Building User Interfaces and Using Controls

Thus far, we have covered the fundamentals of Android but have not touched the user interface (UI). In this chapter, we are going to discuss user interfaces and controls. We will begin by discussing the general philosophy of UI development in Android, then we’ll describe the common UI controls that ship with the Android SDK. We will also discuss layout managers and view adapters. We will conclude by discussing the Hierarchy Viewer tool—a tool used to debug and optimize Android UIs.

UI Development in Android

UI development in Android is fun. It’s fun because the unattractive features in some other platforms are absent from Android. Swing, for example, has to support desktop applications as well as Java applets. The Java Foundation Classes (JFC) contains so much functionality that it’s frustrating to use and difficult to navigate. JavaServer Faces (JSF) is another example. JSF, a common framework used to build web applications, is actually built on top of JavaServer Pages (JSP) and servlets. So you have to know all of the underlying frameworks before you can begin working with JSF.

Fortunately, this type of baggage carried by other platforms does not exist in Android. With Android, we have a simple framework with a limited set of out-of-the-box controls. The available screen area is generally limited. This, combined with the fact that the user usually wants to do one specific action, allows us to easily build a good user interface to deliver a good user experience.

The Android SDK ships with a host of controls that you can use to build user interfaces for your application. Similar to other SDKs, the Android SDK provides text fields, buttons, lists, grids, and so on. In addition, Android provides a collection of controls that are appropriate for mobile devices.

At the heart of the common controls are two classes: android.view.View and android.view.ViewGroup. As the name of the first class suggests, the View class
represents a general-purpose View object. The common controls in Android ultimately extend the View class. ViewGroup is also a view, but contains other views too. ViewGroup is the base class for a list of layout classes. Android, like Swing, uses the concept of layouts to manage how controls are laid out within a container view. Using layouts, as we’ll see, makes it easy for us to control the position and orientation of the controls in our user interfaces.

You can choose from several approaches to build user interfaces in Android. You can construct user interfaces entirely in code. You can also define user interfaces in XML. You can even combine the two—define the user interface in XML and then refer to it, and modify it, in code. To demonstrate this, we are going to build a simple user interface using each of these three approaches.

Before we get started, let’s define some nomenclature. In this book and other Android literature, you will find the terms view, control, widget, container, and layout in discussions regarding UI development. If you are new to Android programming or UI development in general, you might not be familiar with these terms. We’ll briefly describe them before we get started (see Table 4–1).

### Table 4–1. UI Nomenclature

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>View, Widget, Control</td>
<td>Each of these represents a user interface element. Examples include a button, a grid, a list, a window, a dialog box, and so on. The terms “view,” “widget,” and “control” are used interchangeably in this chapter.</td>
</tr>
<tr>
<td>Container</td>
<td>This is a view used to contain other views. For example, a grid can be considered a container because it contains cells, each of which is a view.</td>
</tr>
<tr>
<td>Layout</td>
<td>This is an XML file used to describe a view.</td>
</tr>
</tbody>
</table>

Figure 4–1 shows a screenshot of the application that we are going to build. Next to the screenshot is the layout hierarchy of the controls and containers in the application.

![Image of user interface and layout](image)

**Figure 4–1. The user interface and layout of an activity**

We will refer to this layout hierarchy as we discuss the sample programs. For now, know that the application has one activity. The user interface for the activity is composed of three containers: a container that contains a person’s name, a container that contains the address, and an outer parent container for the child containers.