FoxTalk

Январь 2003
№ 1 (67)

русское издание

Solutions for Microsoft® FoxPro® and Visual FoxPro® Developers

Публикация вашей первой службы Web Service, часть 2

Уилл Хентцен (Whil Hentzen)

В прошлом месяце мы создали свою первую службу Web Service, опубликовали ее и воспользовались ее услугами — все это мы проделали на инструментальной машине. Пользы от этой службы Web Service немного, но не в этом ее назначение. Она предназначена для того, чтобы продемонстрировать вам, какие шаги необходимо предпринять для ее создания, где взять соответствующие инструменты и на какую «секретную кнопку» надо нажать. В заключение я обещал вам, что в этом месяце мы введем эту службу Web Service в эксплуатацию на рабочем сервере, но по прошествии времени я изменил свое решение. После неоднократного повторения процесса создания служб Web Servicies возникают неизбежные вопросы. Эта статья отвечает на многие из таких вопросов и готовит нас к серьезному делу в следующем месяце — вводу в эксплуатацию на рабочем сервере.

В этом месяце я собираюсь дать ответы на вопросы, относящиеся к изучаемому процессу, дабы объяснить, что делается за кулисами и какими возможностями вы располагаете. Очень многие из этих сведений не нашли бы своего применения во время нашего первого опыта, но теперь, когда вы увидели весь процесс, вы сможете найти подходящий контекст для применения некоторых из этих более сложных возможностей. Как только мы закончим отвечать на вопросы, вы обретете большую уверенность в обращении с основными понятиями, и тогда мы будем готовы к вводу службы в эксплуатацию — да, в следующем месяце.

«Правильные» имена

Прежде всего, давайте поговорим о выборе имени для класса службы Web Service.

```
o=createobject("wsc.hwpclass")
? o.getnews()
```

Такая запись означает, что wsc — это проект, hwpclass — это имя класса, а GetNews — это метод данного класса. Наилучшее выработанное практикой правило, применяемое для выбора имен программных идентификаторов компонентов (ProgID), гласит, что про-

Январь 2003

материал имеет отношение к соответствующей версии

материал имеет отношение к соответствующей платформе

исходные тексты программ можно скачать из Интернета

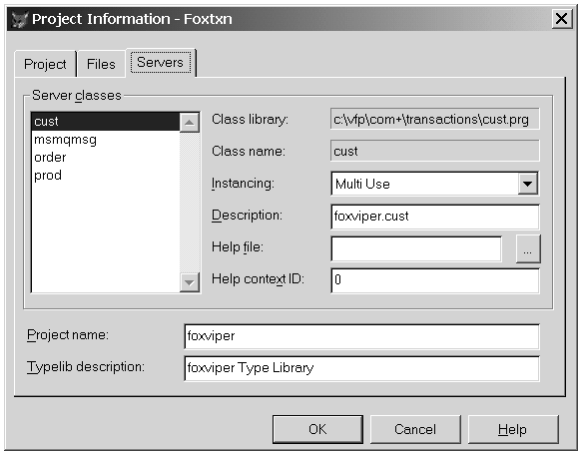

Рис. 1. Дайте COM-компоненту подходящее имя.

граммные идентификаторы должны формироваться по следующему шаблону: company.project.class. Следовательно, данный программный идентификатор следовало бы определить вот так:

```
Hentzenwerke.wsc.hwpclass
```

В действительности, такой идентификатор мало что нам говорит, поскольку его составные элементы — "wsc" и "hwpclass" — это имена-пустышки, взятые для данного примера. С учетом того, что нашей целью была «добыча» новостей, возможно более подходящее имя должно было бы включать название проекта FoxTalkDemo, а сам класс следовало бы назвать NewsServices. Тогда программный идентификатор выглядел бы следующим образом:

```
Hentzenwerke.FoxTalkDemo.NewsServices
```

Итак, как вам это сделать? Вы используете вкладку Servers в диалоговом окне Project Information (см. рис. 1).

Регистрация DLL-библиотеки

Когда вы компилируете DLL-библиотеку, используя диалоговое окно Build Options (см. рис. 3 в статье, опубликованной в прошлом месяце), вы не ограничиваетесь одним лишь созданием файла с расширением .DLL. Вы также регистрируете эту DLL-библиотеку в реестре Windows. Таким образом, когда другая программа получает вот такой запрос на создание объекта:

```
o=createobject("wsc.hwpclass")
```

или (на языке программирования Visual Basic):

```
Dim o As New wsc.hwpclass
```

то для поиска в реестре Windows соответствующего идентификатора класса CLSID используется программный идентификатор "wsc.hwpclass". Идентификатор класса CLSID нужен для того, чтобы узнать полный путь доступа к DLL/EXE-модулю и связанной с ним библиотеке типов. Библиотека типов затем используется для определения интерфейса.

Вы можете компилировать свой класс снова и снова, если вам так хочется, и каждый раз будет использован один и тот же программный идентификатор: запись в реестре Windows просто будет обновляться. Однако, если вы внесли изменения в интерфейс, вам следует выставить флажок в независимом переключателе Regenerate Component IDs. Вам необходимы новые: программный идентификатор ProgID и идентификатор класса CLSID, потому что вам не хотелось бы, чтобы пользователь создал экземпляр объекта новой версии этого компонента (с новым интерфейсом), используя старые ProgID и CLSID. Представьте только, как трудно было бы это отлаживать!

Публикация службы Web Service

Впрочем, одного лишь создания DLL-библиотеки недостаточно для того, чтобы превратить ее в службу Web Service. Это то, что логики называют «необходимым, но недостаточным» условием. Служба Web Service — это DLL-библиотека, над которой еще немного поработали, так сказать. Мы использовали принадлежащий Visual FoxPro построитель Web Services Publisher для публикации DLL-библиотеки, как службы Web Service. Построитель Web Services Publisher делает следующее:

- Обязательно создает файлы WSDL и WSML.
- Опционально регистрирует службу с тем, чтобы о ней было известно технологии IntelliSense.
- Опционально инсталлирует надстройку (project hook) к VFP-проекту с тем, чтобы при последующих компиляциях этой DLL-библиотеки автоматически вызывался и исполнялся построитель Web Services Publisher.

В принципе, файлы WSDL и WSML аналогичны библиотекам типов в языках программирования VFP и VB. Другими словами, они сообщают внешнему миру о том, какими методами располагает данная служба Web Service.

WSDL является аббревиатурой для «Web Services Description Language» (язык описания служб Web Services) — это XML-документ, в котором описано, какие услуги предоставляет работающая на сервере служба Web Service. Это что-то вроде меню. WSDL-файл определяет также формат, которому

должен следовать клиент, обращаясь к службе с запросом.

WSML — это аббревиатура для «Web Services Meta Language» (метаязык служб Web Services). Этот файл относится к протоколу SOAP 2.0, и в нем представлена информация, необходимая для установления соответствия между операциями описанной в WSDL-файле службы и конкретными методами из DLL-библиотеки. WSML-файл определяет, какой COM-объект следует загрузить, чтобы предоставить запрашиваемую службу Web Service.

Вы можете получить дополнительную информацию об этих файлах, намного больше сведений, чем вы когда-нибудь захотите узнать, обратившись к справочному Help-файлу для протокола SOAP 2.0, размещенному, вероятно, в каталоге Program Files\MSSOAP\SOAP.CHM.

Впоследствии, когда вы станете потребителем службы Web Service, вы обратитесь к WSDL-файлу, который сообщит вам: законен ли способ вашего обращения к службе Web Service или нет. Например, если в службе Web Service есть метод, который называется GetNews, тогда в WSDL-файле хранится следующая информация:

```
<portType name='hwpclassSoapPort'>
   <operation name='getnews' parameterOrder=''>
     <input message='wsdlns:hwpclass.getnews' />
     <output message='wsdlns:hwpclass.getnewsResponse' />
   </operation>
```

Если вы попытаетесь воспользоваться этой службой Web Service, обратившись с запросом к методу GetNewsOfTomorrow, это обращение потерпит неудачу, поскольку WSDL-файлу ничего не известно о методе GetNewsOfTomorrow.

Выбор класса

Когда вы публикуете свою службу Web Service, вы используете входящий в состав Visual FoxPro построитель Web Services Publisher (см. рис. 8 в предыдущей статье). В моем примере комбинированный список Select Class заблокирован. Почему использован элемент управления combobox, и почему он заблокирован?

В COM-компоненте может быть несколько классов, объявленных как OLEPUBLIC, но служба Web Service может опубликовать только один из этих классов. (Разумеется, каждый из этих классов может открыть любое количество методов.) Элемент управления combobox используется для того, чтобы выбрать тот класс, который вы хотите опубликовать для своей конкретной службы Web Service. Если вы хотите опубликовать несколько классов, необходимо создать еще одну службу Web Service.

