

COMP 3111: Introduction to Software Engineering

Accessing a Database Using ADO.NET Tutorial and Lab Notes

Many web applications require access to a database. The most popular type of database management system (DBMS) is the relational DBMS, such as Microsoft Access, SQL Server, Oracle, DB2 and MySQL. Structured Query Language (SQL) is universally used among relational DBMSs to access a database. ASP.NET enables developers to use SQL to access a database in ASP.NET through ADO.NET¹ and bound controls². ADO.NET provides a way to access almost any database (local or remote) and provides an easy-to-use data connection between a web form and the data you want to access. This data connection can be created by specifying a database to access, placing a control that supports data binding on a web form and setting a few properties of the control. After the data connection is established, data from a database can be displayed on a web form through a bound control. Appendix D provides an overview of the ADO.NET architecture.

VIEWING DATA IN A GRIDVIEW CONTROL

An ASP.NET control that supports data binding is needed to be able to display the data retrieved from a database in a web page. Data binding allows retrieved data to be associated with one or more controls, which will show the data automatically. We will demonstrate how you can set up a data connection to the *Enrollment.mdf* SQL Server database that we created in a previous lab and how to place data into a GridView control on a web page through which you can view the data in the database. The GridView control displays data in a table format and is one of several that support repeated-value binding³, which allows a collection of data (e.g., many rows of a database table) to be displayed on a web page.

Adding a GridView Control and its SqlDataSource Control to a Web Form

1. Start Visual Studio and open the "ASUWebApplication" web application.
2. Add a new web form, name it "StudentRecords.aspx" and set "Site.Master" as its master page.
3. Enter the text "Student Records" on the first line of the MainContent(Custom) ContentPlaceHolder control of the "StudentRecords.aspx" web form, select the text and then select "Heading 2 <h2>" from the dropdown list in the Formatting toolbar.
4. Add a <div> tag from the Toolbox's HTML palette.
5. Place the cursor inside the <div> tag and double-click a GridView control or drag it from the Toolbox's Data palette onto the web form.
6. Change the ID property of the GridView control to "gvStudentRecords" as shown in Figure 1.

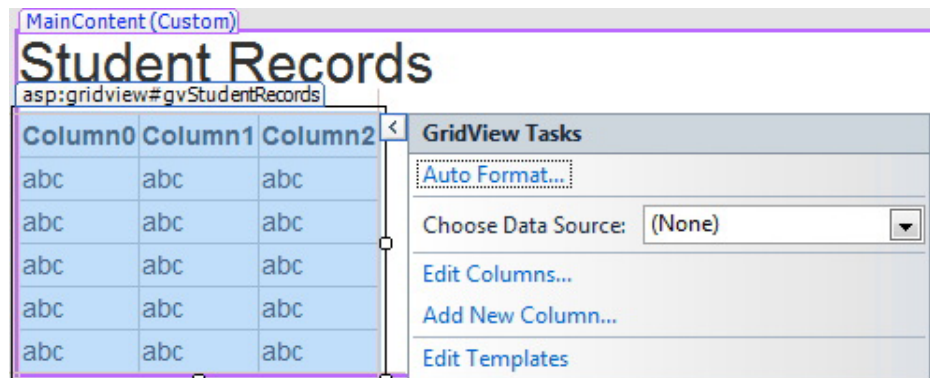


Figure 1: Adding a GridView control.

Connecting to an SQL Server Database

A data connection to a database can be set up in ADO.NET using a data source control, which allows the connection between a bound control and a data source to be declaratively defined. ASP.NET automatically generates the code needed to perform the data access. For accessing a SQL Server database, ASP.NET provides the SqlDataSource control. This control only needs to be configured using the *Configure Data Source* wizard in Visual Studio to set up a connection to an SQL Server database.

A connection to the *Enrollment.mdf* SQL Server database can be set up as follows⁴.

¹ ADO stands for ActiveX Data Object.

² A bound control is one into which you can place data at run time from an external data source such as a file or a database.

³ Other controls that support repeated-value binding are described in Appendix A.

⁴ If necessary the *Enrollment.mdf* database can be downloaded from the Tutorial and Lab Schedule section of the course web page.

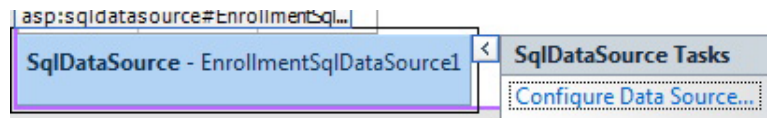
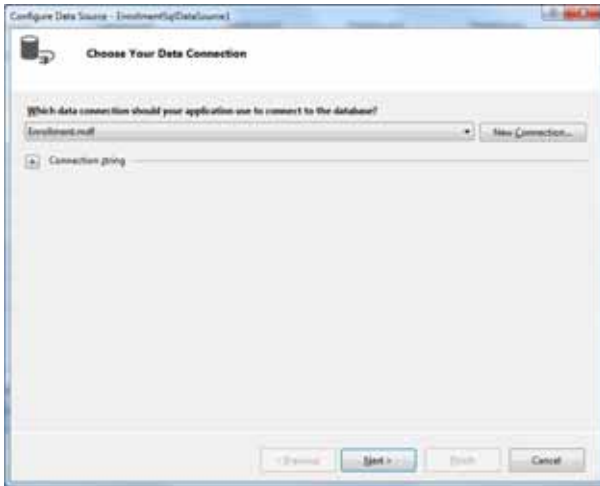
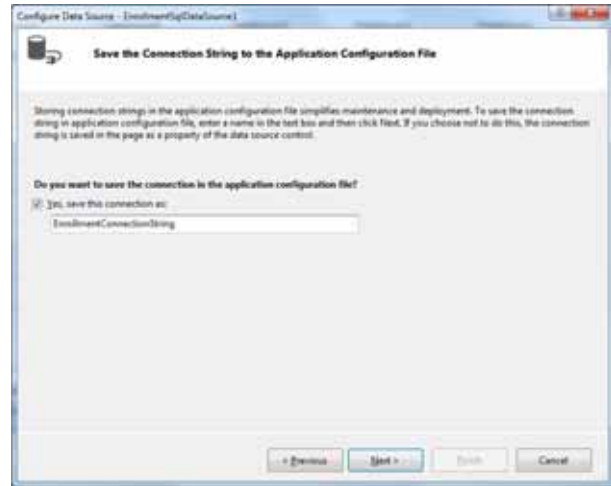


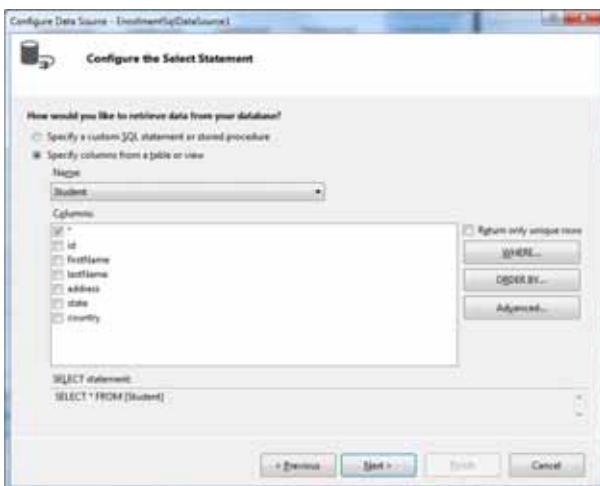
Figure 2: SqlDataSource control for *Enrollment.mdf* database.



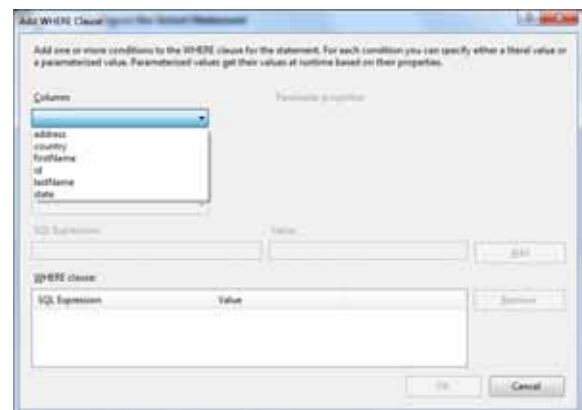
Step 1: Choose a data connection.



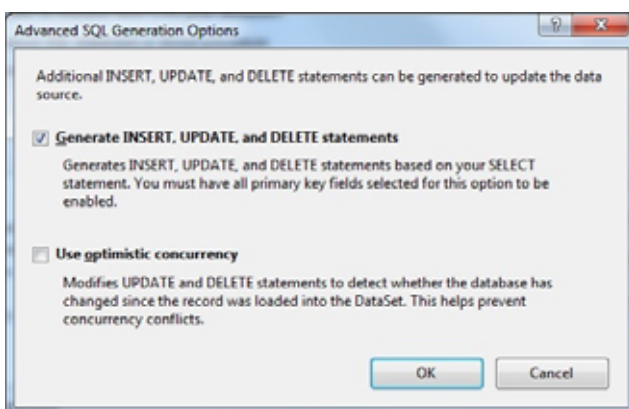
Step 2: Save the connection string.



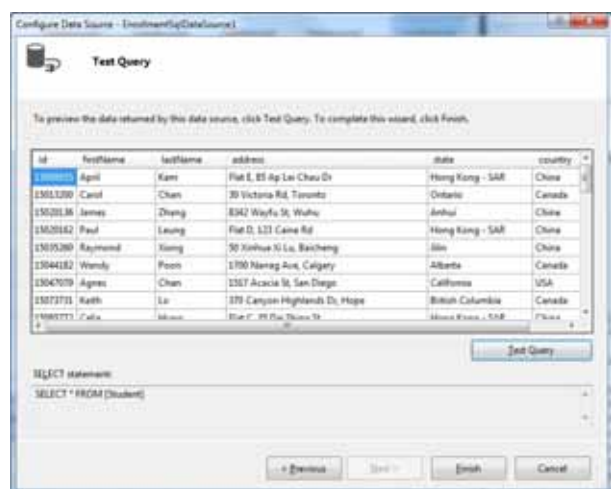
Step 3: Configure the SELECT statement.



