|  |
| --- |
| Record Navigation |
|  | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Overview of Data Navigation | |  |      |  | | --- | | Introduction | |  |   From the user's point of view, data navigation consists of moving from one piece of information of a database to another. This could be done from one control to another or from one record to another. To support the various scenarios of data navigation, there are three main categories of data display the user will face. Put it another way, there are three types of scenarios you as the database developer will present to the user.   |  | | --- | | Scenarios of Data Navigation | |  |   There are three main ways you [display data](http://www.functionx.com/vcsharp2003/databases/Lesson02.htm) to the user:  **Datasheet Display**: A datasheet displays its information in series of columns and rows, the intersections of which are called cells. A datasheet is meant to display as many records as possible all at once in the same view:  http://www.functionx.com/vcsharp2003/images/grid2.gif  To navigate among cells of a datasheet, the user can click a value and click another as desired. Alternatively, the user can press Tab continuously to move from one cell to another. Some datasheet controls also allow pressing Enter to move from one cell to another. In some cases, some cells can be made to display controls such as check or combo boxes.  **Windows Controls Display**: While a datasheet displays its information in cells, you may prefer to use more elaborate controls to display data. This means that you can use edit boxes, list-based controls, button-based controls (such as radio buttons or check boxes), etc. Here is an example:  http://www.functionx.com/vcsharp2003/forms/bcr1.gif  To navigate from one control to another, the user can click continuously. In most cases, the user can also press Tab to move among records.  An alternative to this scenario is to add a datasheet portion to the form, combined with other controls, in what is referred to as master/detail.  **Record Navigation**: A datasheet is meant to display all of its records or as many records as possible. The above display of various controls is used display one record at a time. When the user has finished using the record, such as during data entry, you must provide the user with a way to restart. An alternative is to allow the user to move from one record to another. Of course, this is taken care of by the datasheet. If there are many fields for each record, a datasheet may not be suitable. Therefore, you can display one record at a time in one view but allow the user to navigate to the next or the previous record. This can be done by creating appropriate menu items or by adding navigation buttons on the form. You can position such buttons in the bottom section of the form as follows:  Record Navigation Using Buttons  In this case, to navigate among records, the user can click the appropriate button.  To implement any other these scenarios, you must appropriately bind the database fields and records to the desired control(s). | |  | | --- | |  | |

|  |
| --- |
| Datasheet Binding |
|  |

|  |
| --- |
| Introduction |
|  |

|  |
| --- |
| Although you will create your databases as SQL files, to provide a friendlier environment, you create forms and other graphical accessories that allow the users to access and view information. In Visual Studio, you would create a Windows application.  ADO.NET provides various ways to connect to a database, the simplest consists of using a **SqlDataAdapter** variable because this gives you direct access to the SQL Server database. With other techniques, or as traditionally done, you would have to create an ODBC Data Source. The Microsoft Visual Studio .NET programming environment is so close to SQL Server that you can create and manage a database as if you were using a single application to take care of this. |

