Walkthrough: Retrieving, Updating, Inserting, and Deleting Data with the LinqDataSource and DetailsView Controls

Visual Studio 2010

In this walkthrough, you will create a simple database table and a Web page that uses the LinqDataSource control. The Web page enables users to retrieve, update, insert, and delete data from the database table. You will use a DetailsView control to display the data. The LinqDataSource control enables you to perform all these operations without writing Select, Update, Insert, or Delete statements.

You will use the Object Relational Designer to create a class that represents the database table that contains the values. The LinqDataSource control will interact with this generated class to retrieve, update, insert and delete the data.

Prerequisites

To implement the procedures in your own development environment you need:

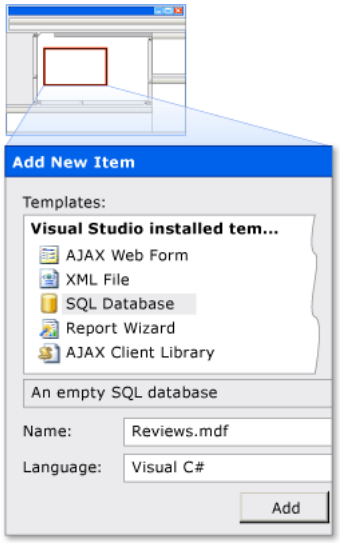
- Visual Studio 2010 or Visual Web Developer Express.
- SQL Server Express installed on your computer. If you have SQL Server installed, you can use that instead, but you must make small adjustments to some of the procedures.
- An ASP.NET Web site.

Creating a Database Table

To perform the steps in this walkthrough, you must have a database table. If you do not already have a table, you can create one by using the following procedure. If you use an existing table, the steps in some of the procedures will not match your database exactly. However, the concepts illustrated in the walkthrough will be the same.

To create a database table

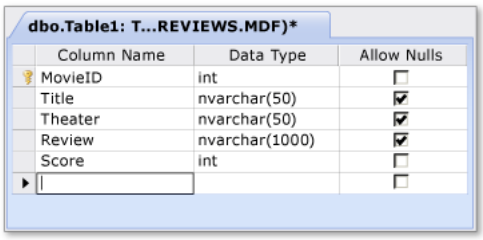
1. If the Web site does not already have an App_Data folder, in **Solution Explorer**, right-click the project, click **Add ASP.NET Folder**, and then click **App_Data**.
2. Right click the App_Data folder and then click **Add New Item**.
3. Under **Installed Templates**, select **SQL Database**, change the file name to Reviews.mdf, and then click **Add**.



4. In Server Explorer, open the Reviews.mdf node and then right-click the **Tables** folder.
5. Click **Add New Table**.
6. Create the following columns in the table:

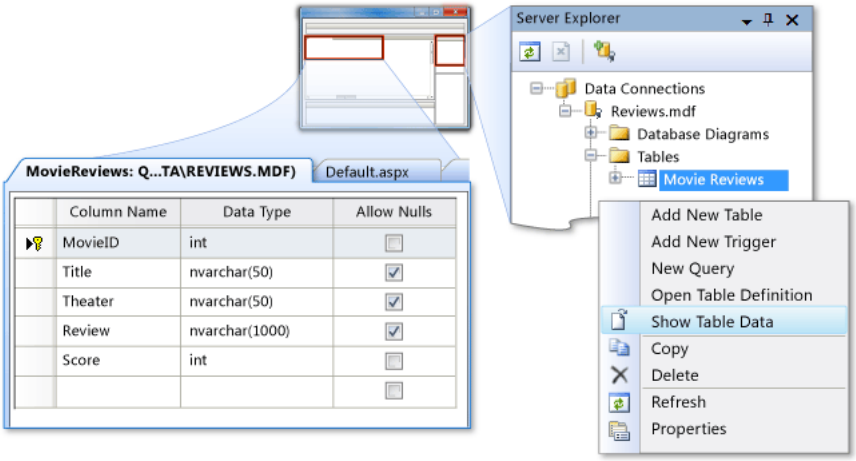
Column Name	Data Type	Properties
MovieID	int	IsIdentity = Yes Not null Primary Key

Title	nvarchar(50)	
Theater	nvarchar(50)	
Review	nvarchar(1000)	
Score	int	Not null

