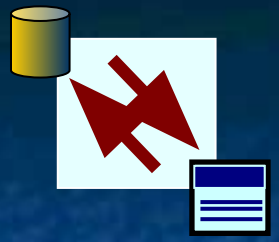


Об использовании класса CursorAdapter в Microsoft® Visual FoxPro® приложениях

Фирма: ИИС, Пермь
Ссылка в Интернет: http://www.ics.perm.ru/
Докладчик: Михаил Дроздов
Электронный адрес: mailto:vfpdev@narod.ru
Интернет-страница: http://vfpdev.narod.ru

Уфа, 2006г.

Содержание

- Назаначение VFP-класса CursorAdapter (CA)
- Основные функциональные возможности CA-класса
- Особенности использования различных типов источников данных в CA-классе
- Настройка обновляемых данных в CA-классе (Auto-update)
- Настройка обновляемых данных в CA-классе (Advanced)
- Настройка обновляемых данных в CA-классе (XML)
- Примеры фрагментов кода для XML-источников данных
- Использование одного соединения и параметров в SQL-запросе для ADO-источника
- Назначение свойства __VFPSetup в CA-классе при использовании построителя в vcx/scx
- Журнал событий (DatasourceType=ADO,DeteEnvironment.AutoOpenTables=.T.)
- Схема обмена данными между клиентом и сервером
- Ссылки на ресурсы в Интернет

Назначение VFP-класса CursorAdapter (CA)

- Данный класс появился в VFP 8.0 и расширяет существовавшую ранее в рамках VFP-баз данных (dbc-фалов) технику работы через редактируемые представления (View) основанную на SQL-запросах к VFP-таблицам (Local View), а также к внешним ODBC-источникам (Remote View)
  - во-первых, на возможность работы с ADO и XML источниками данных,
  - во-вторых, на объектный подход, поскольку эта возможность представлена VFP-классом. Обладая большим количеством свойств, событий и методов, этот класс существенно расширяет и обогащает возможности, существовавшие ранее в рамках представлений (View).

  Таким образом, этот класс обеспечивает работу VFP-приложений с такими внешними источниками данных, как ODBC, ADO, XML и VFP-таблицами.
- С помощью его вы можете абстрагироваться от конкретики внешнего источника и работать с данными независимо от того, какой именно источник данных вами используется, т.е. взаимодействовать с внешними данными исключительно только через методы/свойства CA-класса. Естественно, что значения свойств, также как и код в методах/событиях экземпляра CA-класса, будут зависимы от обслуживаемого внешнего источника данных.
- Средствами этого класса внешние данные могут быть как преобразованы во временные VFP-курсоры, так и наоборот, изменённые данные из VFP-курсоров могут быть преобразованы и переданы на соответствующие внешние источники.
- В своей работе данный класс может использовать классы-помощники там, где средств самого CA-класса недостаточно. Так, при работе с XML-данными обычно требуется использование VFP-класса XMLAdapter, а при работе с MS ADO (Microsoft® ActiveX® Data Objects) возникает необходимость в использовании объектов, входящих в объектную модель ADODB: ADODB.Connection, ADODB.Recordset, ADODB.Command
- Данные в VFP-курсорах, получаемых средствами CA представляют из себя «локальную копию» внешних данных и всегда являются буферизованными. А VFP-функция TABLEUPDATE(), сохраняющая изменения клиента из VFP-курсоров на свои внешние источники, поддерживает работу с CA

Основные функциональные возможности CA-класса

- **Свойство DataSourceType**. CA-класс поддерживает работу с внешними источниками данных: ODBC, ADO, XML, VFP-Narive. Какой именно источник данных обслуживается экземпляром класса определяется значением свойства DataSourceType
- **Метод CursorFill()** – выполняет команду (из свойства SelectCmd), чтобы заполнить данными курсор (свойство Alias). Заполнение курсора может выполняться с использование метода AutoOpen(), если класс CA помещён в экземпляр класса DataEnvironment (DE). При выполнение метода CursorFill() также вызываются соответствующие события: BeforeCursorFill(), AfterCursorFill()
- **Метод AutoOpen()** – вызывает метод CursorFill(), по умолчанию без параметров. Если CA-класс принадлежит экземпляру класса DE, то метод DE.OpenTables() вызывает метод AutoOpen() для каждого экземпляра CA. Такой механизм позволяет (при DE.AutoOpenTables=.T.) как открыть данные формы до наступления события Load() формы, так и более точно настроить параметры вызова метода CursorFill() в случае необходимости, например при параметризованных запросах для ADO-источника данных.
- **Метод CursorRefresh()** – обновляет данные в курсоре (свойство Alias) данными с сервера, повторно выполняя команду (из свойства SelectCmd) в соответствии типом источника данных (свойства DataSourceType) При выполнение этого метода также вызываются соответствующие события: BeforeCursorRefresh(), AfterCursorRefresh().
- **Метод RecordRefresh()** – обновляет данные указанной(ых) записи(ей) в курсоре (свойство Alias) данными с сервера. Этот метод интегрирован с функцией REFRESH(). Команда обновления записи может быть уточнена с помощью свойства RefreshCmd При выполнение метода также вызываются соответствующие события: BeforeRecordRefresh(), AfterRecordRefresh().
- **Методы: CursorAttach(),CursorDetach()** – позволяют подсоединиться/отсоединиться к/от заданному курсору, чтобы автоматически заполнить свойства экземпляра CA-класса, взяв их из указанного курсора.
- **Свойство: CursorSchema** – позволяет (пере)определить структуры данных источника. В VFP 8.0 используется для XML-данных, в VFP 9.0 распространяется на все типы данных. В VFP 9.0 значение свойства CA.UseCursorSchema определяет значение по умолчанию для передачи как параметра lUseSchema в метод CursorFill(), если этот параметр не использован при обращении к этому методу
- **Свойство NoData** - в VFP 9.0 определяет значение по умолчанию, передаваемое в качестве lNoData параметра методу CursorFill() в том случае, если этот параметр не использован при обращении к этому методу.
- **и т.д.** См. описания в документации…