|  |
| --- |
| Practical LearningPractical Learning: Creating a Data Source |
|  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. Start Microsoft Visual Studio .Net 2. From the Server Explorer window, expand the Servers node. Expand the server that has your SQL Server installation or the database you want to use. Then expand the SQL Servers node 3. Expand the name of the server and expand the CIC1 database that was created in the previous lesson 4. Right-click the Tables node and click New Table 5. Fill the table with the following fields:  |  |  |  |  | | --- | --- | --- | --- | | Column Name | [Data Type](http://www.functionx.com/vcsharp2003/databases/Lesson02.htm) | Length | All Nulls | | EmployeeID | int |  |  | | EmployeeNo | char | 6 | Cleared | | DateHired | smalldatetime |  |  | | FirstName | varchar | 20 |  | | LastName | varchar | 20 | Cleared | | Salary | smallmoney |  |  | | IsMarried | bit |  |  |  1. Right-click EmployeeID and click Set Primary Key 2. While EmployeeID still has focus, in the lower part of the table, set its Identity to Yes 3. Save the table as **Employees** and close it 4. From Server Explorer, under the Tables node of CIC1, double-click Employees 5. Fill it up with the following fields:  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | EmployeeNo | DateHired | FirstName | LastName | Salary | IsMarried | | 43-957 | 02/14/02 | Albert | Samson | 16.58 | 1 | | 68-205 | 02/14/02 | Leslie | Ellison | 16.24 | 0 | | 14-528 | 02/25/02 | Elias | Hawkins | 14.26 | 0 | | 92-253 | 03/02/02 | Anselme | Wagner | 22.05 | 1 | | 68-828 | 05/18/02 | James | Haught | 24.32 | 1 |  1. Close the table 2. Start a new Windows Application named **CIC2** 3. Change the Text of the form to **Clarksville Ice Cream** 4. Set its **StartPosition** to **CenterScreen** 5. On the main menu, click File -> New -> File... 6. In the New File dialog box, click Icon File (.ico) and click Open 7. Design the icon as follows:  http://www.functionx.com/vcsharp2003/icondesign/cis1.gif 8. Right-click an empty area in the design section, position the mouse on Current Icon Image Types, and click 16x16, 16 colors 9. Design the 16x16 version of the icon as follows:   http://www.functionx.com/vcsharp2003/icondesign/cis1a.gif 10. To save the icon, on the main menu, click File -> Save Icon1 As... 11. Set the name to cic and note where (the folder) you are saving the icon 12. Close the icon tab 13. While the form is selected, in the Properties window, click the Icon field and click its ellipsis button. Then select the above cic.ico and click Open 14. To add another form to the project, on the main menu, click Project -> Add Windows Form 15. In the **Name** text box, replace the string with **Employees** and click Open 16. Change its Icon to the above cic.ico and change its Text to Clarksville Ice Cream 17. Set its **ShowInTaskbar** property to **False** and its **StartPosition** to **CenterScreen** 18. From the Server Explorer, under the Tables node of CIC1, drag the Employees table and drop it on the Employees form |

|  |
| --- |
| Introduction to Data Sets |
|  |

|  |
| --- |
| The information stored in an object such as a database table is called a set, which is simply the group of its records. Because information is in fact referred to as data, the group of records in a table is also called a [data set](http://www.functionx.com/vcsharp2003/databases/Lesson02.htm). To identify and manage the set of records, or data set, of a table, the Microsoft .[NET Framework](http://www.functionx.com/vcsharp2003/databases/Lesson02.htm) provides a class called **DataSet**.  After getting a connection to a database, you are ready to process it. You can use the data directly or, to better manage it, you can pass it to **DataSet** variable. |

|  |
| --- |
| Practical LearningPractical Learning: Creating a Data Set |
|  |

|  |
| --- |
| 1. On the main menu, click Data -> Generate Dataset... 2. Click the text box to the right of the New radio button and change it to **dsCIC**   Generate Dataset 3. Click OK |

|  |
| --- |
| The [DataGrid](http://www.functionx.com/vcsharp2003/databases/Lesson02.htm) Control |
|  |

|  |
| --- |
| Data of a table is represented as a series of columns and rows. The columns are horizontal and hold the categories of information. A row contains entries various columns. The intersection of a column and a row is called a cell. The group of values stored in cells of a particular row is called [a record](http://www.functionx.com/vcsharp2003/databases/Lesson02.htm).  To display information of a table, in the Server Explorer, you can just double-click it.  Instead of using a table to display data in cells, the Microsoft .NET Framework provides a control called **DataGrid**. This object also organizes its information in series of columns and rows whose intersections, called cells, hold the data of a table. To use a **DataGrid**, click it in the Toolbox and click the form. After doing this, you can specify that the information that displays in the cells will come from a **DataSet**. To do this, you can set the **DataSource** property of the **DataGrid** to the **DataSet** you have created. To actually display data in the **DataGrid**, you can call the Fill() method of the data adapter you are using. |