Step 4: Set the WHERE clause condition for the SELECT statement.



Step 5: Generate INSERT, UPDATE and DELETE statements.



Step 6: Test the query.

Figure 3: *Configure Data Source* wizard.

Add a SqlDataSource Control

1. Place the cursor outside the end `</div>` tag.
2. Double-click the SqlDataSource control in the Toolbox's Data palette. It will appear in the Document window at the cursor position.
3. Set its ID property to "EnrollmentSqlDataSource1" as shown in Figure 2⁵.

Configure the SqlDataSource Control

1. Select the "Configure Data Source..." option from the "EnrollmentSqlDataSource1" control's smart tag dropdown menu shown in Figure 2. This opens the *Configure Data Source* wizard, shown in Figure 3.
2. On the *Choose Your Data Connection* page, select "Enrollment.mdf" from the dropdown list as shown in Step 1 of Figure 3 and click the "Next" button⁶.
3. On the *Save the Connection String to the Application Configuration File* page, leave the checkbox checked, enter "EnrollmentConnectionString" in the textbox as shown in Step 2 of Figure 3 and click the "Next" button. The connection string is saved in the `Web.config` file of the web site folder and can be viewed in the `<connectionStrings>` section of the `Web.config` file as shown in Figure 4. You will be able to reuse the connection string in other SqlDataSource controls that access the same database as demonstrated in the next section. Moreover, giving the connection string a meaningful name will allow you to remember which database the connection string is for if you are accessing multiple databases.
4. The *Configure Data Source* wizard now allows you to configure and generate the SQL SELECT, INSERT, UPDATE and DELETE statements for the data source as shown in Steps 3, 4 and 5 of Figure 3. On the *Configure the Select Statement* page, set the SQL statement for the *Student* table to "SELECT * FROM [Student]" as follows.
 - a. Select the *Student* table in the *Name* field dropdown list.
 - b. Check only the "*" option in the *Columns* field. This retrieves all columns of the *Student* table⁷.
 Step 3 of Figure 3 shows the constructed SELECT statement. To generate INSERT, UPDATE and DELETE statements based on the SELECT statement, click the "Advanced..." button in the *Configure the Select Statement* page and check the checkbox in the Advanced SQL Generation Options window as shown in Step 5 of Figure 3. Click the "OK" button.
5. On the *Test Query* page, click the "Test Query" button and verify that the query executes as expected. Click the "Finish" button.

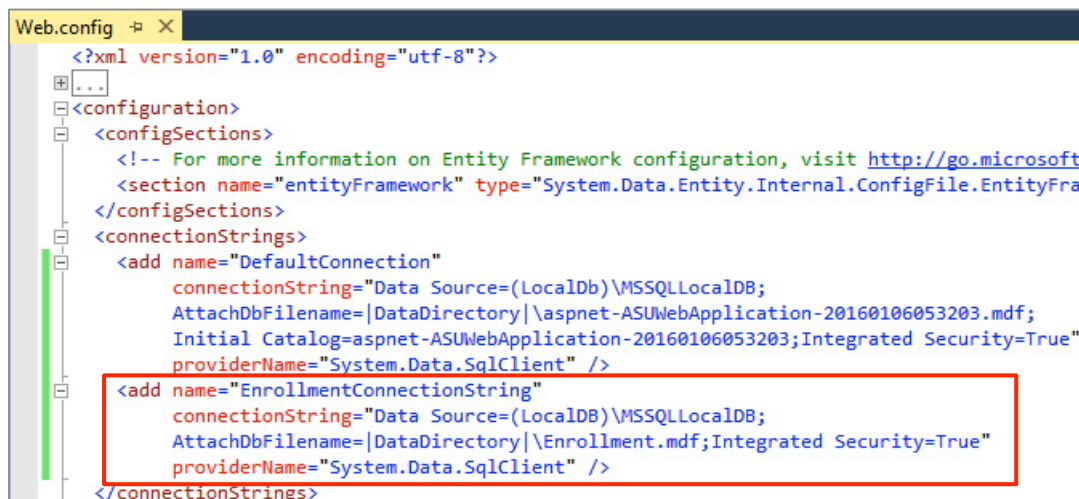


Figure 4: Connection string in `Web.config` file.

⁵ Non-visual controls, such as a data source control, can be hidden in the design surface by unselecting the option "ASP.NET Non Visual Controls" in the "View→Visual Aids" Visual Studio toolbar or pressing "ctrl+Shift+N". The control can still be selected from the Properties window when you want to configure it.

⁶ A connection string is a string that specifies information about a data source and the means of connecting to it. It is passed to ADO in order to initiate the connection. To see the connection string used to make a connection to the *Enrollment.mdf* database as shown in Step 1 of Figure 3, click the "+" button under the database name.

⁷ A criteria for the SELECT statement can also be set by selecting the "WHERE..." button, which displays the *Add WHERE Clause* page as shown in step 4 of Figure 3. To use a parameter in an SQL query, set the value in the Criteria column equal to "?". If the SQL CommandText property contains parameters, then you may also need to edit the Parameters property of the Command object.

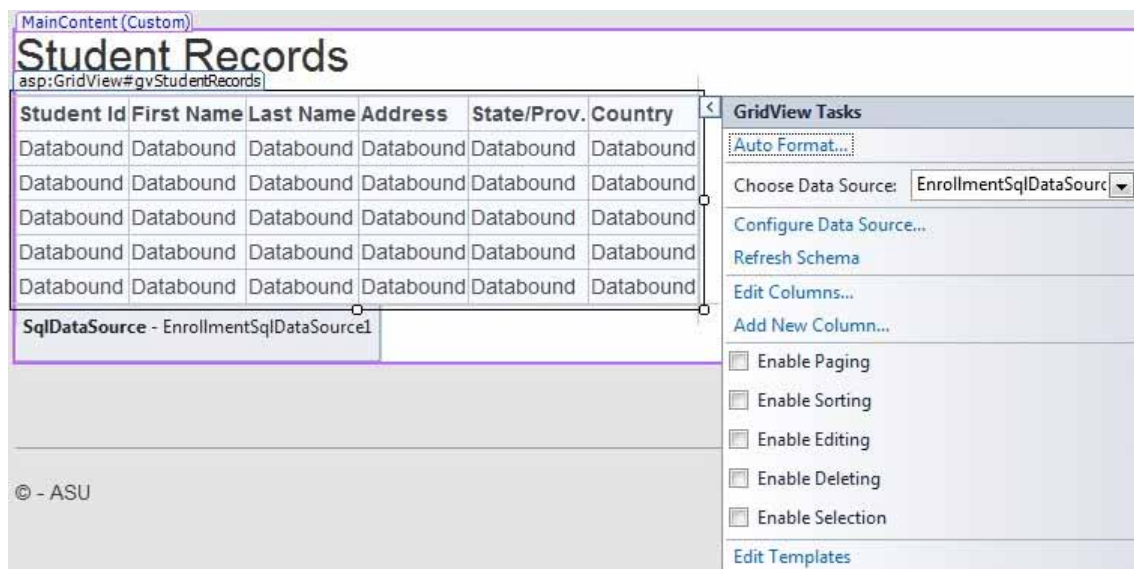


Figure 5: GridView control configured for *Enrollment.mdf* database.

Configuring and Formatting a GridView Control

A GridView control can be configured and formatted from the control's smart tag dropdown menu as shown in Figure 5. Configure and format the "gvStudentRecords" GridView control using its smart tag menu as follows.

1. Select "EnrollmentSqlDataSource1" in the "Choose Data Source" dropdown list as shown in Figure 5. ASP.NET automatically configures the "gvStudentRecords" GridView control with the names of the columns from the *Student* table.
2. Select "Edit Columns..." from the GridView's smart tag dropdown menu. The **Fields** window opens as shown in Figure 6. In this window you can edit the properties of the columns of the GridView. By default, the column headers are assigned the column names in the data source, but they can be changed to be more meaningful. For our example, make the following changes in the **Fields** window.
 - a. Select each field in the **Selected fields:** pane and change its **HeaderText** property in the **BoundField properties:** pane to, respectively, "Student Id", "First Name", "Last Name", "Address", "State/Prov." and "Country" as shown in Figure 6.

