

FoxTalk

Октябрь 2003

№ 10 (76)

русское издание

Solutions for Microsoft® FoxPro® and Visual FoxPro® Developers

Извлечение и обновление данных при помощи CursorAdapters

WIN



Дуг Хенниг (Doug Hennig)

Октябрь 2003

- **Извлечение и обновление данных при помощи CursorAdapters** 1
Дуг Хенниг
- **The Straight POOP: Вся энчилада целиком** 6
Нэнси Фолсом
- **Playing With the GUI in VFP 7: Градиентные и активные кнопки** 10
Предраг Боснич
- **The Kit Box: С дерева вы можете обозреть весь лес!** 13
Энди Крамек и Марсиа Акинз
- **Глядите-ка! Это Outlook!** 17
Уилл Хентцен
- **Стань плодотворнее с VFP, часть 1** 21
Ричард А. Шуммер



материал имеет отношение к соответствующей версии

UNIX MAC DOS WIN

материал имеет отношение к соответствующей платформе

DOWNLOAD

исходные тексты программ можно скачать из Интернета

Добавленный в VFP 8 новый базовый класс *CursorAdapter* обеспечивает последовательный, простой в использовании интерфейс доступа к данным. В сегодняшней статье Дуг Хенниг показывает, как использовать *CursorAdapter* для доступа к локальным и удаленным данным через ODBC, ADO и XML и поясняет особые требования для каждого из этих механизмов доступа.

Как я уже отмечал в предыдущей статье «Переделайте ваши курсоры в *CursorAdapter*», одной из наиболее важных и волнующих функций VFP 8 является новый базовый класс *CursorAdapter*. В той статье мы рассмотрели свойства, события и методы *CursorAdapter*, а также его преимущества по сравнению с использованием удаленных представлений (remote view), транзитной пересылки SQL (SQL Pass Through), ADO и XML.

Прежде чем вы начнете использовать *CursorAdapter*, вам необходимо знать о специфических требованиях этого класса при использовании его для доступа к собственным данным или удаленным данным через ODBC, ADO и XML. В сегодняшней статье приводится детальное описание работы с каждым типом источников данных.

Использование собственных данных

Хотя ясно, что *CursorAdapter* предназначен для стандартизации и упрощения доступа к данным не VFP-формата, вы можете использовать его в качестве замены для объекта *Cursor*, устанавливая свойство *DataSourceType* в значение “Native”. Зачем это нужно? Главным образом в расчете на будущее использование вашего приложения — путем простого изменения *DataSourceType* на один из других вариантов (и возможного изменения некоторых свойств, таких как настройка информации о соединении), вы можете легко переходить к другому формату, такому как SQL Server.

Когда *DataSourceType* установлено в значение “Native”, VFP игнорирует свойство *DataSource*. Свойство *SelectCmd* должно хранить оператор SQL SELECT (а не команду USE или выражение), что означает, что вы всегда работаете с эквивалентом локального

представления, а не напрямую с таблицей. Вы отвечаете за то, чтобы VFP могла найти любую таблицу, указанную в операторе SELECT. Если таблицы не находятся в текущем каталоге, то вам нужно либо задать путь, либо открыть ту базу данных, которой принадлежит эта таблица. Как обычно, если вы хотите, чтобы курсор был обновляемым, убедитесь, что установили свойства обновления (KeyFieldList, Tables, UpdatableFieldList и UpdateNameList).

Следующий пример (NativeExample.prg, включенный на дискету) создает обновляемый курсор из таблицы Customer в примере базы данных VFP — TestData:

```
local loCursor as CursorAdapter, ;
    laErrors[1]
open database (_samples + 'data\testdata')
loCursor = createobject('CursorAdapter')

with loCursor
    .Alias          = 'customercursor'
    .DataSourceType = 'Native'
    .SelectCmd      = ;
        "select CUST_ID, COMPANY, CONTACT from CUSTOMER " + ;
        "where COUNTRY = 'Brazil'"
    .KeyFieldList   = 'CUST_ID'
    .Tables         = 'CUSTOMER'
    .UpdatableFieldList = 'CUST_ID, COMPANY, CONTACT'
    .UpdateNameList = 'CUST_ID CUSTOMER.CUST_ID, ' + ;
        'COMPANY CUSTOMER.COMPANY, CONTACT CUSTOMER.CONTACT'

    if .CursorFill()
        browse
        tableupdate(1)
    else
        aerror(laErrors)
        messagebox(laErrors[2])
    endif .CursorFill()
endwith

close databases all
```

Использование ODBC

Фактически ODBC является наиболее простым из четырех возможных вариантов значения свойства DataSourceType. Вы задаете в качестве DataSource указатель на открытое ODBC-соединение, настраиваете обычные свойства и вызываете CursorFill для получения данных. Если вы заполняете свойства KeyFieldList, Tables, UpdatableFieldList и UpdateNameList, то VFP автоматически создаст соответствующие операторы UPDATE, INSERT и DELETE для внесения любых изменений в серверную часть СУБД. Если вы хотите использовать вместо этого хранимые процедуры, установите свойства *Cmd, *CmdDataSource и *CmdDataSourceType соответственно (замените "*" на "Delete", "Insert" и "Update").

Вот пример, взятый из ODBCExample.prg, вызывающий хранимую процедуру CustOrderHist базы данных Northwind (поставляемой вместе с SQL Server) для получения общего числа единиц продукта для определенного заказчика:

```
local lcConnString, ;
    loCursor as CursorAdapter, ;
    laErrors[1]
lcConnString = 'driver=SQL Server;server=(local); ' + ;
    'database=Northwind;uid=sa;pwd=;trusted_connection=no'

* изменяем пароль на соответствующее значение
* для вашей базы данных
loCursor = createobject('CursorAdapter')
with loCursor
    .Alias          = 'CustomerHistory'
    .DataSourceType = 'ODBC'
    .DataSource     = sqlstringconnect(lcConnString)
    .SelectCmd      = "exec CustOrderHist 'ALFKI'"
    if .CursorFill()
        browse
    else
        aerror(laErrors)
        messagebox(laErrors[2])
    endif .CursorFill()
endwith
```

Использование ADO

Использование ADO в качестве механизма доступа совместно с CursorAdapter вызывает большее число проблем, чем использование ODBC:

- В качестве DataSource должен быть задан ADO RecordSet, свойству ActiveConnection которого соответствует открытое соединение (объект ADO Connection).
- Если вы хотите использовать параметризованный запрос (являющийся, вероятно, более распространенным случаем, чем получение всех записей), то пересылаете объект ADO Command в качестве четвертого параметра для CursorFill. VFP позаботится о заполнении для вас коллекции Parameters объекта Command (в поисках параметра анализируется SelectCmd), но, конечно, переменные, содержащие значения параметров, должны быть в области видимости.
- Использовать CursorAdapter с ADO в DataEnvironment достаточно просто: вы можете установить, если желаете, для UseDEDataSource значение .T., а затем задать свойства DataSource и DataSourceType для DataEnvironment также, как и в случае с CursorAdapter. Однако это не будет работать, если существует более одного объекта CursorAdapter в DataEnvironment. Причина заключается в том, что ADO RecordSet, на который ссылается DataEnvironment.DataSource, может содержать данные только одного CursorAdapter. Когда вы вызываете CursorFill для второго CursorAdapter, вы получаете сообщение об ошибке «RecordSet is already open» (RecordSet уже открыт). Так, если ваш DataEnvironment имеет больше чем один CursorAdapter, то вы должны установить для UseDEDataSource значение .F. и вручную управлять свойствами DataSource и DataSourceType для каждого CursorAdapter (или,

возможно, использовать подкласс DataEnvironment для управления этими свойствами).

Приведенный ниже пример кода, взятый из ADOExample.prg, показывает, как извлекать данные, используя параметризованный запрос с помощью объекта ADO Command. Этот пример также демонстрирует применение новых структурированных функций обработки ошибок в VFP 8. Вызов метода ADO Connection Open включен в оператор TRY ... CATCH ... ENDTRY для перехвата ошибки COM, вызываемой этим методом в случае, если он не сработает.

```
local loConn as ADOConnection, ;
    loCommand as ADOCommand, ;
    loException as Exception, ;
    loCursor as CursorAdapter, ;
    lcCountry, ;
    laErrors[1]
loConn = createobject('ADOConnection')
with loConn
    .ConnectionString = ;
        'provider=SQLOLEDB.1;data source=(local);' + ;
        'initial catalog=Northwind;uid=sa;pwd=' + ;
        'trusted_connection=no'
    * изменяем пароль на соответствующее значение
    * для вашей базы данных
    try
        .Open()
    catch to loException
        messagebox(loException.Message)
        cancel
    endtry
endwith
loCommand = createobject('ADOCommand')
loCursor = createobject('CursorAdapter')
with loCursor
    .Alias = 'Customers'
    .DataSourceType = 'ADO'
    .DataSource = createobject('ADORecordSet')
    .SelectCmd = ;
        'select * from customers where country=?lcCountry'
    lcCountry = 'Brazil'
    .DataSource.ActiveConnection = loConn
    loCommand.ActiveConnection = loConn
    if .CursorFill(.F., .F., 0, loCommand)
        browse
    else
        aerror(laErrors)
        messagebox(laErrors[2])
    endif .CursorFill(.F., .F., 0, loCommand)
endwith
```

Использование XML

Использование XML с CursorAdapters требует некоторых дополнительных тонких настроек. Вот некоторые из возникающих проблем:

- Свойство DataSource игнорируется.
- Свойство CursorSchema должно иметь значение (даже если вы передаете .F. в качестве первого параметра для метода CursorFill), в противном случае вы получите сообщение об ошибке.
- Свойству SelectCmd должно соответствовать выражение, такое как пользовательская функция или имя метода объекта, возвращающее XML для курсора.

- Изменения, сделанные в курсоре, преобразуются в DiffGram, являющееся XML, содержащим значения «до» и «после» для измененных полей и записей, которые помещаются в свойство DiffGram когда требуется обновление.
- Чтобы записать изменения в источник данных, UpdateCmdDataSourceType должно быть установлено в значение "XML" и свойству UpdateCmd должно соответствовать выражение (опять, наиболее вероятно, это будет пользовательская функция или метод объекта), обрабатывающее это обновление. Вы, возможно, захотите переслать "This.DiffGram" в пользовательскую функцию таким образом, чтобы она могла послать эти изменения в источник данных.

Источник XML для курсора может иметь разное происхождение. Например, вы можете вызвать пользовательскую функцию, преобразующую VFP-курсор в XML, используя CURSORTOXML(), и возвращающую следующий результат:

```
use CUSTOMERS
cursortoxml('customers', 'lcXML', 1, 8, 0, '1')
return lcXML
```

Пользовательская функция может вызывать службу Web Service, возвращающую результирующий набор в виде XML. Вот пример, который IntelliSense сгенерировала для меня из службы Web Service, созданной и зарегистрированной на моей собственной системе (детали не так важны, просто здесь показан пример службы Web Service).

```
loWS = newobject('Wsclient', ;
    home() + 'ffc\_webservices.vcx')
loWS.cwsName = 'dataserver web service'
loWS = loWS.SetupClient('http://localhost/' + ;
    'SQLDataServer/dataserver.WSDL', 'dataserver', ;
    'dataserverSoapPort')
lcXML = loWS.GetCustomers()
return lcXML
```

Пример может использовать SQLXML 3.0 для выполнения запроса к SQL Server 2000, хранящегося в файле-шаблоне на Web-сервере (для получения дополнительной информации по SQLXML, посетите сайт <http://msdn.microsoft.com> и проведите поиск по ключевому слову "SQLXML"). Следующий код использует объект MSXML2.XMLHTTP для получения всех записей из таблицы Customers базы данных Northwind по протоколу HTTP; более подробное объяснение я приведу позднее.

```
local loXML as MSXML2.XMLHTTP
loXML = createobject('MSXML2.XMLHTTP')
loXML.open('POST', 'http://localhost/northwind/' + ;
    'template/getallcustomers.xml', .F.)
loXML.setRequestHeader('Content-type', 'text/xml')
loXML.send()
return loXML.responseText
```

Обработка обновлений — более сложная задача. Источник данных должен либо быть способным принимать и потреблять DiffGram (как в случае с SQL Server 2000), либо вы должны вычислить эти изменения самостоятельно и создать серию SQL-операторов (UPDATE, INSERT и DELETE) для проведения обновлений.