Сохранение настроек

Настройки, выполненные в диалоговом окне дополнительных настроек, запоминаются и будут использованы для последующих публикаций данной службы Web Service. Эти настройки хранятся в файле FOXWS.DBF, местонахождение которого указано в системной переменной памяти _FOXCODE.

Виртуальный каталог

В диалоговом окне дополнительных настроек (см. рис. 9 в предыдущей статье) при указании путей доступа для WSDL-файла и «слушателя» (listener) используется имя виртуального каталога, который использует данный Web-сервер.

«Стоп, — говорите вы. — Виртуальный каталог?» Да, я сказал «виртуальный каталог». И даже если вы уже имели дело с такой вещью, как виртуальный каталог, тем не менее, у вас может возникнуть необходимость продолжить чтение. Дело вот в чем.

У меня есть сервер IIS, настроенный так, что «домашним» является каталог E:\Inetpub\WWWRoot. Но на рис. 9, опубликованном в прошлом месяце, показано, что WSDL-файл (и, по-видимому, WSML-файл) находится в каталоге http://foxdotnet/webpub. Мне хотелось бы понимать это так, что физически WSDL- и WSML-файлы находятся в каталоге E:\INETPUB\WWWROOT\WEBPUB. Однако, когда я публиковал свою службу Web Service, построитель поместил эти файлы в каталог C:\INETPUB\WEBPUB, иначе говоря, куда-то еще. Поначалу я был совершенно обескуражен.

Оказалось, что когда вы впервые запускаете построитель Web Services Publisher, автоматически создается виртуальный каталог. Этот виртуальный каталог является семафором, который указывает куда-то еще. Таким образом, в адресе http://FOXDOTNET/webpub слово FOXDOTNET обозначает корневой каталог сервера IIS на моей машине, а “webpub” — это виртуальный каталог, который отображается на конкретный физический каталог, находящийся на этой же машине. В рассматриваемом случае, виртуальный каталог “webpub” отображается в каталог C:\INETPUB\WEBPUB. Обратите внимание на то, что этот виртуальный каталог вовсе не обязан ссылаться непременно на такой адрес, который находится в близком соседстве с корневым каталогом сервера IIS. Вот почему WSDL- и WSML-файлы, в конечном итоге, попали в каталог C:\INETPUB\WEBPUB, а не в каталог E:\INETPUB\WWWROOT\WEBPUB.

Как вам создать виртуальный каталог, если вы этого еще не сделали? Или что, если вы захотите его изменить? Прежде всего, откройте узел Internet

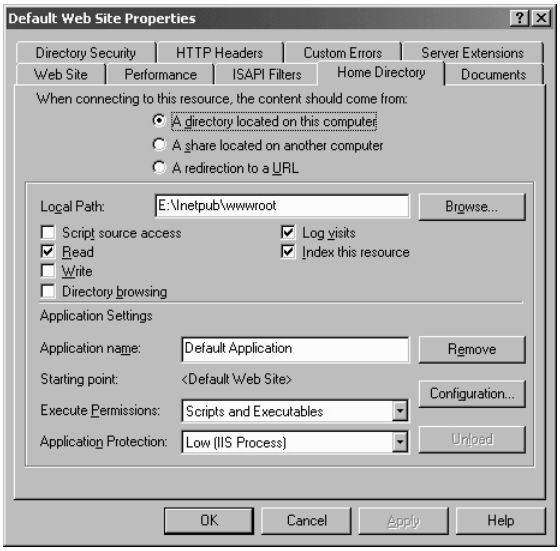

Рис. 2. Вкладка Home Directory диалогового окна Properties.

Information Services в апплете Computer Management панели управления. Щелкните правой кнопкой мыши по узлу Default Web Site и выберите в меню пункт Properties. Щелкните мышью по вкладке Home Directory, чтобы увидеть на экране свойства «домашнего» каталога — Home Directory, — как показано на рис. 2. Это тот «домашний» каталог, с которым работает IIS.

Для того, чтобы создать виртуальный каталог, физически расположенный где-то еще, щелкните правой кнопкой мыши по узлу Default Web Site и выберите в меню пункт New | Virtual Directory (на экране появится приветственный экран мастера Virtual Directory Creation Wizard), а затем нажмите кнопку Next. В открывшемся диалоговом окне введите в отведенное для этого поле имя виртуального каталога. Например, входящий в состав VFP построитель Web Services Publishing использует для создаваемого им виртуального каталога имя “WebPub”.

Потом укажите тот каталог, куда будет отображаться виртуальный каталог, другими словами, физический каталог, расположенный на вашей машине. Вы можете выбрать этот каталог в режиме просмотра (browse) или просто ввести его путь доступа в текстовое поле. После этого вам необходимо сконфигурировать права доступа. Не беспокойтесь, если вы не сможете корректно определить все эти права, я сейчас покажу, где можно будет внести исправления.

После того, как будет нажата кнопка Next в диалоге Access Permissions, появится диалоговое окно Finished — и все готово. Результат — новый виртуальный каталог — будет виден в дереве Default Web Site апплета Computer Management. Теперь вы можете взглянуть на свойства этого виртуального каталога и изменить их значения, щелкнув правой кнопкой мыши по соответствующему узлу в апплете Computer Management и выбрав в меню пункт Properties. Появится диалоговое окно Properties, и можно будет видеть независимые переключатели, определяющие права доступа: они расположены в средней части этого диалогового окна.

ISAPI-«слушатель» против ASP-«слушателя»

Еще одна опция, которую вы видите в диалоговом окне дополнительных настроек построителя Web Services Publisher, — это выбор «слушателя» (listener). Что такое «слушатель»? «Слушатель» — это тот механизм (физически это DLL-библиотека), который обрабатывает поступающие на данный сервер запросы к протоколу SOAP. У вас есть два варианта для выбора SOAP-«слушателя»: Internet Server API (ISAPI) и Active Server Pages (ASP).

Если вы используете ISAPI-«слушателя», используется библиотека SOAPISAP.DLL. Если вы делаете свой выбор в пользу ASP-«слушателя», тогда указанная ASP-страница задействует для обработки полученного запроса ASP-сценарий.

В файле Web Services Description Language (WSDL) URL-указатель, объявленный как обработчик SOAP-запросов, определяет способ обработки SOAP-запроса на сервере. Например, в следующем фрагменте WSDL-файла задается URL-указатель, который инициирует ISAPI-«слушателя»:

```
<definitions>
 ...
   <service name='DocSample1' >
     <port name='DocSample1PortType'
       binding='tns:DocSample1Binding' >
       <soap:address
           location='http://localhost/DocSample1Test/
             DocSample1.wsdl' />
     </port>
   </service>
 ...
 </definitions>
```

Если бы вместо этого URL-указатель определял ASP-файл, это выглядело бы следующим образом:

```
'http://localhost/DocSample1Test/DocSample1.asp'
```

И тогда этот URL-указатель инициировал бы указанный ASP-сценарий. Проблема заключается в том, что VFP, по умолчанию, пытается зарегистрировать службы Web Services для работы с ISAPI-«слушателем». Однако, в типичной конфигурации сервера IIS «слушатель», реализованный в библиотеке SOAPISAP.DLL, не используется, и чтобы он рабо-

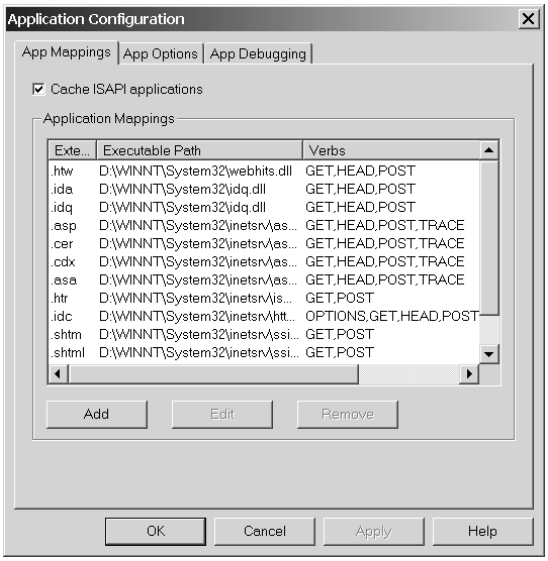

Рис. 3. В диалоговом окне Application Configuration демонстрируется соответствие между расширениями и серверными DLL-библиотеками.

тал, его следует сконфигурировать как ISAPI-«слушатель». Здорово, верно? Быстрое решение этой проблемы — просто определить для службы WS такую конфигурацию, в которой использован ASP-«слушатель». Однако, фирма Microsoft рекомендует использовать ISAPI-«слушателей» по причинам, связанным с производительностью.

Вот как определить в конфигурации ISAPI-«слушателя» (библиотека SOAPISAP.DLL), если вам это необходимо. (Вы можете найти дополнительные сведения об этом в разделе «ISAPI listener» в файле SOAP.CHM.)

Во-первых, щелкните правой кнопкой мыши по узлу Default Web Site в поддереве IIS, выберите во всплывающем меню пункт Properties и щелкните мышью по вкладке Home Directory (см. рис. 2).

