FoxTalk

Март 2003
№ 3 (69)

русское издание

Solutions for Microsoft® FoxPro® and Visual FoxPro® Developers

Публикация вашей первой службы Web Service, часть 3

Уилл Хентцен (Whil Hentzen)

В двух первых статьях этой серии вы создали службу Web Service, опубликовали ее, а затем воспользовались ее услугами — и все это было проделано на вашей инструментальной машине. Потом я подготовил обзор наиболее часто задаваемых вопросов. И вот, наконец, пришло время избавиться от коротких штанишек и шапочки с пропеллером — давайте внедрим эту штуку по-настоящему!

Развертывание службы Web Service на рабочем сервере (также называемое «публикацией») включает в себя пять шагов. Во-первых, вам необходимо создать компоненты, которые вы собираетесь установить на рабочий сервер — DLL-библиотеку и файлы WSDL и WSML. Во-вторых, нужно создать инсталляционный пакет, позволяющий должным образом установить эти компоненты. В-третьих, установите на рабочий сервер SOAP Toolkit. В-четвертых, установите DLL-библиотеку. В-пятых, установите и перенастройте упомянутые выше связанные компоненты. И, в заключение, вызовите службу Web Service с другой машины, например, с вашей инструментальной системы.

1. Создаем компоненты

Вам потребуется три набора компонентов. Первый из них — это сама DLL-библиотека, выполняющая всю работу. Второй — это файлы службы Web Service, такие как WSDL и WSML. И третий — это... данные!

Создание DLL-библиотеки

Детальное описание этого шага содержится в моей декабрьской статье 2002 года. Здесь же достаточно упомянуть, что у вас должна быть действующая служба Web Service. Помните, что она представляет собой обычную DLL-библиотеку. В этой статье я буду называть эту DLL-библиотеку FoxTalkDemo5.DLL и ее внутренний класс — NewService.

Поскольку прошло некоторое время, будет нелишним напомнить, что этот класс принимает в качестве параметра дату, но только в строчном формате, таком как “2001/11/1”. Кроме того, код в классе предназначен для поиска таблицы, называемой NEWS.DBF, в каталоге database (расположенном ниже DLL-библи-

материал имеет отношение к соответствующей версии

материал имеет отношение к соответствующей платформе

исходные тексты программ можно скачать из Интернета

отеки). Вот код, который ищет интересующую нас запись:

```
select cNews, dNews from database\NEWS ;
   where dNews >= ldDate ;
   order by dNews ;
   into array aNews
```

Вскоре я еще раз обращусь к нему.

Создание файлов WSDL и WSML службы Web Service

Чтобы опубликовать службу Web Service на инструментальной машине, используйте мастер Visual FoxPro Web Service Wizard (этот мастер вызывается из меню Tools | Wizards). С его помощью мы создадим файлы WSDL и WSML. Где же они находятся? Их расположение указывалось в диалоговом окне Advanced Settings мастера (см. рис. 9 декабрьской статьи 2002 года). На моем компьютере это c:\inetpub\webpub.

Если вы используете ASP-«слушателя» (listener), то обнаружите ASP-файл в том же каталоге, что и файлы WSDL. Не удаляйте ASP-файл, он вам еще потребуется.

2. Создаем пакет InstallShield

Теперь нужно установить DLL-библиотеку службы Web Service на рабочем Web-сервере. Еще раз подчеркну, на рабочей машине. Это значимое замечание, не так ли?

Есть много способов выполнения этой задачи, но простейший из них — использовать инструмент InstallShield Express. Вы получите инсталляционный пакет, который вы могли бы запустить на рабочем сервере. Применение InstallShield — это простой способ избежать избыточности (вам потребуется только четыре из более чем 20 допустимых опций). Кроме того, этот путь наиболее безопасен и скор (для получения дополнительной информации по использованию InstallShield Express, посмотрите мою статью в ноябрьском выпуске FoxTalk за 2002 год).

Из меню Start запустите InstallShield Express, затем щелкните Create new project и Create и далее щелкните узел General Information дерева установки, появившегося на первом шаге. Создайте InstallDir и DatabaseDir по своему выбору и/или там, где предписано вашими требованиями. Для этой статьи я поместил InstallDir в папку Program Files\Hentzenwerke\FoxTalkDemo5, а DatabaseDir — в InstallDir\Database (по соображениям безопасности вам стоит изменить расположение каталога DatabaseDir на вашем сервере и даже, если это возможно, поместить его на другом компьютере).

Перейдите к шагу 2 и выберите узел Files. В нижней части окна щелкните правой кнопкой мыши узел Destination Computer и выберите пункт меню Show Predefined Folder для того, чтобы отобразить узлы InstallDir и DatabaseDir. Используя верхнюю часть окна, перейдите к вашей DLL-библиотеке и «перетащите» ее в узел InstallDir, затем «перетащите» файл NEWS.DBF в узел DatabaseDir.

Теперь нужно определить, какие файлы runtime-модулей вы хотите установить «за компанию». Выберите узел Objects/Merge Modules на шаге 2 и выберите все три файла VC++, первый файл runtime-модуля Visual FoxPro и любые runtime-модули поддержки иных, чем английский, языков по вашему выбору.

И в заключение, выберите узел Build Your Release на шаге 6. Я выбрал SingleImage потому, что при этом создается только один файл SETUP.EXE вместо кучи дополнительных файлов, замусоривающих ваш жесткий диск. Нажмите F7 или щелкните правой кнопкой мыши SingleImage, выберите из выпадающего меню Build — и вы получите ваш файл SETUP.EXE, готовый к пересылке на рабочий сервер.

3. Устанавливаем SOAP Toolkit на ваш Web-сервер

На момент написания этой статьи последней версией SOAP Toolkit остается SP2, поэтому если у вас его до сих пор нет, перейдите на сайт msdn.microsoft.com (поищите по ключевому слову “SOAP”) и скачайте его. Сам Toolkit (без примеров и т. д.) занимает около 1,5 Мб.

Вам также стоит удостовериться, что на IIS настроен надлежащий тип «слушателя», в зависимости от того, собираетесь ли вы использовать ISAPI или ASP. Детальное описание того, как это делается на рабочем сервере, вы найдете во второй статье этой серии.

4. Устанавливаем файлы InstallShield на ваш рабочий Web-сервер

Скопируйте файл SETUP.EXE на ваш рабочий сервер — куда, не имеет значения. Будем надеяться, у вас есть специальное расположение по умолчанию для файлов, подобных этому. Если у вас работают какие-либо приложения Visual FoxPro, вам следует остановить их перед запуском процесса установки. Если вы не сделаете этого, то получите сообщение об ошибке, как показано на рис. 1.

Затем запустите файл SETUP.EXE. Инсталлятор поместит вашу DLL-библиотеку в надлежащее место — в моем случае, это Program Files\Hentzenwerke\FoxTalkDemo5, — а также зарегистрирует

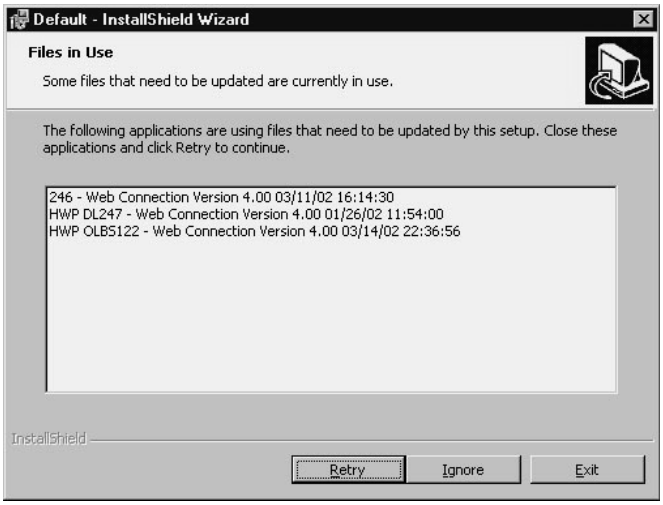

Рис. 1. Убедитесь, что остановили все приложения, использующие файлы, обновляемые в ходе установки вашей службы Web Service.

ее, как и должно быть со всеми COM-серверами.

Теперь пришло время создать папку в каталоге Program Files\Hentzenwerke, называемую “database”, и скопировать в нее файл NEWS.DBF.

5. Устанавливаем сопутствующие компоненты на ваш рабочий Web-сервер

Служба Web Service, если вы помните, это просто DLL-библиотека, помещенная на ваш Web-сервер, к которой могут обращаться другие компьютеры из Интернета (или вашей внутренней сети — Intranet). Методы этой библиотеки ничем не отличаются от методов обычных COM-серверов. Но прямого доступа к библиотеке нет, для обращения к ее методам необходим файл WSDL, описывающий каким должен быть интерфейс, какие типы данных могут читаться и т. д. Итак, вам также необходимо поместить файлы WSDL, WSML и ASP на рабочий сервер.

Когда пользователи службы Web Service пытаются воспользоваться ею, они компилируют небольшой фрагмент кода, ссылающийся на следующий URL:

```
http://www.yourwebsite.com/SomeDir/NewsService.WSDL
```

Таким образом, вам необходимо создать виртуальный каталог, отображающийся на физический каталог SomeDir. Затем поместите файлы WSDL, WSML и ASP в этот виртуальный каталог так, чтобы они могли исполняться вашими пользователями.

Но это еще не все, и этот шаг сбивает с толку множество людей. Дело в том, что файл WSDL был сгенерирован на вашей инструментальной системе и, вероятно, содержит ссылки на этот компьютер. Когда вы переместите ваш WSDL-файл на рабочий компьютер, он будет продолжать ссылаться на инструментальный. Например, WSDL-файл, созданный на моем инструментальном компьютере, содержит следующие строки (ближе к концу):

```
<service name='NewsService' >
   <port name='NewsServiceSoapPort'
      binding='wsdlns:NewsServiceSoapBinding' >
     <soap:address location=
       'http://HERMAN/Webpub/NewsService.ASP' />
   </port>
</service>
```

Имя моего инструментального компьютера — HERMAN, и каталог Webpub находится на диске C: этого компьютера. Эта информация не будет верной для тех, кто пытается воспользоваться службой NewsService на моем рабочем Web-сервере.

Поэтому я изменил эту строку следующим образом:

```
<soap:address location=
   'http://www.yourwebsite.com/sns/NewsService.ASP' />
```

где “sns” — имя моего виртуального каталога.

6. Тестируем

Итак, служба NewsService установлена и готова к запуску на вашем рабочем сервере. Волнующий момент! Давайте посмотрим, будет ли она на самом деле работать.

Вернитесь снова на вашу инструментальную машину и запустите Visual FoxPro 7.0. Пришло время зарегистрировать эту службу Web Service, чтобы вы могли пользоваться ею. Запустите IntelliSense Man-

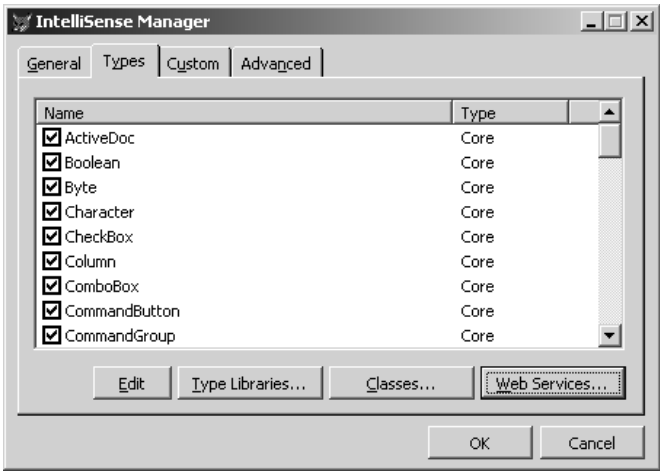

Рис. 2. Вкладка Types IntelliSense Manager.

ager (из меню Tools) и щелкните вкладку Types, как показано на рис. 2.

Щелкните кнопку Web Services, чтобы вызвать диалоговое окно Visual FoxPro Web Services Registration. Введите имя службы Web Service и введите URL для WSDL-файла на вашем рабочем сервере, как это показано на рис. 3.

Если вы используете вашу инструментальную

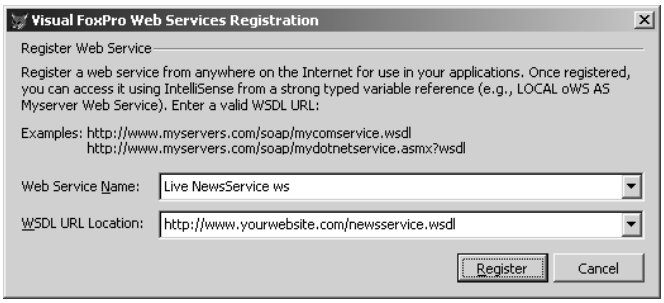

Рис. 3. Ввод URL для вашей рабочей службы Web Service.

