
# Hands-On Lab

## Lab Manual

## Developing Office Solutions with Microsoft Visual Studio 2005 Tools for the Microsoft Office System (Part 2)

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

# Contents

# Setting Up the Labs

To work through the lab exercises, you must install:

- Microsoft® Windows® 2000 or later
- Microsoft Visual Studio® 2005 Beta 2
- Microsoft Visual Studio 2005 Tools for the Microsoft Office System Beta 2 (included with Visual Studio 2005 Beta 2)
- Microsoft Office Professional Edition 2003 with Microsoft Excel® Analysis ToolPak add-in installed
- Microsoft SQL Server™ 2000 (or higher) or MSDE

  **Note** When applicable, instructions in the lab manuals refer to files by a full path; it is assumed that the download files are extracted to the root of C: drive. The files needed to complete the labs are in a folder named C:\VSTO2005\Files\Excel.

# Lab 1 - Data and Controls in an Excel Workbook

The objective of this lab is to demonstrate how to connect to a SQL Server data source for the purpose of binding controls in the Excel workbook to the data source. You will also use Windows Forms controls on the workbook to navigate records in the data source.

Estimated time to complete:

- Exercise 1: Connect and Bind Data – 15 minutes
- Exercise 2: Use Windows Forms Controls – 15 minutes

## Exercise 1: Connect and Bind Data

### Create a new Excel Workbook project

1. On the **File** menu, click **New Project**.
2. In the list of **Project Types**, expand **Visual Basic** and click **Office**.
3. Select **Excel Workbook** in the list of project **Templates**.
4. Type **AssetAllocations** in the project **Name** box. If you have a Location field, type c:\VSTO2005\Labs. click **OK**. The **Visual Studio Tools for Office Project Wizard** appears.
5. In the wizard, click **Copy an existing document**.
6. Click the Browse button to navigate to **C:\VSTO2005\Files\Excel\Asset Allocations.xls** for the file path.
7. Click **Finish**. If this is your first Visual Studio Tools for Office 2005 project for Microsoft Excel, you may see the dialog box shown in **Figure 1**. If you do, click **OK** to acknowledge and close the dialog box.

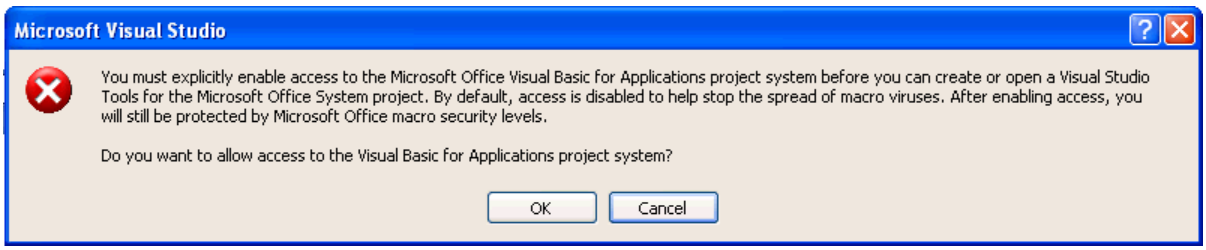


**Figure 1. You receive a security notice the first time you create a Visual Studio Tools for Office project for Excel.**

8. On the **File** menu, select **Save All.**
9. If you didn't have a Location field on step 4, the **Save Project** dialog box appears. n the project **Location** box, type **C:\VSTO2005\Labs** and click **Save**.

### Create the sample database using a SQL script

This lab uses a SQL Server database included in the lab files. You will use a SQL script to add the sample database to your instance of SQL Server. The name of the new database is VSTO2Lab.

1. On the Windows **Start** menu, select **Run**. Type **cmd** and click **OK**. A Command window opens.
2. Change to the folder where the lab files reside:

```
cd C:\VSTO2005\Files\Excel
```

3. Execute the batch procedure:

```
osql.exe –n -E -i VSTO2005ExcelLab.sql
```

   You will receive a message that the database is successfully restored.

4. Close the Command window.

### Add a data source to your project

1. On the **Data** menu, select **Show Data Sources**. The **Data Source** window appears.
2. Click **Add New Data Source** in the **Data Source** window. You see the **Data Source Configuration** window.
3. Select **Database** for the data source type and click **Next**.
4. Click **New Connection**. If you have not created a data connection before, the **Choose Data Source** dialog box appears. If you have created a data connection, you see the **Add Connection** dialog box shown in **Figure 3** instead and you can skip to step 7.
5. In the **Choose Data Source** dialog box, select Microsoft SQL Server, as shown in **Figure 2**.