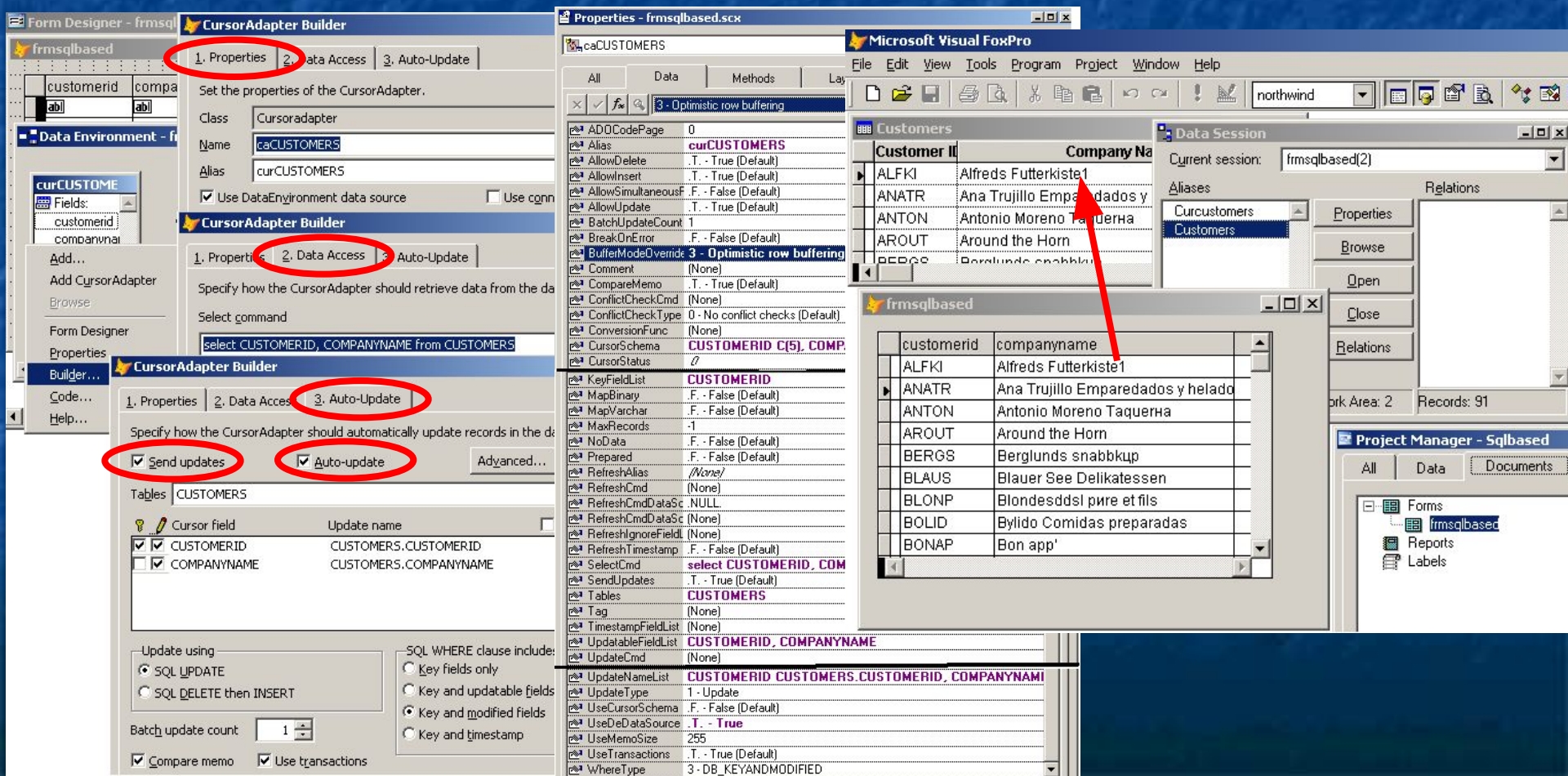
- CA-класс тесно интегрирован с функциями: **CURSORSETPROP(), CURSORGETPROP(), TABLEUPDATE()**
- **См. также ряд статей Юрия ШутенкоСм. также ряд статей Юрия Шутенко на начиная с «Часть I Создание CursorAdapter'а с помощью построителя».**

