EXAMPLE-02

42 calls `EndEdit` on the `authorBindingSource`, which forces it to save any pending changes into the `BooksEntities` model in memory. Finally, line 47 calls `SaveChanges` on the `BooksEntities` object `dbcontext` to store any changes into the database. We placed this call in a `try` statement, because the `Authors` table does not allow empty values for the first name and last name—these rules were configured when we originally created the database. When `SaveChanges` is called, any changes stored into the `Authors` table must satisfy the table's rules. If any do not, a `DBEntityValidationException` occurs.

EXAMPLE 02

22.6 Dynamically Binding Query Results

Next we show how to perform several different queries and display the results in a `DataGridView`. 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—simply select each button and set its **Enabled** property to `False` in the **Properties** window. You also could delete these buttons from the `BindingNavigator`. Later, we'll explain why we do not support modifying the database in this example.

The **Display Query Results** app (Figs. 22.23–22.25) allows the user to select a query from the `ComboBox` at the bottom of the window, then displays the results of the query.

Display Query Results

1 of 11

ISBN	Title1	EditionNumber	Copyright
	Android 6 for Programmers: An App-Driven Approach	3	2016
0134444302	Android How to Program	3	2016
0133976890	C How to Program	8	2016
0134448235	C++ How to Program	10	2016
0132151006	Internet & World Wide Web How to Program	5	2012
0133965260	iOS 8 for Programmers: An App-Driven Approach with Swift	3	2015
0133807800	Java How to Program	10	2015
0132575655	Java How to Program, Late Objects Version	10	2015
0134021363	Swift for Programmers	1	2015
0133406954	Visual Basic 2012 How to Program	6	2014
0134601548	Visual C# How to Program	6	2017

Fig. 22.23 | Results of the Display Query Results app's **All titles** query, which shows the contents of the `Titles` table ordered by the book titles.

Display Query Results

1 of 4

ISBN	Title1	EditionNumber	Copyright
	Android 6 for Programmers: An App-Driven Approach	3	2016
0134444302	Android How to Program	3	2016
0133976890	C How to Program	8	2016
0134448235	C++ How to Program	10	2016

Fig. 22.24 | Results of the Display Query Results app's **Titles with 2016 copyright** query.

Display Query Results

1 of 7

ISBN	Title1	EditionNumber	Copyright
	Android How to Program	3	2016
0133976890	C How to Program	8	2016
0134448235	C++ How to Program	10	2016
0132151006	Internet & World Wide Web How to Program	5	2012
0133807800	Java How to Program	10	2015
0133406954	Visual Basic 2012 How to Program	6	2014
0134601548	Visual C# How to Program	6	2017

Fig. 22.25 | Results of the Display Query Results app's Titles ending with "How to Program" query.

22.6.1 Creating the Display Query Results GUI

Perform the following steps to build the **Display Query Results** app's GUI.

Step 1: Creating the Project

Perform the steps in Section 22.5.2 to create a new **Windows Forms Application** project named `DisplayQueryResult` in the same solution as the `DisplayTable` app. Rename the `Form1.cs` source file to `TitleQueries.cs`. Set the `Form`'s **Text** property to `Display Query Results`. 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 `DisplayQueryResult` project as the startup project.

Step 2: Creating a `DataGridView` to Display the `Titles` Table

Follow *Steps 1* and *2* in Section 22.5.3 to create the data source and the `DataGridView`. For this example, select the `Title` class (rather than `Author`) as the data source, and drag the **Title** node from the **Data Sources** window onto the form. Remove the `Authors` column from the `DataGridView`, as it will not be used in this example.

Step 3: Adding a `ComboBox` to the `Form`

In **Design** view, add a `ComboBox` named `queriesComboBox` below the `DataGridView` on the `Form`. Users will select which query to execute from this control. Set the `ComboBox`'s `Dock` property to `Bottom` and the `DataGridView`'s `Dock` property to `Fill`.

Next, you'll add the names of the queries to the `ComboBox`. Open the `ComboBox`'s **String Collection Editor** by right clicking the `ComboBox` and selecting **Edit Items...**. You also can access the **String Collection Editor** from the `ComboBox`'s *smart tag menu*. A smart tag menu provides you with quick access to common properties you might set for a control (such as the `Multiline` property of a `TextBox`), so you can set these properties directly in **Design** view, rather than in the **Properties** window. You can open a control's *smart tag menu* by clicking the small arrowhead

that appears in the control's upper-right corner in **Design** view when the control is selected. In the **String Collection Editor**, add the following three items to `queriesComboBox`—one for each of the queries we'll create:

1. All titles
2. Titles with 2016 copyright
3. Titles ending with "How to Program"

22.6.2 Coding the Display Query Results App

Next you'll create the code for this app (Fig. 22.26).

```
// Fig. 22.26: TitleQueries.cs
// Displaying the result of a user-selected query in a DataGridView.
using System;
using System.Data.Entity;
using System.Linq;
using System.Windows.Forms;

namespace DisplayQueryResult
{
   public partial class TitleQueries : Form
   {
      public TitleQueries()
      {
         InitializeComponent();
      }

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

      // load data from database into DataGridView
      private void TitleQueries_Load(object sender, EventArgs e)
      {
         dbcontext.Titles.Load(); // load Titles table into memory

         // set the ComboBox to show the default query that
         // selects all books from the Titles table
         queriesComboBox.SelectedIndex =  ;
      }

      // loads data into titleBindingSource based on user-selected query
      private void queriesComboBox_SelectedIndexChanged(
         object sender, EventArgs e)
      {
         // set the data displayed according to what is selected
         switch (queriesComboBox.SelectedIndex)
         {
            case  : // all titles
               // use LINQ to order the books by title
               titleBindingSource.DataSource =
                  dbcontext.Titles.Local.OrderBy(book => book.Title1);
               break;
```

Fig. 22.26 | Displaying the result of a user-selected query in a DataGridView. (Part 1 of 2.)