UPDATING DATA WITH .NET CONTROLS AND A PROBINDINGSOURCE

John Sadd Fellow and OpenEdge Evangelist Document Version 1.0 March 2010





DISCLAIMER

Certain portions of this document contain information about Progress Software Corporation's plans for future product development and overall business strategies. Such information is proprietary and confidential to Progress Software Corporation and may be used by you solely in accordance with the terms and conditions specified in the PSDN Online (http://www.psdn.com) Terms of Use (http://psdn.progress.com/terms/index.ssp). Progress Software Corporation reserves the right, in its sole discretion, to modify or abandon without notice any of the plans described herein pertaining to future development and/or business development strategies. Any reference to third party software and/or features is intended for illustration purposes only. Progress Software Corporation does not endorse or sponsor such third parties or software.

This paper accompanies a two-part presentation that extends other materials on data binding and using the ProBindingSource control, by showing you simple ways to handle update operations. Here I introduce more of the remaining **ProBindingSource** properties, methods and events that support updates, and a few of the basic UI control events that you can interact with as well. As you can see here in the **Resources** View, I've created a new subdirectory in my project called **Updatable** where I'll put my new code samples:

📴 OpenEdge Editor - OpenEdge Architect - C:\Gui4Dothet	
File Edit Navigate Search Project OpenEdge Run Window Help	
12 • 12 △ ∡山 20 歩 • 0 • 24 • タ • 2 • 21 • 1 + + + + + +	
🗈 🖧 OpenEdge Editor	
🗱 Resources 🖂 🧧 🗖	
🗏 🛱 Samples 🔺	
B 30 Referenced Assemblies	
B-JL Procedure Libraries	
🖲 🗁 rcode	
E C TreeView	
, propath	
- appModel.74bi	
- B assembles.xml	
1 3 × V	
Property Value	
derive: false	
editabil true	
last mc January 14, 2010	
locator C/QuPDotNet/Sa	
name Updatable	
path /Samples/Updatable	
🕒 Console 🔝 Problems 22 🖉 Tasks	*□
T* E Samples/Lipdatable	

That directory corresponds to the notion of a package, in object-oriented terms, for organizing the code in my project. To show you how to specify the package when you create new source files, I start by creating a new ABL Interface. Because I had selected the **Updatable** folder in the **Resources View**, that package name is already filled in for me as the default. As I've done before, I can just enter the name of the interface. This is a variation on the **IModel** interface that I've used before, so I can use the same name, because it will be stored in a new folder with all the rest of my sample code for trying out update operations:

😭 OpenEdge Editor - OpenEdge Architec	t - C:\Gui4DotNet		_1012
File Edit Navigate Search Project Ope	nEdge Run Window Help		
] 📬 • 🖫 🛆] 🔏 📖 🕲] 🏇 •	(Q • Q ₄ •] A ² •] ≥ × ≥ × + + + + +		
🛅 💱 OpenEdge Editor	📧 New ABL Interface	_ 🗆 🗶	
🗱 Resources 🗶 🖳 🗖	Create a user-defined interface		
	Optionally enter a description for the interface. This text will appear in the header.	fie I	
E Samples			
B-JL Procedure Libraries	Package root: \Samples	Browse	
B Contraction	Package: Updatable	Browse	
E D Updatable	Interface name: []Model		
- in .doconnection	Description:		
		-	
appModel.t4bl			
assembliesml			
950 500 50 57 50		<u>×</u>	
	Purpose:	~	
Property Value			
🗆 Info			
editabil true		-	
last mc January 14, 2010			
location C:\Gui4DotNet\Sa			
name Updatable			
path /Jamples/Updatable	(V) Ens	Cancel	
	Console 🖹 Problems 💠 🕢 Tasks		* ~ - 0
Samplar () indatable			

I start with the Interface code skeleton that Architect generates, paste in the three method definitions used in the read-only operations created in other sessions on the **ProBindingSource**, and then add a new method to support saving data, which takes a buffer name as a parameter, and returns a **LOGICAL** value to tell me whether the save succeeded or not.

USING Progress.Lang.*.
INTERFACE Updatable.IModel:
METHOD PUBLIC VOID FetchData (INPUT pcFilter AS CHARACTER).
METHOD PUBLIC VOID SortData (INPUT pcSort AS CHARACTER).
METHOD PUBLIC HANDLE GetQuery().
METHOD PUBLIC LOGICAL SaveData(INPUT pcBufferName AS CHARACTER).
END INTERFACE.

Next I create a new class based on the **CustomerModel** class, and place it into my **Updatable** package. The first change I need to make is to qualify the name of the interface it implements so that the compiler finds the right one.

CLASS Updatable.CustomerModel IMPLEMENTS Updatable.IModel:

Then I need to make a change to the **FetchData** method that tells the model class to populate its ProDataSet. A ProDataSet can keep track of changes you make after it's been filled with data from its data source. But before I fill it I have to make sure that the **TRACKING-CHANGES** property on the **ttCustomer** temp-table is set off. Then after the **FILL** is done, I can set the property to **True** so that any changes made from that point on will be tracked in the **ttCustomer** temp-table's before-table:

```
METHOD PUBLIC VOID FetchData( INPUT pcFilter AS CHARACTER ):
DEFINE VARIABLE cPrepare AS CHARACTER NO-UNDO.
TEMP-TABLE ttCustomer:TRACKING-CHANGES = FALSE.
cPrepare = "FOR EACH AutoEdge.Customer".
IF pcFilter NE "" THEN
cPrepare = cPrepare + " WHERE " + pcFilter.
QUERY qCustomer:QUERY-PREPARE (cPrepare).
DATASET dsCustomer:FILL().
httCustQuery:QUERY-OPEN ().
TEMP-TABLE ttCustomer:TRACKING-CHANGES = TRUE.
END METHOD.
```

Now I'm ready to accept updates from the user interface. Remember that the **Source** menu in Architect helps you make all sorts of edits to a source file, including new methods, constructors, and event and property definitions for a class. I need to add a new method, but because it's a method defined in my interface **IModel**, I can select the **Override / Implement Members** option to implement the **SaveData** method that I defined in the **Updatable** version of the Interface. I have Architect add it to the source file in alphabetical order, and tell Architect to generate an empty comments block at the top of the method.

00 OpenEdge Editor - Samples/Upda	table/Customer	ModeLcls - OpenEdge Architect - C:\Gui4DotNet	_0×
File Edit Source Navigate Search	Project OpenEdg	e Run Window Help	
] 🗈 • 🔛 🗅] 🔏 💷 🦦] 🕸	🌣 • 🕡 • 💁	0E Override/Implement Members	
📑 💱 OpenEdge Editor		Generate stubs to override or implement members	
St Resources 😂 📃 🗆	Model.cls	Select this option to include a comment block with each generated	
	298 CC	member.	-
C 19 Survey	30		_
Referenced Accembles	31	Select members to overnde or implement: Customer: HANDLE) .	
B-JL Procedure Libraries	32	Select All	
🖲 🗁 OOSamples 📃	33	Constant all	
🗄 😂 rcode	35 EN	SaveData(INPUT CHARACTER):LOGICAL	
E C TreeView	3.6		
Controvertificated de	37		
CustomerUkraGrid.	380 ME	():	
IModel.ds	39		
- dbconnection	40 DE		
	42		
🚼 O 📅 D 🗔 P 🐹 🖓 🗆	43		
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	44		
Property Value	45	Insertion position : Alphabetical order	
⊡ Info	40		
derived false	48	Generate comments	
editable true	49 EN	Salast which style of member code you would like to constant	
linked false	50		- 10
location C:\Gui4DotNet\Sa	S18 ME	Generace exceptions for required members	- 10
name CustomerModel.cl	52 /1	Generate default values for required members	
path /Samples/Updatab	54	e pacasec/	
, see 0, 999 bytes	55 EN		- 10
	56		-
	•	Generate Cancel	<u> </u>
N N	🕒 Console 🖾		9 - 🗆 🗖
		mash true 17.1	
10 0		wracee useft 5715	

To the default code I start out with, as generated by Architect, I first add a couple of variable definitions to point to the temp-table's buffer, and its before-buffer, where changes are kept track of:

```
METHOD PUBLIC LOGICAL SaveData( INPUT pcBufferName AS CHARACTER ):
DEFINE VARIABLE hCustBuffer AS HANDLE NO-UNDO.
DEFINE VARIABLE hBeforeBuffer AS HANDLE NO-UNDO.
```

Also, I put in a sanity check to make sure that the buffer name passed in is the right one. In this case there's only one temp-table in the DataSet, but in other cases there could be more than one, which is why the buffer name parameter is here:



And then I initialize the two buffer handle variables:

hCustBuffer = BUFFER ttCustomer:<mark>HANDLE</mark>. hBeforeBuffer = hCustBuffer:BEFORE-BUFFER.

