

Unlocking the Mysteries of the ProBindingSource

Laura Stern
Principal Software Engineer
October, 2013

PROGRESS
EXCHANGE 2013
DISCOVER. DEVELOP. DELIVER.

Agenda

- What is the ProBindingSource?
- What should you bind to?
- Design time setup for the BindingSource
- Internals
- The cursor
- Changing a query
- Inherited methods and properties
- Summary

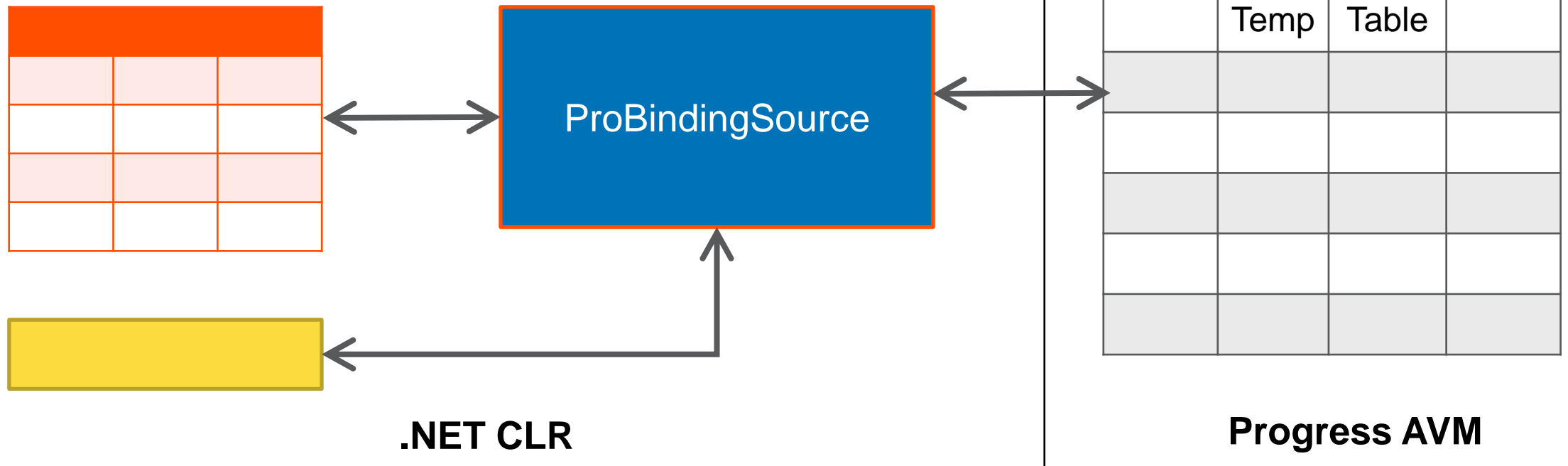
Why have a ProBindingSource (PBS)?

- The ABL is uniquely designed to associate data with UI widgets
 - **UPDATE Customer**, **DISPLAY Customer**, ...
 - Browse widget

- In 10.2A: Introduced GUI for .NET
 - Native .NET uses data binding
 - GUI for .NET needed a way to do data binding
 - Between a .NET control and ABL data

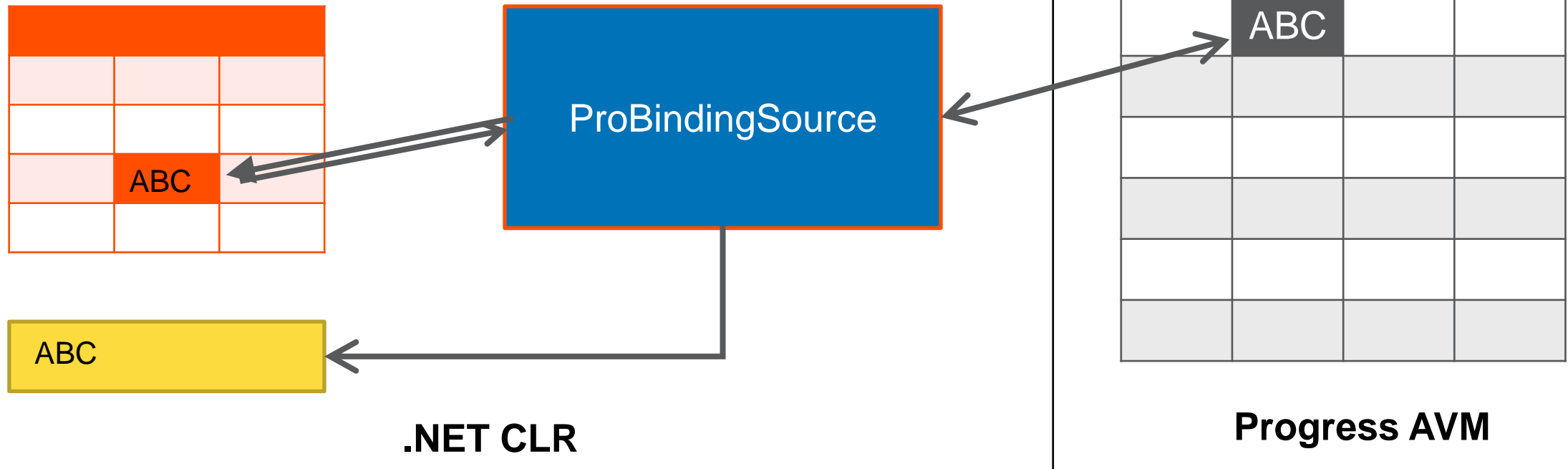
What is the ProBindingSource?

- .NET object: Progress.Data.BindingSource
 - inherits from System.Windows.Forms.BindingSource
- A **conduit** between a .NET control and an ABL data source
 - Data source: temp-table, DB table, ProDataSet, buffer



What is the ProBindingSource?

