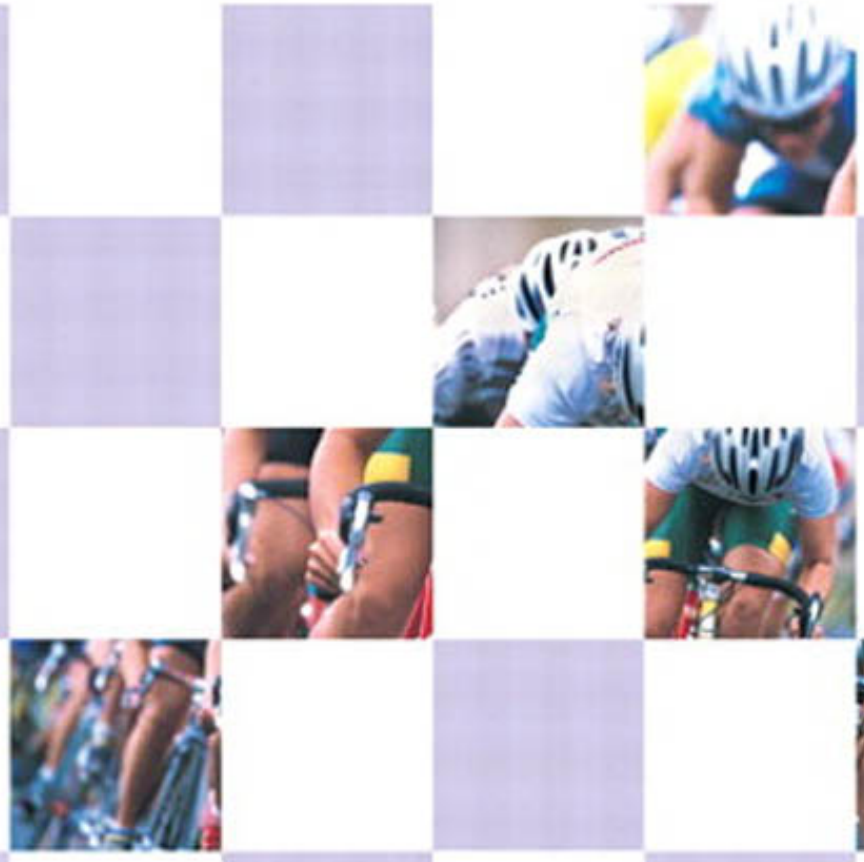
Foreword by Scott Guthrie, General Manager, Microsoft Developer Division

Essential ASP.NET 2.0



Microsoft .net Development Series

Fritz Onion
with Keith Brown

Praise for *Essential ASP.NET 2.0*

"No one knows ASP.NET like Fritz Onion. And no one knows .NET security like Keith Brown. Combine the two and what do you get? The most comprehensive and enlightening book on ASP.NET 2.0 industrywide. I'm sure you'll find the book you're holding was worth every penny."

—Aaron Skonnard, member of technical staff and cofounder, Pluralsight

"***Essential ASP.NET 2.0*** gets under the hood and dismantles the engine before your eyes. Fritz and Keith understand that we as developers need to understand how it works and this book does exactly that. Their explanation of the ASP.NET 2.0 page event sequence is worth the price of the book alone."

—Shawn Wildermuth, Microsoft MVP (C#), "The ADO Guy"

"***Essential ASP.NET 2.0*** is an incredibly useful must-read for any developer. Many books drag you through theory and mindless detail, but this one actually sets up the problems you may encounter with ASP.NET 2.0 and rolls out the alternatives."

—Patrick Hynds, President and Microsoft Regional Director, CriticalSites

"This book is essential for any ASP.NET developer moving from version 1.x to 2.0. Onion and Brown not only cover the new features, but provide a wealth of insight and detail about how to use them effectively."

—Ron Petrusha, author of *Visual Basic 2005: The Complete Reference*

"Drawing on their deep technical knowledge and real-world experience, Fritz and Keith take the reader into some of the less explored and much improved areas of ASP.NET such as diagnostics and state management and performance. Readers will turn to this book over and over again."