Особенности использования различных типов источников данных в CA-классе

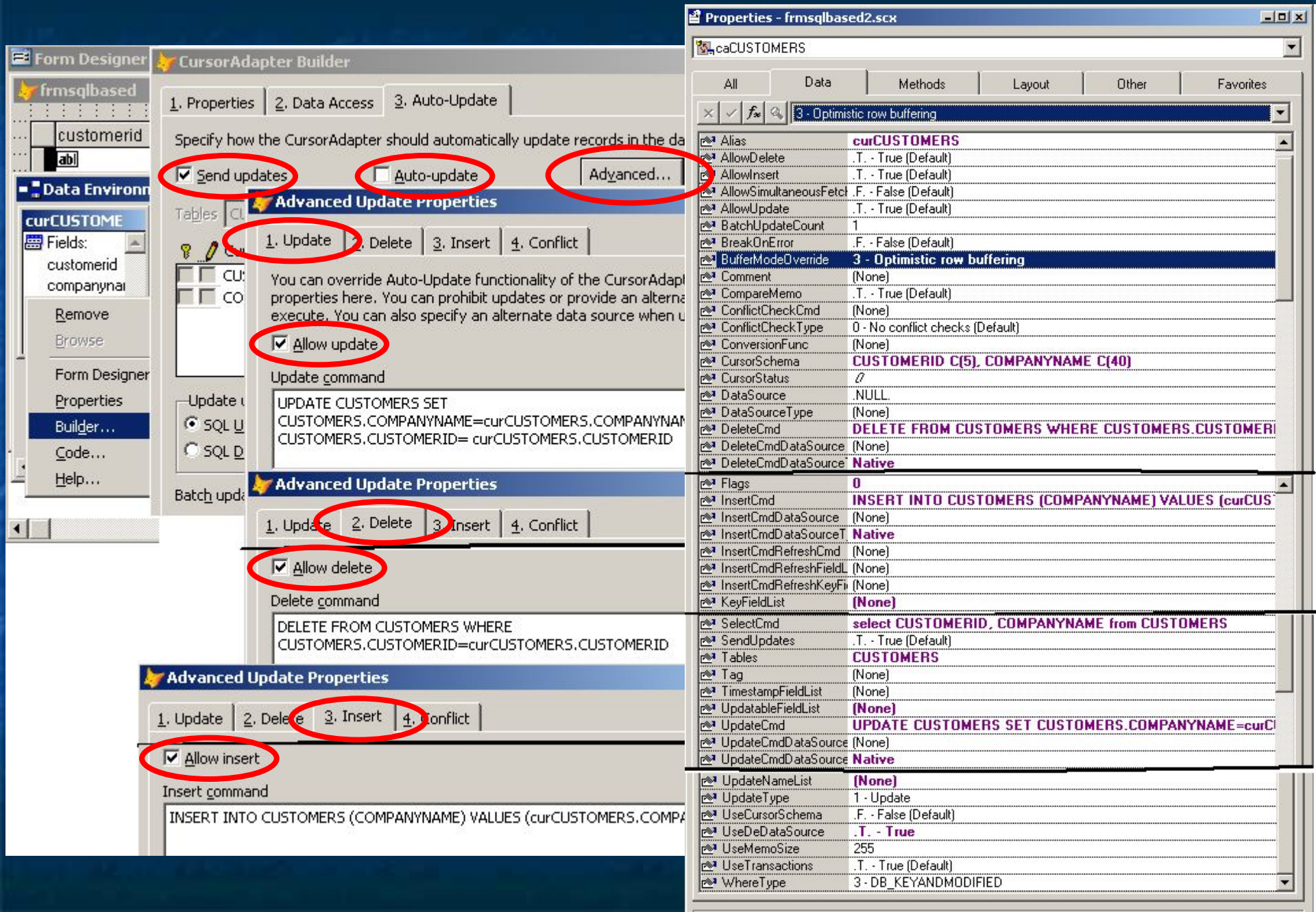
- Свойства: ***Tables, KeyFieldList, UpdateNameList, UpdatableFieldList*** обеспечивают настройку обновления данных (могут быть настроены на закладке [Auto-Update] в CA Builder). Если эти свойства неопределенны, то предполагается указание явного механизма обновления данных, например используя SQL-команды в качестве свойств: UpdateCmd, InsertCmd, DeleteCmd при соответствующих *CmdDataSource и *CmdDataSourceType (по кнопке Advanced… на закладке [Auto-Update] в CA Builder)
- Значение свойства ***DataSourceType*** определяет тип источника данных. Возможные значения строки: 'ODBC', 'ADO', 'XML', 'Native'.
- ***Native*** – означает, что источниками данных являются VFP-таблицы. Единственно что здесь нужно хорошо осознавать, - это то, что мы обращаемся не к VFP-таблицам непосредственно (чрез открытие её по команде USE), а к буферизованным VFP-курсорам, являющимся результатами выполнения SQL-команд к соответствующим VFP-таблицам. Как следствие в частности, имя полученного курсора (CA.Alias) не может совпадать с именем VFP-тблицы (в команде SQL-SELECT … FROM tblName)
- ***ODBC*** – внешним источник является совместимые с Open Database Connectivity приложения, доступ к данным которых, обеспечивается через соответствующие ODBC driver-ы
  - Значением свойства CA.DataSource в этом случае, является числовое значение, возвращаемые функцией SQLSTRINGCONNECT() или SQLCONNECT()
  - Если к примеру, для получение данных с сервера используется хранимая процедура, то значением свойства SelectCmd может быть команда обращения к такой процедуре
