EXAMPLE-03

Ordering the Books By Title

Lines 40–41 invoke the `OrderBy` extension method on `dbcontext.Titles.Local` to order the `Title` objects by their `Title1` property values. As we mentioned previously, the IDE renamed the `Title` column of the database's `Titles` table as `Title1` in the generated `Title` entity data model class to avoid a naming conflict with the class's name. Recall that `Local` returns an `ObservableCollection<T>` containing the row objects of the specified table—in this case, `Local` returns an `ObservableCollection<Title>`. When you invoke `OrderBy` on an `ObservableCollection<T>`, the method returns an `IOrderedEnumerable<T>`. We assign that object to the `titleBindingSource`'s `DataSource` property. When the `DataSource` property changes, the `DataGridView` iterates through the contents of the `IEnumerable<T>` and displays the data.

Selecting Books with 2016 Copyright

Lines 46–49 filter the titles displayed by using the `Where` extension method with the lambda expression

```
book => book.Copyright == [illegible]
```

as an argument. The `Where` extension method expects as its parameter a `Func` delegate representing a method that receives one parameter and returns a `bool` indicating whether the method's argument matches the specified criteria. The lambda expression used here takes one `Title` object (named `book`) as its parameter and uses it to check whether the given `Title`'s `Copyright` property (a `string` in the database) is equal to 2014. A lambda expression that's used with the `Where` extension method must return a `bool` value. Only `Title` objects for which this lambda expression returns `true` will be selected. We use `OrderBy` to order the results by the `Title1` property so the books are displayed in ascending order by title. The type of the lambda's `book` parameter is *inferred* from `dbcontext.Titles.Local`, which contains `Title` objects. As soon as the `titleBindingSource`'s `DataSource` property changes, the `DataGridView` is updated with the query results.

Selecting Books with Titles That End in "How to Program"

Lines 54–58 filter the titles displayed by using the `Where` extension method with the lambda expression

```
book => book.Title1.EndsWith([illegible])
```

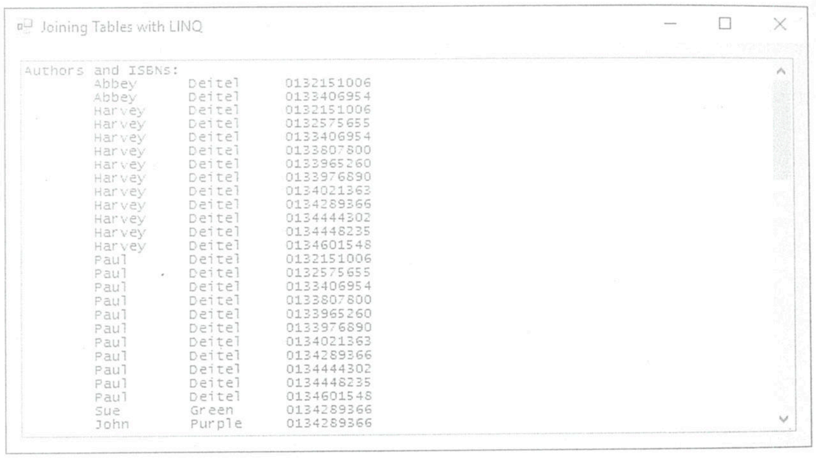
as an argument. This lambda expression takes one `Title` object (named `book`) as its parameter and uses it to check whether the given `Title`'s `Title1` property value ends with `"How to Program"`. The expression `books.Title1` returns the `string` stored in that property, then we use the string class's `EndsWith` method to perform the test. We order the results by the `Title1` property so the books are displayed in ascending order by title.