—John Timney, Microsoft MVP, Senior Web Services Consultant, British Telecom

Data Source Parameters

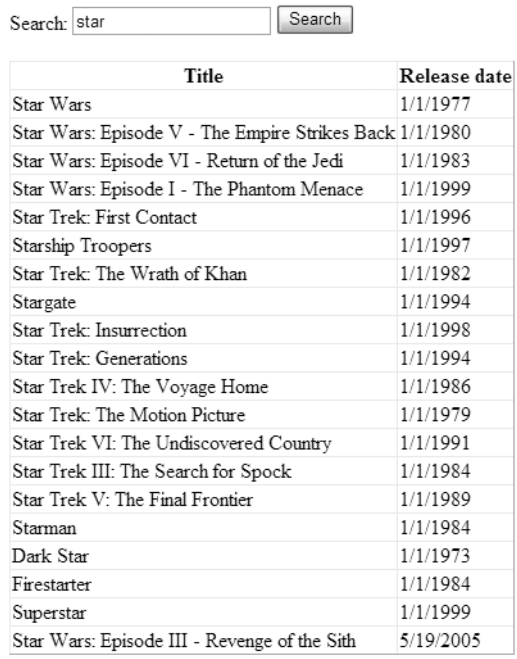
So far in our exploration of declarative data sources, we have issued a single select statement to retrieve data, without any option for altering that query. However, it can be necessary to change a query dynamically, often in response to user input. Declarative data sources support modifications to queries in the form of parameter collections. Each data source supports several collections of parameters; for the SqlDataSource these include UpdateParameters, DeleteParameters, InsertParameters, SelectParameters, and FilterParameters. Each collection of parameters must correspond to parameters in the underlying query or stored procedure associated with the command type.

Parameter values can be specified programmatically, either through two-way data binding of controls (more on this later), or through a predefined set of parameter sources that includes

- The value of a property from another control on the same form
- The value of a cookie
- A form parameter
- A query string parameter
- Data stored in the profile associated with a user
- Data stored in the session associated with a user

Each of these is a convenient way to populate parameters of a data source from alternate data sources in your application. For example, if you wanted to let the user specify a search string to constrain the list of movies displayed in a GridView, you could add a TextBox to the form and a button to issue a post-back, and then add a ControlParameter to the list of FilterParameters for the SqlDataSource control. Point the ControlParameter class to the identifier of the TextBox you added, and specify a filter expression that constrains the query to list only results whose *title* contains the text typed into the text box. Listing 3-16 shows this example in its entirety, and Figure 3-5 shows the sample running with a search for "star" being performed.

Note that the FilterExpression is only available with the SqlDataSource when running in DataSet mode, and that it expects syntax identical to that required by the RowFilter property of the DataView class, which means



Title	Release date
Star Wars	1/1/1977
Star Wars: Episode V - The Empire Strikes Back	1/1/1980
Star Wars: Episode VI - Return of the Jedi	1/1/1983
Star Wars: Episode I - The Phantom Menace	1/1/1999
Star Trek: First Contact	1/1/1996
Starship Troopers	1/1/1997
Star Trek: The Wrath of Khan	1/1/1982
Stargate	1/1/1994
Star Trek: Insurrection	1/1/1998
Star Trek: Generations	1/1/1994
Star Trek IV: The Voyage Home	1/1/1986
Star Trek: The Motion Picture	1/1/1979
Star Trek VI: The Undiscovered Country	1/1/1991
Star Trek III: The Search for Spock	1/1/1984
Star Trek V: The Final Frontier	1/1/1989
Starman	1/1/1984
Dark Star	1/1/1973
Firestarter	1/1/1984
Superstar	1/1/1999
Star Wars: Episode III - Revenge of the Sith	5/19/2005

FIGURE 3-5: Search feature implemented with a filter parameter

that parameters are specified using string placeholder syntax and are populated in order of declaration. Also remember that this feature is implemented completely by the DataSet, so it will not be as efficient as a WHERE clause in a database query for larger result sets. Note that we could have used any of the parameter types for this filter; for example, if the user navigated to this page with a query string to specify a search, we would have used the QueryStringParameter instead of the ControlParameter.

LISTING 3-16: Specifying a filter parameter for a SqlDataSource with input from a TextBox

```
Search: <asp:TextBox ID="_searchTextBox" runat="server" />
<asp:Button ID="_searchButton" runat="server" Text="Search" /><br />
<br />
<asp:GridView ID="_moviesGrid" runat="server"
          AutoGenerateColumns="False" DataKeyNames="movie_id"
          DataSourceID="_moviesDataSource"
          EmptyDataText="There are no data records to display.">
  <Columns>
 <asp:BoundField DataField="title" HeaderText="Title" />
```

```
 <asp:BoundField DataField="release_date" DataFormatString="{0:d}"
                 HeaderText="Release date" HtmlEncode="False" />
  </Columns>
</asp:GridView>

<asp:SqlDataSource ID="_moviesDataSource" runat="server"
       ConnectionString=
           "<%$ ConnectionStrings:moviereviewsConnectionString %>"
SelectCommand="SELECT [movie_id], [title], [release_date] FROM [mov-
ies]"
FilterExpression="title LIKE '%{0}%'">
<FilterParameters>
  <asp:ControlParameter ControlID="_searchTextBox"
                        PropertyName="Text" />
</FilterParameters>
</asp:SqlDataSource>
```

Another interesting application of parameters is to create a master-detail relationship between two data-bound controls. The GridView control supports the concept of selecting a row by enabling a CommandField with the ShowSelectButton attribute set to true. With this enabled, the GridView displays a hyperlink column whose default text is *Select*; when the client clicks this, that row is marked as selected. You can then set up a second data-bound control and data source to display details for the currently selected row by adding a parameter to the Select statement and populating the parameter using a ControlParameter set to the GridView's currently selected row (actually the value of the primary key for that row).