- The PBS has a copy of the schema
 - Usually a subset of the table's schema
- **The PBS does not have a copy of the table's data**



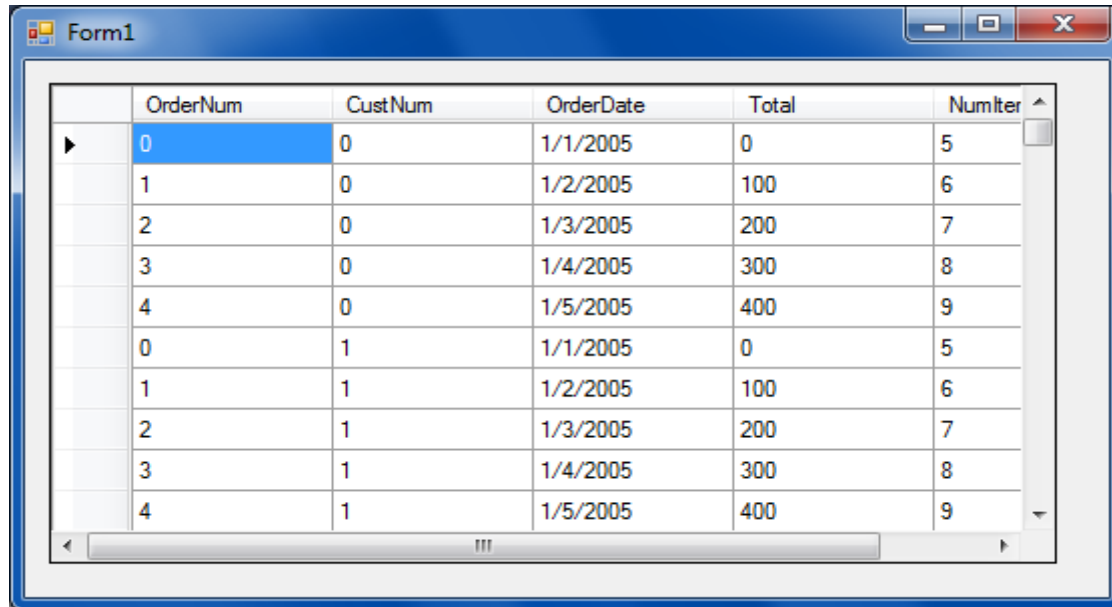
Agenda

- What is the ProBindingSource?
- What should you bind to?
- Design time setup for the BindingSource
- Internals
- The cursor
- Changing a query
- Inherited methods and properties
- Summary

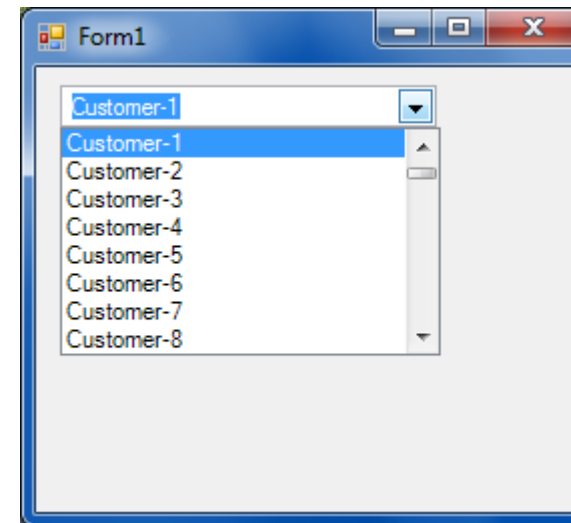
What Should the BindingSource be Bound to?

- For data from only one table: Bind to a query

```
CREATE QUERY qryHandle.  
pbs = NEW Progress.Data.BindingSource (qryHandle) .  
pbs = NEW Progress.Data.BindingSource () .  
pbs:Handle = qryHandle.
```



	OrderNum	CustNum	OrderDate	Total	Numiter
▶	0	0	1/1/2005	0	5
	1	0	1/2/2005	100	6
	2	0	1/3/2005	200	7
	3	0	1/4/2005	300	8
	4	0	1/5/2005	400	9
	0	1	1/1/2005	0	5
	1	1	1/2/2005	100	6
	2	1	1/3/2005	200	7
	3	1	1/4/2005	300	8
	4	1	1/5/2005	400	9



Customer-1
Customer-1
Customer-2
Customer-3
Customer-4
Customer-5
Customer-6
Customer-7
Customer-8

What Should the BindingSource be Bound to?

- If you have a ProDataSet
 - To show data from **one** table: Still bind to a query

```
qryHdl = myDataSet:TOP-NAV-QUERY.  
pbs = NEW Progress.Data.BindingSource (qryHdl) .
```

```
qryHdl = myDataSet:GET-RELATION (2) :Query.  
pbs = NEW Progress.Data.BindingSource (qryHdl) .
```


What Should the BindingSource be Bound to?

Form1


	CustNum	Name	City	State	Balance
	2	Customer-2	City-2	State-2	2000
	3	Customer-3	City-3	State-3	3000
	4	Customer-4	City-4	State-4	4000
	5	Customer-5	City-5	State-5	5000
▶	6	Customer-6	City-6	State-6	6000
	7	Customer-7	City-7	State-7	7000
	8	Customer-8	City-8	State-8	8000

	OrderNum	CustNum	OrderDate	Total	NumItems
	1	6	1/2/2005	100	6
▶	2	6	1/3/2005	200	7
	3	6	1/4/2005	300	8
	4	6	1/5/2005	400	9
*					

What Should the BindingSource be Bound to?

