Hands-On Lab

Lab Manual

HOL057 –
Data Features in Windows Forms 2.0

Please do not remove this manual from the lab
The lab manual will be available from CommNet

Contents

Lab 1: Data Features in Windows Forms 2.0

Lab Objective

80 minutes
The objective of this lab is to introduce a few of the new Microsoft® Windows® Forms data features in Microsoft Visual Studio® 2005.
The goal for these new features is to increase developer productivity and simplify data binding to business objects.

This lab will also introduce the new Windows Forms DataGridView control, a flexible, scalable and easier to use alternative to the DataGrid control.

You will complete the following exercises in this lab:

- Addng DataSources to the Datasources Window
- Binding a DataSource to a form
- Customizing DataGridViews
- Binding to Business Objects

Exercise 1 –Create a datasource and add it to DataSources Window

In this exercise, you will create a datasource that connects to a SQL Server database.

Task 1 – Create Windows Application

- Navigate to **Start | Programs | Microsoft Visual Studio Beta2 | Microsoft Visual Studio Beta2**
- Select **File | New | Project** menu command
- Select **C#** or **Visual Basic Windows Application**
    - Name = 'DataLab'
    - Location = 'C:\labs\hol057'
- Click **OK** to create the project

Task 2 – Add a DataSource to your project

- Select **Data | Add New DataSource** menu command
- Data Source Configuration Wizard appears (see Figure 1.1)

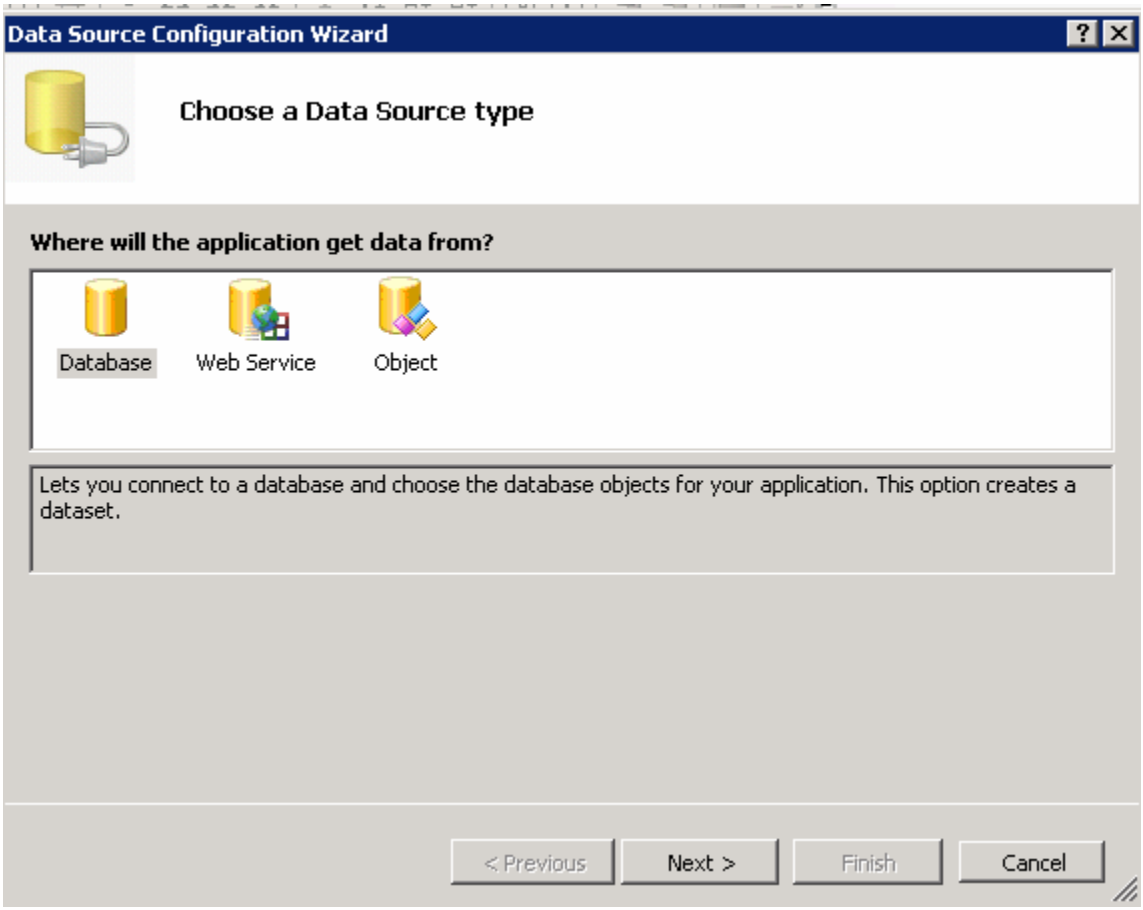

Figure 1.1. DataSource Configuration Wizard

Note: The data source configuration wizard supports three different types of data sources:

- Web Services,
- Database (any data that a .NET Data Provider) can attach to
- Business objects

- Select **Database** to connect to a database
- Click **Next**
- Click **New Connection** button

- [Optional step, might show depending on machine's last configurationlocal]
    - The **Choose Data Source** dialog appears
    - Select "**Microsoft SQL Server**" in the DataSource window
    - Click **Continue**
- The "**Add Connection**" dialog appears (see Figure 1.2)

