e of rows and col-
ılts. Each element
the author wrote.
:. The outer query
fic author and re-
creates an anony-

›arates the author's

:sult of the nested
:ent author.

ymous type. When
perty by using the

```
author.LastName,

);
ped by author

ne} );

e} );
```

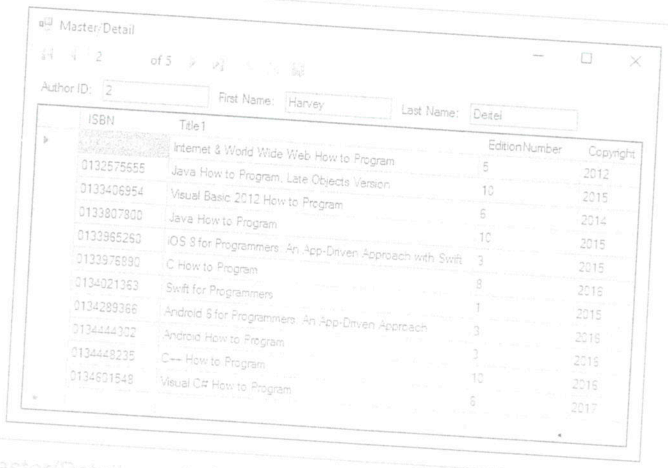
rent author's books
s the Title column

LAB Example : 04

The nested foreach statements (lines 67–77) use the anonymous type's properties to output the hierarchical results. The outer loop displays the author's name and the inner loop displays the titles of all the books written by that author.

✓22.8 Creating a Master/Detail View App

Figure 22.34 shows a so-called master/detail view—one part of the GUI (the master) allows you to select an entry, and another part (the details) displays detailed information about that entry. When the app first loads, it displays the name of the first author in the data source and shows that author's books in the DataGridView. When you use the buttons on the BindingNavigator to change authors, the app displays the details of the books written by the corresponding author—Fig. 22.34 shows the second author's books. This app only reads data from the entity data model, so we disabled the buttons in the BindingNavigator that enable the user to add and delete records. When you run the app, experiment with the BindingNavigator's controls. The DVD-player-like buttons of the BindingNavigator allow you to change the currently displayed row.



Author ID: 2 First Name: Harvey Last Name: Deitel

ISBN	Title1	EditionNumber	Copyright
[illegible]	Internet & World Wide Web How to Program	5	2012
0132575655	Java How to Program, Late Objects Version	10	2015
0133406954	Visual Basic 2012 How to Program	6	2014
0133807800	Java How to Program	10	2015
0133965260	iOS 8 for Programmers: An App-Driven Approach with Swift	3	2015
0133976890	C How to Program	8	2016
0134021363	Swift for Programmers	1	2015
0134289366	Android 6 for Programmers: An App-Driven Approach	3	2016
0134444302	Android How to Program	3	2016
0134448235	C++ How to Program	10	2016
0134601548	Visual C# How to Program	6	2017

Fig. 22.34 | Master/Detail app displaying books for an author in the data source.

22.8.1 Creating the Master/Detail GUI

You've seen that the IDE can automatically generate the BindingSource, BindingNavigator and GUI elements when you drag a data source onto the Form. You'll now use two BindingSources—one for the master list of authors and one for the titles associated with a given author. Both will be generated by the IDE. The completed GUI that you'll now build is shown in Fig. 22.35.

✓*Step 1: Creating the Project*

Follow the instructions in Section 22.5.2 to create and configure a new **Windows Forms Application** project called MasterDetail. Name the source file Details.cs and set the Form's Text property to **Master/Detail**. Be sure to add references to the BooksExamples and EntityFramework libraries, add the connection string to the project's App.Config file and set the MasterDetail project as the startup project.

