RED RIVER COLLEGE

ADEV-2005 Programming 2
Applied Computer Education
Red River College

DATA BINDING

RED RIVER COLLEGE

Topics

- What is data binding?
- Advantages of data binding
- Simple vs. Complex
- BindingSource object

Data Binding

- Creating a connection between UI (controls) and data
- Automatically populate controls
- Update controls when data changes
- Update data when UI changes

Advantages of Data Binding

- Quick and efficient
- Decreased code weight
- Execution time increases
- Events for handling data processes

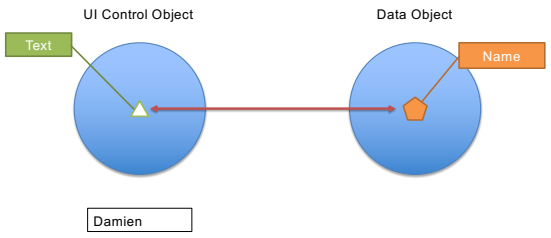
Types of Binding

- Simple
- Complex

Simple Binding

- Control property is bound to a single data element

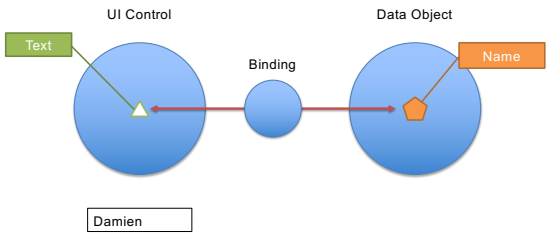
Visual Example



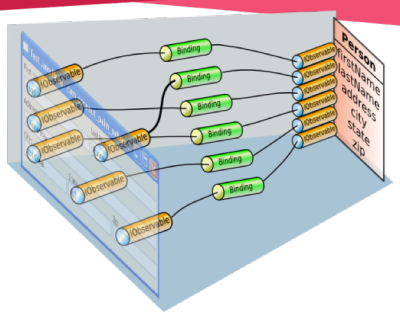
Binding Object

- Facilitates the connection between control and data
- Needs to know:
  - Property of the UI control to bind to
  - Data source
  - Property within the data object

Visual Example



Visual Example



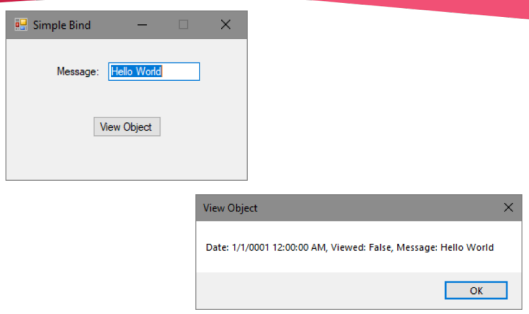
DataBindings Property

- Controls that can data bind have this property
- Collection of data bindings

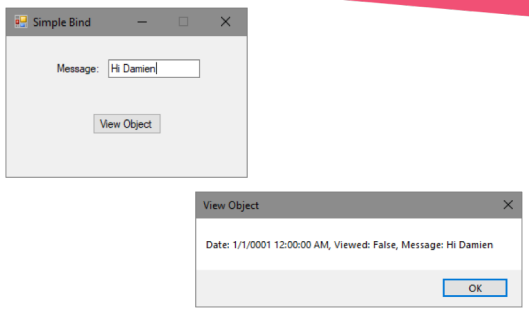
DataBindings Property Example

```
Response response = new Response();
response.Message = "Hello World";

// Binding(string propertyName, object dataSource, string
// dataMember)
Binding messageBind = new Binding("Text", response, "Message");

this.txtMessage.DataBindings.Add(messageBind);
```

Test (1 of 2)



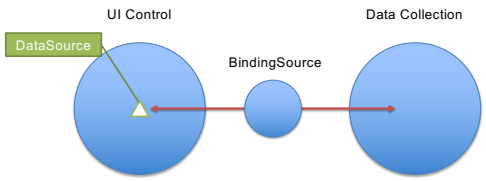
Test (2 of 2)



Complex Data Binding

- AKA: List Based Data Binding
- Control is bound to a data collection
  - Example: List
- Controls that can do complex binding:
  - DataGridView
  - ListBox
  - ComboBox

Visual Example



BindingSource Class

- Encapsulates a data source
- Provides
  - Currency Management
  - Change notifications
  - and other services between control and data

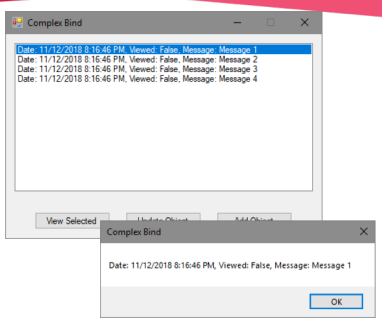
DataSource Property

- A control with a DataSource property can do complex binding

DataSource Property Example

```
BindingList<Response> responses = new BindingList<Response>();

// Add items to BindingList

BindingSource bindingSource = new BindingSource();
bindingSource.DataSource = responses;

this.lstResponses.DataSource = bindingSource;
```

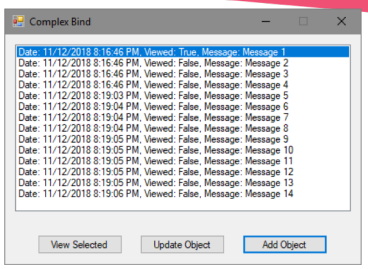
Test



Add An Item To The List

```
private void btnAdd_Click(object sender, EventArgs e)
{
    responses.Add(new Response(DateTime.Now,
                               "Message " + ++count));
}
```

Test



ComboBox Binding

- Default: objects display result of ToString()

