

# DEV-40: Using SmartDataObjects (SDO) with the Advanced GUI

Mike Fechner, Consultingwerk Ltd.

Progress Exchange 2008, Paris, France



## Consultingwerk Ltd.

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

# Agenda



- Introduction
- SmartDataObjects Reviewed
- Data Binding in the Advanced GUI
- Basic Data Binding with SmartDataObjects
- Controlling Batching and Sorting
- Updating
- Summary

## Introduction

- Consultingwerk has developed a replacement for the Dynamics rendering engine
- Dynamics4.NET is based on the Advanced GUI for OpenEdge 10.2A and Dynamics and offers “load and go” migration to the .NET user interface
- Compatibility between .NET UI and ADM2 major design goal to support migration of business logic as well as repository information

# Agenda



- Introduction
- SmartDataObjects Reviewed
- Data Binding in the Advanced GUI
- Basic Data Binding with SmartDataObjects
- Controlling Batching and Sorting
- Updating
- Summary

## SmartDataObjects Reviewed

- Important part of ADM2 Architecture
- Business Logic component, data retrieval and update
- Introduced in V9.0, continuously enhanced in V9.1, OpenEdge 10
- Built using
  - persistent procedures, super procedures
  - dynamic queries
  - temp-tables

## SmartDataObjects Reviewed

- Usage in ADM2 applications driven by wizards (AppBuilder supported)
- Usage outside ADM2 applications possible and documented (“Open SDO” white paper from John Sadd available on PSDN)

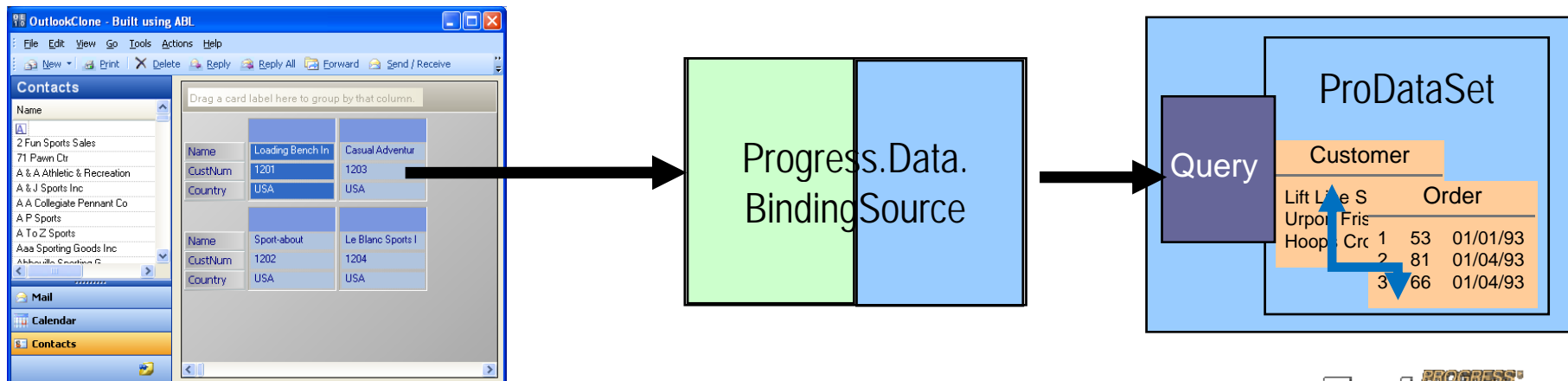
# Agenda



- Introduction
- SmartDataObjects Reviewed
- **Data Binding in the Advanced GUI**
- Basic Data Binding with SmartDataObjects
- Controlling Batching and Sorting
- Updating
- Summary

## Data Binding in the Advanced GUI

- 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™



## Data Binding in the Advanced GUI

- ABL behavior
  - Automatic data synchronization
  - Automatic sorting
  - Automatic updating
  - Automatic batching
  - Automatic currency
  
- Properties
  - AllowEdit, AllowNew, AllowRemove
  - NewRow, RowModified
  - Position, Count
  - InputValue, ChildInputValue

## Data Binding in the Advanced GUI

- Simple .NET control ( UltraEdit )

```
editBox:DataBindings.Add ( "Text", pbs, "OrderNum" ).
```

- .NET table-like control ( UltraGrid )

```
grid:DataSource = pbs.
```

- .NET list control ( UltraListView )

```
list:DataSource = pbs.  
list:DataTextField = "State".  
list:DataValueField = "State-Name".
```

# Agenda



- Introduction
- SmartDataObjects reviewed
- Data Binding in the Advanced GUI
- **Basic Data Binding with SmartDataObjects**
- Controlling Batching and Sorting
- Updating
- Summary