EXAMPLE-03

22.7 Retrieving Data from Multiple Tables with LINQ

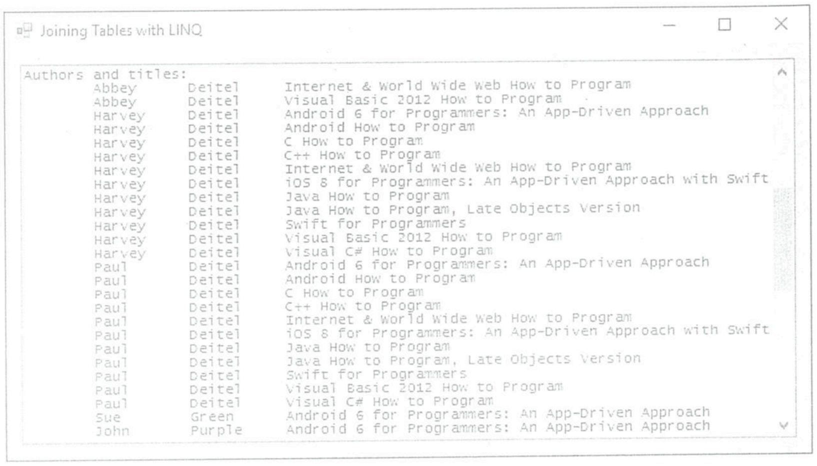
In this section, you'll perform LINQ to Entities queries using the LINQ query syntax that was introduced in Chapter 9. In particular, you'll learn how to obtain query results that combine data from multiple tables (Figs. 22.27–22.29). The **Joining Tables with LINQ** app uses LINQ to Entities to combine and organize data from multiple tables, and shows the results of queries that perform the following tasks:

- Get a list of all the authors and the ISBNs of the books they've authored, sorted by last name, then first name (Fig. 22.27).
- Get a list of all the authors and the titles of the books they've authored, sorted by last name, then first name; for each author sort the titles alphabetically (Fig. 22.28).
- Get a list of all the book titles grouped by author, sorted by last name, then first name; for a given author sort the titles alphabetically (Fig. 22.29).



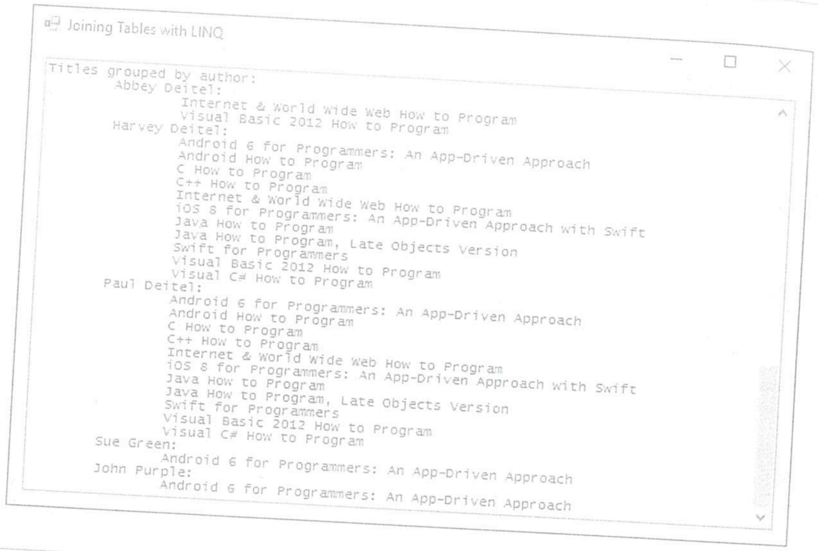
```
Authors and ISBNs:
        Abbey     Deitel    0132151006
        Abbey     Deitel    0133406954
        Harvey    Deitel    0132151006
        Harvey    Deitel    0132575655
        Harvey    Deitel    0133406954
        Harvey    Deitel    0133807800
        Harvey    Deitel    0133965260
        Harvey    Deitel    0133976890
        Harvey    Deitel    0134021363
        Harvey    Deitel    0134289366
        Harvey    Deitel    0134444302
        Harvey    Deitel    0134448235
        Harvey    Deitel    0134601548
        Paul      Deitel    0132151006
        Paul      Deitel    0132575655
        Paul      Deitel    0133406954
        Paul      Deitel    0133807800
        Paul      Deitel    0133965260
        Paul      Deitel    0133976890
        Paul      Deitel    0134021363
        Paul      Deitel    0134289366
        Paul      Deitel    0134444302
        Paul      Deitel    0134448235
        Paul      Deitel    0134601548
        Sue       Green     0134289366
        John      Purple    0134289366
```