систему для тестирования рабочей службы Web Service, то можете столкнуться с одной проблемой. Когда вы публиковали рабочую службу Web Service на вашей инструментальной системе для создания WSDL-файлов, вы дали ей имя (будучи человеком логического склада ума), что-то означающее для вас в тот момент. Теперь, когда вы хотите воспользоваться вашей службой Web Service (установленной теперь на рабочем сервере), то, возможно, соблазнитесь логично звучащим для нее именем – и это окажется то же имя, что вы использовали ранее. К счастью, мастер VFP Web Services Registration Wizard предупредит вас о повторном использовании имени.

После того, как вы успешно зарегистрируете службу, вы получите показанное на рис. 4 диалоговое окно.

Для подтверждения внесенных изменений щелк-

Рис. 4. Диалоговое окно, указывающее, что ваша служба Web Service успешно зарегистрирована.

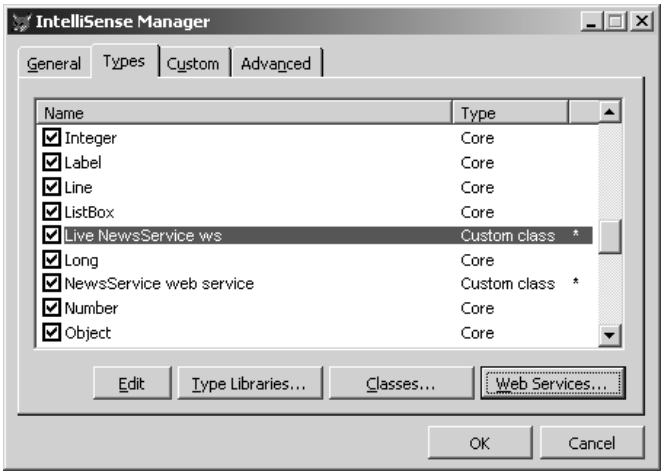

Рис. 5. После регистрации ваша служба Web Service имеет запись в IntelliSense.

ните OK в этом диалоговом окне, и вы вернетесь в окно IntelliSense Manager с новым доступным типом, как это показано на рис. 5.

Теперь пришло время провести тестирование. Создайте файл программы тестирования на вашей инструментальной системе и введите следующий код:

```
local ox as
```

IntelliSense Manager отобразит все доступные типы, как это показано на рис. 6. Обратите внимание на то, что “Live NewsService ws” указана в списке

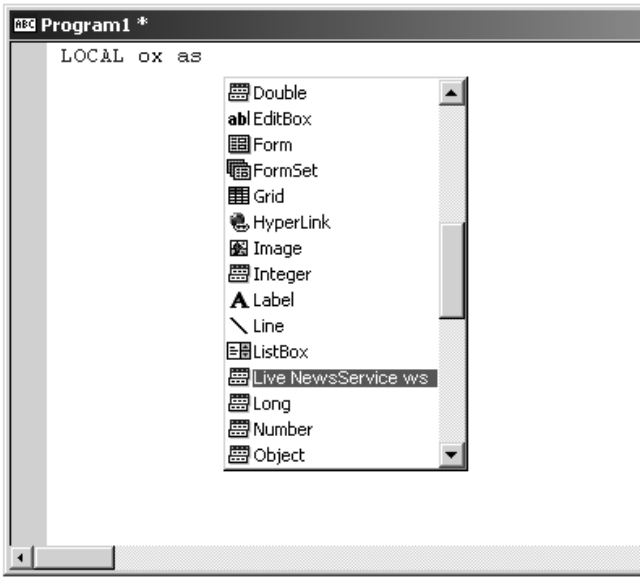

Рис. 6. Регистрация вашей рабочей службы Web Service на инструментальной системе позволяет отразить ее как тип, доступный в IntelliSense.

служб, доступных для выбора. Вот для чего мы воспользовались помощью приложения IntelliSense Manager.

Выбор Live NewsService ws из раскрывающегося списка (combo box) приведет к полному изменению кода, вводимого в файл вашей тестовой программы:

```
local ox as Live NewsService ws
 LOCAL loWS
 loWS = NEWOBJECT("Wsclient",HOME() + ;
   "ffc\_webservices.vcx")
 loWS.cWSName = "Live NewsService ws"
 ox = loWS.SetupClient( ;
   "http://www.yourwebsite/sns/newsservice.wsdl", ;
   "NewsService", "NewsServiceSoapPort")
```

Теперь введите:

```
m.lcNews = ox.GetNews("2001/11/1")
messagebox(m.lcNews)
```

и запустите программу. Вы получите диалоговое окно, отображающее запись из NEWS.DBF, как это было и в предыдущей статье.

Уверен, она работает, но…

Итак, как и в предыдущих статьях, она также не будет работать. По крайней мере, не с первого раза. И не со второго. К счастью, на этот раз большинство ошибок относилось к самому процессу, а не к синтаксическим неточностям. Поскольку теперь вы знаете для чего предназначены эти файлы и какие шаги вам следует выполнить, все должно работать гладко. Но все-таки остается несколько возможных проблем. Давайте рассмотрим их.

Во-первых, вы должны разобраться с кучей ошибок, описанных в моих двух предыдущих статьях. Они связаны с невозможностью обнаружения файла NEWS.DBF, или это ошибки OLE, подобные следующим:

```
OLE error code 0x800a12be: Unknown COM status code
OLE error code ox80070057: The parameter is incorrect
```

В предыдущих статьях подробно описано как исправлять их. Думаю, нет смысла повторять эти ответы еще раз. Единственное различие заключается в том, что после исправления ошибок вы должны восстановить вашу DLL-библиотеку (и, возможно, WSDL-файл) на рабочем сервере.

Другой проблемой, с которой вы можете столкнуться, могут стать необъяснимые или недопустимые отклики вашего рабочего сервера — иногда ваш сервер даже не будет отвечать на запрос. Единственный совет в этой ситуации заключается в том, что вы должны удалить SOAP и затем убедиться, что ваши «слушатели» настроены корректно. Вспомните шутку о четырех специалистах по программному обеспечению, у которых спустила шина. Один из них предложил всем выйти из машины, обойти вокруг нее и затем вернуться назад и тогда, возможно, проблема решится сама собой! Нечто подобное происходит и в этом случае. Многие вещи в программном обеспечении остаются настолько несовершенными, что не стоит ожидать от него абсолютно надежной работы. Иногда вам стоит просто выйти из машины, немного подождать, а затем вернуться назад. И все будет в порядке!

Заключение

На этом мы завершаем серию статей о вашей первой службе Web Service. Это было достаточно долгое и напряженное путешествие, но оно могло оказаться еще длиннее без поддержки команды Fox: Стивена Блэка (Steven Black), Гэри Де Витта (Gary DeWitt), Кевина МакНиша (Kevin McNeish) и Теда Роше (Ted Roche). Я даже не знаю, как простые смертные надеются преодолеть его самостоятельно. Я же приберегу мои язвительные комментарии о том, насколько эта материя несовершенна, чтобы потом рекламировать ее в качестве следующей «большой вещи». Сегодня же будьте готовы потратить некоторое дополнительное время на несколько первых внедрений ваших служб Web Service. Будем надеяться, что приложения, с которыми вы работаете, будут приносить результаты, стоящие затраченного вами труда.

❑❑❑

Не пропустите главного события!

Майк Хелланд (Mike Helland)


Visual FoxPro является прекрасным объектно-ориентированным языком — практически нет вещей, которые бы этот язык не мог делать. Однако есть функция, которую объектная модель VFP выполняет не лучшим образом — это «инициирование» пользовательских событий из объектов. Майк Хелланд уже рассмотрел некоторые подходы к выполнению подобных трюков в собственных классах VFP. Но его сегодняшняя статья представляет другое решение.

Прежде всего, что я имею в виду, говоря: «инициировать событие». Есть ли в основных классах FoxPro такие события, как Click(), Init() и Destroy()? Да, есть, но они не очень полезны, когда вы хотите добавить события в ваши классы бизнес-логики. Например, в классе, обрабатывающем заказы, может появиться неправильный заказ. Приложение, использующее этот объект, может нуждаться в отображении информации о подобных проблемах по мере их появления. Это может быть сделано либо через пользовательский интерфейс, либо, если этот интерфейс отсутствует, неверная транзакция должна быть записана в журнал. Также приложение может решить, что этот заказ может быть исправлен и даже предъявлен повторно без участия пользователя. Все, что нужно сделать, — дать классу способность «инициировать» событие, или, другими словами, извещать приложение, что что-то случилось и дать ему способность разумно реагировать на это событие.

Прежде чем я возьмусь за решение этой задачи, позвольте мне представить класс, который я буду часто использовать в этой статье. Этот класс просто создает строку, загружает ее в файл, а затем удаляет этот файл. Я знаю, не очень продуктивно использовать быстрые процессоры, но, по крайней мере, выполнение этой задачи займет 1-2 секунды. Вот этот класс:

```
DEFINE CLASS routine AS Session
Iterations = 500
fileName = 'file.txt'
fileContents = ''

 FUNCTION Run
 * повторяет заданное количество раз
 LOCAL iteration
 FOR iteration = 1 TO this.iterations
   * создает файл
   this.createFileContents()
   * помещает его на диск
   this.sendFileToDisk()
   * удаляет этот файл
   this.deleteFile()
 ENDFOR
 RETURN

 FUNCTION createFileContents
 * добавляет произвольную строку к другой строке 1000 раз
 this.fileContents = ''
 LOCAL i
 FOR i = 1 TO 1000
   this.fileContents = this.fileContents + SYS(2015)
 ENDFOR
 RETURN

 FUNCTION sendFileToDisk
 STRTOFILE(this.fileContents, this.fileName)
 RETURN

 FUNCTION deleteFile
 ERASE (this.fileName)
 RETURN

 ENDDEFINE
```

Теперь мне необходимо найти способ, позволяющий записать в журнал, что случилось в ходе процесса, а также проинформировать об этом пользователя. Первым подходом будет изменение метода Run() за счет включения в него метода, вызывающего регистрацию событий в журнале и информирующего пользователя, вот так:

```
FUNCTION Run
 * повторяет заданное количество раз
 LOCAL iteration
 FOR iteration = 1 TO this.iterations
   * создает файл
   this.createFileContents()
   *записывает в журнал, что эта строка была создана
   this.Log(iteration)
   * добавляет его на диск
   this.sendFileToDisk()
   *сообщает пользователю, что файл был помещен на диск
   this.Progress(iteration)
   * удаляет этот файл
   this.deleteFile()
 ENDFOR
 RETURN
```

Теперь я добавил мои функции Log() и Progress() в мой класс и фактически реализовал их.

Что-то здесь выглядит не так. Представляется, что теперь этот класс содержит функциональные возможности, выходящие за пределы того, что класс Process должен содержать. Полагаю, что могу не помещать никакого кода в функции Log() и Progress() и оставить их для реализации как подкласс. Поступив так, я могу создать различные подклассы для Web- или Windows-приложений... Постойте, но это будет означать, что я продублировал код, выполняющий регистрацию в журнале, если индикатор хода процесса был другим. Я не уверен, что такое решение мне нравится.

У меня возникла и другая идея, заключающаяся в том, чтобы изменить ход программы и позволить пользовательскому интерфейсу запрашивать ее части отдельно. Я имею в виду, что форма будет выполнять цикл FOR и вызывать методы createFileContents() и sendFileToDisk(), учитывая расположенный между ними специальный код. Все же это опять не очень устраивает меня. Мне трудно угодить, но в идеальном случае логика процесса, логика регистрации и бизнес-логика должны быть реализованы в собственных классах, и каждый класс не должен абсолютно ничего знать о существовании других.

Поэтому для выполнения задачи я решил использовать вызов пользовательских событий и функциональность делегатов (delegate). Делегаты — это методы обработки событий в языках .NET, таких как C# и VB.NET. Я предпринял попытку воссоздать эту функциональность в Visual FoxPro 7.0.