## Data Binding with SmartDataObjects

- Use similar concepts as SmartDataBrowser
- Bind .NET Controls to ProBindingSource (Progress.Data.BindingSource) control
- Get the **DataHandle** of the SDO
  - the query to client side RowObject temp-table  
{get DataHandle hDataHandle hSDO}
- Open the SDO Query (if not done at initialization)
- Assign the **HANDLE** Property of the ProBindingSource control

## Sample without Batching

```
METHOD PRIVATE VOID InitializeDataObject ( ):
    RUN samples/SDOForm/dcustomer.w PERSISTENT SET hSDO .

    {set OpenOnInit True hSDO} .
    {set RowsToBatch 0 hSDO} .

    RUN initializeObject IN hSDO .

    {get DataHandle hDataQuery hSDO} .

    bindingSource1:HANDLE = hDataQuery .

    bindingSource1:Batching = FALSE .

END.
```

Launch SDO as  
persistent procedure

Get handle to RowObject  
query and assign to  
Binding Source

# Sample without Batching

Drag Country and State column headers to group by area...

SDOForm

Drag a column header here to group by that column.

Cust Nu /	Name	Country	State	Address	Ad
1	Lift Tours	USA	MA	276 North Drive	
2	Urpon Frisbee	Finland	Uusima	Rattipolku 3	
3	Hoops	USA	GA	Suite 415	40
4	Go Fishing Ltd	United Kingdom	Middlesex	Unit 2	83
5	Match Point Tennis	USA	MA	66 Homer Pl	Adc
6	Fanatical Athletes	United Kingdom	AL	20 Bicep Bridge...	Ley
7	Aerobics valine Ky	Finland	Uusimaa	Peltolantie 3	
8	Game Set Match	USA	AL	Box 60	
9	Pihtiputaan Pyora	Finland	Etela-Savo	Putikontie 2	
10	Just Joggers Limited	United Kingdom	Sussex	Fairwind Trading...	Shc
11	Keilailu ja Biljardi	Finland	Uusimaa	Vattuniemenkuja...	
12	Surf Lautaveikkoset	Finland	Varsinais-Suomi	Venemestarinkat...	
13	Biljardi ja tennis	Finland	Uusimaa	Urheilutie 1	
14	Paris St Germain	France	Seine	113, avenue du...	BP
15	Hoopla Basketball	USA	NJ	87 Calumnet St	

# Sample without Batching

... to group SDO  
resultset by  
Country and State

The screenshot shows a window titled "SDOForm" with a tree view on the left and a table on the right. The tree view is expanded to show "State : Bretagne (2 items)". The table below shows the data for these two items.

Cust Nu /	Name	Address	Address2	City
36	La voile en folie	rue de l'Ocean	allee de Paris	Brest
77	Le Hollandais volant	35, rue des Oise...	3eme etage	Rennes

## Designing the Form

- Assigning the **HANDLE** property of the ProBindingSource creates schema information during run time
- To „design“ the fields in the UltraGrid the schema information is required at design time
- The ProBindingSource designer can import an XML schema definition (XSD)
- XSD can be generated from temp-table
- **RowObject** temp-table contains custom fields (SDO include file) and ADM2 control fields

