

DataBinding in the OpenEdge GUI for .NET

Mike Fechner, Consultingwerk Ltd.

Consultingwerk Ltd.

- Independent Consulting Organisation
- Located in Cologne, Germany
- Customers in Germany, Europe and USA
- Specialized on OpenEdge platform
 - Dynamics®
 - ADM2
 - OERA
 - .NET™, OpenClient
 - OpenEdge GUI for .NET® 10.2A

Agenda

- Introduction
- DataBinding in the ABL
- Designing DataBinding in the Visual Designer
- Setting Binding Properties for typical Controls
- Updating Data using the ProBindingSource
- Issues

Introduction

- DataBinding is a key concept in .NET applications
- Used to display and update Data in a UI
- Supported by WinForms and WebForms
- Abstract view on the DataSource
 - DataSource usually shielded by BindingSource
- Flat or relational data

Introduction

- DataSource can be of various types:
 - Arrays of Objects, Lists, Collections, DataTable, DataSets
 - Anything that has a number of elements of the same type
 - Usually “in-memory-store” of data
- Visual Controls (TextBox, ComboBox, Grids) display data members (properties) of DataSource elements

Introduction

- Visual Control does not know the actual DataSource – knows just the BindingSource
- DataSource does not know the Visual Controls that display or modify it's data
- Multiple Controls may display / update the same record or field (sorry, already talking ABL here)
- BindingSource controls DataSource position (CurrencyManager), use multiple BindingSources if more than one row needs to be positioned at a time

Vocabulary

- BindingSource, ProBindingSource, Progress BindingSource, ...
 - Synonym for Progress.Data.BindingSource class
- DataSource
 - Object “delivering” Data to the BindingSource
 - (Query, ProDataset, Buffer)
- Binding Properties
 - Set for a Control via code or Visual Designer
 - Define which properties of a Control should be synchronized with a field of the DataSource

Agenda

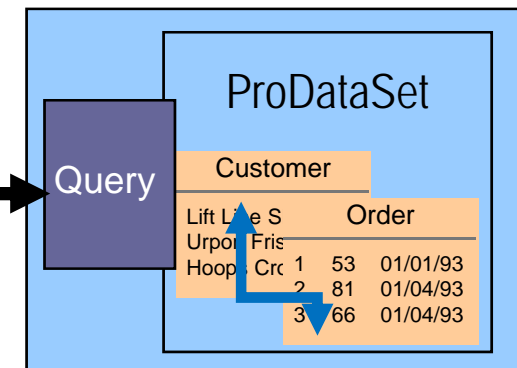
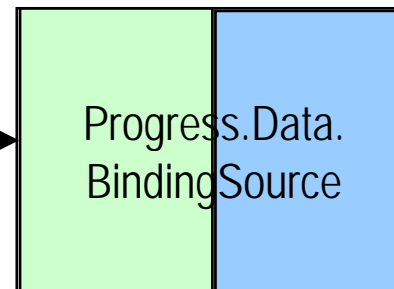
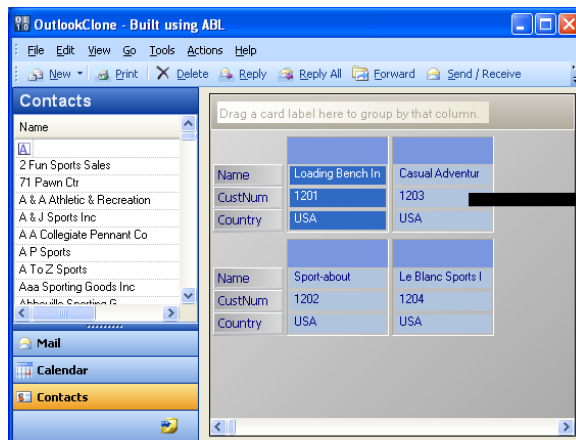
- Introduction
- **DataBinding in the ABL**
- Designing DataBinding in the Visual Designer
- Setting Binding Properties for typical Controls
- Updating Data using the ProBindingSource
- Issues

DataBinding in the ABL

- Progress has implemented a BindingSource
 - Progress.Data.BindingSource
 - .NET class, so it's name is case-sensitive
- Binds to Query, ProDataset or Buffer
- DataSource represented by *WIDGET-HANDLE*
- Documentation in “*OpenEdge Development*” collection, “*Advanced GUI Programming*” (~ 50 pages on DataBinding)

Data Binding in the OpenEdge GUI for .NET

- Progress.Data.BindingSource
 - Provides data for .NET Controls
 - Implements required APIs for .NET Controls (IList)
 - OpenEdge data provided as .NET needs
 - Any ABL Query (TT, TABLE), ProDataSet™