I want to have some simple logic in the Model to give me a way to show what happens if the user enters invalid data. My table has **CustomerBirthCountry** values of only the USA and Germany, so my check says that if any other value is entered, I reject the change to the temp-table, and return false to signal the error to the View:

```
IF ttCustomer.CustomerBirthCountry NE "USA" AND
   ttCustomer.CustomerBirthCountry NE "Germany" THEN
DO:
    MESSAGE "Invalid Birth Country value, must be USA or Germany."
        VIEW-AS ALERT-BOX.
    hBeforeBuffer:REJECT-ROW-CHANGES ().
    RETURN FALSE.
END.
```

Note that the Model class expects the changes the user has made to have been saved to the DataSet's temp-table already. This is important because you don't want even simple logic like this in the View, which just handles the user interface. And you don't want the Model, which is in charge of the data, to know how to look up into the UI and see the values in the user interface controls. So the UI has to get the values assigned to the temp-table before it runs **SaveData**, and to do this, it uses the ProBindingSource as an intermediary. If there's no error the Model uses the ProDataSet's **Save-Row-CHANGES** method to save data back to the DataSource, the database table it came from. **Save-Row-CHANGES** itself could fail, because it does conflict checks if more than one user is changing the same row at the same time, and there might be underlying database triggers that could fail as well. So if **Save-Row-CHANGES** succeeds, the **ACCEPT-ROW-CHANGES** method keeps the changes in the temp-table and marks the change as complete, and I tell the UI that the save succeeded:

```
/* If we get here all client-side validation succeeded. */
    IF hBeforeBuffer:SAVE-ROW-CHANGES () THEN
    DO:
        hBeforeBuffer:ACCEPT-ROW-CHANGES ().
        RETURN TRUE.
    END.
```

But if **SAVE-ROW-CHANGES** returns an error, then **REJECT-ROW-CHANGES** scrubs the change from the temptable, and the method returns an error flag.

```
ELSE DO:

hBeforeBuffer:REJECT-ROW-CHANGES ().

RETURN FALSE.

END.

END METHOD.
```

That's a start to the changes to the Model class to handle the data management side of the update. Now let's look at the Form where we'll also make changes, which once again is based on a form used in the earlier data binding examples. If I select its **ProBindingSource** control, I can show you a few more of its properties. The **AllowEdit** property, for instance, determines whether updates through the binding source are enabled or not. You can also enable individual controls in the user interface, but this property lets you control updates programmatically from the binding source. It's true by default, so updates are enabled. but here I set it to false to see how it affects the user interface.

DenEdgeEditor - Samples/Updatable/	CustomerUltr OperEdge	aGridUpd.cls - O Run Window He	penEdge Archit	ect - C:\Gui4DotN	iet				1	-1012
□ · □ △ ∠ □ ⊗ ∞ · €		A + 1 2 +		•						
📑 💐 OpenEdge Editor										
Resources 😫 🔍 🗆	I 🔄 IModel.	ds 💽 Custo	merModel.cls	SustomerUB	aGridUpd.cls (Desig	n) 🛙				00
	7			1.				11	oolbox	9
E 🖉 Samples	Cu:	tomerUltraGrid	Upd						CustomControls	
B Professional Assemblies	1 —								CustomControls	-
Image: Procedure Libraries				Updatable Cu	astomer Grid			- 14	Microsoft Controls	_
B- > rcode			-					. 8	OpenEdge Controls	
TreeView	T	two two	Text	Text					OpenEdge Ultra Con	trols
B 🔁 Updatable	Te	test	Text	Test		7				
best Counterer Programs Data Bindry Source St Properties Events Prome ModeSCustomer Advanced St Advanced Teale Advanced St Advanced Teale Advanced St Advanced St		Start! Cic	ik "Startt" to be	gin designing th	a UltraGrid		,	Ĩ		
Gets a value indicating whether items in the.	Conso	le 🔝 Problems 🗧	🕄 🌏 Tasks							- 0
) 📬 🖻							J			

I save the form with that property setting, and run the form to see what happens to my grid. If I select a cell and try to type into it, nothing happens. So the **ProBindingSource** has effectively disabled any controls that are bound to it, which can be a more effective way to manage updates than enabling and disabling individual controls. But it's clear that I want to keep **AllowEdit** set to **True**, so I reset the property to its default.

Another property of the binding source that looks interesting is **AutoUpdate**. This does what the name implies: it will automatically do the binding source **Assign** for you, which in my case would write changes back to the temp-table. But it does not do any error checking or validation or any of the work to get changes back to the actual data source, so it's not recommended except for quick prototyping. That's why it's **False** by default, and I leave it that way.

Next let me look at the events on the grid to see what I want to intercept to handle updates. The **UltraWinGrid** supports a whole host of events that you can subscribe to. Most of the names are pretty self-explanatory, and you can learn more about them from the Infragistics documentation. The one I want is **BeforeRowUpdate**, whose meaning should be pretty clear.

🔯 OpenEdge Editor - Samples/Updatable/D	ustomer	UltraGridU	lpd.cls - OpenEdge Archi	tect - C:\Gui4Do	atNet					-0×
File Edit Design Navigate Search Project	OpenEd	pe Run W	rindow Help							
] 📬 - 🔛 🛆 🔏 📖 🕲 🎭 - 🖸	- 💁	- 🔗 -] 🗄 = 🗄 = 🗫 🖨	•						
📑 🎳 OpenEdge Editor										
🍣 Resources 🖾 🔅 🗖	💽 1M	del.ds	CustomerModel.cls	Customer	UltraGridUpd.cls (Design	n) 82				- D
(+ -+ k) (= 🗞 🏱								-	Toolbox	4
🗏 🐸 Samples 🔺	1 🖻	Customer	UltraGridUpd						EustomControl	4
Referenced Assemblies	P-			Heelatable	Customer Grid			- 1	Microsoft Con	trols
	1 🗆			Optionable	Customer and			- 1	C Priciosoft Com	in the second se
Tradiew		Text	Text	Text	11	R	11	- 1	U Openic age Co	NICES
🗉 🕗 Updatable	I F	Text	Text	Test		1		-11	1 OpenEdge Ult	ra Controls
CustomerModel.cls	1 1	Tum	1 cm			14		-11		
CustomerUltraGridUpd.cls										
C Model.ck	1									
🔠 Outine 📅 DB S 💷 Pro 😣 🖓 🗖	0									
筆刻 砚 ?	1									
mol IbaGridCustomer : Inharistics Win Librahafe 💌	1									
Properties Events	1									
BeforeRowLayouttemResized	1 6							- 1		
BeforeRowRegionRemoved	l r		otel Marcallas h		de en la disco disco d			- 11		
BeforeRowRegionScroll		Start!	Click Stare to b	egin designing	the UltraGrid					
BeforeFlowFlegonSize ReforeRcmRegionSize	° .				0					
BeforeRowResize										
BeforeRowsDeleted								-		
BeforeRowUpdate •	1							•		
BeforeSelectUhange ReforeSelectUhange										
BeforeSummaryDialog		moBSCust	omer							
About, UltraGrid Designer, Lavout										
BeforeRowUpdate								_		
Occurs before a row is updated.	Co	nsole 🔝	Problems 🕮 🕗 Tasks							>
1 🕆 🗈										
,							1			

Double-clicking on that event, I get a handler for it. And in the code I see it takes an event args subclass called **CancelableRowEventArgs**.