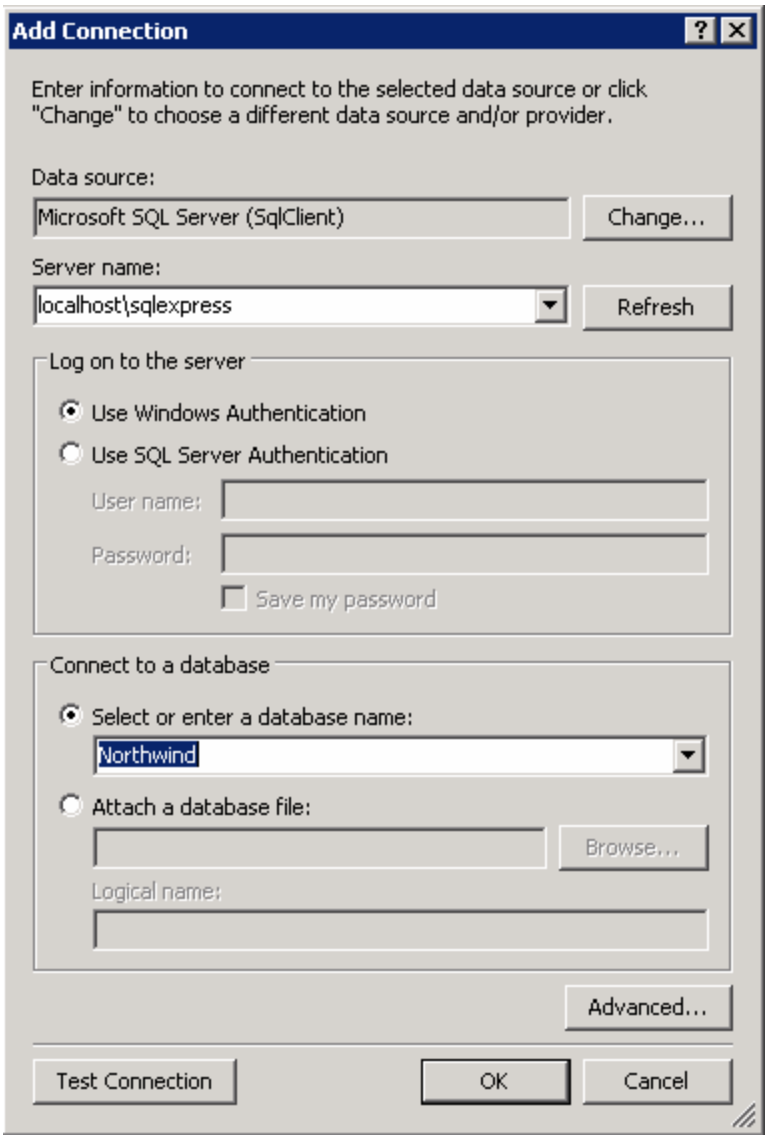

Figure 1.2 Add Connection dialog for configuring SQL Server connections

- Enter "localhost\SQLExpress" as **Server name**
- Leave the "Use Windows Authentication" as the option to **Log on to the server**
- Locate (or type) "Northwind" in the **Select or enter a database name dropdown**
- Click **OK** to close the dialog and go back to DataSource Configuration Wizard

- Click **Next**
- Leave the “Yes, save the connection as..” unchanged
- Click **Next**

Note: The designer might take a second or two to perform this operation.

- The “**Choose your data objects”** dialog displays (see Figure 1.3)

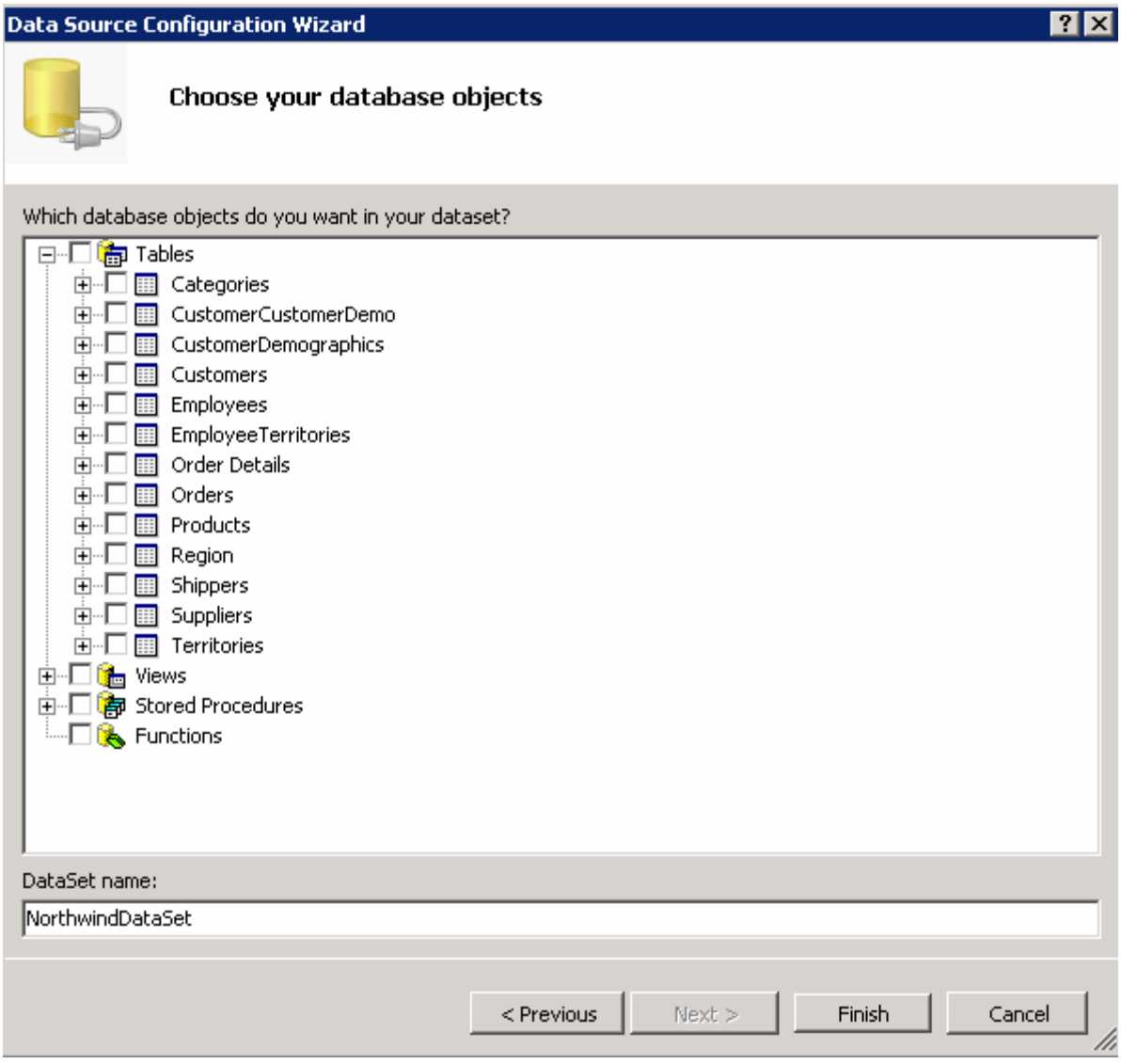

Figure 1.3 Database object selection dialog

- Click the following objects:
    - **Tables | Orders**
    - **Tables | Order details**
    - **Tables | Customers**
- Click **Finish**

Note: The designer might take a second or two to perform this operation.