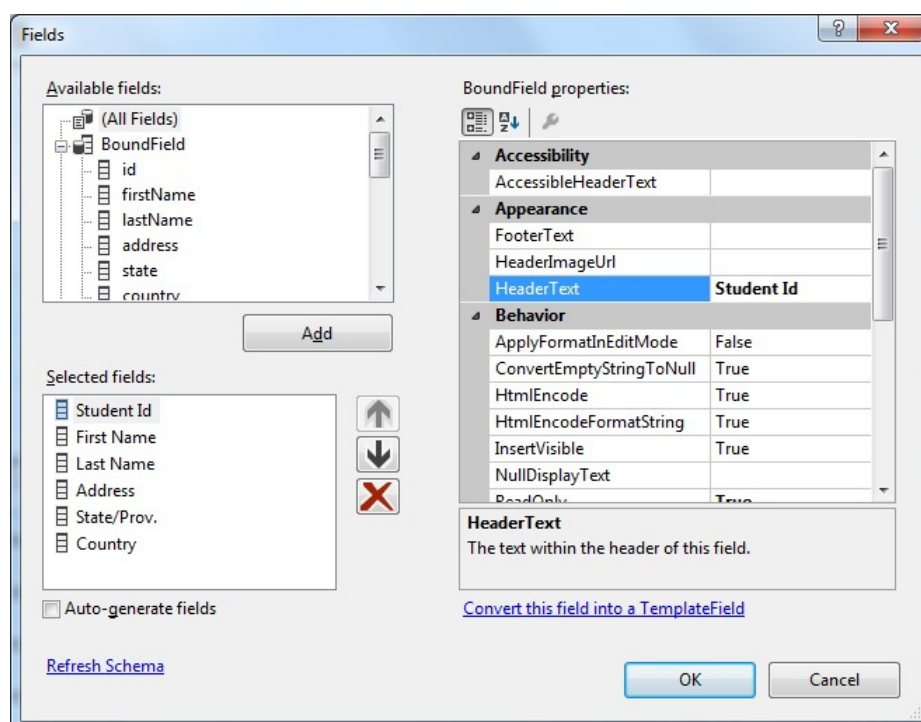


Figure 6: Formatting a GridView control in the **Fields** window.


```

StudentRecords.aspx
<? Page Title="" Language="C#" MasterPageFile="" Site.Master" AutoEventWireup="true" CodeBehind="StudentRecords.aspx.cs" Inherits="ASWebApplication.StudentRecords" %>
<asp:Content ID="Content1" ContentPlaceHolderID="MainContent" runat="server">
<h2>Student Records</h2>
<div>
<asp:GridView ID="gvStudentRecords" runat="server" AllowSorting="true" DataSourceID="EnrollmentSqlDataSource1" AutoGenerateColumns="false">
<Columns>
<asp:BoundField DataField="id" HeaderText="Student Id" ReadOnly="true" SortExpression="id" />
<asp:BoundField DataField="firstName" HeaderText="First Name" SortExpression="firstName" />
<asp:BoundField DataField="lastName" HeaderText="Last Name" SortExpression="lastName" />
<asp:BoundField DataField="address" HeaderText="Address" SortExpression="address" />
<asp:BoundField DataField="state" HeaderText="State/Prov." SortExpression="state" />
<asp:BoundField DataField="country" HeaderText="Country" SortExpression="country" />
</Columns>
</asp:GridView>
</div>
<asp:SqlDataSource ID="EnrollmentSqlDataSource1" runat="server" ConnectionString="" ConnectionStrings:EnrollmentConnectionString %>
DeleteCommand="DELETE FROM [Student] WHERE [id] = @id"
InsertCommand="INSERT INTO [Student] ([id], [firstName], [lastName], [address], [state], [country]) VALUES (@id, @firstName, @lastName, @address, @state, @country)"
SelectCommand="SELECT * FROM [Student]"
UpdateCommand="UPDATE [Student] SET [firstName] = @firstName, [lastName] = @lastName, [address] = @address, [state] = @state, [country] = @country WHERE [id] = @id"
<DeleteParameters>
<asp:Parameter Name="id" Type="String" />
</DeleteParameters>
<InsertParameters>
<asp:Parameter Name="id" Type="String" />
<asp:Parameter Name="firstName" Type="String" />
<asp:Parameter Name="lastName" Type="String" />
<asp:Parameter Name="address" Type="String" />
<asp:Parameter Name="state" Type="String" />
<asp:Parameter Name="country" Type="String" />
</InsertParameters>
<UpdateParameters>
<asp:Parameter Name="firstName" Type="String" />
<asp:Parameter Name="lastName" Type="String" />
<asp:Parameter Name="address" Type="String" />
<asp:Parameter Name="state" Type="String" />
<asp:Parameter Name="country" Type="String" />
<asp:Parameter Name="id" Type="String" />
</UpdateParameters>
</asp:SqlDataSource>
</asp:Content>

```

Figure 7: HTML view of the web form.

Select the Source tab of the Document window to view the HTML that is generated by Visual Studio as shown in Figure 7. The “@...” in the INSERT, UPDATE and DELETE statements are parameters. If a value for them is not specified at design time, then a value needs to be supplied by the user through an ASP.NET web server control at run time. In our example, we see that no parameter values are specified for the SELECT, INSERT, UPDATE and DELETE statements⁸. One way to allow a user to specify values for these parameters at runtime is by using a DetailsView control, which will be discussed shortly in these notes.

View the *Student* table data in a web page as shown in Figure 8⁹.

ASU Web Site Home About Contact Register Log in					
Student Records					
Student Id	First Name	Last Name	Address	State/Prov.	Country
15000655	April	Kam	Flat E, 85 Ap Lei Chau Dr	Hong Kong - SAR	China
15013200	Carol	Chan	30 Victoria Rd, Toronto	Ontario	Canada
15020136	James	Zhang	8342 Wayfu St, Wuhu	Anhui	China
15020162	Paul	Leung	Flat D, 123 Caine Rd	Hong Kong - SAR	China
15035260	Raymond	Xiong	50 Xinhua Xi Lu, Baicheng	Jilin	China
15044182	Wendy	Poon	1700 Narrao Ave, Calgary	Alberta	Canada

Figure 8: *Student* table data viewed in a browser.

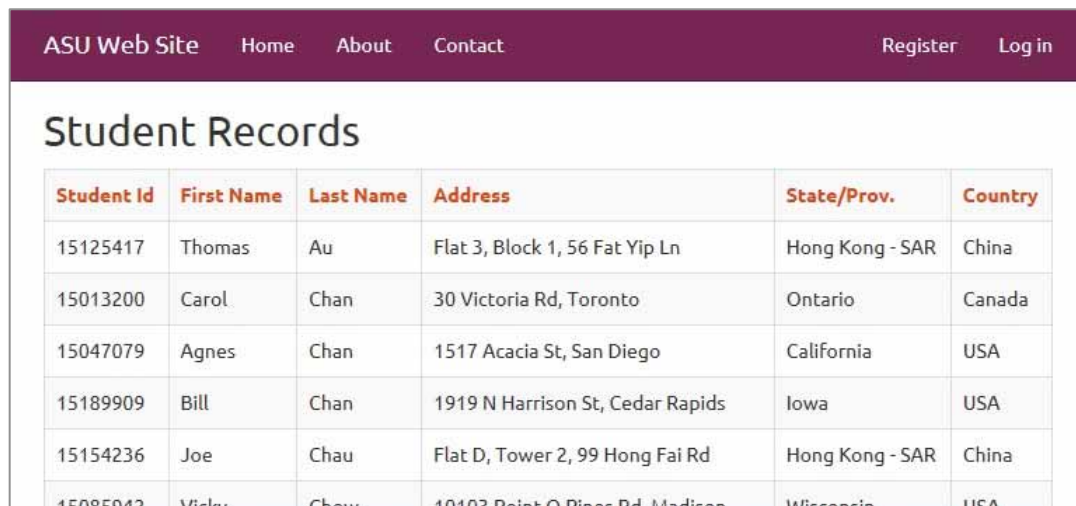
⁸ Figure 7 shows that Visual Studio automatically encloses attribute names in an SQL statement in square brackets []. While the brackets are optional, they are required if you use SQL keywords or special characters in the column names or identifiers. For example, you could name a column [First Name] (with a space), which would require brackets every time you referred to that column. For consistency and safety, Visual Studio encloses all table and column names in brackets even if it is not required.

⁹ Since ASP.NET renders a GridView as a HTML table, the Bootstrap table styles can be used to style a GridView. The StudentRecords.aspx web form uses the Bootstrap styles table, table-striped and table-bordered.

Sorting Records in a GridView

When a column header is clicked, we want the records in the GridView to be sorted on that column. To enable sorting on a GridView control's columns, click the control's smart tag and check the "Enable Sorting" checkbox from the dropdown menu.

View the web form in a browser. You can click on any of the column headers to sort the records by the selected column. For example, Figure 9 shows the records sorted by the "Last Name" column.



ASU Web Site Home About Contact Register Log in					
Student Records					
Student Id	First Name	Last Name	Address	State/Prov.	Country
15125417	Thomas	Au	Flat 3, Block 1, 56 Fat Yip Ln	Hong Kong - SAR	China
15013200	Carol	Chan	30 Victoria Rd, Toronto	Ontario	Canada
15047079	Agnes	Chan	1517 Acacia St, San Diego	California	USA
15189909	Bill	Chan	1919 N Harrison St, Cedar Rapids	Iowa	USA
15154236	Joe	Chau	Flat D, Tower 2, 99 Hong Fai Rd	Hong Kong - SAR	China
15085942	Vicky	Chow	10103 Point O Pines Rd, Madison	Wisconsin	USA

Figure 9: Sorting *Student* data in a GridView.

Paging Records in a GridView

When there are a large number of records, it may be difficult to browse the records in a GridView. Therefore, displaying the data in a GridView as a sequence of pages is recommended. To enable paging in the GridView control, click the control's smart tag and check the "Enable Paging" checkbox from the dropdown menu. You can also customize the paging in the `Properties` window.

For the "gvStudentRecords" GridView control, make the following changes in the `Properties` window.

- Under the `PagerSettings` property:
 - Set the `Mode` to "NextPrevious".
 - Edit the `NextPageText` to "Next".
 - Edit the `PreviousPageText` to "Previous".
- Under the `PagerStyle` property:
 - Set the `HorizontalAlign` to "Center".
- Set the `PageSize` property to "5".

View the web form in a browser. The paging controls are displayed at the bottom and only five records are displayed at a time as shown in Figure 10.