Вот пример (XMLExample.prg), использующий CursorAdapter с источником данных XML. Обратите внимание, что как SelectCmd, так и UpdateCmd вызывают пользовательские функции. В случае с SelectCmd, получаемый customer ID пересылается пользовательской функции, называемой GetNWCcustomers, которую мы скоро рассмотрим. Для UpdateCmd, VFP пересылает свойство DiffGram к SendNWXML, которое мы также рассмотрим чуть позже.

```
local loCustomers as CursorAdapter, ;
laErrors[1]
```

```
loCustomers = createobject('CursorAdapter')

with loCustomers
  .Alias                = 'Customers'
  .CursorSchema        = ;
  'CUSTOMERID C(5), COMPANYNAME C(40), ' + ;
  'CONTACTNAME C(30), CONTACTTITLE C(30), ' + ;
  'ADDRESS C(60), CITY C(15), REGION C(15), ' + ;
  'POSTALCODE C(10), COUNTRY C(15), ' + ;
  'PHONE C(24), FAX C(24)'

  .DataSourceType      = 'XML'
  .KeyFieldList        = 'CUSTOMERID'
  .SelectCmd           = 'GetNWCcustomers([ALFKI])'
  .Tables              = 'CUSTOMERS'
  .UpdatableFieldList = ;
  'CUSTOMERID, COMPANYNAME, CONTACTNAME, ' + ;
  'CONTACTTITLE, ADDRESS, CITY, REGION, ' + ;
  'POSTALCODE, COUNTRY, PHONE, FAX'
  .UpdateCmdDataSourceType = 'XML'
  .UpdateCmd           = 'SendNWXML(This.DiffGram)'
  .UpdateNameList     = ;
  'CUSTOMERID CUSTOMERS.CUSTOMERID, ' + ;
  'COMPANYNAME CUSTOMERS.COMPANYNAME, ' + ;
  'CONTACTNAME CUSTOMERS.CONTACTNAME, ' + ;
  'CONTACTTITLE CUSTOMERS.CONTACTTITLE, ' + ;
  'ADDRESS CUSTOMERS.ADDRESS, ' + ;
  'CITY CUSTOMERS.CITY, ' + ;
  'REGION CUSTOMERS.REGION, ' + ;
```

Настройка XML-доступа в SQL Server 2000

Для того чтобы подключиться к SQL Server 2000, используя URL в браузере или ином HTTP-клиенте, мы должны сделать несколько вещей. Прежде всего, вам необходимо загрузить и установить SQLXML 3.0 с Web-сайта MSDN (<http://msdn.microsoft.com>), выполнив поиск по ключевому слову “SQLXML”, а затем выбрав ссылку для загрузки.

Далее вам следует настроить виртуальный каталог IIS. Чтобы сделать это, выберите значок Configure IIS Support в папке SQLXML 3.0 меню Programs, открываемого с помощью кнопки Start панели задач Windows. Раскройте узел вашего сервера, выберите Web-сайт, с которым будете работать, затем щелкните его правой кнопкой и выберите опцию New | Virtual Directory. На вкладке General появившегося диалогового окна введите имя виртуального каталога и “NorthwindTemplates” в качестве физического каталога. Используя Windows Explorer, создайте где-либо на вашей системе физический каталог и создайте его подкаталог, называемый “Template” (мы скоро будем использовать его). Скопируйте два шаблонных файла, включенные в загружаемый файл этой статьи (GetAllCustomers.xml и CustomersByID.xml), в подкаталог Template.

На вкладке Security введите соответствующую информацию для доступа к SQL Server (например: имя пользователя и его пароль или специальный механизм аутентификации, который вы хотите использовать). На вкладке Data Source выберите SQL Server и, если потребуется, используемую базу данных. Например, для этой статьи выберите базу данных Northwind. На вкладке Settings установите требуемые настройки, но, как минимум, включите опции Allow Template Queries и Allow POST.

На вкладке Virtual Names выберите “template” в раскрывающемся списке Type и введите виртуальное имя (в

этой статье используйте “template”) и физический путь (который должен указывать на подкаталог виртуального каталога; для этой статьи используйте “Template”), чтобы использовать шаблоны. Щелкните ОК.

Для того чтобы проверить, что все настроено должным образом, соединитесь с SQL Server с помощью файла шаблона GetAllCustomers.xml, скопированного вами в подкаталог Template. Он имеет следующее содержание:

```
<root xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <sql:query client-side-xml="0">
    SELECT *
    FROM Customers
    FOR XML AUTO
  </sql:query>
</root>
```

Для того, чтобы провести проверку, запустите ваш браузер и введите в адресную строку следующий URL: <http://localhost/northwind/template/getallcustomers.xml>. В браузере вы должны увидеть содержание таблицы Customers базы данных Northwind в виде документа XML. Вот начало того, что вы должны увидеть:

```
<root xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <Customers CustomerID="ALFKI" CompanyName="Alfreds
  Futterkiste" ContactName="Maria Anders"
  ContactTitle="Sales Representative"
  Address="Obere Str. 57" City="Berlin" PostalCode="12209"
  Country="Germany" Phone="030-0074321"
  Fax="999-999-9999" />
```

Теперь вы уже готовы к запуску примеров SQLXML этой статьи.

```

        'POSTALCODE CUSTOMERS.POSTALCODE, ' + ;
        'COUNTRY CUSTOMERS.COUNTRY, ' + ;
        'PHONE CUSTOMERS.PHONE, ' + ;
        'FAX CUSTOMERS.FAX'

    if .CursorFill(.T.)
        browse
    else
        aerror(laErrors)
        messagebox(laErrors[2])
    endif .CursorFill(.T.)
endwith

```

Ниже приведен код для GetNWCcustomers. Он использует объект MSXML2.XMLHTTP для доступа на Web-сервере к XML-шаблону SQL Server 2000, называемому CustomersByID.xml, и возвращает эти результаты. Извлекаемый customer ID передается в этот код в качестве параметра.

```

lparameters tcCustID
local loXML as MSXML2.XMLHTTP
loXML = createobject('MSXML2.XMLHTTP')
loXML.open('POST', 'http://localhost/northwind/' + ;
    'template/customersbyid.xml?customerid=' + tcCustID, ;
    .F.)
loXML.setRequestHeader('Content-type', 'text/xml')
loXML.send()
return loXML.responseText

```

XML-шаблон, на который ссылается этот код (CustomersByID.XML), выглядит следующим образом:

```

<root xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <sql:header>
    <sql:param name="customerid">
    </sql:param>
  </sql:header>
  <sql:query client-side-xml="0">
    SELECT *
    FROM Customers
    WHERE CustomerID = @customerid
    FOR XML AUTO
  </sql:query>
</root>

```

Поместите этот файл в виртуальный каталог базы данных Northwind (см. врезку «Настройка XML-доступа в SQL Server 2000» для детального ознакомления с конфигурированием IIS для работы с SQL Server и для получения подробной информации, необходимой для этой статьи).

SendNWXML выглядит подобно GetNWCcustomers, за исключением того, что ожидает передачи DiffGram, загружает DiffGram в объект MSXML2.DOMDocument и пересылает этот объект на Web-сервер, который, в свою очередь, передает его через SQLXML на SQL Server 2000 для обработки.

```

lparameters tcDiffGram
local loDOM as MSXML2.DOMDocument, ;
    loXML as MSXML2.XMLHTTP

loDOM = createobject('MSXML2.DOMDocument')
loDOM.async = .F.
loDOM.loadXML(tcDiffGram)
loXML = createobject('MSXML2.XMLHTTP')
loXML.open('POST', 'http://localhost/northwind/', .F.)
loXML.setRequestHeader('Content-type', 'text/xml')
loXML.send(loDOM)

```

Чтобы посмотреть, как это работает, запустите XMLExample.prg. Вы увидите единственную запись (заказчик ALFKI) в окне просмотра. Измените значение в некотором поле, затем закройте это окно и запустите программу PRG снова. Вы увидите, что ваше изменение будет записано в БД на сервере.

Заключение

Хотя основной класс CursorAdapter обеспечивает последовательный интерфейс для доступа к удаленным данным, независимо от того, используете ли вы ODBC, ADO или XML, все же, в зависимости от выбранного механизма доступа к данным, существуют некоторые отличия в настройке CursorAdapter. Эти отличия основываются на требованиях, предъявляемых к самим механизмам доступа к данным.

В следующей статье я закончу рассмотрение CursorAdapter, создам некоторые классы данных многократного пользования и расскажу о том, как использовать CursorAdapter в отчетах.

*Дуг Хенниг — один из партнеров в канадской фирме Stonefield Systems Group, Inc. Он является автором набора инструментальных средств для FoxPro-разработчиков Stonefield Database Toolkit for Visual FoxPro, а также соавтором книг «What's New in Visual FoxPro 7.0» и «The Hacker's Guide to Visual FoxPro 7.0», вышедших в издательстве Hentzenwerke Publishing, и автором книги «The Visual FoxPro Data Dictionary», вышедшей в издательстве Pinnacle Publishing. Дуг являлся техническим редактором книг «The Hacker's Guide to Visual FoxPro 6.0» и «The Fundamentals», вышедших в издательстве Hentzenwerke Publishing. Дуг в качестве докладчика участвовал в работе всех конференций Microsoft FoxPro Developers Conference (DevCon), и, кроме того, он выступает перед группами пользователей и на конференциях разработчиков, проводимых по всей Северной Америке. Профессионализм Дуга подтверждается сертификатами Microsoft Most Valuable Professional (MVP) и Certified Professional (MCP).
Его адрес: www.stonefield.com, dhennig@stonefield.com*



Вся энчилада целиком

Нэнси Фолсом (Nancy Folsom)



В своей последней статье «Если бы вариант использования мог использовать вариант...» Нэнси Фолсом взглянула в разработку варианта использования (use case) для задачи импортирования данных в систему. Для того чтобы осуществить это, она представила какими должны быть требования заказчика. В сегодняшней статье она иллюстрирует структурированный подход для небольшого дополнения к существующей программе. Даже в такой обычной ситуации структурированный подход позволяет создать более мощный продукт, а также избежать проблем.

Мы можем углубиться в вопрос о том, что представляет собой структурированный подход, но это лишит удовольствия нашу беседу, тем более что существует около 10 тысяч книг, написанных на эту тему. Тем не менее, не стоит быть слишком легкомысленным, поскольку этот предмет настолько глубок, насколько мы хотим заниматься им. В этой статье я обращаюсь к проблемам разработки действующих проектов. Вот главные шаги процесса структурированной разработки, используемого мною. Во-первых, определяем, что требуется заказчику, кому это нужно и кто будет это использовать. Во-вторых, излагаем эти требования на формальном, но удобочитаемом языке, соединяющем точку зрения заказчика и вашу собственную. В-третьих, оцениваем возможность реализации этой задачи с учетом времени и бюджета, выделенного для этого. В-четвертых, продумываем подход для программирования этого решения. В-пятых, пишем программу и тестируем ее реализацию. В-шестых, развертываем ее для вашего заказчика. В-седьмых, возвращаемся назад и... добавляем освежающий напиток по вашему выбору.

Прежде всего, что же им нужно?

Однажды я назвала эту стадию «требуется оценка». Независимо от того, как ее называете вы, это неизбежный первый шаг. Вы не можете составлять программу для того, что не можете определить. И хотя это необходимый шаг, похоже, неожиданно большое число проектов пропускают его или, по крайней мере, не уделяют ему должного внимания. Очень просто решить, что вы знаете что имеет в виду заказчик и то, что он дает вам возможность поупражняться в умении читать мысли, выглядит заманчивым. Думаю, однако, что опасность принятия неправиль-

ных предположений обратно пропорциональна трудности решаемой задачи, хотя, возможно, могут быть и отклонения от этого правила. Другими словами, когда два человека обсуждают что-либо, очень просто для каждого решить, что он использует одинаковые слова для обозначения одной и той же вещи.

Недавно я добавила утилиту для резервного копирования в приложение клиента. Путь, по которому пошло ее обсуждение, выглядел следующим образом: «Резервное копирование является реальной проблемой. Существует ли какой-либо способ сделать его проще?». Это стандартный тип дискуссии в проектах технической поддержки. «Мне необходимо сделать «нечто», можете ли вы это написать?». Вопросы, заданные мною в ответ, приведены ниже:

1. «Как вы выполняете резервное копирование в настоящее время?»
2. «Кто выполняет резервное копирование в настоящее время? Кто будет выполнять его после создания новой версии?»
3. «Что конкретно не устраивает вас в текущем процессе?»
4. «Что вам нравится в текущем процессе?»
5. «Что вы понимаете под словом «проще»?»
6. «Что еще, не являющееся жизненно важным, было бы хорошо добавить или удалить из текущего процесса?»

Возможно, эти вопросы являются очевидными. Я считаю, что поскольку программисты являются специалистами, разрешающими проблемы, мы склонны переходить к решению задачи до осознания самой проблемы. Хуже того, мы часто работаем, понимая проблему с нашей точки зрения. И то, что, возможно, не является проблемой для нас, может оказаться важным для заказчика.

Вопросы 2 и 4 являются менее очевидными. Зачем задавать вопросы, не относящиеся явно к сформулированной проблеме? Проще говоря, зачем кодировать больше, чем нужно или чем просит заказчик? В описываемом мною случае, заказчик просит о том, чтобы не только он один отвечал за выполнение резервного копирования, а чтобы некоторые другие люди были способны выполнять резервное копирование, а это влияет на разработку. Вопрос 4 помогает определить, какие функции теку-

щей системы должны присутствовать в следующей версии, а также дает мне нечто, служащее примером того, что любит заказчик.

Вопрос 3 относится к практически важным деталям. Что конкретно не устраивает его или что приложение должно делать, но не делает. Какую бы проблему перед вами не ставили, вы должны просить ваших заказчиков быть конкретными.

Основываясь на этих вопросах, мой клиент и я неформально беседуем некоторое время, что позволяет обсуждению выйти за рамки всех тех проблем, которые, по нашему мнению, имеют отношение к резервному копированию. Из сделанных мною записей я выбрала ответы и систематизировала их.