Task 3 – Explore the recently added datasource

- Select **Data | Show Data Sources** menu command
- The data sources window should appear (as in Fig 1.4)

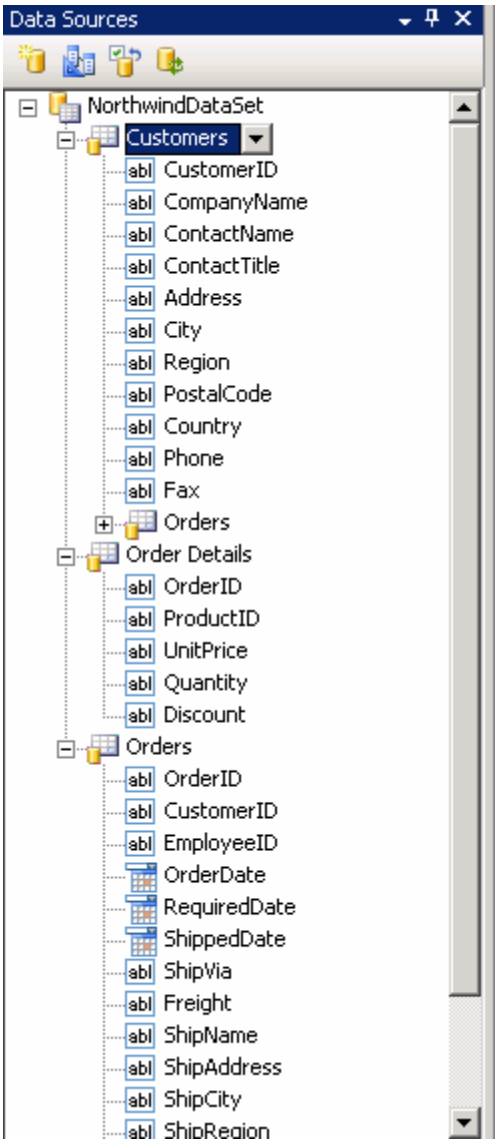

Figure 1.4 DataSources Window

- Notice that your tables selected have been wrapped into a single, strong typed DataSet
- Right Click in “NorthWindDataSet” at the very top, select **Edit DataSet with designer**
- The new DataSet designer appears (see Figure 1.5)

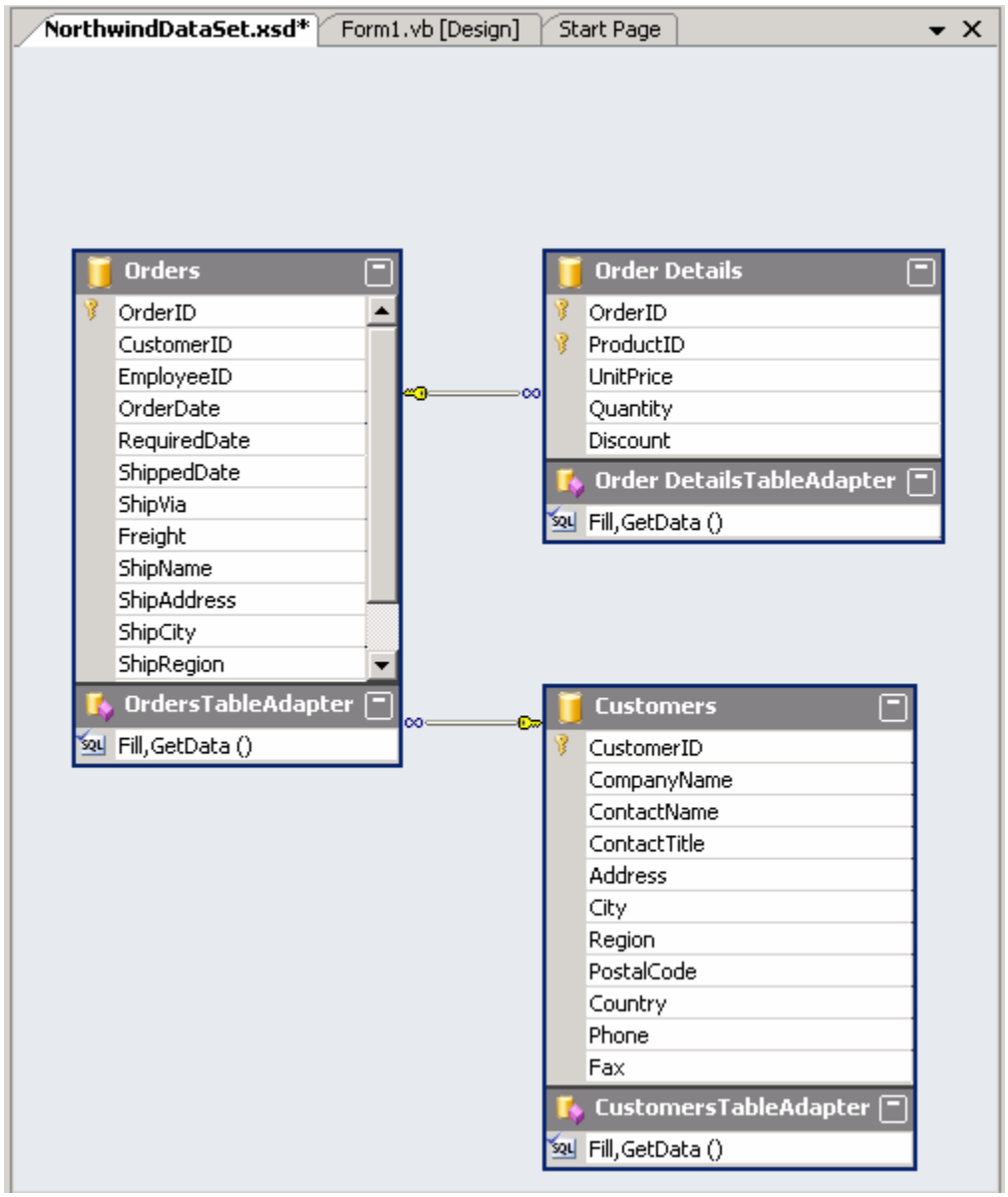

Figure 1.5 DataSet Designer

- Notice how the designer imported the relationships that existed in the database. In the DataSet designer you can get rid of these, or even add new relationships or Foreign Key constraints.
- Double click on the relationship from OrderDetails to Order

- A Relation dialog appears (see Figure 1.6)