Подготовка функции Process()

Во-первых, я решил изменить мой метод Run() для «инициирования» пользовательского события, вызывая RaiseEvent в коде процессора событий. В этом случае, процессор — это просто объект, на который можно сослаться при помощи глобальной переменной “oE”. Реализовав эту задачу, вы сможете выполнять и подобные ей, добавляя этот процессор событий к другому объекту или даже создавая экземпляр процессора в каждом используемом вами объекте. Это — по вашему усмотрению. В любом случае, новый код для метода Run() выглядит так:

```
* создает файл
this.createFileContents()
oE.RaiseEvent(this, 'STRINGCREATED', ;
  TRANSFORM(iteration))

* записывает его на диск
this.sendFileToDisk()
oE.RaiseEvent(this, 'FILEONDISK', TRANSFORM(iteration))
```

Эти две новые строки попросту «инициируют» события StringCreated и FileOnDisk объекта, передаваемого в первый параметр. Последний параметр метода RaiseEvent является строчным представлением любых параметров, которые будут передаваться в процессор событий. Кроме того, нам необходимо иметь способ отвечать на эти события, и именно тут нам пригодятся делегаты.

Метод BindEvent() класса процессора событий содержит четыре параметра. Первые два параметра описывают, какое событие ожидается (имя этого события и объект, его сформироваший), а два последних параметра описывают, как обработать это событие (объект и метод выполнения, когда событие свершится). Поскольку код, который на самом деле обрабатывает событие, делегируется в метод другого объекта, эти два последних параметра составляют то, что принято называть делегатом. Вот как выглядит типичный метод BindEvent():

```
oE.BindEvent(oO, 'STRINGCREATED', oX, 'Update')
```

Этот код может быть прочитан так:

```
"Когда событие StringCreated объекта oO
произойдет, вызывается oX.Update()"
```

BindEvent() обычно вызывается извне класса, сформировавшего события. Как следствие, код, используемый классом, создающим эти события, часто выглядит подобным образом:

```
* Создает процессор событий и метод (процедуру)
 oE = CREATEOBJECT('eventprocessor')
 oO = CREATEOBJECT('routine')

 * Создает обработчики событий
 oX = NEWOBJECT('logger')
 oY = NEWOBJECT('progress')

 * Связывает сформированное событие с обработчиками событий
 oE.BindEvent(oO, 'STRINGCREATED', oX, 'Update')
 oE.BindEvent(oO, 'FILEONDISK', oY, 'Update')

 * Выполняет «черновую» работу
 oO.Run()
```

Комментарии в этом коде поясняют, что происходит. Создаются четыре объекта: процессор событий, процедура, с которой я начинал, и два обработчика событий, работу которых я объясню позже. И, в заключение, пятая и шестая строка связывают все строки кода вместе. После чего метод Run() приводит все это в движение.

Осталось рассмотреть еще две вещи в этом коде: классы процессора событий и двух обработчиков событий.

Обработчики событий и процессор событий

Два объекта, обрабатывающие события (oX и oY), извлекаемые из процедурного класса (oO), очень просты. Оба они являются классами с функцией Update, отображающей переданный параметр:

```
DEFINE CLASS logger AS Session
 FUNCTION Update
 LPARAMETERS iteration
 ? iteration
 RETURN
 ENDDEFINE

 DEFINE CLASS progress AS Session
 FUNCTION Update
 LPARAMETERS iteration
 WAIT WINDOW TRANSFORM(iteration) NOWAIT
 RETURN
 ENDDEFINE
```

Этого в основном достаточно. Теперь соберем все вместе: процессор событий собственной персоной.

```
DEFINE CLASS eventProcessor AS custom
 DIMENSION delegates[1, 3]
 delegateCount = 0

 FUNCTION BindEvent(object,event,eventHandler,delegate)
 WITH this
   .delegateCount = .delegateCount + 1
   DIMENSION .delegates[.delegateCount, 4]
   .delegates[.delegateCount, 1] = object
   .delegates[.delegateCount, 2] = event
   .delegates[.delegateCount, 3] = eventHandler
   .delegates[.delegateCount, 4] = delegate
 ENDWITH
 RETURN

 FUNCTION RaiseEvent(object, event, parameters)
 IF EMPTY(parameters)
   parameters = ''
 ENDIF
 event = UPPER(event)
 WITH this
   FOR i = 1 TO .delegateCount
     IF .delegates[i, 2] == event AND ;
        .delegates[i, 1] = object
       .handleEvent(.delegates[i, 3], ;
         .delegates[i, 4], parameters)
     ENDIF
   ENDFOR
 ENDWITH
 RETURN

 PROTECTED FUNCTION handleEvent(object, ;
   delegate, parameters)
 LOCAL expression
 expression = 'object.'+delegate+'('+parameters+')'
 EVALUATE(expression)
 RETURN

 ENDDEFINE
```

Как мы видим, метод BindEvent() просто сохраняет всех делегатов в массиве, а метод RaiseEvent() просто просматривает этот массив в поисках любых делегатов, определенных для этого события. Если они есть, защищенный метод handleEvent() выполняет делегата.

Прежде чем отправиться дальше, я хотел бы остановиться на том, где имеет смысл использовать события и делегатов. Как вы можете видеть из моего простого примера, я могу создать более инкапсулированные классы для запуска процедуры, внесения записей в журнал о статусе процедуры и отображения хода выполнения процедуры, не обременяя себя ограничениями и правилами и используя более гибкие схемы. Также я могу использовать эти технические приемы и для более стандартных функций. Например, я могу создать объект menu, формирующий события при выборе его строк. Используя это, приложение может связывать такое событие с открытием формы.

Также я могу использовать мой процессор событий для решения более сложных задач. Скажем, у меня есть процедура выполнения заказа. Когда заказ приходит, получатель заказа формирует событие, и процессор заказов принимает его, чтобы выполнить черновую работу. Теперь представим, что ваш босс хочет получить электронное письмо с деталями заказов, сумма которых превышает $1000. Это достаточно простое усовершенствование — просто используем для этого метод BindEvent(), а коды для получателя заказа и процессора обработки могут оставаться неизменными.

Время ожидания в очереди

Прежде чем я завершу эту статью, давайте поговорим вот о чем. Обычно существует потребность в «инициировании», по большей части, достаточно тривиальных событий. Индикатор хода процесса хорошо иллюстрирует это. Не думаю, что пользователю важно видеть каждый шаг его обработки. На самом деле, для нашего восприятия легче видеть постепенно растущий счетчик, чем его постоянно обновляемый аналог.

В приведенном ниже блоке кода я модифицировал процессор событий. Эта усовершенствованная версия принимает больше параметров в метод BindEvent(), показывающий, как долго (в секундах) это событие может быть «отложенным». По существу, если событию «приказано» ждать одну секунду до того, как оно будет обработано, процессор событий должен допускать накапливания множества событий в течение одной секунды перед тем, как он сможет обработать их все одновременно. Вот как выглядит новый процессор событий:

```
DEFINE CLASS eventProcessor AS custom
 DIMENSION queuedEvents[1, 3]
 DIMENSION delegates[1, 3]
 eventCount = 0
 delegateCount = 0

 FUNCTION BindEvent(object, event, eventHandler, ;
     delegate, delay, afterBatch)
 WITH this
   .delegateCount = .delegateCount + 1
   DIMENSION .delegates[.delegateCount, 8]
   .delegates[.delegateCount, 1] = object
   .delegates[.delegateCount, 2] = event
   .delegates[.delegateCount, 3] = eventHandler
   .delegates[.delegateCount, 4] = delegate
   .delegates[.delegateCount, 5] = IIF(EMPTY(delay), ;
       0, delay)
   .delegates[.delegateCount, 6] = 0 && Next run
   * это - delegateID
   .delegates[.delegateCount, 7] = SYS(2015)
   .delegates[.delegateCount, 8] = afterBatch
 ENDWITH
 RETURN

 FUNCTION RaiseEvent(object, event, parameters)
 IF EMPTY(parameters)
   parameters = ''
 ENDIF
 event = UPPER(event)
 WITH this
   FOR i = 1 TO .delegateCount
     IF .delegates[i, 2] == event AND ;
        .delegates[i, 1] = object
       * если в столбце задержки содержится значение,
       * помещает это событие в очередь
       * на более позднее время.
       IF .delegates[i, 5] > 0
         .queueEvent(.delegates[i, 7], .delegates[i, 3], ;
           .delegates[i, 4], parameters)
         * Если отсутствует 6-й элемент (следующий запуск),
         * устанавливает его сейчас
         IF .delegates[i, 6] = 0
```

```
        .delegates[i, 6] = SECONDS() ;
            + .delegates[i, 5]
        ENDIF
      ELSE
        * в противном случае выполняет его немедленно
        .handleEvent(.delegates[i, 3], ;
            .delegates[i, 4], parameters)
      ENDIF
    ENDIF
  ENDFOR
ENDWITH
RETURN

PROTECTED FUNCTION handleEvent(object, ;
    delegate, parameters)
LOCAL expression
expression = 'object.'+delegate+'('+parameters+')'
EVALUATE(expression)
RETURN

PROTECTED FUNCTION queueEvent(delegateId, object, ;
    delegate, parameters)
WITH this
* добавляет это в набор ждущих очереди событий
.eventCount = .eventCount + 1
DIMENSION .queuedEvents[.eventCount, 4]
.queuedEvents[.eventCount, 1] = delegateId
.queuedEvents[.eventCount, 2] = object
.queuedEvents[.eventCount, 3] = delegate
.queuedEvents[.eventCount, 4] = parameters
ENDWITH
RETURN

FUNCTION DoEvents(forceEvents)
LOCAL i, seconds
seconds = SECONDS()
* когда таймер срабатывает, она должна знать, какие
* события вызывать
WITH this
  FOR i = 1 TO .delegateCount
    IF forceEvents OR .delegates[i, 6] <= seconds
      * Обрабатывает все события,
      * помещенные в очередь для делегирования
      .doEventsForDelegate(.delegates[i, 7])
      .delegates[i, 6] = 0
      IF NOT EMPTY(.delegates[i, 8])
        .handleEvent(.delegates[i, 3], ;
          .delegates[i, 8], '')
      ENDIF
    ENDIF
  ENDFOR
ENDWITH
RETURN

PROTECTED FUNCTION doEventsForDelegate(delegateId)
LOCAL i, j
j = 1
FOR i = 1 TO .eventCount
  IF .queuedEvents[j, 1] == delegateID
    .handleEvent(.queuedEvents[j, 2], ;
        .queuedEvents[j, 3], .queuedEvents[j, 4])
    .removeQueuedEvent(j)
  ELSE
    j = j + 1
  ENDIF
ENDFOR
RETURN
PROTECTED FUNCTION removeQueuedEvent(eventIndex)
WITH this
  ADEL(.queuedEvents, eventIndex)
  .eventCount = .eventCount - 1
  IF .eventCount > 0
    DIMENSION .queuedEvents[.eventCount, 4]
  ENDIF
ENDWITH
RETURN
ENDDEFINE
```

Новый код работает несколько по другому, чем тот, который я вам представил ранее. Вот быстрое повторение того, как он работает:

- 1. Когда создан делегат и «задержка» передана, второй метод можно запускать после того, как все делегаты в очереди будут выполнены. Новый вызов метода BindEvent() может выглядеть следующим образом:

```
oE.BindEvent(oO, 'STRINGCREATED', oX, 'Update', ;
   .25, 'After_Update')
```

  Число .25 означает задержку на четверть секунды.
- 2. Когда событие происходит и появляется делегат с задержкой, событие помещается в очередь.
- 3. Метод DoEvents() вызывается для запуска из очереди и запускает каждое «созревшее» событие. Если передано .T., то форсируется исполнение всех событий. Это означает, что метод Routine.Run() был изменен следующим образом:

```
* удаляет файл
  this.deleteFile()
  oE.DoEvents()
ENDFOR
* Удостоверяется, что в очереди
* остаются некоторые события
oE.DoEvents(.T.)
RETURN
```

- 4. После того, как делегат будет выполнен, вызывается следующий за ним делегат, если таковой определен. Чтобы использовать преимущества этого способа, объект logger может быть изменен следующим образом:

```
DEFINE CLASS progress AS Session OLEPUBLIC
 iteration = 0
 FUNCTION Update
 LPARAMETERS iteration
 this.iteration = iteration
 RETURN
 FUNCTION After_Update
 ?this.iteration
 RETURN
 ENDDEFINE
```

И это все на сегодня. Есть еще пара других причин для того, чтобы делать нечто подобное. Например, если вы хотите послать сообщения на диск или службе Web Service, то вместо того, чтобы посылать каждое сообщение отдельно (что может потребовать множества возвратов), вы можете просто позволить им накапливаться огромными порциями перед отправкой.

Будем надеяться, что вы сможете использовать приведенный здесь исходный код и найдете хорошее, творческое применение этому подходу. Я также надеюсь, что в будущем продолжу обсуждение событий и делегатов и того, как они могут влиять на ваш стиль разработки.