Нажмите кнопку Configuration, чтобы открыть диалоговое окно Application Configuration, как показано на рис. 3. Найдите в списке расширение .wsdl. Если этого расширения в списке нет, щелкните мышью по кнопке Add. В диалоговом окне Add/Edit Application Extension Mapping, показанном на рис. 4, щелкните мышью по кнопке Browse и в режиме просмотра выберите файл “C:\Program Files\Common Files\MSSoap\Binaries\soapisap.dll”, а все остальные параметры настройте как показано на рисунке.

Если вы не можете найти DLL-библиотеку в режиме работы кнопки Browse, не волнуйтесь. Я настраивал соответствие расширений приложений на

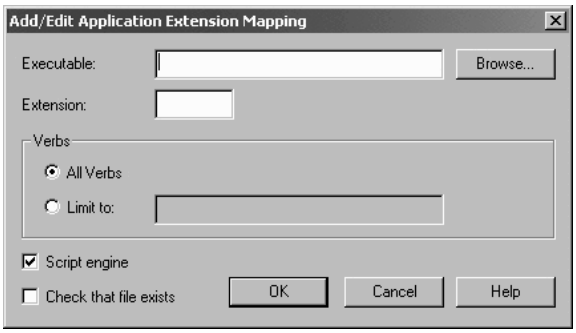

Рис. 4. Диалоговое окно Add/Edit Application Extension Mapping позволяет вам устанавливать и менять соответствие между расширениями и серверными DLL-библиотеками.

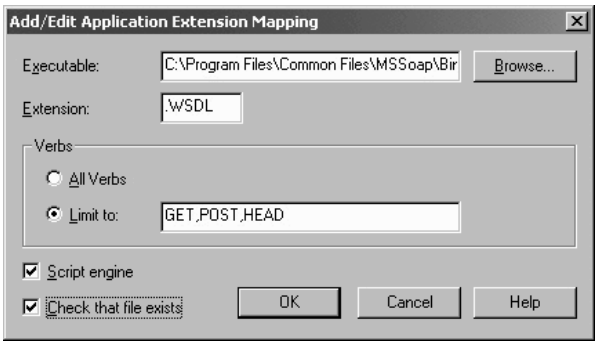

Рис. 5. Диалоговое окно Add/Edit Application Extension Mapping после получения информации.

нескольких машинах, работающих и под управлением NT4, и под управлением Windows 2000 применительно к различным приложениям, и мне ни разу не удалось найти DLL-библиотеку в диалоге File Open. Вы можете выбрать аналогичный файл (я часто буду использовать INI-файл с тем же самым именем), а затем, как только окно Add/Edit воспримет эту информацию, отредактировать имя файла.

На рис. 5 показано, как будет выглядеть диалоговое окно Add/Edit после получения информации.

Гэри Де Витт (Gary DeWitt) сообщает, что этот процесс прекрасно работает, но он не смог заставить ISAPI-«слушателя» работать на сервере, где функционирует ОС Windows XP Professional. Может быть, вам придется использовать на XP-машинах ASP-«слушателей». Однако, вы, вероятно, столкнетесь с этим только во время разработки, поскольку вряд ли захотите использовать версию XP Professional для эксплуатации приложения в реальных условиях, вам захочется использовать NT4-сервер или Windows 2000-сервер.

Сценарии IntelliSense

Флажок IntelliSense scripts, для которого предусмотрен независимый переключатель, и заданное имя создадут запись в таблице FOXWS.DBF, предназначенную для ее использования механизмом IntelliSense. Если вы не выставите упомянутый флажок, придется написать весь соответствующий код вручную при создании программы, использующей службу Web Service.

Вы можете указать свое собственное имя вместо того, которое предлагается по умолчанию. В примере из предыдущей статьи по умолчанию используется имя "hwpclass web service" — не слишком информативное. Если бы я назвал этот класс немного удачнее, например NewsService, механизм IntelliSense мог бы использовать имя "NewsService web service" — что намного лучше.

Кодировка UTF-16 Unicode

Для представления английского и большинства европейских языков используют однобайтовый символьный набор — все необходимые комбинации символов могут быть представлены с помощью одного байта. В других языках, таких как восточные диалекты, насчитывается больше символов, нежели можно обработать с помощью одного байта, следовательно, необходимо использовать два байта. Кодировка Unicode — это вариант набора двухбайтовых символов. Если вам необходимо использовать двухбайтовый символьный набор, вы должны выставить этот флажок.

Результаты публикации служб Web Services

В следующем диалоговом окне (см. рис. 11 в предыдущей статье) вы видите, что произошло. Был создан COM-сервер; был создан WSDL-файл; определен «слушатель»; и была создана запись для механизма IntelliSense. Зачем вам беспокоиться о механизме IntelliSense, если вы публикуете службу Web Service? Я имею в виду, что вы создаете файлы для эксплуатации, зачем вам понадобились записи для технологии IntelliSense, работающей на инструментальной машине? Главным образом затем, что вам понадобится опробовать службу Web Service сразу же после тестирования. Вы не обязаны это делать, если будете проводить тестирование службы на другой машине, но, вероятно, это крайне редкий случай.

Регистрация служб Web Services

С другой стороны, вы являетесь клиентом, который хочет воспользоваться услугами службы Web Service. Чтобы легко можно было сделать это, вы создали запись в диспетчере IntelliSense Manager, щелкнув по кнопке Web Services на вкладке Types и открыв диалоговое окно Visual FoxPro Web Services Registration. Чтобы напомнить себе, см. рис. 13 в предыдущей статье.

После нажатия кнопки Web Services вы можете определить имя для записи, создаваемой в диспетчере IntelliSense Manager, а также указать интересующую вас службу Web Service. В предыдущей статье мы указали ту службу Web Service, которую только что создали на этой же машине; вероятнее всего, выступая в роли потребителя службы Web Service, вы будете указывать на другую машину, расположенную где-то еще во Вселенной Internet.

Говоря точнее, эта процедура помещает записи в таблицу FoxCode, так что вы можете проделать следующее: выберите в выпадающем списке пункт "hwpclass web service", щелкните правой кнопкой мыши рядом с другими классами, например, рядом с элементами управления text box и grid, и тогда ассоциированный с данной службой Web Service программный код в полном объеме будет помещен в вашу программу (или в командное окно):

```
LOCAL ows as
```

Зачем вставлять «заплатку», чтобы подкаталог \NEWS оказался в корневом каталоге (и где исполняется COM-сервер)

Когда вы запускаете COM-сервер из командного окна VFP, вот так:

```
o=createobject("wsc.hwpclass")
? o.getnews()
```

то Visual FoxPro создает объект класса из текущего каталога VFP. Она не исполняет тот COM-сервер, который был зарегистрирован в реестре Windows Registry. Вот почему VFP ищет данные на диске E:.

Однако, при исполнении тестовой программы, создавшей службу Web Service, вы на самом деле создали DLL-библиотеку, используя информацию из реестра Windows. Моя тестовая программа располагается в корневом каталоге диска E:. Но COM-компонент не инсталлируется в этот каталог и не исполняется из корневого каталога диска E:. Откуда же он исполняется? Из каталога WINNT\System32! Откуда вы это знаете? Кто-нибудь мог бы сказать вам об этом, или вы могли бы выяснить это сами с помощью вот такого временного фрагмента кода, вставленного в файл wsc.prg непосредственно перед предложением SELECT:

```
strtofile("Where is my web service DLL is running?", ;
    "myveryownwebservice.txt")
```

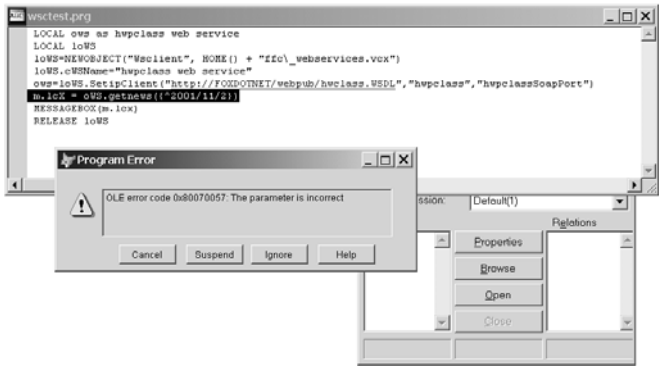

Рис. 6. Некорректная передача параметров приводит к неудобоваримому сообщению об ошибке.

При запуске службы Web Service в текущем каталоге будет создан файл myveryownwebservice.txt, в данном случае он будет создан в каталоге WINNT\System32.

Иначе говоря, когда вы исполняете COM-компонент, по умолчанию используется тот каталог, в котором находятся файлы runtime-модулей Visual FoxPro (узнать, что это за каталог, можно с помощью функции HOME()). Как правило, в этой роли выступает системный каталог, а он не является подходящим местом для хранения всяких там вещей... Вроде ваших данных!