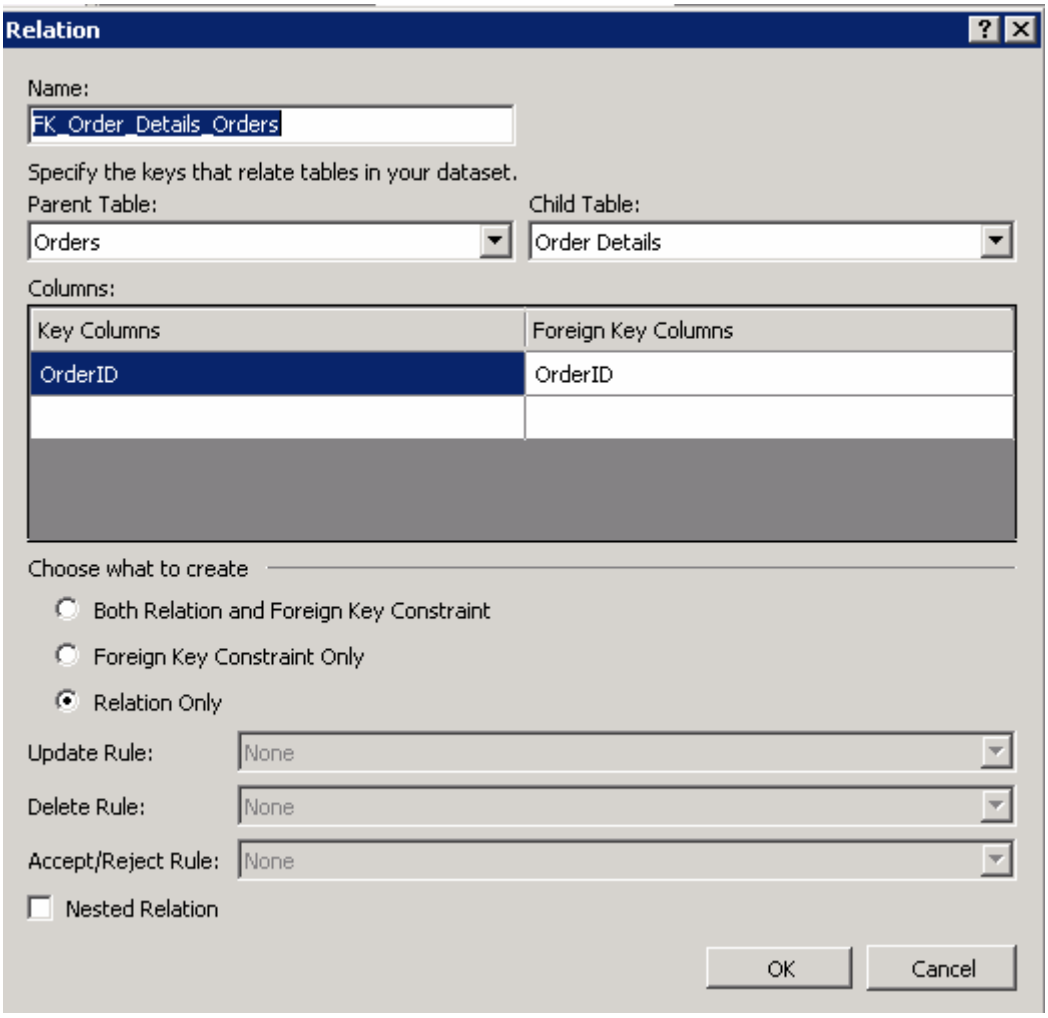

Figure 1.6 Relation Dialog in the DataSet designer

- Notice the designer has options to specify:
  - Parent/Child relations
  - Foreign key constraints
  - Custom rules for accepting/rejecting the constraints, etc.
- Click **Cancel** to discard any changes you made in the Relation Dialog (and close it)

  Note: With the new DataSet designer you have a lot of "design-time flexibility"; you do not have to write any code to represent complex relationships.
- Close the DataSet designer
- Go back to DataSource window
- Click on the drop-down arrow for the **Orders** table(see Figure 1.7)

- Select "**Details**" in the popup menu (see Figure 1.7)

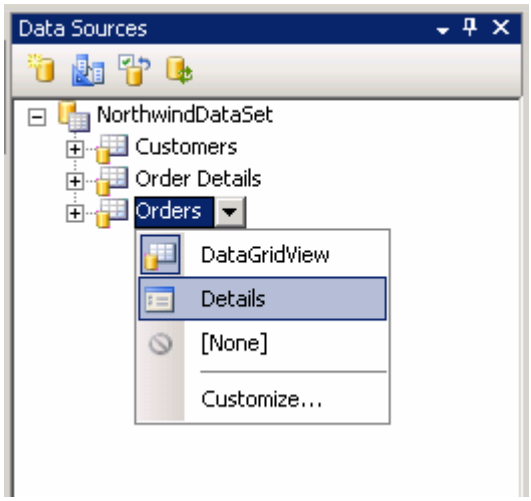

Figure 1.7 DataSource Window customization for details view

Because you selected "Details view" when you drag this table into a Windows Form, it will create a details view, by creating a control for every column in the table (refer to Task 3 in Exercise 2 in this same lab).

You can customize the type of the control created (for each column).

Exercise 2 – Databinding a Datasource to a form

Task 1 – Create a Master/Details View

- Drag the **Orders** table from the DataSources Window to Form1 in the Windows forms designer

Note: When dragging the table, a lot of work is done by the designer:

- An instance of the NorthwindDataSet is created, so it can fetch the data
- An OrdersBindingSource is created. A bindingsource acts as a generic proxy to a datasource.
- An OrdersBindingNavigator is also associated with the bind. The binding navigator is a Toolstrip control you can use to get VCR-like navigation of your data.
- A lot of labels and textbox controls are added to the form – one for each column in the Orders table. The reason controls were added instead of a datagridview is because we selected "Details" in the previous exercise.

o See Figure 1.8 for what the form should look like.