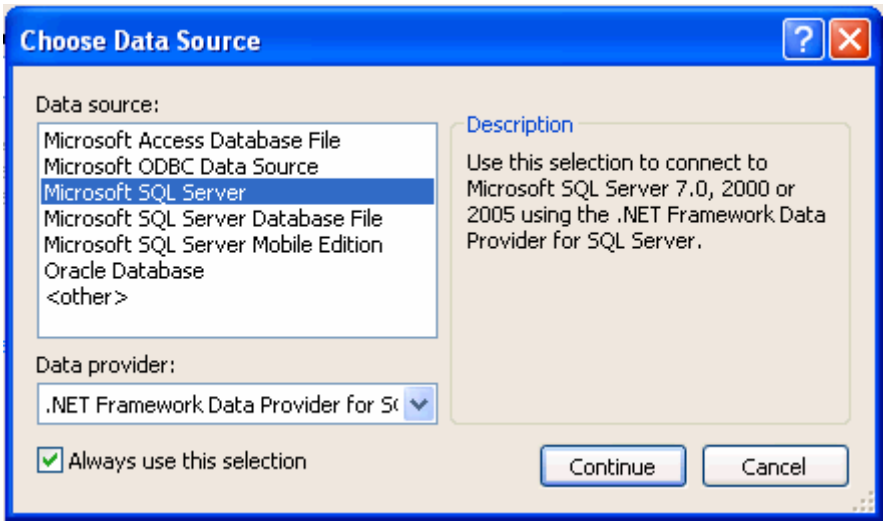


**Figure 2. The Microsoft SQL Server data source is used for Microsoft SQL Server 7.0 and up.**

6. Click **Continue**. You see the **Add Connection** dialog box shown in **Figure 3**.

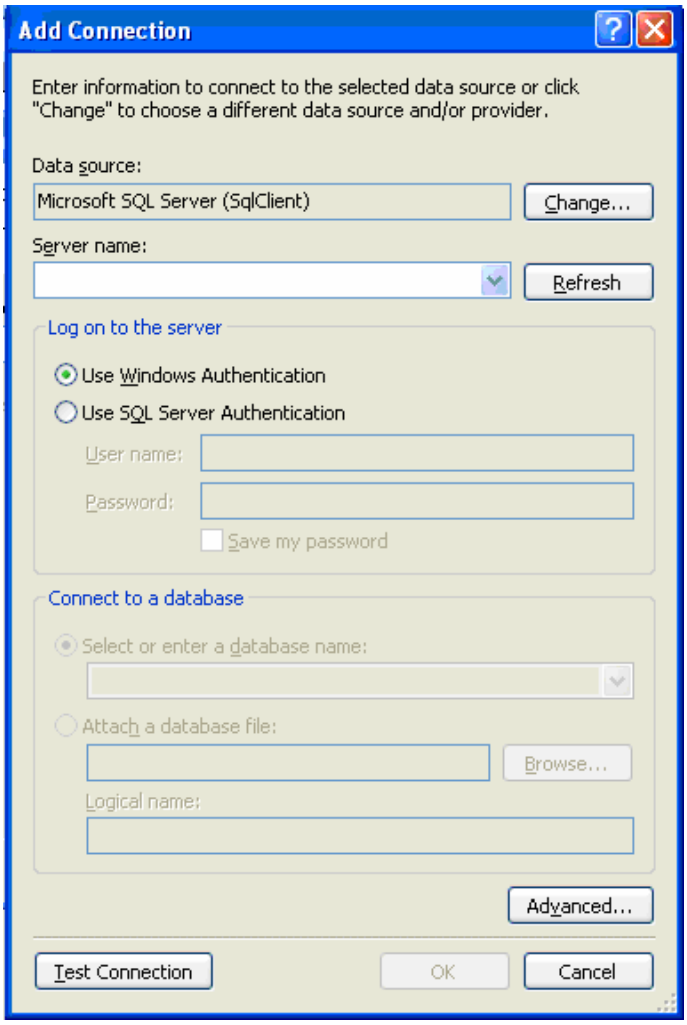


**Figure 3. The Add Connection dialog box prompts you for information about the new connection.**

7. Create the new connection:
   - If the **Data Source** is not **Microsoft SQL Server (SqlClient)**, click **Change** and choose the SQL Server data source.
   - In **Server Name**, type **(local)**.
   - Select **Use Windows Authentication** for the authentication mode.
   - In the **Connect to a database** section, select **VSTO2005Lab** in the **Select or enter a database name** list.
   - Click **Test Connection** to confirm that the connection is valid and then click **OK** to add the new connection. The new connection has the default name **[YourServerName].VSTO2005Lab.dbo**.
8. Click **Next**.

9. Save the connection using the default name, **VSTO2005LabConnectionString**.
10. Click **Next**.
11. In the list of database objects, select the **Tables** and **Views** check boxes and click **Finish**.

### Add a relation to the data source

1. Right-click **VSTOLab2005DataSet** in the **Data Sources** window and select **Edit DataSet with Designer** on the context menu.
2. On the **Data** menu, select **Add** and choose **Relation**. The **Relation** dialog box appears.
3. Specify the following for the new relation:
   - **Name** is **Customers_PortfolioView**
   - **Parent Table** is **Customers**
   - **Child Table** is **PortfolioView**
   - **PK Fields** is **CustomerID**
   - **FK Fields** is **CustomerID**

   The completed dialog box looks like **Figure 4**.