1. «Как вы выполняете резервное копирование в настоящее время?»

1.1. Резервная копия данных за каждый день создается после закрытия магазина.

1.1.1. Все рабочие станции выключены, поэтому не могут быть заняты.

1.1.1.1. Если данные заняты, они не должны быть повреждены в ходе попытки скопировать их.

1.1.2. Так как это происходит после закрытия, кому-то придется сидеть в магазине до завершения процесса резервного копирования и здесь могут быть другие завершающие задачи.

1.2. Резервное копирование состоит из процесса копирования каталога приложения сервера на жесткий диск рабочей станции.

1.2.1. Рабочая станция – это компьютер, с которого выполняется резервное копирование, т. е. оно является локальным относительно сервера.

1.3. Копия данных с рабочей станции копируется затем на Zip-диск.

1.3.1. Теперь объем Zip-диска достаточен для хранения одной резервной копии, тогда как раньше он использовался для нескольких. Было бы хорошо иметь возможность сохранять более одной резервной копии на Zip-диске.

1.4. Каждую ночь Zip-диск помещается в безопасное место хранения.

1.4.1. Сервер и рабочая станция никогда не оставляются из-за возможных выбросов напряжения. Поэтому новая система должна принимать это во внимание.

2. «Кто выполняет резервное копирование в настоящее время? Кто будет выполнять его после создания новой версии?»

2.1. В настоящее время эту работу выполняют владельцы данных по запросу супервизоров. Когда и те и другие отсутствуют в офисе, резервное копирование, возможно, не выполняется или зачастую выполняется некорректно.

2.2. Любая назначенная персона может запустить процесс резервного копирования, можно назвать их администраторами. Они не должны беспокоиться о подготовке аппаратного обеспечения (отсутствии накопителей на магнитных лентах) или принятии любых решений о присвоении имен файлов, назначении путей или чего-либо подобного.

3. «Что конкретно не устраивает вас в текущем процессе?»

3.1. Резервное копирование должно выполняться с сервера для того, чтобы пользователи с административны-

ми правами не находились в офисе супервизора.

3.2. Этот процесс занимает много времени в конце рабочего дня, поэтому люди не уделяют ему должного внимания, поскольку они хотят домой или думают о событиях прошедшего дня.

3.3. Он заставляет людей тратить свое личное время после работы.

3.4. Он не автоматизирован. Требуется принятия большого числа решений и избыточных знаний о процессе.

3.5. Возрастающий объем резервных копий означает, что емкость Zip-дисков начинает накладываться ограничение.

3.5.1. Есть ли здесь файлы, которые могут быть удалены? Да.

3.5.2. Требуется ли резервное копирование всех файлов каждую ночь? Не обязательно, но вполне вероятный, наихудший сценарий предполагает, что сервер выходит из строя и потребуются возможность поставить на его место резервную машину, провести восстановление предыдущей резервной копии, чтобы быть готовыми запустить сервер в течение часа.

3.5.3. Zip-диск не требуется. Перезаписываемые компакт-диски или другое решение является приемлемым при условии его надежности и простоты в использовании. Решено, что подходят переносные жесткие диски и они доступны по цене, поэтому для этой цели приобретается пара.

3.5.4. Резервные копии могут быть сжаты с помощью программы типа WinZip.

3.5.4.1. Программа WinZip является инструментом компании.

4. «Что вам нравится в текущем процессе?»

4.1. Он работает. Он выполняет эту работу.

4.2. Файлы без труда доступны на zip-дисках и, если они потребуются, легко получить их обратно.

4.3. Zip-диск компактен, поэтому владельцы могут пров�ерять и просматривать данные за день у себя дома.

4.4. Zip-диски недороги и надежны.

4.5. Zip-диск компактен, поэтому он не требует использования только на одной рабочей станции.

4.6. Избыточность является обнадеживающей.

5. «Что вы понимаете под словом «проще»?»

5.1. «Проще» означает, что резервное копирование выполняется автоматически, но не ценою меньшей доступности для восстановления (магнитная лента). Поэтому меньшее количество принимаемых решений = проще.

5.2. «Быстрее» — это тоже «проще». Весь процесс должен занимать самое большее 15 минут или около того.

6. «Что еще, не являющееся жизненно важным, было бы хорошо добавить или удалить из текущего процесса?»

6.1. Было бы хорошо, если бы расположение резервной копии не было жестко закодированным, но это не должно конфликтовать с задачей простоты использования.

6.2. Было бы хорошо, если бы резервное копирование можно было бы проводить по расписанию и запускать в фоновом режиме.

6.3. Было бы хорошо, если бы на носитель для резервного копирования помещалось более одной резервной копии, выполняемой ночью.

6.4. Было бы хорошо, если бы этот процесс копировал не только приложения и их данные, но и другие вещи тоже.

6.5. Было бы хорошо, если бы резервное копирование могло выполняться с любого компьютера, на котором установлено это приложение.

Таблица 1. Таблица разбирает наш пример варианта использования.

Действующий субъект	Система резервного копирования
1. Джефф выбирает опции резервного копирования из административной программы.	2. Резервное копирование предоставляет Джеффу возможность задавать для чего будет создаваться резервная копия, куда она будет копироваться, а также расположение и имя zip-файла. Резервная копия «помнит» выбор для каждой опции, так что супервизор может настраивать опции один раз и делать их доступными впоследствии для других пользователей. Однако эти настройки могут быть изменены.
3. Джефф либо принимает эти настройки как есть, щелкнув кнопку Backup, либо изменяет источник для резервного копирования, расположение копии или zip-файла.	4. Если Джефф изменяет расположение источника, процесс резервного копирования проверяет, существует ли этот источник. Если да, то сценарий возвращается к шагу 3. Если нет, то процесс извещает Джеффа, что источник не существует и выводит диалоговое окно для выбора другого источника. Сценарий продолжается с шага 8. 5. Если Джефф изменяет размещение копии, резервное копирование проверяет, существует ли это расположение. Если да, то резервное копирование спрашивает Джеффа, не хочет ли он перезаписать его и сценарий продолжается с шага 11. 6. Если нет, то резервное копирование извещает Джеффа о том, что источник не существует и выводит диалоговое окно для выбора источника, а затем сценарий продолжается с шага 8. 7. Если Джефф принимает эти изменения, резервное копирование создает резервную копию и отображает сообщение о состоянии, когда резервное копирование будет завершено и сценарий завершится.
8. Джефф выбирает каталог источника или отменяет этот выбор.	9. Если Джефф отменяет этот выбор, резервное копирование восстанавливает первоначальный источник. 10. Сценарий возвращается назад, к шагу 3.
11. Джефф выбирает Yes или No для перезаписывания существующего расположения для копирования.	12. Если Джефф выбирает No, то восстанавливается значение по умолчанию. 13. Сценарий продолжается с шага 3.

Разрабатываем варианты использования

Как вы помните из предыдущей статьи, варианты использования затрагивают взаимодействие «действующих субъектов» (actor) с «системой». Системой в данном случае, является планируемая нами новая программа резервного копирования. Действующими субъектами — администраторы и супервизоры, а также, потенциально, другие системы, такие как Windows scheduled task. Варианты использования рассказывают истории о том, как могут быть выполнены эти требования. Они преодолевают разрыв между заказчиками и дизайнерами, испытателями и разработчиками. Они могут служить основой для составления документации и планов проведения испытаний.

В этом примере не так много вариантов использования. Оба вида пользователей-людей практически одинаковы. В зависимости от того, является ли пользователь человеком или другой системой, могут быть определены, тем или иным способом, одинаковые типы вещей. Пора с чего-нибудь начать и для краткости я рассмотрю историю о том, как «Джефф» будет использовать программу резервного копирования. Исследование этого случая показано ниже.

Вариант использования 1: Джефф, пользователь с административными правами, выполняет ночное резервное копирование. Джефф выполняет резервное копирование дневной системы учета в розничной торговле.

Входные условия:

1. Допустим, что Джефф знает, как подключать и отключать съемный жесткий диск.
2. Допустим, что целевая рабочая станция для повторного резервного копирования отображается и доступна с сервера.
3. Джефф выбирает опции резервного копирования из приложения учета в розничной торговле, запускающего систему резервного копирования.

Выходные условия:

1. Получаются две резервные копии. Одна из них — это сжатый архив, хранящий копию каталога приложения с сервера на съемном жестком диске или другом сменном устройстве. Вторая резервная копия — это простая копия серверного каталога приложения на жесткий диск рабочей станции.
2. Предполагается, что ответственный работник переносит одну из копий в место хранения, расположенное вне офиса.

Этот пример варианта использования проиллюстрирован в таблице 1 и на рис. 1.

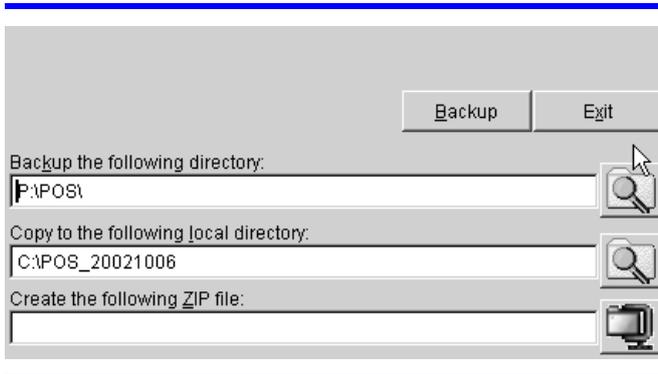


Рис. 1. Пример интерфейса задачи создания резервной копии.

Перепрыгиваем в модель

Большой загадкой для меня являлось то, как перейти от варианта использования к модели. Существует несколько различных подходов, не говоря уже о языках моделирования. Я использую язык моделирования UML, а также, как правило, диаграммы классов и диаграммы последовательности действий.

В прошлом году я изменила мой процесс и теперь начинаю с диаграмм последовательности действий вместо диаграмм классов, хотя диаграмма последовательности действий отражает только один путь через приложение. Возможно, она не настолько содержательна, как вариант использования. Однако поскольку я могу создавать классы по мере создания диаграммы последовательности действий, я трачу меньше времени, занимаясь диаграммой классов, и больше времени работаю над взаимодействиями. В любом случае, попытка встать на формальную позицию не всегда продуктивна, если только вы не являетесь системным архитектором. В моем случае я лучше буду думать о грамматике варианта использования. Я не могу поверить, что сказала это. Грамматика? Да, только существительные, глаголы и связи, вот все, что я имею в виду на самом деле.

После того, как мы разложили изучаемый случай на компоненты, я могу сделать первую диаграмму последовательности действий. Под «первой» я подразумеваю ту, что показывает простой путь, без каких-либо частичных усложнений. Это дает старт для работы над классами, что видно на рис. 2.

Это также намекает на потенциальные усложнения. Однако даже элементарная диаграмма последовательности действий дает некоторые классы, которые могут складываться в диаграмму классов, показывающую основные классы и даже некоторую иерархию наследования. Пример приведен на рис. 3.

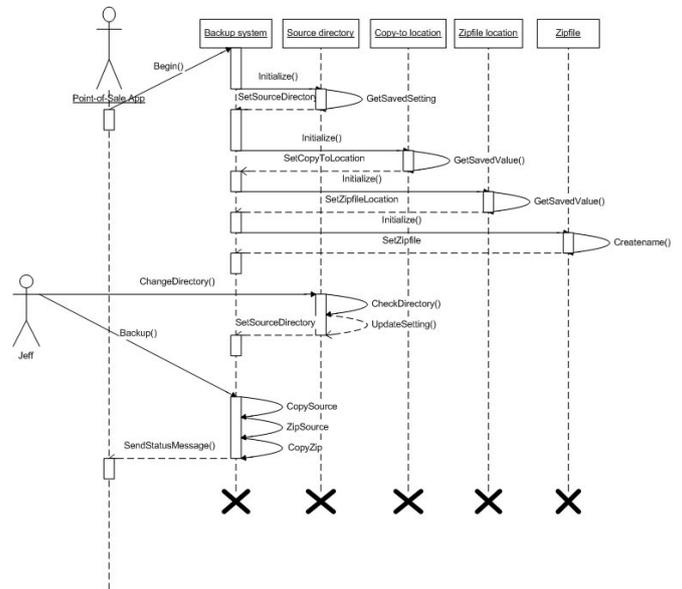


Рис. 2. Диаграмма последовательности действий задачи резервного копирования.

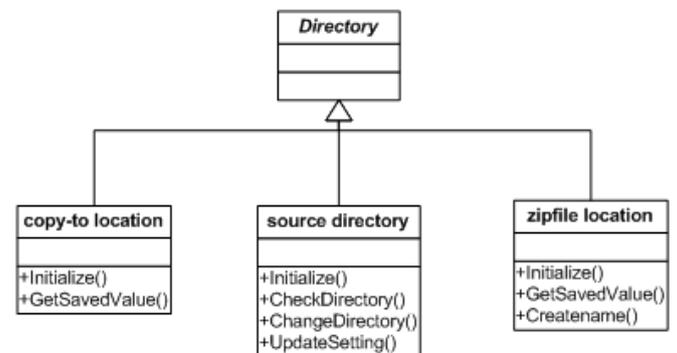


Рис. 3. Диаграмма классов для задачи резервного копирования.