Let me find out what I can about that. In the **Class Browser**, I enter the name of the event args class, and look at its properties. There's a **Cancel** property, first of all, which is of type **Logical**, so I know that if I set it to **True**, then the update will be cancelled.

COPENED OPENED OF CONTRACT OF CONTRACT.	sGridUpd.cls - OpenEdge Architect - C:\Gui4DotNet	IOX
File Edit Source Navigate Search Project OpenEdge R	tun Window Help	
🔁 • 🗟 🛆 🔏 💷 🦦 🏇 • 🔕 • 💁 •	$\mathscr{G} \bullet [[] : \mathscr{G} \circ \mathfrak{G} \circ \mathfrak{G} \bullet \bullet \bullet \circ$	
📑 👫 OpenEdge Editor		
Class Browser 23	수 -> 😽 🛃 😣 😑 🔶 🏱 🗖	
Search CancelableRowEventArgs	Constructors	-
Infragistics. Win. Ultra WinGrid. CancelableRowEventArg	GancelableRowEventArgs (Infragistics.Win.UltraWinGrid.UltraGridRow)	
	E Vol Methods	
	- G Concel	
	Next-Sibing Next-Sibing	
	Row	.*/
	🖻 😵 Data Members	
	PUBLIC PROPERTY Cancel AS logical	
	SET	
	Nember of Sustem ComponentRadel, CancelTuentings	1. 11.
	nember of System component concernment of the second go	
	Summary:	
		2
] U" (<u>B</u>)		

There's also a **Row** property, and if I drill down into that, and look through *its* properties, I see that it holds a reference to the **Band** in the grid where the selected row is.



And again, looking through the properties, I see a **Key** property.

e Edt	Source Navigate Search Project OpenEdge Ri	in Window Help	
- 1	🗟 🗅 🛛 🖧 📖 🗞 + 🔕 + 🗛 + 📝	8 - 1 2 - 3 - 6 6 - + + -	
1 💒 o	DenEdge Editor		
	10		0
Class	Browser 23		
earch	8	🔒 HeaderVisible	1 6
	C I Best cidland	- G Hidden	
	G the Colored Colored bible bi	- Q HiddenResolved	
	C 10x30100000 Countribute Oscorr.cmps	- G Indentation	
	C Life a CridB and J. Br a CridB and Tune Converter	IndentationGroupByRow	
	C 1000000000000000000000000000000000000	- 🚱 IndentationGroupByRowExpansionIndicator	
	UtraGidCadSettion	IndentationGroupByRowResolved	
	the Grid Call	-Q Index	1.1
	UltraGridCella CellAssessibleChiest	- Co Viey	
	G thraced allows	- Q KeyLowercaseInterned	
	UtraGidCelProv/CuperInfo	- 🚱 Layout	-
	- G UltraGridCellProxyRole	G LevelCount	
	UltraGridCelProvid (IElement	- G MaxRows	
	- G ultraGridChildBand	G MinRows	
	UltraGridColumn	📃 🖳 Next-Sibling	
	G LibraGridColump+LibraGridColumpTypeCon-	-	
	G UltraGridColumnChooser	PUBLIC PROPERTY Key AS character	
	UltraGridComboEditor	GET	
	UltraGridComboEditorOwner	130	
	UkraGridDisplayLayout	Member of Infracistics Vin HitraWinGrid HitraGridBand	
	UltraGridEmptyRow	Rember of Intragracies.ern.orcraving in.orcravitaband	
	O UkraGridFilterCell	Summary :	
	UltraGridFilterRow	The internally assigned key value for the band. The Key property is read-	
	- G UltraGridGroup	only for the Band object.	
	UltraGridGroup+GroupLayoutInfo		
	G UltraGridGroup+UltraGridGroupTypeConves-		
	G UltraGridGroupByRow		
	- G UltraGridLayout		Def
	G UltraGridLayout+DisolavLayoutTypeConveg		
1)	2	

This is often used to hold the name of the data item that's displayed. For instance, the **Key** property of a **GridColumn** holds the name of the field displayed there, which can be useful. Here it's the name of the buffer whose fields are displayed in this band. That will be useful to me in my event handler.

So I've learned a little about how to manipulate the event args parameter to **BeforeRowUpdate**: I need that buffer name, because I want to be able to tell **SaveData** what buffer was updated, in case there's more than one. And I learned from the **Class Browser** how to drill down into the event args to get the **Row**, then the **Band** for the **Row**, and the **Key** value for the **Band**. That's the buffer name;

```
METHOD PRIVATE VOID moUltraGridCustomer_BeforeRowUpdate
( INPUT sender AS System.Object,
   INPUT e AS Infragistics.Win.UltraWinGrid.CancelableRowEventArgs ):
DEFINE VARIABLE cBuffer AS CHARACTER NO-UNDO.
DEFINE VARIABLE cCustName AS CHARACTER NO-UNDO.
cBuffer = e:Row:Band:Key.
```

There might be circumstances where an event fires when no actual changes were made to a row, so I check the binding source **RowModified** property:

IF moBSCustomer:RowModified THEN DO:

This property will be true if anything in the row was changed, and it will stay true as long as the grid is positioned to that row. Next comes the key step in the update through the binding source. When I invoke its **Assign** method, the changes in the grid – which you can think of as being like the screen buffer in older ABL terms – are transferred to the underlying record buffer, in this case, the temp-table row that the data came from. If there are any errors in that **Assign**, for instance if the UI let the user type in a value of the wrong data type into a cell, then the **Assign** will fail, so I check for an error return from **Assign**, set the **Cancel** property in the event args that I learned about in the **Class Browser**, and leave the update block:

```
IF NOT moBSCustomer:Assign() THEN
DO:
    /* Invalid data was entered in the grid row */
    e:Cancel = TRUE.
    LEAVE.
END.
```

But if the **Assign** succeeds, my changes have been moved to the temp-table, so that the code in the Model can validate them.

This part of the sequence is important. If there were update format masks you wanted to define in the UI itself to help determine what the user enters, to make sure that an integer is entered into an integer field, for instance, or to define a dropdown list of possible values, that's one thing, but you wouldn't want anything else in the way of business logic to execute in the View. In the case of my simple validation of **CustomerBirthCountry** being USA or Germany, I could have defined a grid cell dropdown that I then populated with those selection values. But ABL code that does calculations or validations that constitute real business logic doesn't belong there. My validation check in the **CustomerModel** class is really a placeholder for business logic that would normally execute on the server-side in a distributed application.

The key thing is that the **Assign** has transferred the updated values to the Model's temp-table, without the UI itself knowing anything about the underlying data storage, and the Model can then execute its validation without knowing anything about the UI. In any case, if **SaveData** returns an error, then I set **Cancel** just as before.

```
IF NOT moCustomerModel:SaveData(cBuffer) THEN
DO:
    e:Cancel = TRUE.
    LEAVE.
END.
```

I can now save this and run it, select a cell, make a change to it, and select another row. Anything I do to leave a row causes **BeforeRowUpdate** to fire, so my update has been executed. There's no visible confirmation of that, though, which I will address later.