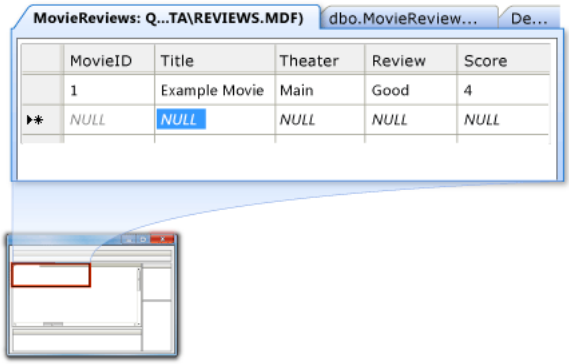


7. Save the table and name it MovieReviews.

8. In **Server Explorer**, right-click the MovieReviews table and click **Show Table Data**.



9. Manually add values for the fields. You do not have to specify a value for MovieID because that value is generated by the database.



Creating Classes to Represent Database Entities

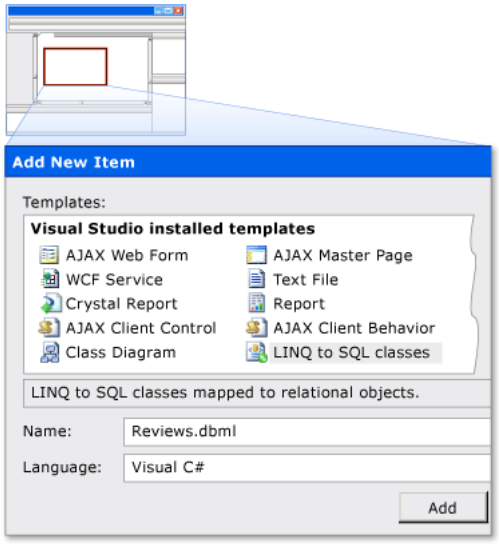
To work with the LinqDataSource control, you work with classes that represent database entities. You can use a tool in Visual Studio 2010 or Visual Web Developer Express to create these classes.

To create a class for the MovieReviews table

1. If the Web site does not already have an App_Code folder, in **Solution Explorer**, right-click the project, click **Add ASP.NET Folder**, and then click **App_Code**.

2. Right-click the App_Code folder and then click **Add New Item**.

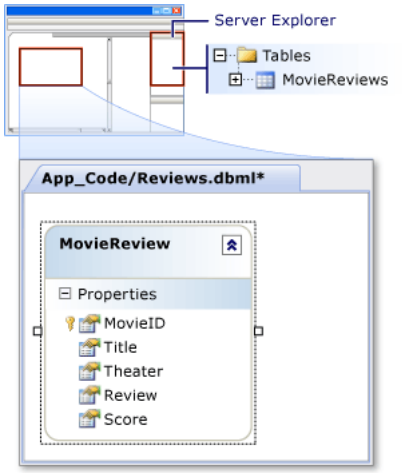
3. Under **Installed templates**, select **Linq to SQL Classes** template, rename the file Reviews.dbml, and then click **Add**.



The **Object Relational Designer** is displayed.

4. In Server Explorer, drag the MovieReviews table into the **Object Relational Designer** window.

   The MovieReviews table and its columns are represented as an entity named `MovieReview` in the designer window.



5. Save the Reviews.dbml file.

6. In Solution Explorer, open the Reviews.designer.cs or Reviews.designer.vb file.

   Notice that it now has classes for `ReviewsDataContext` and `MovieReview`. The `ReviewsDataContext` class represents the database and the `MovieReview` class represents the database table. The parameterless constructor for the `ReviewsDataContext` class reads the connection string from the Web.config file.

7. Open the Web.config file.

   Notice that the connection string has been added in the **connectionStrings** element.

8. Close the class file and the Web.config file.

Creating and Configuring a LinqDataSource Control

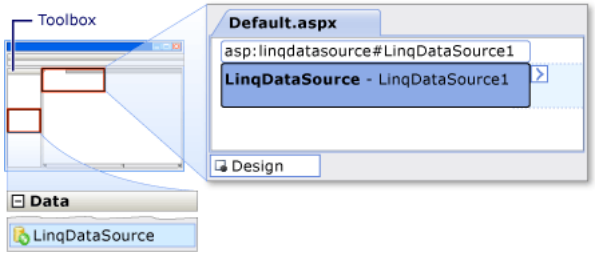
Now that you have a database table and classes that represent database entities, you can use a LinqDataSource control on an ASP.NET Web page to manage data.

