Creating Activity User Interfaces with Views

A new Activity starts with a temptingly empty screen onto which you place your User Interface. To set the User Interface, call setContentView, passing in the View instance (typically a layout) to display. Because empty screens aren't particularly inspiring, you will almost always use setContentView to assign an Activity's User Interface when overriding its onCreate handler.

The setContentView method accepts either a layout resource ID (as described in Chapter 3) or a single View instance. This lets you defi ne your User Interface either in code or using the preferred technique of external layout resources.

Using layout resources decouples your presentation layer from the application logic, providing the fl exibility to change the presentation without changing code. This makes it possible to specify different layouts optimized for different hardware confi gurations, even changing them at run time based on hardware changes (such as screen orientation).

The following code snippet shows how to set the User Interface for an Activity using an external layout resource. You can get references to the Views used within a layout with the findViewById method. This example assumes that main.xml exists in the project's res/layout folder.

```
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.main);
TextView myTextView = (TextView)findViewById(R.id.myTextView);
}
```

If you prefer the more *traditional* approach, you can specify the User Interface in code. The following snippet shows how to assign a new TextView as the User Interface:

```
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
TextView myTextView = new TextView(this);
setContentView(myTextView);
myTextView.setText("Hello, Android");
}
```

The setContentView method accepts a single View instance; as a result, you have to group multiple controls to ensure that you can reference a layout using a single View or View Group.