File Edis Source Nampules Sourch Provide Coperative File Source Nampules Source Provide Source So	🕅 OpenEdge Editor - Samples/Upda	tab	le/CustomerUltraGri	dUpd.cls - OpenEdge i	Architect - C:\Gui4Dot?	iet			
Image: Source of the source	File Edit Source Navigate Search	Pro	ject OpenEdge Run	Window Help					
Electronic Control Product de ContrometVorsichided de (Crearger) ContrometVorsichided de (Crearger) Electronic Control Electronic Control Electronic Control Electronic Control Electronic Control Electronic Control <] 📬 • 🖂 👜 🔏 📖 🗞 🕸	þ.	• 💽 • 💁 • 🗍 🛷	- 🖗 - 🌍 - 🕸	• 🔶 • 🔿 •				
Preservers 33 Product de Coutoment/brackided de Coutoment/brachided de Coutoment/brackided de Coutoment/brackided de Coutoment/b	🖹 💱 OpenEdge Editor								
Image: State of the state	💞 Resources 🕴	C	🗋 💽 Model.cls	CustomerModel.	cls 🛛 🆏 CustomerUitra	rGridUpd.cls (Design)	CustomerUbraG	ridUpd.ds 🕺	° 8
Contract Dirac All All All All All All All All All Al		2	204	Notes:					-
Arrorad Accentre Arrorad Accentre Arrorad State Arrorad State Ar	😑 🎒 Samples		205						•/
Produk de Lavarde	Referenced Assemblies	1 -1	CustomerUltraGridU	od				1	
Image: Sector Updatuble Cultures Gid Image: Sector Cultures FirstNeee Cultures Gid Image: Sector Cultures FirstNeee Cultures Gid Image: Sector Cultures FirstNeee Cultures Gid Image: Sector Pate Wagnet Germany Image: Sector Pate Wagnet Germany OD/14/1897 Image: Sector Pate Wagnet Germany OD/14/1897 Image: Sector Image: Sector Pate Wagnet Germany OD/14/1897 Image: Sector Image: Sector Image: Sector Pate Wagnet Germany OD/19/1900 Image: Sector Image: Sector Pate Mapply Germany OD/19/1900 Image: Sector Image: Sector Value Germany OD/19/1900 Image: Sector Image: Sector Image: Sector Value Germany ID/16/1902 Image: Sector Image: Sector Image: Sector Image: Sector Value Germany ID/16/1902 Image: Sector Image: Sector Image: Sector Image: Sector Image: Sector Image: Sector	IL Procedure Libraries	_							
Image: Trender ContromeErickSame Contr	B 🗁 rcode			U	pdatable Customer Grid			EArgs 1:	
Woldskilde Robett Koncestflödded USA (07/13/197) P Peter Wagned Genomethod de Genomethod d	TreeView		CustomerFirstName	CustomerLastName	CustomerBithCountry	CustomerBithdate	CustomerGender	concept 71	
Outcomerchen of an and a state of a state o	Updatable		Robert	KennedyModified	USA	07/13/1979			
Produkt.ds After Weinbrigton USA 05/13/1922 Produkt.ds Particik Levrisi Germany 00/13/1920 Produkt.ds Property Value Torrer Germany 0.0/13/1920 Produkt.ds Property Value Could with the standard withe standard with the sta	CustomerProce.cs		Peter	Wagnet	Germany	09/14/1987	4		
Politik Levis Germany QU/19/1970 IP Brocetty Value Fic Myees Germany GGR/950 IP Brocetty Value Casa Torrer Germany QU/19/1970 IP Brocetty Value Casa Morphy Germany QU/16/1982 IP Brocetty Value Casa Morphy Germany 12/01/1982 IP Brocetty Value Casa Morphy Germany 12/20/1983 IP Brocetty Justin Water USA 00/11/1980 IP Brocetty Germany USA 00/11/1980 IP Annoda Robetts USA 00/11/1971 IP atter 22.03 END RETUPN 22.0 IP 21.0 PETUPN 22.0 IP IP IP	Model.ds		Alice	Washington	USA	05/19/1982			
Construction Construction<	St Cutine TR DE S		Patrick	Lewis	Germany	03/19/1970	2		
Image Ty Value Lews Torrer Germany 07/16/1994 Image Image Ty Value Cacl Murphy Germany 1201/1992 Image Image Ty Value Cacl Murphy Germany 1201/1992 Image Image Status Justin Wation USA 0f/22/1994 Image Image Status Justin Wation USA 0f/22/1993 Image Image Status Justin Wation USA 0f/22/1993 Image Image Status Justin Wation USA 0f/22/1993 Image Image Status Justin Torres USA 0f/27/1993 Image Image Status Justin USA 0f/27/1993 Image Image Image Status Image Image Image 0f/27/1993 Image Image Status Image Image Image Image Image Image Image Status Image Image Image			Eric	Myers	Germany	06/08/1950	2		
Image Contraction Line Cool Multiply Gennary 12/01/1922 Produces Junin Image Contraction State Junin Image Contraction State Rests Gennary 12/01/1922 Image Contraction State Junin Value USA Optimized State Rests Image Contraction State Rest Rest Image Contrect Rest Rest Rest Re	Barrette Little		Laura	Torres	Germany	07/16/1984			
Converting Thomas Junin Warkins Cernary 12/22/1554 FF ediabit true Junin Warkins Cernary 12/22/1554 FF Junin Warkins Cernary 12/22/1554 FF Junin Warkins Cernary 10/27/1983 FF Junin Warkins Use 0/17/17870 FF Inded Figure Annoda Roberts USA 0/11/1/1971 F pth /Sompless/bidstall F EXTUPRI. Size 220 PETURE. 220 PETURE. 220 EXTUPRI. Size 220 EXTUPRI. 221 EXTUPRI. Size 220 EXTUPRI. Size	Property Value		Carol	Murphy	Germany	12/01/1982			
etchd frae Jaun Waton USA 07/27/983 PF Isit an Saway 14, 2010 Anthu Tuner USA 04/11/1800 IF Isit an Saway 14, 2010 Anthu Tuner USA 04/11/1800 IF Isit and Saway 14, 2010 Anthu Tuner USA 04/11/1800 IF Isit and Saway 14, 2010 Anthu Tuner USA 03/11/1971 Image: Saway 14, 2010 ptbl // SomeRit Botadia 220 PETUPNI. 220 Image: Saway 14, 2010 I	derivec false		Thomas	Jerikins	Germany	12/29/1954	R		
biter C. Souvery 15, 2010 Antruk Tureer USA 04/11/1990 FF bited Status Antruk Tureer USA 04/11/1990 FF bited Status Roberts USA 03/11/1971 T other 220 EXTURN. 223 END RETWOR. 223 END RETWOR. 224 Console (1, Problems 52) (2 Tasks	editabil true		Justin	Watson	USA	01/27/1983	<u>ସ</u>		1.1
Dotation ClinedWolfModel Ammoda Roberts USA 00/11/1571 name ClinedWolfModel size 12,033 bytes Size 12,03 bytes Size 12,03 bytes Size 12,00 bytes Size 10,00 bytes	last mc January 14, 2010		Arthur	Tuner	USA	04/11/1980	2		
Name Customerite visual. I pdf Size 220 220 size 12,033 bytes 220 PETUPN. 230 END RETUDN. 231 END RETUDN. 211 Image: Size 200 PETUPN. 213 END RETUDN. 201 Image: Size 214 Conside Problems 32 Image: Size 200	location C:\Gui4DotNet\Sa		Amanda	Roberts	USA	03/11/1971			
pohr (Somples) Lodot al size 12,033 bytes 220 PET UPR. 231 END RETHON. 231 END RETHON. 231 END RETHON. 232 FET UPR. 233 END RETHON. 234 END RETHON. 235 END RETHON. 236 END RETHON.	name CustomerUltraGrid	1					<u>)</u>		
S20 12,033 bytes 220 PETUPN. 233 ■ END RETUD. 234 ■ D RETUD. 235 ■ D RETUD. 235 ■ D RETUD. 235 ■ D RETUD. 236 ■ D RETUD. 237 ■ D RETUD. 238 ■ D RETUD. 239 ■ D RETUD. 230 ■ D RETUD. 240 ■ D RETU	path /Samples/Updatabl								
220 PETUPN. 233 END RETHON. 233 END RETHON. Image: Conside []. Problems 22 @ Tasks > Image: Conside []. Problems 22 @ Tasks >	size 12,833 bytes								- 1
Console Problems 22 Crads Brown 212:9			220	RETURN.					
233 EXD RETHON. ✓ Image: Console [] Problems 22 [] Tasks [] Problems 22 [] Tasks			230						
			231 1	END METHOD.					
□ Console [] Problems 12 [] Tasks [] □ [1.0					<u> </u>
T° E Writeble Insert 212:9			Console	🖸 Problems 🕄 🏼 🙆 Ta	sks				
	1 🗈 🗈				Writable	Insert 21	2:9		

Next I change another name, and then tab to the **CustomerBirthCountry** cell, and enter an invalid value. When I leave the row, I see the error that came back from **SaveData**, which is checking the values that were assigned to the temp-table.