- It is possible to bind to a DataSet but show only one table

```
pbs = NEW Progress.Data.BindingSource (datasetHdl) .  
ultraGrid1:DataSource = pbs .  
ultraGrid1:DataMember = "Orders" .
```

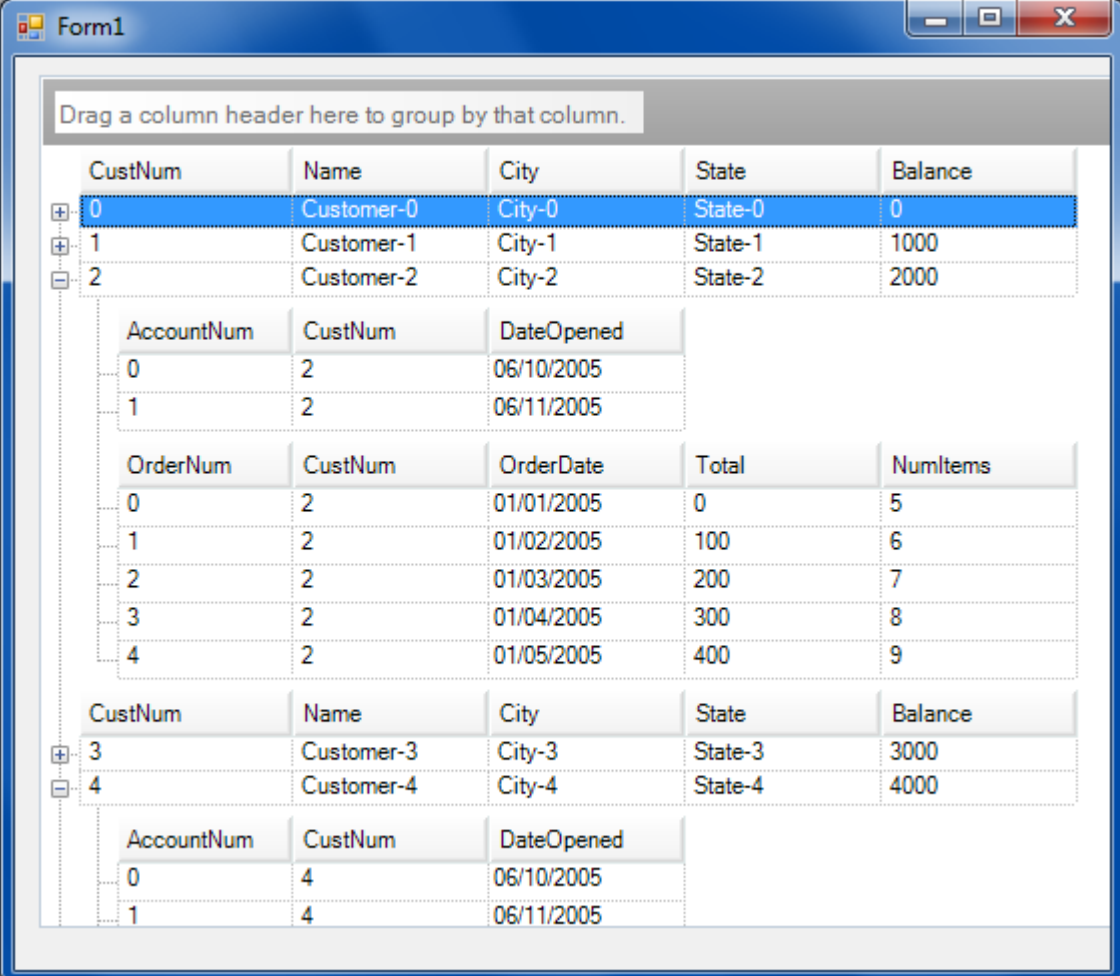


- **But don't do this!**
 - Lots of overhead that you don't need

What Should the BindingSource be Bound to?

- You want a hierarchical display

```
pbs = NEW  
Progress.Data.BindingSource  
(myDataSet) .
```



Drag a column header here to group by that column.

CustNum	Name	City	State	Balance
0	Customer-0	City-0	State-0	0
1	Customer-1	City-1	State-1	1000
2	Customer-2	City-2	State-2	2000

AccountNum	CustNum	DateOpened
0	2	06/10/2005
1	2	06/11/2005

OrderNum	CustNum	OrderDate	Total	NumItems
0	2	01/01/2005	0	5
1	2	01/02/2005	100	6
2	2	01/03/2005	200	7
3	2	01/04/2005	300	8
4	2	01/05/2005	400	9