- ***ADO*** – определяет, что внешним источником является Microsoft® ActiveX® Data Objects (ADO). Это может быть совместимое с MS OLE DB Provider приложение, обеспечивающее стандартный интерфейс для работы с данными, например SQL Server (SQLOLEDB), либо приложение совместимое с Microsoft OLE DB Provider for ODBC (MSDASQL).
  - В этом случае, в качестве значения свойства CA.DataSource принимается объект ADODB.Recordset
  - Т.к. объект ADODB.Recordset не имеет возможности работать с параметрами, то по умолчанию метод CA.CursorFill() неспособен обрабатывать команды, содержащие параметры. Поэтому, в случае параметризованных запросов к данным, вам следует переопределить метод CA.CursorFill() так, чтобы создать дополнительный объект ADODB.Command и передать его чрез параметр методу CursorFill(), тем самым обеспечить обработку параметров в запросе.
  - Было обнаружено, что один общий ADODB.Recordset не способен обрабатывать несколько экземпляров CA-классов. Т.е. не следует пытаться одним DataSource, помещённым например в DataEnvironment пытаться обслужить несколько CA-классов.
- ***XML*** – означает, что внешними источниками являются XML-данные (Extensible Markup Language).
  - В этом случае свойство CA.SelectCmd должно быть:
    - либо строкой-результатом работы функции CURSORTOXML(), (т.е. корректной строкой XML-данных для получения VFP-курсора применением функции XMLTOCURSOR())
    - либо строкой, вычисляемой до объекта XMLTable, например «this.oXmlAdapter.Tables(1)»
  - Если определены свойства: Tables, KeyFieldList, UpdateNameList, UpdatableFieldList и значения свойств UpdateCmdDataSourceType, InsertCmdDataSourceType, и DeleteCmdDataSourceType установлены в 'XML', то свойство CA.UpdateGram на момент сохранения изменений содержит XML-схему изменений в формате, принятом для SQLXML 3.0, а свойства: UpdateCmd, InsertCmd и DeleteCmd могут быть строками, вычисляющимися до методов, к которым следует обратиться при возникновении соответствующих событий.