Итак, когда вы инсталлируете свою DLL-библиотеку (и все вспомогательные файлы), инсталлируйте ее в каталог по своему выбору. После этого выполните в программном коде команду:

```
set default to (JUSTPATH(_VFP.ServerName))
```

Идеальным местом для размещения этой строки было бы событие Init вашего класса, дабы остальной код мог найти все необходимое.

Таким образом, когда служба Web Service из рассматриваемого в предыдущей статье примера искала таблицу NEWS.DBF, она искала ее в каталоге WINNT\System32. Поскольку таблица NEWS.DBF исходно была «запрятана» где-то на диске E:, что ж, сожалею, приятель. Поэтому, ради данного примера я явно указал на корневую вершину текущего каталога, а затем скопировал файл NEWS.DBF в корневой каталог диска C:. В следующем месяце я немного «подчищу» это место, но на данный момент нас меньше всего должны волновать какие-то вопросы, относящиеся к определению путей доступа и имеющие дело с самими данными.

Передача параметров

В том небольшом фрагменте программного кода, который я написал в предыдущей статье, предусмотрена передача параметров. Хотя использованный мной механизм работает применительно к обычным старым COM-компонентам, компоненты, которые публикуются как службы Web Services, должны использовать ключевое слово AS, чтобы определить тип передаваемого параметра и тип возвращаемого значения. Если бы вы сделали попытку передать службе Web Service параметр без внесения этого изменения, то получили бы сообщение об ошибке, показанное на рис. 6.

Говоря точнее, несколько строк, с которых начинается определение класса:

```
DEFINE CLASS hwpclass AS session OLEPUBLIC
Name = "hwpclass"
PROCEDURE getnews
LPARAMETERS ldDate
DO case
```

следует заменить следующим:

```
DEFINE CLASS hwpclass AS session OLEPUBLIC
Name = "hwpclass"
FUNCTION getnews (ldDate as Date) as String
DO case
```

В этом фрагменте кода определяется параметр ldDate типа Date, а также возвращаемое значение типа String. После внесения этих изменений вам необходимо повторно опубликовать службу Web Service (используя пункт меню Tools | Wizards | Web Services). Если вы не опубликуете службу повторно, то обнаружите в WSDL-файле вот эти строки:

```
<message name='hwpclass.getnews'>
</message>
```

В этом объявлении параметр отсутствует. Если бы WSDL-файл знал о существовании параметра, он перечислил бы его в предыдущем XML-коде и включил бы тип параметра в описание, вот так:

```
<message name='hwpclass.getnews'>
   <part name='ldDate' type='xsd:dateTime'/>
</message>
```

Затем в «плохом» WSDL-файле вы увидите вот это:

```
<message name='hwpclass.getnewsResponse'>
   <part name='Result' type='xsd:anyType'/>
</message>
```

Тип возвращаемого значения указан как anyType — это «жаргонизм» из лексикона инструментального набора MS SOAP Toolkit, обозначающий тип variant. Это означает одно из двух: либо метод был определен в VCX-библиотеке, либо не был указан тип возвращаемого значения (согласен с обвинением). После определения типа возвращаемого значения WSDL-файл продемонстрирует следующее:

```
<message name='hwpclass.getnewsResponse'>
   <part name='Result' type='xsd:string'/>
</message>
```

Гораздо лучше. Далее, в файле WSCTEST.PRG, вы используете вот такую строку:

```
m.lcX = oWS.getnews({^2001/11/2})
```

для передачи параметра даты службе Web Service.

Обратите внимание на то, что вы не обязаны оформлять свой код в виде программы PRG. Вы можете использовать VCX-библиотеку, необходимо только сделать ее доступной для COM-компонентов или служб Web Services в программе PRG, вот так:

```
DEFINE CLASS MyCOMInterface AS Custom OLEPUBLIC
FUNCTION MyMethod (MyParm AS String) AS String
   LOCAL o AS myclass
   o = CREATEOBJECT("myclass")
   RETURN o.MyMethod(MyParm)
ENDDEFINE
```

Тем самым обеспечиваются разные полезные вещи. Во-первых, это хороший способ разделить интерфейс и реализацию. (Если эта концепция все еще остается для вас туманной, рассматривайте интерфейс просто как набор свойств и методов, которые видны внешнему миру, например, пользователям службы Web Service, тогда как реализация — это волшебство программирования, которое вы «творите» внутри компонента — магия, которую не увидит никто другой.)

Во-вторых, вы не можете определить типы в VCX-библиотеках, только в программах PRG: посредством inline-синтаксиса, который я вам только что продемонстрировал.

И наконец, если вы следуете этим указаниям, то можете определить интерфейс, откомпилировать свой компонент и опубликовать службу Web Service (иначе выражаясь, создать WSDL-файл) только один раз. Вы можете вернуться обратно и сконфигурировать готовую реализацию без обязательной повторной публикации службы Web Service. Библиотека COM-типов экспортирует реальные типы данных, а не типы данных variant, и полученная в результате этого служба Web Service также будет отображать типы данных корректно.

Ошибки типа «DLL in use»

Наиболее вероятно, что вам неоднократно придется перекомпилировать свою DLL-библиотеку в Visual FoxPro. Когда вы будете это делать, то, вероятно, столкнетесь с сообщением об ошибке «XXX.DLL is in use», и вам не позволят закончить процесс компиляции. Происходит следующее: IIS кэширует ваш модуль в памяти, чтобы повысить производительность. Ага, я знаю, нашу производительность как разработчиков это не повышает, верно?

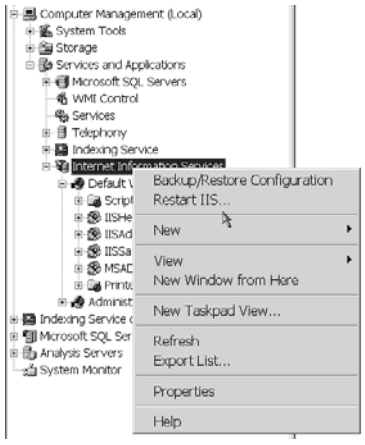

Рис. 7. Перезапуск сервера IIS из апплета Computer Management.

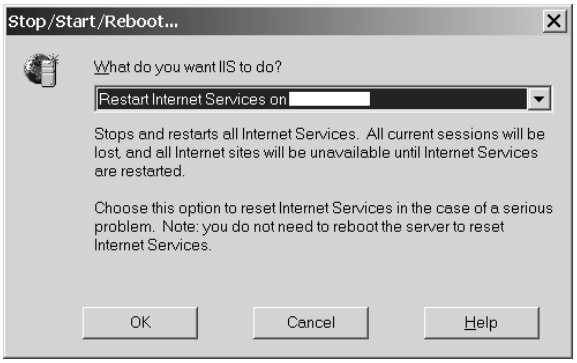

Рис. 8. Выбор опции Restart Internet Services.

Хорошо, поскольку DLL-библиотека по-прежнему работает, как вам ее остановить? Верный способ — все закрыть и перезагрузить машину; хорошо, но на это уйдет время и это скучно. Более приемлемый способ — исполнить команду Restart IIS.

Откройте апплет Computer Management панели управления Control Panel, выберите сервер IIS, щелкните правой кнопкой мыши и выберите в меню пункт Restart IIS, как показано на рис. 7.

Вы увидите диалоговое окно, показанное на рис. 8. Выберите опцию Restart Internet Services, щелкните мышью по кнопке OK и ждите, может быть, полминуты или около того. В конце концов, вас вернут к экрану апплета Computer Management, и вы увидите, что ваши службы снова работают. Теперь можете перекомпилировать свою DLL-библиотеку.

В следующий раз мы создадим более надежную службу Web Service и введем ее в эксплуатацию!

❑❑❑

Возвращаясь к кросс-платформенности: использование Visual FoxPro для подключения к серверу MySQL в ОС Linux

Боб Ли (Bob Lee)

WIN 7.0 6.0 5.0 3.0 DOWNLOAD

Помните те времена, когда, помимо прочего, притязания FoxPro на славу подкреплялись и возможностью работы на различных платформах — «кросс-платформенностью» этого инструментального средства? Вы могли бы написать приложение и, проявив минимальную предусмотрительность и затратив минимальные усилия на планирование, заставить один и тот же базовый программный код работать на платформах DOS, Windows, Unix и Mac. Затем некие маркетинговые гении низвели термин «кросс-платформенный» до обозначения версий операционной системы Windows 3.1, Windows 95 и Windows NT, и Fox постепенно утратила возможность работы на других платформах. Однако, эти счастливые дни вернулись. В своей статье Боб Ли показывает, как использовать настольные Visual FoxPro-приложения для получения доступа к данным, размещенным на другой платформе, уделяя при этом особое внимание установлению соединения с базами данных MySQL, размещенными на сервере Linux.