Майк Хелланд — директор по разработкам компании ARS Solutions, LTD, расположенной в центральной Миннесоте, США. Майк обладает статусом Microsoft MVP и является международно признанным автором и лектором. Его адрес: mgh@arss.com

❑❑❑

Вопрос жизни и смерти

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

WIN 7.0 DOWNLOAD

В этом месяце Энди Крамек и Марсиа Акинз приступают к установлению точной последовательности событий, которые происходят при создании экземпляра объекта формы или при его удалении. То, что они обнаруживают, не вполне соответствует тому, что они ожидали, и это заставляет их пересмотреть свои представления относительно определения элементарного класса формы.

Энди: Недавно меня спросили, обязательно ли включать обращение к методу Refresh() формы в программный код для события Init()? (Я полагаю, задавший этот вопрос разработчик выполнял настройку элемента управления combobox прямо в событии Init()).

Марсиа: Только в том случае, если приложение исполняется слишком быстро, и ты хочешь немного замедлить его работу. Нет, разумеется, ты не обязан явно вызывать метод Refresh() из события Init(); такое обращение ты получаешь задаром.

Энди: Хорошо, но я думаю, что вопрос не столько в том, обновляется ли форма при создании ее объекта, сколько в том, когда это происходит.

Марсиа: Это просто. Если хочешь знать последовательность событий, спроси об этом у “Lisar G”.

Энди: Кто это — Lisar G?

Марсиа: Последовательность событий при инициализации формы, дурачок: Load, Init, Show, Activate, Refresh, GotFocus.

Энди: Но это не все события, которые наступают в момент запуска формы! Как насчет среды окружения данных? И где все размещенные в форме элементы управления?

Марсиа: А, теперь это совершенно другой вопрос. Давай закончим с первым, прежде чем переходить к новому вопросу, согласен? Краткий ответ таков — нет необходимости явно обращаться к методу Refresh() формы во время ее инициализации, поскольку все равно этот метод вызывается сразу же после наступления события Activate() формы.

Энди: Единственным исключением была бы ситуация, в которой тебе потребовалось выполнить метод Refresh() перед исполнением некоторого программного кода, размещенного в методе Show() или событии Activate() формы.

Марсиа: Но в любом случае, чего ради тебе понадобилось бы выполнять подобные действия? Единственной причиной могла бы стать необходимость заполнить данными элементы управления, однако, следует начать с того, что ты не должен был полагаться при этом на метод Refresh() формы. Вместо этого тебе следовало бы иметь отдельный метод (с именем SetUpForm() или что-то в этом роде), к которому ты можешь обращаться всякий раз при необходимости заполнить данными элементы управления.

Энди: Эй, я не сказал, что это был правильный способ действий! Как бы там ни было, теперь, когда мы разобрались с вопросом о методе Refresh(), как насчет остальных вопросов? Где события среды окружения данных?

Марсиа: Это уже более конкретный вопрос, и тут нам необходима полная уверенность, поэтому давай проведем проверку и посмотрим на результаты. «Включи» протоколирование событий и создай простую форму (используй только основные классы) с таблицей в среде окружения данных этой формы, а затем исполни эту форму с помощью команды DO FORM.

Энди: Хорошо, если ты настаиваешь. Но я тебя предупреждаю — это не слишком надежный инструмент. Прежде всего, он лжет! Согласно списку событий, ты можешь проследить за наступлением событий OpenTables() и BeforeOpenTables(); оба они являются событиями среды окружения данных, но не присутствуют в полученных результатах. Во-вторых, как бы там ни было, этот способ контроля в действительности охватывает не все события формы, потому что он исключает метод Show(). И третье, этот способ не показывает также и Refresh(), потому что это метод. Наконец, если ты помещаешь выходные данные в файл, то не можешь в этом случае найти

способ для просмотра полученного протокола работы, если не закончишь сеанс работы с Visual FoxPro и не перезапустишь ее. Установки Event Tracking Off, Clear All, Close All и Release All по-прежнему оставляют log-файл в таком состоянии, когда он недоступен. Но, так или иначе, вот результаты, полученные для простой формы:

```
34274.138, form.Load()
34274.138, form.dataenvironment.cursor1.Init()
34274.138, form.dataenvironment.Init()
34274.138, form.txtcust_id.Init()
34274.138, form.lblcust_id.Init()
34274.138, form.txtcompany.Init()
34274.148, form.lblcompany.Init()
34274.148, form.Init()
34274.168, form.Activate()
34274.168, form.txtcust_id.When()
34274.178, form.GotFocus()
34274.178, form.txtcust_id.GotFocus()
34274.178, form.txtcust_id.Message()
```

Марсиа: Что ж, действительно, это мало помогает, верно? Такое впечатление, что все, произошедшее до наступления события Load() формы, просто не было занесено в протокол. Я думаю, в конце концов, нам лучше придумать что-то свое. Принимайся за дело и создай еще одну форму, и мы будем явно фиксировать исполнение и наступление всех методов и событий, которые нас особенно интересуют. К счастью, функция STRTOFILE() делает эту задачу совсем легкой; единственная строка программного кода — вот и все, что нам надо (но не потеряй какие-либо предложения Parameter!):

```
STRTOFILE (PROGRAM() + CHR(13) + CHR(10), "logfile.txt", 1)
```

Энди: Так, я создал форму (см. рис. 1), все основные методы и события которой снабжены предложенным тобой программным кодом.

Марсиа: Хорошо. Я вижу, ты включил элементы управления page frame и grid. Очень мило с твоей стороны! Это должно облегчить выяснение того, что же делается в действительности. Итак, что происходит, когда ты исполняешь эту форму?

Рис. 1. Форма для регистрации событий и методов.

Энди: Вот все, что произошло, вплоть до наступления события Init() формы:

```
FORM.DATAENVIRONMENT.OPENTABLES
FORM.DATAENVIRONMENT.BEFOREOPENTABLES
FORM1.LOAD
FORM1.DATAENVIRONMENT.CURSOR1.INIT
FORM1.DATAENVIRONMENT.INIT
FORM1.CMDEXIT.INIT
FORM1.CMDREFRESH.INIT
FORM1.PGFMAIN.PAGE1.GRDCUSTOMER.COLUMN2.TXTCOL2.INIT
FORM1.PGFMAIN.PAGE1.GRDCUSTOMER.INIT
FORM1.PGFMAIN.PAGE1.INIT
FORM1.PGFMAIN.PAGE2.TB01.INIT
FORM1.PGFMAIN.PAGE2.CB03.INIT
FORM1.PGFMAIN.PAGE2.TB02.INIT
FORM1.PGFMAIN.PAGE2.INIT
FORM1.PGFMAIN.INIT
FORM1.INIT
```

Марсиа: Что ж, этот перечень достаточно понятен и очень хорошо отвечает на поставленные вопросы. Все начинается с события OpenTables() среды окружения данных, которое является событием, вызывающим метод BeforeOpenTables(), а затем наступает очередь события Load() формы. После этого последовательность почти совпадает с нашими ожиданиями, верно? Первыми инициализируются объекты, помещенные в контейнеры, в том порядке, в котором они добавлялись в свои контейнеры, за ними следуют сами контейнеры, начиная со среды окружения данных. Эта часть последовательности событий и методов заканчивается наступлением события Init() формы. Кстати, эта последовательность проясняет также вопрос, почему передаваемые в форму параметры должны быть получены в методе Init(). Это первый метод формы, который исполняется после того, как стали доступны все элементы управления, но до того, как какой-либо из них был реально обновлен.

Энди: Да. Следующая стадия процесса связана с обновлением элементов управления из предусмотренных для них источников данных (в действительности, это именно то, что делает метод Refresh()) и визуализацией формы, и выглядит вот так:

```
FORM1.SHOW
FORM1.PGFMAIN.PAGE1.ACTIVATE
FORM1.ACTIVATE
FORM1.REFRESH
FORM1.CMDEXIT.REFRESH
FORM1.CMDREFRESH.REFRESH
FORM1.PGFMAIN.REFRESH
FORM1.PGFMAIN.PAGE1.REFRESH
FORM1.PGFMAIN.PAGE1.GRDCUSTOMER.REFRESH
FORM1.CMDREFRESH.WHEN
FORM1.GOTFOCUS
FORM1.CMDEXIT.WHEN
```

Обрати внимание на то, что обновление получает только активная страница (Page 1) страничного блока pageframe. Это означает, что элементы управления, размещенные на странице Page 2, еще не были обновлены и объясняет, почему всегда необходимо явно вызвать метод Refresh() при переходе на новую страницу.

Марсиа: Постой! Ты утверждаешь, что метод Refresh() необходим также в событии Activate() страницы Page 1? Если так, тогда ты получишь двойное обращение к методу Refresh(), тройное при создании экземпляра объекта формы, если у тебя к тому же есть такое обращение и в событии Init() тоже.

Энди: Совершенно верно! Но это лишь подчеркивает, как важно избегать необоснованных обращений к методу ThisForm.Refresh(). В качестве общего правила, я предложил бы только тогда обращаться к методу Refresh() на уровне формы, когда есть уверенность в его необходимости — другими словами, внутри некоторого рода проверки. Это правило применимо к событию Activate() страницы Page 1 в той же мере, что и где-либо еще.

Марсиа: Согласна. Между прочим, если ты создаешь экземпляр объекта формы по определению, данному в VCX-, а не в SCX-файле, последовательность событий остается той же самой за исключением, конечно, того, что отсутствуют события среды окружения данных. Хорошо, теперь, когда мы знаем, что происходит при «рождении» формы, как насчет противоположной стороны ее существования? Что происходит, когда форма «умирает»?

Энди: С этим несколько сложнее, поскольку существуют три способа «убийства» формы. Во-первых, можно удалить объектную ссылку на эту форму (то есть выполнить команду RELEASE ThisForm). Во-вторых, можно явно обратиться к ее методу Release() (то есть ThisForm.Release()), и третье, можно щелкнуть мышью по командной кнопке Close, что на самом деле приведет к вызову метода QueryUnload().

Марсиа: Эти способы чем-то отличаются?

Энди: Честно говоря, я не знаю. Давай посмотрим. Чтобы протестировать первый способ, нам необходимо исполнить (и удалить) форму, используя вот такой код:

```
DO FORM logform NAME fred LINKED
RELEASE fred
```

А вот результат:

```
FORM1.DESTROY
FORM1.PGFMAIN.DESTROY
FORM1.PGFMAIN.PAGE2.DESTROY
FORM1.PGFMAIN.PAGE2.TB02.DESTROY
FORM1.PGFMAIN.PAGE2.CB03.DESTROY
FORM1.PGFMAIN.PAGE2.TB01.DESTROY
FORM1.PGFMAIN.PAGE1.DESTROY
FORM1.PGFMAIN.PAGE1.GRDCUSTOMER.DESTROY
FORM1.PGFMAIN.PAGE1.GRDCUSTOMER.COLUMN2.TXTCOL2.DESTROY
FORM1.CMDREFRESH.DESTROY
FORM1.CMDEXIT.DESTROY
FORM1.UNLOAD
FORM1.DATAENVIRONMENT.CLOSETABLES
FORM1.DATAENVIRONMENT.AFTERCLOSETABLES
FORM1.DATAENVIRONMENT.DESTROY
FORM1.DATAENVIRONMENT.CURSOR1.DESTROY
```

Марсиа: Это почти то же, что я ожидала. Объекты уничтожаются в порядке, обратном порядку их создания. Поэтому мы начинаем с вызова метода Destroy() самого внешнего контейнера (самой формы), а затем «спускаемся» по иерархии контейнеров и повсюду сеем на своем пути разрушение!

Энди: Очень наглядно! Но это совершенно правильное представление того, что происходит. Заметь также, что хотя метод Unload() является последним вызываемым методом формы, все таблицы и среда окружения данных в этот момент все еще доступны.

Марсиа: Что ж, это, возможно, любопытный факт, но толку от него мало, поскольку у нас все равно сейчас нет способа получить доступ к этим объектам. По сути, вопрос заключается в следующем: что происходит при обращении к методу Release() формы? Я бы предположила, что последовательность событий не изменится!

Энди: Хорошо, давай посмотрим. Вот результат использования командной кнопки Exit (которая обращается к методу ThisForm.Release()):

```
FORM1.RELEASE
FORM1.CMDEXIT.VALID
FORM1.CMDEXIT.WHEN
FORM1.DESTROY
FORM1.PGFMAIN.DESTROY
FORM1.PGFMAIN.PAGE2.DESTROY
FORM1.PGFMAIN.PAGE2.TB02.DESTROY
FORM1.PGFMAIN.PAGE2.CB03.DESTROY
FORM1.PGFMAIN.PAGE2.TB01.DESTROY
FORM1.PGFMAIN.PAGE1.DESTROY
FORM1.PGFMAIN.PAGE1.GRDCUSTOMER.DESTROY
FORM1.PGFMAIN.PAGE1.GRDCUSTOMER.COLUMN2.TXTCOL2.DESTROY
FORM1.CMDREFRESH.DESTROY
FORM1.CMDEXIT.DESTROY
FORM1.UNLOAD
FORM1.DATAENVIRONMENT.CLOSETABLES
FORM1.DATAENVIRONMENT.AFTERCLOSETABLES
FORM1.PAINT
FORM1.DATAENVIRONMENT.DESTROY
FORM1.DATAENVIRONMENT.CURSOR1.DESTROY
```

Марсиа: Этот список выглядит точно так же, кроме одного исключения, которое я упустила из виду

прежде. На этот раз метод Release() вызывается и инициирует цепочку разрушений, но когда мы просто удаляем объектную ссылку, обращения к методу Release() не происходит вообще — мы перешли прямо к методу Destroy() формы.