|  |
| --- |
| Practical LearningPractical Learning: Displaying Data in a [Data Grid](http://www.functionx.com/vcsharp2003/databases/Lesson02.htm) |
|  |

|  |  |  |
| --- | --- | --- |
| 1. On the Windows [Forms](http://www.functionx.com/vcsharp2003/databases/Lesson02.htm) section of the Toolbox, click DataGrid and click the form 2. Position the DataGrid to the top-left corner and enlarge it 3. Right-click the DataGrid and click Auto Format 4. In the Formats list, click Colorful 3   Auto Format 5. Click OK 6. On the form, click the DataGrid to make sure it is selected. In the Properties windows, change the following properties: CaptionText: **Employees Records** DataSource: dsCompAssets1.Employees Anchor: Top, Bottom, Left, Right   Data Grid 7. Double-click an unoccupied area of the form to generate its Load event 8. Implement it as follows:  |  | | --- | | private void Employees\_Load(object sender, System.EventArgs e)  {  this.sqlDataAdapter1.Fill(this.dsEmployees1);  } |  1. Display the first form, Form1.cs [Design] 2. On the Toolbox, click Button and click the top-left section of the form 3. Change the new button's **Text** to **Employees** and change its **Name** to **btnEmployees** 4. Double-click the new Employees button and implement its Click event as follows:  |  | | --- | | private void btnEmployees\_Click(object sender, System.EventArgs e)  {  Employees frmEmployees = new Employees();  frmEmployees.ShowDialog();  } |  1. Press Ctrl + F5 to execute the application 2. On the first form, click the Employees button   Data grid 3. After viewing the form, close it and close the main form |

|  |
| --- |
| Data Navigation With Windows Controls |
|  |

|  |
| --- |
| Introduction |
|  |

|  |
| --- |
| Although the records of a database are stored in tables, data sheets sometimes provide unfriendly environment for data entry. Of course, to avoid using database tables that can appear boring, you can use the **DataGrid** control. One of the characteristics of data grids is that they display all of their record set, or at least as much as their width and height can allow. If some fields require much room, a data grid as good looking as it can be, would not be suitable. Imagine that one of the columns contains fields of text of various paragraphs. In this case, the records should be displayed one at a time.  If you want to display one record at a time on a form, you can use Windows controls and provide a means for the user to navigate back and forth in the records. |