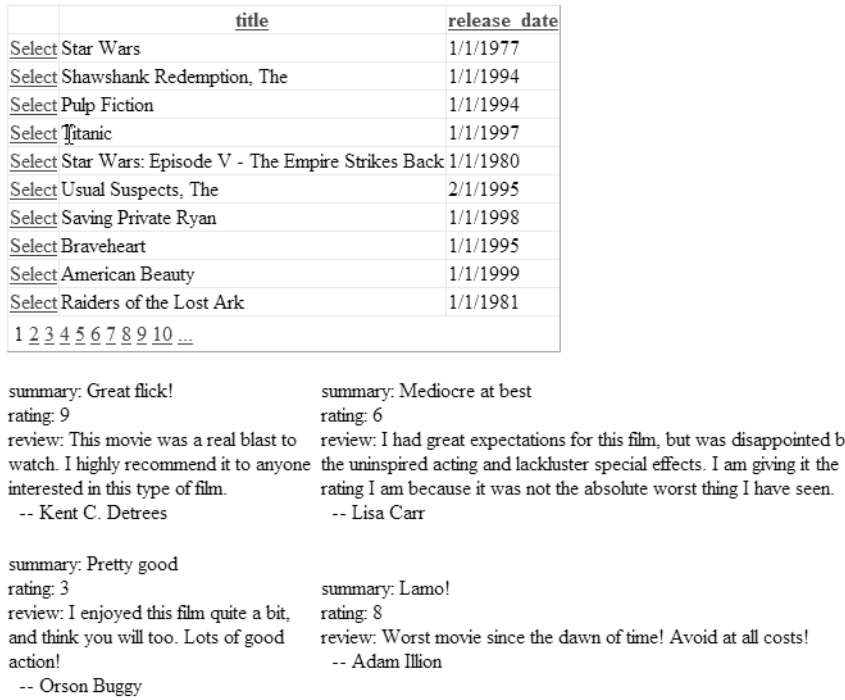
For example, we could create a GridView that listed all of the movies from our database, enable selection, and then create a DataList control to display all of the reviews associated with that movie. Listing 3-17 shows the control declarations to accomplish this, and Figure 3-6 shows the GridView and associated DataList displaying movies and the selected movie's associated reviews, respectively.

LISTING 3-17: A master-detail relationship between a GridView and a DataList

```
<asp:GridView ID="_moviesGrid" runat="server" AllowPaging="True"
        AllowSorting="True"
        AutoGenerateColumns="False" DataKeyNames="movie_id"
        DataSourceID="_moviesDataSource"
        EmptyDataText="There are no data records to display.">
  <Columns>
    <asp:CommandField ShowSelectButton="True" />
```

continues

```
    <asp:BoundField DataField="movie_id" HeaderText="movie_id"
         ReadOnly="True" SortExpression="movie_id" Visible="False" />
    <asp:BoundField DataField="title" HeaderText="title"
                    SortExpression="title" />
    <asp:BoundField DataField="release_date" DataFormatString="{0:d}"
            HeaderText="release_date"
           HtmlEncode="False" SortExpression="release_date" />

  </Columns>
</asp:GridView>

<asp:SqlDataSource ID="_moviesDataSource" runat="server"
     ConnectionString=
          "<%$ ConnectionStrings:moviereviewsConnectionString %>"

     SelectCommand=
          "SELECT [movie_id], [title], [release_date] FROM [movies]"
/><br />

<asp:DataList ID="_reviewsDataList" runat="server" RepeatColumns="2"
              RepeatDirection="Horizontal" DataKeyField="review_id"
              DataSourceID="_reviewsDataSource">
  <ItemTemplate>
    summary: <asp:Label ID="summaryLabel" runat="server"
                    Text='<%# Eval("summary") %>' /><br />
    rating: <asp:Label ID="ratingLabel" runat="server"
                    Text='<%# Eval("rating") %>' /><br />
    review: <asp:Label ID="reviewLabel" runat="server"
                    Text='<%# Eval("review") %>' /><br />
      --
    <asp:Label ID="reviewerLabel" runat="server"
               Text='<%# Eval("reviewer") %>' /><br /><br />
  </ItemTemplate>
</asp:DataList>

<asp:SqlDataSource ID="_reviewsDataSource" runat="server"
     ConnectionString=
       "<%$ ConnectionStrings:moviereviewsConnectionString %>"
     SelectCommand="SELECT review_id, movie_id, summary, rating, review,
reviewer FROM reviews WHERE (movie_id = @movie_id)">
  <SelectParameters>
    <asp:ControlParameter ControlID="_moviesGrid" Name="movie_id"
                          PropertyName="SelectedValue" Type="Int32" />
  </SelectParameters>
</asp:SqlDataSource>
```