DataBinding in the ABL

- ProBindingSource required to display data using DataBinding
- Constructors of the Progress.Data.BindingSource:

```
PUBLIC BindingSource (HANDLE to DataSource Widget  
[, Include Fields, Except Fields])
```

```
PUBLIC BindingSource (DatasetHandle,  
[ParentBufferHandle or ParentBufferName,]  
[Include Fields, Except Fields])
```

```
PUBLIC BindingSource ()
```

- Visual Designer uses the last constructor. Set Handle property in your code, to set DataSource

DataBinding in the ABL

- BindingSource needs to “count” rows after query is opened
- Works faster for PRESELECT Queries (OPEN QUERY qCustomer PRESELECT EACH Customer)
- Don't strain your databases capabilities
- Batching may save system resources even on Client/Server systems

BindingSource properties (excerpt)

- AllowEdit, AllowNew, AllowRemove
- AutoSync (on OPEN-QUERY or REPOSITION)
- AutoUpdate (for rapid prototyping)
- Batching (activates OffEnd event handler)
- Count
- Handle
- NewRow
- Position
- RowModified

BindingSource methods (excerpt)

- Assign ()
- Refresh () – comparable to DISPLAY in ABL Frames
- RefreshAll ()

BindingSource events (excerpt)

- All events use parameter objects! Some properties are important to return to the binding source
- CreateRow
 - request new row, set key values
- CancelCreateRow
 - remove new row
- OffEnd
 - only fired when Batching = True
- PositionChanged
- SortRequest
 - not supported by UltraGrid

Why use DataBinding?

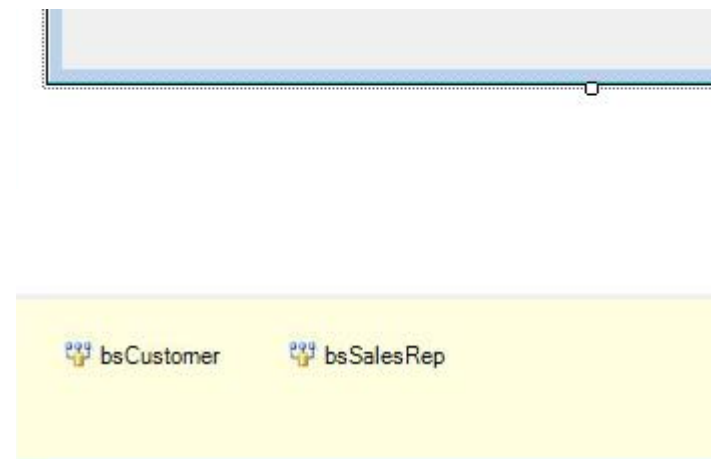
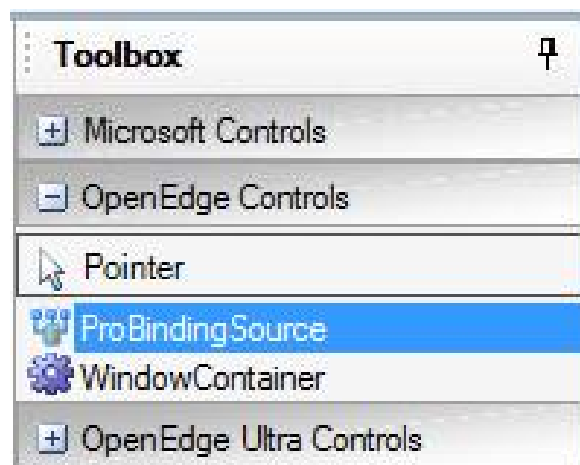
- For simple Controls the BindingSource does really nothing else than synchronizing selected properties of Visual Controls with fields in a DataSource (such as a DB table)
- This can be done manually in event handlers as well:
 - PositionChanged of the BindingSource
 - Validation (or something similar) of the visual Controls
- The key advantage of DataBinding vs. Coding are
 - Keep aligned with .NET samples available at Google
 - Define a large portion of behavior in Design, not code
- Grids usually require a BindingSource (of any kind)

Agenda

- Introduction
- DataBinding in the ABL
- **Designing DataBinding in the Visual Designer**
- Setting Binding Properties for typical Controls
- Updating Data using the ProBindingSource
- Issues

Designing DataBinding

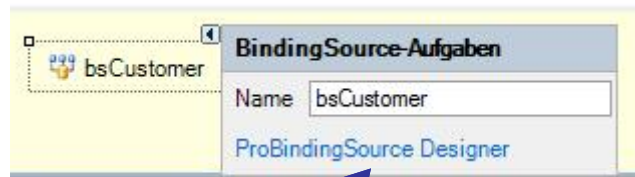
- ProBindingSource is a Component
 - So it can be dragged on the design canvas
 - Displayed in the non-visual Component area below the designed Form.
 - Use multiple Binding Sources as required



