14

How to use data sources with databases (part 1)

Visual Studio 2005 makes it easier than ever to generate Windows forms that work with data that's available from a data source. In this chapter, you'll learn how to develop forms that work with data that's stored in a database. However, you can use similar techniques to work with data from web services or from business objects.

How to create a data source

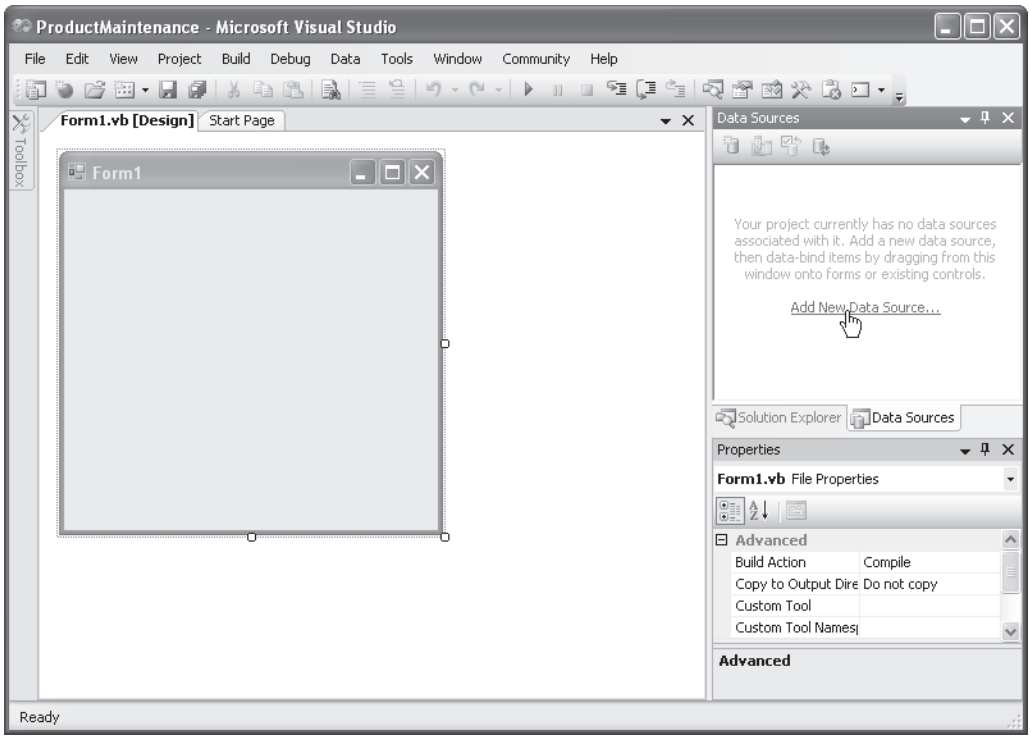
Before you can take advantage of Visual Studio 2005's new features for working with data, you must create a *data source* for the application. As its name implies, a data source specifies the source of the data for an application. Since most applications get their data from a database, the next five figures show how to create a data source that gets data from a database.

How to use the Data Sources window

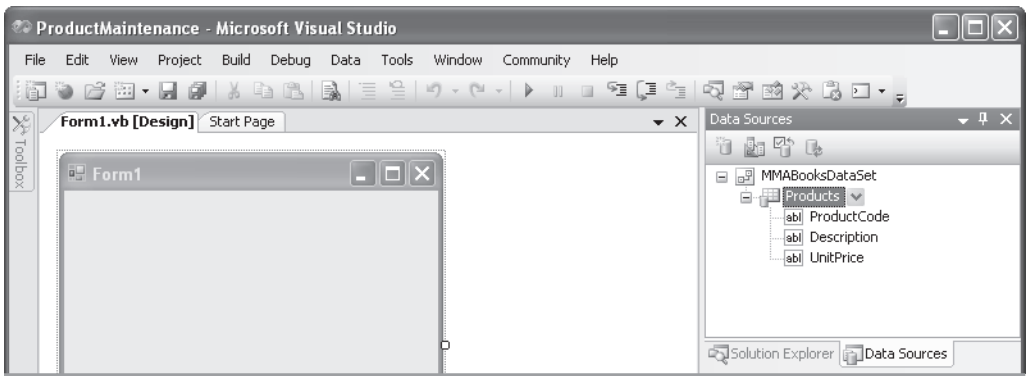
The data sources that are available to a project are listed in the Data Sources window as shown in figure 14-1. Here, the second screen shows a data source for the Products table that's available from the MMABooks database described in the previous chapter. As you can see, this data source includes three columns from the Products table named ProductCode, Description, and UnitPrice.

If no data sources are available to a project, the Data Sources window will display an Add New Data Source link as shown in the first screen. Then, you can click on this link to start the Data Source Configuration Wizard described in figures 14-2 through 14-4. This wizard lets you add a new data source to the project. When you're done, you can drag the data source onto a form to create bound controls as described later in this chapter.

An empty Data Sources window



A Data Sources window after a data source has been added



Description

- A *data source* shows all the tables and columns in the dataset that are available to your application.
- You can display the Data Sources window by clicking on the Data Sources tab that's usually grouped with the Solution Explorer at the right edge of the Visual Studio window or by selecting the Show Data Sources command from the Data menu.
- To create a data source, you can click the Add New Data Source link. Then, you can drag the data source to a form to create controls that are bound to the data source.