To create and configure a LinqDataSource control

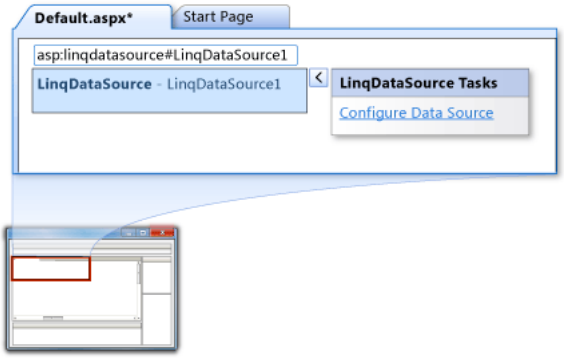
1. In Visual Studio, create a new ASP.NET Web page and switch to **Design** view.

2. From the **Data** tab of the **Toolbox**, drag a LinqDataSource control and drop it inside the **form** element on the Web page.

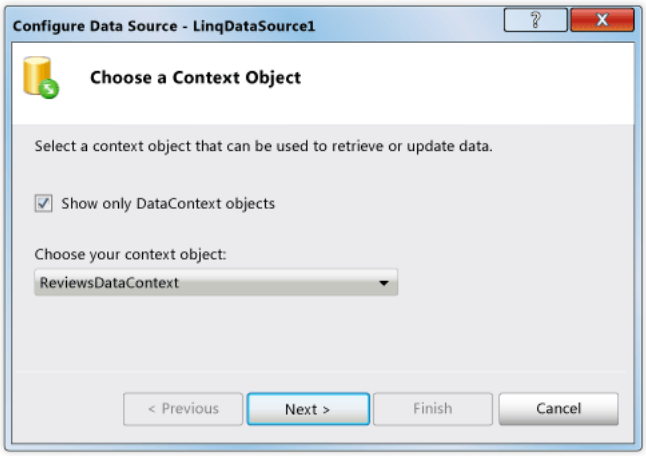
   You can leave the ID property as **LinqDataSource1**.



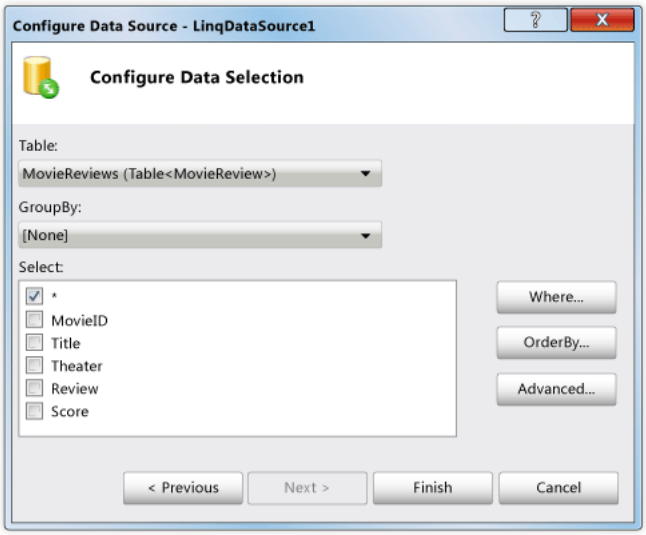
3. In the Smart Tag panel, click **Configure Data Source**.



4. In the **Configure Data Source** dialog box, select **ReviewsDataContext**.



5. Set the TableName property to "MovieReviews".



The following example shows the markup for the LinqDataSource control.

```
<asp:LinqDataSource
  ContextTypeName="ReviewsDataContext"
  TableName="MovieReviews"
```

```
  ID="LinqDataSource1"
  runat="server">
</asp:LinqDataSource>
```

Notice that you did not have to specify any database commands for selecting the data.

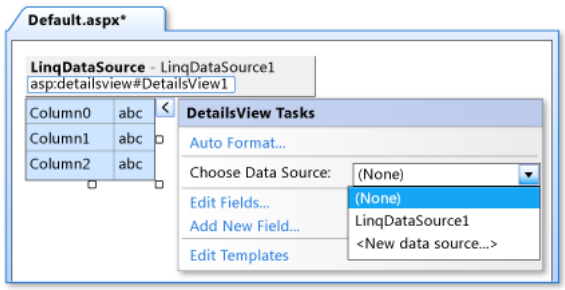
Adding a Control to Display Data

You can now add a DetailsView control and bind it to the LinqDataSource control. The DetailsView control enables users to view data that is managed by the LinqDataSource control.