Гарантирую, что существует путь для перехода от этой простой иерархии классов к работающей форме. С другой стороны, когда кто-либо уже переходит к программированию, эта часть работы является легкой. Относительно.



Градиентные и активные кнопки

Предраг Боснич (Predrag Bosnic)



Предраг Боснич продолжает исследовать способы украсить ваш пользовательский интерфейс, делая его более похожим на Web-страницу. Сегодняшняя тема: градиентные и активные кнопки.

В своей последней статье я писал об элементах интерфейса Gradient Form и Gradient Edit-Box. Градиентная заливка вдохновляла не только создателей Web-страниц, но и создателей элементов Web-интерфейса. В то же самое время появились кнопки с градиентной заливкой. Сегодня я создам элемент, симулирующий градиентные Web-кнопки, используя только элементы Visual FoxPro. Конечно, как обычно, существует множество возможных решений, но я буду использовать то, которое мне больше всего нравится и которое хорошо работает в VFP — это составной компонент (composite control). Для тех, кто не знаком с составными компонентами, скажем, что такой компонент получается из одного или большего числа элементов, содержащихся в некоторой разновидности контейнера. Для нашего случая я использую контейнер Visual FoxPro.

Основная идея использования градиентной кнопки заключается в том, чтобы сделать внешний вид формы более интересной (привлекающей пользователя щелкнуть по этой кнопке) и чтобы улучшить графический дизайн в целом.

Анализ

Командная кнопка в Visual FoxPro имеет свойство Picture, но, к сожалению, рисунок не заполняет все пространство кнопки (свойства stretch (растянуть) не существует). Надпись может быть показана вместе с рисунком, но она всегда располагается в нижней части кнопки. Все, чего хочу я, — это использовать рисунок в качестве фона, заполняющего все пространство кнопки (кнопка может иметь любой размер), и центрировать надпись по вертикали и горизонтали. Конечно, моя кнопка должна иметь все свойства и методы обычной кнопки Visual FoxPro. После всего сказанного становится очевидным, что я должен использовать контейнер, изображение, надпись и саму кнопку. Рис. 1 иллюстрирует эту основную идею.

Также я хочу, чтобы моя кнопка имела необязательное свойство Move type, которое может прини-



Рис. 1. Составной компонент со всеми его элементами.

мать значения 1, 2 и 3. Когда свойство имеет первое значение и я щелкаю кнопку, кнопка остается неизменной, но надпись на ней меняет свой цвет. При втором значении кнопка поддерживает стандартное поведение кнопок — когда я щелкаю

кнопку, она перемещается вправо и вниз на значение смещения, как если бы кнопка утапливалась вниз. Третий случай производит эффект приближения — когда я щелкаю кнопку, она вырастает на определенное значение смещения.

И, в заключение, я хочу, чтобы возникал визуальный эффект, когда мышь находится над кнопкой. Стандартный указатель мыши будет изменяться на указатель в виде пальца, когда кнопка активирована, и на указатель в виде знака «стоп», когда кнопка отключена.

Решение

Как обычно, моим первым шагом является создание библиотеки класса GradientButton.vcx для хранения определения моего класса. Следующий шаг — создание класса, основанного на классе container, также называемого GradientButton. Я добавлю надпись, изображение и кнопку в этот контейнер.

Вот свойства элемента image:

```
Stretch = 2 (Stretch)
```

Вот свойства элемента label:

```
Alignment = 2 (Center)
BackStyle = 0 (Transparent)
MouseIcon = 'h_point.cur'
MousePointer = 99 (Custom)
```

Вот свойства commandbutton:

```
Style = 1 (Invisible)
MouseIcon = 'h_point.cur'
MousePointer = 99 (Custom)
```

Контейнер имеет следующие определенные пользователем свойства:

```
nButtonType: 0 = Gradient button, 1 = Hover button
nMoveType:    0 = фиксированная, 1 = стандартное движение,
              2 = движение с расширением (zoom)
nOffSet:      величина перемещения кнопки (значение в пикселах)
```

Метод Init контейнера содержит следующий код:

```
DODEFAULT()
this.Resize()
this._commandbutton1.ZOrder(0)
```

Как видите, я добился того, чтобы элемент commandbutton был сверху и получал все действия мыши через zOrder. Метод Resize будет подгонять все размеры в зависимости от размера контейнера:

```
DoDefault()
WITH this._commandbutton1
  .Width = this.Width - 6
  .Left = 3
  *
  .Height = this.Height - 6
  .Top = 3
ENDWITH
*
WITH this._Image1
  .Width = this.Width
  .Height = this.Height
ENDWITH
*
WITH this._label1
  .left = 0
  .Top = (this.Height - .Height)/2 + 1
  .width = this.width
ENDWITH
this.Refresh()
```

Метод MouseDown имеет следующий код:

```
DO case
CASE this.nMoveType = 1
  this.left = this.left + this.nOffSet
  this.top = this.top + this.nOffSet

CASE this.nMoveType = 2
  this.left = this.left - 1*this.nOffSet
  this.top = this.top - 1*this.nOffSet
  this.Width = this.Width + 2*this.nOffSet
  this.Height = this.Height + 2*this.nOffSet
  this._Label1.FontBold = .t.
ENDCASE
this._label1.ForeColor = RGB(255,255,255)
this._commandbutton1.GotFocus()
```

Метод MouseUp содержит такой же код, но с другим знаком перед this.nOffSet.

Метод MouseEnter устанавливает свойство MouseIcon:

```
DoDefault()
IF this.Enabled = .t.
  this.MouseIcon = 'c:\program files\microsoft ;
  visual foxpro 7\graphics\cursors\h_point.cur'
  this._commandbutton1.MouseIcon = 'c:\program ;
  files\microsoft visual foxpro 7\graphics\
  cursors\h_point.cur'
ELSE
  this.MouseIcon = 'nodrop01.cur'
  this._commandbutton1.MouseIcon = 'c:\program ;
  files\microsoft visual foxpro 7\graphics\
  cursors\nodrop01.cur'
ENDIF
```

И, в заключение, метод Refresh содержит следующий код:



Рис. 2. Форма с градиентными кнопками.

```
DoDefault()
this._commandbutton1.ZOrder(0)
```

Методы событий элемента commandbutton — Click, GotFocus, LostFocus, MouseDown, MouseUp, MouseMove, MiddleClick, MouseEnter, MouseLeave и KeyPress — все содержат похожую строчку кода: “this.parent.Click()” в методе Click, “this.Parent.GotFocus()” в методе GotFocus и так далее. Это делает элемент commandbutton прозрачным для действий мыши. Когда вы щелкаете кнопку, срабатывает метод Click контейнера и так далее. Еще раз, когда вы хотите, чтобы кнопка выполняла какое-то действие, просто поместите нужный код в метод Click контейнера. Эта функция очень полезна в процессе разработки — она избавляет меня от необходимости перехода ниже структуры контейнера для достижения методов и свойств командной кнопки.

Эта прозрачность является прекрасным примером совместимости на уровне элементов. Командная кнопка и контейнер имеют много одинаковых методов. Смотрите рис. 2, чтобы увидеть форму с примерами законченных GradientButtons.

HoverButton

Подобно GradientButton, HoverButton имеет такие же характеристики, но когда указатель мыши находится над кнопкой ведет себя по-другому. Цвет заливки активной кнопки (hover button) обычно является одним из основных темных цветов, таких как темно-синий или темно-красный, и, когда мышь находится над кнопкой, появляется растровое изображение, являющееся по определению градиентным растровым изображением. Это способ для получе-

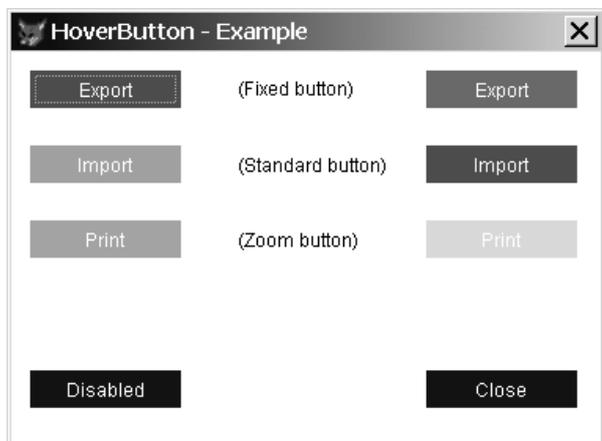


Рис. 3. Форма с активной кнопкой.

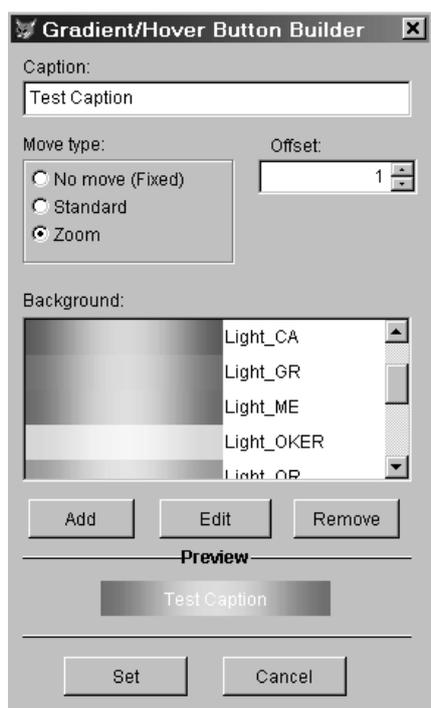


Рис. 4. Основная форма построителя.

ния эффекта «мышь над кнопкой». Когда мышь покидает кнопку, растровое изображение становится невидимым, и цвет заливки становится прежним (см. рис. 3).

Для того, чтобы изменить GradientButton на HoverButton, все, что нужно сделать, это установить следующее свойство:

```
nButtonType = 1 (Hover button)
```

Конструктор

Оба эти элемента используют один и тот же конструктор, помогающий мне устанавливать свойства и выбирать градиентный рисунок (см. рис. 4). Чтобы создать градиентный рисунок, я использую тот же метод, что был описан мною ранее для градиентных форм. Предлагаемый размер для градиентного рисунка, используемого мною для элемента GradientButton, равен 42 x 24 пикселя.

Усовершенствования

Вы, возможно, уже заметили очень большое сходство между этими двумя кнопками и возможность поместить их в один класс. Это выглядит логическим шагом, но я советую вам не делать этого, если вы хотите использовать только один тип кнопок в вашем приложении. Излишняя перегруженность должна быть устранена.

Заключение

Используемые вместе с Gradient Form и Gradient EditBox из моей предыдущей статьи, эти кнопки позволяют создать пользовательский интерфейс, значительно отличающийся от того, который я обычно использую для приложений Windows, и очень похожий на тот, что используется для Web-страниц. Грань между формами Windows и Web-страницами становится все менее и менее заметной, и однажды мы и вовсе не заметим ее.

Эти две кнопки являются «легкими», но помните, что для одной кнопки компьютер (Fox) управляет четырьмя или пятью элементами. Не забывайте одного из основных принципов: «Скорости обработки и памяти компьютера никогда не бывает много».

Это чистое решение Visual FoxPro 7 демонстрирует, возможно, простейший путь для создания привлекательных визуальных эффектов и улучшения вашего пользовательского интерфейса. Это то, что я называю «мощью Visual FoxPro». Все мои примеры кода находятся на дискете, и вы можете использовать их в ваших приложениях. В следующий раз я покажу вам, как создавать элемент ComboTree. (Чтобы примеры сразу заработали, соответствующий архив можно раскрыть в каталог Program Files\microsoft visual foxpro 8\wizards\GradColor — прим. ред.).

Предраг Боснич начал IT карьеру в 1979 году с UNIVAC 1100, Fortran и Mapper. В течение 20 лет он постоянно живет в мире персональных компьютеров, dBase, Clipper и Fox, изредка возвращаясь домой, в Лондон, где работает ведущим разработчиком в компании Westwood Forster Ltd. и прodelывает варварские штуки с Visual FoxPro и SQL Server.

С дерева вы можете обозревать весь лес!

Энди Крамек и Марсиа Акинс (Andy Kramek and Marcia Akins)



Разработка простого пользовательского интерфейса для отображения сложной связанной информации — не всегда простая задача, особенно, если вы столкнулись с ошибкой в базовом классе `pageframe` Visual FoxPro! В этой статье Энди Крамек и Марсиа Акинс показывают, как можно использовать стандартный элемент `Treeview` (управляемый с помощью данных для минимизации предварительной загрузки и, следовательно, оптимизации производительности) в качестве альтернативного способа отображения сложных иерархических данных.

Энди: Думаю, мы столкнулись с новой (по крайней мере, для меня) ошибкой в базовом классе `pageframe` Visual FoxPro.

Марсиа: Какого рода ошибкой?

Энди: Стэнли Барнетт (Stanley Barnett) поднял этот вопрос на форуме MS Developer Apps компании CompuServe (<http://forums.compuserve.com/vlforums/default.asp?SRV=MSDevApps>). Он обнаружил, что когда динамически добавляешь страницы в страничный блок (`pageframe`), то добавление новой строки вкладок приводит к «сползанию» страницы и стиранию элементов интерфейса (см. рис. 1).