Figure 14-1 How to use the Data Sources window

How to start the Data Source Configuration Wizard

You can use the Data Source Configuration Wizard to create a data source for an application. Figure 14-2 shows the first step of this wizard. Since the steps of this wizard are mostly self-explanatory, this chapter only shows the ones that need some additional explanation. In particular, figure 14-3 shows how to create a new connection to a database, and figure 14-4 shows how to select the database objects that will be included in the dataset for the data source.

If your project doesn't already contain a data source, you can start the Data Source Configuration Wizard by clicking the Add New Data Source link that's in the Data Sources window. However, if your project already contains a data source, the Data Sources window will display a data source and this link won't be available. In that case, you can start the Data Source Configuration Wizard by selecting the Add New Data Source command from the Data menu.

You can also start the Data Source Configuration Wizard by adding a SQL Server or Access database file to the project. You may want to do that if the application is for a single user. That way, the database can easily be distributed with the application as described in chapter 25.

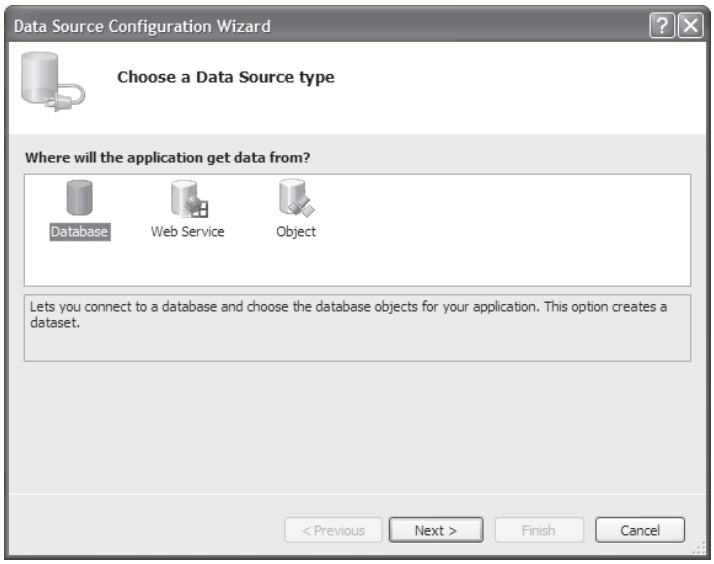
If you add a database file to your project, you should know that by default, that file is copied to the output directory for the project every time the project is built. (The output directory is the directory where the executable file for the application is stored.) Then, when you run the application, the application works with the copy of the database file in the output directory. That means that any changes that you make to the database aren't applied to the database file in the project directory. And each time you rebuild the application, the database in the output directory is overwritten by the unchanged database in the project directory so you're back to the original version of the database.

If you want to change the way this works, you can select the database file in the Solution Explorer and change its "Copy to Output Directory" property from "Copy always" to "Copy if newer." Then, the database file in the output directory won't be overwritten unless the database file in the project directory contains more current data.

How to choose a data source type

The first step of the Data Source Configuration Wizard lets you specify the source from which your application will get its data. To work with data from a database as described in this chapter, you select the Database option. However, you can also select the Web Service option to work with data from a web service that's available from the Internet or from an intranet. Or, you can select the Object option to work with data that's stored in business objects. This option lets you take advantage of the objects that are available from the middle layer of an application as described in chapter 17.

The first step of the Data Source Configuration Wizard



How to start the Data Source Configuration Wizard

- Click on the Add New Data Source link that's available from the Data Sources window when a project doesn't contain any data sources.
- Select the Add New Data Source command from Visual Studio's Data menu.
- Add a SQL Server (.mdf) or Access (.mdb) data file to the project using the Project➔Add➔Existing Item command. Then, the wizard will skip to the step shown in figure 14-4 that lets you choose the database objects you want to include.

How to choose a data source type

- To get your data from a database, select the Database option. This option lets you create applications like the ones described in this chapter.
- To get your data from a web service, select the Web Service option. This option lets you browse the web to select a web service that will supply data to your application.
- To get your data from a business object, select the Object option. This option lets you create applications like the ones described in chapter 17.

Description

- Before you start this procedure, you need to install your database server software on your own PC or on a network server, and you need to attach your database to it. For more information, please refer to appendix A.
- When you click the Next button in the first step of the wizard above, the Choose Your Data Connection step is displayed. Then, if you've already established a connection to a database, you can choose that connection. Otherwise, you can click the New Connection button to display the Add Connection dialog box shown in the next figure.

Figure 14-2 How to start the Data Source Configuration Wizard and choose a data source type

How to create a connection to a database

After you choose the data source type, the Data Source Configuration Wizard displays a dialog box that lets you choose the data connection you want to use. From this dialog box, you can select an existing connection (one you've previously defined), or you can click the New Connection button to display the Add Connection dialog box. This dialog box helps you identify the database that you want to access and provides the information you need to access it. That includes specifying the name of the server that contains the database, entering the information that's required to log on to the server, and specifying the name of the database. How you do that, though, varies depending on whether you're running SQL Server Express on your own PC or whether you're using a database server that's running on a network server.