To connect the data from the LinqDataSource to a DetailsView

1. Switch to **Design** view.
2. From the **Data** tab of the **Toolbox**, drag a DetailsView control onto the page.

   You can leave the ID property as **DetailsView1**.
3. In the Smart Tag panel, set the data source to **LinqDataSource1.**



4. Select the **Enable Paging** option.

   The following example shows the markup for the DetailsView control. Notice that the DataKeyNames property was automatically set to the primary key field in the table, and that a bound field was added for each field in the table.

```
<asp:DetailsView
  DataSourceID="LinqDataSource1"
  DataKeyNames="MovieID"
  AllowPaging="true"
  AutoGenerateRows="False"
  ID="DetailsView1"
  runat="server"
  Height="50px"
  Width="125px">
  <Fields>
    <asp:BoundField
         DataField="MovieID"
         HeaderText="MovieID"
         InsertVisible="False"
         ReadOnly="True"
         SortExpression="MovieID" />
    <asp:BoundField
         DataField="Title"
         HeaderText="Title"
         SortExpression="Title" />
    <asp:BoundField
         DataField="Theater"
         HeaderText="Theater"
         SortExpression="Theater" />
    <asp:BoundField
         DataField="Review"
         HeaderText="Review"
         SortExpression="Review" />
    <asp:BoundField
         DataField="Score"
         HeaderText="Score"
         SortExpression="Score" />
  </Fields>
</asp:DetailsView>
```

5. Save the changes and then press CTRL+F5 to view the page in a browser.

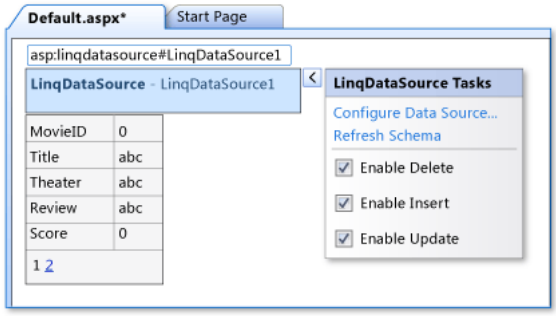
   The DetailsView control displays the values for the current record from the MovieReviews table.

Enabling Users to Update, Insert, and Delete Data

The LinqDataSource control can create the commands for updating, inserting, and deleting data.

To enable update, insert, and delete operations

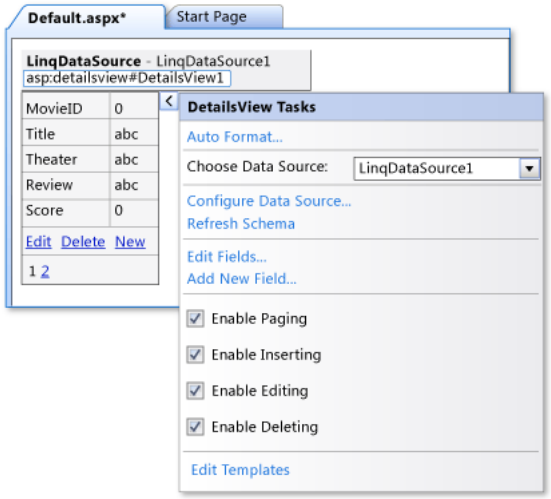
1. In the LinqDataSource control, select the **Enable Delete**, **Enable Insert**, and **Enable Update** options.



The following example shows the declarative markup for the LinqDataSource control.

```
<asp:LinqDataSource
  ContextTypeName="ReviewsDataContext"
  TableName="MovieReviews"
  EnableUpdate="true"
  EnableInsert="true"
  EnableDelete="true"
  ID="LinqDataSource1"
  runat="server">
</asp:LinqDataSource>
```

2. In the DetailsView control, select the **Enable Inserting**, **Enable Editing**, and **Enable Deleting** options.



The following example shows the declarative markup for the DetailsView control.

```
<asp:DetailsView
  DataSourceID="LinqDataSource1"
  DataKeyNames="MovieID"
  AutoGenerateRows="false"
  AllowPaging="true"
  ID="DetailsView1"
  runat="server"
  Height="50px"
  Width="125px">
  <Fields>
    <asp:BoundField
         DataField="MovieID"
         HeaderText="MovieID"
         InsertVisible="False"
         ReadOnly="True"
         SortExpression="MovieID" />
    <asp:BoundField
         DataField="Title"
         HeaderText="Title"
```

```
            SortExpression="Title" />
        <asp:BoundField
            DataField="Theater"
            HeaderText="Theater"
            SortExpression="Theater" />
        <asp:BoundField
            DataField="Review"
            HeaderText="Review"
            SortExpression="Review" />
        <asp:BoundField
            DataField="Score"
            HeaderText="Score"
            SortExpression="Score" />
        <asp:CommandField
            ShowDeleteButton="True"
            ShowEditButton="True"
            ShowInsertButton="True" />
      </Fields>
    </asp:DetailsView>
```

The `MovieID` column is selected from the data source with the other columns. However, it is not displayed in the DetailsView control and the user will not be able to modify its value. The `MovieID` property must be selected to enable the LinqDataSource control to automatically create the commands for updating, inserting, and deleting data.

