的集合，每章对应一个项目文件。

按以上链接下载文件后，找到并打开02_MVC/GeoQuiz/res目录。在该目录下，再找到drawable-hdpi、drawable-mdpi和drawable-xhdpi三个目录。

三个目录各自的后缀名代表设备的像素密度。

- mdpi：中等像素密度屏幕（约160dpi）。

- hdpi：高像素密度屏幕（约240dpi）。
- xhdpi：超高像素密度屏幕（约320dpi）。

（还有一个low-density–ldpi目录。不过，目前大多数低像素密度的设备基本已停止使用，可以不用理会。）

每个目录下，可看到名为arrow_right.png和arrow_left.png的两个图片文件。这些图片文件都是按照目录名对应的dpi进行定制的。

在正式发布的应用里，为不同dpi的设备提供定制化的图片非常重要。这样可以避免使用同一套图片时，为适应不同设备，图片被拉伸后带来的失真感。项目中的所有图片资源都会随应用安装在设备里，Android操作系统知道如何为不同设备提供最佳匹配。

2.6.1 向项目中添加资源

接下来，需将图片文件添加到GeoQuiz项目资源中去。

在包浏览器中，打开res目录，找到匹配各类像素密度的子目录，如图2-7所示。

图2-7 GeoQuiz应用drawable目录中的箭头图标

然后将已下载文件目录中对应的图片文件复制到项目的对应目录中。

注意，如果采用拖曳方式复制文件，将会得到如图2-8所示的选择提示。此时要选择Copy files。

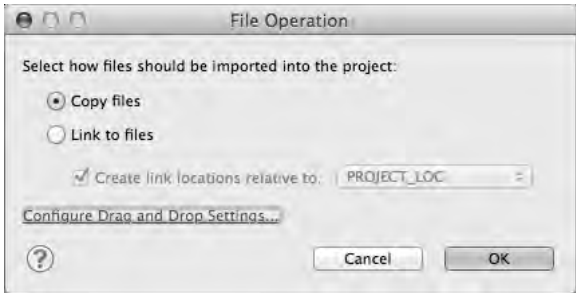

图2-8　复制而非链接

向应用里添加图片就这么简单。任何添加到res/drawable目录中，后缀名为.png、.jpg或者.gif的文件都会被自动赋予资源ID。（注意，文件名必须是小写字母且不能有任何空格符号。）

完成图片资源文件复制后，打开gen/R.java文件，在R.drawable内部类中查看新的图片资源ID，可以看到系统仅新生成了`R.drawable.arrow_left`和`R.drawable.arrow_right`两个资源ID。

这些资源ID没有按照屏幕密度匹配。因此不需要在运行的时候确定设备的屏幕像素密度，只需在代码中引用这些资源ID就可以了。应用运行时，操作系统知道如何在特定的设备上显示匹配的图片。

从第3章起，我们将学习到更多有关Android资源系统的运作方式等相关知识。而现在，Next按钮上能够显示右箭头图标就可以了。

2.6.2　在XML文件中引用资源

在代码中，可以使用资源ID引用资源。但如果想在布局定义中配置Next按钮显示箭头图标的话，又要如何在布局XML文件中引用资源呢?

语法只是稍有不同。打开activity_quiz.xml文件，为`Button`组件新增两个属性，如代码清单2-12所示。

代码清单2-12　为Next按钮增加图标（activity_quiz.xml）

```
<LinearLayout
  ... >

  ...

  <LinearLayout
    ... >

    ...

  </LinearLayout>

  <Button
```

```
        android:id="@+id/next_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/next_question_button"
        android:drawableRight="@drawable/arrow_right"
        android:drawablePadding="4dp"
        />

</LinearLayout>
```

在XML资源文件中，通过资源类型和资源名称，可引用其他资源。以@string/开头的定义是引用字符串资源。以@drawable/开头的定义是引用drawable资源。

从第3章起，我们会学到更多资源命名以及res目录结构中其他资源的使用等相关知识。

运行QeoQuiz应用。新按钮很漂亮吧？测试一下，确认它仍然正常工作。

然而，GeoQuiz应用有个bug。GeoQuiz应用运行时，单击Next按钮显示下一道题。然后旋转设备。（如果是在模拟器上运行的应用，请按组合键Control+F12/Ctrl+F12实现旋转。）

我们发现，设备旋转后应用又显示了第一道测试题。怎么回事？如何修正？

要解决此类问题，需了解activity生命周期的概念。第3章将会做专题介绍。

2.7 关于挑战练习

本书大部分章末尾都安排有挑战练习，需要你独立完成。有些很简单，就是练习所学知识。有些难度较大，需要较强的解决问题能力。

希望你一定完成这些练习。攻克它们不仅可以巩固所学，建立信心，而且可以很快让自己从被动学习成长为自主开发的Android程序员。

在解答挑战练习的过程中，若一时陷入困境，可休息休息，理理头绪，然后以新的思路重新再来。如果仍然无法解决，可访问本书论坛http://forums.bignerdranch.com，参考其他读者发布的解决方案。当然你也可以自己发布问题和答案与读者们一起交流学习。

