



COMPSCI 280 S1 2011 Enterprise Software Development

ADO.NET
Data Binding



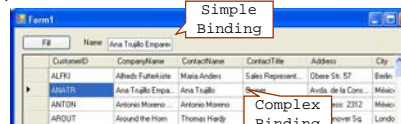
Agenda

- ▶ **Agenda:**
 - ▶ Data Binding
 - ▶ Understanding Data Bindings
 - ▶ Types of Data Binding
 - Simple Binding
 - Complex Binding
 - ▶ Record Navigation
 - ▶ The Position Property
 - ▶ The BindingNavigator Control
- ▶ **Recommended Reading:**
 - ▶ Microsoft ADO.NET 2.0 step by step, Rebecca M. Riordan
 - ▶ Chapter 12 : Data Binding in Windows Forms by Using the BindingSource Component
 - ▶ www
 - ▶ <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndotnet/html/databindingadonet.asp>
- ▶ **Hands-on Lab:**
 - ▶ Lecture23Lab: Binding a combo box control to a data source.



Binding DataSets

- ▶ Associating a control instance (e.g. textbox) with a field in a data source is called data binding
- ▶ Two types of data binding
 - ▶ Simple binding
 - ▶ Display ONE value (Example: TextBox)
 - ▶ Complex binding
 - ▶ Display multiple rows from a data source (Example: DataGridView)
- ▶ **BindingSource Component**
 - ▶ Associates controls on a form to data.
 - ▶ Attach the BindingSource component to your data source, and then
 - ▶ Bind the controls on your form to the BindingSource component.
 - ▶ All further interaction with the data, including navigating, sorting, filtering, and updating, is accomplished with calls to the BindingSource component.
 - ▶ Supports both simple and complex data binding as indicated by its DataSource and DataMember properties.

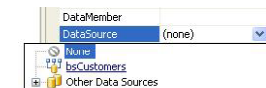
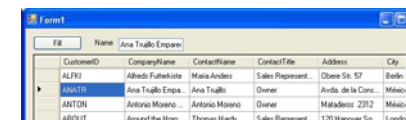
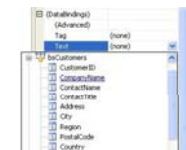


- ◆ Simple data binding is the ability of a control to bind to a single data element, such as a value in a column in a DataSet table. For example, TextBox control or Label control - that is, a control that typically only displays a single value
- ◆ Complex data binding is the ability of a control to bind to more than one data element, or typically more than one record in a DataSet



Binding DataSets (con't)

- ▶ To create a BindingSource component
 - ▶ Drag a BindingSource component onto a form
 - ▶ Set the DataSource property to the dataset
 - ▶ Set the DataMember property to the Customers table
- ▶ To bind a control to the BindingSource component
 - ▶ Simple Binding
 - ▶ Select the control: TextBox
 - ▶ Set the **DataBindings | Text** property to bsCustomers | CompanyName
 - ▶ Complex Binding
 - ▶ Select the DataGridView control
 - ▶ Set the **DataSource** property to bsCustomers





Creating Simple Binding

- ▶ Uses with TextBox, Label, Button, CheckBox, DateTimePicker
- ▶ To bind a control at run time
 - ▶ Create a new BindingSource
 - ▶ DataSource
 - The The data source for the BindingSource. Set it to a data source.
 - ▶ DataMember
 - The specific column or list name within the data source to bind to.
 - ▶ Create a new binding object
 - ▶ PropertyName: Contains the name of the control property to bind:
 - e.g. Text property of a TextBox/Label, Checked property of a checkbox, Enabled property of a button
 - ▶ DataSource
 - The BindingSource component
 - ▶ DataMember
 - Provides the navigation path to a field in the DataSet
 - ▶ Add the new binding object to the DataBindings collection of the control
 - ▶ By adding Binding objects to the collection, you can bind any property of a control

```
bsCustomers = new BindingSource(northwindDataSet, "Customers");
b = new Binding("Text", bsCustomers, "ContactName");
txtContactName.DataBindings.Add(b);
```

- ◆ Simple binding allows you to display a single data element, such as a column value from a dataset table, in a control. You can simple-bind any property of a control to a data value.
- ◆ Instead of binding a control directly to a data source, you bind the control to a BindingSource, and you attach the data source to the BindingSource component's DataSource property