			-		1.00		
Resources 23	IModel.ds	CustomerModel.	ds 🗳 CustomerUitra	GridUpd.cls (Design)	CustomerUltraG	ndJpd.ds 🗙	
← → & €	204	Notes:					
Samples	A	A					····· ,
B & Referenced Assemblies	🛃 CustomerUltraGrid	Upd			_ 🗆 X		
			odatable Contorner Grid				
I Co TreeView	CurtomerFirstName	Customed astName	CustomeRithCountry	CustomerRittydate	CurtomeGeoder	ttkrgs):	
😑 🇁 Updatable	Robert	KennedsModified	USA	07/13/1979	R -		
CustomerModel.cls	P./ Peter	WagnerChanged	France	09/14/1907			
CustomerUltraGridUpd	Alice Mes	sage (Press HELP to vie	w stack trace)	05/19/1982			
and an and a second	Palnck			03/19/1970	<u>.</u>		
Outline M DB S D Pro 2	Enc In	valid Birth Country value, r	nust be USA or Germany.	06/08/1950	R I	1	
E 1	Lava	ОК	Help	07/16/1984		1	
operty Value	Carol			12/01/1982		1	
deriver false	Thomas	Jenkins	Germany	12/29/1954	8	1	
editabl true	Justin	Watson	USA	01/27/1983			
last mc January 14, 2010	Athur	Turer	USA	04/11/1990		1	
Inked False	Amanda	Roberts	LISA	03/11/1971			
name CustomerUltraGrid		Lumence		1		1	
path /Samples/Updatabl						1	
size 12,833 bytes							
	220						
	229	RETURN.					
	230						
	231	END METHOD.					

By default the **Cancel** just erases my changes altogether, which is not very friendly, so that's something else I'll improve on before I'm done. I have at least confirmed that a basic update goes through, and that **SaveData** has written it back to the data source, the database table.

[The following part of the document corresponds to the second part of the video presentation.]

Since there are a few things that I don't like about how my sample grid behaves so far, I'm going to make some changes to show a few of the user interface options that can help improve the form's usability, as well as more of the ProBindingSource properties and methods.

One issue is that if I enter invalid data in a row, such as France for the CustomerBirthCountry name, I get an error message, but then lose my changes when the code sets the **Cancel** property. I could write some ABL code to make it behave differently, but if I select the grid in Visual Designer and look through its properties, I see that there is one called **RowUpdateCancelAction**, which looks like a perfect description of my situation. It's defined in terms of an enumeration, and if I drop down its list of values, I see that the default is **CancelUpdate**, which is what I saw happen, but the other choice is **RetainDataAndActivation**. That's exactly what I want to have happen: to leave the changes in place and leave me on the row with the error – that's what Activation means, that the row is still active – so that I can see and either correct the error or press Escape to cancel the changes myself.

🗑 OpenEdge Editor - Samples/Updatable/Cust	omerUltraGridUpd.cls - (IpenEdge Archite	ct - C:\Gui4DotNet					_IO ×
File Edit Design Navigate Search Project Op	enEdge Run Window H	ыþ						
	0 1	5 6 6	-					
		4- 4- 4-						
El le Obeurade Faco.		1-	1					
Resources 🕄 🗌 🗆	CustomerModel.cls	IModel.cls	🔤 🌄 *CustomerUltra	vGridUpd.cls (Design)	22			
						4	Toolbox	4
🖻 🚰 Samples 🔺	🛃 CustomerUltra	FridUpd					CustomControls	
Referenced Assembles	Q		0				Customcontron	
B-JL Procedure Libraries			Updatable Cu	stomer Grid		_	Microsoft Contr	rols
🖲 😂 rcode							OpenEdge Con	alorte
TreeView	Text	Text	Text		☑ _/_/		DpenEdge Ultr	a Controls
🕀 🧀 Updatable	Text	Text	Text		E _/_/			
CustomerModel.cls								
CustomerUltraGridUpd.cls								
K IModel.ds								
🚼 Outine 📅 DB Str 💷 Prope 🕺 🦳 🗆	0							
10 AL 10 V	1							
moultraandcustomer : Intragistics.win.Ultrawiniar								
Properties Events								
MaximumSize 0,0	191							
E MinimumSize 0,0					_			
Modifiers Private	Start!		egin designing the					
RightToLeft No	0		0					
RowUpdateCancelAction RetainDataA -								
E Size 602, 283								
StyleLbraryName						× 1		
Stylesenvane	1					· 1		
Tableday 0								
TabSton True	moBSCustomer							
these like Cold Designs of some hids and								
Book, Unablid Designer, Lavok Wizard								
RowUpdateCanceAction						_		
Specifies the action that should be taken when	🕒 Console 🔝 Problem	ns 🕮 🧭 Tasks						🖗 🗸 🗖 🗖
] 📑 🖪								

This is a perfect example of both the value of these .NET controls, especially the UltraControls, and also the challenge of using them: On the one hand, there are many properties, methods, and events to sort through to find what you need to solve a problem. On the other hand, there's almost always built-in support for what you need the control to do, so it's worth your while getting to know the controls through the documentation and just reviewing the names of everything the controls support, which are usually clear in telling you what they do. Naming conventions are consistent enough that with a little experience you will get a sense for what to expect and what to look for in a control that's new to you.

If I run the form again with **RowUpdateCancelAction** changed to **RetainDataAndActivation**, I can make a change to a Customer, and then change the CustomerBirthCountry to an invalid value. Now I see the error message as I did before, but my changes haven't been canceled:

OpenEdgeEditor - San e Edit Design Navigat	iples/Updata ie Search Pr	ible/EustomerUltraß roiect OpenEdge Ru	ridUpd.cls - OpenEdge n Window Help	Architect - C:\Gui4DotM	iet		_10
** • • • <u>/</u> •	*	· Q · Q ·	2 • 1 5 • 5 • 5	• (• • • • • •			
OpenEdge Editor							
Resources 23		- Custon	erModel.ds 💽 IMod	lel.ds 🖏 Customert		×	
		28 7				-	Toolbox
- 🎏 Samples		🔺 🔛 Cur	tomerUltraGridUpd				al CastonControls
Referenced Ass	enbles 👔	CustomerUltraGrid	Upd			_10 ×1	
IL Procedure Libra	ries						Hicrosoft Controls
🖲 🧼 rcode			. U	pdatable Customer Grid			OpenEdge Controls
TreeView		CustomerFirstName	CustomerLastName	CustomerBirthCountry	CustomerBirthdate	CustomerGender	🗈 OpenEdge Ultra Control
Custometh	what size	Robert	KennedyModified	USA	07/13/1979		
Customerta	braGid Inda	Ø Peter	WagnerModified	France	09/14/1987	R	
Model.ck		Alice	Washington	USA	05/19/1902		
Cushoa 📅 ne Sar	Brone	Patrick	Lewis	Germany	03/19/1970	2	
Coone 189 pe port	- Prope	Eric	Myers	Gemany	06/08/1950	2	
	8	Laza	Torres	Gemany	07/16/1984	-	
ustomerUltraGridUpd : Pro	gress.Windo	Canol	Mumbu	Germany	12/01/1992		
Properties Events		Thomas	Jacking	Camaru	10/00/1054		
Opacity	100%	h alia	Makes	10°A	01/22/1982		
2 Padding	0.0.0.0	Jugan	watton	0.04	01/2//1903	<u> </u>	
previousState	Normal	AUNA	Tuner	USA	04/11/1380	M	
RightToLett	No	Amanda	Roberts	USA	03/11/1971		
Showloop	Taxe						
ShowInTaskbar	True					L.	
1 Size	614, 36					,F	
SizeGripStyle	Auto						
StarPosition	WindowsDe	lault.c 🐺 🗤	BSCustomer				
Text	Customed	litra6					
TooMost	Ealte	- III					
Text The text associated with th	e control.	Conso	le 🖹 Problems 💠 🌏	Tasks			≫ ♡ □

The row in error is still the current row, as indicated by the row edit icon over on the left, so I can click in the cell where the error is, and correct it, and then move out of the row and save all my changes. That's much more user friendly. I try another row, and once again enter an invalid CustomerBirthCountry, and once again see the error message. It's a characteristic of the UltraGrid that I can press **Escape** to cancel an update, so I have that option to go through with the Cancel myself.