Настройка обновляемых данных в CA-классе (Auto-update)

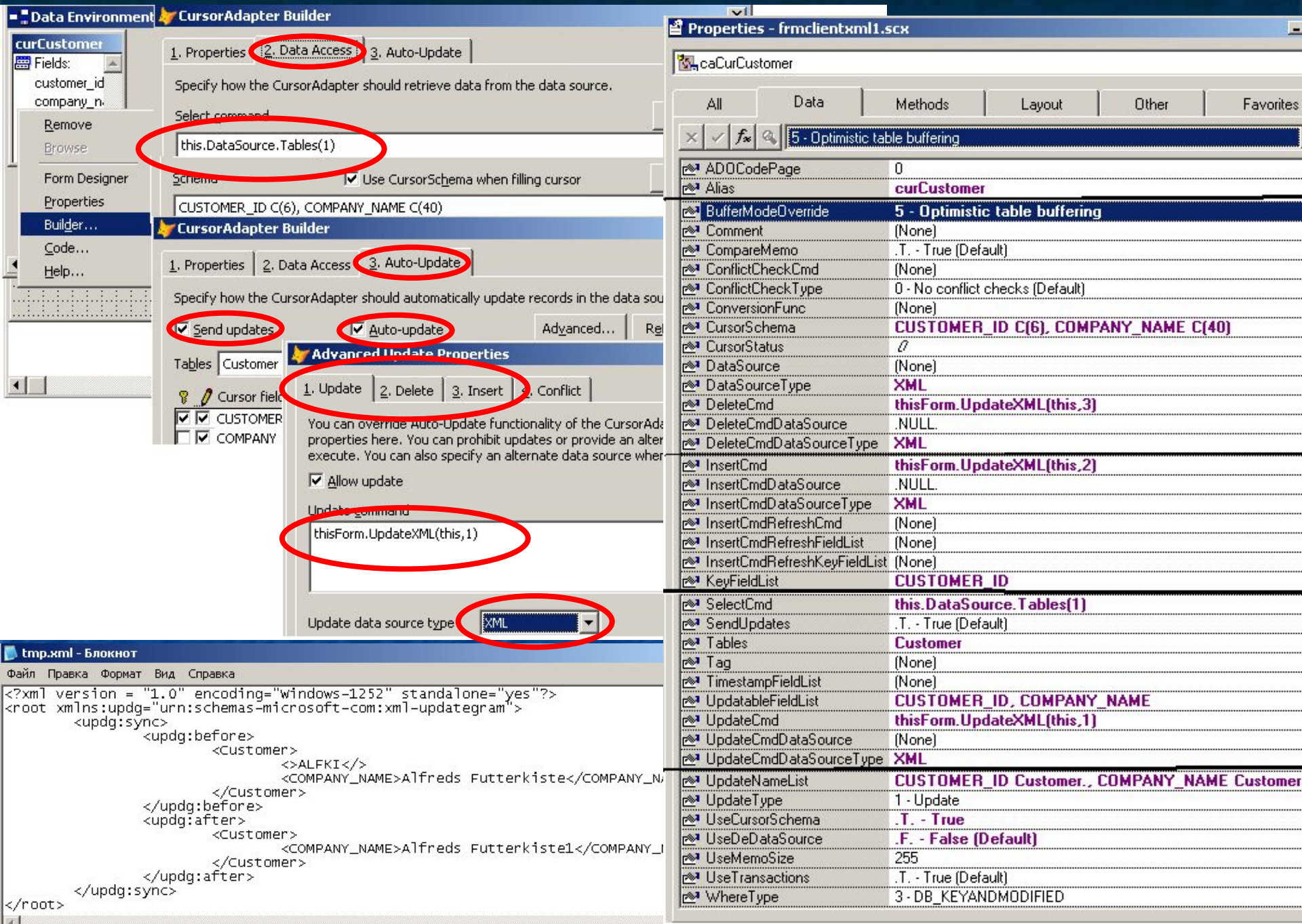
- Чтобы изменённые данные в курсорах, полученных через Local/Remote View могли быть сохранены в данных первоисточника, следует
  - - либо во View Designer на закладке [Update criteria] установить ключевые/редактируемые поля таблиц и отметить флажок SendUpdate
  - - либо используя функцию CURSORSETPROP() установить свойства: Tables, KeyFieldList, UpdateNameList, UpdatableFieldList, SendUpdates соотвествующим образом (см. статью в MDSN KB: "Q138094 How to Create Updatable Views by Using SQL Passthrough")
- В CA-классе имеется ряд аналогичных свойств: ***Tables, KeyFieldList, UpdateNameList, UpdatableFieldList***, - обеспечивающих настройку изменяемых данных.
- Изменённые данные из буферизованных Local/Remote View и из курсоров, полученных с помощью CA-классов, могут быть сохранены, используя функцию TableUpdate()



Настройка обновляемых данных в CA-классе (Advanced)



Настройка обновляемых данных в CA-классе (XML)