Марсиа: Ох! Это ужасно. Я полагаю, что показанные тобой поля в правильной позиции находятся? То есть хотя страничный блок становится выше, его свойство `height` не изменяется, поэтому ты не можешь обнаружить этот факт.

Энди: Да, это ошибка! Поскольку страница не имеет собственного свойства `Top`, единственным способом вычислить, где должен размещаться элемент, является использование формулы, вроде приведенной ниже, для определения верхнего края видимой части страницы:

```
VisibleTop = PageFrame.Top + (PageFrame.Height ;
- PageFrame.PageHeight)
```

Марсиа: Извини, но эта формула не будет работать. Тебе необходимо использовать `ObjToClient()` для нахождения верхнего края страницы. Даже если страница не имеет своего собственного свойства `Top`, `ObjToClient()` даст тебе положение верхнего края этой страницы относительно самой формы. Необходимая тебе формула выглядит так:

```
VisibleTop = (PageFrame.Height - PageFrame.PageHeight) ;
- OBJTOCLIENT(Page, 1)
```

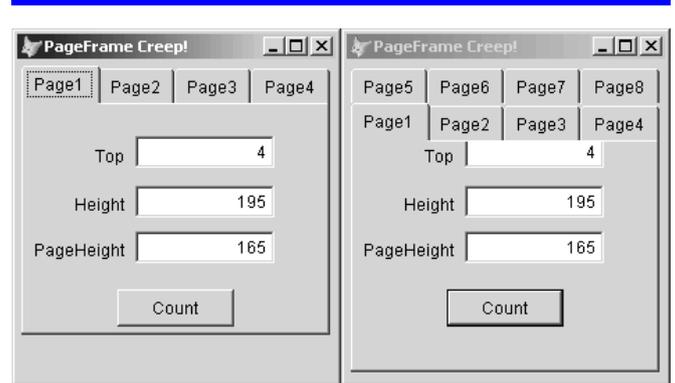


Рис. 1. Сползание элементов страничного блока.

Энди: Да, пожалуй. Хотя даже это нам не поможет, поскольку когда страничный блок вырастает, чтобы вместить новую строку вкладок, его свойства не изменяются. Несмотря на то, что эта формула прекрасно работает, как бы много строк вкладок страничного блока не инициализировалось, она дает сбой, как только еще одна строка добавляется динамически. На рис. 1 значение, вычисленное для верхнего края видимой страницы (для единственной строки вкладок) было +34, но для второй строки -29!

Марсиа: Зачем Стэнли все равно пытался проделать это? Мне кажется, что даже если существует какой-то путь для решения этой проблемы, интерфейс все равно будет довольно неуклюжим. Неудобно перемещаться более чем по трем или четырем страницам.

Энди: Проблема заключалась в том, что он пытался представить неопределенное число деталей, основываясь на иерархическом выборе.

Марсиа: Что ты имеешь в виду? Не понимаю, о чем ты говоришь. Ты что, опять наслушался политиков?

Энди: Давай используем конкретный пример. Рассмотрим проблему отображения почтовых индексов (ZIP-кодов). Нам нужно выбрать штат, затем округ, город и, в заключение, список всех почтовых индексов, относящихся к этому городу. Идея заключается в том, чтобы добавлять одну страницу с информа-

цией для каждого города, когда пользователь выбирает определенную комбинацию штата и округа.

Марсиа: Но в этом случае использовать одну страницу для каждого города было бы глупо. Это подходящая работа для элемента интерфейса Treeview. Он разработан, чтобы отображать данные в иерархическом формате и позволяет углубляться в данные до определенного элемента. Вы начинаете со штата, раскрываете штат, чтобы показать его округа, затем раскрываете округ, чтобы показать города. Когда вы добираетесь до города, вы можете использовать это для получения требующейся вам информации.

Энди: Проблема заключается в том, что это работает слишком медленно. У меня есть пример таблицы почтовых индексов, содержащий около 50 тысяч строк. Потребуется вечность, чтобы загрузить все это в Treeview.

Марсиа: Возможно, но в США только 50 штатов! Поэтому все, что нам нужно, это населить первый уровень (округа) до тех пор, пока не будет выбран штат, и нам, уж конечно, пока не нужны все города.

Энди: Для меня это довольно сложно.

Марсиа: Не думаю. Поступая так, ты структурируешь данные должным образом. Все, что нам нужно, это одна таблица для каждого уровня иерархии, и они должны быть связаны с помощью соответствующих ключей. Для этого примера нам потребуется четыре таблицы: для штатов (states), округов (counties), городов (cities) и почтовых кодов (zipcodes). Хотя, в случае нужды, ты можешь добиться того же результата, используя единственную таблицу, с помощью создания курсоров «на лету».

Энди: У меня уже есть данные, структурированные подобным образом (см. рис. 2).

Марсиа: Хорошо. Достаточно просто настроить Treeview на использование этих данных. Но то, что нам нужно на самом деле сделать теперь, это создать подкласс элемента Treeview, так чтобы мы могли управлять им с помощью данных и дать ему возможность заполняться самому из любого набора связанных данных.

Энди: Хорошо. Я готов к работе. С чего мы начнем?

Марсиа: Отлично. Первая вещь, которую мы сделаем, это создадим подкласс самого элемента Treeview, так чтобы мы могли добавлять к нему некоторые индивидуальные свойства и методы.

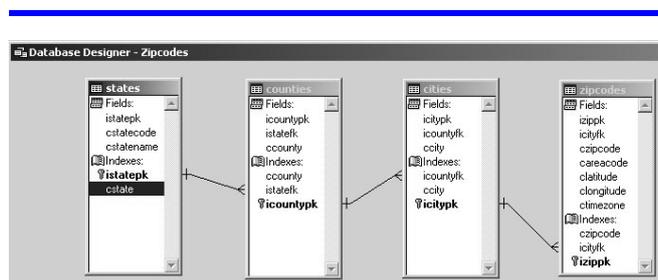


Рис. 2. Структура данных для отображения почтовых кодов.

Энди: Прекрасно. Мы создадим наш новый подкласс элемента Treeview (основанный на базовом классе OLEContainer), а затем используем его как основу для рабочего элемента. Итак, начнем с класса acxTVBase и затем создадим его подкласс acxTreeView (на дискете есть пример, содержащий все необходимые классы).

Марсиа: Этот подход на самом деле очень важно использовать, когда работаешь с элементами управления ActiveX. Проблема заключается в том, что когда появляется новая версия элемента, вы не можете вставить ее в существующий контейнер и поэтому всегда должны создавать новый контейнер. Если вы просто добавите ваши индивидуальные свойства в исходный подкласс этого элемента, вы не сможете затем обновить этот элемент без полной его потери.

Энди: Другой известный мне подход заключается в использовании простого класса container Visual FoxPro для ваших индивидуальных свойств и методов и добавления элемента управления ActiveX во время работы, используя его свойство VersionIndependentProgID. Преимущество этого подхода заключается в том, что он не основывается на определенной версии элемента ActiveX, установленного на компьютере пользователя.

Марсиа: Но мне не нравится этот подход. Он рискован, потому что может скрыть тот факт, что на компьютере установлена некорректная версия элемента — результатом, возможно, явится отказ приложения. При использовании OLEContainer можно быть уверенным, что если определенная версия недоступна, то мы получим ошибку инициализации при запуске (которую легко идентифицируем) скорее, чем некоторую непрогнозируемую ошибку в ходе работы. К тому же она может оказаться трудной для отслеживания, особенно если описывается чем-то двусмысленным, например: «operator/operand mismatch!» (несоответствие оператора/операнда).

Энди: Это обоснованная точка зрения. Мы будем придерживаться подхода, основанного на OLEContainer.

Марсиа: Ключом к работе с Treeview является его коллекция Nodes. Узел в данном контексте — это просто элемент данных, занимающий определенную позицию в этой иерархии. Вовлечены только две основные операции: добавление узлов (выполняемое в нашем классе с помощью метода AddNode()) и синхронизация данных при перемещении между ними (выполняемая с помощью метода SynchCursors()).

Чтобы действительно добавить узел, мы используем метод Add() коллекции элементов Nodes элемента Treeview следующим образом:

```
oTree.Nodes.Add( relative, relationship, key, ;
                text, image, selectedimage)
```

где Relative является либо индексом, либо ключом уже существующего узла.

Энди: Итак, чтобы добавить узел, мы должны иметь возможность определить его связь с уже существующим узлом. Как же тогда ты добавишь первый из них?

Марсиа: Не определяя связанный узел. Если ты не определяешь связь, новый узел просто добавляется как элемент высшего уровня. Дополнительная информация потребуется только при создании более глубокой части этой структуры. Тогда будет необходимо идентифицировать тот узел, к которому относится новый узел (через его ключ), и тип связи, использующий один из следующих численных констант:

- 1. Новый узел размещается после всех узлов того же уровня иерархии, что и узел, упоминаемый в аргументе Relative.
- 2. (Значение по умолчанию). Новый узел размещается после узла Relative в качестве узла-брата.
- 3. Новый узел располагается перед узлом Relative в качестве узла-брата.
- 4. Новый узел добавляется как дочерний к коллекции Relative Nodes.

Энди: Итак, следующий вопрос заключается в том, как назначать ключи узлам при их добавлении? Только ли используя первичный ключ из таблицы?

Марсиа: Нет, это не будет работать. Так как мы определяем иерархию, узлы которой приходят из различных, но связанных таблиц, нам также необходимо идентифицировать таблицу, являющуюся источником для любого заданного узла. Итак, то, что мы должны сделать, это создать сцепленный «ключ

узла» (node key), состоящий из псевдонима таблицы, значений первичного и внешнего ключа, связывающего его с родителями. Нам также необходимо выделить определенную часть ключа (так, чтобы мы могли просто находить его), используя узнаваемый символ. Я люблю использовать звездочку, поскольку она не может применяться в именах файлов.

Энди: Подожди, я пытаюсь представить себе это. Итак, для узла city ключ может выглядеть примерно так:

```
cities*10020*34221
```

Ясно, что “10020” является первичным ключом в таблице cities, но как я могу знать, что “34221” указывает на таблицу counties? Не должны ли мы включить также это имя?

Марсиа: Нам нет необходимости делать это. Чуть раньше мы сказали, что Treeview является иерархической структурой. Это означает, что мы можем управлять при помощи данных процессом заполнения, определив массив, каждая строка которого определяет необходимую информацию для равнозначного уровня иерархии. Это означает, что каждая строка содержит псевдоним таблицы, используемой для заполнения этого уровня, имя поля первичного ключа в этой таблице, имя поля внешнего ключа в этой таблице и имя поля, содержащего текст, появляющийся на узле.

Энди: Понимаю. Итак, для наших данных о почтовых кодах мы должны явно создать массив, похожий на следующий:

Уровень	Col 1 (Alias)	Col 2 (PK)	Col 3 (FK)	Col 4 (Text)
1	states	istatepk		cStateName
2	counties	icountypk	istatefk	cCounty
3	cities	icitypk	icountyfk	cCity
4	zipcodes	izippk	icityfk	czipcode

Марсиа: Да. Все, что нам потребуется еще, это только специфический для экземпляра код. Теперь, когда у нас есть этот массив, все остальное может управляться из него. Наша форма нуждается в методе (называемом SetForm()), вызываемом из Init() для заполнения массива aLevels[] элемента Treeview.

Энди: Это деликатно подводит нас к списку индивидуальных свойств и методов, необходимых нашему подклассу Treeview. То, что у нас получилось, приведено в таблице 1.

Таблица 1. Индивидуальные свойства и методы нашего подкласса Treeview.

Свойство/метод	Назначение
aLevels[1,5]	Хранит имя курсора, имя поля первичного ключа, имя поля внешнего ключа и имя поля, используемого для описывающего узел текста для курсора, используемого для заполнения каждого уровня Treeview, а также флажок для переопределения поведения по умолчанию при использовании тега для поиска записи.
cmenu	Имя меню быстрого вызова, появляющегося при щелчке правой кнопкой мыши (если таковое требуется).
nfactorx	Множитель координаты x для преобразования пикселей в твипы (twip) (необходим для метода HitTest).
nfactory	Множитель координаты y для преобразования пикселей в твипы (twip) (необходим для метода HitTest).
addnodes()	Добавляет узлы для определенного курсора на определенном уровне для заданного родительского узла. Если ни один параметр не передается, все узлы на уровне 1 заполнены. В противном случае, заполнены все дочерние узлы для пересылаемого узла.
findrec()	Для заданного номера уровня (индекс в массиве aLevels) и значения ключа находит запись в псевдониме, связанном с этой строкой массива.
getchildnode()	Для заданного ключа узла и индекса возвращает ссылку на этот дочерний объект, если он существует, или, в противном случае, значение null. Если порядковый номер в индексе не пересылается, возвращает объектную ссылку на первый дочерний узел, если только это не концевой узел (leaf node).
getparentnode()	Для заданного ключа для узла возвращает объектную ссылку на родительский узел или null, если это корневой узел.
istag()	Для заданного имени тега и имени псевдонима возвращает значение True, если тег существует.
selfactors()	Устанавливает множители, требуемые для преобразования пикселей в твипы.
showmenu()	Шаблонный метод, вызываемый для отображения меню быстрого вызова и выполняющий соответствующее действие.
str2exp()	Для заданной строки и типа данных возвращает преобразованное значение этой строки в значение этого типа данных.
synhcursors()	Вызывается из NodeClick элемента Treeview, размещает RecordPointer на нужной записи каждого из курсоров, связанных с текущим выбранным узлом Treeview.