	title	release_date
Select	Star Wars	1/1/1977
Select	Shawshank Redemption, The	1/1/1994
Select	Pulp Fiction	1/1/1994
Select	Titanic	1/1/1997
Select	Star Wars: Episode V - The Empire Strikes Back	1/1/1980
Select	Usual Suspects, The	2/1/1995
Select	Saving Private Ryan	1/1/1998
Select	Braveheart	1/1/1995
Select	American Beauty	1/1/1999
Select	Raiders of the Lost Ark	1/1/1981

1 2 3 4 5 6 7 8 9 10 ...

summary: Great flick!
rating: 9
review: This movie was a real blast to watch. I highly recommend it to anyone interested in this type of film.
-- Kent C. Detrees

summary: Mediocre at best
rating: 6
review: I had great expectations for this film, but was disappointed by the uninspired acting and lackluster special effects. I am giving it the rating I am because it was not the absolute worst thing I have seen.
-- Lisa Carr

summary: Pretty good
rating: 3
review: I enjoyed this film quite a bit, and think you will too. Lots of good action!
-- Orson Buggy

summary: Lamo!
rating: 8
review: Worst movie since the dawn of time! Avoid at all costs!
-- Adam Illion

FIGURE 3-6: Master-detail showing reviews associated with the selected film

New Data-Bound Controls

There are several new data-bound controls introduced with this release of ASP.NET 2.0, including most prominently the GridView, DetailsView, and FormView. As you have seen, the GridView is the new grid control, and it is intended to replace the DataGrid (although the DataGrid remains for backward compatibility). It is in general a simpler, more modularly constructed control, and it's easier to use and customize than the DataGrid was. You will notice several differences from the DataGrid when you first start working with the GridView, including the addition of several new column types, including ImageField and CheckBoxField. Most of the features of the GridView are shown throughout the examples in this chapter.

The DetailsView fills the hole of a one-page form display that was missing in ASP.NET 1.1. It supports the ability to show, edit, insert, or delete a single row at a time, and will automatically generate default controls for displaying and updating individual fields based on their type in the

underlying data source. The DefaultMode property of the control lets you specify which of the three modes of operation it should initially display: Edit, Insert, or ReadOnly. You typically will include a CommandField that displays hyperlinks to let the user switch between the three modes as well (although you can also restrict the user to only using one or two modes if you like).

Like most data-bound controls, it also supports template fields so that you can customize the appearance of any field as much as needed. Each field has three core templates that are displayed when the control is in each of the possible display modes: ItemTemplate, EditItemTemplate, and InsertItemTemplate. If you find that you are writing a lot of template fields, however, you might instead consider using the FormView control, which we will discuss next. Listing 3-18 shows an example of the DetailsView control being bound to a SqlDataSource, and Figure 3-7 shows the resulting control rendered in each of its three available states of operation.

Listing 3-18: DetailsView example

```
<asp:DetailsView ID="_moviesDetailsView" runat="server"
     AllowPaging="True" AutoGenerateRows="False"
     DataKeyNames="movie_id" DataSourceID="_moviesDataSource"
     Height="50px" Width="125px">
  <Fields>
    <asp:BoundField DataField="movie_id" HeaderText="movie_id"
                    InsertVisible="False"
                    ReadOnly="True" Visible="False" />
    <asp:BoundField DataField="title" HeaderText="Title" />
    <asp:BoundField DataField="release_date" DataFormatString="{0:d}"
                    HeaderText="Release date" HtmlEncode="False" />
    <asp:CommandField ShowDeleteButton="True" ShowEditButton="True"
                      ShowInsertButton="True" />
  </Fields>
</asp:DetailsView>

<asp:SqlDataSource ID="_moviesDataSource" runat="server"
     ConnectionString=
         "<%$ ConnectionStrings:moviereviewsConnectionString %>"
     DeleteCommand="DELETE FROM movies WHERE movie_id = @movie_id"
     InsertCommand="INSERT INTO movies (title, release_date) VALUES
(@title, @release_date)"
     SelectCommand="SELECT movie_id, title, release_date FROM movies"
     UpdateCommand="UPDATE movies SET title = @title, release_date =
@release_date WHERE movie_id = @movie_id">
  <DeleteParameters>
    <asp:Parameter Name="movie_id" Type="Int32" />
```