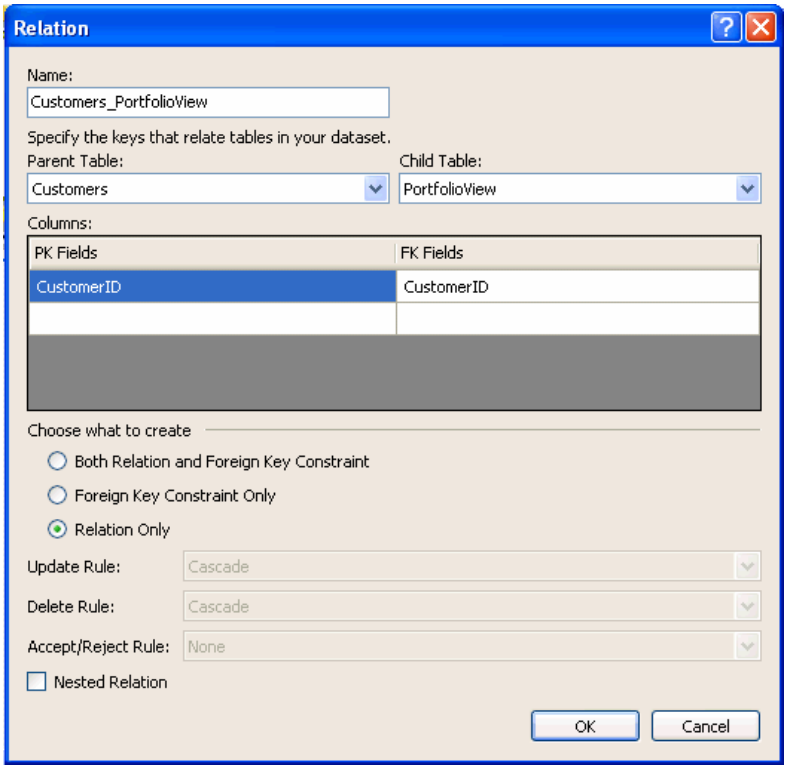


**Figure 4. The default selections should match the choices you need for this exercise.**

4. And click **OK** to add the new relation.

5. The **Data Source** window now looks like **Figure 5**.

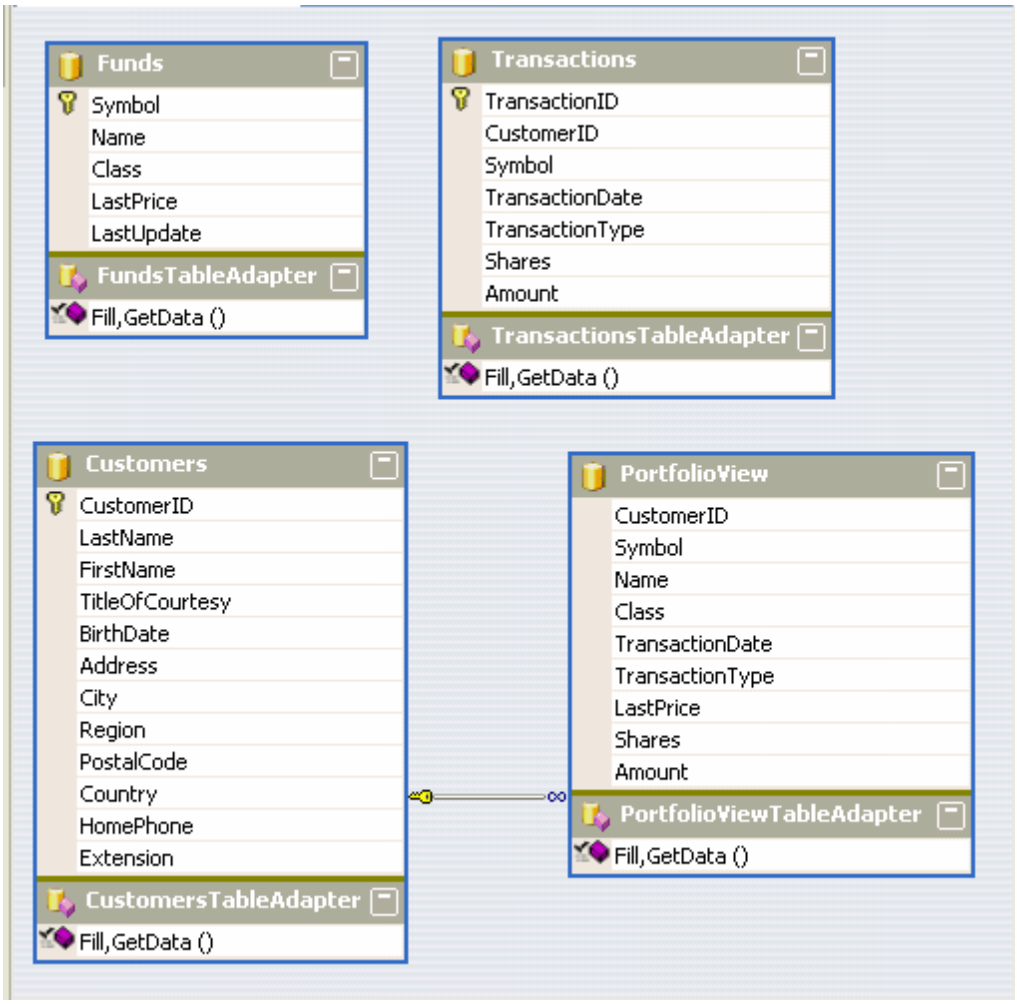


**Figure 5. You can see the relation you just created as a line between Customers and PortfolioView.**

### Bind NamedRange controls to fields in the data source

1. In **Solution Explorer**, right-click **Sheet1.vb** and select **View Designer** on the context menu.
2. In the **Data Sources** window, expand the **Customers** table.
3. Click **CustomerID** under the **Customers** table and drag it onto cell B2 of Sheet1.
4. Click **FirstName** under the **Customers** table and drag it onto cell E2 of Sheet1.
5. Click **LastName** under the **Customers** table and drag it onto cell E3 of Sheet1.
6. Click **BirthDate** under the **Customers** table. Click the drop-down arrow to the right of the **BirthDate** field and select **NamedRange** from the list. Then, click **BirthDate** and drag it onto cell E4 of Sheet1.

**Complex-bind a List control to fields in the data source**

1. In the **Data Sources** window, click to select **PortfolioView** and drag it onto cell A7 of Sheet1. A new list named PortfolioViewListObject is created. It contains one column for each field in **PortfolioView**. Leave **PortfolioViewListObject** selected.
2. On the **Data** menu, click **Microsoft Office Excel Data**, click **Filter** and select **AutoFilter** to remove the AutoFilter drop-down lists from the list object.
3. In the **Properties** window, click the drop-down arrow next to the **DataSource** property. Expand **CustomersBindingSource** in the list and select **Customers_PortfolioView**.
4. The Asset Allocations workbook will use named ranges in the **List** for calculations. Name the Sheet1 range B7:B8 **Symbol** by following these steps:
   - Select the range **B7:B8**.
   - In the **Name Box** shown in **Figure 6**, type **Symbol** and press **Enter**.

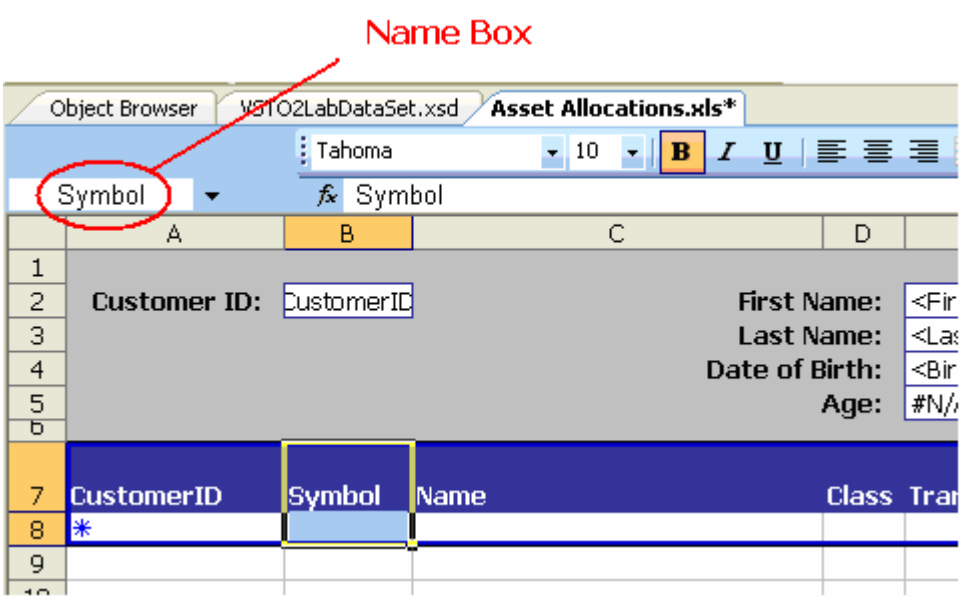


**Figure 6. Use the Name Box to name a range.**

5. Repeat the steps to name the ranges described in **Table 1**.

| Name | Range |
|---|---|
| Symbol | Sheet1!$B$7:$B$8 |
| Class | Sheet1!$D$7:$D$8 |
| LastPrice | Sheet1!$G$7:$G$8 |
| Shares | Sheet1!$H$7:$H$8 |
| Amount | Sheet1!$I$7:$I$8 |

**Table 1. Named ranges needed for the Asset Allocation workbook.**

### Checkpoint

1. On the **File** menu, select **Save All**.
2. On the **Debug** menu, select **Start** to build and run the project. The workbook Asset Allocations.xls opens in Excel as shown in **Figure 7**.