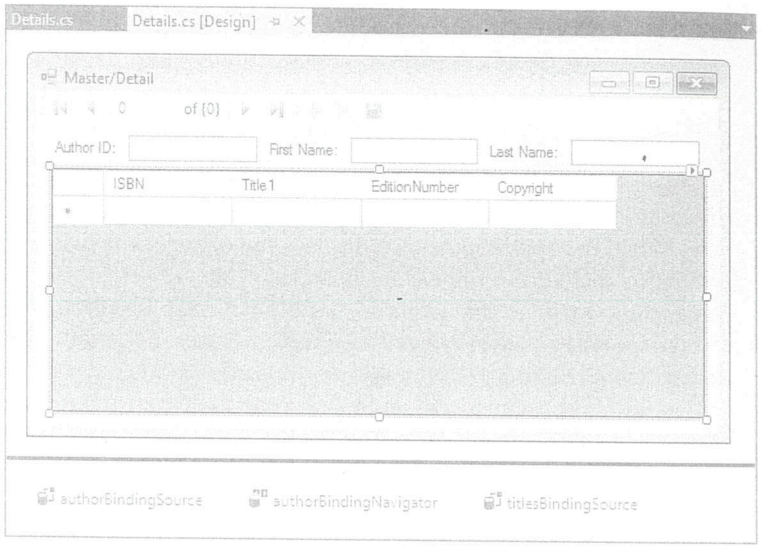


Fig. 22.35 | Finished design of the **Master/Detail** app.

Step 2: Adding a Data Source for the **Authors** *Table*

Follow the steps in Section 22.5.3 to add a data source for the `Authors` table. Although you'll be displaying records from the `Titles` table for each author, you do not need to add a data source for that table. The title information will be obtained from the `Titles` navigation property in the `Author` entity data model class.

Step 3: Creating GUI Elements

Next, you'll use the **Design** view to create the GUI components by dragging-and-dropping items from the **Data Sources** window onto the `Form`. In the earlier sections, you dragged an object from the **Data Sources** window to the `Form` to create a `DataGridView`. The IDE allows you to specify the type of control(s) that it will create when you drag-and-drop an object from the **Data Sources** window onto a `Form`. To do so:

1. Switch to **Design** view for the `Details` class.
2. Click the **Author** node in the **Data Sources** window—it should change to a drop-down list. Open the drop-down by clicking the down arrow and select the **Details** option—this indicates that we'd like to generate `Label`–`TextBox` pairs that represent each column of the `Authors` table.
3. Drag the **Author** node from the **Data Sources** window onto the `Form` in **Design** view. This creates the `authorBindingSource`, the `authorBindingNavigator` and the `Label`–`TextBox` pairs that represent each column in the table. Initially, the controls appear as shown in Fig. 22.36. We rearranged the controls as shown in Fig. 22.35.
4. By default, the `Titles` navigation property is implemented in the entity data model classes as a `HashSet<Title>`. To bind the data to GUI controls properly, you must change this to an `ObservableCollection<Title>`. To do this, expand

Fig. 22.36 | `Details`

the class librar
pand the `Boo`
statement for
`Author` const
`tles` proper
Right click t
compile the

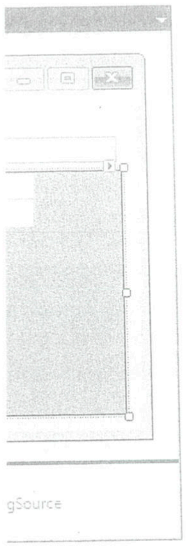
5. Select the `M`
node that's
change to a
ensure that
will be use
author.
6. Drag the `T`
`ingSource`
`ReadOnly`
Titles nod
will auto
bind the

We used the `Data`
also set the `Form`'
size and minimu

22.8.2 Codin

The code to dis
ward. Lines 18-
orders the `Autho`
into memory (l
`BindingSource`

- the `Bi`
the fi
- the `T`
`Last`
- the `c`
`Bind`
title



: Authors table. Although
or, you do not need to add
ned from the Titles navi-

by dragging-and-dropping
rlier sections, you dragged
a DataGridView. The IDE
hen you drag-and-drop an

it should change to a drop-
arrow and select the **Details**
l–TextBox pairs that repre-

w onto the Form in **Design**
horBindingNavigator and
n in the table. Initially, the
ed the controls as shown in

emented in the entity data
n to GUI controls properly,
Title>. To do this, expand