If you're using SQL Server Express on your own PC and you've downloaded and installed it as described in appendix A, you can use the localhost keyword to specify that the database server is running on the same PC as the application. This keyword should be followed by a backslash and the name of the database server: SqlExpress.

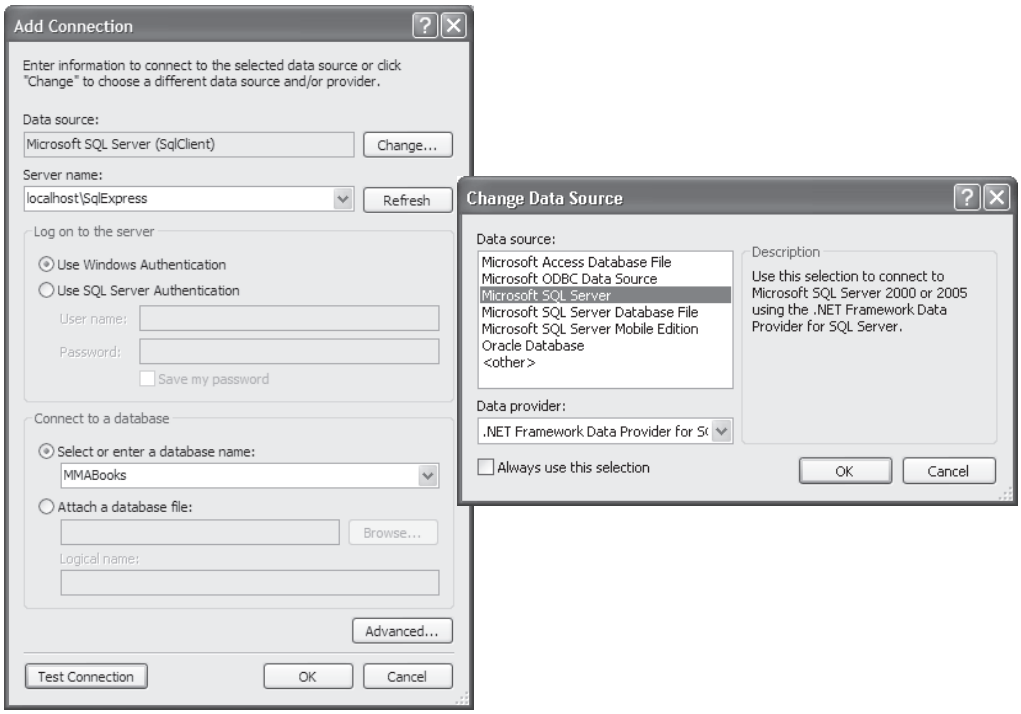
For the logon information, you should select the Use Windows Authentication option. Then, SQL Server Express will use the login name and password that you use to log in to Windows as the name and password for the database server too. As a result, you won't need to provide a separate user name and password in this dialog box.

Last, you enter or select the name of the database that you want to connect to. In this figure, for example, the connection is for the MMABooks database that's used throughout the chapters in this section of the book. When you're done supplying the information for the connection, you can click the Test Connection button to be sure that the connection works.

In contrast, if you need to connect to a database that's running on a database server that's available through a network, you need to get the connection information from the network or database administrator. This information will include the name of the database server, logon information, and the name of the database. Once you establish a connection to the database, you can use that connection for all of the other applications that use that database.

By default, Visual Studio assumes you want to access a SQL Server database as shown here. This works for SQL Server 7, 2000, and 2005 databases including SQL Server Express databases. If you want to access a different type of database, though, you can click the Change button to display the Change Data Source dialog box. Then, you can select the data source and the data provider you want to use to access that data source. If you want to access an Oracle database, for example, you can select the Oracle Database item in the Data Source list. Then, you can choose the data provider for Oracle or the data provider for OLE DB from the Data Provider drop-down list.

The Add Connection and Change Data Source dialog boxes



Description

- By default, a connection uses the SQL Server data provider. If that isn't what you want, you can click the Change button in the Add Connection dialog box to display the Change Data Source dialog box. Then, you can choose the right data source and data provider.
- To be sure that the connection is configured properly, you can click the Test Connection button in the Add Connection dialog box.
- The next dialog box (not shown) asks if you want to save the connection string to the application configuration file (app.config), and we recommend that you do that. Then, if the connection string changes later on, you can change the string in the app.config file rather than in each form that uses the connection string.

Express Edition differences

- The Change Data Source dialog box provides only two options: Microsoft Access Database File and Microsoft SQL Server Database File.
- The Add Connection dialog box is simpler, and it includes a Database File Name text box that you use to specify the database. To do that, you click the Browse button to the right of the text box and use the resulting dialog box to point to the data file for the database.

Figure 14-3 How to create a connection to a database

How to choose database objects for a data source

Figure 14-4 shows how you can use the last step of the Data Source Configuration Wizard to choose the database objects for a data source. This step lets you choose any tables, views, stored procedures, or functions that are available from the database. In some cases, you can just select the table you need from the list of tables that are available from the database. Then, all of the columns in the table are included in the dataset.

