Working with Dialogs

Android SDK offers extensive support for dialogs. Dialogs that are explicitly supported in Android include the alert, pick list, single choice, multiple choice, progress, time picker, and date picker dialogs. (This list could vary depending on the Android release.) Android also supports custom dialogs for other needs. The primary focus of this chapter is not to cover each of these dialogs but to cover the underlying architecture of Android dialogs. Android 3.0 has added dialogs based on fragments. This aspect of dialogs is covered in the fragments Chapter 29. Fragment based dialogs are expected to gradually replace the traditional dialogs that are covered here. However these dialogs are not yet deprecated and still the norm on phones.

Dialogs in Android are asynchronous, which provides flexibility. However, if you are accustomed to a programming framework where dialogs are primarily synchronous (such as Microsoft Windows, or JavaScript dialogs in web pages), you might find asynchronous dialogs a bit unintuitive.

After giving you the basics of creating and using Android dialogs, we will provide an intuitive abstraction that will make working with asynchronous dialogs easier. We will then use this abstraction to implement a few sample dialogs. We also provide a link to a downloadable project at the end of this chapter in the References section. You can use this download to experiment with the code and the concepts presented in this chapter.

Using Dialogs in Android

If you are coming from an environment where dialogs are synchronous (especially modal dialogs), you need to think differently with Android dialogs. Dialogs in Android are asynchronous. Not only that but they are also managed; that is, they are reused between multiple invocations, perhaps to help improve performance.
Designing an Alert Dialog

We will begin the discussion with alert dialogs. Alert dialogs commonly contain simple messages about validating forms or sometimes (rightly or wrongly) for debugging. Consider the following debug example that you often find in HTML pages:

```java
if (validate(field1) == false)
{
    //indicate that formatting is not valid through an alert dialog
    showAlert("What you have entered in field1 doesn't match required format");
    //set focus to the field
    //...and continue
}
```

You would likely program this dialog in JavaScript through the `alert` JavaScript function, which displays a simple synchronous dialog box containing a message and an OK button. After the user clicks the OK button, the flow of the program continues. This dialog is considered modal as well as synchronous because the next line of code will not be executed until the `alert` function returns.

This type of alert dialog proves useful for debugging. But Android offers no such direct function or dialog. Instead, it supports an alert-dialog builder, a general-purpose facility for constructing and working with alert dialogs. So you can build an alert dialog yourself using the `android.app.AlertDialog.Builder` class. You can use this builder class to construct dialogs that allow users to perform the following tasks:

- Read a message and respond with Yes or No.
- Pick an item from a list.
- Pick multiple items from a list.
- View the progress of an application.
- Choose an option from a set of options.
- Respond to a prompt before continuing the program.

We will show you how to build one of these dialogs and invoke that dialog from a menu item. This approach, which applies to any of these dialogs, consists of these steps:

1. Construct a `Builder` object.
2. Set parameters for the display such as the number of buttons, the list of items, and so on.
3. Set the callback methods for the buttons.
4. Tell the `Builder` to build the dialog. The type of dialog that's built depends on what you've set on the `Builder` object.
5. Use `dialog.show()` to show the dialog.