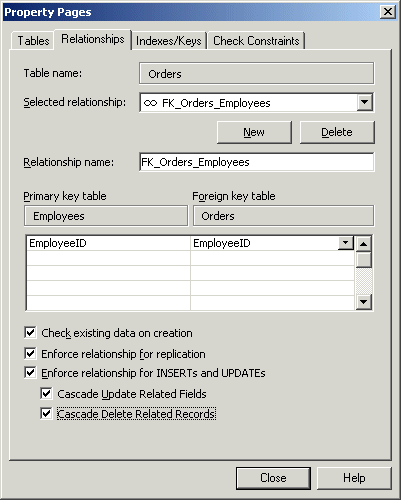
|  |
| --- |
| Practical LearningPractical Learning: Display Data in Windows Controls |
|  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. To add a new form, on the main menu, click Project -> Add Windows Form... 2. Replace the Name of the form with **Flavors** and press Enter 3. Change its Icon to the above cic.ico and change its **Text** to **Clarksville Ice Cream - Flavors** 4. Set its **ShowInTaskbar** property to **False** and its **StartPosition** to **CenterScreen** 5. From the Server Explorer, under the Tables node of CIC1, drag the Flavors table and drop it on the form 6. On the main menu, click Data -> Generate Dataset... 7. Click the New radio button and change the string to **dsFlavors**   Generate Dataset 8. Click OK 9. Click the Toolbox tab, click [Windows Forms](http://www.functionx.com/vcsharp2003/databases/Lesson02.htm), and select the controls to design the form as follows:  |  | | --- | | http://www.functionx.com/vcsharp2003/forms/flavors1.gif | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Control | Name | DataBindings -> Text dsFlavors1 - Flavors. | Text | Other Properties | | Form |  |  |  |  | | Label |  |  | Flavor ID: |  | | TextBox | txtFlavorID | FlavorID |  | ReadOnly: True TextAlign: Right | | Label |  |  | Flavor: |  | | TextBox | txtFlavor | Flavor |  |  | | Label |  |  | Composition: |  | | TextBox | txtComposition | Composition |  | Multiline: True ScrollBars: Vertical | | Button | btnClose |  | Close |  | | Button | btnFirst |  | | < |  | | Button | btnPrevious |  | < |  | | Button | btnNext |  | > |  | | Button | btnLast |  | > | |  | |  1. Double-click an empty area on the form to access its Load event and implement it as follows:  |  | | --- | | private void Flavors\_Load(object sender, System.EventArgs e)  {  this.sqlDataAdapter1.Fill(this.dsFlavors1);  } |  1. Double-click each of the buttons and implement them as follows:  |  | | --- | | private void btnClose\_Click(object sender, System.EventArgs e)  {  Close();  }  private void btnFirst\_Click(object sender, System.EventArgs e)  {  this.BindingContext[this.dsFlavors1, "Flavors"].Position = 0;  }  private void btnPrevious\_Click(object sender, System.EventArgs e)  {  this.BindingContext[this.dsFlavors1, "Flavors"].Position =  this.BindingContext[this.dsFlavors1, "Flavors"].Position - 1;  }  private void btnNext\_Click(object sender, System.EventArgs e)  {  this.BindingContext[this.dsFlavors1, "Flavors"].Position =  this.BindingContext[this.dsFlavors1, "Flavors"].Position + 1;  }  private void btnLast\_Click(object sender, System.EventArgs e)  {  this.BindingContext[this.dsFlavors1, "Flavors"].Position =  this.BindingContext[this.dsFlavors1, "Flavors"].Count - 1;  } |  1. Display the first form, Form1.cs [Design] 2. On the Toolbox, click Button and click under the existing Employees button of the form 3. Change the new button's **Text** to **Flavors** and change its **Name** to **btnFlavors** 4. Double-click the new Flavors button and implement its Click event as follows:  |  | | --- | | private void btnFlavors\_Click(object sender, System.EventArgs e)  {  Flavors frmFlavors = new Flavors();  frmFlavors.ShowDialog();  } |  1. Execute the application 2. On the form, click the Flavors button   Flavors 3. Close the forms and return to your programming environment |

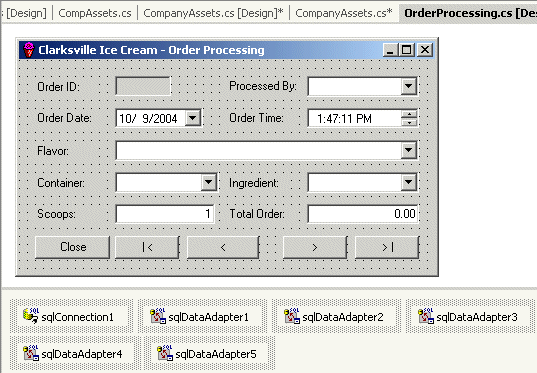
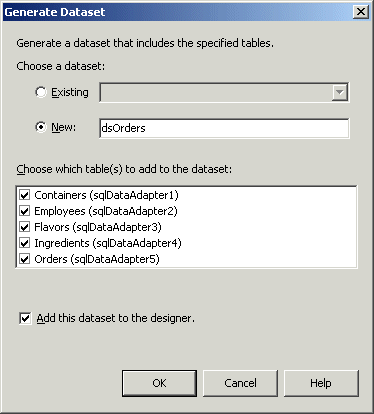
|  |
| --- |
| Data Display with List-Based Controls |
|  |