ASU Web Site Home About Contact Register Log in					
Student Records					
Student Id	First Name	Last Name	Address	State/Prov.	Country
15000655	April	Kam	Flat E, 85 Ap Lei Chau Dr	Hong Kong - SAR	China
15013200	Carol	Chan	30 Victoria Rd, Toronto	Ontario	Canada
15020136	James	Zhang	8342 Wayfu St, Wuhu	Anhui	China
15020162	Paul	Leung	Flat D, 123 Caine Rd	Hong Kong - SAR	China
15035260	Raymond	Xiong	50 Xinhua Xi Lu, Baicheng	Jilin	China
Next					

(a) First page of *Student* records.

ASU Web Site Home About Contact Register Log in					
Student Records					
Student Id	First Name	Last Name	Address	State/Prov.	Country
15044182	Wendy	Poon	1700 Narrag Ave, Calgary	Alberta	Canada
15047079	Agnes	Chan	1517 Acacia St, San Diego	California	USA
15073731	Keith	Lo	370 Canyon Highlands Dr, Hope	British Columbia	Canada
15085772	Celia	Hung	Flat C, 35 Dai Shing St	Hong Kong - SAR	China
15085942	Vicky	Chow	10103 Point O Pines Rd, Madison	Wisconsin	USA
Previous Next					

(b) Subsequent page of *Student* records.Figure 10: Paging *Student* table data in a GridView.

Editing and Deleting Records in a GridView

To change the data in a database using a GridView, click the control's smart tag and check the "Enable Editing" and "Enable Deleting" checkboxes. Two additional columns are added to the GridView containing Edit and Delete links as shown in Figure 11.

View the web form in a web browser. The GridView should look as shown in Figure 12 and you should be able to edit and delete rows from the *Student* table¹⁰. Notice that the "Student Id" column cannot be edited since it is the table's primary key.

MainContent (Custom)						
Student Records						
	Student Id	First Name	Last Name	Address	State/Prov.	Country
Edit Delete	Databound	Databound	Databound	Databound	Databound	Databound
Edit Delete	Databound	Databound	Databound	Databound	Databound	Databound
Edit Delete	Databound	Databound	Databound	Databound	Databound	Databound
Edit Delete	Databound	Databound	Databound	Databound	Databound	Databound
Edit Delete	Databound	Databound	Databound	Databound	Databound	Databound
Next						

Figure 11: Adding Edit and Delete links to a GridView.

¹⁰ In place editing in a GridView is not always the best solution to providing editing, especially when there are a large number of columns. Alternatively, you can insert a link column to an edit form as will be discussed shortly.

	Student Id	First Name	Last Name	Address
Update Cancel	15000655	April	Kam	Flat E, 85 A
Edit Delete	15013200	Carol	Chan	30 Victoria
Edit Delete	15020136	James	Zhang	8342 Wayfu
Edit Delete	15020162	Paul	Leung	Flat D, 123
Edit Delete	15035260	Raymond	Xiong	50 Xinhua X

Next

Figure 12: Updatable GridView viewed in a browser.

Try to enter a numeric value as a *State* value. What happens? It is still necessary to add validation code to the control to make sure that only valid data can be entered for an attribute. Additional code can be added in the code-behind file for a web form for handling the specific GridView control events described in Appendix B.

VIEWING DATA IN A DETAILSVIEW CONTROL

The GridView control allows you to select, edit and delete data in the *Student* table, but does not allow you to insert new student data. To support inserting new data, a DetailsView control can be used that is bound to the GridView control and will show the details of the record selected in the GridView control.

Adding a DetailsView Control and Its SqlDataSource Control to a Web Form

A DetailsView control and its SqlDataSource control can be added to the “StudentRecords.aspx” web form as follows.

1. Place the cursor after the `</div>` tag that encloses the GridView control and add another `<div>` tag from the Toolbox’s HTML palette.
2. Place the cursor within the newly added `<div>` tag and, from the Toolbox’s Data palette, add a DetailsView control (see Figure 13).
3. Set the ID property of the DetailsView control to “dvStudentRecords”.
4. Add an SqlDataSource control below the “EnrollmentSqlDataSource1” control and set its ID property to “EnrollmentSqlDataSource2” as shown in Figure 13.

MainContent (Custom)

Student Records

	Student Id	First Name	Last Name	Address	State/Prov.	Country
Select	Databound	Databound	Databound	Databound	Databound	Databound
Select	Databound	Databound	Databound	Databound	Databound	Databound
Select	Databound	Databound	Databound	Databound	Databound	Databound
Select	Databound	Databound	Databound	Databound	Databound	Databound
Select	Databound	Databound	Databound	Databound	Databound	Databound

Next

Student Id	Databound
First Name	Databound
Last Name	Databound
Address	Databound
State/Prov.	Databound
Country	Databound

Edit Delete New

SqlDataSource - EnrollmentSqlDataSource1

SqlDataSource - EnrollmentSqlDataSource2

Figure 13: Layout of StudentRecords.aspx web form.

Configuring the DetailsView Control's SqlDataSource Control

To configure the DetailsView control's SqlDataSource control, do the following.

1. Select "Configure Data Source" from the "EnrollmentSqlDataSource2" control's smart tag menu.
2. On the *Choose Your Data Connection* page, select "EnrollmentConnectionString" from the dropdown list. Click the "Next" button.
3. On the *Configure the Select Statement* page, set the SELECT statement to "SELECT * FROM [Student] WHERE [id] = @id" as follows.
 - a. Select the *Student* table in the *Name* field dropdown list.
 - b. Check only the "*" option in the *Columns* field.
 - c. Click the "WHERE" button.
 - d. In the *Add WHERE Clause* page, select "id" for the *Column* field, "=" for the *Operator* field, "Control" for the *Source* field, "gvStudentRecords" for the *Control ID* field and leave the *Default value* field empty. The student id of the record selected in the "gvStudentRecords" GridView control will be passed to the SQL Select statement of the "dvStudentRecords" DetailsView control allowing it to display all the information about the selected student. Click the "Add" button and then the "OK" button.
 - e. Click the "Advanced..." button and check the checkbox next to the option "Generate INSERT, UPDATE and DELETE statements". Click the "OK" button.
 - f. Click the "Next" button.
4. On the *Test Query* page, click the "Finish" button¹¹.

Configuring and Formatting a DetailsView Control

Make the following changes to the DetailsView control using the control's smart tag dropdown menu.

1. Set the Data Source to "EnrollmentSqlDataSource2".
2. Select "Edit Fields..." in the dropdown menu.
 - a. Set the HeaderText property of "id" to "Student ID", "firstName" to "First Name", etc. as shown in Figure 13. Click the "OK" button.
 - b. Check the "Enable Inserting", "Enable Editing" and "Enable Deleting" checkboxes.

Viewing the Web Form


Make the following changes to the GridView control¹².

1. Uncheck the "Enable Editing" and "Enable Deleting" checkboxes in the control's smart tag menu.
2. Check the "Enable Selection" checkbox in the control's smart tag menu.

View the web form in a browser.

You will notice that when you update, delete or insert a record in the "dvStudentRecords" DetailsView control, the corresponding record is not changed in the "gvStudentRecords" GridView control until you go to another page of the control and then return to the previous page. To fix this problem you need to rebind the "gvStudentRecords" GridView control (i.e., call its `DataBind` method) whenever you make a change in the "dvStudentRecords" DetailsView control (i.e., whenever an update, delete or insert event occurs).

To create event handlers for these events, do the following.

1. Open the **Properties** window of the "dvStudentRecords" DetailsView control and click the events tab  to show the DetailsView control's events as shown in Figure 14.
2. In turn, double-click on the events `ItemDeleted`, `ItemInserted` and `ItemUpdated` in the **Properties** window and, as shown in Figure 15(a), add the code `"gvStudentRecords.DataBind()"` to each event handler in the web form's code-behind file.