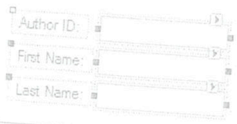


Fig. 22.36 | Details representation of an Author.

the class library project's `BooksModel.edmx` node in the **Solution Explorer**, then expand the `BooksModel.tt` node and open `Author.cs` in the editor. Add a `using` statement for the namespace `System.Collections.ObjectModel`. Then, in the `Author` constructor change `HashSet` to `ObservableCollection`, and in the `Titles` property's declaration, change `ICollection` to `ObservableCollection`. Right click the class library project in the **Solution Explorer** and select **Build** to recompile the class.

5. Select the `MasterDetail` project in the **Solution Explorer**. Next, click the **Titles** node that's nested in the **Author** node in the **Data Sources** window—it should change to a drop-down list. Open the drop-down by clicking the down arrow and ensure that the `DataGridView` option is selected—this is the GUI control that will be used to display the data from the `Titles` table that corresponds to a given author.

6. Drag the **Titles** node onto the `Form` in **Design** view. This creates the `titlesBindingSource` and the `DataGridView`. This control is only for *viewing* data, so set its `ReadOnly` property to `True` using the **Properties** window. Because we dragged the **Titles** node from the **Author** node in the **Data Sources** window, the `DataGridView` will automatically display the books for the currently selected author once we bind the author data to the `authorBindingSource`.

We used the `DataGridView`'s `Anchor` property to anchor it to all four sides of the `Form`. We also set the `Form`'s `Size` and `MinimumSize` properties to `550, 300` to set the `Form`'s initial size and minimum size, respectively.

22.8.2 Coding the Master/Detail App

The code to display an author and the corresponding books (Fig. 22.37) is straightforward. Lines 18–19 create the `DbContext`. The `Form`'s `Load` event handler (lines 22–32) orders the `Author` objects by `LastName` (line 26) and `FirstName` (line 27), then loads them into memory (line 28). Next, line 31 assigns `dbcontext.Authors.Local` to the `authorBindingSource`'s `DataSource` property. At this point:

- the `BindingNavigator` displays the number of `Author` objects and indicates that the first one in the results is selected,
- the `TextBox`es display the currently selected `Author`'s `AuthorID`, `FirstName` and `LastName` property values, and
- the currently selected `Author`'s titles are automatically assigned to the `titlesBindingSource`'s `DataSource`, which causes the `DataGridView` to display those titles.

Now, when you use the `BindingNavigator` to change the selected `Author`, the corresponding titles are displayed in the `DataGridView`.

```
// Fig. 22.37: Details.cs
// Using a DataGridView to display details based on a selection.
using System;
using System.Data.Entity;
using System.Linq;
using System.Windows.Forms;

namespace MasterDetail
{
   public partial class Details : Form
   {
      public Details()
      {
         InitializeComponent();
      }

      // Entity Framework DbContext
      BooksExamples.BooksEntities dbcontext =
         new BooksExamples.BooksEntities();

      // initialize data sources when the Form is loaded
      private void Details_Load(object sender, EventArgs e)
      {
         // load Authors table ordered by LastName then FirstName
         dbcontext.Authors
            .OrderBy(author => author.LastName)
            .ThenBy(author => author.FirstName)
            .Load();

         // specify DataSource for authorBindingSource
         authorBindingSource.DataSource = dbcontext.Authors.Local;
      }
   }
}
```

Fig. 22.37 | Using a `DataGridView` to display details based on a selection.

22.9 Address Book Case Study

Our final example implements a simple `AddressBook` app (with sample outputs in (Figs. 22.38–22.40) that enables users to perform the following tasks on the database `AddressBook.mdf` (which is included in the directory with this chapter's examples):

- Insert new contacts.
- Find contacts whose last names begin with the specified letters.
- Update existing contacts.
- Delete contacts.

We populated the database with six fictional contacts.

Fig. 22.38 | Use the Bi

Fig. 22.39 | Type
contacts whose last na
`BindingNavigator`

Fig. 22.40 |