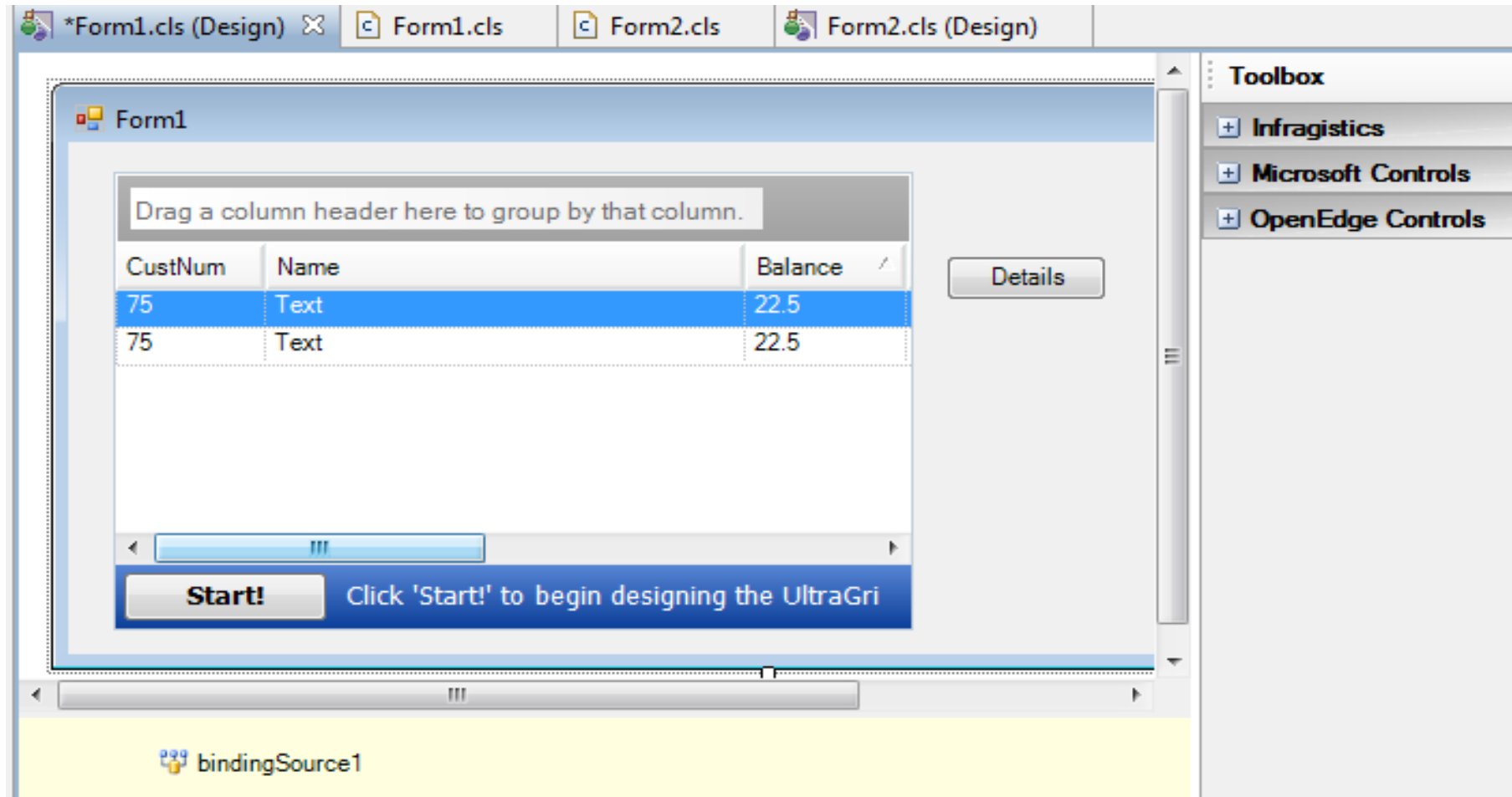
CustNum	Name	City	State	Balance
3	Customer-3	City-3	State-3	3000
4	Customer-4	City-4	State-4	4000

AccountNum	CustNum	DateOpened
0	4	06/10/2005
1	4	06/11/2005

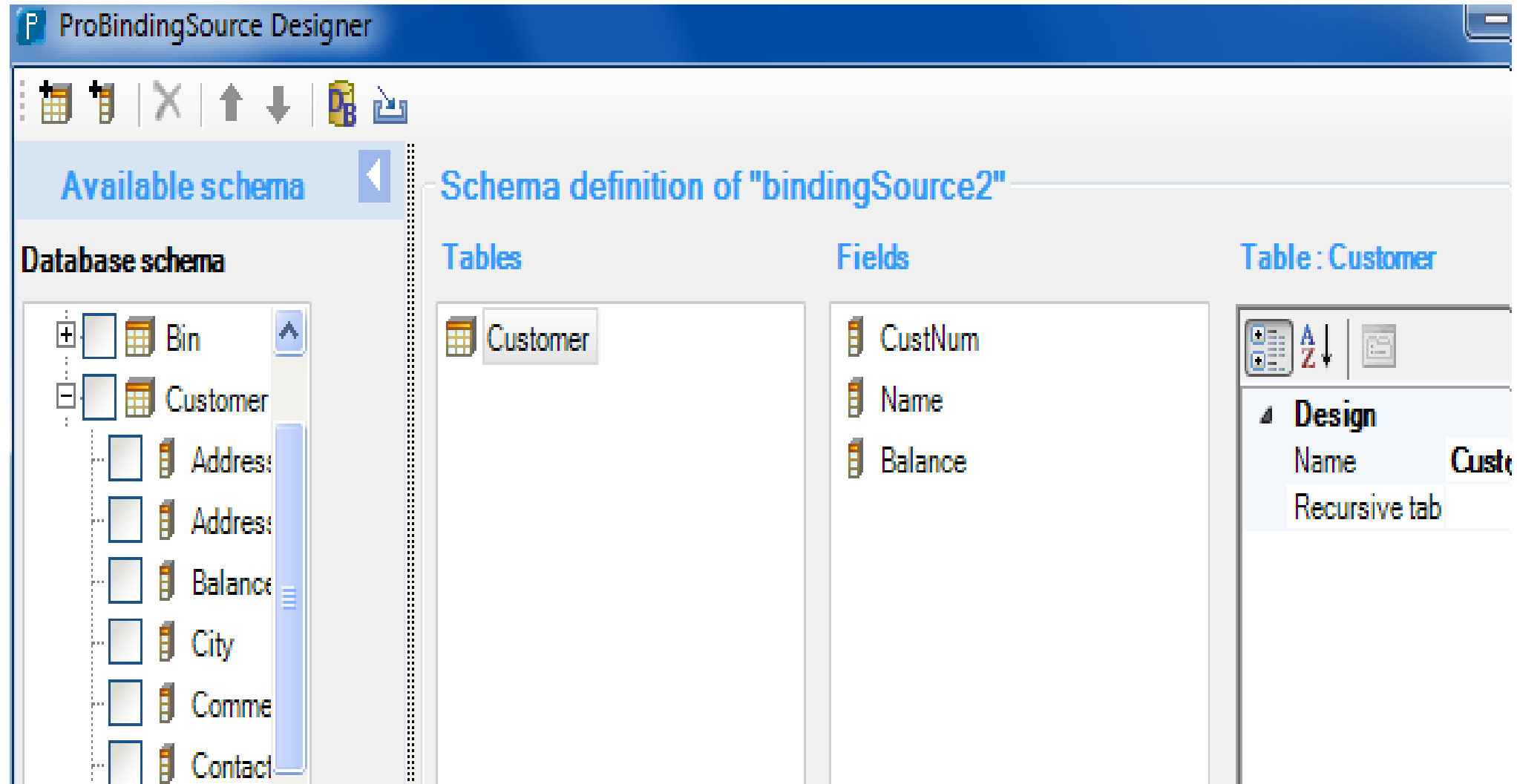
Agenda

- What is the ProBindingSource?
- What should you bind to?
- Design time setup for the BindingSource
- Internals
- The cursor
- Changing a query
- Inherited methods and properties
- Summary