Fig. 22.27 | Joining Tables with LINQ app showing the list of authors and the ISBNs of the books they've authored. The authors are sorted by last name, then first name.



```
Authors and titles:
        Abbey     Deitel    Internet & World Wide Web How to Program
        Abbey     Deitel    Visual Basic 2012 How to Program
        Harvey    Deitel    Android 6 for Programmers: An App-Driven Approach
        Harvey    Deitel    Android How to Program
        Harvey    Deitel    C How to Program
        Harvey    Deitel    C++ How to Program
        Harvey    Deitel    Internet & World Wide Web How to Program
        Harvey    Deitel    iOS 8 for Programmers: An App-Driven Approach with Swift
        Harvey    Deitel    Java How to Program
        Harvey    Deitel    Java How to Program, Late Objects Version
        Harvey    Deitel    Swift for Programmers
        Harvey    Deitel    Visual Basic 2012 How to Program
        Harvey    Deitel    Visual C# How to Program
        Paul      Deitel    Android 6 for Programmers: An App-Driven Approach
        Paul      Deitel    Android How to Program
        Paul      Deitel    C How to Program
        Paul      Deitel    C++ How to Program
        Paul      Deitel    Internet & World Wide Web How to Program
        Paul      Deitel    iOS 8 for Programmers: An App-Driven Approach with Swift
        Paul      Deitel    Java How to Program
        Paul      Deitel    Java How to Program, Late Objects Version
        Paul      Deitel    Swift for Programmers
        Paul      Deitel    Visual Basic 2012 How to Program
        Paul      Deitel    Visual C# How to Program
        Sue       Green     Android 6 for Programmers: An App-Driven Approach
        John      Purple    Android 6 for Programmers: An App-Driven Approach
```

Fig. 22.28 | Joining Tables with LINQ app showing the list of authors and the titles of the book's they've authored. The authors are sorted by last name, then first name, and the titles for a given author are sorted alphabetically.



```
Titles grouped by author:
        Abbey Deitel:
                Internet & World Wide Web How to Program
                Visual Basic 2012 How to Program
        Harvey Deitel:
                Android 6 for Programmers: An App-Driven Approach
                Android How to Program
                C How to Program
                C++ How to Program
                Internet & World Wide Web How to Program
                iOS 8 for Programmers: An App-Driven Approach with Swift
                Java How to Program
                Java How to Program, Late Objects Version
                Swift for Programmers
                Visual Basic 2012 How to Program
                Visual C# How to Program
        Paul Deitel:
                Android 6 for Programmers: An App-Driven Approach
                Android How to Program
                C How to Program
                C++ How to Program
                Internet & World Wide Web How to Program
                iOS 8 for Programmers: An App-Driven Approach with Swift
                Java How to Program
                Java How to Program, Late Objects Version
                Swift for Programmers
                Visual Basic 2012 How to Program
                Visual C# How to Program
        Sue Green:
                Android 6 for Programmers: An App-Driven Approach
        John Purple:
                Android 6 for Programmers: An App-Driven Approach
```

Fig. 22.29 | Joining Tables with LINQ app showing the list of titles grouped by author. The authors are sorted by last name, then first name, and the titles for a given author are sorted alphabetically.

GUI for the Joining Tables with LINQ *App*

For this example (Fig. 22.30–Fig. 22.33), perform the steps in Section 22.5.2 to create a new **Windows Forms Application** project named `JoinQueries` in the same solution as the previous examples. Rename the `Form1.cs` source file to `JoiningTableData.cs`. Set the `Form`'s **Text** property to `Joining Tables with LINQ`. 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 `JoinQueries` project as the startup project. We set the following properties for the `outputTextBox`:

- `Font` property: Set to `Lucida Console` to display the output in a fixed-width font.
- `Multiline` property: Set to `True` so that multiple lines of text can be displayed.
- `Anchor` property: Set to `Top, Bottom, Left, Right` so that you can resize the window and the `outputTextBox` will resize accordingly.
- `Scrollbars` property: Set to `Vertical`, so that you can scroll through the output.