Creating Complex Binding

- ▶ Uses with ComboBox, ListBox, DataGrid
- ▶ To create a Complex Binding at run time
 - ▶ Create a new BindingSource
 - ▶ Set the DataSource property of the BindingSource to the DataSet
 - ▶ Set the DataMember property of the BindingSource to the table
 - ▶ Select the control
 - ▶ Set the DataSource property of the control to the BindingSource
 - ▶ Set the DisplayMember property of the ComboBox/ListBox control



```
bsProducts = new BindingSource(northwindDataSet, "Products");
dgvProducts.DataSource = bsProducts;
```

```
lstProductName.DataSource = bsProducts;
lstProductName.DisplayMember = "ProductName";
```

ListBox

```
bsCustomers = new BindingSource(northwindDataSet, "Customers");
cboCompanyName.DataSource = bsCustomers;
cboCompanyName.DisplayMember = "CompanyName";
cboCompanyName.DropDownStyle = ComboBoxStyle.DropDownList;
```

Combo box

- ◆ If you set the DropDownStyle property to DropDownList, users can select only valid values from the list.

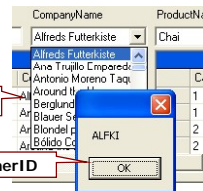


Binding Combo Boxes

- ▶ To Bind a combo box
 - ▶ Create a new BindingSource
 - ▶ Select the combo box
 - ▶ Data Source
 - Set the DataSource to the BindingSource component
 - ▶ DisplayMember
 - Name of the DataSource's property to bind to **show** the item in the list/combo box
 - ▶ Optional: ValueMember:
 - Name of the DataSource's property to bind to as the **value**

Display CompanyName

Selected Value: CustomerID



```
cboCompanyName.DataSource = bsCustomers;
cboCompanyName.DisplayMember = "CompanyName";
cboCompanyName.ValueMember = "CustomerID";
```

```
private void cboCompanyName_SelectedIndexChanged(object sender, EventArgs e) {
    if (cboCompanyName.SelectedIndex >= 0)
        Console.WriteLine(cboCompanyName.SelectedValue);
    ...
}
```

SelectedValue: Prints CustomerID instead

- ◆ For example, you want the user to select by company name, but your internal logic actually needs the ID of the customer. In that case, you can set DataSource to Customers table, DisplayMember to CompanyName, and ValueMember to CustomerID. Then the list box shows the company name (because of the DisplayMember setting), but when you retrieve a selected item from the combo box, it returns the customer ID associated with the selected name!
- ◆ Note: if the combo box is using the DisplayMember and ValueMember properties to return different information from that displayed, then the SelectedValue property of the combo box should be used to display the value of the selection.



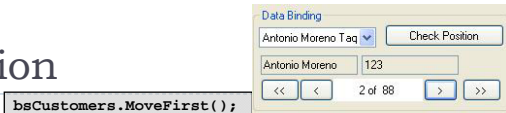
BindingSource

- ▶ Properties
 - ▶ Count: Gets the total number of items
 - ▶ Current: Gets the current item
 - ▶ DataMember: Gets or sets the specific list in the data source to which the connector currently binds to.
 - ▶ DataSource: Gets or sets the data source that the connector binds to.
 - ▶ Position: Gets or sets the index of the current item in the underlying list.
 - ▶ Sort: Gets or sets the column names used for sorting
 - ▶ Filter: Gets or sets the expression used to filter which rows are viewed.
- ▶ Methods
 - ▶ EndEdit: Applies pending changes to the underlying data source.
 - ▶ CancelEdit: Cancels the current edit operation.
 - ▶ MoveFirst: Moves to the first item in the list.
 - ▶ MoveLast: Moves to the last item in the list.
 - ▶ MoveNext: Moves to the next item in the list.
 - ▶ MovePrevious: Moves to the previous item in the list.
 - ▶ RemoveCurrent: Removes the current item from the list.
- ▶ Events
 - ▶ PositionChanged: Occurs after the value of the Position property has changed.



