The previous chapter focused on the form controls contained in Silverlight. In this chapter, you will look at two controls that are made to display lists of data: the ListBox and DataGrid. These controls are typically bound to data through a technique known as data binding, which I’ll explore first.

Data Binding

Through data binding, UI elements (called targets) are “bound” to data from a data source (called the source), as illustrated in Figure 5-1. When the data sources change, the UI elements bound to those data sources update automatically to reflect the changes. The data can come from different types of sources, and the target can be just about any UI element, including standard Silverlight controls.
Data binding simplifies application development. Since changes are reflected automatically, you do not need to manually update the UI elements. Also, by using data binding, you are able to separate the UI from the data in your application, which allows for a cleaner UI and easier maintenance.

The Binding Class

Data binding in Silverlight is accomplished by using the Binding class. The Binding class has two components—the source and target—and a property that defines the way the two are bound, called the binding mode. The source is the data that is to be bound, the target is a property of the control that the data is to be bound to, and the mode defines how the data is passed between the source and the target (one-way, one-time, or two-way). You’ll see how this works in the upcoming exercise.

To define the binding of a control’s property, you use XAML markup extensions, such as {Binding path}. For example, to bind the Text property of a TextBox to a data source’s FirstName element, you would use the following XAML:

```xml
<TextBox Text="{Binding FirstName }" />
```

Try It Out: Simple Data Binding in Silverlight

To help explain data binding in Silverlight, let’s build a very simple application. The application will include a Book object that contains two properties: Title and ISBN. These properties will be bound to two TextBox controls. Figure 5-2 shows the end result of the example.