Creating the `DbContext`

The code uses the entity data model classes to combine data from the tables in the `Books` database and display the relationships between the authors and books in three different ways. We split the code for class `JoiningTableData` into several figures (Figs. 22.30–22.33) for presentation purposes. As in previous examples, the `DbContext` object (Fig. 22.30, line 19) allows the program to interact with the database.

```
 1  // Fig. 22.30: JoiningTableData.cs
 2  // Using LINQ to perform a join and aggregate data across tables.
 3  using System;
 4  using System.Linq;
 5  using System.Windows.Forms;
 6
 7  namespace JoinQueries
 8  {
 9     public partial class JoiningTableData : Form
10     {
11        public JoiningTableData()
12        {
13           InitializeComponent();
14        }
15
16        private void JoiningTableData_Load(object sender, EventArgs e)
17        {
18           // Entity Framework DbContext
19           var dbcontext = new BooksExamples.BooksEntities();
20
```

Fig. 22.30 | Creating the BooksEntities for querying the Books database.

Combining Author Names with the ISBNs of the Books They've Written
The first query (Fig. 22.31, lines 22–26) *joins* data from two tables and returns a list of author names and the ISBNs representing the books they've written, sorted by `LastName`, then `FirstName`. The query takes advantage of the properties in the entity data model classes that were created based on foreign-key relationships between the database's tables. These properties enable you to easily combine data from related rows in multiple tables.

```
21           // get authors and ISBNs of each book they co-authored
22           var authorsAndISBNs =
23              from author in dbcontext.Authors
24              from book in author.Titles
25              orderby author.LastName, author.FirstName
26              select new {author.FirstName, author.LastName, book.ISBN};
27
28           outputTextBox.AppendText(                  );
29
30           // display authors and ISBNs in tabular format
31           foreach (var element in authorsAndISBNs)
32           {
33              outputTextBox.AppendText(         {element.FirstName,    }   +
34                  {element.LastName,    } {element.ISBN,    } );
35           }
36
```

Fig. 22.31 | Getting a list of authors and the ISBNs of the books they've authored.

The first `from` clause (line 23) gets each `author` from the `Authors` table. The second `from` clause (line 24) uses the `Author` class's `Titles` property to get the ISBNs for the current `author`. The entity data model uses the foreign-key information stored in the data-

base's `AuthorISBN` table t
`from` clauses is a collection
The two `from` clauses int
clauses can access both
orders the results by the a
ymous type containing a
the `ISBN` for one of that

Anonymous Types
Recall from Section 9.3
type with the properties
and `ISBN` (line 26). No
only. Because the type
references to objects of
method in an anonym
the properties of the a
that it receives as an a

Combining Author
The second query (Fi
relationships to get t

```
37              // ge
38              var a
39                 fr
40                 fr
41                 or
42                 se
43
44              outp
45
46              // c
47              fore
48              {
49
50
51              }
52
```

Fig. 22.32 | Gett

The first fr
clause (line 40)
authors for the
stored in the d
objects give us
42) uses the au
author from th
table.

base's AuthorISBN table to get the appropriate ISBNs. The combined result of the two from clauses is a collection of all the authors and the ISBNs of the books they've authored. The two from clauses introduce *two* range variables into the scope of this query—other clauses can access both range variables to combine data from multiple tables. Line 25 orders the results by the author's LastName, then FirstName. Line 26 creates a new anonymous type containing an author's FirstName and LastName from the Authors table and the ISBN for one of that author's books from the Titles table.