Должен внести ясность. Изначально я писал эту статью с целью продемонстрировать VFP-разработчикам, что создание клиент/серверного приложения отныне не относится к тем задачам, необходимость решения которых неизбежно приводит в уныние. По-прежнему остаются на своем месте известные преграды, обусловленные технологией, не утратили своего значения кривые графиков времени и обучения, а также существенные затраты на программное обеспечение, но все эти факторы можно преодолеть. Я хотел осуществить переход от классического «настольного» VFP-приложения к системам клиент/сервер, но постоянно наталкивался на препятствия. Обращение к справочному Help-файлу привело к тому, что я погряз в болоте имеющихся возможностей и незнакомого синтаксиса, при этом ни то, ни другое не указывало верного направления, и мне пришлось бы затратить много времени на тестирование, чтобы выяснить, как использовать все эти средства и обеспечить «правильное» соотношение мощи и простоты. Microsoft SQL Server, хотя и не такой дорогой как Oracle, все-таки недешев, а его «младший брат» MSDE со своим максимумом в пять пользователей был настолько ущербным, что оказался непригоден для каких-либо иных целей, кроме практического обучения и создания тривиальных приложений. (На самом деле, MSDE не имеет ограничения в пять пользователей. В нем встроено то, что называется concurrent workload governor, принудительно замедляющий работу сервера, когда число одновременно выполняющихся пакетов команд (batch) превышает пять. Т. е. шесть будут работать медленнее чем пять, седьмой сделает работу еще медленнее и т. д. — *прим. ред.*)

Но оказалось, эта статья явилась хорошим учебным пособием, позволяющим выяснить, как установить соединение с теми базами данных, которые размещаются на иной, «не-Windows»-платформе. Одним выстрелом я ухитрился убить двух зайцев.

Почему выбран сервер MySQL?

Когда я начал искать возможность сделать следующий шаг и перейти от VFP-приложений для Web к истинному клиент/серверному настольному приложению, я снова и снова сталкивался с сервером MySQL. Похоже, он был повсюду. Доступны были и другие базы данных, и полагаю, они в равной степени хорошо работали бы, но меня поразило абсолютное число инсталляций СУБД MySQL у поставщиков услуг ISP и ASP. Хорошая документация, много сопутствующей литературы и благотворное влияние жаждущих помочь «посвященных» пользователей стали теми дополнительными факторами, которые подтолкнули меня навстречу серверу MySQL. Как и все прочие варианты, сервер MySQL не является тем единственно верным решением, пригодным для любого приложения баз данных или приложения, связанного с VFP, но это достойный соперник. Прежде чем вы проведете долгие часы, работая с этим инструментом, вы должны осознать присущие ему ограничения. Например, сервер баз данных MySQL пока еще не поддерживает подзапросы (subqueries) и хранимые процедуры.

Основы СУБД MySQL и ОС Linux

Linux-серверы вездесущи, невзирая на то, до какой степени хотелось бы обратного фирме Microsoft. Нет, ну где же вы прятались, если до сих пор не сталкивались с сервером Linux? Все крупнейшие

компьютерные компании, производящие мэйнфреймы, проявляют интерес к операционной системе Linux. Фирмы IBM, HP, Silicon Graphics и Sun — все они в качестве своей основной платформы выбрали операционную систему Linux. На мощных серверах этих фирм работают Oracle и DB2 — эти «киты» на рынке баз данных. Сегодня многие из таких СУБД по-прежнему существуют под управлением операционной системы Unix, но со временем многие перейдут на платформу Linux.

Серверная СУБД MySQL (www.mysql.com) — это самая распространенная в мире база данных с открытым исходным кодом, и это стандартный пакет, входящий в большинство дистрибутивных поставок операционной системы Linux (в этом смысле он очень похож на приложение Notepad, входящее в поставку ОС Windows). Этот сервер доступен бесплатно с лицензией GNU General Public License (GPL). В отличие от пакета MSDE — «бесплатной» версии SQL Server фирмы Microsoft — СУБД MySQL не имеет ограничений на количество подключаемых к серверу пользователей (число пользователей — это опция конфигурации, и по умолчанию ее значение равно 100).

Сервер MySQL не является ни самым мощным или лучшим, ни самым дешевым или самым дорогим, но он претендует на то, чтобы быть самым быстрым сервером баз данных. Возможность сделать такое заявление обусловливается тем, что MySQL — это «облегченный» сервер, у которого отсутствуют некоторые развитые функциональные возможности, присущие более мощным отказоустойчивым базам данных. (Дополнительные сведения о сравнительной производительности (benchmarks) сервера MySQL ищите по адресу www.mysql.com/information/benchmarks.html.)

Вместе с тем, этот сервер не является игрушечной базой данных. В последней статье, опубликованной на посвященном MySQL Web-сайте, обращается внимание на то, что U.S. Census Bureau (бюро по переписи населения США) создает и эксплуатирует Web-сайты, служащие теми ресурсами, которые необходимы для сбора национальной и местной статистики в целях проведения переписи населения, и большая часть этих сайтов используют сервер MySQL.

Вплоть до недавнего времени вы могли бы отправиться по адресу www.freesql.org и «нанять на работу» свой собственный бесплатный сервер баз данных. Прелесть этого сайта в том, что его владельцы предоставляют вам бесплатный хостинг для базы данных MySQL. В настоящее время у них создано и работает 10 404 действующих баз данных MySQL. Ко времени написания этой статьи регистрация новых клиентов временно приостановлена, но они рассчитывают на скорое возобновление своей деятельности и прием новых заявок.

Для вас, как для VFP-разработчика, это означает…

Что вы можете начать использовать мощь базы данных SQL, используя в качестве клиента (front end) VFP, и без дополнительных денежных вложений создавать клиент/серверные приложения виртуально. Синтаксис языка запросов SQL определяется стандартом ANSI 92. Почти каждая такая база данных имеет доступные ODBC-драйверы, так что с их помощью VFP может устанавливать соединение с сервером.

Наши клиенты ожидают, что мы сможем предложить им клиент/серверный вариант приложения, нет, поправка, они настаивают на этом. Все чаще и чаще наши клиенты полагают, что все приложения должны быть таковы, чтобы их без внесения изменений в программы можно было использовать из дома, офиса и гостиничного номера. Разработка распределенных приложений баз данных, которые некогда рассматривались как отличительная особенность программного обеспечения корпоративного уровня (enterprise-level), теперь доступна и вам.

Начало

Так, пора приступить к делу. Как насчет того, чтобы сначала представить обзор в виде общей картины? VFP-приложение с помощью ODBC-драйвера посылает SQL-предложение на сервер MySQL, и в соответствии с протоколом TCP/IP это предложение доставляется по вашей компьютерной сети (совершенно не имеет значения, является ли эта сеть локальной или это Internet) на ту машину, где работает сервер MySQL. База данных MySQL обрабатывает полученное SQL-предложение и возвращает результаты в VFP в виде редактируемого курсора (read-write cursor). Вы обрабатываете этот курсор, а затем завершаете транзакцию (commit) и возвращаете внесенные изменения обратно на сервер.

Давайте рассмотрим каждый шаг в отдельности.

Доступ к БД MySQL

Первое, что вам необходимо сделать, это получить доступ к базе данных MySQL. Я буду исходить из предположения, что у вас есть доступ к серверу Linux, на котором размещена база данных MySQL, и вы знаете его IP-адрес. В отличие от SQL Server, работая с которым вы должны обратиться к операци-

онной системе, чтобы узнать, где находится база данных (скольким из вас довелось оказаться в ситуации, когда SQL Server без предупреждения «исчезал» из списка?), в данном случае вы просто используете IP-адрес Linux-машины для идентификации сервера MySQL. (Точно также вы можете получить доступ к SQL Server, используя IP-адрес его компьютера. Единственно, что плохо: если этот сервер использует DHCP для получения адреса, то в один прекрасный момент вы получите сообщение «Server does not exist or access denied» в случае изменения IP-адреса, и придется менять настройки клиента — *прим. ред.*)

ODBC-соединения

Следующее, что вам потребуется, это средства для установления соединения между VFP и MySQL. Это делается с помощью ODBC-драйвера для сервера MySQL. ODBC — это набор протоколов, которые соблюдают все серверы баз данных и для которых предоставлен общепринятый интерфейс. Сейчас фирма Microsoft переходит от интерфейса ODBC к новому интерфейсу, который называется OLEDB, но большая часть сложившихся баз данных, представленных на рынке, не была переработана «под» использование интерфейса OLEDB, так что пока останемся верными надежному и испытанному интерфейсу ODBC. Вы можете бесплатно загрузить последнюю версию ODBC-драйвера с Web-сайта, посвященного серверу MySQL.

Редактируемый курсор

Да, редактируемый курсор! Когда вы внесли в данные все изменения, вы отправляете эти изменения обратно на сервер, используя для этого стандартный язык запросов SQL. Вы работаете со знакомой и понятной вам вещью — курсором VFP. Выполните команды SKIP и REPLACE применительно непосредственно к содержимому редактируемого курсора. В заключение, напишите одну строку программного кода для того, чтобы поместить внесенные вами изменения на сервер. Дело сделано. Только и всего.