为保持当前工作项目的完整性，建议你在Eclipse中先复制当前项目，然后在复制的项目上进行练习。

右键单击包浏览器中的项目，选择Copy选项，然后再右键单击选择Paste选项。Eclipse会提示为新项目命名。输入新项目名称后确认完成项目复制。

2.8 挑战练习一：为 TextView 添加监听器

Next按钮很好，但如果用户单击应用的TextView文字区域（地理知识问题），就可跳转到下一道题，用户体验应该会更好。你来试一试。

提示 TextView也是View的子类，因此就如同Button一样，可为TextView设置View.OnClick-Listener监听器。

2.9　挑战练习二：添加后退按钮

在GeoQuiz应用的用户界面上新增后退按钮，用户单击时，可以显示上一道测试题目。完成后的用户界面应如图2-9所示。

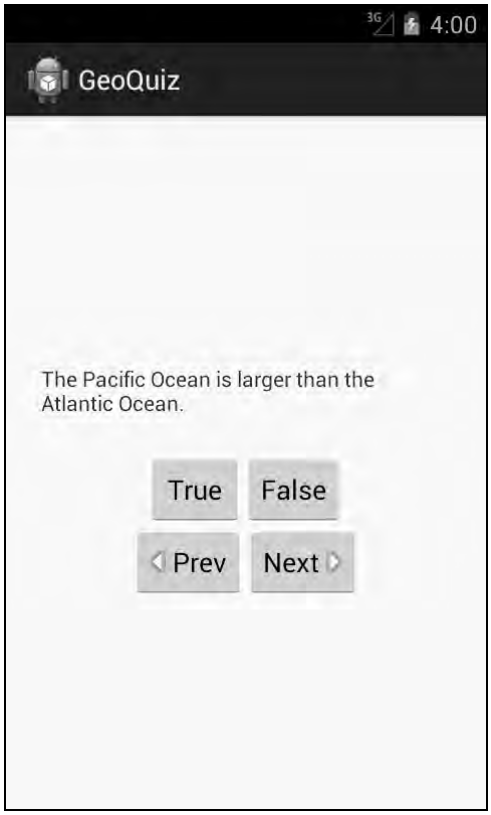

图2-9　添加了后退按钮的用户界面

这是个很棒的练习，需回顾前两章的内容才能完成。

2.10　挑战练习三：从按钮到图标按钮

如果能实现前进与后退按钮上只显示指示图标，用户界面看起来可能会更加简洁美观。只显示图标按钮的用户界面如图2-10所示。

完成此练习，需将用户界面上的普通Button组件替换成ImageButton组件。

ImageButton组件继承ImageView。Button组件则继承Textview。ImageButton和Button与View间的继承关系如图2-11所示。

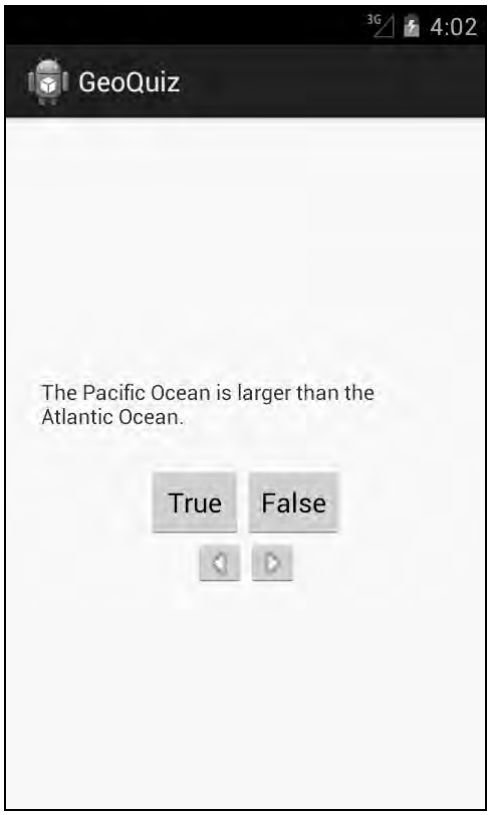

图2-10 只显示图标的按钮

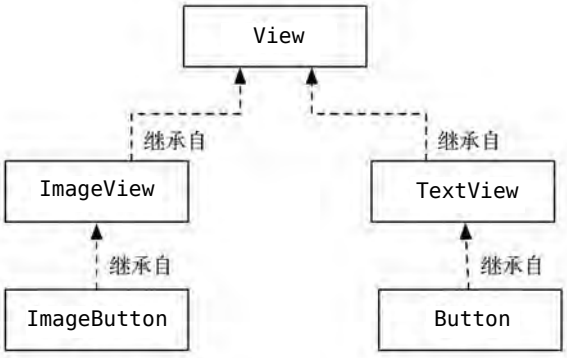

图2-11 ImageButton和Button与View间的继承关系图表

如以下代码所示，将Button组件替换成ImageButton组件，删除Next按钮的text以及drawable属性定义，并添加ImageView属性：

```
<Button ImageButton
  android:id="@+id/next_button"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="@string/next_question_button"
  android:drawableRight="@drawable/arrow_right"
  android:drawablePadding="4dp"
  android:src="@drawable/arrow_right"
  />
```

当然，别忘了调整QuizActivity类代码，使替换后的ImageButton能够正常工作。

将按钮组件替换成ImageButton后，Eclipse会警告说找不到android:contentDescription属性定义。该属性为视力障碍用户提供方便，在为其设置文字属性值后，如果用户设备的可访问性选项作了相应设置，那么当用户点击图形按钮时，设备便会读出属性值的内容。

最后，为每个ImageButton都添加上android:contentDescription属性定义。