[1. Brief Introduction 1](#_Toc367270526)

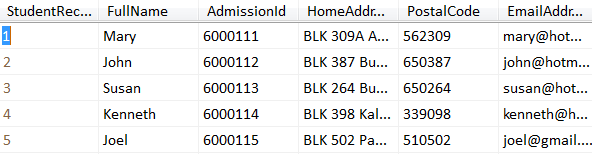
[2. Experiment: DataTable, DataRow and ListView 2](#_Toc367270527)

[2.1 How It Works 4](#_Toc367270528)

**Data Access Topic (Part 2) - Managing Data Inside Your Application**

# 1. Brief Introduction

Assume that you have used this statement, string studentData = ""; to declare a string data type variable. Refer to the Fig. 1, will this studentData variable be able to hold information that is retrieved from a Student table in the database? It is not possible at all. As a result, you need to consider using DataTable.



You need to create an object of a suitable class, which is capable of storing: (i) The information and (ii) The structure of the information.

Fig. 1

studentData variable

Declaring a DataTable is straightforward:

DataTable studentDataTable = new DataTable();

A DataTable instance such as studentDataTable will be able to hold relational data information retrieved from the database. It will also have various useful properties and methods for you to use in your programming logic to manage the data within the DataTable variable.

# 2. Experiment: DataTable, DataRow and ListView

Before jumping right into the topic on programming database driven applications, you need to grasp some background basic concepts listed below:

i) Private scope variable in a Class (In our case study here, we are referring the studentDataTable object which is declared inside the Windows Form class).

Extensive reference is found in <http://www.dotnetperls.com/class>

ii) DataTable Class (Reference: <http://www.dotnetperls.com/datatable>)

iii) DataRow Class (Reference: <http://www.dotnetperls.com/datarow>)

iv) ListView control (Reference: <http://www.dotnetperls.com/listview>)

Create a new project, ExperimentBasicSystemDataObjects and add a new Windows Form, frmExperimentDataTableDataRowAndListView.cs. This Windows Form has 2 Button controls (btnLoad and btnUpdate). It also has a ListView control, lvStudent. You will use this project to experiment the frequently used fundamental code which is applied to DataTable, DataRow and ListView.

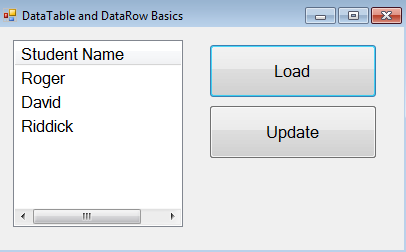
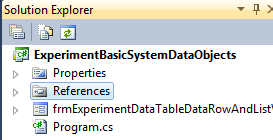


Fig. 2

btnLoad

btnUpdate

lvStudent

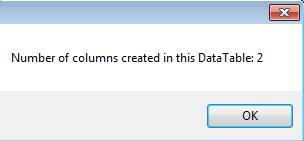
The code for the Windows Form, **frmExperimentDataTableDataRowAndListView** is shown in Listing 1. Refer to Listing 1, you need to have the following 2 important observations:

1. A private member variable, studentDataTable is created within the Windows Form class (line 13). This variable (which uses the private modifier). As a result, all the code which is residing inside the btnLoad\_Click, btnUpdate\_Click and frmExperimentDataTableDataRowAndListView\_Load event handler can use this studentDataTable variable. These 3 sub routines are declared inside the same form class block as the studentDataTable.
2. DataTable is ideal for storing data. Additional reading can be found in <http://www.dotnetperls.com/datatable>.

