Introducing Adapters

Adapters are bridging classes that bind data to user-interface Views. The adapter is responsible for creating the child views used to represent each item and providing access to the underlying data.

User-interface controls that support Adapter binding must extend the AdapterView abstract class. It's possible to create your own AdapterView-derived controls and create new Adapter classes to bind them.

Introducing Some Android-Supplied Adapters

In many cases, you won't have to create your own Adapter from scratch. Android supplies a set of Adapters that pump data into the native user-interface widgets.

Because Adapters are responsible both for supplying the data and selecting the Views that represent each item, Adaptors can radically modify the appearance and functionality of the controls they're bound to.

The following list highlights two of the most useful and versatile native adapters:

❑ ArrayAdapter The ArrayAdapter is a generic class that binds Adapter Views to an array of objects. By default, the ArrayAdapter binds the toString value of each object to a TextView control defi ned within a layout. Alternative constructors allow you to use more complex layouts, or you can extend the class to use alternatives to Text View (such as populating an ImageView or nested layout) by overriding the getView method.

❑ SimpleCursorAdapter The SimpleCursorAdapter binds Views to cursors returned from Content Provider queries. You specify an XML layout defi nition and then bind the value within each column in the result set, to a View in that layout. The following sections will delve into these Adapter classes in more detail. The examples provided bind data to List Views, although the same logic will work just as well for other AdapterView classes such as Spinners and Galleries.