A Closer Look at Android Activities

To create user-interface screens for your applications, you extend the Activity class, using Views to provide user interaction.

Each Activity represents a screen (similar to the concept of a Form in desktop development) that an application can present to its users. The more complicated your application, the more screens you are likely to need.

You'll need to create a new Activity for every screen you want to display. Typically this includes at least a primary interface screen that handles the main UI functionality of your application. This is often supported by secondary Activities for entering information, providing different perspectives on your data, and supporting additional functionality. To move between screens in Android, you start a new Activity (or return from one).

Most Activities are designed to occupy the entire display, but you can create Activities that are semitransparent, floating, or use dialog boxes.

Creating an Activity

To create a new Activity, you extend the Activity class, defi ning the user interface and implementing your functionality. The basic skeleton code for a new Activity is shown below:

```
package com.paad.myapplication;
import android.app.Activity;
import android.os.Bundle;
public class MyActivity extends Activity {
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
}
}
```

The base Activity class presents an empty screen that encapsulates the window display handling functionality. An empty Activity isn't particularly useful, so the fi rst thing you'll want to do is lay out the screen interface using Views and layouts.

Activity UIs are created using Views. Views are the user-interface controls that display data and provide user interaction. Android provides several layout classes, called *View Groups*, that can contain multiple Views to help you design compelling user interfaces.

Chapter 4 examines Views and View Groups in detail, detailing what's available, how to use them, and how to create your own Views and layouts. To assign a user interface to an Activity, call setContentView from the onCreate method of your Activity.

In this fi rst snippet, a simple instance of MyView is used as the Activity's user interface:

```
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
MyView myView = new MyView(this);
setContentView(myView);
}
```

More commonly you'll want to use a more complex UI design. You can create a layout in code using layout View Groups, or you can use the standard Android convention of passing a resource ID for a layout defi ned in an external resource, as shown in the snippet below:

```
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.main);
}
```

In order to use an Activity in your application, you need to register it in the manifest. Add new activity tags within the application node of the manifest; the activity tag includes attributes for metadata such as the label, icon, required permissions, and themes used by the Activity. An Activity without a corresponding activity tag can't be started.

The following XML snippet shows how to add a node for the MyActivity class created in the snippets above:

```
<activity android:label=”@string/app_name”
android:name=”.MyActivity”>
<intent-filter>
<action android:name=”android.intent.action.MAIN” />
<category android:name=”android.intent.category.LAUNCHER” />
</intent-filter>
</activity>
```

Within the activity tag, you can add intent-filter nodes that specify the Intents your Activity will listen for and react to. Each Intent Filter defi nes one or more actions and categories that your Activity supports. Intents and Intent Filters are covered in depth in Chapter 5, but it’s worth noting that to make an Activity available from the main program launcher, it must include an Intent Filter listening for the Main action and the Launcher category, as highlighted in the snippet below:

```
<activity android:label=”@string/app_name”
android:name=”.MyActivity”>
<intent-filter>
<action android:name=”android.intent.action.MAIN” />
<category android:name=”android.intent.category.LAUNCHER” />
</intent-filter>
</activity>
```