| Listing 1 The frmExperimentDataTableDataRowAndListView.cs C# code |
| --- |
| 1. using System; 2. using System.Collections.Generic; 3. using System.ComponentModel; 4. using System.Data; 5. using System.Drawing; 6. using System.Linq; 7. using System.Text; 8. using System.Windows.Forms; 9. namespace ExperimentBasicSystemDataObjects 10. **{** 11. public partial class frmExperimentDataTableDataRowAndListView : Form 12. **{** 13. private DataTable studentDataTable; 14. public frmExperimentDataTableDataRowAndListView()   This variable can be manipulated by any code written in the sub-routines such as btnLoad\_Click, etc. within the form class.   1. **{** 2. InitializeComponent(); 3. **}** 4. private void frmExperimentDataTableDataRowAndListView\_Load(object sender, EventArgs e) 5. **{** 6. this.studentDataTable = new DataTable(); 7. this.studentDataTable.Columns.Add("FullName"); 8. this.studentDataTable.Columns.Add("MobileContact"); 9. lvStudent.View = View.Details; 10. lvStudent.Columns.Add(“Student Name”, 200); 11. MessageBox.Show("Number of columns created in this DataTable: " + this.studentDataTable.Columns.Count.ToString()); 12. **}***//form load method* 13. private void btnLoad\_Click(object sender, EventArgs e) 14. **{** 15. this.studentDataTable.Rows.Add("Roger", "91111111"); 16. this.studentDataTable.Rows.Add("David", "92222222"); 17. this.studentDataTable.Rows.Add("Riddick", "92113333"); 18. MessageBox.Show("Number of DataRows in this DataTable is " + this.studentDataTable.Rows.Count); 19. *//-- Show inside a ListView Control --* 20. lvStudent.View = View.Details; 21. lvStudent.Items.Clear();*//Clear the ListView control first before adding ListViewItems* 22. foreach (DataRow dr in this.studentDataTable.Rows) 23. **{** 24. ListViewItem item = new ListViewItem(); 25. item.Text = dr["FullName"].ToString(); 26. item.SubItems.Add(dr["MobileContact"].ToString()); 27. lvStudent.Items.Add(item); 28. **}** 29. **}***//btnLoad\_Click* 30. private void btnUpdate\_Click(object sender, EventArgs e) 31. **{** 32. DataRow studentRow = this.studentDataTable.Rows[1]; 33. studentRow["FullName"] = "Davidson"; 34. *//-- Show inside a ListView Control --* 35. lvStudent.View = View.Details; 36. lvStudent.Items.Clear(); *//Clear the ListView control first before adding ListViewItems* 37. foreach(DataRow dr in this.studentDataTable.Rows)**{** 38. ListViewItem item = new ListViewItem(); 39. item.Text = dr["FullName"].ToString(); 40. lvStudent.Items.Add(item); 41. **}** 42. **}***//btnUpdate\_Click* 43. **}***//form class frmExperimentDataTableDataRowAndListView* 44. **}***//namespace ExperimentBasicSystemDataObjects* |

## *2.1 How It Works*

Here is a descriptions of this sample program when executed:

i) When you start the program, a dialog box window will appear showing a text information "Number of columns created in this DataTable: 2".

This dialog box will immediately appear when the windows form is loading.

When you start the program, the form is loading. As a result, all the code (line 20 – 25) inside the frmExperimentDataTableDataRowAndListView\_Load event handler will execute. At line 20, the this.studentDataTable = new DataTable(); will create the object studentDataTable. The this.studentDataTable means "The studentDataTable which belongs to the form".

The code at line 21 and line 22:

this.studentDataTable.Columns.Add("FullName"); and this.studentDataTable.Columns.Add("MobileContact");

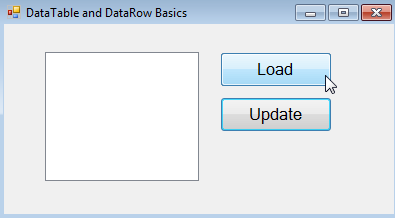
is calling the Add method of the Columns collection property twice to add 2 columns into the DataTable object, studentDataTable. You can imagine the empty studentDataTable variable storage which looks like:



The code at line 25 is just using the Count property of the Columns collection which belongs to the studentDataTable to return a value of 2. The dialog box message helps to verify that the code at line 21 and 22 has indeed successfully added two columns into the DataTable, studentDataTable.

ii) When the Windows form appears, you click the btnLoad Button control so that the code inside the btnLoad\_Click event handler can execute.

Fig. 2

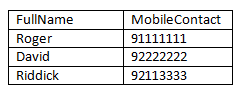


btnUpdate

btnLoad

lvStudent

The code at line 29 to 31 will call the Add method of the Rows collection which belongs to the studentDataTable 3 times to create 3 DataRows of information inside the studentDataTable.



this.studentDataTable.Rows.Add("Roger", "91111111");

this.studentDataTable.Rows.Add("David", "92222222");

this.studentDataTable.Rows.Add("Riddick", "92113333");

The code at line 32 MessageBox.Show("Number of DataRows in this DataTable is " + this.studentDataTable.Rows.Count); will display a dialog box message "Number of DataRows in this DataTable is 3" because the .NET engine will evaluate the code this.studentDataTable.Rows.Count as 3. This message provides an assurance that the studentDataTable does contain 3 rows of information.