Add BindingSource to Form via Visual Designer

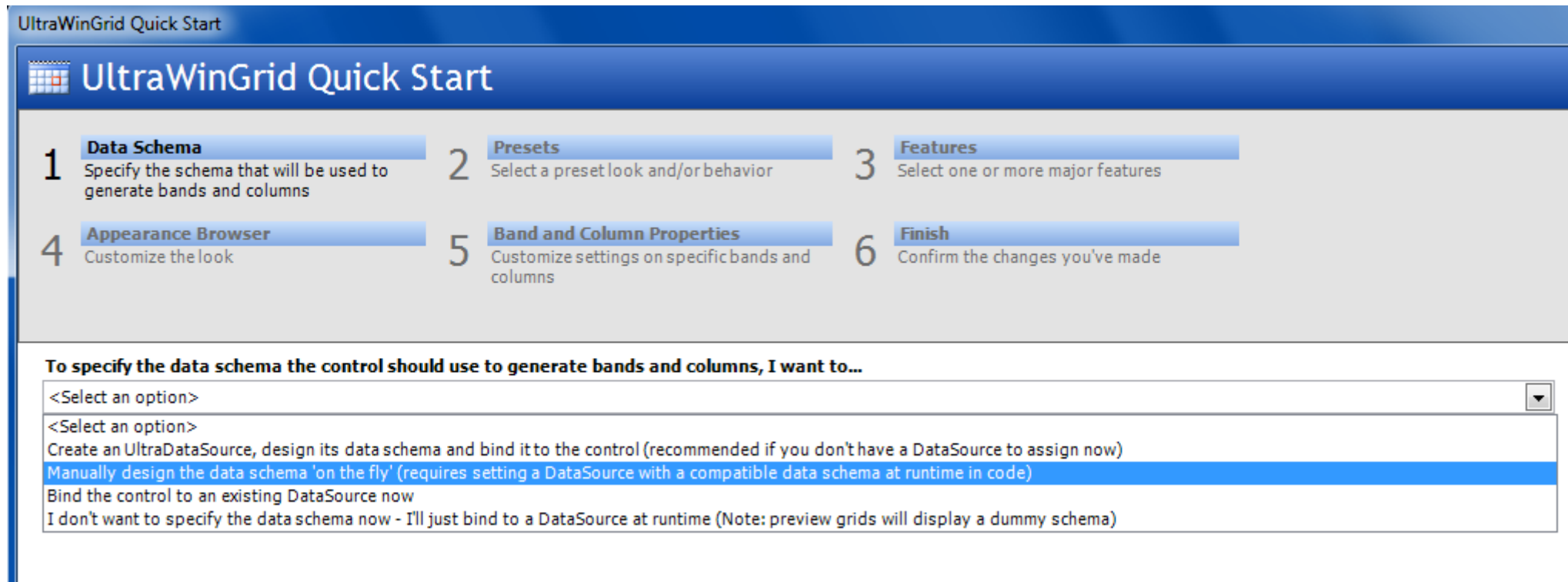


Specify schema for the BindingSource



Alternative: Add schema to control via Control's Designer

- Add schema to Grid itself
 - “Manually design the data schema ‘on the fly’
(requires setting a DataSource with a compatible data schema at runtime in code)”



UltraWinGrid Quick Start

UltraWinGrid Quick Start

- 1 Data Schema**
Specify the schema that will be used to generate bands and columns
- 2 Presets**
Select a preset look and/or behavior
- 3 Features**
Select one or more major features
- 4 Appearance Browser**
Customize the look
- 5 Band and Column Properties**
Customize settings on specific bands and columns
- 6 Finish**
Confirm the changes you've made

To specify the data schema the control should use to generate bands and columns, I want to...

<Select an option>

<Select an option>

Create an UltraDataSource, design its data schema and bind it to the control (recommended if you don't have a DataSource to assign now)

Manually design the data schema 'on the fly' (requires setting a DataSource with a compatible data schema at runtime in code)

Bind the control to an existing DataSource now

I don't want to specify the data schema now - I'll just bind to a DataSource at runtime (Note: preview grids will display a dummy schema)

Alternative: No Schema in BindingSource at Design Time

- Do not add the BindingSource to the form
- Do not bind controls to a BindingSource
- Add code like this either :
 - In form constructor after InitializeComponent
 - in event handler for form's Load event

```
OPEN QUERY qry FOR EACH ttCust WHERE ttCust.State = "MA".  
bindingSource1 = NEW Progress.Data.BindingSource(  
    QUERY qry:HANDLE, "CustNum,Name,City,State", "") .  
ultraGrid1:DataSource = bindingSource1.
```


Comparing the Design-Time Alternatives

- Alternatives
 - Define schema via PBS wizard
 - Define schema via control's wizard
 - No schema at design time

- Design-time effort - 2 ends of the spectrum:
 - Setting schema thru the PBS may be easiest (DB, .xsd file)
 - No schema – no design-time work

- Effect on runtime efficiency
 - Given design-time schema, either PBS or control will compare it to runtime schema