Примеры фрагментов кода для XML-источников данных

Object: Form1 Procedure: updatexml

```
LPARAMETERS toCA, tnCmdType
IF VARTYPE(toCA) # 'O'
    RETURN .F.
ENDIF
LOCAL loXA as XMLAdapter, lcUpdateGram as String
loXA = toCA.DataSource
IF VARTYPE(loXA) # 'O' OR !('Xmladapter' $ loXA.
    RETURN .F.
ENDIF
*////////////////////////////////
*// UpdateGram for SQLXML, ADO
lcUpdateGram = toCA.UpdateGram
IF EMPTY(lcUpdateGram)
    RETURN .F.
ENDIF
*////////////////////////////////
*// UpdateGram for .NET
lcNetUpdateGram = ""
LOCAL loXMLAdapter as XMLAdapter
loXMLAdapter = NEWOBJECT("XMLAdapter")
LOCAL llIncludeBefore as Boolean;
        ,llChangesOnly as Boolean;
        ,llIsFile as Boolean;
        ,lcSchemaLocation as String
    llIncludeBefore = .T.    && Include <diffgrar
    llChangesOnly = .T.      && Generate only cha
    llIsFile = .F.           && XML is a file.
    lcSchemaLocation = ""    && Generate inline s
WITH loXMLAdapter
    .AddTableSchema(toCA.Alias) && Add a new XMI
    *.UTF8Encoded = .T.      && Indicates Interna
    .RespectCursorCP = .T.   && ... or get CODEPA
    .ReleaseXML(.F.)         && Release XML docur
    .IsDiffgram = .T.        && Generate XML Diff
    .ToXml("lcNetUpdateGram", lcSchemaLocation,
ENDWITH
WITH loXA
    .RespectCursorCP = .T.
    .LoadXML(lcNetUpdateGram)                && 1
    .Tables(1).ApplyDiffgram(loXA.Alias)     && .
ENDWITH
```

Dataenvironment.BeforeOpenTables

Object: Dataenvironment Procedure: BeforeOpenTables View Parent C

```
#DEFINE C_SRV_PROC  HOME(8) + "cainde\caxml\server\serverxml.prg"
this.AddProperty('oSeData', NEWOBJECT('seData', C_SRV_PROC))
```

Object: caCurCustomer Procedure: CursorFill View Parent Code

```
LOCAL lbIsVfp9 as Boolean
lbIsVfp9 = (VERSION(5) > 800
IF VARTYPE(this.DataSource) # 'O'
    this.DataSource = NEWOBJECT('XmlAdapter')
ENDIF
IF TYPE('this.Parent.oSeData') = 'O' AND !ISNULL(this.Parent.oSeData)
    *
    *-- Is used alias
    LOCAL lbUsedAlias as Boolean
    lbUsedAlias = USED(this.Alias)
    *
    *-- Get XML-data
    LOCAL lcXml As String
    lcXml = this.Parent.oSeData.GetXmlData(.T., "caCustomer")
    IF EMPTY(lcXml)
        ERROR "Unexpected value return by oSeData.GetXmlData()"
        NODEFAULT
        RETURN .F.
    ENDIF
    *
    *-- SelectCmd
    this.DataSource.LoadXML(lcXml)
    this.SelectCmd = [this.DataSource.Tables(1)]
    *
    *-- CursorFill
    IF !lbUsedAlia
        IF lbIsVfp9
            DODEFAULT(luseCursorSchema, lNoData, nOptions, Source)
        ELSE
            = EVALUATE(this.SelectCmd + '.ToCursor()')
        ENDIF
    ELSE
        IF lbIsVfp9
            NODEFAULT
        ELSE
            DELETE FOR !DELETED() IN (this.Alias)
            = EVALUATE(this.SelectCmd + '.ToCursor(.T.)')
        ENDIF
    ENDIF
```

Создаём экземпляр класса XMLAdapter

Получаем XML-строку для таблицы

Загружаем XML-таблицу в XMLAdapter

Выполняем заполнение локального курсора

Использование одного соединения и параметров в SQL-запросе для ADO-источника

Object: caOrders Procedure: Init

```
*** Setup code: DO NOT REMOVE
local llReturn
do case
    case not pemstatus(This, '__VFPSetup', 5)
        This.AddProperty('__VFPSetup', 0)
    case This.__VFPSetup = 1
        This.__VFPSetup = 2
    case This.__VFPSetup = 2
        This.__VFPSetup = 0
        return
endcase
set multilocks on
llReturn = dodefault()
*** End of Setup code: DO NOT REMOVE
IF llReturn
    LOCAL loConnDataSource as ADODB.Connection
    IF TYPE('This.Parent.oConnection') = 'O' AND !ISNUL
        loConnDataSource = This.Parent.oConnection
    ELSE
        loConnDataSource = CREATEOBJECT("ADODB.Connecti
        loConnDataSource.ConnectionString = [Provider=S
            [nitial Catalog=Northwind;Data Source=(loca
        loConnDataSource.Open()
    ENDIF
    This.DataSourceType="ADO"
    This.DataSource = CREATEOBJECT("ADODB.Recordset")
    This.DataSource.CursorLocation    = 3   && adUseClier
    This.DataSource.LockType          = 3   && adLockOpti
    This.DataSource.ActiveConnection = loConnDataSource
ENDIF
*** Setup code: DO NOT REMOVE
if This.__VFPSetup = 1
    This.__VFPSetup = 2
ENDIF
return
*** En
```