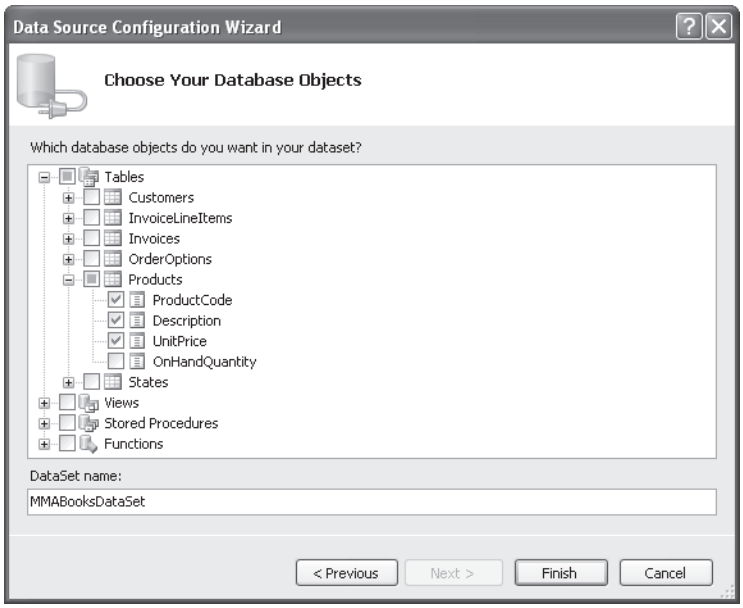
If you want to include selected columns from a table, you can expand the node for the table and select just the columns you want. In this figure, for example, the node for the Products table has been expanded and the three columns that will be used by the Product Maintenance applications in this chapter are selected. Note that although these applications will allow data to be added to the Products table, the OnHandQuantity column can be omitted because it's defined with a default value in the database. So when a new row is added to the database, the database will set this column to its default value.

If you include a column with a default value in a dataset, you need to realize that this value isn't assigned to the column in the dataset, even though the dataset enforces the constraints for that column. For instance, the OnHandQuantity column in the MMABooks database has a default value of zero and doesn't allow nulls. But if you include this column in the dataset, its definition will have a default value of null and won't allow nulls. As a result, an exception will be thrown whenever a new row is added to the dataset with a null value for the OnHandQuantity column.

This means that either the user or the application must provide an acceptable value for the OnHandQuantity column. One way to do that is to provide a way for the user to enter a value for the column. Another way is to use the Dataset Designer to set the DefaultValue property for this column as described in this figure. You'll learn more about working with the Dataset Designer later in this chapter.

In a larger project, you might want to include several tables in the dataset. Then, the dataset will maintain the relationships between those tables whenever that's appropriate. Or, you might want to use views, stored procedures, or functions to work with the data in the database. If you have experience working with views, stored procedures, and functions, you shouldn't have any trouble understanding how this works. Otherwise, you can get another book such as *Murach's SQL for SQL Server* to learn more about working with these types of objects.

The last step of the Data Source Configuration Wizard



Description

- In the last step of the Data Source Configuration Wizard, you can choose the database objects that you want to include in the dataset for your project.
- In this step, you can choose from any tables, views, stored procedures, or functions that are available from the database. In addition, you can expand the node for any table, view, stored procedure, or function and choose just the columns you want to include in the data source.
- You can also enter the name you want to use for the dataset in this dialog box. By default, the name is the name of the database appended with "DataSet".

How to work with columns that have default values

- If a column in a database has a default value, that value isn't included in the column definition in the dataset. Because of that, you may want to omit columns with default values from the dataset unless they're needed by the application. Then, when a row is added to the table, the default value is taken from the database.
- If you include a column that's defined with a default value, you must provide a value for that column whenever a row is added to the dataset. One way to do that is to let the user enter a value. Another way is to display the Dataset Designer as described in figure 14-21, click on the column, and use the Properties window to set the DefaultValue property.