Comparing the Design-Time Alternatives

- Effect on runtime efficiency
 - No PBS schema eliminates lots of calls from InitializeComponent

```
DEFINE VARIABLE tableDesc1 AS Progress.Data.TableDesc NO-UNDO.  
tableDesc1 = NEW Progress.Data.TableDesc("Customer").  
DEFINE VARIABLE arrayvar1 AS "Progress.Data.TableDesc[]" NO-UNDO.  
arrayvar1 = NEW "Progress.Data.TableDesc[]" (0).  
tableDesc1:ChildTables = arrayvar1.  
DEFINE VARIABLE arrayvar2 AS Progress.Data.ColumnPropDesc EXTENT 3 NO-UNDO.  
arrayvar2[1] = NEW Progress.Data.ColumnPropDesc("CustNum", "CustNum",  
Progress.Data.DataType:INTEGER).  
arrayvar2[2] = ... <one line like this for each field>  
tableDesc1:Columns = arrayvar2.  
THIS-OBJECT:bindingSource1:TableSchema = tableDesc1.
```

Agenda

- What is the ProBindingSource?
- What should you bind to?
- Design time setup for the BindingSource
- Internals
- The cursor
- Changing a query
- Inherited methods and properties
- Summary

Internals

- Control communicates to the PBS via methods and properties
 - Based on well-defined interfaces defined by Microsoft
 - We implement **IList**, **ITypedList**, **IBindingList** and **ICancelAddNew**
- The PBS does not know what is bound to it
 - The PBS communicates to the control via events
- The PBS is basically a slave to the control
 - It does not know what the control is doing
 - Doesn't know the context; why is it being asked for data
 - It just responds to requests

Internals

- Controls do not cache field data
 - Ensures that the most recent data is always displayed
- Controls often do cache one or more row objects
- It asks for field values from a row over and over and over
 - Paints
 - Scrolling
 - Editing
 - Internal sorting

ProBindingSource Rows

- Control gets a count of rows from the PBS
 - If this is not accurate, things will not work right
 - Use `pbs:MaxDataGuess` for large result sets
- It requests row objects based on the count
- A PBS row object is a `Progress.Data.DummyRow`

Progress.Data.DummyRow

- A typical .NET data source: **System.Data.DataTable** object
 - **System.Data.DataRow** contains the data
- For ABL we don't store data in .NET
 - DummyRow contains:
 - an Index of the row's position in the query result set
 - the underlying data source (e.g., which query) it belongs to
 - This enables us to get the real data from the ABL table
- The control asks the BindingSource for field values from a row object
 - It doesn't know what the row object is
 - It doesn't know how to get column values from it

Agenda

- What is the ProBindingSource?
- What should you bind to?
- Design time setup for the BindingSource
- Internals
- The cursor
- Changing a query
- Inherited methods and properties
- Summary

The Cursor

- In ABL, we cannot get data without positioning the cursor
- Different from a typical data source in .NET
 - Data is accessed directly; Like an array
- Cursor is always positioned so buffer contents match the selected row in the UI
 - Matches pbs:Position

```
PROCEDURE pbsPositionChanged:
```

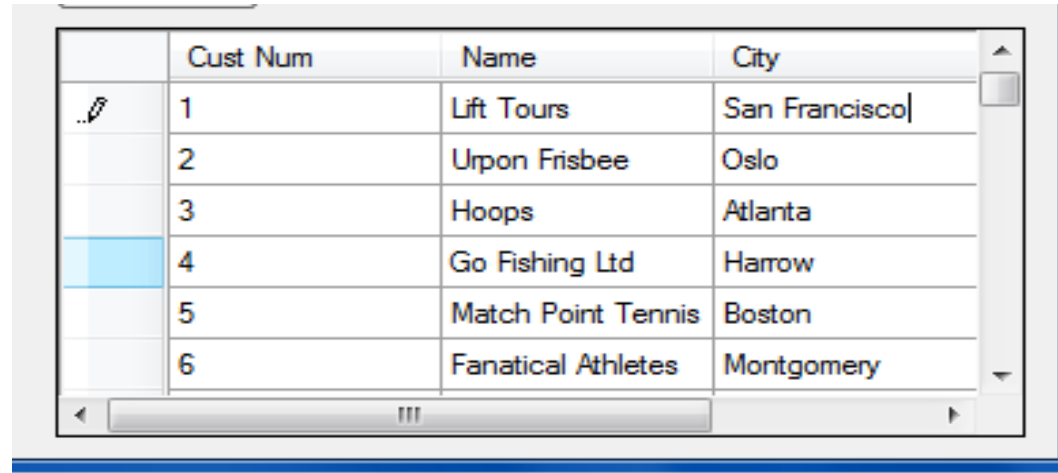
```
    DEFINE INPUT PARAMETER sender AS Progress.Data.BindingSource.
```


```
    DEFINE INPUT PARAMETER args AS System.EventArgs.
```

```
        MESSAGE Customer.Name VIEW-AS ALERT-BOX.
```

```
END.
```

The Cursor



	Cust Num	Name	City
	1	Lift Tours	San Francisco
	2	Urpon Frisbee	Oslo
	3	Hoops	Atlanta
	4	Go Fishing Ltd	Harrow
	5	Match Point Tennis	Boston
	6	Fanatical Athletes	Montgomery

```
USING System.Windows.Forms.*.
```

```
PROCEDURE RowValidating:
```

```
    DEFINE INPUT PARAMETER sender AS DataGridView.
```

```
    DEFINE INPUT PARAMETER args AS DataGridViewCellCancelEventArgs.
```

```
    MESSAGE Customer.City VIEW-AS ALERT-BOX.
```

```
    pbs:Assign().
```

```
END.
```

Example: CustomUnboundColumnData event

- DevExpress.XtraGrid has an event called CustomUnboundColumnData
- Used to display calculated columns
- Grid needs to paint all rows – not just the selected row
- Most of the time the row in your ABL buffer will NOT match the row of interest