Anonymous Types

Recall from Section 9.3.5 that a LINQ query's select clause can create an anonymous type with the properties specified in the initializer list— in this case, FirstName, LastName and ISBN (line 26). Note that all properties of an anonymous type are public and *read-only.* Because the type has no name, you must use *implicitly typed local variables* to store references to objects of anonymous types (e.g., line 31). Also, in addition to the ToString method in an anonymous type, the compiler provides an Equals method, which compares the properties of the anonymous object that calls the method and the anonymous object that it receives as an argument.

Combining Author Names with the Titles of the Books They've Written

The second query (Fig. 22.32, lines 38–42) gives similar output, but uses the foreign-key relationships to get the title of each book that an author wrote.

```
37
38          // get authors and titles of each book they co-authored
39          var authorsAndTitles =
40             from book in dbcontext.Titles
41             from author in book.Authors
42             orderby author.LastName, author.FirstName, book.Title1
43             select new {author.FirstName, author.LastName, book.Title1};
44
45          outputTextBox.AppendText(
46                                                                );
47          // display authors and titles in tabular format
48          foreach (var element in authorsAndTitles)
49          {
50             outputTextBox.AppendText(          {element.FirstName,   }  +
51                {element.LastName,   } {element.Title1} );
52          }
```

Fig. 22.32 | Getting a list of authors and the titles of the books they've authored.

The first from clause (line 39) gets each book from the Titles table. The second from clause (line 40) uses the generated Authors property of the Title class to get only the authors for the current book. The entity data model uses the foreign-key information stored in the database's AuthorISBN table to get the appropriate authors. The author objects give us access to the names of the current book's authors. The select clause (line 42) uses the author and book range variables to get the FirstName and LastName of each author from the Authors table and the title of one of the author's books from the Titles table.

Organizing Book Titles by Author

Most queries return results with data arranged in a relational-style table of rows and columns. The last query (Fig. 22.33, lines 55–62) returns hierarchical results. Each element in the results contains the name of an Author and a list of Titles that the author wrote. The LINQ query does this by using a *nested query* in the select clause. The outer query iterates over the authors in the database. The inner query takes a specific author and retrieves all titles that the author wrote. The select clause (lines 58–62) creates an anonymous type with two properties:

- The property Name (line 58) is initialized with a string that separates the author's first and last names by a space.
- The property Titles (lines 59–62) is initialized with the result of the nested query, which returns the title of each book written by the current author.

In this case, we're providing names for each property in the new anonymous type. When you create an anonymous type, you can specify the name for each property by using the format *name* = *value*.

```
53          // get authors and titles of each book
54          // they co-authored; group by author
55          var titlesByAuthor =
56             from author in dbcontext.Authors
57             orderby author.LastName, author.FirstName
58             select new {Name = author.FirstName + [illegible] + author.LastName,
59                Titles =
60                   from book in author.Titles
61                   orderby book.Title1
62                   select book.Title1};
63
64          outputTextBox.AppendText([illegible] by author:");
65
66          // display titles written by each author, grouped by author
67          foreach (var author in titlesByAuthor)
68          {
69             // display author's name
70             outputTextBox.AppendText([illegible]{author.Name}[illegible]);
71
72             // display titles written by that author
73             foreach (var title in author.Titles)
74             {
75                outputTextBox.AppendText([illegible]{title}[illegible]);
76             }
77          }
78       }
79    }
80 }
```

Fig. 22.33 | Getting a list of titles grouped by authors.

The range variable book in the nested query iterates over the current author's books using the Titles property. The Title1 property of a given book returns the Title column from that row of the Titles table in the database.

The nested foreach s
output the hierarchical re
loop displays the titles of

22.8 Creating a

Figure 22.34 shows a so-
lows you to select an er
about that entry. When
data source and shows t
tons on the BindingNav
written by the correspo
app only reads data fro
ingNavigator that ena
periment with the Bi
BindingNavigator all

Fig. 22.34 | Mast

22.8.1 Creati

You've seen tha
gator and GUI
BindingSource
a given author.
build is shown

Step 1: Creat
Follow the ins
Application pr
Form's Text p
EntityFrame
set the Maste