Figure 14-4 How to choose database objects for a data source

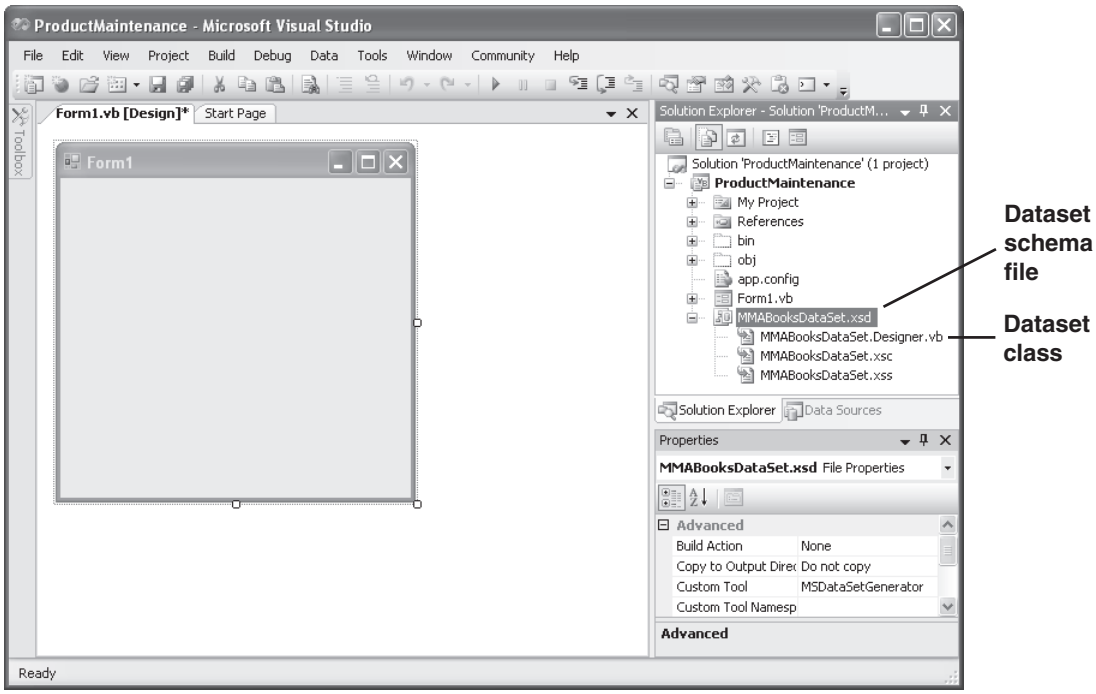
The schema file created by the Data Source Configuration Wizard

After you complete the Data Source Configuration Wizard, the new data source is displayed in the Data Sources window you saw in figure 14-1. In addition to this data source, Visual Studio generates a file that contains the *schema* for the DataSet class. This file defines the structure of the dataset, including the tables it contains, the columns that are included in each table, the data types of each column, and the constraints that are defined for each table. It is listed in the Solution Explorer window and is given the same name you specified for the dataset in the last step of the Data Source Configuration Wizard with a file extension of *xsd*. In figure 14-5, for example, you can see the schema file named MMABooksDataSet.xsd. As you'll learn later in this chapter, you can view a graphic representation of this schema by double-clicking on this file.

Beneath the schema file, the Solution Explorer displays the file that contains the generated code for the DataSet class. In this figure, this code is stored in the MMABooksDataSet.Designer.vb file. When you create bound controls from the data source as shown in this chapter, the code in this class is used to define the DataSet object that the controls are bound to. Although you may want to view this code to see how it works, you shouldn't change it. If you do, the dataset may not work correctly.

By the way, you should know that a dataset that's created from a dataset class like the one shown here is called a *typed dataset*. The code in the dataset class makes it possible for you to refer to the tables, rows, and columns in the typed dataset using the simplified syntax you'll see in this chapter and the next chapter. In contrast, in chapter 16, you'll learn how to create an *untyped dataset*. As you'll see, you create this type of dataset using code.

A project with a dataset defined by a data source



Description

- After you create a data source, it's displayed in the Data Sources window. Then, you can use it to create bound controls as shown in this chapter.
- Visual Studio also generates a file that contains the *schema* for the dataset defined by the data source. This file appears in the Solution Explorer and has a file extension of *xsd*. It defines the structure of the dataset, including the tables it contains, the columns in each table, the data types of each column, and the constraints for each table.
- Subordinate to the schema file is a file that contains the generated code for the dataset class. Visual Studio uses this class to create a dataset object when you add the data source to a form.

Note

- To see the files that are subordinate to the schema file, click the Show All Files button at the top of the Solution Explorer. Then, expand the node for the schema file.

Figure 14-5 The schema file created by the Data Source Configuration Wizard

How to use a data source with a DataGridView control

Once you've created a data source, you can use a DataGridView control to display a grid that can be used to add, update, or delete that data. The DataGridView control is new to .NET 2.0 and has been designed to work with data sources. Although this control provides much of the same functionality as the DataGrid control that was available with previous versions of .NET, it also contains some significant enhancements.

How to generate a DataGridView control from a data source

By default, if you drag a table from the Data Sources window onto a form, Visual Studio adds a DataGridView control to the form and *binds* it to the table as shown in figure 14-6. This creates a DataGridView control that lets you browse all the rows in the table as well as add, update, and delete rows in the table. To provide this functionality, Visual Studio adds a toolbar to the top of the form that provides navigation buttons along with Add, Delete, and Save buttons.