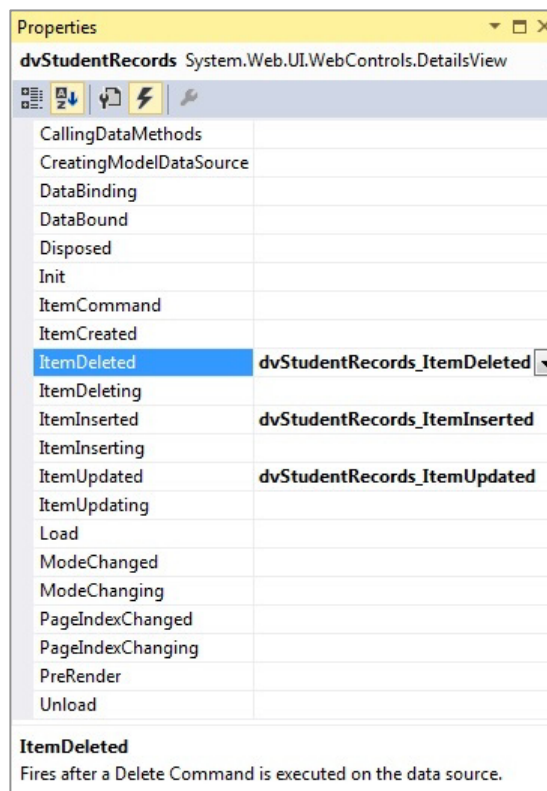
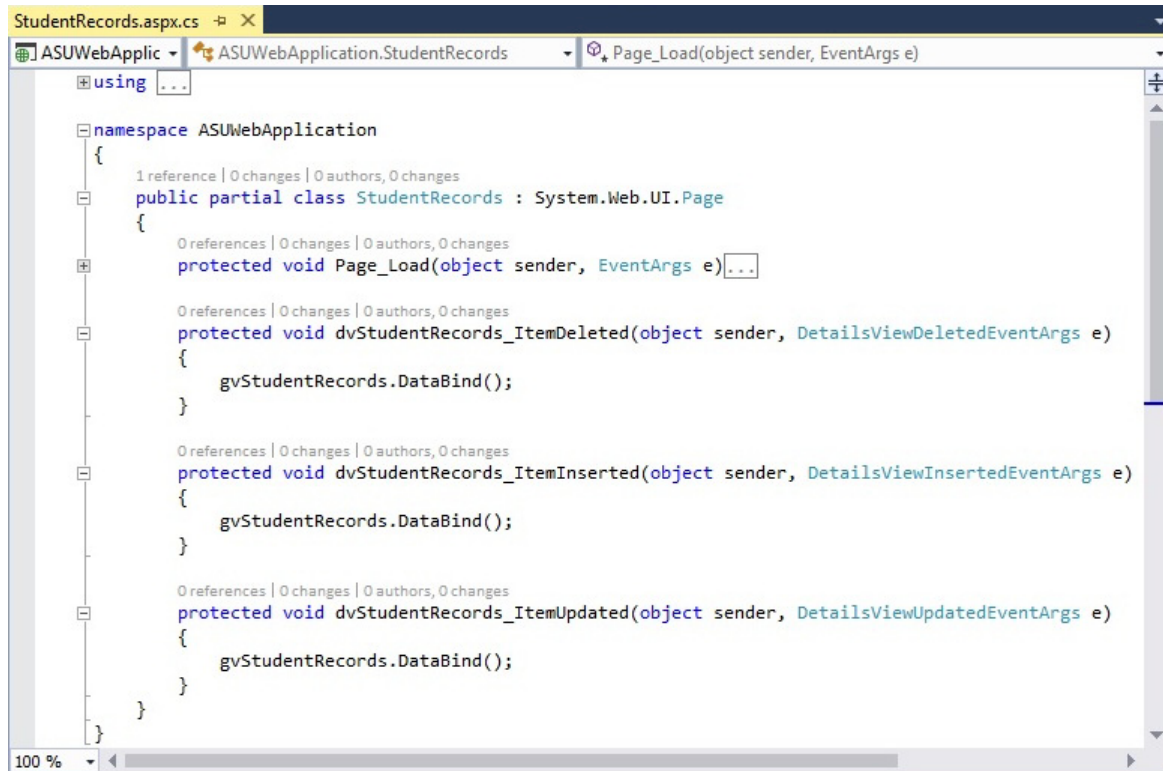


Figure 14: DetailsView control events¹³.

¹¹ If you test the query, no result will be returned!

¹² If not already set, then set the `DataKeysName` property to "id" in the **Properties** window.



(a) Code to rebind the “gvStudentRecords” GridView control.

```

<div>
  <asp:DetailsView ID="dvStudentRecords" runat="server" AutoGenerateRows="False"
    CssClass="table table-striped table-bordered" DataKeyNames="id"
    DataSourceID="EnrollmentSqlDataSource2" Height="50px" Width="125px"
    OnItemDeleted="dvStudentRecords_ItemDeleted"
    OnItemInserted="dvStudentRecords_ItemInserted"
    OnItemUpdated="dvStudentRecords_ItemUpdated">
    <Fields>
      <asp:BoundField DataField="id" HeaderText="Student Id" ReadOnly="True" SortExpression="id" />
      <asp:BoundField DataField="firstName" HeaderText="First Name" SortExpression="firstName" />
      <asp:BoundField DataField="lastName" HeaderText="Last Name" SortExpression="lastName" />
      <asp:BoundField DataField="address" HeaderText="Address" SortExpression="address" />
      <asp:BoundField DataField="state" HeaderText="State/Prov." SortExpression="state" />
      <asp:BoundField DataField="country" HeaderText="Country" SortExpression="country" />
      <asp:CommandField ShowDeleteButton="True" ShowEditButton="True" ShowInsertButton="True" />
    </Fields>
  </asp:DetailsView>
</div>

```

(b) Code to call the “dvStudentRecords” DetailsView control’s event handlers.

Figure 15: Code to rebind the “gvStudentRecords” GridView control when the “dvStudentRecords” DetailsView control changes.

Switch to the Source view of the “StudentRecords.aspx” web form. Notice that Visual Studio has inserted the lines of code highlighted in Figure 15(b) into the DetailsView control’s <asp> tag. This code causes the event handlers in the DetailsView control’s code-behind file to be executed at the server when the corresponding event occurs.

What happens when you try to insert a student record with the same student id as an existing record? Validation code needs to be added to the web form that checks whether the student id of the record to be inserted already exists and returns a meaningful error message to the user. That is, you need to check whether the student id of a new record already exists in the database *before* you try to insert the new record and then cancel the insert operation if a record already exists with that student id. The ItemsInserting event of the DetailsView control can be used to do this.

¹³ The DetailsView control events are described in Appendix C.

ACCESSING DATA IN A DATABASE PROGRAMMATICALLY FROM A WEB FORM

While having Visual Studio generate SQL statements is convenient, it may not be appropriate in all cases. You can write your own SQL statements to access a database in a web form's code-behind file. We will illustrate how to do this by creating a search page for student records that allows us to search for students either in one country or in all countries and possibly filtered by matching (part of) their last name¹⁴. The search result will be displayed in a master/detail style where the "master" web page will display only the student id, first name, last name and country, while the "detail" web page will display all the information about a student selected in the master web page and allow the information to be updated.

Construct the Student Records Search Page

To construct the search page, which will act as the "master" web page containing the search criteria and the result of a search, do the following.

1. Add a new web form named "StudentSearch.aspx" and inherit its layout from the "Site.Master" page.
2. Enter "Search Student Records" on the first line of the MainContent(Custom) ContentPlaceHolder, and set its HTML style to "Heading 2 <h2>".
3. Add three <div> tags from the Toolbar's HTML palette.
4. Inside the first <div> tag add the following controls as shown in Figure 16.
 - a. A Label with ID property cleared, Text property set to "Country:" and AssociatedControlID property set to "ddlCountry".
 - b. A DropDownList with ID property set to "ddlCountry".
 - c. A Label with ID property cleared, Text property set to "Last Name:" and AssociatedControlID property set to "txtLastName".
 - d. A TextBox with ID property set to "txtLastName".
 - e. A Button control with ID property set to "btnSearch" and Text property set to "Search".
5. Inside the second <div> tag add a Label control with ID property set to "lblSearchResultMessage" and Text property cleared.
6. Inside the third <div> tag add a GridView control with ID property set to "gvStudentSearchResult".
7. Select "Edit Items..." in the "ddlCountry" DropDownList control's smart tag dropdown menu and add the items "All", "Canada", "China" and "USA" in the Fields window.

Column0	Column1	Column2
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc

Figure 16: "StudentSearch.aspx" web form.


Configure the GridView Control

1. In the "gvStudentSearchResult" GridView control's smart tag dropdown menu, select "Edit Columns..."
2. In the Fields window, check the "Auto-generate fields" checkbox and click the "OK" button.

Add Code to Retrieve the Student Records

Except for the SQL statement itself, the additional code required to execute an SQL statement programmatically is common to all SQL statements of the same type. Therefore, rather than repeat this common code everywhere an SQL statement is executed, it is good programming practice to "encapsulate" this code in one place and then call it when needed. Therefore, we will use a C# class called "EnrollmentData.cs" containing all of the specific code required to execute SQL statements¹⁵. This class resides inside a folder called "Code_File"¹⁶ that we will add into the Solution Explorer. Download the file "ADONETTutorialCode-Download.zip" from the Tutorial and Lab Schedule section of the course web page, unzip it and drop the folder "Code_File" onto the "ASUWebApplication" node in the Solution Explorer.

In the Solution Explorer, do the following.

1. If the "Code_File" folder cannot be opened, select the "Show All Files"  tab at the top of the Solution Explorer window, open the "Code_File" folder, right-click the "EnrollmentData.cs" file and select "Include In Project" from the popup menu.
2. Right-click the "ASUWebApplication" node and select "Add→Add Reference..." from the popup menu.

¹⁴ In this case, it is actually much easier to construct the search page using an SqlDataSource control.

¹⁵ You should study this code after the tutorial to make sure that you understand what it does.

¹⁶ It is good programming practice to place shared code in one code file. Although ASP.NET has a special folder called "App_Code", it is recommended not to use this folder in a web application project (see <http://vishaljoshi.blogspot.hk/2009/07/appcode-folder-doesnt-work-with-web.html>).

3. In the Reference Manager window, find the entry “System.Windows.Forms”, check the checkbox to its left and click the “OK” button. This will include the methods in this namespace in the web application.

In the Design tab of the Document window, double-click the “Search” button of the “StudentSearch.aspx” web form to create the “btnSearch_Click” event handler. In the “StudentSearch.aspx.cs” file, do the following.

1. Add the following two statements at the top of the file under the last “using...” statement.

```
using System.Data;  
using ASUWebApplication.Code_File;
```

These statements allow the methods in these namespaces to be accessed.

2. From the ADONETTutorialCode-Download folder, add the code in the file “btnSearch_Click.txt” into the “btnSearch_Click” event handler¹⁷. You should study this code after the tutorial to make sure that you understand what it does.

Construct the Page to View the Detailed Student Records

To construct the web page that will display all the information about a student selected in the student search page and allow this information to be updated, do the following.

1. Add a new web form, named “StudentDetails.aspx” and inherit its layout from the “Site.Master” page.
2. Enter “Student Record Details” on the first line of the MainContent(Custom) ContentPlaceHolder control and set its HTML style to “Heading 2 <h2>”.
3. Add two <div> tags from the Toolbar’s HTML palette.
4. Inside the first <div> tag, add a DetailsView control and set its ID property to “dvStudentDetails”.
5. Inside the second <div> tag add a HyperLink control from the Toolbox’s Standard palette, clear its ID property, set its Text property to “Back to search page” and for its NavigateUrl property click the button in the NavigateUrl field and select “StudentSearch.aspx” in the *Contents of folder:* pane of the Select URL window.
6. Add an SqlDataSource control to the web form and set its ID property to “EnrollmentSqlDataSource”.