Столь же просты отчеты. Получите курсор с интересующими вас данными. Выполните команду REPORT FORM <YOUR_REPORT> TO PRINT. Закройте этот курсор. Вот и все. Так же просто добавить запись. Получите курсор с пустыми записями, выполните необходимые команды REPLACE применительно к этому курсору и затем добавьте новые записи в размещенную на сервере таблицу.

Как только вы запомните, что всегда работаете с курсорами (а не с самими таблицами или представлениями VFP), возврат внесенных изменений обратно на сервер станет таким же простым и привычным делом, как и обращение к функции TableUpdate(.T.) языка SQL.

Советы

На практике вам необходимо написать свои собственные оболочки для SQL-предложений, обеспечивающих установление соединения (или же воспользоваться коммерческими каркасами — framework). Но как только вы это сделаете, такие оболочки становятся частью вашей библиотеки, и вы просто снова и снова используете их.

Второй совет — остерегайтесь удаленных представлений (Remote Views). Хотя складывается впечатление, что они легко настраиваются, используя их, вы теряете возможность напрямую управлять SQL-предложениями или буферизацией VFP. Если вы применяете удаленные представления, страдают и надежность, и быстродействие.

Поскольку серверы класса SQL Server не поддерживают такого понятия, как удаленная запись (deleted record), у вас может возникнуть необходимость изменить структуру хранящихся на сервере таблиц, чтобы продублировать возможность работы с удаленными записями DBF-таблицы и добавить в каждую таблицу поле для флажка удаления (deleted flag). Я использую для этого символьное поле единичной длины (например, поле типа TINYINT) и заполняю его значениями 1 или 0, чтобы указать, надо ли рассматривать эту запись как удаленную (deleted()). В приводимом мной примере программного кода в качестве поля удаления записи для каждой таблицы используется поле с именем DLD.

Во всех MySQL-таблицах должно быть определено поле первичного ключа (Primary Key). Это связано с тем, что для SQL-таблицы не реализовано понятие номера записи. Воспользуйтесь возможностью автоинкремента, предоставляемой СУБД MySQL. Вы можете определить для поля тип INT и сделать это поле автоинкрементным. Такой тип поля, которого очень не хватает в VFP, реализован в СУБД MySQL, и он мгновенно предоставляет вам в случае необходимости уникальный идентификатор записи ID и первичный ключ PK. (Версия 8.0 Visual FoxPro будет поддерживать тип данных с автоматическим приращением значения — *прим. ред.*).

СУБД MySQL, работающая под управлением ОС Linux, учитывает регистр в именах таблиц, поскольку операционная система Linux, как и другие разновидности ОС Unix, является чувствительной к регистру. Знайте, что когда вы пишете свои SQL-предложения, вы должны убедиться в том, что имена таблиц указаны с учетом регистра, и их напи-

сание совпадает с написанием имен тех таблиц, к которым вы пытаетесь обратиться с запросом. Если СУБД MySQL работает под управлением ОС Windows, имена таблиц не чувствительны к регистру.

Не заходите слишком далеко в использовании тех функций СУБД MySQL, которые обеспечивают предоставление различных прав (permissions). Хотя при первом знакомстве с ними создается впечатление, что они в состоянии сделать абсолютно все, что могло бы понадобиться вашему приложению, я обнаружил, что гораздо лучше ограничиться определением трех классов пользователей. Можно организовать достаточно сложную систему предоставления прав доступа к SQL-таблицам, но подробное описание этого выходит за рамки данной статьи. Достаточно сказать, что я исходил из того предположения, что вы создаете следующие три уровня доступа. Первый уровень — это администратор вашей базы данных, второй уровень — постоянные пользователи, которые могут вносить данные в таблицы и читать их, и третий уровень — это случайный пользователь, который может только читать хранящиеся в таблицах данные, но не может их туда вносить. Права доступа, предоставляемые СУБД MySQL, не накладывают ограничений на типы SQL-предложений, которые вы можете составить, и, таким образом, для того, чтобы обеспечить деление пользователей на различные группы, вам необходимо запрограммировать отдельные предложения, предназначенные для использования каждой группой. Следовательно, большая часть такой системы безопасности должна программироваться в вашем приложении. Так что не злоупотребляйте раздачей прав доступа средствами СУБД MySQL.

Поскольку большинство из нас приступает к делу, уже имея в некотором каталоге DBF-файлы, вас, может быть, успокоит понимание того, как выглядит файловая структура базы данных MySQL. По существу, база данных MySQL соотносится с каталогом на диске, а таблицы — это находящиеся в данном каталоге файлы. Есть одна особая база данных, которая называется “mysql” и в которой хранятся данные о пользователях и правах доступа к таблицам, размещенным на используемом сервере. Структура MySQL — это структура серверного приложения: работая под управлением ОС Windows, оно исполняется как служба, тогда как в среде Linux это приложение называется Daemon-процессом, аналогично HTTP-серверу. Эта служба обычно стартует при загрузке компьютера, работает в режиме 24x7 и, по существу, активна и потребляет ресурсы только тогда, когда к ней обращаются с SQL-запросом.

Для взаимодействия с самим сервером есть интерфейс командной строки, но я, например, после появления первого пользователя им не пользуюсь. В таком случае я предпочитаю для любого взаимодействия с этим сервером использовать утилиту MySQLFront или средства VFP. СУБД MySQL конфигурируется тем же способом, что и большинство Linux-приложений, путем модификации одного текстового файла (my.cnf), синтаксис которого похож на синтаксис INI-файлов в операционной системе Windows. Следовательно, вы, по существу, запускаете сервер MySQL, позволяете ему работать и управляете им с помощью утилиты MySQLFront.

Установка соединения

Используя VFP, вы располагаете множеством способов для установки соединения с сервером MySQL, но в действительности все они делают одно и то же. Все возможные способы посылают серверу MySQL предложение, которое включает в себя IP-адрес или имя сервера, имя пользователя, пароль, номер порта и другие, указываемые вами, опции.

Хранение этой информации в записи источника данных DSN/ODBC — это один способ, с помощью которого можно установить соединение, но такой метод требует, чтобы конечный пользователь прошел сквозь очень длинный настроечный сценарий для конфигурации DSN. Давайте все вместе: Фу-у! Другой вариант, который обеспечивает нам VFP, — использовать контейнер DBC для хранения этой информации в чем-то таком, что называется удаленное соединение (remote connection). Если вас это устраивает, вы могли бы поступить именно так, но, поскольку мне нравится идея, заключающаяся в отказе от локального хранения данных любого сорта, предложение использовать для хранения такого рода метаданных контейнер DBC показалось мне несколько излишним.

Затем наступает очередь моего любимого метода, который заключается в обращении к функции SQLstringconnect(). Эта функция пересылает всю информацию, необходимую для создания соединения с сервером. Соединение считается установленным, если упомянутая функция возвращает любое значение кроме значения -1: вы получаете целое число больше 0, и это значение является дескриптором (handle) соединения. В программе вы в течение необходимого времени храните это значение дескриптора соединения, а затем выполняете команду SQLdisconnect(), передавая указанное значение в качестве параметра, чтобы разорвать соединение. Я полагаю, что вы поддерживаете установленное соединение только до момента получения необходимого результата.

Вот пример того, как может выглядеть строка соединения. Я храню эту информацию в INI-файле, который называется DNS.INI, и при необходимости считываю его содержимое, используя функцию strtofile(). Может оказаться, что вы отдаете предпочтение использованию DBF-таблицы, в которой хранится локальная база метаданных для ваших пользователей, или возможно MEM-файлу или чему-то в формате XML. Какой бы способ вы не выбрали, вам необходимо иметь доступ к строке соединения.

Вот пример моей строки соединения, она разбита на несколько текстовых строк для того, чтобы обеспечить представление, удобное для рассмотрения в данной статье:

```
DRIVER={MySql Server}
   SERVER=216.XXX.XXX.13X;
   PORT=3306;
 UID=bob;
 PWD=XXXXXX;
 DATABASE=mydatabase;
 option=131609
```

А вот что означают все эти отдельные фрагменты:

```
DRIVER={MySql Server}
```

Это значение должно быть заключено в фигурные скобки, потому что имя содержит внутри себя символ пробела.

```
  SERVER=216.XXX.XXX.13X;
```

Данное значение определяет местонахождение серверной машины, на которой установлена СУБД MySQL. Синтаксис разрешает использовать вместо IP-адреса имя, но такая замена приводит к некоторому замедлению, поскольку каждый раз при обращении к серверу приходится выполнять процедуру разрешения имен.

```
  PORT=3306;
```

Указывать порт необязательно. Обычно это значение определяется сервером как порт 3306. На моем сервере СУБД MySQL работает с портом 1433, поэтому удалите это значение из своей строки соединения, если вы взаимодействуете с обычным сервером MySQL.

```
  UID=bob; PWD=XXXXXX;
```

Эта строка должна быть понятна без объяснений.

```
  DATABASE=mydatabase;
```

Хорошая идея — передать имя в параметре DATABASE. Таким образом указывается база данных, используемая по умолчанию, и вам не надо указывать в каждом SQL-предложении к какой именно базе данных вы подключаетесь. Эта настройка является необязательной.