# Generating XML Schema Definition

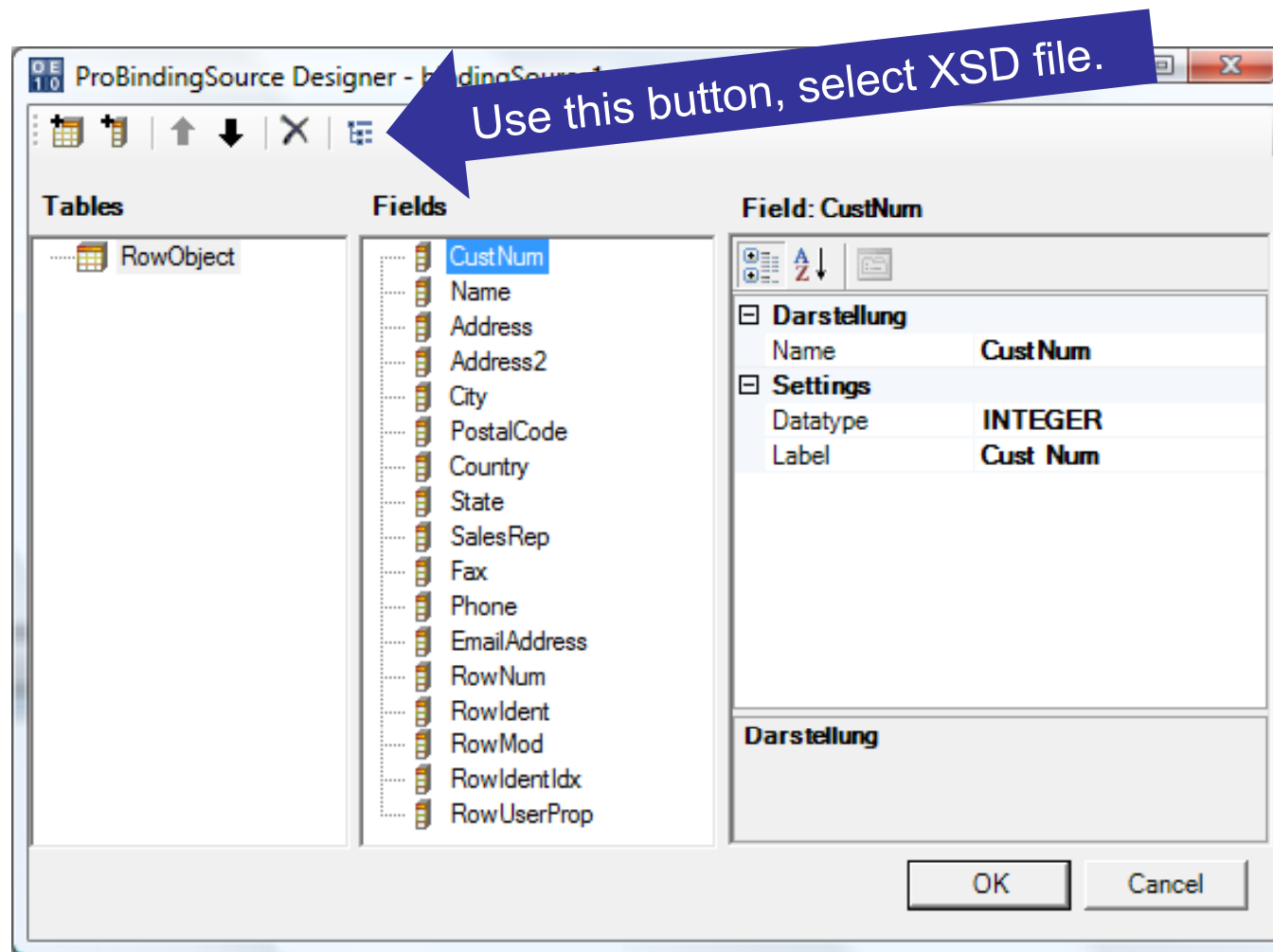
```
&SCOPED-DEFINE DATA-FIELD-DEFS {samples/SDOForm/dcustomer.i}

DEFINE TEMP-TABLE RowObject NO-UNDO
  {&DATA-FIELD-DEFS}
  {src/adm2/robjflds.i}
  .

/* ***** Main Block ***** */

TEMP-TABLE RowObject:WRITE-XMLSCHEMA ("FILE", "samples/SDOForm/dcustomer.xsd",
                                     TRUE, /* formatted */
                                     ?, /* Default encoding */
                                     FALSE /* no min schema */).
```

# Importing the Schema from XSD



## Demo



- Creating the RowObject XSD file
- Importing the XSD file into ProBindingSource designer
- Modify design time properties of UltraGrid
- Launch form

# Agenda



- Introduction
- SmartDataObjects Reviewed
- Data Binding in the Advanced GUI
- Basic Data Binding with SmartDataObjects
- **Controlling Batching and Sorting**
- Updating
- Summary

## Support for Batching

- Batching is the process of browsing a large result set in small, reasonable chunks
- The primary goal of Batching is to save system resources (db server load, network bandwidth, client memory)
- Batching has been a major design goal for SmartDataObjects (RowsToBatch property)
- Batching is a major feature of the ProBindingSource object as well

## Enabling Batching in the SDO

- Enabling Batching in SmartDataObjects is easy
- Just set the **RowsToBatch** property to a value greater than 0 (200 is the default)
- While opening the query (i.e. during initialization) the SDO will read the first chunk of rows and enable fetching of the next chunks on request

```
{set OpenOnInit True hSDO} .  
{set RowsToBatch 75 hSDO} .  
  
RUN initializeObject IN hSDO .
```

## Enabling Batching in the Binding Source

- Enabling Batching in ProBindingSource is easy
- Just set the **Batching** Property to True

```
bindingSource1:HANDLE = hDataQuery .
```

```
bindingSource1:Batching = TRUE .
```