|  |
| --- |
| Besides the DataGrid, the .NET Framework provides many other controls that are list-based. These include the combo boxes, list boxes, three views, list views, etc. These controls are ready for data display so much that they is little or not code involved but they must be appropriately configured. The list-based controls are usually meant to display data from another table. For this reason, they may use a foreign key that represents a child table. This means that, as done for the master/detail scenario, you should first have a parent, then a child table and the child table must have a foreign key that represents the table of the records that would be displayed on the list-based control. |

|  |
| --- |
| Practical LearningPractical Learning: Binding Data With a Combo Box |
|  |

1. In Server Explorer, under the Tables node of CIC1, right-click Orders and click Design Table
2. To add a new column, right-click OrderDate and click Insert Column
3. Change the properties of the new column as follows:  
   **Column Name: EmployeeID  
   Data Type: int  
   Allow Nulls: cleared  
   Default Value: 1**
4. Right-click anywhere in the table and click Relationships...
5. In the Property Pages, click New
6. In the Primary Key Table combo box, select Employees
7. In the first combo box under it, select EmployeeID
8. In the combo box under Orders, select EmployeeID  
      
   
9. Click Close and close the table
10. When asked whether you want to save, click Yes twice
11. To add a new form, on the main menu, click Project -> Add Windows Form...
12. Set the Name to **OrderProcessing** and press Enter
13. Change its Icon to the above App.ico and change its **Text** to  
    **Clarksville Ice Cream - Order Processing**
14. Set its **StartPosition** to **CenterScreen**
15. Design the form as follows:

|  |
| --- |
| http://www.functionx.com/vcsharp2003/forms/cic3.gif |
| |  |  |  |  | | --- | --- | --- | --- | | Control | Name | Text | Additional Properties | | Label |  | Order ID: |  | | TextBox | txtOrderID |  | ReadOnly: True | | Label |  | Processed By: |  | | ComboBox | cboProcessedBy |  | DropDownStyle: DropDownList | | Label |  | Order Date: |  | | DateTimePicker | dtpOrderDate |  | Format: Short | | Label |  | Order Time: |  | | DateTimePicker | dtpOrderTime |  | Format: Time: ShowUpDown: True | | Label |  | Flavor: |  | | [ComboBox](http://www.functionx.com/vcsharp2003/databases/Lesson02.htm) | cboFlavor |  | DropDownStyle: DropDownList | | Label |  | Container: |  | | ComboBox | cboContainer |  | DropDownStyle: DropDownList | | Label |  | Ingredient: |  | | ComboBox | cboIngredient |  | DropDownStyle: DropDownList | | Label |  | Scoops: |  | | TextBox | txtScoops | 1 | TextAlign: Right | | Label |  | Total Order: |  | | TextBox | txtTotalOrder | 0.00 | TextAlign: Right | | Button | btnClose | Close |  | | Button | btnFirst | | < |  | | Button | btnPrevious | < |  | | Button | btnNext | > |  | | Button | btnLast | > | |  | |

1. In Server Explorer, under the Table node, click Containers
2. Press and hold Ctrl
3. Click Employees, Flavors, Ingredients, and Orders
4. Release Ctrl
5. Click and drag the selection and drop it on the form  
      