В CA.Init() создаём/настраиваем ADODB.Recordset

В DE.BeforeOpenTables создаём/открываем ADODB.Connection

caOrders.AutoOpen

Object: caOrders Procedure: AutoOpen View Parent Code

```
*** Setup code: DO NOT REMOVE
if not pemstatus(This, '__VFPSetup', 5)
    This.AddProperty('__VFPSetup', 1)
    This.Init()
endif
*** End of Setup

WITH this
    .UseDeDataSource = .F.
    IF .DataSourceType = "ADO" ;
            AND ('?' $ .SelectCmd) ;
            AND VARTYPE(.DataSource) = 'O'
        LOCAL loCmd as ADODB.Command, lbIsFill as Boolean
        loCmd = CREATEOBJECT("ADODB.Command")
        loCmd.ActiveConnection = .DataSource.ActiveConnection
        NODEFAULT
        IF VERSION(5) < 900
            lbIsFill = .CursorFill(.F., .F., 1, loCmd) && (adCmdText = 1)
        ELSE
            lbIsFill = .CursorFill(.UseCursorSchema, .NoData, 1, loCmd)
        ENDIF
        IF !lbIsFill
            .Error(1098, .Name + '.AutoOpen', LINENO(1))
            RETURN .F.
        ENDIF
    ENDIF
ENDWITH
```

В CA.AoutoOpen() создаём/используем ADODB.Command, при явном вызове CursorFill()

```
            LOCAL lbIsConnection as Boolean
            .AddProperty('oConnection', CREATEOBJECT('ADODB.Connection'))
            IF TYPE('this.oConnection') # 'O' OR ISNULL(.oConnection)
                RETURN .F.
            ENDIF
            WITH this.oConnection
                .ConnectionString = lcConnectionString
                .Open()
                lbIsConnection = (.State <> 0)
            ENDWITH
            IF !lbIsConnection
                .oConnection = NULL
            ENDIF
        ENDIF
    ENDWITH
ENDIF
```

Назначение свойства __VFPSetup в CA-классе при использовании построителя в vcx/scx

Если при создании класса CA вы воспользовались построителем (Builder), то наверняка обратили внимание на код, помещаемый построителем в метод AutoOpen() и событие Init(), где динамически создаётся свойство __VFPSetup. Это делается по следующим причинам:

- При использовании CA-класса в классе DataEnvironment(DE) все источники данных должны быть открыты во время выполнения метода DE.OpenTables().
- Если при этом DE-класс принадлежит форме и если у него DE.AutoOpenTables=.T. (т.е. значение, принятое по умолчанию), то метод DE.OpenTables() вызывается ещё до наступления события Form.Load(). Это делается для того, чтобы открыть все используемые в форме данные ещё до создание контролов формы, но это и нарушает общепринятую очерёдность создания объектов и событий для классов-контейнеров, т.к. к этому моменты экземпляр класса DE принадлежащий классу формы ещё не создан.
- В этом случае, и метод CA.AutoOpen(), вызываемый из DE.OpenTables(), также нарушает общепринятую очерёдность событий, т.к. к моменту его выполнения экземпляр класса CA, принадлежащий классу DE, ещё не создан.
- Наконец, событие Init() любого класса-контейнера должно происходить «лишь однажды» и «в самую последнюю очередь», т.е. тогда, когда все объекты, принадлежащие этому классу-контейнеру, уже существуют (т.е. успешно отработали их события Init()), но «перед выполнением каких-либо событий/методов объекта». Кодом в событии Init() обычно пользуются, чтобы выполнить окончательную настройку класса перед началом его работы (в определённом смысле «конструктор» объекта).

Добавленный построителем «условный код», включая и динамически создаваемое дополнительное свойство _VFPSetup, в начало метода CA.AutoOpen() и в событие CA.Init(), является попыткой решить перечисленные проблемы, т.е. обеспечивает вызов события CA.Init() лишь однажды в т.ч. и до начала выполнения «прикладного кода» в CA.AutoOpen()

Журнал событий (DatasourceType=ADO,DeteEnvironment.AutoOpenTables=.T.)