- The ProBindingSource will now invoke the **OffEnd** event handler when the UltraGrid control requests missing data (at the 76. row)

## Reading the next chunks on OffEnd

- The OffEnd event is invoked when the UltraGrid requests more data from the ProBindingSource
- The SDO can read additional data while the property LastRowNum is ?  
(not yet at the last chunk of data)
- The fetchBatch method of the SDO reads the next batch (SDO maintains all context!)
- ProBindingSource needs to know the number of records added
- Set RowsAdded property of the event argument

## OffEnd event handler

```
METHOD PRIVATE VOID bindingSource1_OffEnd
( INPUT sender AS System.Object, INPUT args AS Progress.Data.OffEndEventArgs ):

/* Test for last batch. When LastRowNum is set, the SDO has read
the last batch already. */
{get LastRowNum iLastRowNum hSDO} .

IF iLastRowNum = ? THEN DO:
    {get LastResultRow cLastResultRow hSDO} .

    ASSIGN iLastRow = INTEGER(ENTRY(1, cLastResultRow, ";")) .

    /* Invoke SDO fetch routine */
    RUN fetchBatch IN hSDO
        ( TRUE ).

    {get LastResultRow cLastResultRow hSDO} .

    /* Calculate number of rows read */
    ASSIGN iNewLastRow = INTEGER(ENTRY(1, cLastResultRow, ";")) .

    /* Tell binding Source */
    args:RowsAdded = iNewLastRow - iLastRow .

END.
END METHOD.
```

## Demo



- Reading data from the SDO in batches

## Enable sorting through Grid events

- Batching requires sorting of data in the SDO
- Most .NET grid controls (like UltraGrid) have built in sort capabilities
- But the grid may have only partial data (Batching)
- Sorting the first 200 rows by CustNum by a different column is nonsense (unpredictable)
- Sort order of character data might be different between database and .NET Controls (Umlaute)
- .NET Controls need to execute SDO logic to enable proper sorting in the database

## Enable sorting through Grid events

- Sorting behavior is dependent on the .NET Grid
- Some grids rely on the sort capabilities of the ProBindingSource object (Microsoft's DataGridView)
- This will result in execution of the **SortRequest** event handler of the Binding Source
- The UltraGrid uses it's own event logic to intercept sort requests
- Use the **BeforeSortChange** event handler of the UltraGrid

## Demo



- Review sort event handler code
- Demo of sorting in the UltraGrid

# Agenda



- Introduction
- SmartDataObjects Reviewed
- Data Binding in the Advanced GUI
- Basic Data Binding with SmartDataObjects
- Controlling Batching and Sorting
- **Updating**
- Summary

## Updating Data in the SDO

- SmartDataObject has strict interfaces for updating data
- Two temp-tables are used internally for before-image and current values: **RowObject** and **RowObjUpd**
- Manipulation of RowObjUpd rows is done internally in the SDO (function **submitRow**)
- Update based just on APIs and modified field lists
- Create and delete have different APIs

## Updating Data in the Binding Source

- Similar to sorting there is no standard behaviour to manage data updates between the Binding Source and connected visual control(s)
- Even with similar concepts, most Grid controls use different events for updates and validation
- The UltraGrid offers two events for updating
  - **BeforeRowUpdate**
  - **AfterRowUpdate**
- **BeforeRowUpdate** is cancelable

## Updating Checklist

- Hook into validation event – before/while control expects data to be changed
- Build list of changed fields by comparing values in the control and the current **RowObject** row
- Call submitRow (RowIdent, ChangedValues)
- In case of an error (validation inside SDO) submitRow returns FALSE
  - Use **showDataMessagesProcedure** to display error message to the user
  - Cancel update event: **e:Cancel = TRUE**

## Demo



- Code review, update event handler
- Updating data
- Handling validation error messages

# Agenda



- Introduction
- SmartDataObjects Reviewed
- Data Binding in the Advanced GUI
- Basic Data Binding with SmartDataObjects
- Controlling Batching and Sorting
- Updating
- **Summary**

## Summary

- SmartDataObjects are compatible with the Advanced GUI for OpenEdge 10.2A
- Manual coding required
- Most coding reusable in components (OO or procedural)
- SmartDataObjects are still the most up to date data retrieval and business logic component supported by OpenEdge tools!

## Related sessions

- DEV-6: Introduction to the OpenEdge Advanced GUI, Jim Lundy
- PSDN: Introduction to the OpenEdge Advanced GUI Webinar on Tuesday July 15 on [psdn.com/library/entry.jspa?externalID=5393](http://psdn.com/library/entry.jspa?externalID=5393)

# Questions



# Thank You

# Consultingwerk Dynamics 4net