   
6. On the main menu, click Data -> Generate Dataset...
7. Click the New radio button and change the name to **dsOrders**  
      
   
8. Click OK
9. On the form, click each control and, in the Properties window, bind it as follows

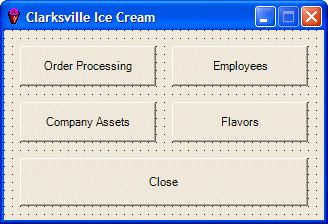
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Control | DataBindings-> Text-> dsOrders1 | DataSource | DisplayMember | ValueMember | DataBindings-> SelectedValue-> dsOrders1 |
| txtOrderID | Orders.OrderID |  |  |  |  |
| cboProcessedBy |  | dsOrders1.Employees | LastName | EmployeeID | Orders.EmployeeID |
| dtpOrderDate | Orders.OrderDate |  |  |  |  |
| dtpOrderTime | Orders.OrderTime |  |  |  |  |
| cboFlavor |  | dsOrders1.Flavors | Flavor | FlavorID | Orders.FlavorID |
| cboContainer |  | dsOrders1.Containers | Container | ContainerID | Orders.ContainerID |
| cboIngredient |  | dsOrders1.Ingredients | Ingredient | IngredientID | Orders.IngredientID |
| txtScoops | Orders.Scoops |  |  |  |  |

1. Double-click an empty area of the form and change the Load event as follows:

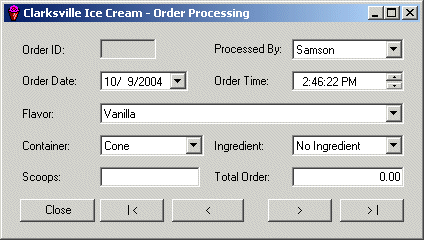
|  |
| --- |
| private void OrderProcessing\_Load(object sender, System.EventArgs e)  {  this.sqlDataAdapter5.Fill(this.dsOrders1);  this.sqlDataAdapter4.Fill(this.dsOrders1);  this.sqlDataAdapter3.Fill(this.dsOrders1);  this.sqlDataAdapter2.Fill(this.dsOrders1);  this.sqlDataAdapter1.Fill(this.dsOrders1);  } |

1. Double-click the buttons and implement their events as follows:

|  |
| --- |
| private void btnClose\_Click(object sender, System.EventArgs e)  {  Close();  }  private void btnFirst\_Click(object sender, System.EventArgs e)  {  this.BindingContext[this.dsOrders1, "Orders"].Position = 0;  }  private void btnPrevious\_Click(object sender, System.EventArgs e)  {  this.BindingContext[this.dsOrders1, "Orders"].Position =  this.BindingContext[this.dsOrders1, "Orders"].Position - 1;  }  private void btnNext\_Click(object sender, System.EventArgs e)  {  this.BindingContext[this.dsOrders1, "Orders"].Position =  this.BindingContext[this.dsOrders1, "Orders"].Position + 1;  }  private void btnLast\_Click(object sender, System.EventArgs e)  {  this.BindingContext[this.dsOrders1, "Orders"].Position =  this.BindingContext[this.dsOrders1, "Orders"].Count - 1;  } |

1. Display the first form, Form1.cs [Design]
2. On the Toolbox, click Button and click the form
3. Change the new button's **Text** to **Order Processing** and change its **Name** to **btnOrders**
4. Add another button to the form. Change its Text to Close and its Name to btnClose
5. Reposition the buttons as follows:  
      
   
6. Double-click the Order Processing and the Close buttons
7. Implement their Click events as follows:

|  |
| --- |
| private void btnOrders\_Click(object sender, System.EventArgs e)  {  OrderProcessing frmOrders = new OrderProcessing();  frmOrders.Show();  }  private void btnClose\_Click(object sender, System.EventArgs e)  {  Close();  } |

1. Execute the application and click the different buttons  
      
   
2. Close the forms

|  |
| --- |
|  |

|  |  |  |
| --- | --- | --- |
|  | | |
| [Previous](http://www.functionx.com/vcsharp2003/databases/Lesson01.htm) | [Copyright © 2004-2010 FunctionX, Inc.](http://www.functionx.com/vcsharp2003/index.htm) | [Next](http://www.functionx.com/vcsharp2003/databases/Lesson03.htm) |
|  | | |