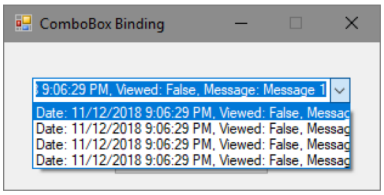
ComboBox Binding Example

Red River College

```
BindingList<Response> responses = new BindingList<Response>();

// Add items to BindingList

BindingSource bindingSource = new BindingSource();

bindingSource.DataSource = responses;

this.cboResponses.DataSource = bindingSource;
```

Test

Red River College



DisplayMember Property

Red River College

- string Property
- The name of the property within the data object to display in the ComboBox

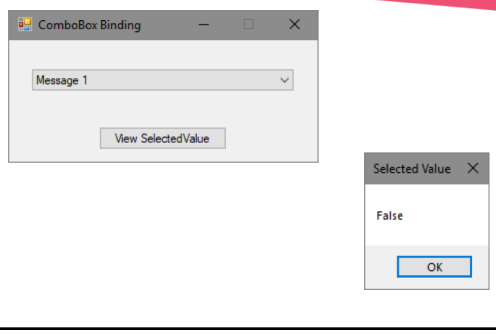
ValueMember Property

- string Property
- The name of the property of the data object to be the actual value of the ComboBox

ComboBox Binding Example

```
BindingList<Response> responses = new BindingList<Response>();

// Add items to BindingList

BindingSource bindingSource = new BindingSource();

bindingSource.DataSource = responses;

this.cboResponses.DataSource = bindingSource;
this.cboResponses.DisplayMember = "Message";
this.cboResponses.ValueMember = "WasViewed";
```

Test



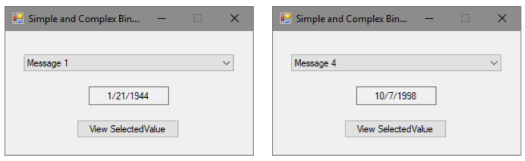
Combining Simple and Complex Binding

- When doing complex binding, the selected item is referred to as ”Current”
- BindingSource object keeps track of Current
- When current changes, all bound objects will update to display “Current”

Bind Multiple Controls to BindingSource

```
BindingSource bindingSource = new BindingSource();
bindingSource.DataSource = responses;

// Complex Binding
this.cboResponses.DataSource = bindingSource;
this.cboResponses.DisplayMember = “Message”;
this.cboResponses.ValueMember = “WasViewed”;

// Simple Binding
this.lblDate.DataBindings.Add(“Text”, bindingSource, “Date”);
```

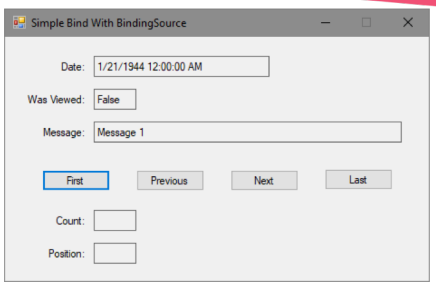
Test



BindingSource With Simple Bind Only

```
bindingSource = new BindingSource();
bindingSource.DataSource = responses;

this.lblDate.DataBindings.Add("Text", bindingSource, "Date");
this.lblWasViewed.DataBindings.Add("Text", bindingSource, "WasViewed");
this.lblMessage.DataBindings.Add("Text", bindingSource, "Message");
```

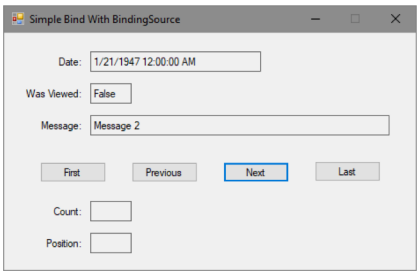
Test



Update Current

```
private void btnNext_Click(object sender, EventsArgs e)
{
    bindingSource.MoveNext();
}
```

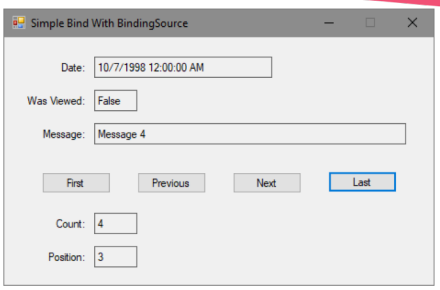
Test



Can't Bind Everything

```
private void BindingSource_PositionChanged(object sender, EventArgs e)
{
    UpdateUI();
}

private void UpdateUI()
{
    this.lblCount.Text = bindingSource.Count.ToString();
    this.lblPosition.Text = bindingSource.Position.ToString();
}
```

Test



BindingSource Class Members

- DataSource Property
- Current Property
- Position Property
- Count Property
- MoveFirst() Method
- MoveLast() Method
- MoveNext() Method
- MovePrevious() Method
- PositionChanged Event

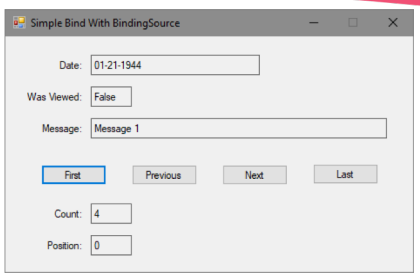
Format Binding

- Two Steps:
  1. Enable formatting
  2. Set FormatString Property

Format Binding Example

```
Binding dateBind = new Binding("Text", response, "Date", true);

dateBind.FormatString = "MM-dd-yy";

this.lblDate.DataBindings.Add(dateBind);
```

Test



Summary

- Connection between UI and Data
- Simple and Complex
- Binding object (simple)
- BindingSource object (complex)

QUESTIONS?