Энди: Хм-м. Хорошее замечание. Я тоже не обратил на это внимания, но ты права. Это подразумевает, что метод Release() является не самым подходящим местом для размещения в нем программного кода, который ты хочешь исполнить при выходе из формы. Давай попробуем третий способ удаления нашей формы, с помощью командной кнопки Close:

```
FORM1.QUERYUNLOAD
FORM1.DESTROY
FORM1.PGFMAIN.DESTROY
FORM1.PGFMAIN.PAGE2.DESTROY
FORM1.PGFMAIN.PAGE2.TB02.DESTROY
FORM1.PGFMAIN.PAGE2.CB03.DESTROY
FORM1.PGFMAIN.PAGE2.TB01.DESTROY
FORM1.PGFMAIN.PAGE1.DESTROY
FORM1.PGFMAIN.PAGE1.GRDCUSTOMER.DESTROY
FORM1.PGFMAIN.PAGE1.GRDCUSTOMER.COLUMN2.TXTCOL2.DESTROY
FORM1.CMDREFRESH.DESTROY
FORM1.CMDEXIT.DESTROY
FORM1.UNLOAD
FORM1.DATAENVIRONMENT.CLOSETABLES
FORM1.DATAENVIRONMENT.AFTERCLOSETABLES
FORM1.DATAENVIRONMENT.DESTROY
FORM1.DATAENVIRONMENT.CURSOR1.DESTROY
```

Марсиа: И снова метод Release() не вызывается! Разумеется, теперь мы вместо этого проходим через метод QueryUnload(). Что происходит, если ты поместишь команду RETURN .F. в метод Destroy()? Предотвратит ли это удаление формы?

Энди: Нет! И даже команда NODEFAULT также не остановит этот процесс. Похоже на то, что как только инициировано исполнение метода Destroy() формы, ты оказываешься в ловушке, и нет пути назад.

Марсиа: В этом есть смысл. В конце концов, как можно отменить разрушение чего-то после того, как оно произошло, без повторного создания разрушенного?

Энди: Это также означает, что если ты явно удаляешь объектную ссылку, нет способа остановить разрушение объекта формы, поскольку при этом происходит обращение прямо к методу Destroy(), минуя все остальное.

Марсиа: Это также имеет свой смысл. Visual FoxPro исходит из предположения, что если ты, как программист, явно удаляешь объект, то ты сознаешь что делаешь, и можно спокойно игнорировать все возгласы протеста, которые мог бы издавать объект. Но тогда никто вообще не стал бы использовать такой способ удаления объекта в программе, верно?

Энди: Нет, не стали бы. Это довольно радикальный подход и он попадает в категорию «палить из пушки по воробьям». Не подходит!

Марсиа: Впрочем, это говорит о том, что если ты хочешь иметь форму, которая выполняет какую-то проверку и принимает некоторое решение, когда пользователь (в отличие от программиста) говорит ей, что она должна «удалиться», необходимо иметь метод, который вызывается как из метода Release(), так и из метода QueryUnload().

Энди: Но можно ли остановить процесс разрушения из этих методов?

Марсиа: Да. Необходимо только выполнить в них команду NoDefault. Итак, вот такой код выполнит этот трюк в обоих методах:

```
IF NOT ThisForm.OK2Destroy()
  NODEFAULT
ENDIF
```

Энди: Да, и как мы уже видели, тебе определенно необходимо разместить этот код в явном виде в обоих методах. Таким образом, складывается впечатление, что это должно найти свое отражение в определении нашего класса формы, разве нет?

Марсиа: Безусловно. Не могу понять, почему там еще нет этого кода. Без какой-либо опасности в определении класса формы можно предусмотреть шаблонный метод OK2Destroy(), потому что, как бы там ни было, пустой метод всегда возвращает значение истины — True. Я добавлю этот метод в классы, когда буду в следующий раз обновлять свою библиотеку. Теперь как насчет последовательности событий для элемента управления grid?

Энди: Я думаю, с этим придется подождать до статьи в следующем месяце. Тогда мы рассмотрим как ведут себя события, предусмотренные для всех главных элементов управления, не возражаешь?

Энди Крамек (Andy Kramek) — опытный FoxPro-разработчик со стажем, имеет статус FoxPro MVP, является независимым контрактором и время от времени выступает как автор книг и статей. Родом он из Англии, но в настоящее время проживает в Акроне, шт. Огайо.
andykr@compuserve.com

Марсиа Акинз (Marcia Akins) имеет статус FoxPro MVP, является независимым консультантом и совладельцем фирмы Tightline Computers Inc., находящейся в Акроне, шт. Огайо. «Заслуженный» спикер многих конференций, она часто публикует свои работы и хорошо известна как активный участник форумов CompuServe и Universal Thread.
marciagakins@compuserve.com

❑❑❑

VFP 8: интервью с Кеном Леви

Клаудио Лассала, Мартин Сальяс (Claudio Lassala, Martin Salhas)

Только что на проходившей в Форт-Лодердейле конференции DevCon была представлена версия Visual FoxPro 8. Кен Леви (Ken Levy) сделал основной доклад и продемонстрировал множество замечательных возможностей этого нового продукта. Редколлегия электронного издания UTMag/RapoZine с удовольствием публикует в этом специальном выпуске своего журнала интервью с Кеном, которое было взято у него сразу же после выступления.

Учитывая тот факт, что реализованные в версии VFP 8 новые возможности обеспечивают более тесную интеграцию этого продукта с серверами SQL Server, .NET Server и множеством других широко распространенных продуктов фирмы MS, можем ли мы, как сообщество пользователей, придерживаться следующего мнения: в настоящий момент важно не стремиться купить или продать большее количество лицензий для VFP (что, как мне кажется, не способствует получению прибыли), но важно сформировать у наших заказчиков потребность в приобретении в большем количестве лицензий для, скажем, Windows, SQL Server и Exchange?

На самом деле, важно и то, и другое. Мы обнаружили, что многие VFP-разработчики до сих пор работают с версией VFP 6.0, так что пока еще наши действия направлены на поддержку перехода к версии VFP 7.0. Мы следим за тем, какие еще продукты фирмы Microsoft используют VFP-разработчики и их заказчики для эксплуатируемого приложения. Эту свою деятельность мы продолжим: как только в начале будущего года на рынке появится новая версия продукта, мы сосредоточимся на обновлении используемых версий VFP до версии 8.0 и будем добиваться этого путем широкого оповещения разработчиков о том, что версия 8.0 прекрасно работает с SQL Server 2000 и Visual Studio .NET, а оформление одного из вариантов подписки MSDN Subscription является наилучшим выбором для тех разработчиков, которые пользуются при создании программного обеспечения не только VFP. Использование новых версий Windows, например Windows XP, также важно с точки зрения наших усилий, и разработчики найдут в версии 8.0 много новых дополнений, которые поддерживают эту платформу, и другие продукты и технологии, предлагаемые фирмой Microsoft.

Теперь, когда мы можем обрабатывать наборы данных dataset, когда нам обеспечена поддержка протокола SOAP 3 и расширенная поддержка COM-технологии, интеграция с платформой .NET стала еще более тесной. Такое впечатление, что в роли инструмента для создания приложений среднего уровня VFP сейчас имеет даже более радужные перспективы. Известны ли Вам какие-либо реальные факты в подтверждение такого предположения, которыми Вы могли бы поделиться с нами?

Способ применения «номер один» — это использование этой СУБД для создания интеллектуальных клиентских настольных приложений, включающих, как правило, ввод данных, оперирующий таблицами VFP или данными, хранящимися на SQL Server. Число разработчиков, использующих VFP в качестве компоненты среднего уровня, растет и будет продолжать расти, особенно вследствие появления всех этих новых возможностей в версии VFP 8.0, включающих и расширенную поддержку стандарта XML, и более тесную интеграцию с SQL Server 2000. Средствами новых классов, XMLAdapter и CursorAdapter, можно оперировать данными в формате XML или иного источника, доступного по OLE DB или ODBC. Кроме того, в версии 8.0 существенно расширена поддержка XML Web services наряду с предоставлением новых инструментов для их использования, а это жизненно важно для общей стратегии фирмы Microsoft на базе платформы .NET, которая предусматривает взаимодействие данных, получаемых от различного программного обеспечения и разных устройств.

Значительно лучше стала документация. Не могли бы Вы сформулировать те цели, достигнуть которые планировалось в этой области, и оценить, насколько — по Вашему мнению — удалось приблизиться к решению поставленных задач?

Мы получили огромное количество откликов от членов VFP-сообщества, касающихся улучшения документации и справочной help-системы в версии VFP 8.0. В ходе этой работы мы тесно сотрудничали с обладателями статуса MVP в области VFP, и команда разработчиков непосредственно взаимодействовала с теми членами VFP-сообщества, которые занимаются разработ-

кой приложений средствами VFP, в особенности на форуме Universal Thread. Мы даже добавили в версию 8.0 возможность, обеспечивающую организацию обратной связи, позволяющую отослать комментарии прямо команде разработчиков VFP в фирму Microsoft. Разумеется, стимулом к реализации такой возможности по осуществлению обратной связи лежали полученные нами отклики пользователей.

Вам известно, какую большую заботу проявляет VFP-сообщество относительно продвижения этого продукта на рынке. Существуют ли какая-то новая стратегия или планы, направленные на то, чтобы уведомить тех, кто не принадлежит сообществу, о существовании такого замечательного инструмента?

Как только версия VFP 8.0 появится на рынке, Microsoft известит общественность об этом событии и предпримет дальнейшие усилия по маркетингу продукта. Хотя наша клиентская цель «номер один» — это уже сформировавшаяся для VFP база заказчиков, мы добавляем в версию VFP 8.0 такие новые возможности, которые будут в значительной мере способствовать привлечению к этому продукту внимания новых разработчиков. Кроме того, частью доклада, сделанного 29-го сентября на конференции разработчиков, была демонстрация новой видеозаписи одного из руководителей компании старшего вице-президента Эрика Руддера (Sr. Vice President Eric Rudder), который отвечает за формирование положительного имиджа средств разработки Microsoft на рынке программных продуктов. Эрик был главным архитектором VFP версии 3.0 и глубоко понимает и VFP как продукт, и VFP-сообщество. Представленные в этой видеозаписи комментарии Эрика относительно поддержки, предоставляемой VFP от Microsoft, предназначены не для одних лишь членов VFP-сообщества, и было бы замечательно, если бы «не-VFP» разработчики и ИТ-менеджеры услышали данные комментарии и узнали об отношении фирмы MS к этому продукту. Видеозапись с выступлением Эрика доступна для просмотра в режиме online по адресу http://gotdotnet.com/team/vfp/videos/Eric_Rudder_VFP8_56K.wmv.

Следует ли нам приступить к обдумыванию списка пожеланий для версии VFP 9?

В докладе, сделанном на конференции DevCon, мы также продемонстрировали демо-версию того, что последует за версией VFP 8.0; нечто такое, для чего у нас есть кодовое название Europa (по имени одного из спутников планеты Юпитер). Мы продемонстрировали несколько возможностей, которые, по нашему мнению, последуют за версией 8.0, но, принимая решение о том, что надо делать после выхода версии 8.0, мы, главным образом, будем ориентироваться на отклики пользователей, особенно на те из них, которые включены в формируемый на форуме UT список пожеланий.

Нас всегда впечатлял тот факт, что VFP является таким продуктом, который неизменно обладает возможностями, затребованными сообществом пользователей. С появлением версии 8 история продолжается (дополнения для конструктора View Designer, возможность добавлять свойства к «отдельным» объектам, класс Collection, автоинкрементные поля и так далее). Какова истинная причина того, что команда разработчиков Fox так замечательно откликается на пожелания сообщества?