DataEnvironment.**OpenTables**() [if DataEnvironment.AutoOpenTables=.T.]
DataEnvironment.**BeforeOpenTables**()
caCustomers.**AutoOpen**()
caCustomers.**Init**() [explicit call from AutoOpen()]
caCustomers.**CursorFill**(.F.,.F.,.F.,.F.)
caCustomers.**BeforeCursorFill**(.F.,.F.,select CustomerID, CompanyName from Customers)
caCustomers.**AfterCursorFill**(.F.,.F.,select CustomerID, CompanyName from Customers,.T.)
caOrders.**AutoOpen**()
caOrders.**Init**() [explicit call from AutoOpen()]
caOrders.**CursorFill**(.F.,.F.,1,[Object]) [Object!=.F., if ADODB.Command in AutoOpen() is used]
caOrders.**BeforeCursorFill**(.F.,.F.,select OrderID, OrderDate, CustomerID from Orders WHERE Orders.CustomerID=?Customers.CustomerID)
caOrders.**AfterCursorFill**(.F.,.F.,select OrderID, OrderDate, CustomerID from Orders WHERE Orders.CustomerID=?Customers.CustomerID,.T.)
caOrder_details.**AutoOpen**()
caOrder_details.**Init**() [explicit call from AutoOpen()]
caOrder_details.**CursorFill**(.F.,.F.,1,[Object]) [Object!=.F., if ADODB.Command in AutoOpen() is used]
caOrder_details.**BeforeCursorFill**(.F.,.F.,select ProductID, Quantity, UnitPrice, OrderID from [Order Details] WHERE [Order Details].OrderID=?Orders.OrderID)
caOrder_details.**AfterCursorFill**(.F.,.F.,select ProductID, Quantity, UnitPrice, OrderID from [Order Details] WHERE [Order Details].OrderID=?Orders.OrderID,.T.)
Form.**Load**()
caCustomers.**Init**() [skiped call]
caOrders.**Init**() [skiped call]
caOrder_details.**Init**() [skiped call]
DataEnvironment.**Init**()
Form: **Init** all form controls()
Form.**Init**()
caOrders.**CursorRefresh**() [as example of some operation through user interface]
caOrders.**BeforeCursorRefresh**(select OrderID, OrderDate, CustomerID from Orders WHERE Orders.CustomerID=?Customers.CustomerID)
caOrders.**AfterCursorRefresh**(select OrderID, OrderDate, CustomerID from Orders WHERE Orders.CustomerID=?Customers.CustomerID,.T.)
caOrder_details.**CursorRefresh**() [as example of some operation through user interface]
caOrder_details.**BeforeCursorRefresh**(select ProductID, Quantity, UnitPrice, OrderID from [Order Details] WHERE [Order Details].OrderID=?Orders.OrderID)
caOrder_details.**AfterCursorRefresh**(select ProductID, Quantity, UnitPrice, OrderID from [Order Details] WHERE [Order Details].OrderID=?Orders.OrderID,.T.)
form.**Destroy**()
form: **Destroy** all form controls()
DataEnvironment.**CloseTables**() [if DataEnvironment.AutoCloseTables=.T.]
DataEnvironment.**AfterCloseTables**()
DataEnvironment.**Destroy**()
caOrder_details.**Destroy**()
caOrder_details.**BeforeCursorClose**(order_details)
caOrder_details.**AfterCursorClose**(order_details,.T.)
caOrders.**Destroy**()
caOrders.**BeforeCursorClose**(orders)
caOrders.**AfterCursorClose**(orders,.T.)
caCustomers.**Destroy**()
caCustomers.**BeforeCursorClose**(customers)
caCustomers.**AfterCursorClose**(customers,.T.)

Схема обмена данными между клиентом и сервером

Ссылки на ресурсы в Интернет

- Статьи Московского издательства FoxTalk на http://newsletter.narod.ru/foxtalk/FoxTalk.htm
  - «Извлечение и обновление данных при помощи CursorAdapters», Дуг Хенниг http://newsletter.narod.ru/foxtalk/oct2003.htm#oct01 «Извлечение и обновление данных при помощи CursorAdapters», Дуг Хенниг http://newsletter.narod.ru/foxtalk/oct2003.htm#oct01 www.newsletter.narod.ru/foxtalk/pdf2publish/2003-10.pdf
  - «Многократно используемые классы данных», Дуг Хенниг http://newsletter.narod.ru/foxtalk/nov2003.htm#nov01 «Многократно используемые классы данных», Дуг Хенниг http://newsletter.narod.ru/foxtalk/nov2003.htm#nov01 www.newsletter.narod.ru/foxtalk/pdf2publish/2003-11.pdf
- Ряд статей Юрия Шутенко на http://kodu.neti.ee/~juri4/vfp60/index_ru.htmРяд статей Юрия Шутенко на http://kodu.neti.ee/~juri4/vfp60/index_ru.htm начиная с http://kodu.neti.ee/~juri4/vfp60/ca_01_ru.htm
- Примеры кода на стр. http://vfpdev.narod.ru/util_r.html
  - cainde.zip [11.05.2006] (56,0KB) - примеры кода для VFP 8.0 (или выше) использования нескольких классов CursorAdapter, содержащих параметризованные SQL-запросы…
  - xasampls.zip [13.05.2006] (8,0KB) - примеры кода для VFP 9.0 (или выше) использования VFP-класса XMLAdapter. Показано использование для получения/применения xml-diffgram…
- Обсуждение вопросов, связанных с использованием класса CursorAdapter на форуме http://forum.foxclub.ru/Обсуждение вопросов, связанных с использованием класса CursorAdapter на форуме http://forum.foxclub.ru/ Используя страничку