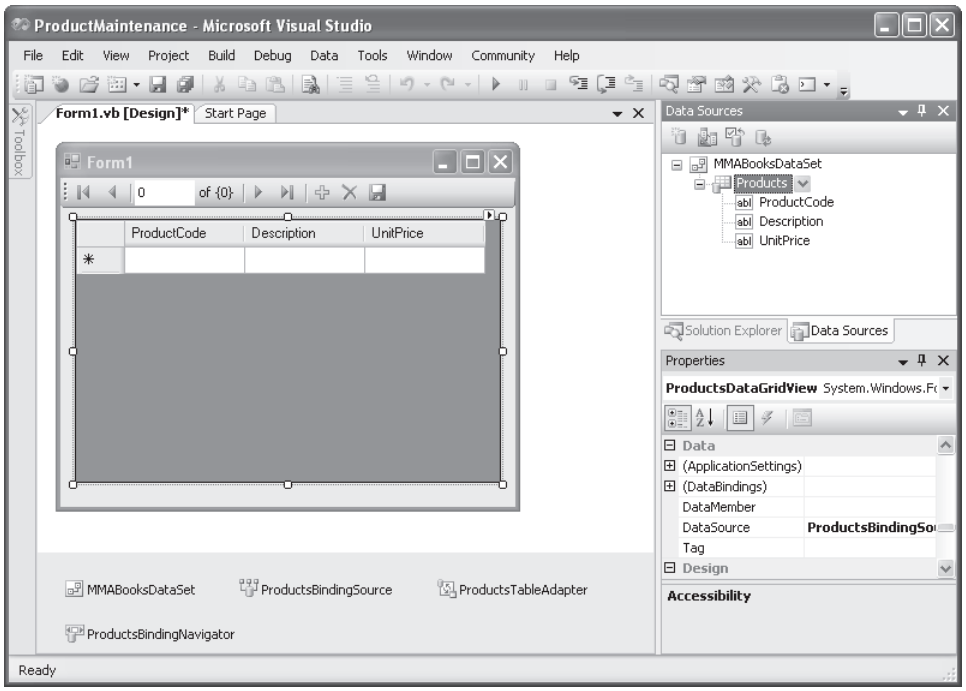
To bind a DataGridView control to a table, Visual Studio uses a technique called *complex data binding*. This just means that the *bound control* is bound to more than one data element. The DataGridView control in this figure, for example, is bound to all the rows and columns in the Products table.

When you generate a DataGridView control from a data source, Visual Studio also adds four additional objects to the Component Designer tray at the bottom of the Form Designer. First, the DataSet object defines the dataset that contains the Products table. Second, the TableAdapter object provides commands that can be used to work with the Products table in the database. Third, the BindingSource object specifies the data source (the Products table) that the controls are bound to, and it provides functionality for working with the data source. Finally, the BindingNavigator defines the toolbar that contains the controls for working with the data source.

Before I go on, I want to point out that the TableAdapter object is similar to the DataAdapter object you learned about in the previous chapter. However, it can only be created by a designer. In addition, it has a built-in connection and, as you'll see later in this chapter, it can contain more than one query.

I also want to mention that, in general, you shouldn't have any trouble figuring out how to use the binding navigator toolbar. However, you may want to know that if you click the Add button to add a new row and then decide you don't want to do that, you can click the Delete button to delete the new row. However, there's no way to cancel out of an edit operation. Because of that, you may want to add a button to the toolbar that provides this function. You can learn how to add buttons to a toolbar in the next chapter.

A form after the Products table has been dragged onto it



The controls and objects that are created when you drag a data source to a form

Control/object	Description
DataGridView control	Displays the data from the data source in a grid.
BindingNavigator control	Defines the toolbar that can be used to navigate, add, update, and delete rows in the DataGridView control.
BindingSource object	Identifies the data source that the controls on the form are bound to and provides functionality for working with the data source.
DataSet object	Provides access to all of the tables, views, stored procedures, and functions that are available to the project.
TableAdapter object	Provides the commands that are used to read and write data to and from the specified table in the database.

Description

- To *bind* a DataGridView control to a table in a dataset, just drag the table from the Data Sources window onto the form. Then, Visual Studio automatically adds a DataGridView control to the form along with the other controls and objects it needs to work properly. Because the DataGridView control is bound to the table, it can be referred to as a *bound control*.
- To bind a DataGridView control to a data table, Visual Studio uses a technique called *complex data binding*. This means that the control is bound to more than one data element, in this case, all the rows and columns in the table.

Figure 14-6 How to generate a DataGridView control from a data source

How to edit the properties of a DataGridView control

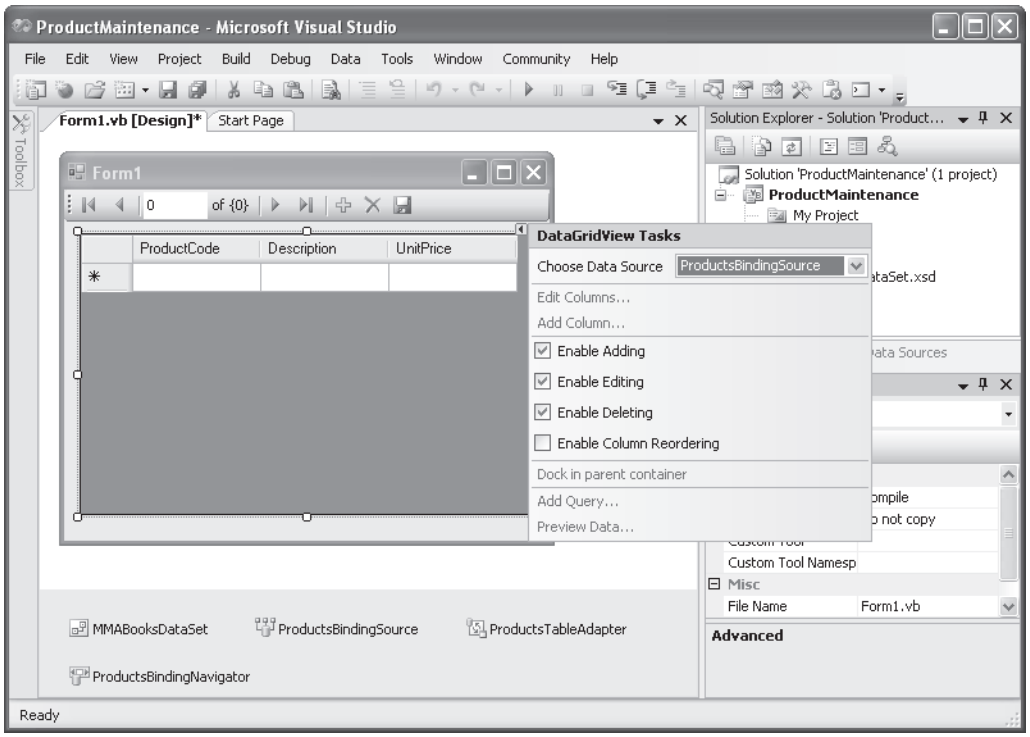
When you generate a DataGridView control from a data source, Visual Studio usually sets the properties of this control and the other objects it creates the way you want them. However, if you want to modify any of these properties, you can do that just as you would for any other type of object. In particular, you'll probably want to edit the properties of the DataGridView control to change its appearance and function.