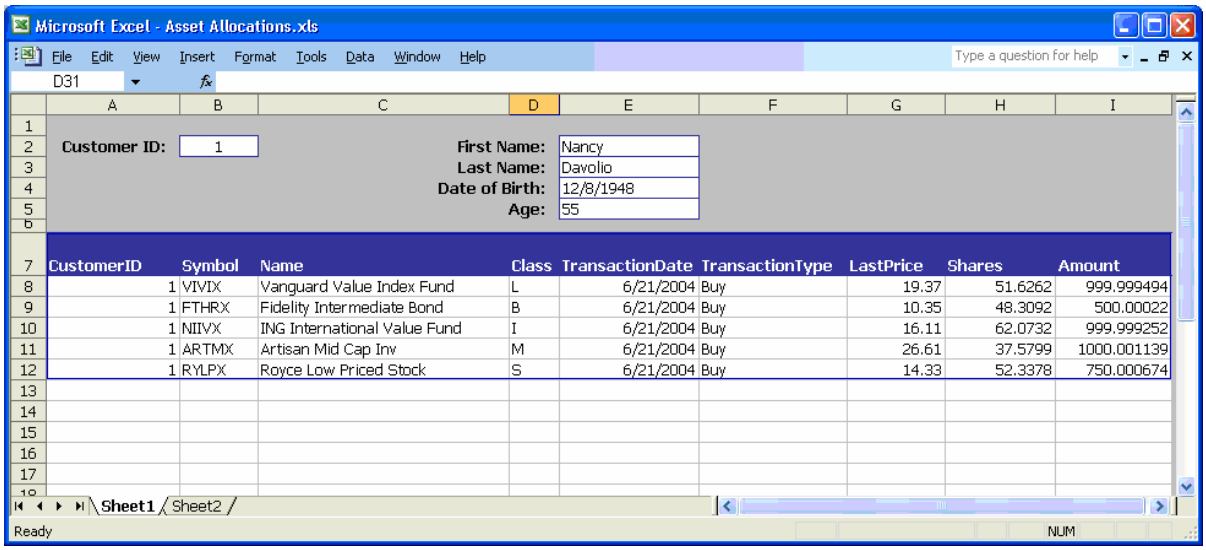


**Figure 7. Ranges in the workbook are now bound to data in the SQL Server tables.**

The data for the first record in the **Customers** table is displayed in the bound **NamedRange** controls; additionally, the corresponding details from the **PortfolioView** view appear in the **List** control.

3. Exit Excel without saving the workbook.

## Exercise 2: Use Windows Forms Controls

### Add Windows Forms controls to the workbook for navigating the data source

1. If the Toolbox window is not visible, then select the **View** menu and choose **Toolbox**.
2. Select a **Button** control from the **Toolbox** window and drag it to cell **A4** of Sheet1. **Button1** is added to the worksheet.
3. Select another **Button** control from the **Toolbox** window and drag it to cell **B4** of Sheet1. **Button2** is added to the worksheet.
4. Use the **Properties** window to change the properties of the buttons as shown in **Table 2**.

| Control | Property | Value |
|---|---|---|
| Button1 | (Name) | btnPrevious |
| | Text | << |
| | TextAlign | MiddleCenter |

| Control | Property | Value |
|---|---|---|
| Button2 | (Name) | btnNext |
| | Text | >> |
| | TextAlign | MiddleCenter |

**Table 2. Property settings for button controls.**

5. In **Solution Explorer**, right-click **Sheet1.vb** and select **View Code** on the context menu.
6. Observe that the designer has automatically created a class named **Sheet1** for the worksheet. Additionally, note that there are two lines of code already in the **Startup** event for Sheet1; the designer added this code when you bound the **NamedRange** and **List** controls to fields in the data source:

```
    Private Sub Sheet1_Startup(ByVal sender As Object, _
     ByVal e As System.EventArgs) Handles Me.Startup
        'TODO: Delete this line of code to remove the default AutoFill
        'for 'VSTO2005LabDataSet.PortfolioView'.
        If Me.NeedsFill("VSTO2005LabDataSet") Then
            Me.PortfolioViewTableAdapter.Fill( _
             Me.VSTO2005LabDataSet.PortfolioView)
        End If
        'TODO: Delete this line of code to remove the default AutoFill
        'for 'VSTO2005LabDataSet.Customers'.
        If Me.NeedsFill("VSTO2005LabDataSet") Then
            Me.CustomersTableAdapter.Fill(Me.VSTO2005LabDataSet.Customers)
        End If

    End Sub
```

7. In the **Class Name** list, select **btnPrevious**.
8. In the **Method Name** list, select **Click**.
9. Add code to the **Click** events of **Button1** to navigate the data source:

```
Private Sub btnPrevious_Click(ByVal sender As Object, _
 ByVal e As System.EventArgs) Handles btnPrevious.Click
    'Move to the previous record if not already at the beginning of the list
    If Me.CustomersBindingSource.Position > 0 Then
       Me.CustomersBindingSource.MovePrevious()
    Else
       MessageBox.Show("The current record is at the beginning of the list.", _
          "Asset Allocation", MessageBoxButtons.OK, MessageBoxIcon.Information)
    End If
End Sub
```

10. Follow the same process to add code to the **Click** event of **btnNext**:

```
Private Sub btnNext_Click(ByVal sender As Object, _
  ByVal e As System.EventArgs) Handles btnNext.Click
    'Move to the next record if not already at the end of the list.
    If Me.CustomersBindingSource.Position < _
      Me.CustomersBindingSource.Count - 1 Then
      Me.CustomersBindingSource.MoveNext()
    Else
      MessageBox.Show("The current record is at the end of the list.", _
          "Asset Allocation", MessageBoxButtons.OK, MessageBoxIcon.Information)
    End If
End Sub
```

### Checkpoint

1. On the **File** menu, select **Save All**.
2. On the **Debug** menu, select **Start** to build and run the project. The Asset Allocations workbook opens in Excel.
3. Click the buttons to move between the records in the data source. Notice that the details in the **List** object update when the bound fields for the parent table (**Customers**) change.
4. Exit Excel without saving changes.

# Lab 2 – Using Controls in the Actions Pane

In this lab, you will create an actions pane for an Excel solution using a Windows Forms control and a user control that you will create.

Estimated time to complete:

- Exercise 1: Add a Windows Forms Control to the Actions Pane - 5 minutes
- Exercise 2: Add a User Control to the Actions Pane – 5 minutes

## Exercise 1: Add a Windows Forms Control to the Actions Pane

### Create a new Excel Workbook project

1. On the **File** menu, select **New Project**.
2. In the list of **Project Types**, expand **Visual Basic** and choose **Office**.
3. Select **Excel Workbook** in the list of project **Templates**.
4. Type **DebtConsolidation** in the project **Name** box, and C:\VSTO2005\Labs on the location box (if you have it) and click **OK**. The **Visual Studio Tools for Office Project** wizard appears.
5. In the wizard, click **Copy an existing document**.
6. Type **C:\VSTO2005\Files\Excel\Debt Consolidation.xls** for the file path and click **Finish**.
7. On the **File** menu, click **Save All**.
8. If you didn't get a location field on step 4, A save dialog will appear, In the project **Location** box, type **C:\VSTO2005\Labs** and click **Save**.