I present this as an example of how you can expect that the .NET controls will provide countless variants of behavior that you can capture and take advantage of. You just need to be thorough enough to find and use them.

The next issue I had with my updatable grid was that there was no visible feedback when an update succeeded. As one example of how I can deal with that, I'll add a status bar to the form to display an update status. Here among the Ultra controls is a **StatusBar** that I can drop onto the bottom of the form.

9 . D A I X	101 Pa 42 - O	0	10	1.61 -							
CoenEdge Editor											
Resources 23	• •		ustomerMod	el.ds	Model.ds	Customer	UltraGridUpd.cls (Desig	n) 82			•
					0				2	Toolbox	
🐸 Samples			🛃 Custome							222UltraLisWew	
Referenced A	ssembles									100 UltraMaskedEdt	
-JL Procedure Lb	raries -	11				Updatable	Customer Grid			28 UltraMonthViewMulti	
🕀 🗁 roode		11								The UltraMonth/iewSingle	
E E TreeView		11	▶ Test		Test	Test		R		UltraNavigationBar	
🖻 😂 Updatable		1	Text		Test	Text				UltraNavigationBar	
Customer	Model.ds	1 1				1.200		he -		1 UltraNumericEditor	
Customer	Ultra/GridUpd.ds	U I								UltraOptionSet	
Model.ck										UltraPanel	
Outine 📅 DB Str	Prope., 23	1								UltraPictureBox	
COULD THE REAL PROPERTY	C Propont Co									UltraPopupControlConte	iner
	14 2 US									3 UltraPrintDocument	
aStatusBar1 : Infragisti	::: Win:UltraWinStatusB. 💌	1								UltraPrintPreviewContro	4
constine Events	1	1								UltraPrintPreviewDialog	
roperties Evenus										C UltraPrintPreviewThurs	nai
(About)	-		4							III UltraProgressBar	
(DataBindings)		11	Ohent							S UltraSchedulePrintDocu	men
(Name)	utrastaturbari	11	otart							UltraScrolBar	
AccessibleDesciptor							0			UltraSpellChecker	
AccessbleRole	Default		AsaStatusBar	1			0			UltraStatusBar	
AllowOsop	False	1 4								UltraT abbedMdManag	N
AlphaBlendMode	Optimized	l an						1	•	UltraTabControl	
Anchor	Top, Left	1-1-1								UltraTabStripControl	
Appearance										UltraTextEditor	
Appearances	(Collection)		moltSLus	tomer						Ultral meconel dtor	
bout, Custom Property	Peges									IlliuT colTeManager	
										4	

I rename it to be consistent with the other control names I've used. Without going into all the rest of its properties, I initialize its **Text** property to "Status: ", to make that the value initially displayed.

	cs/opostable/cust	omeror	raGridUpd.cls - 0	peneoge Archite	ct - C:\Gui4Dot	465			_	
ile Edit Design Navigate	Search Project Op	enEdge	Run Window H	elp						
😁 • 🔛 🛆 🔏 📖	% - 0 -	Q	🛷 • 🔄 •	11 - 🐤 🔶 -	-					
🝸 🥰 OpenEdge Editor										
Resources 23	- 0	CU 😒	tomerModel.cls	Model.cls	Custome	rUltraGridUpd.cls (Desi	ign) 🛙			- 0
) 🔬 🛛 😑 😫 🌱	_						*	Toolbox	4
🗉 👹 Samples			CustomerUltraG	ridUpd					222 UltraLiofView	-
🖲 🧬 Referenced Assem	bles	_							100 UltraMaskedEdit	
B-JL Procedure Libraries	ب ا				Updatabi	e Customer Grid			22 UltraMonthViewMulti	
🕀 🗁 rcode					_				10 UltraMonthViewSingle	
🗈 🇀 TreeView		•	Text	Test	Text		V	_/_/	UltraNavigationBar	
🖯 🧼 Updatable			Text	Text	Text		V		UltraNavigationB ar	
CustomerMode	i.ds						-		UltraNumericE ditor	
 CustomerUlbrad 	GridUpd.cls								UltraOptionSet	
IModel.cls	-								UltraPanel	
😤 Outline 👎 DB Str 📖	Prope 🕄 🗖 🗖								UltraPictureBox	
	AL IN Z								UltaPopupLontoiLontan	er
	tt: Z+ L≫								UltramintDocument	
moUltraStatusBar : Infragistics.V	Win, UlkaWinStatus 💌								UtraPhintPreviewControl	
Properties Events									10 UtraPrintPreviewThembry	
StatusBarTest on moUltra	-								UhaPagassiRar	
StyleLibraryName	_	1 12	u				_		State UltraSchedulePrintDocum	ert
StyleSetName	_		Start		egin designing				UltraScrolBar	
Tabindex 1	1						_		UtraSpellChecker	
TabStop F	False					0			UltraStatusBar	
Tag		Sta	itus:						UltraTabbedMdManager	
Text	Status:							-	UltraTabControl	
LeathenderingMode [Tore	-						•	Control UltraTabStripControl	
LiteEleMode [Defende								🖽 UltraTextEditor	
UzeMoemonic 1	Ine -I	- 4	moBSDustomer						UltraTimeZoneEditor	_
									UltraToolbarsManager	
About, Custom Property Pages	ha								IlltraTooTinManager	ъČI.
Text The text associated with the co	Jontool	Co	nsole 💽 Problem	s 🕮 🔊 Tasks						
									47	
T° 🗈										

Now I need to add a couple of lines of code to assign a value to display after an update. At the beginning of the **BeforeRowUpdate** event handler. I reset the **Text** property back to its initial value, so that it's reset each time an update happens:

DEFINE	VARIABLE	cBuffer	AS	CHARACTER	NO-UNDO.						
DEFINE	VARIABLE	cCustName	AS	CHARACTER	NO-UNDO.						
cBu	cBuffer = e:ROW:Band:Key.										
mot	JltraStatu	sBar:TEXT	= '	'Status: ".							

Then at the end of the method I construct a string that displays the first and last name fields:

cCustName =	e:ROW:Cells:Item[0]:Text + " " +
	e:ROW:Cells:Item[1]:Text.

This is worth taking a look at for a moment. Remember that the method's **CancelableRowEventArgs** parameter has a **Row** property. The **Row** contains a **Cells** property, which is a .NET collection of all the

cells in the row. The standard way to access an element of a collection is with the **Item** property, which takes a zero-based index to the cell you want. The **Text** property of each cell is the value it contains. So **Item[0]** and **Item[1]** are the first two columns in the grid, the CustomerFirstName and CustomerLastName fields.

To see how my status bar looks, I can rerun the form, click on a row, change the name, and change the CustomerBirthCountry to a valid value. When I select another row, the **BeforeRowUpdate** handler fires, and I see the status message.



So this is just a simple example of integrating another Ultra control into a form. While I'm running the form and changing values, I can show you another aspect of the update that you could consider an issue. If I click on the CustomerLastName column header, the **SortData** method that I coded in the presentation series on sorting data with the ProBindingSource re-opens the temp-table query, sorted by CustomerLastName. Now if I select a customer, and change the CustomerLastName to a value that should sort somewhere else, the change is saved, but the data isn't resorted, because I didn't ask to reopen the query.