To change the most common properties of a DataGridView control, you can use its smart tag menu as shown in figure 14-7. From this menu, you can create a read-only data grid by removing the check marks from the Enable Adding, Enable Editing, and Enable Deleting check boxes. Or, you can let a user reorder the columns by checking the Enable Column Reordering check box.

In addition to editing the properties for the grid, you may want to edit the properties for the columns of the grid. For example, you may want to apply currency formatting to a column, or you may want to change the column headings. To do that, you can select the Edit Columns command to display the Edit Columns dialog box shown in the next figure.

When you run an application that uses a DataGridView control, you can sort the rows in a column by clicking in the header at the top of the column. The first time you do this, the rows are sorted in ascending sequence by the values in the column; the next time, in descending sequence. Similarly, you can drag the column separators to change the widths of the columns. Last, if the Enable Column Reordering option is checked, you can reorder the columns by dragging them. These features let the user customize the presentation of the data.

The smart tag menu for a DataGridView control



Description

- You can use the smart tag menu of a DataGridView control to edit its most commonly used properties.
- To edit the columns, select the Edit Columns command to display the Edit Columns dialog box. Then, you can edit the columns as described in the next figure.
- To prevent a user from adding, updating, or deleting data that's displayed in the DataGridView control, uncheck the Enable Adding, Enable Editing, or Enable Deleting check boxes.
- To allow a user to reorder the columns in a DataGridView control by dragging them, check the Enable Column Reordering check box.
- You can edit other properties of a DataGridView control by using the Properties window for the control.

Figure 14-7 How to edit the properties of a DataGridView control

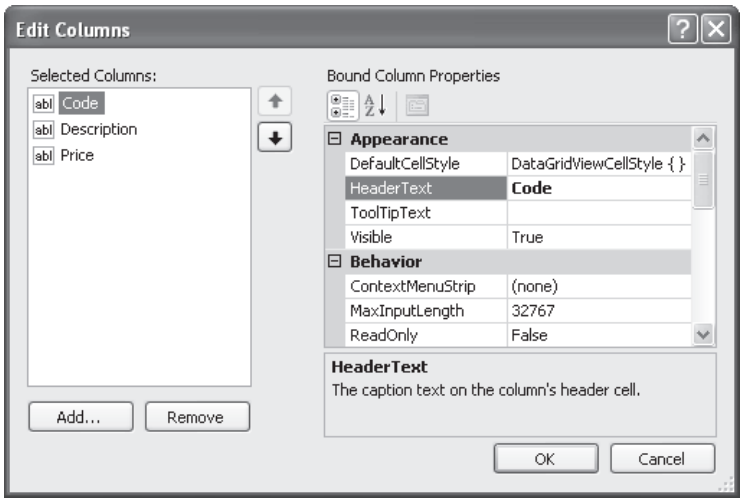
How to edit the columns of a DataGridView control

Figure 14-8 shows how to edit the columns of a DataGridView control using the Edit Columns dialog box. From this dialog box, you can remove columns from the grid by selecting the column and clicking the Remove button. You can also change the order of the columns by selecting the column you want to move and clicking the up or down arrow to the right of the list of columns.

Finally, you can use the Add button in this dialog box to add a column to the grid. You might need to do that if you delete a column and then decide you want to include it. You can also use the dialog box that's displayed when you click the Add button to add unbound columns to the grid. You'll learn how to do that in the next chapter.

Once you've got the right columns displayed in the correct order, you can edit the properties for a column by selecting the column to display its properties in the Bound Column Properties window. When you see the Product Maintenance form in the next figure, for example, you'll see that I changed the HeaderText property for the ProductCode and UnitPrice columns to provide shorter names for the column headers. In addition, I changed the Width property of each column as appropriate. Finally, I used the DefaultCellStyle property to apply currency formatting to the UnitPrice column.

The dialog box for editing the columns of a DataGridView control



Common properties of a column

Property	Description
HeaderText	The text that's displayed in the column header.
Width	The number of pixels that are used for the width of the column.
DefaultCellStyle	The style that's applied to the cell. You can use dialog boxes to set style elements such as color, format, and alignment.