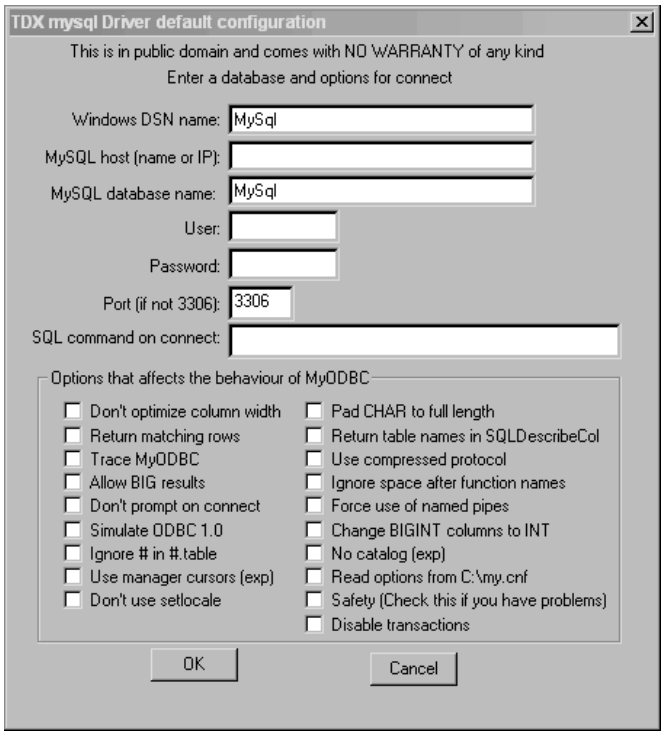

Рис. 1. Конфигурация ODBC-драйвера для сервера MySQL.

```
option=131609
```

Данное числовое значение необязательного параметра составлено из нескольких двоичных флажков: очень похоже на определение параметров для функции MessageBox в Visual FoxPro. Более точно это значение строится следующим образом: перейдите в панель управления ODBC и откройте окно MySQL DSN, как показано на рис. 1. Те опции, на которые ссылаюсь я, расположены внизу (набор независимых переключателей). Они пронумерованы сверху вниз, начиная с первой колонки. Первая опция имеет номер 1, вторая — 2, третья — 4, затем 8, 16, 32... Чтобы указать несколько параметров, вы суммируете значения номеров этих переключателей и передаете полученное число. Итак, если вам необходимы опции 1, 2 и 3, вы передаете 7 (1 + 2 + 4).

Выбранные мной опции, в сумме составляющие число 131 609, включают следующие установки: «Pad char to full length», «Don't optimize column width» и «Safety (check this if you have problems)». Если не выбрать первую опцию, то с сервера возвращаются данные без завершающих пробелов. Это суммарное значение имеет вид очень большого числа, но это просто пять опций, и это числовое значение

эквивалентно двоичным значениям первой, четвертой, пятой, десятой и восемнадцатой опций.

Функции преобразования MySQL-данных

Как я уже упоминал ранее, вам необходим набор функций, которые позволяют VFP взаимодействовать с сервером MySQL. Некоторые из этих функций преобразуют VFP-данные в соответствии с типами данных сервера MySQL, тогда как другие являются оболочками для типичных операций баз данных, например Save и Add. Я поместил каждую из этих функций в отдельный процедурный файл; разумеется, вы могли бы также разместить их в библиотеке классов.

В таблице 1 приводится перечень написанных мной функций.

Таблица 1. Функции MySQL.

Функция	Описание
MySQL	Оболочка для получения VFP-курсора из таблицы MySQL (но без разрыва соединения).
MySQLP	Оболочка для получения VFP-курсора из таблицы MySQL (и разрыв соединения).
DtoSQL	Преобразует VFP-данные в стандартные данные, которые понимает MySQL.
FixSQL	Вызывается из FixSQL.
MySave	Сохраняет данные из курсора в таблице MySQL. Эта функция формирует SQL-предложение в оперативном режиме на основе структуры текущего курсора и посылает серверу MySQL SQL-команду, чтобы сохранить все внесенные в этот курсор изменения.
MyBlank	Создает пустой VFP-курсор из одной записи, готовый для того, чтобы вы заполнили его данными.
MyNew	Добавляет выбранный в данный момент курсор к соответствующей табличной записи MySQL. Эта функция сначала передает курсор в процедуру сервера MySQL (соединение не разрывается), получает новый первичный ключ PK (автоинкрементный), обновляет запись, а затем разрывает связь с сервером.
MyError	Базовый сценарий перехвата ошибок для записи в текстовый файл информации об ошибках.

Единственное отличие функции MySQL от функции MySQLP заключается в том, что функция MySQL не разрывает соединение. Она используется так, что вы можете добавить запись и позволить серверу MySQL автоматически создать для вас первичный ключ в этой таблице.

Черновые варианты каждой из этих функций включены в программу MY.PRG из файла-приложения к этой статье, который вы найдете на сопровождающей дискете.

Как конвертировать данные из DBF-таблицы Fox в формат данных сервера MySQL

Если только вы не намерены заполнять базу данных MySQL с «нуля», вам потребуется переместить VFP-данные на сервер MySQL. Во «всемирной паутине» Web можно найти несколько инструментальных средств, например утилиты DBF2MySQL и DBF2SQL, которые написаны для того, чтобы решить для вас эту задачу. Некоторые из этих инструментов представлены как программы PRG, тогда как другие являются исполняемыми EXE-файлами. Это свободно распространяемые (freeware) проекты, находящиеся на разных стадиях разработки и/или сопровождения, — выполните поиск по ключевым словам “dbf2mysql” или “dbf2sql” в Web, чтобы найти последние версии таких инструментов. Вы также можете для начала воспользоваться моей программой DBF2SQL.PRG, включенной в файл-приложение на сопровождающую дискету.

Как управлять вашей новой базой данных MySQL

Раз уж вы получили свои данные в формате MySQL, вам, вероятно, никогда не придется вносить в них изменения, верно? Я знаю, мне никогда не приходится этого делать. Ой, подождите, кроме только одного того случая, когда...

Итак, как обстоит дело с командой MODIFY STRUCTURE? Изменение ширины колонок, добавление индексов, вставка новых полей — как все это делается? Простой ответ (ну да, на тестирование других приложений ушли недели) заключается в следующем: используйте MySQLFront — бесплатную утилиту, которая предоставляет графический интерфейс для выполнения всех этих функций, а также предоставляет в ваше распоряжение командную строку для обращения к серверу MySQL. Я использую эту утилиту постоянно, во-первых, для генерации своих SQL-предложений и тестирования их прямо на сервере, а тогда, когда получены результаты обработки SQL-предложений, я выполняю применительно к ним операцию «вырезать и вставить», чтобы использовать полученные данные в VFP-коде. На рис. 2 утилита MySQLFront показана в тот момент, когда она установила соединение с одним из моих серверов, на котором исполняется SQL-предложение, и на экран выдается набор результатов. Утилиту MySQLFront можно загрузить по адресу www.anse.de/mysqlfront.

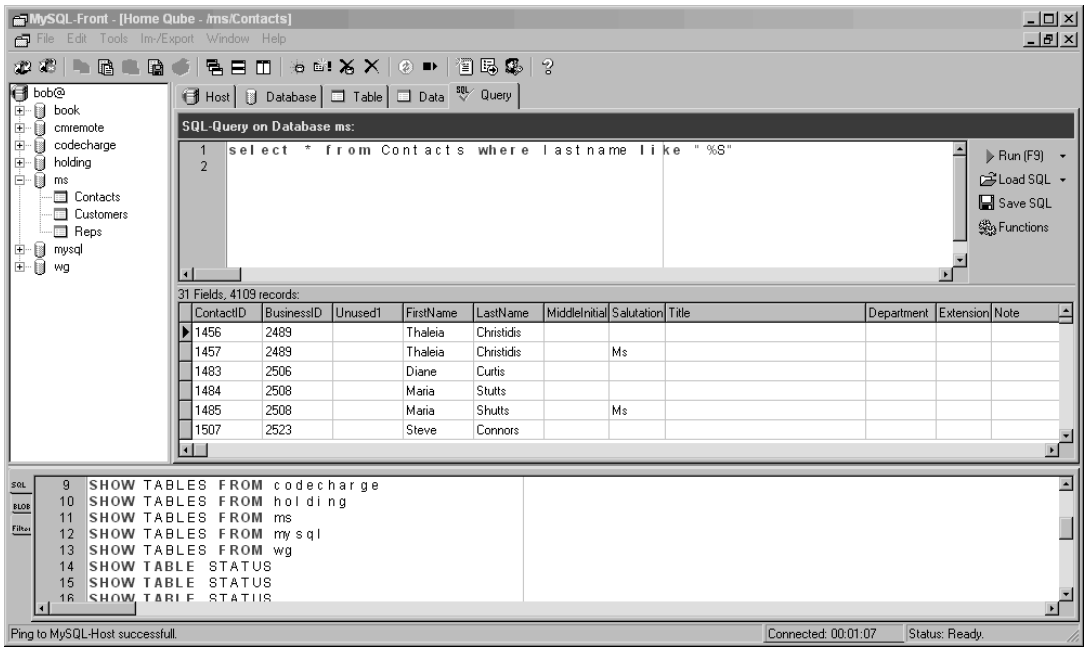

Рис. 2. Интерфейс утилиты MySQLFront позволяет вам манипулировать таблицами точно так же, как вы можете это делать в интегрированной среде разработки VFP.