Notice that you did not have to specify commands for these data operations.

3. Save the changes and press CTRL+F5 to view the page in a browser.

   The DetailsView control displays the fields for the current record from the MovieReviews table. You can update, insert, and delete records by clicking the corresponding buttons on the DetailsView control.

Next Steps

This walkthrough has shown the basic functionality of updating, inserting, and deleting records by using the LinqDataSource control. To learn about additional capabilities of the LinqDataSource control, you can do the following:

- You can filter which data records are returned by specifying a value for the Where property. You can also select only a subset of columns by specifying a value for the Select property. For more information, see Walkthrough: Selecting and Filtering a Subset of Data with the LinqDataSource and GridView Controls.
- To make sure that the data in the database has not changed since the Web page read it, the LinqDataSource controls stores the original values of all the selected data. When the update is posted to the Web server, the LinqDataSource object compares each stored field with the current value in the database. If they match (indicating that the record has not changed), the LinqDataSource object updates or deletes the record. Storing all the original column values can be inefficient. To avoid this issue, you can add a **timestamp** column to your database table. For more information, see Walkthrough: Using a Timestamp with the LinqDataSource Control to Check Data Integrity.

See Also

Concepts

LinqDataSource Web Server Control Overview

Community Additions

Correction on the error 'System.Object System.Web.UI.WebControls.Parameter.GetValue(System.Object, B

*Please can someone help me on this message error **Method not found: 'System.Object System.Web.UI.WebControls.Parameter.GetValue(System.Object, Boolean)'**. I tried using the above method with visual studio 2008 but seems to be given me an error message:*

[MissingMethodException: Method not found: 'System.Object System.Web.UI.WebControls.Parameter.GetValue(System.Object, Boolean)'.]

```
System.Web.UI.WebControls.LinqDataSourceView.MergeDictionaries(Object dataObjectType, ParameterCollection reference, IDictionary source, IDictionary destination, IDic
System.Web.UI.WebControls.LinqDataSourceView.BuildUpdateDataObjects(Object table, IDictionary keys, IDictionary values, IDictionary oldValues) +159
System.Web.UI.WebControls.LinqDataSourceView.ExecuteUpdate(IDictionary keys, IDictionary values, IDictionary oldValues) +94
System.Web.UI.DataSourceView.Update(IDictionary keys, IDictionary values, IDictionary oldValues, DataSourceViewOperationCallback callback) +78
System.Web.UI.WebControls.DetailsView.HandleUpdate(String commandArg, Boolean causesValidation) +1152
System.Web.UI.WebControls.DetailsView.HandleEvent(EventArgs e, Boolean causesValidation, String validationGroup) +440
System.Web.UI.WebControls.DetailsView.OnBubbleEvent(Object source, EventArgs e) +95
System.Web.UI.Control.RaiseBubbleEvent(Object source, EventArgs args) +35
System.Web.UI.WebControls.DetailsViewRow.OnBubbleEvent(Object source, EventArgs e) +109
System.Web.UI.Control.RaiseBubbleEvent(Object source, EventArgs args) +35
System.Web.UI.WebControls.LinkButton.OnCommand(CommandEventArgs e) +115
```

```
   System.Web.UI.WebControls.LinkButton.RaisePostBackEvent(String eventArgument) +163
   System.Web.UI.WebControls.LinkButton.System.Web.UI.IPostBackEventHandler.RaisePostBackEvent(String eventArgument) +7
   System.Web.UI.Page.RaisePostBackEvent(IPostBackEventHandler sourceControl, String eventArgument) +11
   System.Web.UI.Page.RaisePostBackEvent(NameValueCollection postData) +174
   System.Web.UI.Page.ProcessRequestMain(Boolean includeStagesBeforeAsyncPoint, Boolean includeStagesAfterAsyncPoint) +5087
```

Thanks.

CalebSun
6/16/2011