Configure the DetailsView Control’s SqlDataSource Control

To configure the DetailsView control’s SqlDataSource control, do the following.

1. Select “Configure Data Source” from the “EnrollmentSqlDataSource” control’s smart tag dropdown menu.
2. On the *Choose Your Data Connection* page, select “EnrollmentConnectionString” from the dropdown list and click the “Next” button.

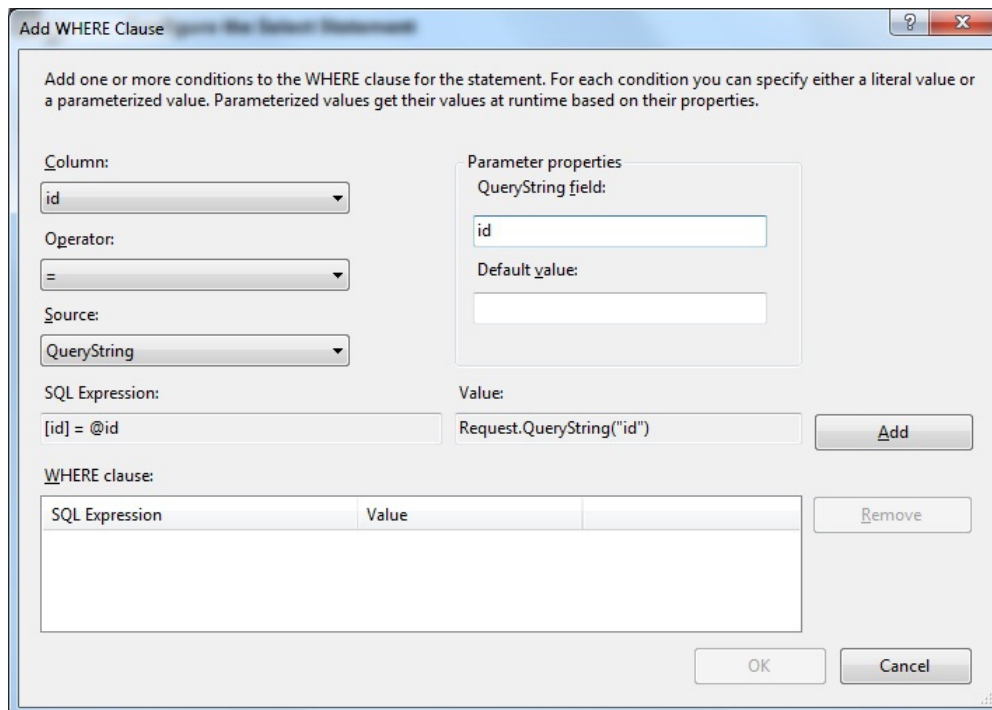


Figure 17: WHERE clause for “dvStudentDetails” DetailsView control.

¹⁷ If several event handlers in the code-behind file will access the database, then the first line of code should be placed outside the btnSearch_Click event handler so that all event handlers that require them have access to the methods in the “EnrollmentData.cs” file.

3. On the *Configure the Select Statement* page, set the SELECT statement to “SELECT [*] FROM [Student] WHERE [id] = @id” as follows.
 - a. Select the *Student* table in the *Name* field dropdown list.
 - b. Check only “*” in the *Columns* field.
 - c. Click the “WHERE” button.
 - d. In the *Add WHERE Clause* page, select “id” for the *Column* field, “=” for the *Operator* field, “QueryString” for the *Source* field, enter “id” in the *QueryString field*: textbox and leave the *Default value*: textbox empty as shown in Figure 17. This will cause the record selected in the “gvStudentSearchResult” GridView control to be displayed in the “dvStudentDetails” DetailsView control. Click the “Add” button and then the “OK” button.
 - e. Click the “Advanced...” button and check the checkbox next to the option “Generate INSERT, UPDATE, and DELETE statements”. Click the “OK” button.
 - f. Click the “Next” button.
4. On the *Test Query* page, click the “Finish” button.

Configure and Format the DetailsView Control

Make the following changes to the “dvStudentDetails” control using the control’s smart tag dropdown menu.

1. Select “EnrollmentSqlDataSource” in the “Choose Data Source” dropdown list.
2. Select “Edit Fields...” and change the HeaderText property for each attribute to that shown in Figure 18.
3. Check the “Enable Editing” and “Enable Deleting” checkboxes.

Student Record Details	
Student Id	abc
First Name	abc
Last Name	abc
Address	abc
State	abc
Country	abc
Edit Delete	
SqlDataSource - EnrollmentSqlDataSource	

Figure 18: Detail web form to display one student record.

Link the Student Records Search Page to the Detailed Student Record Page

To allow the “StudentSearch” page to navigate to the “StudentDetails” page when a record is selected in the GridView control, a hyperlink column needs to be added to the “gvStudentSearchResult” GridView control as shown in Figure 20(a). The URL in this hyperlink field will pass the value of the “id” column of the selected student as a query string.

If the “gvStudentSearchResult” GridView control were *statically* bound to a data source (e.g., via a SqlDataSource control), then a hyperlink column could be added to the GridView control by selecting “Edit Columns...” in the control’s smart tag dropdown menu and doing the following in the *Fields* window (see Figure 19).

1. Select “HyperLinkField” in the *Available fields*: pane and clicking the “Add” button.
2. Make the following changes in the *HyperLinkField properties*: pane as shown in Figure 19.
 - a. Enter “Student Record Link” in the HeaderText property.
 - b. Enter “View Student Record” in the Text property.
 - c. Enter “id” in the DataNavigateUrlFields property.
 - d. Enter “StudentDetails.aspx?id={0}” in the DataNavigateUrlFormatString property.

In general, a hyperlink column can be configured to use either the same text and URL values for the hyperlink in each GridView row or it can set the text and URL values based on the data values bound to each particular row. To specify that the same text and URL be used across all rows, the HyperLinkField’s Text and NavigateUrl properties are used. To set the text and the URL values of the HyperLink field to be based on the data bound to the GridView row, the DataTextField and the DataNavigateUrlFields properties can be used to specify the data fields from which the text and URL values, respectively, should be obtained. While the DataTextField property can only be set to a single data field, DataNavigateUrlFields property can be set to a comma-delimited list of data fields.

Frequently the text or URL is based on a combination of the current row’s data fields *and* some static markup. In our example, we want the hyperlink column’s URLs to be “StudentDetails.aspx?id=id” where *id* is each row’s student id value. Notice that both static and data-driven values are needed since the “StudentDetails.aspx?id=” portion of a hyperlink’s URL is static, whereas the *id* portion is data-driven as its value is each row’s student id value. To indicate a combination of both static and data-driven values, the DataTextFormatString and the DataNavigateUrlFormatString properties are used. In these properties the static markup is entered as needed and then the marker {0} is used to specify where the data field value in the

DataTextField or DataNavigateUrlFields properties are to appear. If the DataNavigateUrlFields property has multiple data fields specified, {0} is used where you want the first field value inserted, {1} for the second field value, etc.

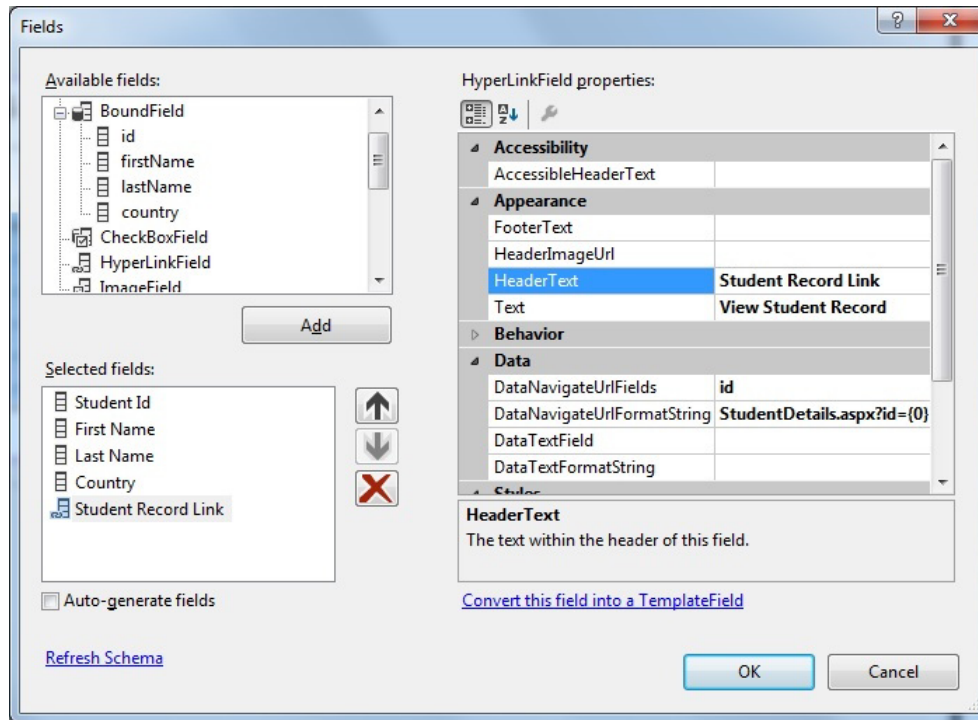


Figure 19: Adding a HyperLinkField field to the “gvStudentSearchResult” GridView control.