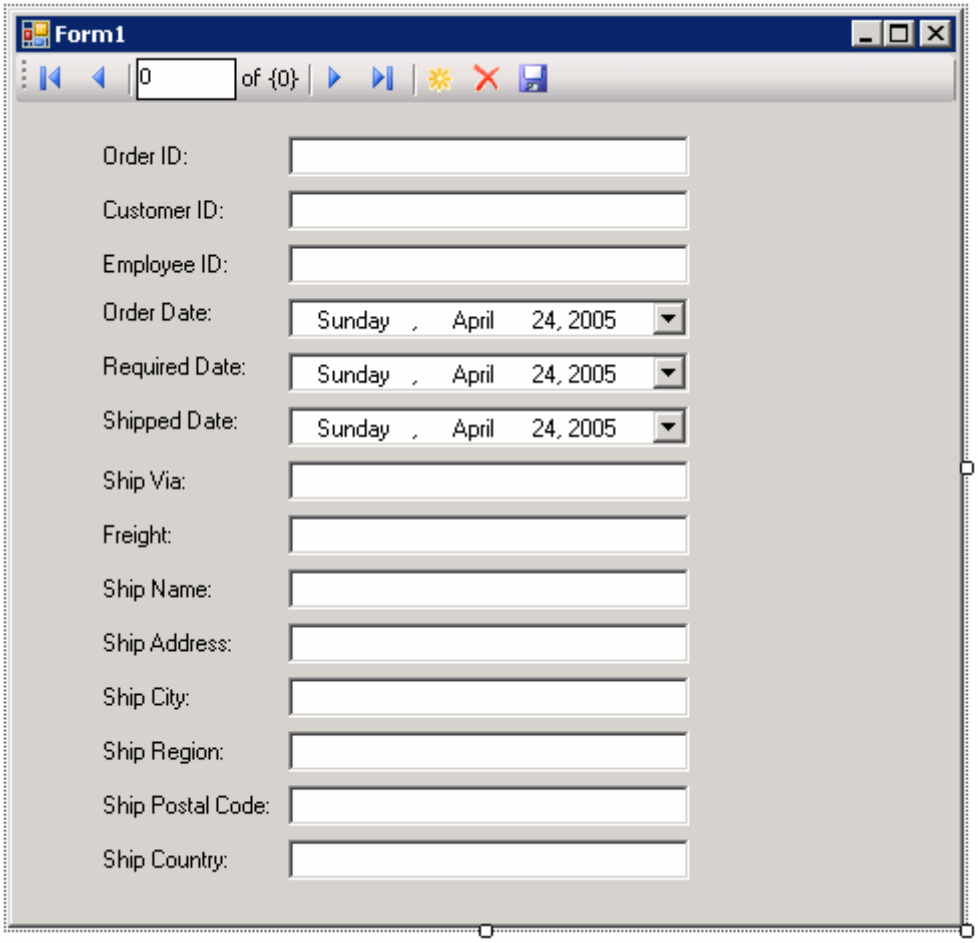

Figure 1.8 – Form after dragging Orders table from DS Window

- In the DataSources window, click the **Orders** node to expand it.
- Locate the **Order Details** node located inside the **Orders** table (represented as part of the table due to the parent/child between Orders and order details)
- Drag the **Order Details** table located inside the **Orders** table to the bottom of Form1

Note: Because you did not select "Details" for Order Details, this time a dataGridView is created. Your form should look like Figure 1.9

- Set the **Dock** Property of the order_DetailsDataGridView to **Bottom**

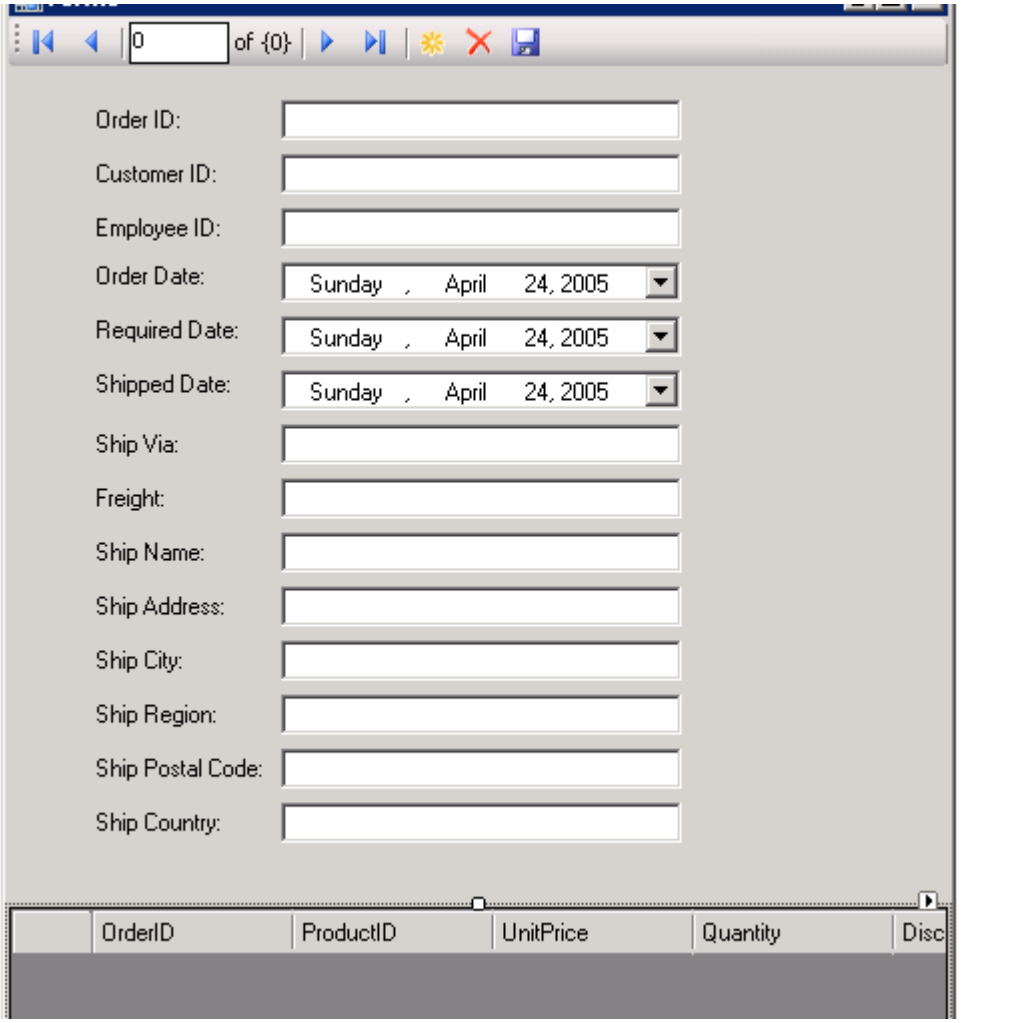

Figure 1.9 – Form1, after dragging two tables from Datasources window (no code written)

- Select **Debug | Start Debugging** menu command
- Form1 appears. Notice that:
    - When you change the position (or selected record) in the Orders this automatically updates the order details datagridview. Use the Binding Navigator to change positions in the orders table.
    - The BindingNavigator has the “new” , “delete” , “goto last” , “go to first”, etc. all of these buttons have been wired and have the functionality needed.

Note: We have accomplished a master/details view with navigation and performant strong-typed databinding without writing a single line of code.

- Close the application or stop debugging

Task 2 – Customize the Detail controls to readonly types

In the previous task, we dragged the Details View and it defaulted to TextBox controls for all the columns. Let's now customize this.

- In DataSources window, Click on the dropdown arrow next to OrderId, in the **Orders** table (see Figure 1.9)

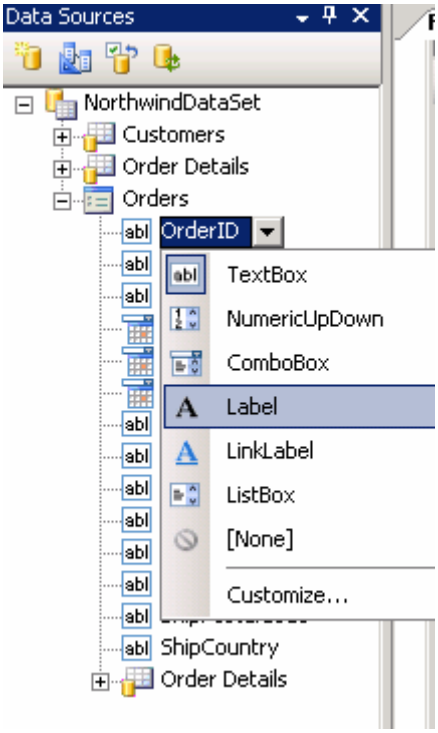

Figure 1.9 – Customizing the control type for details view

- Select Label in the context menu as the type for OrderId
- Delete **orderIdLabel** and **orderIdTextBox** from Form1
- From the datasources window, drag & drop OrderId from the **Orders** Table into Form1; positioning so it aligns with the other details controls (and replacing the two controls you just deleted).
- Select **Debug | Start Debugging** and preview the changes
- DataLab application launches
    - Notice how orderId is now a label instead of a textbox; you can use the same procedure to customize the other columns.
- Close the application (or select **Debug | Stop Debugging** menu command)