Designing DataBinding

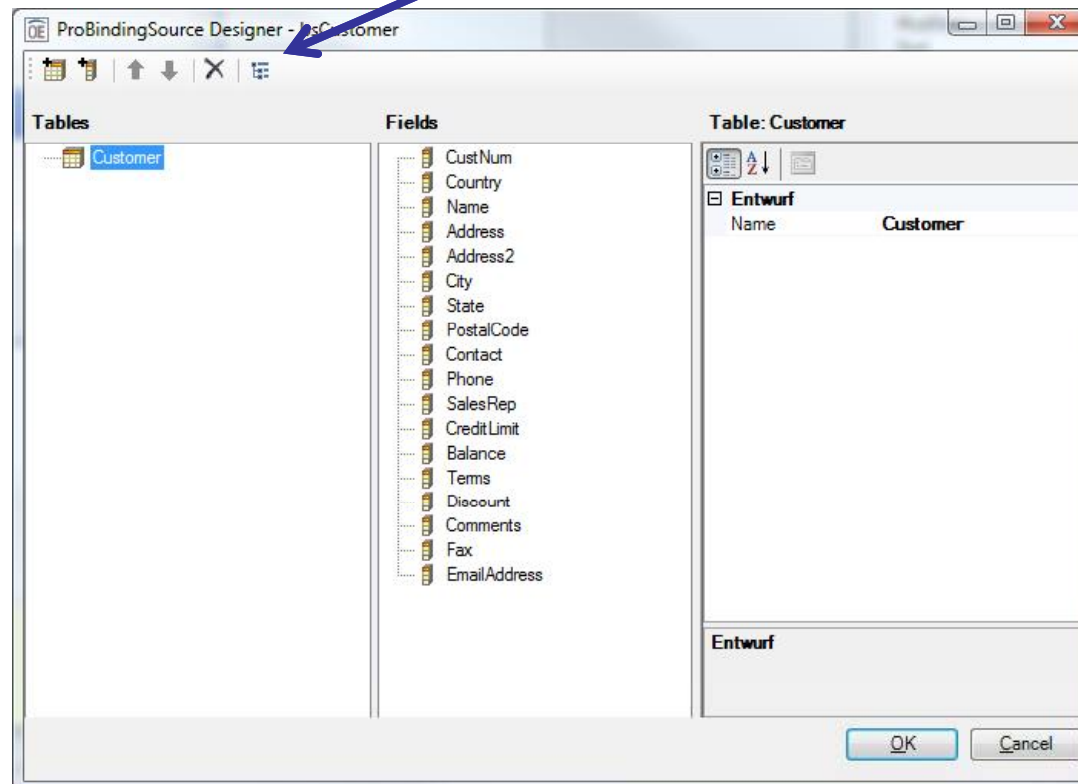
- Use BindingSource Designer to set schema
 - Access via Smart-Tag or Property Grid
 - Design time schema used to set DataBinding properties / design Grids / ...
 - Should match runtime schema
 - Import XSD Files generated using :WRITE-XMLSCHEMA method of Temp-Table/ProDataSet
 - Modify schema from XSD or design custom schema

Designing DataBinding



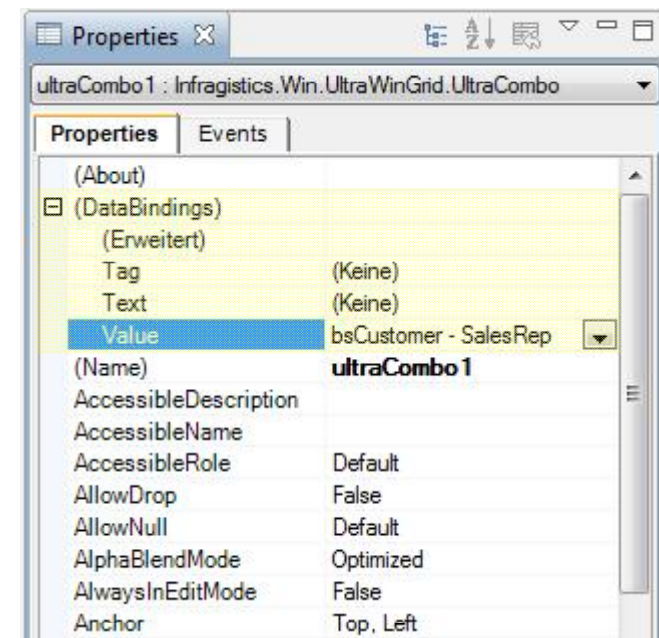
Click here to import generated XSD File

Click here to access ProBindingSource Designer



Designing DataBinding

- Setting Binding Properties in the Property Grid
 - Set required properties in the (DataBindings) section
 - Usually Text or Value, check Control documentation for details
 - Tag (PRIVATE-DATA as Object) is offered for each Control, don't pay too much attention to the Tag property