For our example, since we want the hyperlink text to be the same for all rows in the GridView, we can set the HyperLinkField’s Text property to “View Student Record” as shown in Figure 19. The URL value, however, needs to be different for each row since the URL will pass the value of the student id in the row in a query string to the DetailsView control in the “StudentDetails.aspx” web form. Therefore, the NavigateURL property cannot be used in this case. Instead, as shown in Figure 19, we need to use the DataNavigateUrlFields property set to “id”, since that is the data field whose value we need to customize on a per-row basis, and the DataNavigateUrlFormatString property set to “StudentDetails.aspx?id={0}”, since “id” is the first data field (index 0) in the DataNavigateUrlFields property.

Since we are accessing the *Enrollment.mdf* database programmatically to do the searching (i.e., the GridView control is not statically bound to a data source, but is *dynamically* bound to the result of the search), we cannot add a hyperlink column to the GridView control using the preceding method. Instead, we need to add the hyperlink column programmatically in the web form’s code-behind file. Nevertheless, the basic method of creating a hyperlink column and setting its properties as discussed above are the same.

To add the code required to create a hyperlink column and set its properties, do the following.

1. Double-click the event “RowDataBound” in the “gvStudentSearchResult” GridView control’s list of events in its Properties window.
2. From the ADONETTutorialCode-Download folder, add the code in the file “gvStudentSearchResult_RowDataBound.txt” into the “gvStudentSearchResult_RowDataBound” event handler. This code also edits the text in the Header row of the GridView control to that shown in Figure 20(a) since the header text also cannot be set using the GridView control’s Fields window.

View the web form in a browser.

Enter a query in the Search Student Records web page such as Country = “USA” and Last Name = “wong”. You should get a result similar to Figure 20(a). If you click on the hyperlink “View Student Record” in the first record in the search result you should get a result similar to Figure 20(b).

ASU Web Site

Search Student Records

Country: Last Name:

The following records match your query.

Student Id	First Name	Last Name	Country	Student Record Link
15141966	Otto	Wong	USA	View Student Record
15221340	Carmen	Wong	USA	View Student Record

(a) Student records search page

ASU Web Site

Student Record Details

Student Id	15221340
First Name	Carmen
Last Name	Wong
Address	3503 Bloomer Springs Rd, Richmond
State	Virginia
Country	USA
Edit Delete	

[Back to search page](#)

(b) Student records detail page.

Figure 20: Student records search page and details page.

APPENDIX A: ASP.NET DATA-BOUND WEB SERVER CONTROLS¹⁸

Chart

The Chart control allows charts for complex statistical or financial analysis to be included in a web page.

DataList

The DataList control displays data items in a table, and optionally supports selecting and editing the items. The content and layout of the list items is defined using templates.

DataPager

The DataPager control allows paging through data in either a ListView control or a control that implements the IPagableItemContainer interface.

DetailsView

The DetailsView control renders a single record at a time as a table and provides the capability to page through multiple records, as well as to insert, update and delete records. It is often used in a master-details scenario where the selected record of the master control (e.g., a GridView) determines the record displayed by the DetailsView control.

EntityDataSource

The EntityDataSource control supports data binding scenarios in Web applications that use the ADO.NET Entity Framework by managing create, read, update, and delete operations against a data source on behalf of data-bound controls on the same page.

FormView

The FormView control renders a single record at a time from a data source and provides the capability to page through multiple records, as well as to insert, update, and delete records, similar to the DetailsView control, except that it requires the user to define the rendering of each item using templates, instead of using data control fields.

GridView

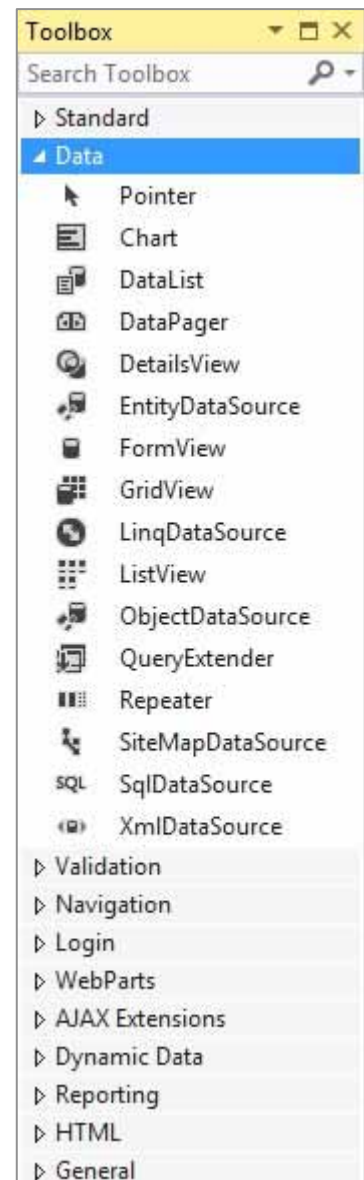
The GridView control displays data as a table and provides the capability to sort columns, page through data, and edit or delete a single record.

LinqDataSource

The LinqDataSource control enables the use of Language-Integrated Query (LINQ) in a web page through markup text to retrieve and modify data from a data object. LINQ simplifies the interaction between object-oriented programming and relational data by applying the principles of object-oriented programming to relational data.

ListView

The ListView control displays data from a data source in a format define by templates. The template contains the formatting, controls, and binding expressions that are used to lay out the data. The ListView control is useful for data in any repeating structure, similar to the DataList and Repeater controls. However, unlike the DataList and Repeater controls, the ListView control implicitly supports edit, insert, and delete operations, as well as sorting and paging functionality.



¹⁸ See <http://msdn.microsoft.com/en-us/library/ms228214%28v=vs.100%29.aspx> for more information.

ObjectDataSource

ObjectDataSource enables a declarative databinding model against a variety of underlying data stores, such as SQL databases or XML. The ObjectDataSource control allows applications to be structured using a traditional three-tiered architecture and still take advantage of the ease-of-use benefits of the declarative databinding model in ASP.NET.

QueryExtender

The QueryExtender control is used to create filters for data that is retrieved from a data source, without using an explicit Where clause in the data source.

Repeater

The Repeater control renders a read-only list from a set of records returned from a data source. All HTML layout, formatting, and style tags must explicitly be declared within the control's templates.

SiteMapDataSource

The SiteMapDataSource retrieves navigation data from a site map provider and passes the data to controls that can display that data, such as the TreeView and Menu controls.

SqlDataSource

The SqlDataSource control represents a connection to an ADO.NET SQL database provider, such as SQL Server, OLEDB, ODBC, or Oracle.

XmlDataSource

The XmlDataSource control allows XML data to be displayed in either a hierarchical or tabular form.

APPENDIX B: GRIDVIEW CONTROL EVENTS¹⁹

The GridView control exposes paging and sorting events, and events that occur when the current row is created or bound to data. Events are also raised when a command control, such as a Button control that is included as part of the GridView control, has been clicked. The following events are exposed by the GridView control.

RowCommand

Occurs when a button is clicked in the GridView control. This event is often used to perform a task when a button is clicked in the control.

PageIndexChanging

Occurs when a pager button is clicked, but before the GridView control performs the paging operation. This event is often handled to cancel the paging operation.

PageIndexChanged

Occurs when a pager button is clicked, but after the GridView control performs the paging operation. This event is commonly handled when you need to perform a task after the user navigates to a different page in the control.

SelectedIndexChanging

Occurs when a row's Select button (a button with its CommandName property set to "Select") within a GridView control is clicked, but before the GridView control performs the select operation. This event is often handled to cancel the selection operation.

SelectedIndexChanged

Occurs when a row's Select button within a GridView control is clicked, but after the GridView control performs the select operation. This event is often handled to perform a task after a row is selected in the control.

Sorting

Occurs when the hyperlink to sort a column is clicked, but before the GridView control performs the sort operation. This event is often handled to cancel the sorting operation or to perform a custom sorting routine.

Sorted

Occurs when the hyperlink to sort a column is clicked, but after the GridView control performs the sort operation. This event is commonly handled to perform a task after the user clicks on a hyperlink to sort a column.

RowDataBound

Occurs when a row in the GridView control is bound to a data record. This event is often handled to modify the contents of a row when the row is bound to data.

RowCreated

Occurs when a new row is created in the GridView control. This event is often handled to modify the layout or appearance of a row when the row is created.

RowDeleting

Occurs when a row's Delete button (a button with its CommandName property set to "Delete") within a GridView control is clicked, but before the GridView control deletes the record from the data source. This event is often handled to cancel the deleting operation.

RowDeleted

Occurs when a row's Delete button is within a GridView control clicked, but after the GridView control deletes the record from the data source. This event is often handled to check the results of the delete operation.

¹⁹ From <http://msdn.microsoft.com/en-us/library/h4f8ekch%28v=vs.100%29.aspx>

RowEditing

Occurs when a row's Edit button (a button with its CommandName property set to "Edit") within a GridView control is clicked, but before the GridView control enters edit mode. This event is often handled to cancel the editing operation.

RowCancelingEdit

Occurs when a row's Cancel button (a button with its CommandName property set to "Cancel") within a GridView control is clicked, but before the GridView control exits edit mode. This event is often handled to stop the cancel operation.

RowUpdating

Occurs when a row's Update button (a button with its CommandName property set to "Update") within a GridView control is clicked, but before the GridView control updates the record. This event is often handled to cancel the updating operation.

RowUpdated

Occurs when a row's Update button within a GridView control is clicked, but after the GridView control updates the record. This event is often handled to check the results of the update operation.

DataBound

This event is inherited from the BaseDataBoundControl control and occurs after the GridView control has finished binding to the data source.