Мы сознаем, что VFP-разработчики активно обеспечивают обратную связь, чтобы помочь нам развивать VFP наилучшим, насколько только это возможно, образом. Мы прислушивались к полученным откликам при проектировании версии 8.0 даже в большей степени, нежели при создании любой из предыдущих версий. У VFP-команды и многих beta-сайтов, с которыми мы взаимодействовали, сложилось прочное мнение, что VFP 8.0 — это самая впечатляющая и богатая возможностями версия продукта с момента появления версии VFP 3.0. Мы думаем, разработчики, использующие версию 7.0, а также те, кто использует версию 6.0, захотят выполнить обновление до версии 8.0 также, как захотят это сделать и те VFP-разработчики, которые до сих пор работают с FoxPro 2.x, но ищут способ для того, чтобы продвинуться вперед в технологии. Новые разработчики, которые только начали использовать VFP, обнаружат, что эта версия обеспечит им более быстрый старт, и что работать с ней легче, чем с любой из предыдущих версий, поскольку VFP 8 обладает такими новыми возможностями, как просмотр в редакторе исходного кода определений программных элементов (View Definition), набор новых построителей, исправленная и доработанная документация, новая панель Task Pane, аналогичная стартовой странице в среде разработки Visual Studio .NET, новая утилита Toolbox, предназначенная для организации элементов управления, и многие другие.

Помимо выполнения массы пожеланий, полученных от сообщества, команда разработчиков Fox включила в продукт множество новых вещей. Что из этого нового представляется наиболее интересным самим Fox-разработчикам?

Разные разработчики из VFP-команды работали над реализацией различных возможностей версии 8.0,

плюс к тому, многие из них являются опытными пользователями VFP и делают с помощью этого инструмента самые различные вещи. Следовательно, я должен сказать, что в VFP 8.0 нет какой-то одной возможности, которой отдавалось бы предпочтение. На этот вопрос не просто ответить, поэтому я скажу, что одной очень важной возможностью VFP является FoxPro-сообщество, которое абсолютно все участники VFP-команды, вплоть до высшего исполнительного руководства фирмы Microsoft, считают самым замечательным явлением — о чем вы услышите в видеозаписи рассказа Эрика Руддера о версии 8.0.

Ряд дополнений получила интегрированная среда разработки IDE. На первый взгляд, самыми замечательными из них являются Task Pane Manager и утилита Toolbox. Может ли разработчик наращивать возможности этих инструментов, также как он делает это применительно к утилитам Class Browser и Object Browser? Какие дополнения в эти инструменты внесли бы Вы сами как разработчик?

Был обновлен Table Designer: он был существенно дополнен и, кроме того, теперь он работает с автоинкрементными полями. Мы также в значительной мере расширили возможности View Designer с целью усовершенствовать управление внешними соединениями (outer join), обеспечить просмотр DBSetProp-данных в полученном SQL-запросе и реализовать в полном объеме двунаправленное редактирование настроек конструктора и сформированного им SQL-запроса так, что вы можете внести изменения прямо в предложение языка запросов SQL и получить автоматическое отображение внесенных изменений в конструкторе View Designer. Кроме того, появилась новая утилита Code References для расширенного поиска в исходном коде и других текстах из файлов, что-то вроде утилиты FilerX с более развитыми возможностями.

И все-таки, как разработчик, какое новшество, появившееся в версии 8, Вы считаете самым замечательным?

Я не особенно задумывался над этим вопросом и воспринимаю версию 8.0 единственно как коллекцию прекрасных новых возможностей, которые делают ее самой впечатляющей из всех версий VFP, выпущенных за очень длительный период времени. Что касается меня лично, то моей излюбленной возможностью, видимо, стал бы класс XMLAdapter. Этот класс в полном объеме поддерживает импорт и экспорт курсоров VFP и иерархических XML-данных, а также XML DiffGramm. Он позволяет организовать передачу данных между VFP-приложениями, передачу VFP-данных в приложения, построенные на базе модели ADO.NET, и обеспечивает работу со многими другими источниками данных, поддерживающими технологию XML, такими как приложения, созданные на базе другой платформы, различные устройства и так далее.

В версии 7 сообщество одобрило появление технологии IntelliSense. Что, как Вам кажется, получит наибольшее одобрение сообщества теперь?

Я не думаю, что предпочтение будет отдаваться какой-то одной выдающейся возможности, и полагаю, что каждый разработчик составит свое собственное мнение относительно лучшей из них, основываясь на типах создаваемых им приложений и на том, какой стиль разработки свойственен этому разработчику. Вот перечень некоторых новых возможностей версии 8.0, список которых «высвечивается» после ее инсталляции:

Продуктивность разработки

- Try..Catch..Finally — структурная обработка исключений.
- Code References — поиск символьных и прочих текстовых ссылок в ваших проектах.
- Collections — встроенная поддержка коллекций объектов.
- Event Binding — инициирование наступления события и связывание событий для встроенных VFP-объектов.
- Toolbox — настраиваемый инструмент для организации всех ваших излюбленных классов, элементов управления и служб XML Web services.
- Task Pane — «подручный» помощник в решении повседневных задач разработчика Data Enhancements.
- Auto Increment — новый табличный тип данных, который действует как счетчик для каждой новой записи.
- CursorAdapter — более простое взаимодействие с удаленными данными посредством нового класса курсора с дополнительными возможностями.
- Data Environment — возможность создания подкласса и повторного использования среды данных DE в классе формы.
- View Designer — новая возможность редактирования запроса в двух направлениях и улучшенная обработка для сложных запросов.
- Delayed Data Binding — простота манипулирования размещенными в форме элементами управления, связанными с данными.
- DBC Field Expressions — вы теперь можете определить выражение для заголовка поля.

Интеллектуальный клиент

- Windows XP Themes — придает вашим приложениям внешний вид и восприятие наимоднейшего пользовательского интерфейса UI ОС Windows XP.
- GDI+ Imaging — обеспечивает поддержку набора новых форматов изображений, включая анимированный формат GIF.
- Pageframes — смена ориентации вкладок страничного блока.
- Grids — настройка цвета для выбранной строки, автоматическое определение размеров столбцов, поддержка селектора в стиле listbox, использование изображений в заголовках и блокировка столбцов.
- Buttons — управление положением изображения, размещенного на командной кнопке, и задание цвета фона.
- Member Classes — возможность образования подклассов на базе классов Page, OptionButton, Column и Header.

Различные языковые и иные дополнения

- XMLAdapter — новый класс с поддержкой иерархических XML-структур и использованием формата diffgram.
- SCATTER NAME ADDITIVE, INSERT FROM NAME — обеспечение использования объектов в операциях вставки и обновления записей.
- DOCK WINDOW, ADOCKSTATE() — управление привязкой окон и инструментальных панелей в интерактивной среде разработки IDE.
- ADDPROPERTY(), REMOVEPROPERTY() — добавление и удаление свойств любого объекта.
- XML Web Services classes — публикация и потребление услуг служб XML Web services с легкостью работы с VFP-данными.
- Reporting — возможность объединять отчеты в цепочку, печать номеров страниц по трафарету “page x of y” и отказ от запоминания настроек рабочей среды для принтера.
- View Parent Code — возможность быстрого просмотра кода родительских классов из редактора методов.
- Menus — простота перемещения существующих панелей и пунктов меню в конструкторе.
- Более полная поддержка использования массивов в COM-модели — использование массивов в качестве параметров и возвращаемых значений при взаимодействии с COM-объектами.

Прекрасным дополнением является структурная обработка исключений. Это дополнение было инспировано платформой .NET?

Обработка ошибок с помощью команды Try/Catch/Finally — это структурная обработка ошибок, которая реализована во многих языках программирования платформы .NET, таких как Visual Basic .NET и C# .NET, но такая возможность имеется также и в других языках. VFP-команда проанализировала возможности, предоставляемые платформой Visual Studio .NET, и объединила их с запросами пользователей, изложенными в списке пожеланий, чтобы отдать приоритет тем возможностям, которые более всего необходимы в процессе разработки с использованием VFP. Новый механизм для обработки ошибок, реализованный в версии 8.0, вероятно, будет использоваться большинством VFP-разработчиков, поскольку он очень прост в применении, но обладает исключительно мощными возможностями и обеспечивает исключительно высокую продуктивность практически для всех сценариев структурированной логики.

Новый механизм обработки исключений удовлетворяет ту потребность, которая существовала всегда: он обеспечивает обработку ошибок в отчетах. Это здорово. К слову об отчетах, со времени появления версии 3 сообщество не перестает надеяться на реальное совершенствование механизма отчетов, например, объективизацию отчетов. Версия 8 по-прежнему не оправдывает наших ожиданий. То же самое относится к механизму создания меню. Существует ли некая особая причина (возможно, техническая), по которой эти компоненты продукта не могут быть подвергнуты «объективизации»?

Работая над версией 8.0, мы много внимания уделили реализации не только новых возможностей, обеспечивающих повышение производительности труда, но и возможностям, позволяющим разработчикам делать такие вещи, которые они вообще не могли делать прежде. В версии 8.0 предусмотрено много новых возможностей для конструктора отчетов, позволяющих управлять выходными данными, обеспечивающих нумерацию страниц по трафарету “x of y page”, и так далее. Новые дополнения, которые получил конструктор меню, включают более простую в применении операцию “copy and paste”. Как нам кажется, новые возможности версии 8.0 прекрасно сбалансированы, что, собственно, и делает этот продукт более мощным и расширяет сферу его применения как средства разработки.

Мы читали в ваших публикациях о том, что версия 8 лучше приспособлена для работы с платформой .NET и SQL Server. Не могли бы вы подытожить, в чем заключаются эти улучшения?

Главными являются новый класс XMLAdapter, обеспечивающий совместимость с объектной моделью ADO.NET, усовершенствованный провайдер OLE DB для интеграции с платформой .NET и новый класс CursorAdapter для доступа и управления удаленными данными, хранящимися, например, на SQL Server 2000. Имеется масса других более мелких деталей, рассказ о которых даст надлежащий ответ на поставленный вопрос, и я полагаю, информационные рассылки, статьи и так далее, посвященные версии 8.0, которые не замедлят появиться, глубже раскроют эту тему. Как нам кажется, платформа Visual Studio .NET является отличным инструментальным средством, дополняющим основной инструмент разработки — VFP 8.0, особенно это касается технологий ASP.NET и ADO.NET, используемых для создания средствами VFP Web-приложений.

Я уверен, что разработчикам понравится возможность определить и создать подкласс среды данных Data Environment. То же самое относится к новому классу CursorAdapter (замечательно!). Исключительно полезной является возможность взять данные из XML- или ADO-источника, а затем воспользоваться механизмом IntelliDrop в сочетании с RAD-подходом, чтобы облегчить себе работу по созданию форм. Но как долго еще фирма Microsoft будет поддерживать и/или дополнять объектную модель ADO, коль скоро теперь у нас есть модель ADO.NET (которая очень сильно отличается от «традиционной» модели ADO)? Я имею в виду, действительно ли VFP-разработчики могут сегодня рассчитывать на модель ADO?

Как и в случае с технологией ODBC, объектная модель ADO — это та технология, которая будет иметь поддержку в полном объеме в течение очень долгого времени, но в плане инноваций или развития этих технологий вы не увидите ничего нового. Технологии OLE DB и ADO.NET — вот то, что вы увидите в развитии с новыми и дополненными возможностями и способностями в будущем. Разработчики могут рассчитывать на существование таких технологий, как COM-объекты, в течение многих, многих лет, но новые возможности будут реализованы в новейших технологиях, о которых Вы упомянули. Даже реализованный в VFP новый класс CursorAdapter полностью поддерживает доступ к удаленным данным посредством технологии ODBC, невзирая на то, что, как мы предполагаем, объектная модель ADO в сочетании с технологией OLE DB будет наиболее распространенным решением по интеграции данных SQL Server 2000 и VFP.

Имеется очень большое количество новшеств, на которые обратят внимание наши конечные пользователи, например, дополнения, предусмотренные для элемента управления Grid, поддержка интерфейса GDI+, оформительских схем Windows XP Themes и так далее. Таков был замысел команды разработчиков, создававших эту версию продукта?

Это те возможности, реализовав которые, мы окажем VFP-разработчикам содействие в продаже созданных на базе VFP решений и обновлений их заказчикам и работодателям. VFP 8.0 — это не только значительное обновление версии, рассчитанное на VFP-разработчиков, и не только серьезная модернизация, адресованная тем конечным пользователям, которые работают с готовыми VFP-приложениями, но также отличное — ориентированное на данные — инструментальное средство разработки, позволяющее создавать всевозможные решения в области применения баз данных, особенно настольные (интеллектуальный клиент) приложения.

Что касается локализованных версий VFP: анализируя статистику по форуму Universal Thread за последние 12 месяцев, мы легко можем видеть, что часть сообщества, говорящая на португальском языке, становится все больше и больше. Вернется ли с выходом версии VFP 8 продукт, локализованный для Португалии, или будет предоставлена соответствующая версия runtime-модулей?