Example: CustomUnboundColumnData event

```
USING DevExpress.XtraGrid.Views.Base.*.
METHOD PRIVATE VOID gridView1_CustomUnboundColumnData (
    INPUT sender AS System.Object,
    INPUT args AS CustomColumnDataEventArgs ):

    DEFINE VARIABLE dummyRow AS Progress.Data.DummyRow NO-UNDO.
    DEFINE VARIABLE rowIndex AS INTEGER.

    dummyRow = CAST ( args:Row, Progress.Data.DummyRow ).
    rowIndex = args:ListSourceRowIndex.

    ...
```

Example: Getting Field Data in an Event Handler

- The control expects you to get data from the row object (args:Row)
- There is an indexed property on this:
 - dummyRow:Item[columnName]
 - This is for another purpose; it will not give you the field value

```
DEFINE VAR colDescs AS System.ComponentModel.PropertyDescriptorCollection.  
DEFINE VAR colDesc AS Progress.Data.ColumnPropDesc.  
DEFINE VAR nm AS CHAR.  
DEFINE VAR ix AS INTEGER.  
  
colDescs = pbs.GetItemProperties ( ... )  
ix = args.Column.AbsoluteIndex.  
colDesc = CAST(colDescs[ix], Progress.Data.ColumnPropDesc) .  
nm = colDesc.GetValue (dummyRow) .
```

Example: Getting Field Data in an Event Handler

- Think simple!

Get the data from where it lives – in the temp-table

```
qryHdl : REPOSITION-TO-ROW (args :RowIndex + 1) .  
qryHdl : GET-NEXT .  
nm = Customer.Name .
```

Does REPOSITION Interfere With Control?

- AutoSync = true
 - OPEN-QUERY, REPOSITION-TO-ROW, REPOSITION-TO-ROWID will sync with the PBS
 - pbs:Position will change: PositionChanged event will fire
 - The current selection in UI will change
 - qryHdl:GET-NEXT is unnecessary
 - **Turn off AutoSync or use a “shadow” query: same criteria, different buffer**
- AutoSync = false:
 - Changing cursor won't interfere with control
 - It never asks for data from the **current** row – it always provides a row #
 - You could theoretically affect your own event handlers
 - **Reposition back when done or use a shadow query**

Agenda

- What is the ProBindingSource?
- What should you bind to?
- Design time setup for the BindingSource
- Internals
- The cursor
- Changing a query
- Inherited methods and properties
- Summary

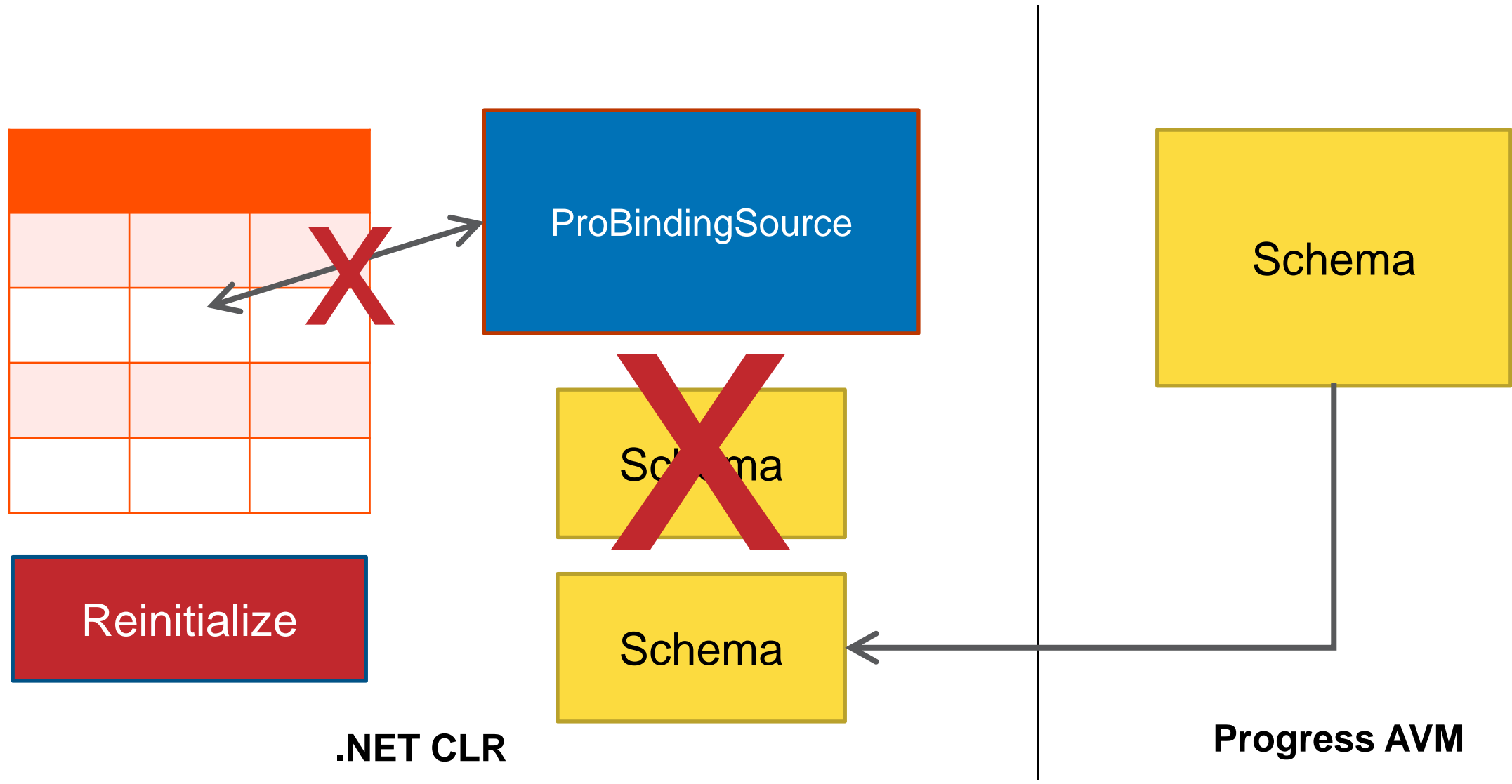
Changing the Query

- Example: User enters search criteria
 - Bound query WHERE clause changes
- Re-open the query in the ABL
 - If `pbs:AutoSync = true...` That's it!
 - Otherwise call `pbs:RefreshAll()`