😿 OpenEdge Editor - Samples/Upd	atab	le/CustomerUltraGr	idUpd.cls - OpenEdge i	Architect - C:\Gui4Dot/	iet		_IO ×
File Edit Source Navigate Search	Pró	ject OpenEdge Run	Window Help				
🗈 • 🗔 👜 🔏 📖 🕲 :	於 •	0 • 94 •] 🛷	(+ <u> 0</u> + 🙀 + 旼	• (• • • •)			
📑 💐 OpenEdge Editor							
Resources 23		Custome	rModel.ds	iel.cls 🖏 Customert	RraGridUpd.cls (Desi	an) 💽 Gustomentile ada	alteite 🗙 🔍 🗖
		4 V C13V	RETHOD PRIVAT	E VOID MODICENSE	IGCUSCOMEL_DE	TOPEROPORT	-
III 1 Samples	0	220	(INPUT sende:	r AS System.Obje	ct,	Canceleble Bow Pres	an brown ba
Referenced Assemblies		Customert IltraGridt	INPUT C AS	Initadistics. wit	. orcrawinoric		ackrgs /:
B-JL Procedure Libraries							
🖲 🗁 rcode				stable Customer Grid			
TreeVew		CustomerFirstName	CustomerLastName /	CustomerBithCountry	CustomerBirthdab	e CustomerGender	
CustomerModel de		Patrick.	Alexander	semany	02/08/1983		
CustomerUltraGridUpd.		Robert	CharlieBaker	JSA	04/29/1974	1	
- Model.cls		Susan	Barnes	Sermany	07/17/1972		
E Outline 23 📅 DB Str 🗔 Prop		Evelyn	Bing	USA	12/10/1965		
		Helen	Brooks	Germany	07/31/1974		
E B USING Declarations		Lany	Brooks	USA	02/29/1972	2	
Variables		Mary	Bums	USA	09/02/1956		
🖻 🚳 Methods		Melissa	Bums	Germany	10/25/1959		
- GustomerUltraGridUpd		Richard	Burns	Germany	01/05/1968	2	
CustomerUltraGridUpd		Frank.	Burns	USA	04/10/1984	2	
mol Brackic storer After		Helen	Carter	USA	05/25/1951		
Conception of the second secon	11						
	Sta	tus: Robert CharlieBake	r update successful.			h.	
		243	e: R	OW:Cells:Item[1]	:Text + " upo	iate successful.".	
		244	END.				
		246	RETURN.				
		0.40					
							* 200
		> Console	E Troblems 23	Tasks			10 × - 0
P 🖪				Writable	Insert	241:27	

Showing you how to make the resort happen will illustrate another part of the interaction between the user interface, the binding source, and the underlying data. Back in the **CustomerModel** class, I want to make a change to the **SaveData** method. If the save operation succeeds, I want to re-open the current query,

with whatever sort order is defined for it. That way, the data will be resorted properly if I make a change that affects a row's sort sequence in the grid.

```
/* If we get here all client-side validation succeeded. */
IF hBeforeBuffer:SAVE-ROW-CHANGES () THEN
DO:
    hBeforeBuffer:ACCEPT-ROW-CHANGES ().
    httCustQuery:QUERY-OPEN ().
    RETURN TRUE.
END.
```

Looking at the ProBindingSource properties again makes it clear what's making the resynchronization of data between ProBindingSource and the UI work. The property that plays a key role here is **AutoSync**.

🔃 OpenEdge Editor - Samples/Updatable	/CustomerUltraGridUpd.cls - OpenEdge	Architect - C:\Gui40	otNet			_ 🗆 X
File Edit Design Navigate Search Proje	ct OpenEdge Run Window Help					
🔁 • 🖂 📥 🖌 🔛 🖓 •	🗿 • 💁 • 🛛 🛷 • 🗍 🖢 • 🖏 • 🌣	Þ � • ↔ •				
🟥 🎳 OpenEdge Editor						
🥰 Resources 😫 🔍 🗖	CustomerModel.cls	: 😽 CustomerUitz	aGridUpd.cls (Design) 🖾 💽 CustomerUltra	GridUpd.cls	° 0
	·/				Ē 1	f oolbox 4
🖻 👹 Samples 📃	EustomerUltraGridUpd					CustomControls
Generation Assemblies		Updatable 0	Customer Grid			Microsoft Controls
Procedure Libraries						OpenEdge Controls
E- Contraction	First Text	Text			12.	OpenEdge Ultra Controls
🖻 🇁 Updatable	Text Text	Test	_/_/		17.6	opencage on a control
CustomerModel.ds						
C IModel.cls	1					
9 O	ก็ไ					
0: Z + 03	7					
mod/SCustomer : Progress.Data.BindingSour	1					
Properties Events						
(Name) moBSCustomer						
AllowNew True	Start! Click 'Start!' te	o begin designing th	e UltraGrid			
AllowRemove True						
AutoSort False	Statur					
AutoUpdate False					_	
Batching False	1				E F	
DataMember v						
ProBindingSource Designer	W moBSCustomer					
AutoSamo						
Indicates whether the BindingSource will						
automatically refresh controls after certai	🕒 Console 🖹 Problems 🕴 🕢 Task	s				≫ [∨] □
1 T* 19						
10 00						

It's **True** by default, which is what supports the behavior I've been showing you. If **AutoSync** is **True**, then any time you re-open a query bound to a ProBindingSource, or use one of the ABL **REPOSITION** statements to change the selected row in the query, the binding source **Position** property is reset automatically, and any user interface controls bound to the binding source are also refreshed to stay in sync with the data. This is probably the behavior you want almost all the time, but if you ever want to control when data gets refreshed yourself, you can set **AutoSync** to **False** and use the binding source **Refresh** method.

To take a look at the result of the query re-open, I save and re-run the form, re-sort the data by CustomerLastName, select a row, and make a change that changes its sort position.

13 • 14 4 Internet		× •	0 • Q •] 🔗	•]2-2-*>	\$ • ÷ •				
OpenEd	ge Editor				47				
Resources	8		CustomerModel.	ds 🔄 IModel.ds	S CustomerUltraGrid	SUpd.ds (Design) 23	 CustomerUltraGridUpd. 	ds	
	- + - + - + 							Toolbox	· · · · ·
🗄 🚝 Samples			LustomerUltraGridU	pa				EustomControls	
🖲 💕 Refi	erenced Assemblies			U	pdatable Customer Grid			Microsoft Contr	ols
E Ch root	edure Libraries	F	CustomerFirstName	CustomerLastName /	CustomerBirthCountry	CustomerBirthdate	CustomerGender	I OpenEdge Cop	trole
E Contract	~ sView	F	Patrick	Alexander	Gemany	02/08/1983	- v	Concept con	
🖯 🇁 Upd	latable	-	Robert	Baker	USA	04/29/1974	E F	_ UpenEdge Ultra Co	
- 6	CustomerModel.cls		Evelyn	Bing	USA	12/10/1965			
	CustomerUltraGridUpd.cl		Helen	Brooks	Germany	07/31/1974			
	umodel.cs	F	Lany	Brooks	USA	02/29/1972			
0ut 🔛 I	08 💷 Pro 🗙 📑		Marv	Burns	USA	03/02/1956			
	臣 순나 빈		Melissa	Burns	Germany	10/25/1959			
oBSCustomer:	Progress.Data.BindingSc		Richard	Burns	Germany	01/05/1968	R		
Properties	Events	F	Frank	Burns	USA	04/10/1984	V		
(Name)	moBSCustomer	F	Helen	Cater	USA	05/25/1951			
AllowEdit	True		Susan	CCBarnes	Germany	07/17/1972			
AllowNew	True	1.							
AutoSort	Falce								
AutoSync	True	Sta	tus: Susan CCBarnes up	date successful.			10		
AutoUpdate	Falce	-							
Batching	raise		<u> </u>				•		
a . A	•			æ,					
ProBindingSou	rce Designer		🍄 moBSCusto	mer					
AutoSvec									
ndicates whet	her the BindingSource will								
automatically r	elresh controls alter cer	lai_	Den 1 De	11 M (17 m 4)				-*:	

Leaving the updated row, the query is re-opened and all the data re-synchronized with the grid.