APPENDIX C: DETAILSVIEW CONTROL EVENTS²⁰

The DetailsView control raises events that occur when the current record is displayed or changed. Events are also raised when a command control such as a Button that is part of the DetailsView control has been clicked. The following events are exposed by the DetailsView control.

PageIndexChanging

Occurs when a Page button (a button with its CommandName property set to "Page") has been clicked, but before navigating to a new page of data.

PageIndexChanged

Occurs when a Page button has been clicked, but after a navigating to a new page of data.

ItemCommand

Occurs when a button within a DetailsView control is clicked.

ItemCreated

Occurs when a record is created in a DetailsView control.

ItemDeleting

Occurs when a Delete button (a button with its CommandName property set to "Delete") within a DetailsView control is clicked, but before the delete operation.

ItemDeleted

Occurs when a Delete button within a DetailsView control is clicked, but after the delete operation.

ItemInserting

Occurs when an Insert button (a button with its CommandName property set to "Insert") within a DetailsView control is clicked, but before the insert operation.

ItemInserted

Occurs when an Insert button within a DetailsView control is clicked, but after the insert operation.

ItemUpdating

Occurs when an Update button (a button with its CommandName property set to "Update") within a DetailsView control is clicked, but before the update operation.

ItemUpdated

Occurs when an Update button within a DetailsView control is clicked, but after the update operation.

ModeChanging

Occurs when a DetailsView control attempts to change between edit, insert, and read-only modes, but before the CurrentMode property is updated.

ModeChanged

Occurs when a DetailsView control attempts to change between edit, insert, and read-only modes, but after the CurrentMode property is updated.

DataBound

This event is inherited from the BaseDataBoundControl control and occurs after the DetailsView control has finished binding to the data source.

²⁰ From <http://msdn.microsoft.com/en-us/library/ms228102%28v=vs.100%29.aspx>

APPENDIX D: ADO.NET ARCHITECTURE OVERVIEW²¹

ADO.NET consists of a set of objects that expose data access services to the .NET environment. These objects provide communication between relational and non-relational systems through a common set of components. The `System.Data` namespace is the core of ADO.NET and contains classes used by all data providers. Visual Studio provides several wizards and other features that can be used to generate ADO.NET data access code. The two key components of ADO.NET are Data Provider and DataSet/DataTable, which are used to perform all data management operations on specific databases.

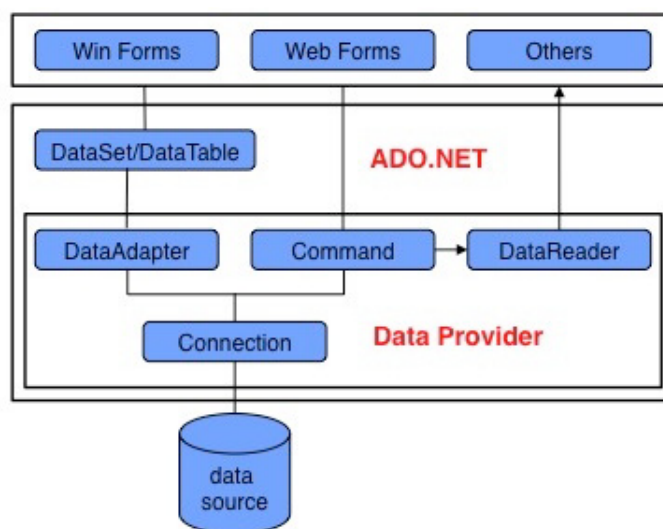


Figure 21: ADO.NET architecture.

Data Provider

The Data Provider classes are meant to work with different kinds of data sources. A data provider contains Connection, Command, DataAdapter and DataReader objects. The .Net Framework includes Data Providers for several data sources in the Data palette of the `Toolbox`.

SqlConnection Object

The `SqlConnection` object handles the physical connection to a SQL data source. The `SqlConnection` object needs certain information to recognize the data source and to log on to it properly. This information is provided through a connection string. When the connection is established, SQL commands will execute with the help of the `Command` object to retrieve or manipulate the data in the data source. Once the data source activities complete, the connection should be closed to roll back any pending transactions and release the data source resources.

Command Object

The `Command` object is used to execute an SQL statement or stored procedure against the data source specified in the `Connection` object. The `Command` object provides a number of `Execute` methods that can be used to perform the SQL queries in a variety of fashions. The `Command` object has a property called `CommandText`, which contains a `String` value that represents the command that will be executed against the data source.

DataAdapter Object

The `DataAdapter` object populates a `DataSet` or `DataTable` object with results from a data source. It serves as a bridge between a `DataSet/DataTable` object and a physical database for retrieving and saving data. An `SqlDataAdapter` object is used in combination with a `DataSet` or `DataTable` object by using its `Fill` method to change the data in the `DataSet` or `DataTable` to match the data in the data source and its `Update` method to change the data in the data source to match the data in the `DataSet` or `DataTable`.

DataReader Object

The `DataReader` object is a stream-based, forward-only, read-only retrieval of query results from a data source and requires a live connection with the database.

²¹ Adapted from <http://asp.net-informations.com/ado.net/ado-architecture.htm>.

DataSet/DataTable

A DataSet and DataTable are ASP.NET data structures used to hold data retrieved from a data source in memory where program code can access and manipulate the data. They provide a disconnected, completely independent representation of the result retrieved from a data source. While a DataTable can contain at most one database table, a DataSet can contain multiple database tables; in effect, a DataSet holds a collection of DataTables. Consequently, DataTable objects contained in a DataSet can be related to each other with DataRelation objects. Figure 22 shows the basic components of a DataTable. The components of a DataSet are the same except that there can be multiple tables each of which has the same components as those shown in Figure 22.

A table within a DataTable or DataSet contains Columns and Rows collections, which means that standard methods for accessing and manipulating collections can be used. The Columns collection contains, for each column, a name, a data type specification and maybe an assigned default value. Each table row in the Rows collection contains one cell for each column. The Table class has an extensive set of methods for editing and managing versions of column and row data and for event notifications when changes occur.

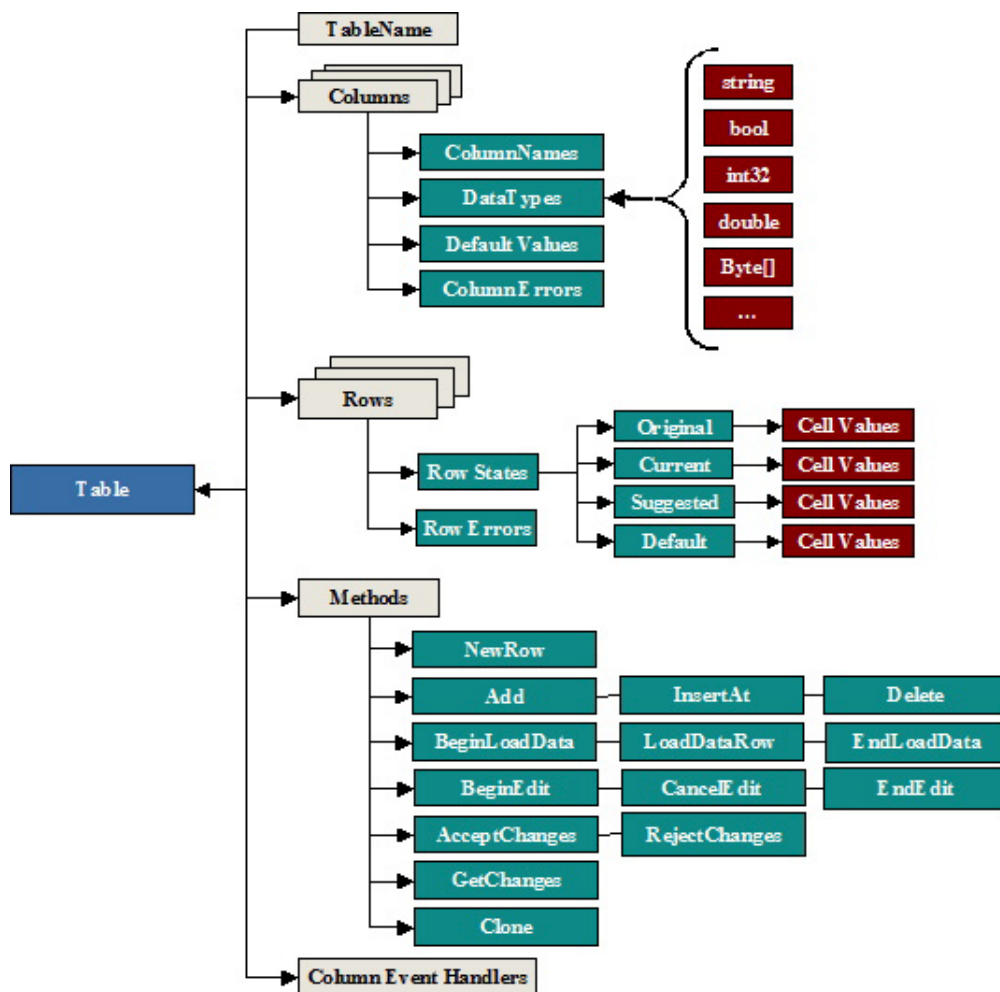


Figure 22: DataTable components.

DataTable/DataSet Tutorials and Methods

DataTable tutorial: <http://www.codeproject.com/KB/grid/PracticalGuideDataGrids2.aspx>

DataTable methods:

https://msdn.microsoft.com/en-us/library/system.data.datatable_methods%28v=vs.110%29.aspx

DataSets tutorial:

<http://www.codeproject.com/Articles/6180/A-Practical-Guide-to-NET-DataTables-DataSets-and>

DataSet methods:

https://msdn.microsoft.com/en-us/library/system.data.dataset_methods%28v=vs.110%29.aspx