Марсиа: В этом классе достаточно много кода, но он в значительной мере самоочевиден (и хорошо прокомментирован), поэтому я думаю, что заинтересованные читатели будут в состоянии понять его, если они захотят в нем разобраться. На самом деле этот класс является полностью инкапсулированным и при использовании не требует никакого дополнительного кодирования, кроме настройки массива aLevels[].

Энди: Для настройки Treeview в этом примере необходим только специфический для экземпляра код:

```

*** Настройка свойств массива для Treeview
*** с последующей загрузкой узлов высшего уровня
WITH This.acxZipTree
    DIMENSION aLevels[ 4, 5 ]
    .aLevels[ 1, 1 ] = 'STATES'
    .aLevels[ 1, 2 ] = 'ISTATEPK'
    .aLevels[ 1, 4 ] = 'CSTATENAME'
    .aLevels[ 2, 1 ] = 'COUNTIES'

```

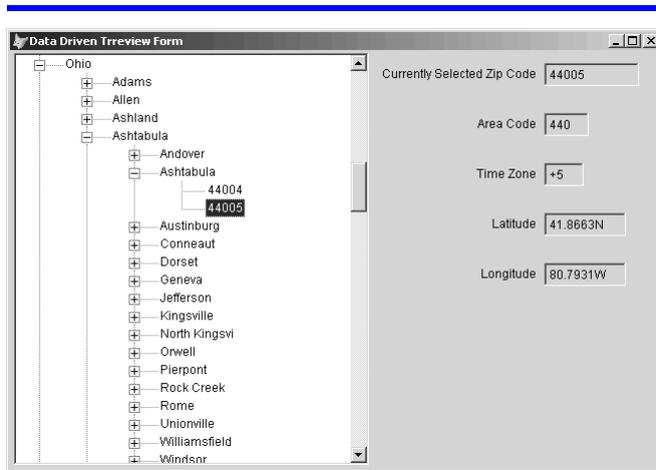


Рис. 3. Управляемый данными объект Treeview в действии (DO FORM frmziptree.scx).

```

.aLevels[ 2, 2 ] = 'ICOUNTYFK'
.aLevels[ 2, 3 ] = 'ISTATEFK'
.aLevels[ 2, 4 ] = 'CCOUNTY'
.aLevels[ 3, 1 ] = 'CITIES'
.aLevels[ 3, 2 ] = 'ICITYFK'
.aLevels[ 3, 3 ] = 'ICOUNTYFK'
.aLevels[ 3, 4 ] = 'CCITY'
.aLevels[ 4, 1 ] = 'ZIPCODES'
.aLevels[ 4, 2 ] = 'IZIPFK'
.aLevels[ 4, 3 ] = 'ICITYFK'
.aLevels[ 4, 4 ] = 'CZIPCODE'
*** Теперь загружаем уровень 1 узлов Treeview
.AddNodes()
*** Получаем форму, настроенную для первого использования
GO TOP IN States
Thisform.Refresh()
ENDWITH

```

Марсиа: Этот пример формы в действии показан на рис. 3. Обратите внимание, что у нас отображены связанные данные в таблицы details (почтовые коды) и что они автоматически обновляются всякий раз, когда щелкаешь новый узел. Фактически, мы можем отображать любые данные из любой таблицы, поскольку этот класс также поддерживает все эти таблицы синхронизированными. Не требуется никакого дополнительного кода — мне это нравится в любом классе!

Энди: Еще одна заключительная вещь, достойная быть отмеченной: эта методология действительно делает набор данных управляемым при помощи Treeview. Несмотря на то, что в таблице почтовых индексов почти 50 тысяч строк, эта форма загружается чрезвычайно быстро, поскольку все, что должно быть сделано в начале, — это заполнение 59 узлов «уровня 1». Общеизвестно, что возникает незначительная задержка при первом щелчке узла (зависящая от числа его дочерних узлов), но она едва заметна и является малой ценой за столь гибкий инструмент и интерфейс.



Глядите-ка! Это Outlook!

Уилл Хентцен (Whil Hentzen)



Надеемся, что статья в колонке «Kit Vox» прошлого месяца не слишком напугала вас. Сегодня Уилл Хентцен возьмет программу, написанную Марсией и Энди, и применит к ней некоторые из своих собственных советов и хитростей, придающих ей дополнительную функциональность.

Если только вы не скрывались в горах последнее время, то знаете, что я абсолютно не впечатлен способом, предоставляемым для экспорта данных пользователям программы Outlook. Функция Export в Outlook может свести с ума уже вследствие плохой разработки и недостаточного внимания к конечному пользователю. Или, может быть, она была намеренно ухудшена, чтобы затруднить перемещение ваших данных из Outlook (насколько я знаю, ваш электронный адрес продолжает оставаться вашим, а не компании Microsoft).

Как бы то ни было, конечный результат один — ужасно трудно экспортировать ту информацию, которая вам нужна. Поэтому наиболее важным для вас является перемещение данных вашей электронной почты из оригинального формата данных в формат, который может быть просто прочитан различными программными продуктами, выпущенными другими производителями. Поскольку даже у солидных клиентов некоторые консультанты внезапно берут и исчезают, нетрудно понять, что складывать все яйца в одну корзину — гиблое дело.

В прошлой моей статье вы видели, что я был успешен в критике... ой, ошибся, в одобрении моего доблестного дуэта Kit Vox за наступление на проблему объектной модели Outlook, чтобы иметь возможность использовать Automation для получения всех моих данных. Энди и Марсия применили несколько очень изощренных способов для достижения результата в элегантной форме. Получение исправленной объектной модели, прохождение дерева каталогов с помощью рекурсивных вызовов и поиск и сохранение нескольких прикрепленных файлов были проблемами, над которыми я ломал голову снова и снова.

Поэтому, когда они закончили свою программу, они передали ее мне для тестирования. У меня было множество данных для ее проверки — около 5-ти гигабайт старых PST-файлов. По результатам этих испытаний я открыл некоторые дополнительные требования. В этой статье я рассказываю о проделанной работе.

Меряем температуру

Первое изменение, выполненное мною, было косметическим. Мой текущий PST-файл имеет размеры примерно 300 Мб — возможно, около 9000 сообщений. Для его обработки требовалось время, поэтому я решил, что некоторый тип обратной связи может оказаться полезным, чтобы я знал, что программа продолжает работать, а не просто «висит», как это случается с Outlook. Это было просто — добавляем свойство к определению класса и затем вставляем окно ожидания (wait window) в метод ReadMsg:

```
* добавляем свойство counter
iCounter = 1
```

```
* отображаем счетчик в окне ожидания
wait window nowait transform(this.iCounter)
```

Где вы хотите разместить ваши файлы?

Необходимость второго изменения стала очевидной, как только я провел тестирование программы на моем текущем PST-файле. В каталоге, содержащем программу, было около 2 тысяч присоединенных файлов, от 17-байтных ATT0109.TXT до мегабайтных Zip-файлов, содержащих главы книги или рисунки детей в парке Great America, сделанные этим летом. (Что? У вас нет копии этого электронного письма? Только дайте мне знать об этом. У меня есть тысячи рисунков детей из каждой поездки в этот парк, которые, я уверен, восхитят вас. А я еще даже не дедушка. Пока.)

Когда я захотел очистить эту папку, то понял, что хорошей идеей может оказаться перенесение присоединенных файлов в подкаталог. (Фактически, Марсия была готова добавить это в программу, но Энди завопил: «Не делай этого!». Или, возможно, Марсия завопила на Энди. Я все время путаю.)

Итак, второе изменение создает второе свойство класса для связанного имени каталога и затем включает это свойство как часть функции Save.

```
* имя для подкаталога для присоединенных файлов
cDirAttach = "ATTACH"
```

```
* пересылаем присоединенные файлы в подкаталог
* curdir() returns "\SOMEDIR\"
* append "ATTACH\"
lCFileName = FULLPATH( CURDIR() + this.cDirAttach );
+ loAttachment.FileName
```

Обработка большого списка получателей

Третье изменение, сделанное мною, заключалось в использовании нескольких учетных записей электронной почты, управляемых единственным экземпляром Outlook. Другими словами, я использовал Outlook для получения почты для нескольких различных учетных записей, таких как `whil@hentzenwerke.com` и `webmaster@hentzenwerke.com`, а также для меня из других доменов (персональный электронный адрес, электронный адрес в домене заказчика и так далее). Я хочу отследить, на какой электронный адрес каждое отдельное электронное письмо было послано.

Однако, поскольку электронное письмо может быть послано более чем одному получателю, список Recipient может содержать много имен, из которых только одно является моим. Поэтому я проходил через коллекцию Recipients и сохранял полный список элементов в поле мемо.

Некоторые из вас могут быть удивлены, почему я сохранил коллекцию полностью, вместо того, чтобы просто найти электронный адрес, связанный со мной. Прежде всего, это может оказаться полезным для того, чтобы знать, кто еще получает электронную почту и, кроме того, я не хотел загружать программу знанием всех возможных электронных адресов, которые я когда-либо использовал или буду использовать. Что будет, если я добавлю другой псевдоним через год? Придется не забыть добавить этот псевдоним в программу, запуская ее для PST-файла за тот период времени. Проще захватить все, а затем, вспоминая историю электронной переписки, я могу загрузить дополнительный псевдоним, если это потребуется.

Эта задача оказалось несколько сложнее. К счастью, Марсиа уже собрала некоторый псевдокод, и я смог с меньшими усилиями доработать и использовать его. Прежде всего, я добавил мемо-поле omRecip в SAVEMAIL.DBF, в которое я буду помещать коллекцию получателей. Затем я добавил приведенный ниже код:

* также хранит список получателей
* необходимо добавить поле мемо omRecip в SaveMail.DBF

```
lnCount = loItem.Recipients.Count
m.lcRecip = ''
FOR lnRecip = 1 TO lnCount
  loRecipient = loItem.Recipients[ lnRecip ]
  m.lcRecip = m.lcRecip + loRecipient.Name + ': ' :
  + loRecipient.Address + CHR( 13 ) + CHR( 10 )
ENDFOR
```

Этот код просто пробегает коллекцию Recipient, выбирая имя и электронный адрес и добавляя его к строке. После завершения целая строка вставляется в поле omRecip в SAVEMAIL вместе с остатком команды INSERT.

Кто послал мне электронное письмо?

Последнее изменение было наиболее хитрым. Я хотел знать, от кого получаю электронное письмо. У нас уже есть поле, называемое omSender, но во многих случаях это поле содержит только имя и фамилию отправителя, но не его электронный адрес. В соответствии с последними RFC (Requests for Comments) для электронной почты, электронное сообщение должно содержать действительный электронный адрес, но может не содержать истинное имя отправителя. Если электронное письмо содержит как истинное имя отправителя, так и его электронный адрес, эта программа извлечет его имя и вставит в omSender. Если истинное имя отсутствует, то на этом месте будет использоваться электронный адрес.

Для того чтобы компенсировать эту проблему, я использовал измененную версию программы из книги «MegaFox: 1002 Things You Wanted to Know About Extending Visual FoxPro», написанной Марсией, Энди и Риком Шуммером (Rick Schummer). Эта программа роется в адресной книге Outlook и помещает имя, фамилию, полное имя и электронный адрес в массив.

Затем, после «откапывания» истинного имени из электронного сообщения, я искал это имя в массиве адресной книги. Если находил, то помещал электронный адрес в новое поле, добавленное мною в SAVEMAIL.DBF. Вот эти шаги.