### Set the size and position for the actions pane

1. In **Solution Explorer**, right-click **ThisWorkbook.vb** and select **View Code**. The code module for the workbook appears; observe that the workbook code-behind is encapsulated in a class named **ThisWorkbook** and that event handlers for **Startup** and **ShutDown** already exist.
2. Add code to the **ThisWorkbook_Startup** event that will size and position the actions pane when the workbook loads and initialize the worksheet view:

```
Private Sub ThisWorkbook_Startup(ByVal sender As Object, _
  ByVal e As System.EventArgs) Handles Me.Initialize

    With Me.Application.CommandBars("Task Pane")
        .Width = 300
        .Position = Microsoft.Office.Core.MsoBarPosition.msoBarLeft
    End With
    Dim labelControl As New Label
    labelControl.Text = "TO DO: Add Controls Here"
```

```
    Me.ActionsPane.Controls.Add(labelControl)
    Globals.Sheet2.Select()
    Me.Application.ActiveWindow.DisplayHeadings = False

End Sub
```

### Checkpoint

1. On the **File** menu, select **Save All**.
2. On the **Debug** menu, select **Start** to build and run the project. The workbook Debt Consolidation.xls opens in Excel.
3. The actions pane is displayed with a label, as shown in **Figure 8**.
4. Close the workbook without saving changes and quit Excel.

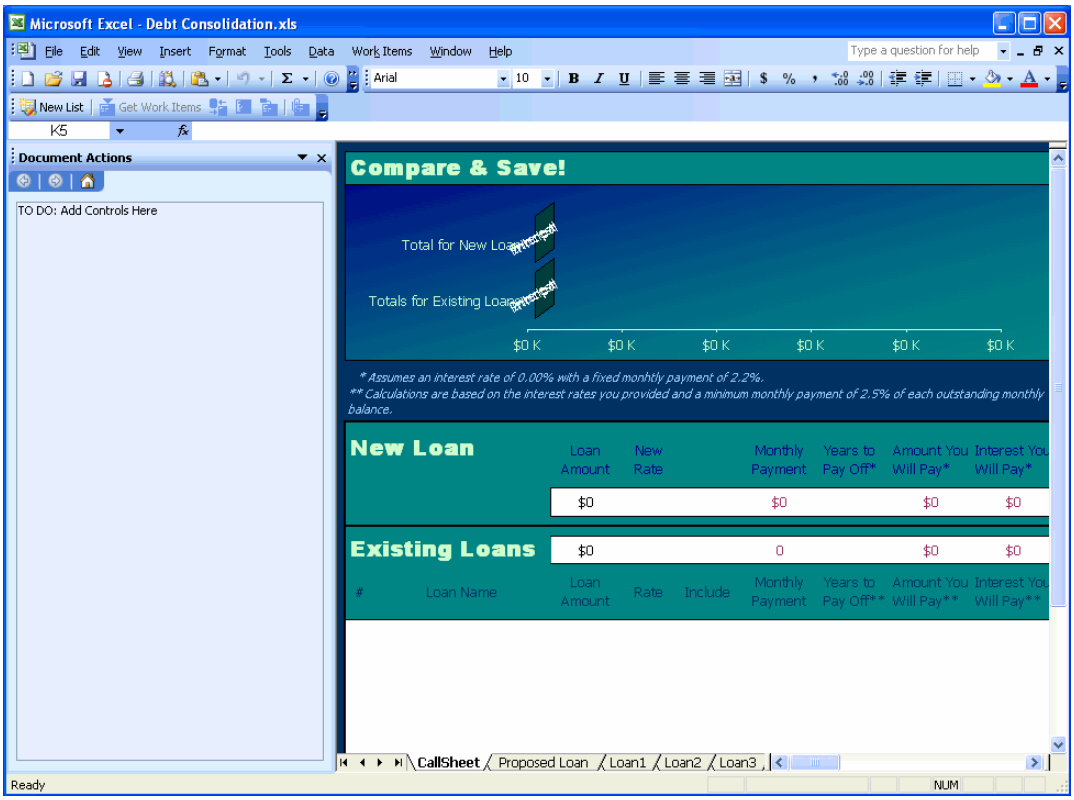


**Figure 8. The code adds a label to the actions pane.**

## Exercise 2: Add a User Control to the Actions Pane

### Create a user control

1. In **Solution Explorer**, right-click the **DebtConsolidation** project, select **Add** on the context menu, and then click **New Item**. The **Add New Item** dialog box appears.
2. Select the **User Control** template.
3. Type **CustomerProfile** in the **Name** box and click **Add**.

4. Set properties for the **CustomerProfile** user control using **Table 3**.

| Property | Value |
|---|---|
| BackColor | InactiveCaptionText |
| ForeColor | Desktop |
| Size | 300,600 |

**Table 3. Profile settings for the CustomerProfile user control.**

### Add the user control to the Excel actions pane

1. In **Solution Explorer**, right-click **Thisworkbook.vb** and select **View Code** on the context menu.
2. Add a new member variable for an instance of a **CustomerProfile** control to the **ThisWorkbook** class:

```
Friend WithEvents profileControl As CustomerProfile
```

3. Append the following code to **ThisWorkbook_Startup** to add an instance of the **CustomerProfile** control to the actions pane:

```
profileControl = New CustomerProfile
Me.ActionsPane.Controls.Add(profileControl)
```

### Checkpoint

1. On the **File** menu, select **Save All**.
2. On the **Debug** menu, select **Start** to build and run the project. The workbook Debt Consolidation opens in Excel.
3. The actions pane now contains two controls, a Label control and a CustomerProfile user control.

   **Note** You will add controls to the CustomerProfile component in another lab. For now, you will only see the user control surface in the actions pane.
4. Close the workbook without saving changes and exit Excel.

# Lab 3 - Data in an Excel Workbook and Actions Pane

In this lab, you will connect to a SQL Server data source, bind controls in both the workbook and actions pane to database objects, and navigate records in the data source.

Estimated time to complete:

- Exercise 1: Set Up a Connection and Add a Data Source – 10 minutes
- Exercise 2: Connect, Bind and Navigate Data – 40 minutes

This lab uses the **DebtConsolidation** project you created in Lab 3.

## Exercise 1: Set Up a Connection and Add a Data Source

### Create the sample database using a SQL script

1. On the Windows **Start** menu, click **Run**. Type **cmd** and click **OK**. A Command window opens.
2. Change to the folder where the lab files reside:

   ```
   cd C:\VSTO2005\Files\Excel
   ```

3. Execute the batch procedure:

   ```
   osql.exe –n -E -i DebtConsolidation.sql
   ```

   You will receive a message that the database is successfully restored.
4. Close the Command window.

### Add a data source to your project

1. On the **Data** menu, click **Show Data Sources**. The **Data Source** window appears.
1. Click Add New Data Source in the Data Source window. The Data Source Configuration window appears.
2. Select **Database** for the data source type and click **Next**.
3. Click **New Connection**. If you have not created a data connection before, the **Choose Data Source** dialog box appears. If you have created a data connection, you see the **Add Connection** dialog box shown in **Figure 10** instead.
4. In the **Choose Data Source** dialog box, select Microsoft SQL Server, as shown in **Figure 9**.