Agenda

- Introduction
- DataBinding in the ABL
- Designing DataBinding in the Visual Designer
- **Setting Binding Properties for typical Controls**
- Updating Data using the ProBindingSource
- Issues

Setting Binding Properties

- Due to different (Binding) capabilities of each Control the properties required for the right usage of DataBinding are different
- This has been criticized in the forum, but there's nothing Progress can do about it!
- Every change Progress would do about it, would risk incompatibilities between Controls and ABL
- The following slides show some guidelines, please consider the Control documentation for additional details

Binding „simple“ input Controls

- Controls that do not offer repositioning typically allow a developer to set (DataBindings) properties in the Visual Designer
- Text is always CHARACTER, Value may be type-safe
- TextBox: **Text**, Value
- Checkbox: **Checked**, Text, ...
- UltraCurrencyEditor, UltraNumericEditor: Text, **Value**
- UltraCalendarCombo: Text, **Value**
- (DataBindings) Text and Tag are inherited from System.Windows.Forms.Control – no need to be useful, the Control vendor can't hide these properties

DataSource property

- Controls that
 - Display more than one record
 - Offer re-positioning of the DataSourcetypically have a DataSource property that accepts a BindingSource
- This is the case for most Grids and controls such as ComboBoxes and ListViews that accept a binding source to fill the value list

Binding a DataGrid

- The Microsoft Grid has a Columns collection. A developer can define grid columns before setting a BindingSource
- The UltraGrid displays all Tables (Data-Relations) of the DataSource as bands (single table/band for Queries/Buffer, multi table/band for ProDatasets) and all columns for these bands by default
- A developer can hide columns/bands by setting them to Hidden (and hiding them from the ColumnChooser Control)

UltraCombo

- Set DataSource to a BindingSource returning the values (the detail table, e.g. SalesRep)
- The value list is displayed in a Mini-UltraGrid, design using Wizard, Hide unwanted Columns
- Set SynchronizeWithCurrencyManager (the BindingSource) = True to reposition the detail query and refresh other Controls connected to the same binding source on value change

UltraCombo

- Set ...Member properties to define fields of the binding source used in “closed” state of the combo
 - DisplayMember: RepName
 - ValueMember: SalesRep
- Set (DataBindings) Value to a second binding source (like a Customer BindingSource)
- By setting the Value to a SalesRep code, the UltraCombo will display the matching RepName
- Binding Value to Customer.SalesRep will ensure that RepName of the current customer is displayed

Agenda

- Introduction
- DataBinding in the ABL
- Designing DataBinding in the Visual Designer
- Setting Binding Properties for typical Controls
- **Updating Data using the ProBindingSource**
- Issues

Updating Data

- When the **AutoUpdate** is set to True the ProBindingSource handles the complete update process without a single line of code
- Error- and Transaction control is minimal
- According to the documentation the AutoUpdate shall only be used for “rapid prototyping”
- Alternative: use a Save button calling the **Assign()** property or appropriate events of the Controls used

Refreshing Data

- In cases where the ABL updates data (validation logic, trigger logic, calculations, ...) these changes are not reflected to the BindingSource and the bound Controls
- Use the **Refresh()** method of the BindingSource to refresh the current row/current record
- Use the **Refresh(n)** method to refresh the nth row in the DataSource (1 based index)
- Use the **RefreshAll()** method to refresh all rows in the DataSource

Agenda

- Introduction
- DataBinding in the ABL
- Designing DataBinding in the Visual Designer
- Setting Binding Properties for typical Controls
- Updating Data using the ProBindingSource
- Issues

UltraGrid and Binding Performance

- No .NET copy of ABL Data is created, data is passed across the bridge as needed
- Some operations (the cool features!) in the UltraGrid require a large amount of data
 - Grouping
 - Summaries / Totals / Averages / Filter combo
 - Performance acceptable with 5.000 – 10.000 rows – depending on various system resources

Creating Rows and Unique Indexes

- The CreateRow event handler fires, when a Visual Control requests the creation of a new row
- We have to create a new row in the DataSource (Query, ProDataset) and add it to the result list of the query. This causes the new row to be written to the DataSource (which can be a DB table!)
- This may cause unique key violation!

Questions



Thank You

Consultingwerk Dynamics 4net