Во-первых, создаем массив свойств этого класса для хранения контакта:

```
* массив для хранения контактов
DIMENSION aContacts[1,1]
```

Во-вторых, создаем DBF, который будет хранить копию этих контактов. Зачем? С тем чтобы я мог просматривать эту таблицу в ходе тестирования, вместо того, чтобы возиться с «невидимым» массивом:

```
create dbf SaveCont (cna1 c(50), cna2 c(50), ;
  cna3 c(100), cemail1 c(80), cemail2 c(80), ;
  cemail3 c(80))
```

В-третьих, изменяем саму программу:

```
LOCAL loAddressBook AS Outlook.MAPIFolder, loContact ;
  AS Object, lnContactCount
*** Получаем ссылку на каталог contacts
* olFolderContacts is 10
loAddressBook = This.oNameSpace.GetDefaultFolder( 10 )
IF VARTYPE( loAddressBook ) = 'O'
  lnContactCount = 0
  *** Получаем информацию
  *** о каждом контакте в этом массиве
  FOR EACH loContact IN loAddressBook.Items
    WITH loContact
      *** убеждаемся, что мы получили
      *** только индивидуальные контакты
      *** и пропускаем любые списки распространения
      wait window nowait "Parsing Contacts: " + ;
        transform(m.lnContactCount) + " processed"
```

```

* olContact is 40
IF .Class = 40
  lnContactCount = lnContactCount + 1
  DIMENSION This.aContacts[ lnContactCount, 9 ]
  This.aContacts[ lnContactCount, 1 ] = .LastName
  This.aContacts[ lnContactCount, 2 ] = .FirstName
  This.aContacts[ lnContactCount, 3 ] = STRTRAN( ;
    .Email1DisplayName, '(E-mail)', '', -1, 1, 1 )
  This.aContacts[ lnContactCount, 4 ] = STRTRAN( ;
    .Email2DisplayName, '(E-mail 2)', '', -1, 1, 1 )
  This.aContacts[ lnContactCount, 5 ] = STRTRAN( ;
    .Email3DisplayName, '(E-mail 3)', '', -1, 1, 1 )
  This.aContacts[ lnContactCount, 6 ] = ;
    .Email1Address
  This.aContacts[ lnContactCount, 7 ] = ;
    .Email2Address
  This.aContacts[ lnContactCount, 8 ] = ;
    .Email3Address
  This.aContacts[ lnContactCount, 9 ] = ;
    upper(.FullName)
  m.lcNaf = .LastName
  m.lcNal = .FirstName
  m.lcNaf1 = .FullName
  m.lcemail1 = .Email1Address
  m.lcemail2 = .Email2Address
  m.lcemail3 = .Email3Address

  insert into SaveCont ;
    (cnaf, cnal, cnafl, cemail1, cemail2, cemail3) ;
values ;
  (m.lcnaf, m.lcnal, m.lcNaf1, m.lcemail1, ;
  m.lcemail2, m.lcemail3)
ENDIF
ENDWITH
ENDFOR
ASORT( This.aContacts )
ENDIF

```

В-четвертых, Init() заполняет эти контакты:

```

* также создаем список контактов
IF llRetVal
  This.GetContacts()
ENDIF

```

И, в заключение, в ReadMsg() ищем данные SenderName в массиве адресного списка и, если находим, сохраняем электронный адрес вместе с другими частями этого сообщения:

```

* ищем SenderName в адресном списке с тем,
* чтобы мы могли сохранить сам электронный адрес

*** Сначала смотрим, не является ли имя
*** отправителя на самом деле электронным адресом
IF '@' $ loItem.SenderName
  m.lcSenderEm = loItem.SenderName
ELSE
  lnRow = ASCAN( This.aContacts, loItem.SenderName, ;
    -1, -1, 3, 15 )
  IF lnRow > 0
    m.lcSenderEm = This.aContacts[ lnRow, 6 ]
  ELSE
    lnRow = ASCAN( This.aContacts, loItem.SenderName, ;
    -1, -1, 4, 15 )
    IF lnRow > 0
      m.lcSenderEm = This.aContacts[ lnRow, 7 ]
    ELSE
      lnRow = ASCAN( This.aContacts, loItem.SenderName, ;
    -1, -1, 5, 15 )
      IF lnRow > 0
        m.lcSenderEm = This.aContacts[ lnRow, 8 ]
      ELSE
        m.lcSenderEm = ''
      ENDIF
    ENDIF
  ENDIF
ENDIF
ENDIF

```

Фрагмент вложенного кода сканирует и пытается найти совпадения для каждого из трех полей электронного адреса, содержащихся в Outlook.

Кто послал мне электронное письмо (повтор)?

Это работает хорошо, если у вас выбрана опция "Put people I reply to in my address book" (помещать людей, которым я ответил, в мою адресную книгу). Но что будет, если некоторые люди, которые посылают вам электронные письма, не внесены в вашу адресную книгу? Давайте еще усилим нашу программу, чтобы обеспечить возможность получения электронных адресов отправителей, посылающих вам электронные письма, но отсутствующих в адресной книге. Когда я с Марсией обсуждал этот вопрос на форуме Universal Thread, Гаррет Фитцджеральд (Garrett Fitzgerald) прислал идею интересного обходного пути. В вашем коде отвечаем на текущее сообщение и получаем значение "To" из электронного сообщения. После того, как у вас есть адрес, вы можете пропустить ответ (другими словами, просто не посылать его). Марсия придумала код, выглядевший так:

```

*** Сначала смотрим, не является ли
*** имя отправителя на самом деле
*** электронным адресом
IF '@' $ loItem.SenderName
  m.lcSenderEm = loItem.SenderName

ELSE
  loReply = loItem.Reply()
  loRecip = loReply.Recipients[ 1 ]
  m.lcSenderEm = loRecip.Address
ENDIF

Это работало замечательно за исключением двух случаев. Первое исключение заключается в том, что элемент электронного письма, который вы обрабатываете, может не содержать значения в поле "To". Это может быть в случае, если вы работаете над черновиком или если сообщение имеет данные только в полях СС и ВСС. Поэтому я добавил оболочку, проверяющую наличие поля "To", следующим образом:

*** Сначала смотрим, не является ли
*** имя отправителя на самом деле
*** электронным адресом
if '@' $ loItem.SenderName
  m.lcSenderEm2 = loItem.SenderName
else

if empty(loItem.To)
  * don't reply - this is just a draft
  m.lcSenderEm2 = "Draft"

else
  loReply = loItem.Reply()
  loRecip = loReply.Recipients[ 1 ]
  m.lcSenderEm2 = IIF( NOT EMPTY( loRecip.Address ), ;
    loRecip.Address, loRecip.Name )
endif
endif

```

Второе исключение — это электронные письма, для которых значение обратного адреса (reply-to) определено явно (с помощью их клиента электронной почты). Для них нужный электронный адрес также не захватывался должным образом.

И вот в чем тут дело. Когда вы настраиваете учетную запись в нескольких клиентах электронной почты, вы можете определить обратный адрес, отличный от электронного адреса, используемого отправителем. Другими словами, вы можете создать учетную запись с bob@yourcompany.com в качестве электронного адреса и, когда вы посылаете электронное письмо людям, они видят, что оно пришло с этого электронного адреса. Однако, вы можете также определить для обратного адреса значение robert@someothercompany.com, так что когда они отвечают на ваше сообщение от bob, поле "To" в их электронном письме будет содержать robert.

Я запустил этот код для примерно 120 тысяч электронных сообщений, полученных мною с 1998 года, и только около 70 писем содержали имена, которые не были найдены. Это определяется тем, что в большинстве из этих писем поле обратного адреса содержит то же самое значение, которое обнаружится, если я запущу Outlook, найду это сообщение, щелкну кнопку Reply и посмотрю на фактическое имя, введенное в поле "To". Однако, не во всех из них. Это странно!

Другие отклонения

Вы, возможно, не будете удивлены, если я скажу вам, что работая с набором данных такого размера, я заметил также и другие отклонения. При обработке этих 120 тысяч сообщений программа не справилась еще в трех других ситуациях, общим числом 25 раз.

Первая проблема заключается в том, что функция Reply() иногда выдавала ошибку «Unable to complete this operation» (невозможно завершить данную операцию). После просмотра конкретных сообщений, вызвавших эту проблему, я не смог найти какой-либо простой зацепки. Пришлось перейти к этому сообщению в Outlook и вручную ответить на него.

Вторая проблема состоит в том, что метод SaveAttachments() также иногда выдает ошибку и опять с сообщением «Unable to complete this operation». На этот раз я смог определить в чем дело.

Эти сбои происходили вследствие ошибки самой программы Outlook, вызванной тем, что URL преобразуется в метафайл типа изображения и некорректно присоединяется к сообщению в качестве прикрепленного файла. Довольно интересно, что хотя сами эти сообщения могут быть просмотрены в Outlook,

попытка просмотреть присоединенные файлы также приводила к «зависанию» Outlook.

Третья проблема, с которой я сталкивался время от времени, было сообщение «array index out of bounds» (индекс массива вне границ) на попытку присваивания значения loRecip в следующей строке кода:

```
loRecip = loReply.Recipients[ 1 ]
```

Когда я останавливал программу и «играл» с объектом loReply, то свойство Recipients просто не существовало. Это было непонятно, поскольку свойство To объекта loItem (из которого создавался объект loReply), несомненно, существовало. Еще один случай воскликнуть «Это странно!».

Если вы используете эту программу постоянно, тогда вам захочется включить некоторый механизм перехвата ошибок в процесс сохранения присоединенных файлов. Хотя нажать 25 раз кнопку Ignore для 120 тысяч сообщений не слишком трудная задача, этот вариант не подходит, если вы хотите, чтобы программа работала автоматически или если у вас есть множество сообщений для экспорта.

Если у вас есть два сообщения, имеющие присоединенные файлы с одинаковыми именами, то поскольку присоединенные файлы сохраняются в одном и том же каталоге, последний будет сохранен поверх предыдущего. Вы можете захотеть включить некоторый код, добавляющий информацию о номере версии или дате, чтобы различать разные копии файла SETUP.EXE или FoxTalkArticleForWhil.Zip.

Заключение

Я использую Outlook так долго, сколько помню себя (даже больше). Ее привычный пользовательский интерфейс оптимален для работы и, поскольку я знаком с несколькими другими почтовыми клиентами, я не нашел для себя ничего лучшего. Тем не менее, содержание ваших архивированных данных в специализированном формате просто не лучшее бизнес-решение. Их преобразование в какой-либо открытый формат, например DBF (или, скажем, XML), будет очень мудрым шагом. В будущем я намерен обсудить, что же делать со 120 тысячами архивированными сообщениями электронной почты.



Стань плодотворнее с VFP, часть 1

Ричард А. Шуммер (Richard A. Shummer)

WIN



Visual FoxPro является очень производительным инструментом разработчика. В этом месяце Рик Шуммер начинает короткую серию статей, содержащих различные советы и хитрости, призванные помочь вам повысить производительность в Visual FoxPro.

Rapid Application Development (RAD) – модное словечко, пережиток 90-х. Настолько ли вы производительны с VFP, как можете или хотите быть? Как другие разработчики используют этот лучший в мире инструмент разработки приложений баз данных, чтобы быстрее поставлять приложения на рынок? Есть ли такие хитрости, которые помогут вам сэкономить 10 минут в день или 1 час в неделю? Эта серия статей представит вам столько советов и идей по повышению производительности, сколько поместится на выделенных мне страницах.

Как утверждает старая поговорка, в Visual FoxPro всегда есть три пути для достижения цели. Обычно мы знаем только один из них, но существует еще два – быстрее и лучше. Иногда мы даже не знаем, что можем выполнить определенные вещи в VFP. Я постоянно удивляюсь, даже после моего восьмилетнего опыта работы с Visual FoxPro, сколь многому я научился, просто заглядывая через плечи других, когда они занимались разработкой с помощью этого продукта.

Object Browser

Object Browser является новым для Visual FoxPro 7 инструментом. Он открывает общие и защищенные интерфейсы библиотек объектов COM и элементов управления ActiveX. В этих библиотеках находится обилие информации о свойствах, событиях, методах, значениях констант и классах, доступных разработчикам. Этот инструмент очень важен для разработчиков, пишущих код Automation и которым необходимо понимать документированные (а иногда и недокументированные) пути использования отдельных объектов Automation.

Определение значений констант, заданных в объекте COM

Одной из действительно изнурительных задач, решаемых при разработке кода Automation, является оп-

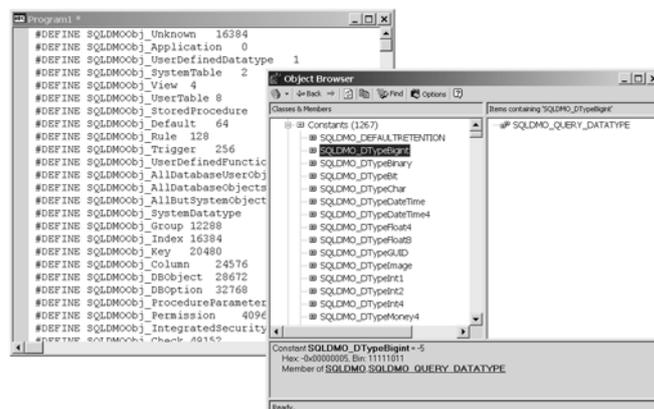


Рис. 1. Утилита Object Browser позволяет быстро получить набор инструкций #DEFINE для констант и шаблон кода класса, реализующий обработку событий.

ределение констант, использованных в этих примерах. Эти константы могут быть переведены в код #DEFINE. До того как у нас появился Visual FoxPro Object Browser, нам приходилось долго копаться в файлах Help, надеясь, что эти значения для примеров задокументированы, или использовать инструмент наподобие Object Browser, найденный в редакторах VBA для компонентов Microsoft Office, чтобы найти эти значения. Ясно, что это был процесс, требующий больших затрат времени. Теперь инструменты наподобие West Wind GetConstants.exe читают библиотеки типов и генерируют код #DEFINE, который легко компилируется в Visual FoxPro.

Object Browser является «родным» для VFP и может эффективно генерировать код #DEFINE, что, очевидно, экономит вам время. Чтобы достичь этого, откройте компонент COM или ActiveX по выбору и спуститесь вниз по TreeView, пока не увидите узел Constants (как показано на рис. 1). Откройте редактор программы (программу или метод класса). Перетащите ветку Constants и бросьте ее в редактор. Не только код #DEFINE, напечатанный вместе с именем константы и ее значением, но и документация для константы также включена в качестве комментария к коду #DEFINE, если у константы есть описание. Если вы перетащите всю ветку Constants,

то в вашем редакторе будут все константы. Вы также можете перетащить только отдельные константы, если вам необходимо определить именно их.