Lastly, I'll show you a simple example of using the ProBindingSource **Refresh** method. To do that, I need a new method in the model, so I first add it to the interface. The new method, FormatColumn, takes a column name and its value as parameters, and re-formats the value under certain rules. If the method returns true, then the value has been re-formatted and needs to be re-displayed. That's where **Refresh** comes in.

INTERFACE Updatable.IModel:
METHOD PUBLIC VOID FetchData (INPUT pcFilter AS CHARACTER).
METHOD PUBLIC VOID SortData (INPUT pcSort AS CHARACTER).
METHOD PUBLIC HANDLE GetQuery().
METHOD PUBLIC LOGICAL SaveData(INPUT pcBufferName AS CHARACTER).
METHOD PUBLIC LOGICAL FormatColumn (INPUT pcColumnName AS CHARACTER,
INPUT pcColumnValue AS CHARACTER).
END INTERFACE.

From the Source menu, I add the skeleton code for the new method:

🕅 OpenEdge Editor - Samples/Updatable/Eu	omerModeLcls - OpenEdge Architect - C:\Gui4DotNet		_02
File Edit Source Navigate Search Project	enEdge Run Window Help		
] 🗂 • 🖩 🛆 🗋 🔏 🖬 🕲 🛛 🎘 • 🕗	Q @Override/Implement Members	<u>d</u>	
🗈 🏭 OpenEdge Editor	Generate stubs to override or implement members		
🕰 Resources 😂 📃 🗖	Check inherited members that you want to override and members internetied via interface for which you want to generate shifts.	ustomerUltraGridUpd.cls	
Contraction of the second	Select members to override or implement; Select members to override or implement; Description of the select All Description of the select All Descript	ucceeded. */	1
Property Value	94 95 96 97 ⁰ 98 99		
	Insertion position : Alphabetical order Insertion positi		
	05 06 07 07 07 07 07 07 07 07 07 07	CTER):	
	venerace deraux values for required methodes	NIT	_
	© Generate Cancel		 ⇒ ⊽ ° ⊂
) 🕈 🖻	Writable Insert 95:2	s J	

This is the code for the method:



If the CustomerBirthCountry column has a value of "usa" with any type of capitalization, the code for that case forces it to be all upper case, and returns **TRUE** to signal that the data value has been changed. Remember that because this is code in the Model, it's the temp-table value I'm changing, which has no direct effect on the UI.

So back in the form I need to invoke **FormatColumn** and check its return value. Looking again through the **UltraGrid** events, I find one named **AfterCellUpdate** that fires after a cell has been updated. A little experience will teach you where to look in these very full-featured controls for support for the behavior you need. The event handler for **AfterCellUpdate** takes an event args class of **CellEventArgs**, and if I were to look that up in the Class Browser, I'd see that, not surprisingly, it has a **Cell** property. The **Column** property of the **Cell** points to the column the cell is in. I have used the Key property once before to get at the buffer name for a band in the grid. Here the **Key** property holds the column name, so that becomes the first parameter to **FormatColumn**. The cell also has a **Text** property, which holds the column value, so that becomes the second parameter. And if **FormatColumn** returns **True**, it has modified the field's value in the temp-table:

```
METHOD PRIVATE VOID moUltraGridCustomer_AfterCellUpdate(
   INPUT sender AS System.Object,
   INPUT e AS Infragistics.Win.UltraWinGrid.CellEventArgs ):
    IF moCustomerModel:FormatColumn(INPUT e:Cell:Column:Key,
        INPUT e:Cell:Text) THEN
        moBSCustomer:Refresh().
        RETURN.
END METHOD.
```

This is where I invoke the ProBindingSource **Refresh** method to redisplay the row. Because the code hasn't re-opened the query or re-positioned it, I have to do the **Refresh** myself.

To test this latest change, I save and re-run the form, select a customer, change the CustomerLastName, and then change the CustomerBirthCountry to "usa", without using all capitals:

🗑 OpenEdge Editor	- Samples/Upd	stab	le/CustomerUltraGri	dUpd.cls - OpenEdge					103
File Edit Source N	Vavigate Search	Pro	ject OpenEdge Run	Window Help					
📫 - 🔛 🌰	1001	ţ.	• 🗛 • 🖉 🛷	- 🖗 - 🤯 - 🍤	• 🔶 • 🔶 •				
😭 縃 OpenEdge Ed	Stor								
🕼 Resources 🖽		•	CustomerModel	.cis 💽 IModel.cis	🖏 CustomerUltraG	idUpd.cls (Design)		lpd.ds 🕷	- 0
			9 31	InitializeCom	ponent().				1
🖂 🎏 Samples		-	A 32	moCustomerMod	iel = NEW Updata	ble.CustomerMod	iel().		
🕀 📌 Referenc	ed Assemblies	12	CustomerUltraGridU	od .			_ [0] ×		
II-JL Procedure	e Libraries								
🗉 🗁 rcode				U	pdatable Customer Grid				
E C Treeview			CustomerFirstName	CustomerLastName	CustomerBirthCountry	CustomerBirthdate	CustomerGender		
- Down	e www.Model.cls		Robert	Kennedy	USA.	07/13/1979			
- Ousto	merUltraGridUpd.	20	Peter	WagnerChanged	Germany	09/14/1907	R		
IMod	el.cls		Alice	Washington	USA	05/19/1902			
e out 🖉 na 🚽	Pro		Patrick.	Lewiston	USA.	03/19/1970	3		
			Eric	Myers	Germany	06/08/1950	V		
I	4 -		Laura	Torres	Gemany	07/16/1984			
ATI Type	Yalue		Carol	Murphy	Gemany	12/01/1982			
File Name			Thomas	Jenkins	Gemany	12/29/1954	3		
Length	1253		Justin	Watson	USA	01/27/1903	V		-
Problems			Athur	Tuner	USA	04/11/1900	2		
Start Position	2240		Amanda	Roberts	USA	03/11/1971	<u> </u>		1
Token Length	0	E							
Token Number	148								
Token Start Posi	-1	Sta	hut:				1		
Token Type	METHOD-DECLAR.	_	199	TE moCuetones	Node 1 - Former Coll	TNEUT ALCOL	La Columna Vers	1	
			189	INPUT e:Cel	litext) THEN	ann (100 - 01 - 100)	areo a dana Key,		
			190	moBSCusto	mer:Refresh().				
			191	RETURN.					-
			-						- E
1			🖌 🖾 Console 🔝	Problems 💠 🅢 Taska				>	° ° (
-0.55		_			una de				
0 9					witikable	insert 19	0136		

When I leave the cell, the **AfterCellUpdate** event fires to run the **FormatColumn** method and refresh the displayed value. That's the last of the changes I'll make to this demonstration of the basics of updating data using the ProBindingSource.

Let's quickly review a few of the things I've shown you in this two-part session:

- Use the **TRACKING-CHANGES** property of a ProDataSet to allow you to **FILL** the DataSet and then keep track of changes to its contents.
- The save-row-changes method saves changes back to the source database table. ACCEPT-ROW-CHANGES marks those changes as being accepted in the dataset itself, and REJECT-ROW-CHANGES removes them.
- In the ProBindingSource, the AllowEdit property lets you manage whether UI controls that are bound to it are enabled for input or not.
- The AutoUpdate property is there just to help you do quick testing of updating data, but should be left False for serious development.
- Use the ProBindingSource RowModified property to check whether a row in the UI has actually been changed, and the Assign method to write changes back to the buffer in the Model that is the binding source's data source.
- Remember that the binding source AutoSync property allows an automatic refresh of data when the underlying query is re-opened or repositioned.
- When you need to synchronize a change that doesn't reposition the query, use the ProBindingSource **Refresh** method to push the change out to the UI.
- Use control events like BeforeRowUpdate and AfterCellUpdate to capture changes that you need to write back to the Model.
- Learn about useful control properties like RowUpdateCancelAction to take advantage of built-in behavior that you want in your user interface.

That concludes this two-part session on managing data updates with the ProBindingSource.