Record Navigation

- ▶ To move to the first record
- ▶ To move to the last record
- ▶ To move to the previous record
- ▶ To move to the next record
- ▶ To display the record position



```
bsCustomers.MoveFirst();
```

```
bsCustomers.MoveLast();
```

```
bsCustomers.MovePrevious();
```

```
bsCustomers.MoveNext();
```

- ▶ Uses the Position and Count properties

```
lblNavLocation.Text = (bsCustomers.Position + 1) + " of " + bsCustomers.Count.ToString();
```

Add 1, since position starts from 0

- ▶ To update a record position automatically

- ▶ The PositionChanged event

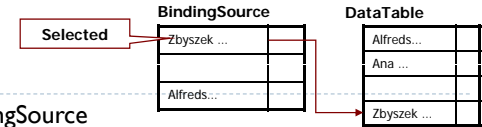
- ▶ Occurs when the Position property has changed

```
private void bsCustomers_PositionChanged(Object sender, EventArgs e) {
    lblNavLocation.Text = (bsCustomers.Position + 1) + " of " + bsCustomers.Count.ToString();
}
```

The Position property: a zero-based index that specifies a position in the underlying list.



Position



- ▶ The Position property of the BindingSource

- ▶ Gets the index of the current item in the underlying list
- ▶ Note: It is not equal to the absolute row position of the DataTable

- ▶ Example 1:

- ▶ Sorted by descending ContactName

- ▶ DataTable

- ▶ ContactName of the first row in the Customers table = Alfreds Futterkiste

- ▶ BindingSource

- ▶ ContactName = Zbyszek Piestrzeniewicz at position 0

But returns Alfreds ...

```
bsCustomers.Sort = "ContactName DESC";
Console.WriteLine(northwindDataSet.Customers.Rows[bsCustomers.Position]["ContactName"]);
```

- ▶ Note: Use the Current Property to get the current DataRowView object

```
DataRowView drv = ((DataRowView)bsCustomers.Current);
Console.WriteLine(drv["ContactName"].ToString());
```

Returns Zbyszek...

The Current object contains the value of the current item in the data source. To use the value of the current item, you must cast the item to the Type of the object contained by the DataSource. For example, a DataTable contains DataRowView objects. (When you bind to a DataTable, you are really binding to the table's default view.)



Data Manipulation

- ▶ Methods:

- ▶ To cancel current edit

```
bsCustomers.CancelEdit();
```

- ▶ The CancelEdit method discards modifications to data since the last save or load operation

- ▶ To end current edit

```
bsCustomers.EndEdit();
```

- ▶ When the EndEdit method is called, all pending changes are applied to the underlying data source

- ▶ To delete a record

```
bsCustomers.RemoveCurrent();
```

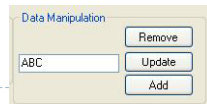
- ▶ The RemoveCurrent method removes the current item from the list.

- ▶ To add a new record

```
DataRowView dr = bsCustomers.AddNew();
```

- ▶ The AddNew method adds a new item. This method sets up the following series of actions

- The EndEdit method is automatically called to commit any pending edit operations.
- The AddingNew event is automatically raised
 - If the AddingNew event is not handled, and then the request is passed to the DataView.AddNew method to create a new DataRowView
- Note: the new item is not committed until an explicit call to EndEdit is made.



If a bound textbox is in the process of being edited and the control doesn't lose focus it may not be flagged as changed. For example, you click a textbox, change its value, then click the RowState button. Then the RowState property is Unchanged. After you edit a value, you must call EndEdit to end the edit operation.

when you move from one row to the next, the system automatically calls DataRow.EndEdit on the row that you just left.



The BindingNavigator Control

- ▶ The BindingNavigator control represents a standardized way to navigate and manipulate data on a form.

- ▶ It is a ToolStrip control with buttons preconfigured for navigation to the first, last, next, and previous record in a data set, as well as buttons to add and delete records.

UI Control	BindingNavigator member	BindingSource member
Move First	MoveFirstItem	Move First
Move Previous	MovePreviousItem	Move Previous
Current Position	PositionItem	Current
Count	CountItem	Count
Move Next	MoveNextItem	Move Next
Move Last	MoveLastItem	Move Last
Add New	AddNewItem	AddNew
Delete	DeleteItem	RemoveCurrent

- ▶ To create a BindingNavigator at design time

- ▶ Drag a BindingNavigator control to the form.
- ▶ Set the BindingNavigator control's BindingSource property to the BindingSource on the form