The Wrong Way to Change the Query

```
pbs:HANDLE = ?.  
qryHdl:QUERY-CLOSE().  
ultraGrid:DataSource = ?.  
<... repopulate table(s) and/or  
  set new WHERE string...>  
qryHdl:QUERY-OPEN().  
pbs:HANDLE = myDataSet:TOP-NAV-QUERY.  
ultraGrid:DataSource = pbs.
```

Changing the Query



Temp-Table is Repopulated

- Example: Re-populate temp-table with different records
 - Bound query WHERE clause **remains the same**
- Re-open the query in the ABL
 - If pbs:AutoSync = true... That's it!
 - Otherwise call pbs:RefreshAll()
- The control needs an accurate count of records
 - Otherwise, it may ask for rows that don't exist
 - You get very strange behavior
- If query results in no qualifying rows – that's fine

Same as
before

Agenda

- What is the ProBindingSource?
- What should you bind to?
- Design time setup for the BindingSource
- Internals
- The cursor
- Changing a query
- Inherited methods and properties
- Summary

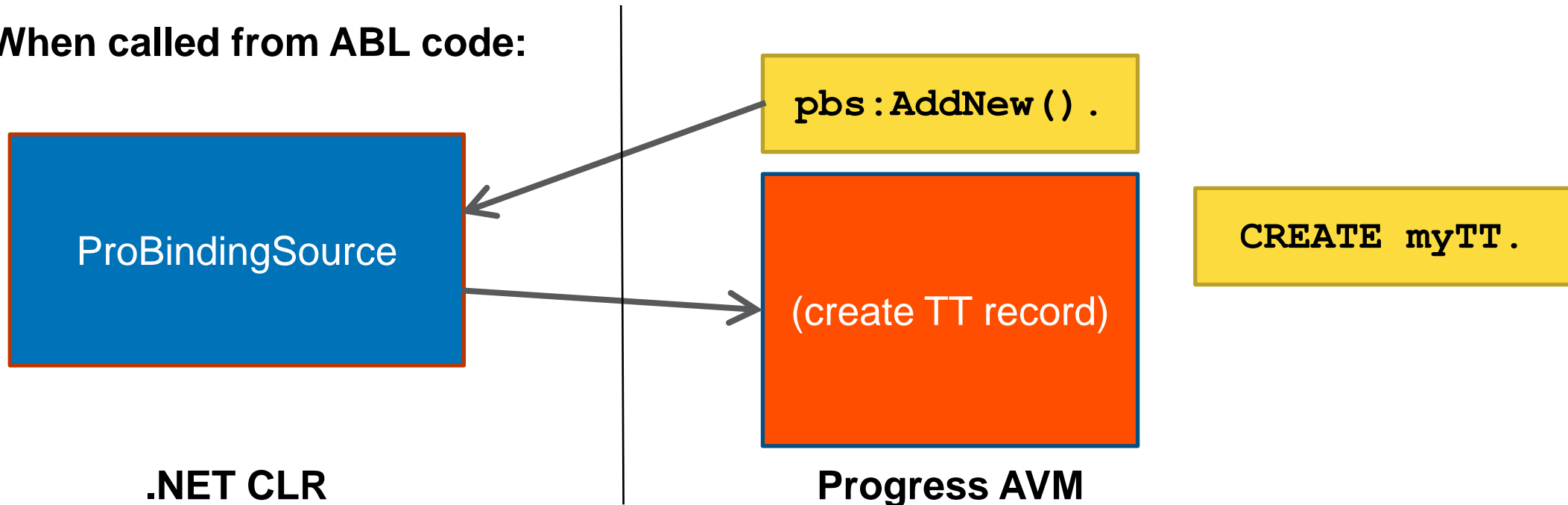
Methods and Properties

- We do Inherit from `System.Windows.Forms.BindingSource`
- We did add some of our own special sauce, e.g.:
 - Properties: `AutoSync`, `AutoUpdate`, `Batching`, `MaxDataGuess`, `InputValue`
 - Methods: `Assign`, `Refresh`
- Majority of data members are inherited
 - Some are designed to be used by the application, e.g.:
 - Properties: `Count`, `Position`, `AllowNew`, `AllowEdit`, `AllowDelete`
 - Methods: `Dispose`
 - Other data members are designed for use by the control

Misuse of Inherited Methods

- `pbs:AddNew()`, `pbs:Remove()`
 - Designed for use by control
 - When user opens a new row in a grid or deletes a row via the UI
 - If `pbs:AutoUpdate = true`, record is created/deleted in underlying table

- **When called from ABL code:**



Misuse of Inherited Methods and Properties

- Find()

```
ix = pbs:Find("Name", "Hoops").
```

- Throws NotImplementedException

- Filter property

```
pbs:Filter = "Name = 'Hoops'".
```

- Change the query instead and re-open it

Summary

- The ProBindingSource is just a conduit
 - It does not cache any data (except for 1 row at a time)
- Bind based on what you want to display
 - Query for one table; ProDataSet for hierarchy
- Options for using the ProBindingSource at Design time
 - Think about whether you need the schema
- Understand how the BindingSource interacts with the cursor
- Always re-open the query when record set changes
 - Don't ever set pbs:Handle = ?
- Use the ABL directly to access your tables
 - Don't use the ProBindingSource to read/update table data
 - Only use methods and properties we've documented



PROGRESS