Exercise 3 – Customizing a DataGridView

Task 1 – Adding UnBound columns to the view

- RightClick on order_DetailsDataGridView
- Select **Add Column** from the popup menu
- The Add Column Dialog appears (see Figure 1.10)

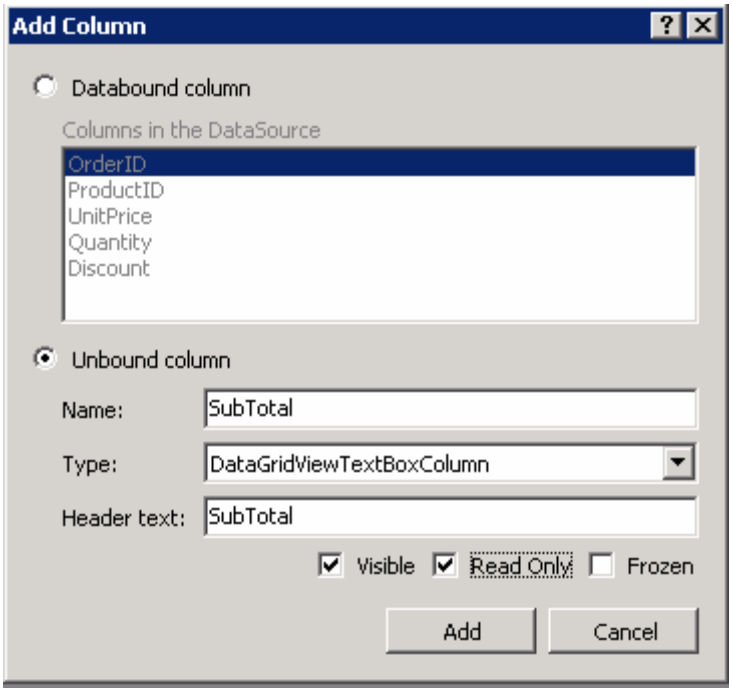

Figure 1.10 Add Column dialog

- Click Unbound Column

Note: Unbound columns are columns not bound to data. Their values are provided dynamically via eventing in the Gridview.

- Enter “SubTotal” for Name
- Select **DataGridViewTextBoxColumn** for Type
- Enter “SubTotal” for Header Text
- Check **Visible**
- Check **Read Only**
- Click **Add** to save to add this column
- Click **Close** to dismiss the Add Column Dialog
- Click OK to close the **Edit Columns** dialog

- Select the order_detailsDataGridView, set the following properties:
    - VirtualMode = True

Note: VirtualMode has to be True when using Unbound columns. If you do not set Virtual Mode = true, the CellValueNeeded (below) will not be called.

Task 2 – Customizing Cell formats

- RightClick on order_DetailsDataGridView
- Select **Edit Columns** from the popup menu
- The **Edit Columns** dialog appears
- Select **UnitPrice** to edit its properties
    - Locate the **DefaultCellStyle** property
    - Double Click **DefaultCellStyle**
    - **CellStyle builder** dialog appears
        - Set **Format** = "C2"
        - Click **OK** to close CellStyle builder dialog

Note: C2 Format will format the contents of this cell as Currency.

- Click **OK** to close the Edit Columns dialog

Task 3 – Populating the Unbound columns

- Click on the Events icon in the properties for order_DetailsDataGridView
- Locate **CellValueNeeded** event

Note: CellValueNeeded will be call once per each Unbound cell that needs a value. Inside this event, we can set the value that needs to be displayed.

In this case SubTotal = UnitPrice * Quantity

- Double-Click on **CellValueNeeded** to add a handler

- Add the following code:

// C#

```
DataGridView grid = sender as DataGridView;
try
{
// There are more efficient ways to do this.. coded this way for demo simplicity
      decimal subtotal =
      (decimal)grid["UnitPrice", e.RowIndex].Value *
      (short )grid["Quantity", e.RowIndex].Value ;
      e.Value = subtotal.ToString("C2");
} finally { }
```

'VB:

```
Dim grid As DataGridView = CType(sender, DataGridView)
        Try
'There are more efficient ways to do this.. coded this way for demo simplicity
            Dim subTotal As Decimal = _
            CType(grid("UnitPrice", e.RowIndex).Value, Decimal) * _
            CType(grid("Quantity", e.RowIndex).Value, Integer)
            e.Value = subTotal.ToString("C2")
        Finally

        End Try
```

Task 4 – Handling CellPainting to highlight cells

Note: CellPainting is called before the cell will be painted; handle this event to highlight any order details with discounts greater than 10%

- Click on the Events icon in the properties for order_DetailsDataGridView
- Locate **CellPainting** event
- Double-Click on **CellPainting** to add a handler
- Add the following code in the handler:

// C#

```
DataGridView grid = (DataGridView)sender;
            // Make sure it is the Discount column & it is not the header
            if (grid.Columns["Discount"].Index == e.ColumnIndex
                && e.Value != null
                && e.RowIndex != -1 )
            {
                float discount = (float) e.Value;
                if ( 0.100f < discount )
                {
                    e.CellStyle.BackColor = Color.Red;
                }
            }
```

'VB:

Task 5 – Removing unneeded columns & freezing columns

- Right-Click on order_detailsDataGridView
- Select **Edit Columns**
- Select the **OrderId** column in the selected columns List
- Click Remove

Note: Since you are viewing the details for an order, this column is already showing on the top of our form.

- Select **ProductId** in the selected columns List, set the following properties:
    - Frozen = True
    - AutoSizeMode = DisplayedCells

Note: The frozen property is used to make sure this column always shows. When the grid is shrinked, all the other columns will be resized first, and this frozen column will be kept constant so it always shows.

The AutoSizeMode is set to Displayed Cells so the grid calculates the column width based on what is showing. Displayed cells are all the cells in the screen, including the header

- Click **OK** to close the Edit Columns Dialog

Task 6 – Preview all the changes

- Select **Debug | Start Debugging** menu command
- DataLab application launches
- Notice the UnitPrice and SubTotal columns are formatted as currency. The formatting for UnitPrice was set in the designer (no coding needed); the formatting for SubTotal was done in the CellValueNeeded event –where its value was provided-.
- Shrink the form (horizontally)
    - Notice the form stops shrinking at ProductId. This column is frozen and will not shrink.
- Click on the Discount column to enter a value. Enter “0.5”
    - Notice when you Tab to the next column, the Discount column changes its BackColor to Red. It is because it has a discount > 0.10
- **Close** the application

Exercise 4– Binding to business objects

Task 1 – Open startup project

- Open Visual Studio 2005
- Select **File | Open | Project/Solution** menu command
- Enter the **File Name**
- For VB,

“C:\Microsoft Hands-On-Lab\HOL-CLI03\Source\Exercises\Starter\VB\BuildingMaintenance”

For C#,

“C:\Microsoft Hands-On-Lab\HOL-CLI03\Source\Exercises\Starter\CS\BuildingMaintenance”

- Select “BuildingMaintenance.sln”
- Click **Open**

Note: The solution has two projects:

BuildingMaintenance is an empty Windows application.