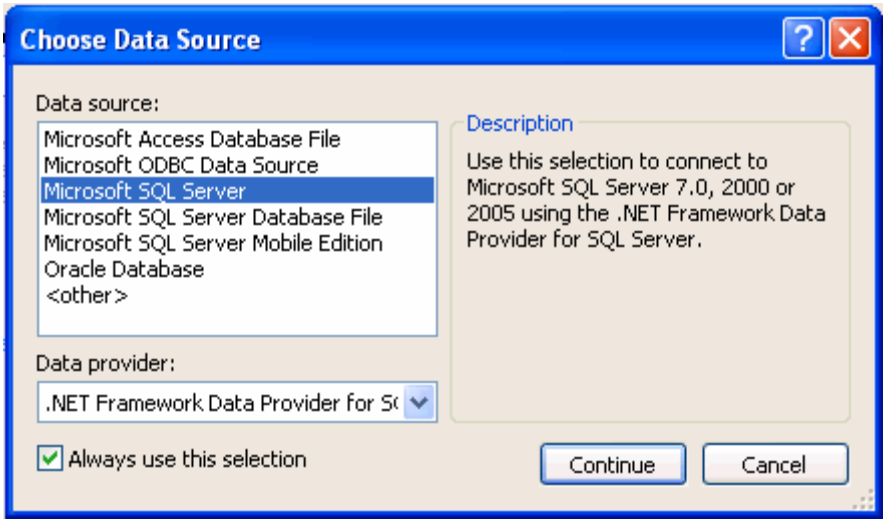


**Figure 9. The Microsoft SQL Server data source is used for Microsoft SQL Server 7.0 and up.**

5. Click **Continue**. You see the **Add Connection** dialog box shown in **Figure 10**.

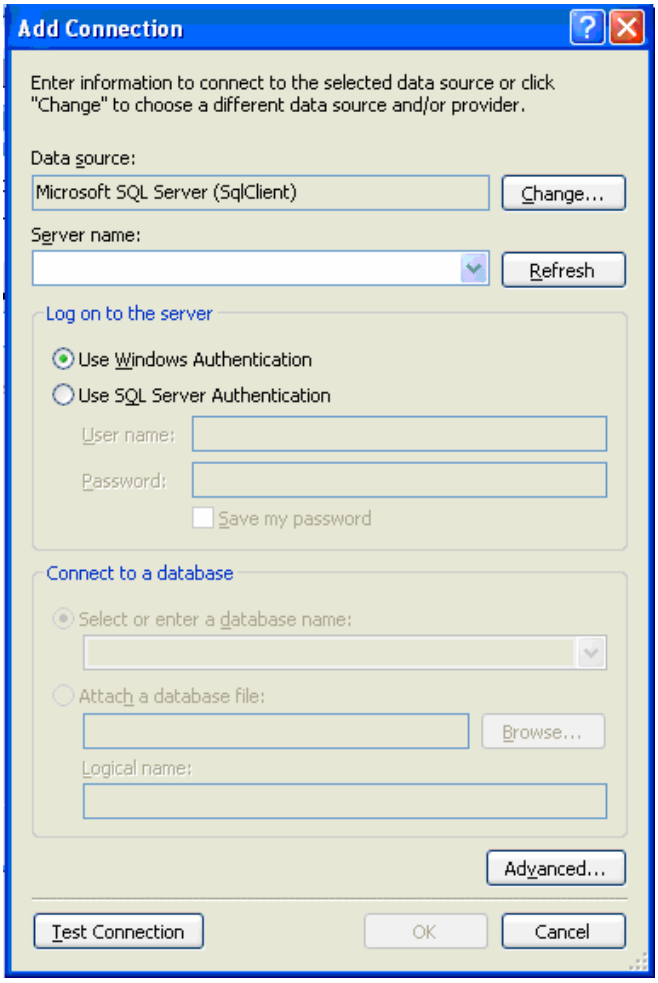


**Figure 10. The Add Connection dialog box prompts you for information about the new connection.**

6. Create the new connection:
   - If the **Data Source** is not **Microsoft SQL Server (SqlClient)**, click **Change** and choose the SQL Server data source.
   - In **Server Name**, type **(local)**.
   - Select **Use Windows Authentication** for the authentication mode.
   - In the **Connect to a database** section, select **VSTO2005DebtConsolidation** in the **Select or enter a database name** list.
   - Click **Test Connection** to confirm that the connection is valid and then click **OK** to add the new connection. The new connection has the default name **[YourServerName].VSTO2005DebtConsolidation.dbo**.
7. Click **Next**.
8. Save the connection using the default name, **VSTO2005DebtConsolidationConnectionString**.
9. Click **Next**.

10. In the list of database objects, select the **Tables** check box and click **Finish**.

### Add a relation to the data source

Right-click **VSTO2005DebtConsolidationDataSet** in the **Data Sources** window and then select **Edit DataSet with Designer** on the context menu. Note that a relation already exists between the NewCustomers and LoanData tables, as shown in **Figure 11.**

Right Click on the relation and select “edit relation” then if the relation name is not FK_LoanData_NewCustomers, change it so that it is, but just the name, not the relation itself. (the primary key is still in NewCustomers)

1. below.

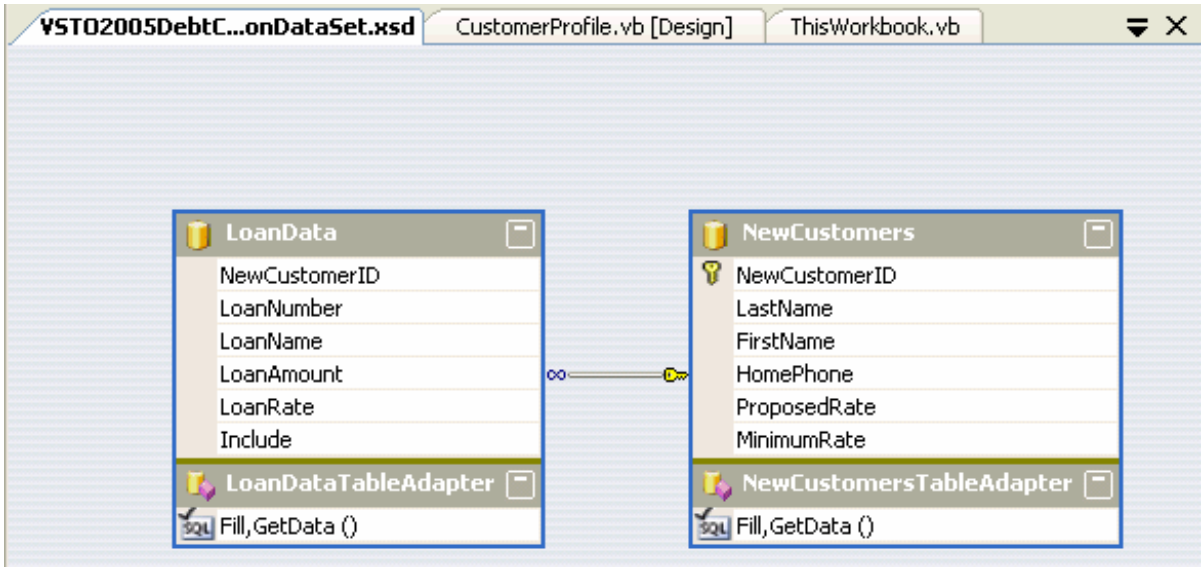


**Figure 11. A relation has already been created between LoanData and NewCustomers.**

Right Click on the relation and select “edit relation” then if the relation name is not FK_LoanData_NewCustomers, change it so that it is, but just the name, not the relation itself. (the primary key is still in NewCustomers)

## Exercise 2: Connect, Bind and Navigate Data

### Select controls to use for binding NewCustomers fields

1. In **Solution Explorer**, right-click **CustomerProfile.vb** and select **View Designer** on the context menu.
2. On the **Data** menu, select **Show Data Sources** to display the **Data Sources** window.
3. Select **NewCustomers** in the **Data Sources** window.
4. Click the drop-down arrow next to **NewCustomers** and select **Details**.
5. Expand **NewCustomers**.