Description

- You can use the Edit Columns dialog box to control which columns are displayed in the grid and to edit the properties of those columns. To display this dialog box, choose the Edit Columns command from the smart tag menu for the control
- To remove columns from the grid, select the column and click the Remove button.
- To add a column to the grid, click the Add button and then complete the dialog box that's displayed. This dialog box lets you add both bound and unbound columns. See chapter 15 for more information on adding unbound columns.
- To change the order of the columns, select the column you want to move and click the up or down arrow to the right of the list of columns.
- To edit the properties for a column, select the column and use the Bound Column Properties window to edit the properties.

Figure 14-8 How to edit the columns of a DataGridView control

A Product Maintenance application that uses a DataGridView control

At this point, the DataGridView control and binding navigator toolbar provide all the functionality needed for an application that can be used to maintain the data in the Products table. Figure 14-9 shows how this application appears to the user at runtime. It also presents the code that Visual Studio generates when you create this application, which includes everything that's necessary to make it work. As a result, you can create an application like this one without having to write a single line of code. If you've ever had to manually write an application that provides similar functionality, you can appreciate how much work this saves you.

When this application starts, the first event handler in this figure is executed. This event handler uses the Fill method of the TableAdapter object to load data into the DataSet object. In this example, the data in the Products table of the MMABooks database is loaded into the Products table of the dataset. Then, because the DataGridView control is bound to this table, the data is displayed in this control and the user can use it to modify the data in the table by adding, updating, or deleting rows.

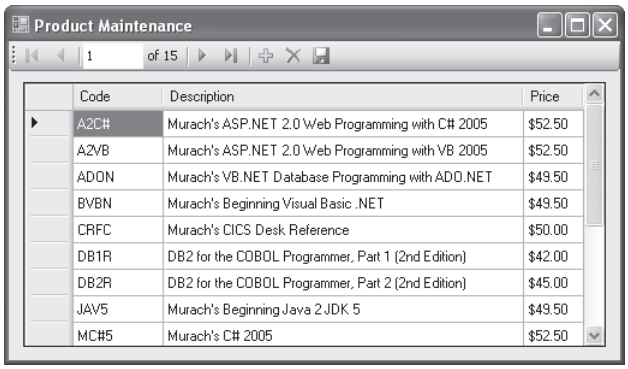
When the user changes the data in the DataGridView control, those changes are saved to the dataset. However, the changes aren't saved to the database until the user clicks the Save button in the toolbar. Then, the second event handler in this figure is executed. This event handler starts by calling the Validate method of the form, which causes the Validating and Validated events of the control that's losing focus to be fired. Although you probably won't use the Validated event, you may use the Validating event to validate a row that's being added or modified. You'll see an example of that later in this chapter.

Next, the EndEdit method of the BindingSource object applies any pending changes to the dataset. That's necessary because when you add or update a row, the new or modified row isn't saved until you move to another row.

Finally, the Update method of the TableAdapter object saves the Products table in the DataSet object to the MMABooks database. When this method is called, it checks each row in the table to determine if it's a new row, a modified row, or a row that should be deleted. Then, it causes the appropriate SQL Insert, Update, and Delete statements to be executed for these rows. As a result, the Update method works efficiently since it only updates the rows that need to be updated.

Now that you understand this code, you should notice that it doesn't provide for any exceptions that may occur during this processing. Because of that, you need to add the appropriate exception handling code for any production applications that you develop so that they won't crash. You'll learn how to do that next.

The user interface for the Product Maintenance application



The code that's generated by Visual Studio

```
Private Sub Form1_Load(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles MyBase.Load
    'TODO: This line of code loads data into the 'MMABooksDataSet.Products'
    'table. You can move, or remove it, as needed.
    Me.ProductsTableAdapter.Fill(Me.MMABooksDataSet.Products)
End Sub

Private Sub ProductsBindingNavigatorSaveItem_Click( _
        ByVal sender As System.Object, ByVal e As System.EventArgs) _
        Handles ProductsBindingNavigatorSaveItem.Click
    Me.Validate()
    Me.ProductsBindingSource.EndEdit()
    Me.ProductsTableAdapter.Update(Me.MMABooksDataSet.Products)
End Sub
```

The syntax of the Fill method

```
TableAdapter.Fill(DataSet.TableName)
```

The syntax of the Update method

```
TableAdapter.Update(DataSet.TableName)
```

Description

- Visual Studio automatically generates the code shown above and places it in the source code file when you drag a data source onto a form. If necessary, you can edit this code.
- The generated code uses the Fill and Update methods of the TableAdapter object that's generated for the table to read data from and write data to the database. It also uses the EndEdit method of the BindingSource object to save any changes that have been made to the current row to the dataset.
- The Validate method causes the Validating and Validated events of the control that is losing the focus to be fired. You can use the Validating event to perform any required data validation for the form.
- Users of a DataGridView control can sort the rows by clicking on a column heading and can size columns by dragging the column separators to the left or right. They can also reorder the columns by dragging them if that option is enabled (see figure 14-7).

Figure 14-9 A Product Maintenance application that uses a DataGridView control