Возросшая активность сообщества не всегда напрямую связана с ростом продаж. Мы будем оценивать продажи локализованных версий VFP, также как и продажи по регионам, чтобы определиться в отношении наших действий. В настоящее время ничего нового, о чем можно было бы объявить по этому вопросу, нет.

Некоторые пользователи всегда просили предоставить для VFP trial-версию. Будет ли версия VFP 8 иметь trial-вариант?

Как я уже упоминал ранее, обсуждать маркетинговую политику в отношении VFP я могу только тогда, когда она претворяется в жизнь, и после завершения этого процесса. Следите за дальнейшей информацией по этой теме.

Мы находимся в состоянии ожидания грядущего события и очень надеемся на то, что и VFP-разработчики также ждут этого события с нетерпением!

❑❑❑

Потеря текста в элементах управления Text Box

Эдвард МакДермотт (Edward McDermott)

«Этот дурацкий компьютер опять потерял данные». Доводилось ли вам когда-либо слышать от пользователей подобную жалобу? Вы вежливо слушаете и наблюдаете как они вводят данные, а затем выбирают на инструментальной панели кнопку Save. Все данные прекрасно запоминаются, исключая последнее поле. Данные, которые пользователи ввели в это последнее поле, иногда теряются. Вы ведь терпеть не можете те ошибки, которые трудно воспроизвести? Эдвард Макдермотт отслеживает указанную ошибку.

Отслеживая трудно воспроизводимую проблемную ситуацию, я иногда заставляю пользователей выполнить ввод данных с помощью карандаша, чтобы я мог видеть, как они нажимают на клавиши. Операторы, работающие в обычном режиме, могут вводить по 60 слов в минуту. При такой скорости я не в состоянии уследить за ними. Не похоже, чтобы пользователи делали что-то особенное. Но, минуту. Эта проблема возникает только для тех полей, которые пользователь покидает, чтобы нажать размещенную на инструментальной панели кнопку Save.

Не в этом ли заключается вопрос? Вы пытаетесь исполнить программу сами. Данные потеряны. Хорошо, вы пытаетесь выполнить трассировку программы. Проблема исчезает. Вы исполняете программу без отладчика. Проблема снова здесь. Вы чертыхаетесь, кидаетесь предметами и повторяете попытку.

Вы организуете простую форму, на которой размещены только два текстовых поля. Затем вы конструируете всплывающее окно для выдачи на экран значений свойств Value и ControlSource элемента управления. Тест. Нажимаете кнопку, чтобы вызвать на экран всплывающее окно. Значения всегда одинаковы. Вы добавляете новые данные. Данные теряются; этого не может быть.

Вы размещаете серию команд DEBUGOUT в программном коде, обрабатывающем нажатие кнопки Save: часть этих команд предназначена для показа значения свойства value элемента управления, другие — для демонстрации значения свойства ControlSource. Код примерно такой:

```
debugout "----------- START of SAVE --------------"
debugout "Before Save"
debugout "ActiveControl value: "
debugout _screen.activeform.ActiveControl.value
debugout "ActiveControl controlsource: "
m.lcX = _screen.activeform.ActiveControl.controlSource
debugout &lcX
=tableupdate(.t.)
debugout "After Save"
debugout "ActiveControl value: "
debugout _screen.activeform.ActiveControl.value
debugout "ActiveControl controlsource: "
m.lcX = _screen.activeform.ActiveControl.controlSource
debugout &lcX
debugout "----------- END of SAVE --------------"
```

На этот раз у вас есть доказательство того, что данные не сохранились. Значение свойства Value отличается от значения свойства ControlSource текстового поля. В чем дело? При выполнении функции TableUpdate то значение, которое уже находилось в свойстве ControlSource (старое значение), будет сохранено. Бац — введенные пользователем данные «потеряны». Примите поздравления: в конце концов, вы только что получили доказательство того, что пребываете в здравом уме.

Потом вас осеняет. Почему эти значения отличаются друг от друга?

Проблема инструментальных панелей и фокуса

Первым делом необходимо усвоить, что Visual FoxPro не обновляет свойство ControlSource (присваивая ему то значение, которое вы ввели в элемент управления) до тех пор, пока вы не завершили работу с этим элементом управления. Точнее говоря, под «завершили работу» я подразумеваю тот момент, когда элемент управления потерял фокус. Наша проблема обусловлена тем фактом, что инструментальная панель никогда не получает фокус. Если у вас в форме есть кнопка Save, реализованная с помощью элемента управления command button, щелчок мышью по этой кнопке приведет к тому, что элемент управления text box потеряет фокус, а элемент управления command button получит фокус. Но щелчок мышью по кнопке, расположенной на инструментальной панели, работает иначе — фокус остается у покинутого элемента управления, несмотря даже на то, что исполняется код метода Click() кнопки, расположенной на инструментальной панели.

Давайте я изложу это иначе. Поскольку кнопка, принадлежащая инструментальной панели, не получает фокус, элемент управления не теряет фокус, поэтому его свойство ControlSource не обновляется, а его событие Valid никогда не наступает. Такая си-

туация не является пороком программирования или ошибкой, присущей VFP. Это намеренное проектное решение, которое позволяет вам проделывать ряд вещей, например таких, как:

- Прервать процедуру ввода данных, включая и ту, для которой метод Valid не сработал.
- Создать инструментальную панель или элемент меню, которые преобразуют текст в верхний регистр или шифруют его, или модифицируют выбранный в данный момент объект.

Однако, такое решение создает проблемы для процедуры сохранения, и прежде всего ту, на которую любезно указал пользователь.

Итак, все, что вы должны сделать, — это инициировать упомянутое изменение фокуса. Легко? Проблема в том, что VFP не дает вам простого способа, позволяющего прямо инициировать потерю фокуса для объекта. Взамен мы намерены прибегнуть к небольшой уловке, установив фокус для того объекта, с которым мы уже работаем. Чтобы некий объект получил фокус, другой объект должен потерять фокус, даже если это один и тот же объект. Следовательно, следующий фрагмент программного кода в методе Save, помещенный перед обращением к функции TableUpdate, принудит интересующий нас элемент управления потерять фокус, перемещая тем самым введенное значение в свойство ControlSource элемента управления в состоянии готовности к «передаче на хранение».

```
* "Сбрасывание" значения в свойство controlsource
_screen.activeform.ActiveControl.setfocus()
```

Чтобы увидеть, как работает такое решение, исполните форму DEMONE.SCX, которая находится в файле-приложении к этой статье на сопровождающей дискете, выделив в ней только верхний флажок. Введите значение в текстовое поле Amount, только не убирайте из него курсор. Затем щелкните по расположенной на инструментальной панели кнопке Save. (Временно я изъял код для кнопки Save из класса формы и поместил его непосредственно в метод Save этой формы так, чтобы мы могли видеть, что происходит.) Окно Debug Output демонстрирует, что значение свойства value текстового поля — это новое значение, но в свойстве ControlSource по-прежнему хранится старое значение. Если вы перейдете к следующей записи, а затем вернетесь обратно к исходной записи, то увидите, что ваши изменения не сохранились. Ах, ну разумеется: мы же еще не выполнили функцию tableupdate. Но даже переход от одной записи к другой не фиксирует значение, введенное в текстовое поле.

Далее снова запустите форму на исполнение, выделите флажок Include tableupdate, повторите ту же самую процедуру внесения изменений в поле Amount и щелкните мышью по кнопке Save, размещенной в инструментальной панели. Вы снова увидите, что значение свойства ControlSource не было заменено на значение, введенное в текстовое поле, следовательно, функция tableupdate записала в таблицу исходное значение.

Теперь исполните форму в третий раз, но выделите флажок Include setfocus (в дополнение к первым двум флажкам). Вы увидите в окне Debug Output, что значение свойства ControlSource обновилось и равно значению, введенному в текстовое поле. Таким образом, при выполнении функции tableupdate в таблицу попадает корректное значение.

Проблема проверки допустимости данных

Так, пока все хорошо. Но теперь мы должны рассмотреть вторую проблему. Что, если в методе Valid текстового поля Amount у нас есть код для проверки допустимости данных?

Если вы просто возвращаете значение false из своего кода проверки допустимости данных, вот так:

```
if this.Value > 500
  wait window nowait "Amount cannot be larger than 500"
  return .f.
endif
```

то обнаружите, что сообщение выдается, но в таблице по-прежнему сохраняется неверное значение.

В чем тут дело? Когда элемент управления теряет фокус, исполняются два метода — метод Valid и метод LostFocus. Но мы выяснили, что элемент управления text box не теряет фокус, следовательно, метод Valid не исполняется до тех пор, пока мы не обратимся к методу setfocus. Однако метод setfocus будет исполняться независимо от того, успешно или нет завершилась работа метода Valid. Похоже, нас надули, не так ли?

Решение заключается в следующем: во-первых, прежде чем разрешить исполнение трюка с обращением к методу setfocus, необходимо убедиться в том, что поле успешно «прошло» свою процедуру проверки допустимости данных. Как это сделать? Вы исполняете метод Valid и сами проверяете полученный результат. Исполнение метода Valid обеспечит тестирование значений, в случае необходимости выдаст сообщение об ошибке и вернет результаты. Какие результаты? Метод Valid может возвращать значения True/False или число. В результате, код метода Save начинается следующим образом:

```
if !_screen.activeform.ActiveControl.valid()
  wait window "Validation failed so no save yet"
```

Что происходит при нажатии на клавишу Tab в форме?

Когда вы с помощью клавиши tab переходите в форме от текстового поля Text1 к текстовому полю Text2, происходят следующие события:

- 1. Свойство ControlSource текстового поля Text1 обновляется и получает то значение, которое появляется на экране — Text1.value. (Свойство ControlSource могло бы быть глобальной переменной, значением свойства, значением другого размещенного в этой форме элемента управления или полем таблицы. В последнем случае мы, в действительности, говорим о значении, находящемся в буфере поля.)
- 2. Инициируется метод Valid текстового поля Text1.
- 3. Инициируется метод When текстового поля Text2.
- 4. Исполняется метод LostFocus текстового поля Text1.
- 5. Инициируется метод GotFocus текстового поля Text2.

Метод When — это метод-«привратник». Он исполняется перед тем, как объект получает фокус. Если бы метод When вернул значение .F., управление было бы передано следующему в определенном для формы порядке обхода объекту. И тогда исполняется метод When объекта, получившего управление. (Да, вы можете создать ситуацию, когда все объекты «уклоняются» от исполнения метода When, и программа «зацикливается».)

Метод GotFocus позволяет вам добавить функциональность после того, как метод получил фокус, но до того, как пользователь получит возможность что-либо сделать. Не забудьте, метод GotFocus не срабатывает до тех пор, пока метод When этого объекта не вернет значение .T..

Метод Valid срабатывает при попытке элемента управления покинуть объект. Он может возвращать логическое или числовое значение. Если метод Valid возвращает значение False, на экран будет выдано сообщение об ошибке, и фокус останется в этом элементе управления. Если метод Valid возвращает числовое значение, отличное от нуля, на экран будет выдано сообщение об ошибке («Invalid Input» или результаты работы метода ErrorMessage), и фокус будет перемещаться в обратном (для отрицательных чисел) или прямом (для положительных чисел) направлении для того количества объектов, которое определено этим значением.

Метод Valid срабатывает раньше метода LostFocus. Метод LostFocus позволяет вам добавить функциональность после того, как пользователь ввел допустимое значение в объект. Например, если оператор ввел допустимый номер кредитной карты, создается экземпляр объекта, обрабатывающего кредитную карту.

```
  return .f.
endif
* продолжение, остальной код метода Save
```

Я исходил из предположения, что все эти действия выполняются до того момента, как инициируются какие-либо процедуры проверки допустимости на уровне формы (как можно быстрее) так, что вы «запускаете» событие Valid объекта и выполняете процедуры проверки допустимости на уровне полей.

Итак, я рассказал о кнопке, которая размещена на инструментальной панели и используется для сохранения данных. Это та ситуация, в которой наиболее вероятно возникновение рассматриваемой проблемы. Однако, такой подход необходимо применять к любому методу, исполнение которого инициируется из инструментальной панели или пункта меню и который требует обновления свойства ControlSource. Можно было бы рассмотреть работу любого метода, который обновляет базу данных или использует функцию GetFldState(). (Поскольку свойство ControlSource не обновляется, состояние поля не установлено, и такая функция вернет не то значение, которое вы ожидаете.) В результате, у вас может возникнуть необходимость вставить в метод вышеуказанный код.