6. Select the **ProposedRate** field.
7. Click the drop-down arrow next to **ProposedRate** and select **Customize**. The **Options** dialog box appears.
8. Select **NumericUpDown** in the **Associated Controls** list (if it is not already selected) and click **OK**.
9. Once again select the **ProposedRate** field.
10. Click the drop-down arrow next to **ProposedRate** and select **NumericUpDown**.
11. Click to select the **MinimumRate** field.
12. Click the drop-down arrow next to **MinimumRate** and select **None**.

### Add controls bound to the data source to the CustomerProfile component

1. Drag **NewCustomers** from the **Data Sources** window and drop it at the upper left corner of the user control design surface.

   Observe that the designer automatically generates bound **TextBox** controls and a **NumericUpdown** control for the selected fields in the **NewCustomers** table. The designer also adds a **DataNavigator** control for navigating the records in the data source at run-time.

2. In the **Data Source** window, select the **LoanData** node of **NewCustomers** and use the mouse to drag it to the user control. Drop it under the NewCustomer fields.

### Adjust the data grid display properties

1. Right-click the **LoanDataDataGridView** control and choose **Edit Columns**.
2. Remove **NewCustomerID**, **LoanNumber**, **LoanAmount**, and **LoanRate** from the list of selected controls by clicking the **Remove** button.
3. Select **LoanName** in the list of selected columns. Set the **Width** property to **150** and the **Readonly** property to **True**.
4. Select **Include** in the list of selected columns and click the move up button. Set the **Width** property to **50** and the **Readonly** property to **True**.
5. Click **OK** to close the **Edit Columns** dialog box.

   **Note** At this point, **Include** and **LoanName** should be the only two columns in **LoanDataDataGridView**.

6. Set the properties for **LoanDataDataGridView** as shown in **Table 4**.

| Property | Value |
|---|---|
| AllowUsersToAddRows | False |
| AllowUsersToDeleteRows | False |
| Location | 15, 200 |
| RowHeadersVisible | False |
| Size | 200,175 |

**Table 4. Property settings for the data grid.**

### Set properties and additional data bindings to the NumericUpDown control

1. Select the **ProposedRateNumericUpDown** control.
2. Set the **DecimalPlaces** property to **1**, the **Increment** property to **0.5** and the **Maximum** property to **24**.
3. Expand the **(DataBindings)** property and click the ellipses (...) next to **(Advanced)**. The **Formatting and Advanced Binding** dialog box appears.
4. Select **Minimum** in the **Property** List. Click the drop-down arrow for the **Binding** list. In the list, expand **NewCustomersBindingSource** and select **MinimumRate**.
5. Click **OK** to close the **Formatting and Advanced Binding** dialog box.
6. Select **NewCustomersBindingNavigator** in the component tray.
7. In the Properties window, expand the **BindingSource** properties and set the **AllowNew** property to **False**.

### Fill the table adapters when the component loads

1. On the **View** menu, select **Code**.
2. Select **(CustomerProfile Events)** from the **Class Name** list.
3. Select **Load** from the **Method Name** list.
4. Add code to fill the table adapters:

```
Private Sub CustomerProfile_Load(ByVal sender As Object, _
     ByVal e As System.EventArgs) Handles Me.Load
     Me.LoanDataTableAdapter.Fill( _
      Me.VSTO2005DebtConsolidationDataSet.LoanData)
     Me.NewCustomersTableAdapter.Fill( _
      Me.VSTO2005DebtConsolidationDataSet.NewCustomers)
End Sub
```

### Toggle the Include field when a DataGridView row is clicked

1. Select **LoanDataDataGridView** from the **Class Name** list and **CellContentClick** from the **Method Name** list.
2. Add code to toggle the **Include** field when the row is clicked:

```
Private Sub LoanDataDataGridView_CellContentClick( _
    ByVal sender As Object, ByVal e As _
    System.Windows.Forms.DataGridViewCellEventArgs) _
    Handles LoanDataDataGridView.CellContentClick
    Try
        Dim r As Integer = Me.LoanDataDataGridView.CurrentCell.RowIndex
        Dim rv As DataRowView = DirectCast( _
            Me.LoanDataDataGridView.Rows(r).DataBoundItem, DataRowView)
        rv("Include") = Not (rv("Include"))
        rv.EndEdit()
    Catch ex As Exception
```

```
        MessageBox.Show(ex.Message)
    End Try
End Sub
```

### Eliminate placeholder label from the actions pane

1. In the **Solution Explorer** window, right-click **ThisWorkbook.vb** and select **View Code**.
2. In the **ThisWorkbook_Startup** procedure, select the three lines used to define and place the label on the actions pane and click the **Comment out the selected lines** toolbar button:

```
        'Dim labelControl As New Label
        'labelControl.Text = "TO DO: Add Controls Here"
        'Me.ActionsPane.Controls.Add(labelControl)
```

### Checkpoint

1. On the **File** menu, click **Save All**.
2. On the **Debug** menu, click **Start** to build and run the project. The workbook Debt Consolidation opens in Excel.
3. The records from the data source appear in the actions pane as illustrated in Figure 15.

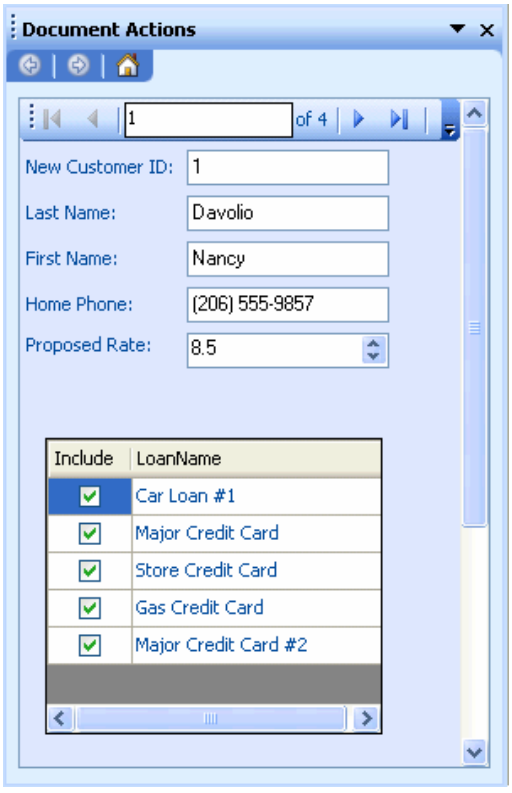


**Figure 12. The actions pane now contains bound data.**

The first record that displays is for the customer **Nancy Davolio**.