Facilities is a class library with a few business objects. With the exception of the campus class implementing INotifyPropertyChanged, the classes are very basic. It is just a few properties per class to describe an object (such as a Building, a Campus, or a Company).

Task 2– Create a business object datasource

- Select **Data | Add New DataSource** menu command
- Select **Object** as the datasource type
- Click **Next**
- Expand the **Facilities** Namespace (see Figure 1.13)

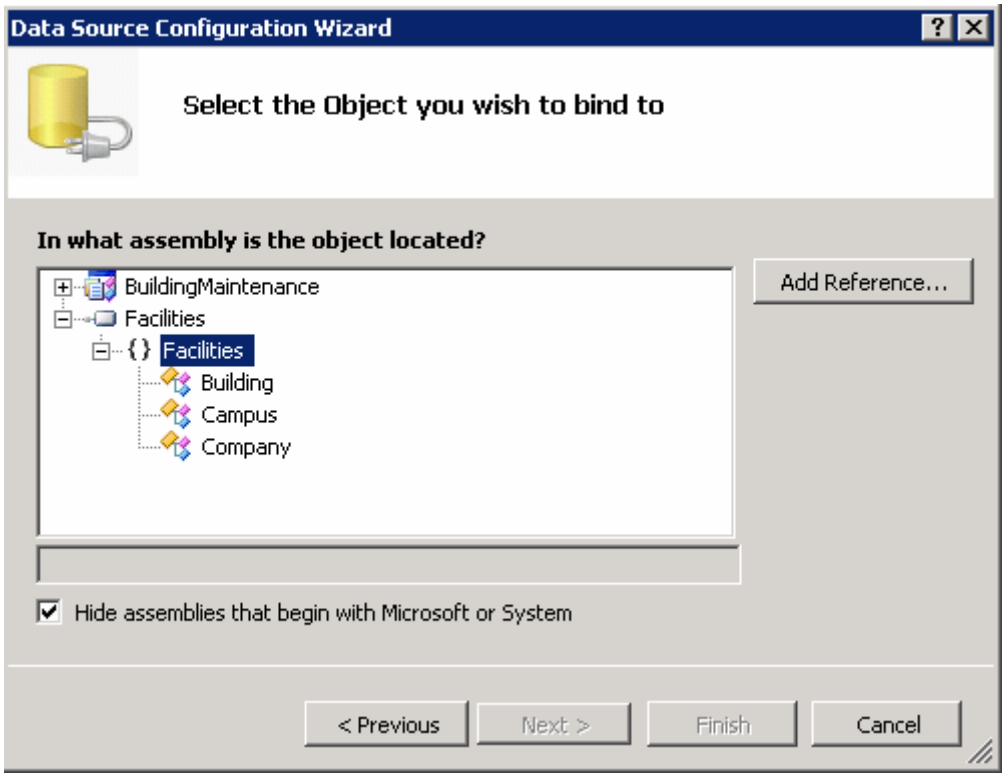

Figure 1.13 - DataSource Configuration Wizard for business objects

- Select **Campus**
- Click **Next**
- Click **Finish**

Task 3 - Create Master/Details from using business object datasource

- Open BuildingMaintenance.Form1 in the forms designer
- Select **Data | Show DataSources** menu command
- The datasources window appears (see Figure 1.14)

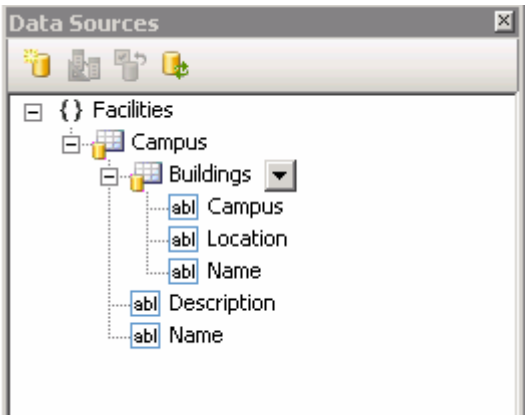

Figure 1.14. DataSources window showing object Datasource

- Click on the Down arrow next to **Campus**
- Select **Details** from the popup-menu (see Figure 1.15)

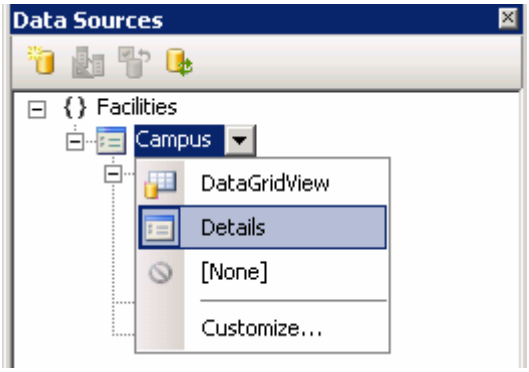

Figure 1.15 Selecting Detailed View for Campus object

- Drag **Campus** from **DataSources Window** to **Form1**
- Notice that a BindingSource and BindingNavigator were added to the form; along with Details Controls for the Campus object. Exact same behavior than earlier exercises when binding to databases.
- Drag the **Buildings** property from the DataSources window to the bottom of Form1
- Notice a DataGridview was created, with one column for each public property in the Building Class.

- Form1 should look like Figure 1.16

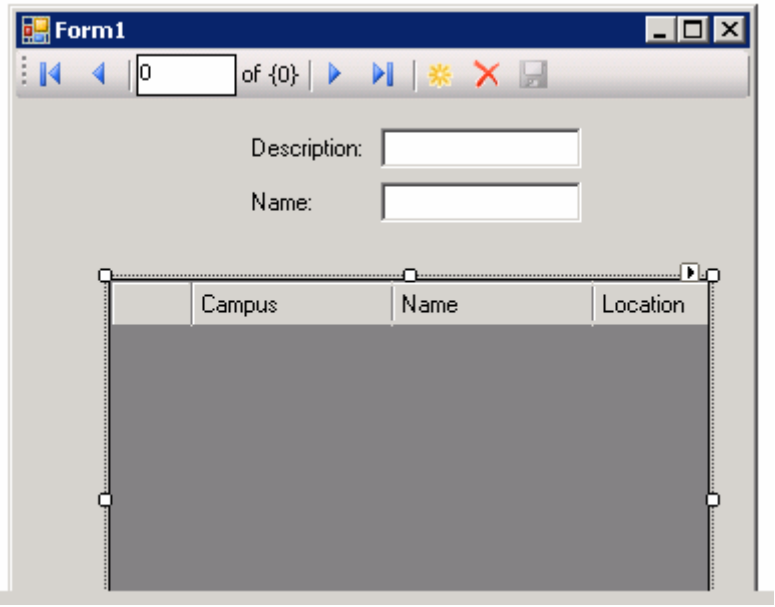

Figure 1.16. Form1 after two drag & drops from the datasources window

- Double-Click on **Form1** to add a Load event handler
- The following code in the handler will bind our form to an instance of a Company's facilities:

// C#

```
this.campusBindingSource.DataSource =
        Facilities.Company.Microsoft  ;
```

'VB

```
CampusBindingSource.DataSource = Facilities.Company.Microsoft
```

- Select Debug | Start Debugging menu command
- BuildingMaintenance launches
- Notice the details and the datagrid are populated. The BindingNavigator works too. You can move forward and back, and can delete a record.

Note: The new button in the binding navigator is disabled because the Campus class does not have a default constructor.

- Close the application (or select **Debug | Stop Debugging**)

Task 4 – Update business object

- Add a button to the campusBindingNavigatorToolstrip (see Figure 1.17)

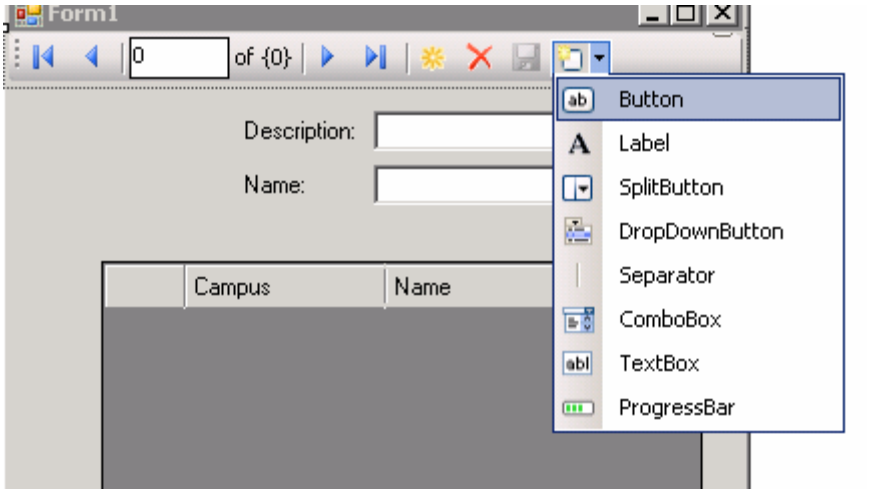

Figure 1.17 – Customizing a BindingNavigator with an extra button

- Set the following properties in the just added toolstripButton1
  - **(Name)** = toolstripButton1
  - **BackColor** = "Desktop"
  - **DisplayStyle = "Text "**
  - **ForeColor =** "ActiveCaptionText"
  - **Text =** "UpdateCampusName"
- Double-Click on toolStripButton1 to add a click handler

Note: In the clickhandler, we will update the name of one of the campuses. The goal is to show that an update to a business object will be reflected in the User Interface because the UI is databound.

- Enter the following code in `toolStripButton1_Click`:

```
// C# :
Facilities.Company.Microsoft[0].Name = "Updated";
‘ VB:
Facilities.Company.Microsoft(0).Name = "Updated"
```

- Select **Debug | Start Debugging** menu command to preview the changes
- The app launches
- Click the **Update Campus Name** button

- Notice the user interface immediately changes the name to Updated (Figure 1.18)

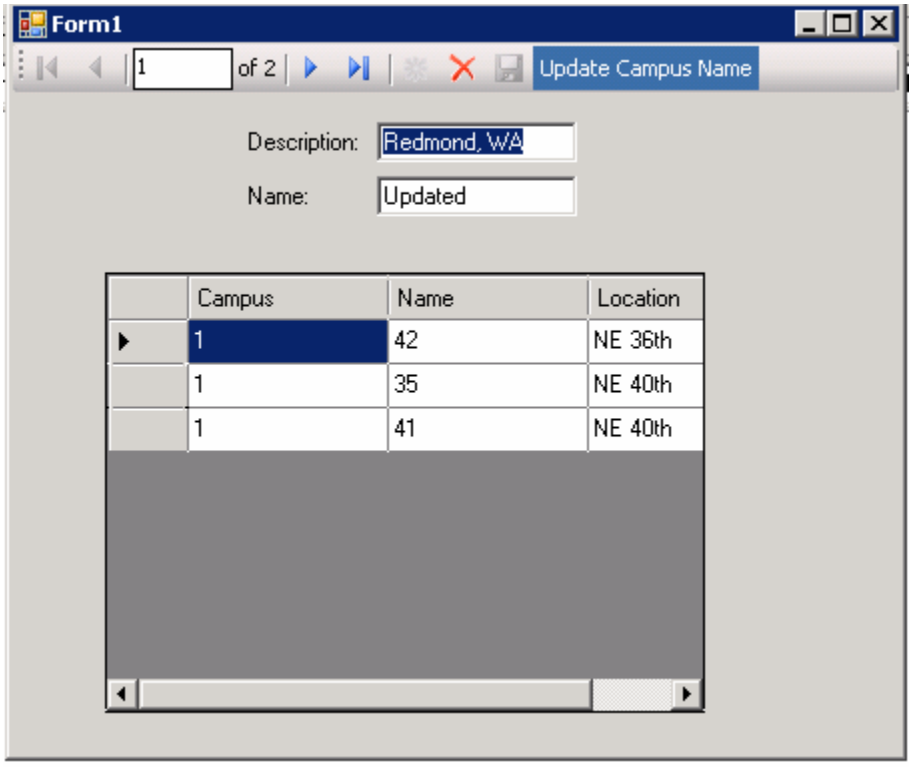

Figure 1.18 – A databound business object triggered the UI to update

Note: The Campus class notifies the UI controls that a property has changed via the INotifyPropertyChanged interface – exposing the PropertyChanged event-.

For any business object that will be wired for a simple (1 to 1) binding, the interface needs to be implemented.

Lab Summary

In this lab you performed the following exercises.

- Create a datasource and add it to the Datasources window
- Databinding a DataSource to a form
- Customizing a DataGridView
- Binding to business object

In this lab, you explored some of the new databinding features in windows forms 2.0. You used the productivity enhancements of the datasources windows to generate master/details for data coming from both a database and a business object. You accomplished both of these with out having to write any code.

We also explored a few of the options in the new DataGridView control. The datagridview allows you to add Unbound columns and provide the value to be displayed via the CellValueNeeded event. It also allows eventing to format, cell paint, and handle events. You explored these features in this lab.