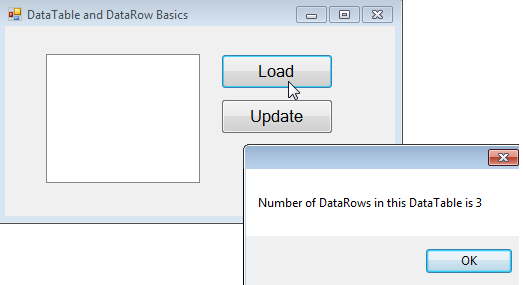


Fig. 3

A dialog box is shown when the btnLoad control is clicked.

After dismissing the dialog box window, the code from line 34 to 41 will execute to show the DataTable object, studentDataTable's content inside a ListView control, lvStudent.

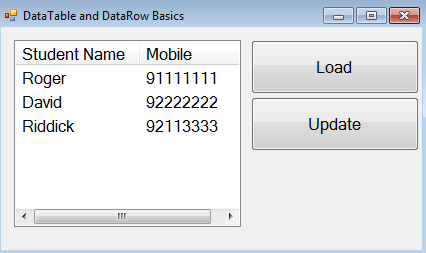


Fig. 4

lvStudent.View = View.Details; tells the ListView control to make the column header visible.

lvStudent.Items.Clear(); tells the ListView control to clear all its ListViewItems (if there are any).

The code at line 36: foreach (DataRow dr in this.studentDataTable.Rows) will get the code at line 38 to 41 to execute 3 times. During each pass of the loop, the DataRow variable, dr will represent one DataRow at a time so that it can be used to create ListViewItems which is later added into the Items collection property of the ListView control, lvStudent.

Let's go through the first pass of the loop to have a good idea on this logic. When the .NET engine sees the code foreach (DataRow dr in this.studentDataTable.Rows), it will automatically get the dr to represent the first DataRow object inside the studentDataTable's Rows collection (Fig. 5).

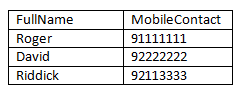


Fig. 5

Represent/Reference

To create one ListViewItem in the ListView control which shows Roger and the respective mobile contact information, the code at line 38 will execute to create a new ListViewItem object, item. Then, the code at 39 will copy the dr object's ["FieldName"] information into the Text property of the item object. The item.Text = dr["FullName"].ToString(); will be evaluated as item.Text = "Roger"; At line 40, the item.SubItems.Add(dr["MobileContact"].ToString()); will be evaluated as item.SubItems.Add("91111111"); After executing the code from 39 to 40, the ListItemView object, item will have the full name and mobile contact information. The code at line 41: lvStudent.Items.Add(item); calls the Add method to add the populated ListViewItems object into the Listview control, lvStudent.

If you go through the idea of the previous paragraph for 2 more times (2nd pass of the loop and 3rd pass of the loop), the foreach looping block creates 3 ListViewItem objects inside the ListView control, lvStudent to display all the information which was stored in the Windows Form's member property, studentDataTable.

iii) Refer to Fig. 6. When you click the btnUpdate Button control, the C# logic (line 46 to 55) within the btnUpdate\_Click event handler will make some changes to the storage by changing the David to Davidson. After making the changes in the this.studentDataTable, the ListView control, lvStudent, is repopulated to display the updated information.

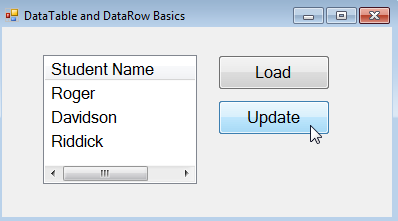


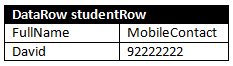
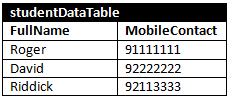
Fig. 6

The code at line 44:

DataRow studentRow = this.studentDataTable.Rows[1];

is creating a DataRow studentRow which references/points at the 2nd DataRow object in the DataTable object, studentDataTable.

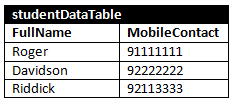
Fig. 7



The next code:

studentRow["FullName"] = "Davidson";

will assign a string "Davidson" into the FullName column of the studentRow object. Actually, the .NET engine evaluates this code as this.studentDataTable[1]["FullName"] = "Davidson". After having this line of code executed, the state of studentDataTable can be described by the Fig. 8.



diagrambelow:

Fig. 8

The rest of the code (line 49 to 55) in btnUpdate\_Click will update the ListView control's display so that we can be assured that, the code really works.