Что ж, я, конечно, могу представить себе тех, кто разрабатывает программный код, в котором используются преимущества такой уловки. Подсказка: подумайте о том, как рассмотренные особенности сказываются на работе таких возможностей меню, как команды Cut, Copy и Paste. Надеюсь на открытия.

Тестирование результатов, возвращенных процедурой проверки допустимости

Создайте форму с контейнером и кнопкой. Поместите следующий программный код в метод Click командной кнопки и нажмите эту кнопку.

```
local llTest, lnresult

ThisForm.Container1.SetFocus()
lnresult = Thisform.ActiveControl.valid()
Wait window "Result from run on activecontrol " ;
  +  vartype(lnresult))
Wait Window "lnresult Result contains " ;
  +  iif(lnresult,"True", "False"))
llTest = ThisForm.Container1.valid()
Wait window "Result from run explicit " +vartype(llTest))
```

Вы обнаружите, что lnresult — это логическая переменная со значением true, и вы не получите никаких ошибок. С другой стороны, явная ссылка на событие valid контейнера Container1 приведет в результате к ошибке. VFP должна обладать некоторым интеллектом, «встроенным» в свойство ActiveControl.

Эдвард МакДермотт занят в компьютерной индустрии более 20 лет, последние 15 из них он работает с ПК, применяя такие языки программирования как C++, Delphi, Powerbuilder, MS Access и Visual FoxPro.
Его адрес: edmcdermott@gosympatico.ca

❑❑❑

Программное управление библиотеками классов в VFP

Чарльз Томас Бланкеншип (Charles Thomas Blankenship)

Программное управление VCX-файлами открывает окно в новый мир автоматизированных возможностей. В этой статье Чарльз Томас Бланкеншип объясняет, как добавить код метода и изменить свойства объекта прямо внутри библиотеки классов VFP (VCX-файлы).

Не знаю как вы, но я изнываю от скуки, связанной с построением приложений VFP с нуля. Даже когда используешь производительные среды исполнения, типа Mere Mortals и Visual FoxExpress, найдется тонна нудной работы по простому созданию бизнес-объектов, представлений, объектов интерфейса, списков прокси, форм, меню и т. д. После разработки множества приложений, основанных на учебниках по кодированию, я, в конце концов, поддался на уговоры и написал расширение к среде разработки Mere Mortals, позволяющей мне помещать драйверы типа GenScreenX в поля комментариев моей Xcase-модели (таблицы и поля). Теперь достаточно нажать кнопку и автоматически получить все объекты уровня приложения. Для решения этой задачи было необходимо изучить, как программно создавать новые подклассы и вносить в них код метода, а также как управлять значениями свойств для вновь созданных определений классов.

Основной процесс требует выполнения следующих задач: 1) создать экземпляр нового класса; 2) изменить его свойства; 3) вставить код метода и 4) сохранить этот новый класс, — к сожалению, необязательно в этом интуитивно понятном порядке. На всем протяжении этой статьи я использовал элемент интерфейса TextBox в качестве примера объекта, поскольку он, возможно, один из наиболее часто используемых элементов при построении приложений VFP.

Создание нового объекта интерфейса

Первым шагом в этом процессе является создание экземпляра поля ввода для отображения имени пользователя. Для этого примера давайте допустим следующее: 1) имя столбца, хранящего имя пользователя, будет v_Person.cPersonFirstName; 2) определения классов для всех элементов интерфейса хранятся в библиотеке классов, называемой aInterfaceControls.VCX; 3) имя элемента — txtPersonFirstName; 4) имя родительского класса, используемого для создания нового поля ввода, — aTextBox. Следующий код создает экземпляр нового объекта интерфейса:

```
LOCAL loTemplateObject
loTemplateObject = CREATEOBJECT("AtextBox")
```

Обратите внимание, что простое использование определения класса aTextBox для создания временного объекта, будет автоматически определять родительский класс нового объекта.

Изменение свойств нового объекта

Итак, loTemplateObject содержит ссылку на то, что станет объектом интерфейса txt_PersonFirstName. Модернизация свойств этого объекта для изменения его имени и определения свойства ControlSource выполняется так:

```
loTemplateObject.Name = 'txt_PersonFirstName'
loTemplateObject.ControlSource = ;
   'v_Person.cPersonFirstName'
```

Заметьте, что попытка изменить свойство ControlSource привела к появлению сообщения об ошибке.

Вследствие того, что loTemplateObject является активным объектом (мы достигаем это программными средствами, другими словами, он является живым, а не записанным на Memorex), ввести неправильное значение для свойства ControlSource во время исполнения невозможно. Значение v_Person.cPersonFirstName неверно потому, что во время присваивания представление v_Person не является активным для сеанса данных. Данное ограничение распространяется и на все другие свойства этого объекта, открытого в период исполнения только на чтение и на чтение/запись на стадии разработки. Так что же делать? Прежде всего, я сохраню новый объект в файле библиотеки класса. После чего смогу напрямую управлять этими свойствами и методами в самом файле библиотеки класса.

Сохранение нового объекта (создание нового определения класса)

Сохранение нового объекта, что, по сути, является созданием определения нового класса объктов, достигается использованием собственного метода объ-

Таблица 1. Содержание записи в файле библиотеки класса, включающей определение класса для txt_PersonFirstName.

Поле	Значение	Содержание
Class	atextbox	Имя родительского класса, используемого для создания объекта.
Class	Loclibs\acontrols.vcx	Расположение и имя библиотеки родительского класса.
Base	Classtextbox	Имя базового класса VFP, используемого для создания объекта.
ObjName	txt_personfirstname	Имя вновь созданного объекта.
Properties	Name = "txt_PersonFirstName"	Имя и значение для любых свойств, не являющимися значениями по умолчанию.
Methods	<пусто> — ненадолго	Любой код метода, связанный с объектом.

екта SaveAsClass(). Синтаксис данного метода выглядит так:

```
Object.SaveAsClass( ClassLibName, ClassName ;
   [, Description] )
```

где:

```
ClassLibName
 Определяет имя .vcx-файла, хранящего
 определение класса.

ClassName
 Определяет имя, назначенное классу.

Description
 Содержит необязательное описание этого класса.
```

Код, создающий новый объект интерфейса, выглядит так:

```
loTemplateObject.SaveAsClass("aInterfaceControls.VCX",;
                             "txt_PersonFirstName",;
                             "The person's first name" )
```

Проверка жесткого диска показывает, что метод SaveAsClass() создает новый файл библиотеки класса с именем aInterfaceControls.VCX. Именно с этим файлом связан последний шаг нашего процесса.

Программное обновление свойств в VCX-файле

Сегодня почти каждый знает, что файлы библиотеки класса в Visual FoxPro представляют собою VFP-таблицы с расширением .VCX. Открыть их также просто, как применить команду USE с определенным полным именем файла. Следующая команда открывает вновь созданную библиотеку класса:

```
USE aInterfaceControls.VCX
```

Простая команда BROWSE показывает структуру этой таблицы, изображенную на рис. 1.

Ainterfacecontrols

	Platform	Uniqueid	Timestamp	Class	Classloc	Baseclass	Objname	Parent	Properties	Protected	Methods
	COMMENT	Class		memo	memo	memo	memo	memo	memo	memo	memo
▶	WINDOWS	_0NZ0STJFU	749300568	Memo	Memo	Memo	Memo	memo	Memo	memo	memo
	COMMENT	RESERVED		memo	memo	memo	Memo	memo	Memo	memo	memo

Рис. 1. Окно Browse библиотеки класса aInterfaceControls.VCX.

Теперь я собираюсь изменить свойства объекта и сделаю это в коде. Однако, прежде чем любое изменение будет выполнено, я нахожу запись, которая содержит определение класса txt_PersonFirstName. Следующая команда выполняет эту задачу:

```
LOCATE FOR UPPER("txt_PersonFirstName") $ ;
           UPPER(ainterfacecontrols.objname) AND ;
           .NOT. EMPTY( timestamp )
```

Теперь, когда я нашел соответствующую запись в файле библиотеки класса, содержащую определение класса для txt_PersonFirstName, я могу начать модифицировать его. Содержание записи показано в таблице 1.

Обратите внимание, что свойство Name в столбце Properties библиотеки класса представлено так же, как оно записано в окне свойств. Следовательно, предоставлять значение, не заданное по умолчанию для свойства ControlSource, так же просто, как выполнить следующее:

```
lcCurrent = ALLTRIM( aInterfaceControls.Properties)
lcCurrent = [ControlSource = "v_Person.cPersonFirstName"] ;
   + CHR(13) + lcCurrent
REPLACE aInterfaceControls.Properties WITH lcCurrent
```

Быстрый взгляд на окно свойств этого объекта подтверждает, что приведенный выше код должным образом заполняет свойство ControlSource. Чтобы это работало, я должен удостовериться, что орфография этого свойства правильна. Лучшим способом выполнения этой задачи будет ручное заполнение свойств объекта, а затем проверка файла библиотеки класса с целью удостовериться, как VFP ввел имя этого свойства... и далее следовать процессу в моей собственной программе (см. рис. 2).

Программное обновление методов в файлах VCX

Кто-то может подумать, что внедрение кода для метода объекта так же просто выполнить, как и для свойства. Мой ответ таков: и да, и нет. Простое программное размещение следующего кода в столбец Methods, к сожалению, не приводит к желаемому результату (если посмотреть на окно свойств объекта):

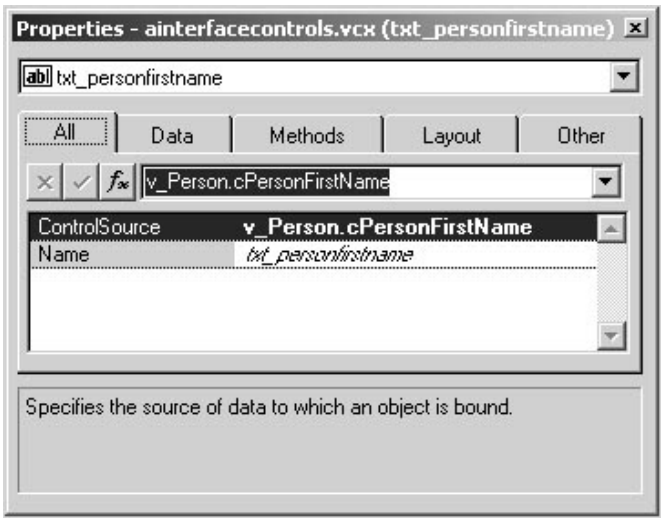

Рис. 2. Иное, чем принятое по умолчанию, значение свойства для txt_ PersonFirstName с заполненным ControlSource.

```
PROCEDURE Init
 THIS.AddViewParameter( "vp_iID", ;
   "vp_iID = v_PersonProxy.iID", "v_Person" )
ENDPROC
```

Результаты использования этого кода различались. Иногда я видел усеченное сообщение; в другой раз я не видел никакого кода метода в окне свойств вообще. Чтобы решить эту проблему, перекомпилируйте библиотеку класса после ее очистки. Приведенный ниже код гарантирует, что я увижу код метода в окне свойств таким, каким я ожидаю видеть его, если введу его вручную в среде разработки VFP.

```
CLEAR CLASSLIB aInterfaceControls.VCX
COMPILE CLASSLIB aInterfaceControls
```

Обратите внимание на то, что код метода, размещенный в столбце Methods, разделен командами PROCEDURE ... ENDPROC. Так VFP отделяет один блок кода метода от другого. Я строго придерживаюсь этого формата, поэтому мой сгенерированный программно код метода выглядит в окне свойств должным образом.

Выводы

Итак, теперь у вас есть возможность достаточно просто программно создать определение визуального класса, заполнить его свойства значениями, не являющимися значениями по умолчанию, и ввести код в эти методы. Эта техника уменьшает количество монотонной работы по созданию элементов пользовательского интерфейса и других приложений уровня объектов при помощи программной идентификации их родительских классов и задания свойств и методов с помощью извлекаемой информации. Вы ограничены только рамками вашего воображения.

Чарльз Томас Бланкеншип является директором по разработке программного обеспечения компании Claims Verification Inc., расположенной в уютной, солнечной и теплой южной Флориде. Он бывший работник компании Flash (ныне канувший в лету тип управленца Flash Creative), ежеминутно стенающий о том, что лучшие компании в мире, для которых стоило работать в качестве консультанта, ушли навсегда, вместе с их великими боссами (Яр Алан Гривер (Yair Alan Griver) и Дэвид Блументаль (David Blumenthal)). cblankenship@cvi.com

❑❑❑

FoxTalk *русское издание*

Печатается ежемесячно

Учредитель и издатель:

ООО Эдэль.

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

Подписано в печать 10/04/03. Формат 60x90 1/8. Тираж 330 экз.