Примечание: я встречал константы, десятичное значение которых округлялось до нуля в VFP 7 и в рыночной бета-версии VFP 8. Если такое случается, я рекомендую программу GetConstants.exe от West Wind, не содержащую подобной ошибки.

Используйте Object Browser для создания шаблонов классов для реализации интерфейсов

Новое и очень удобное свойство Visual FoxPro 7 — это возможность писать код в наших индивидуальных приложениях, способный реагировать на события в других приложениях. Например, теперь мы можем написать код, отвечающий на закрытие пользователем электронной таблицы, или посылку электронного письма через программу Outlook, или составление стандартных писем в Word. Это достигается с помощью новой конструкции IMPLEMENTS оператора DEFINE CLASS, а также новой функции EventHandler().

Object Browser в этом отношении помогает разработчикам в написании трудоемкого кода. Прежде всего, откройте элемент управления COM или ActiveX в Object Browser. Затем спуститесь вниз по TreeView и найдите узел Interfaces. Откройте редактор программ (программу или метод класса). Перетащите узел Interface и бросьте его в редактор. Появится определение класса, включая IMPLEMENTS, и шаблонный код для каждого раскрываемого метода. Все, что вам нужно сделать на этом этапе, это переименовать класс со значения по умолчанию MyClass на что-то более наглядное и добавить код к соответствующему методу. Это поможет сэкономить огромное количество времени, даже если вы очень быстро печатаете.

Узнайте имя ОСХ-файла и файла помощи ОСХ

Одним из простейших, хотя и наиболее полезных, элементов Object Browser является то, что он отображает действительное имя файла для ОСХ и другие детали об этом элементе интерфейса.

Откройте Object Browser и выберите элемент управления ActiveX из списка. Если вы выберете корневой узел для этого элемента, то увидите детали об ОСХ, отображенные в нижней панели Object Browser. Разработчику будет представлена такая информация, как имя файла, файл справки и GUID элемента. Это может оказаться удобным, если вам

необходимо выяснить, какой ОСХ-файл включен в развертываемый пакет и определить, где на жестком диске установлен файл справки. Поиски файлов справки могут быть довольно забавными, но зачем тратить время на просмотр системного каталога Windows или контрольного каталога, в который вы специально их загрузили? Возможно вы, подобно мне, обыскиваете весь жесткий диск в поисках СНМ-файлов или файлов HLP? Впрочем, думаю, у вас появится идея, как можно сэкономить массу времени, воспользовавшись этим простым советом.

Toolbox

Toolbox является новшеством в VFP и был разработан, чтобы «подпитать» панель инструментов Form Controls стероидами. Панель инструментов Form Controls обеспечивает разработчиков единой библиотекой классов, основными классами или элементами управления ActiveX, выбираемыми через диалоговое окно VFP Option. Toolbox обеспечивает группирование классов, элементов управления ActiveX и способность писать сценарии, состоящие из текстовых блоков. Разработчики экономят время на переключение между библиотеками классов на панели инструментов Form Controls.

Текстовые блоки

Текстовые блоки позволяют вам перетаскивать и бросать текст в редактор программ или Command Window. Если вы отмечаете текст как оценочный текст TEXTMERGE, вы можете буквально написать сценарий VFP TEXTMERGE. Итак, если вы прибегаете к открытию программ, чтобы скопировать и вставить шаблон текста, то можете использовать возможность Text Scraps из Toolbox, чтобы сэкономить время. Добавьте новый кусочек текста, напишите текст слияния текста в редакторе (окно редактирования в диалоговом окне Item Properties) и выделите флажок Evaluate using text merge.

```
<<REPLICATE(" ", 80)>>
* PROGRAM NAME:
* AUTHOR: Richard A. Schummer, <<CMONTH(DATE())>>
*                                     <<YEAR(DATE())>>
*
* COPYRIGHT © <<YEAR(DATE())>> All Rights Reserved
* Richard A. Schummer
* 2921 E. Jefferson Ave, Suite 300
* Detroit, MI 48207
* RASchummer@GeeksandGurus.com
*
* PROGRAM DESCRIPTION:
* This program
*
* CALLED BY:
*
* INPUT PARAMETERS:
* <> =
```

```

*
* OUTPUT PARAMETERS:
*   <> =
*
* LANGUAGE/VERSION:
*   <<VERSION()>> or higher
*
<<REPLICATE("**", 80)>>
*
*                               C H A N G E   L O G
*
*   Date      Dev   System   Description
*   -----
* <<PADR(TRANSFORM(DATE()), 10)>> RAS   Created prog
* <<REPLICATE("-", 78)>>
*
<<REPLICATE("**", 80)>>

```

Необходимо отметить, что вы можете делать те же самые вещи, используя IntelliSense. Я думаю, что проще развить быстрый сценарий в редакторе Toolbox, чем создавать необходимое расширение кода в IntelliSense Manager, особенно если вы не знакомы с техническими приемами по созданию кода в IntelliSense. Пара проблем, с которыми я столкнулся в этом примере, заключались в том, что дата печаталась как если бы настройка SET CENTURY имела значение OFF (эта настройка формы Toolbox не была доступна в бета-версии) и WONTOP() возвращала имя формы Toolbox. Я хочу использовать WONTOP() для получения имени программы из редактора, но, к сожалению, этот технический прием не может быть использован.

Элементы управления ActiveX

Перетаскивание элементов управления ActiveX в окно конструктора форм или классов будет инициализировать элементы в форме или классе. Перетаскивание элементов управления ActiveX в окно редактора создает необходимый код NEWOBJECT(). Вы можете просто изменить это в операторе объявления LOCAL, чтобы активизировать IntelliSense в редакторе. Следующий код был создан, когда я бросил DynaZip ActiveX Control в редактор:

```
olecontrol = NEWOBJECT("dzactxctrl.dzactxctrl.1", ;
    "dzactx.dll")
```

Следующая строка может быть скопирована поверх этой строки и изменена следующим образом:

```
LOCAL loZip AS "dzactxctrl.dzactxctrl.1"
```

Этот метод очень продуктивен, поскольку я больше не нуждаюсь в поиске регистрации ActiveX в реестре Windows или в файле справки (если она даже существует). Теперь эта объектная ссылка в виде переменной памяти определена с помощью конструкции AS оператора LOCAL, IntelliSense помогает, когда эта переменная используется в редакторе (как показано на рис. 2). Я нашел, что этот редактор

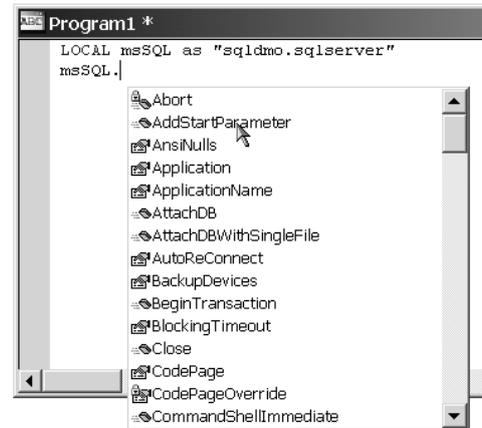


Рис. 2. Функциональность IntelliSense включается, когда ссылка на объект создается посредством конструкции LOCAL AS.

иногда не может распознать переменную памяти как объектную ссылку на элемент ActiveX. Я не могу воспроизвести это по требованию, но нашел, что выбор положения курсора на строке и использование стрелки вниз для перемещения через объявления LOCAL снова вернут IntelliSense в рабочее состояние. Несмотря на то, что я никогда не претендовал на понимание сущности VFP, я полагаю, что это разновидность «перерегистрации» ссылки ActiveX в редакторе.

Другая продуктивная поддержка, с которой мы встречаемся в Toolbox, заключается в том, что его список элементов ActiveX содержит имена элементов в метках. Если вы используете панель инструментов Form Control, то можете выделить значок, чтобы определить, какая кнопка представляет интересующий вас элемент ActiveX, поскольку один и тот же значок используется для каждого элемента ActiveX в панели инструментов. Кроме того, я заметил, что переключение элементов ActiveX на панели инструментов может быть значительно замедлено на старых компьютерах, когда несколько элементов ActiveX было предварительно выбрано в диалоговом окне Options. Подобный список отображается и в Toolbox, но он более быстрый, даже для длинного списка.

Menu Designer

Пользовательский интерфейс Menu Designer не получал значительных улучшений в течение нескольких лет, но в VFP 8 добавлена одна функция, часто требующаяся и являющаяся очень желанным допол-

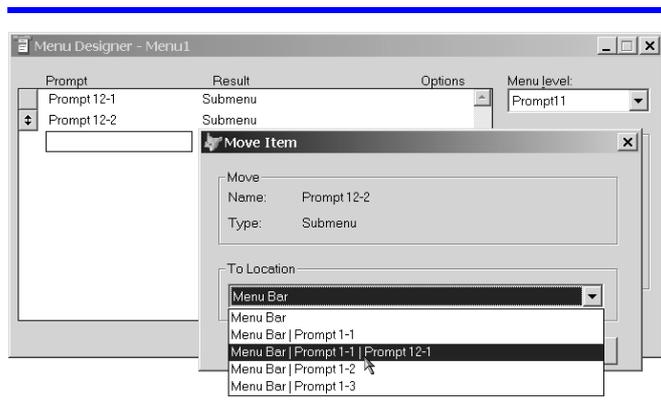


Рис. 3. Новая функциональность перемещения элементов меню давно ожидалась разработчиками.

нением для увеличения нашей производительности — поддержка перемещающихся элементов меню. Ранее разработчики должны были вырезать и вклеивать строки меню из одной панели (menu pad) в другую. Перескакивание вперед-назад для перемещения каждого элемента меню было настолько затруднительным, что большинство разработчиков просто добавляли новый элемент меню и переписывали большинство опций. Теперь вы можете выбрать элемент меню, нажать кнопку Move Item и выбрать ту часть меню, куда хотите переместить выбранный элемент (как показано на рис. 3). Этот элемент переместится вниз списка, так что вам придется редактировать раздел, в который вы переместили элемент, чтобы расположить его на подходящем месте. Я

также рекомендую вам проверить «горячие клавиши».

Вы можете перемещать не только отдельные элементы, но и целые панели из главного меню, а также из подменю. Элементы панели, перемещаемые в другие панели, становятся подменю. Перемещение подменю в главное меню создает новую панель с элементами подменю в качестве новой панели.

Заключение

В этой статье я познакомил вас с некоторыми полезными подсказками при использовании Object Browser, нового инструмента VFP 8 Toolbox и о значительных улучшениях Menu Designer в VFP 8. В следующей статье я рассмотрю пути улучшения нашей производительности с помощью советов по использованию Class Browser и новинке VFP 8 — Task Pane.

Рик Шуммер (Rick Schummer) является партнером в компании Geeks and Gurus, Inc. После работы он наслаждается написанием инструментов разработки, улучшающих производительность компании, и время от времени пишет статьи для своих любимых периодических изданий по Fox и для пользовательских групп новостей. Рик — соавтор книг MegaFox: 1002 Things You Wanted To Know About Extending Visual FoxPro и популярной 1001 Things You Always Wanted to Know About Visual FoxPro и является членом-основателем и секретарем организации Detroit Area Fox User Group (DAFUG). Кроме того, он регулярный ведущий пользовательских групп в Северной Америке на конференциях GLGDW 2000-2002, Essential Fox 2002 и VFE DevCon2K2. www.geeksandgurus.com, www.rickschummer.com, raschummer@geeksandgurus.com.



FoxTalk

русское издание

Печатается ежемесячно

Учредитель и издатель:

ООО Эдэль. Copyright © 1992-2003. Все права защищены.

Страничка в Интернете: <http://newsletter.narod.ru> или <http://msnhomepages.talkcity.redmondave/dartemov/foxtalk.htm>

(095) 325-5278
E-mail: foxtalk@online.ru
115304 Москва, а/я 208

Главный редактор: Д. Артемов
E-mail: dartemov@hotmail.com

Журнал зарегистрирован комитетом Российской Федерации по печати.

Регистрационное свидетельство
№ 015520 от 17.12.1996

FoxBASE+, FoxPro® и Visual FoxPro® являются зарегистрированными товарными знаками Microsoft Corporation.

FoxTalk (русское издание) индекс 72495

Объединенный каталог индекс 45007

Журнал для FoxPro-программистов.

FoxTalk (русское издание) индекс 72496

Журнал для FoxPro-программистов вместе с дискетой с исходными текстами программ.

FoxTalk (русское издание) индекс 72497

Подписка на старые номера журнала FoxTalk.

Библиотека программиста индексы 72769, 72490, 72491, 47771, 80375, 82841

Книги компьютерной тематики по последним версиям популярных программных продуктов.

Подписка в любом почтовом отделении связи по каталогу «Газеты. Журналы» Агентства Роспечать и «Объединенному каталогу».

Подписано в печать 30/10/03. Формат 60x90 1/8. Тираж 300 экз.