4. Click one or more rows in the DataGridView control to check/uncheck the rows.
5. Change the **ProposedRate**.
6. Click **Move next** on the **DataNavigator** control to move to the next record. The next record that displays corresponds to the customer **Andrew Fuller**.
7. Click **Move previous** on the **DataNavigator** control to move to the previous record. The data for the customer **Nancy Davolio** appears. Notice that your changes to the bound **DataGridView** and **NumericUpDown** controls are restored (in other words, your changes were persisted in the in-memory dataset).
8. Close the workbook without saving changes and exit Excel.

### Update a named range based on changes in the actions pane

1. In **Solution Explorer**, right-click **CustomerProfile.vb** and select **View Code** on the context menu.
2. Add code to the **CustomerProfile** class that will raise an event named **RateChanged** when the value in the bound **NumericUpDown** control changes:

```
Public Event RateChanged(ByVal NewRate As Double)

Private Sub ProposedRateNumericUpDown_ValueChanged( _
  ByVal sender As Object, ByVal e As System.EventArgs) _
  Handles ProposedRateNumericUpDown.ValueChanged
    RaiseEvent RateChanged(ProposedRateNumericUpDown.Value * 0.01)
End Sub
```

3. In **Solution Explorer**, right-click **Thisworkbook.vb** and select **View Code** on the context menu.
4. Add an event handler for the **CustomerProfile** control's **RateChanged** event to add the new rate to the worksheet in the named range **InterestRate**:

```
Private Sub CustomerProfile_RateChanged(ByVal NewRate As Double) _
   Handles profileControl.RateChanged
      Globals.Sheet2.InterestRate.Value2 = NewRate
End Sub
```

### Bind an Excel List object to the data source

1. In **Solution Explorer**, right-click **CustomerProfile.vb** and select **View Code** on the context menu.

2. Add a new read-only property named **Connector** that will return the BindingSource that the CustomerProfile control is using for the current data source:

```
Public ReadOnly Property Connector() As BindingSource
    Get
      Return Me.NewCustomersBindingSource
    End Get
End Property
```

3. In **Solution Explorer**, right-click **Sheet2.vb** and select **View Designer** on the context menu.
4. On the **Data** menu, click **Microsoft Office Excel Data**, select **List** and then select **Create List**. You see the **Create List** dialog box.
5. Type **=A12:G13** for the range, select **My list has headers** and click **OK**.
6. On the **Data** menu, click **Microsoft Office Excel Data**, select **Filter** and then select **AutoFilter** to remove the Filter controls from the list.
7. On the **Data** menu, click **Microsoft Office Excel Data**, select **List** and then select **Hide Border of Inactive Lists**.
8. In Solution Explorer, right-click **ThisWorkbook.vb** and click **View Code**.
9. Append the following code to **ThisWorkbook_Startup**:

```
With Globals.Sheet2.List1
    .AutoSetDataBoundColumnHeaders = False
    .SetDataBinding(profileControl.Connector, "FK_LoanData_NewCustomers")
End With
```

   This will bind the **List** object on the **CallSheet** worksheet to the same BindingSource in use by the CustomerProfile control in the actions pane.

### Checkpoint

1. On the **File** menu, click **Save All**.
2. On the **Debug** menu, click **Start** to build and run the project. The workbook Debt Consolidation opens in Excel.
3. Notice that the data in the task pane now appears in the **CallSheet** worksheet as shown in **Figure 13**.
4. Change the **ProposedRate** and observe that your changes are reflected in the **InterestRate** named range on the worksheet.

5. Check and/or uncheck **Include** columns in the **DataGridView** and observe that your changes are reflected in the **List** control on the worksheet.
6. Close the workbook without saving changes and quit Excel.

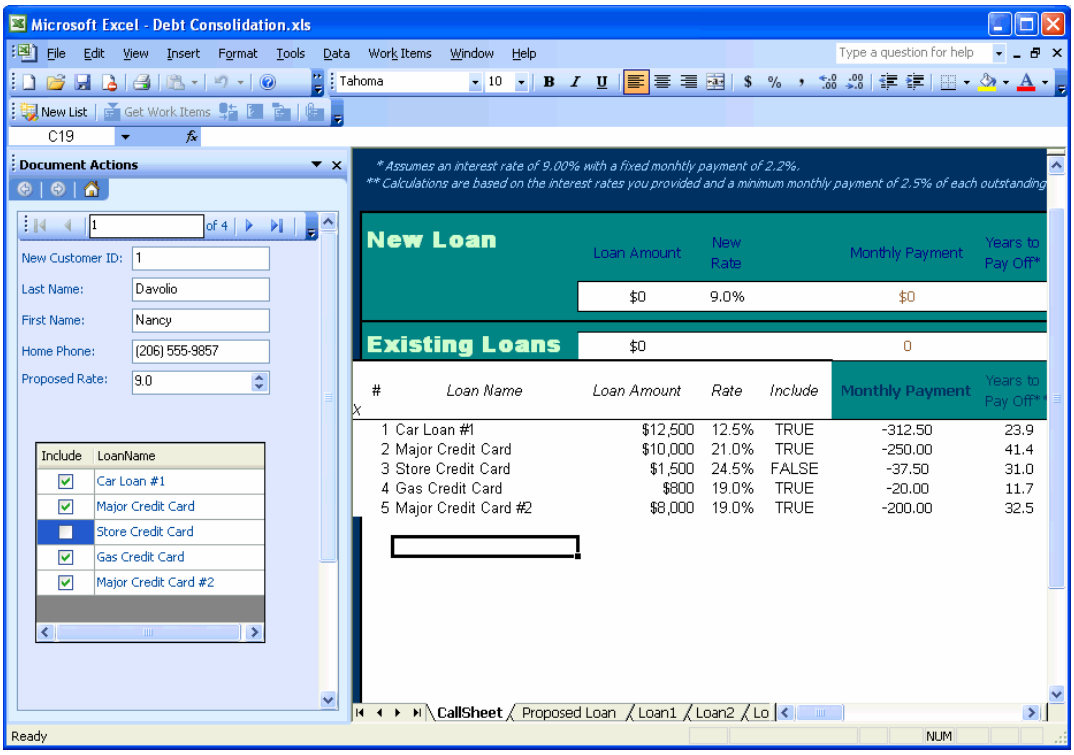


**Figure 13. The data now appears in the worksheet and the actions pane.**