Заключение

Соберите воедино все эти концепции, процедуры и функции — и вот вы уже готовы к тому, чтобы сделать следующий логический шаг по направлению к клиент/серверному миру. Воспользуйтесь Internet и своими навыками использования VFP в качестве клиента для сервера MySQL, и это будет кратчайший путь к любой другой SQL-базе. Вы можете создать отличные настольные приложения, которые будут получать данные из Internet, и вы в состоянии установить соединение с SQL-базами данных, размещенными на платформах, отличных от платформы Windows.

Как сервер MySQL, так и ODBC-драйвер для MySQL в настоящее время доступны бесплатно; а в конечном итоге вы получаете меньшее количество строк VFP-кода, приложения, более быстрые, чем те, которые вы могли бы создать, если бы программировали те же самые функции, используя «родные» VFP-таблицы, и, как я обнаружил, дополнительные возможности и производительность, добиться которых можно было бы только при работе в истинной клиент/серверной среде.

Моя многолетняя любовь к СУБД Fox отчасти объясняется тем, что она дает нам возможность распространять бесплатно runtime-модули для наших приложений. С помощью СУБД MySQL вы можете разрабатывать настоящие, надежные клиент/серверные приложения без необходимости делать внушительные отчисления в уплату за лицензию, которых требует SQL Server. Это решение — бесплатные runtime-модули VFP и отсутствие лицензий для сервера MySQL — является по-настоящему рентабельным кросс-платформенным решением для моих заказчиков и ваших, возможно, тоже.

Боб Ли (Bob Lee) является ИТ-менеджером в фирме Wolf-Gordon. Он разрабатывает решения на базе Xbase начиная с 1987 года и работал с такими СУБД, как dBASE III+/dBman for DOS и Xenix/FoxPro. Боб создает приложения баз данных для Web начиная с 1998 года, а приложения, использующие сервер MySQL, с 2001 года. В настоящее время он разрабатывает замену для приложения Adobe Acrobat, чтобы можно было создать PDF-файлы и осуществить для данных и отчетов из унаследованных систем переход к использованию самых современных интерфейсов и электронного хранилища документов.
Его адрес: bob@1amsoftware.com

❑❑❑

Настоящая фабрика

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

WIN 7.0 DOWNLOAD

В этом месяце Энди Крамек и Марсиа Акинз обсуждают использование, проектирование и реализацию класса на базе шаблона Factory. Адаптация этой методологии может значительно облегчить задачу сопровождения приложения, устранив необходимость явно ссылаться на библиотеки и имена классов в программном коде.

Марсиа: Ты видел материалы новой дискуссии, которую недавно открыл Стив Дингл (Steve Dingle), задав на форуме MSDevApps в сети CompuServe очередной вопрос из своей знаменитой серии «Остановись и задумайся об этом!»?

Энди: Не уверен. Напомни мне.

Марсиа: В принципе, он хотел узнать, используют ли разработчики так называемые factories («фабрики») для создания всех объектов и, если да, то создается ли экземпляр объекта factory-класса каждый раз заново, или он создается как объект глобального доступа при загрузке приложения. Отклики были равно интересны с двух точек зрения. Полученные ответы вскрыли наличие относительной путаницы, во-первых, в вопросе о том, что такое factory-класс, и, во-вторых, в оценке преимуществ, которые обеспечивает применение такого класса в приложении.

Энди: Хорошо, я клюну на эту наживку. Что такое factory-класс?

Марсиа: По существу, это такой класс, функцией которого является создание экземпляров объектов других классов. Но вместо того, чтобы строить догадки, давай получим сначала формальное определение. Согласен?

Энди: Ну вот, опять ты портишь совершенно замечательную дискуссию, сводя все к фактам! Но ты, разумеется, права. Вот определение шаблона Abstract Factory: «Обеспечивает интерфейс для создания семейств связанных или находящихся во взаимозависимости объектов без необходимости указывать конкретные классы для этих объектов». Это определение весьма проясняет вопрос или, по крайней мере, делало бы это, если бы я понимал, что оно означает.

Марсиа: Это просто способ образно сказать о том, что когда ты отделяешь процесс создания экземпляра объекта класса от имени того класса, объект которого создается, ты приобретаешь дополнительную универсальность.

Энди: Следовательно, процедура создания экземпляра объекта, то есть реальное исполнение функций CREATEOBJECT() или NEWOBJECT(), не программируется явно, а поручается этому «factory»-объекту?

Марсиа: Совершенно верно! Поэтому, вместо того, чтобы наблюдать вот такой код:

```
loObject = CREATEOBJECT( 'MyClass', MyClassLibrary' )
```

разбросанный по всему приложению, ты видишь код, который выглядит вот так:

```
loObject = oFactory.New( 'MyKeyword' )
```

Энди: Остановись на этом! Скажи на милость, что такое MyKeyWord?

Марсиа: Ключ к таблице метаданных, в которой хранится информация об именах и библиотеках классов, разумеется. Как известно всем, кто был на семинаре Great Lakes Great Database Workshop в Милуоки в октябре 2001 года, я никогда не касалась данных, которые мне не нужны. Именно так мы добьемся универсальности от factory-класса.

Энди: Ага! Теперь я вижу прямую выгоду. Если тебе когда-либо потребуется заменить один класс на другой, то вместо того, чтобы охотиться по всему проекту за ссылками на этот заменяемый класс и вносить изменения в каждую найденную строку кода, ты можешь просто изменить соответствующую запись в этой таблице метаданных.

Марсиа: Именно это я и имела в виду, говоря об отделении процесса от имени класса. Кроме того, не знаю как ты, но я редко добиваюсь правильной структуры своих библиотек классов с первого раза. Обычно мне в итоге приходится что-то переносить из одной VCX-библиотеки в другую. Следовательно,

даже если я не модифицирую сам по себе класс, я все-таки могу тратить время на поиск, чтобы откорректировать ссылки на библиотеку классов (встречающиеся в предложениях SET CLASSLIB или в обращениях к функции NewObject()).

Энди: Тогда возникает интересный вопрос. Каковы твои предпочтения относительно создания объекта? Ты используешь функцию NEWOBJECT() или функцию CREATEOBJECT()?

Марсиа: По-разному... Я думаю, что factory-объект должен быть достаточно разумным, чтобы использовать функцию CREATEOBJECT() в том случае, если библиотека классов была указана явно, и функцию NEWOBJECT(), если библиотека еще не открыта. Кроме того, должна сказать, что я также вижу реализацию этого объекта такой, в которой используются два отдельных метода, New() и Create(). Вызывающая программа просто обращается к соответствующему методу.

Энди: Это представление ошибочно, потому что ты просто меняешь жестко программируемую комбинацию класс/библиотека на жестко программируемое обращение к методу. Самоочевидно, по-видимому, что ответственность за решение о том, как создается объект, лежит на «фабрике», а не на вызывающей программе. Я предпочел бы, чтобы разумной была эта «фабрика»!

Марсиа: Согласна. Знаю, не часто я могу сказать тебе, что ты прав, следовательно, пользуйся случаем!

Энди: И у меня есть еще один вопрос, пока мы не ушли от этой темы. Как насчет параметров?

Марсиа: Что ты понимаешь под этим: «Как насчет параметров»? «Фабрика» просто передает их объекту.

Энди: Нет, я имею в виду, как ты передаешь параметры «фабрике»? Ты используешь для этого объект параметров или список параметров, и, если ты используешь список, то сколько в нем может быть параметров?

Марсиа: Мое личное мнение таково, что если ты должен передать больше пяти параметров при инициализации, тебе следует пересмотреть свое проектное решение и использовать объект параметров. Поэтому, метод New() factory-класса деспотично принимает до пяти параметров.

Энди: Хорошо, для многих это что-то вроде вопроса религиозной веры, но я, конечно, могу пока прожить с таким определением. Мы всегда можем увеличить допустимое число параметров позже, если в этом действительно возникнет необходимость, в конце концов, такое изменение повлияет только на предложение LPARAMETERS в единственном методе.

Марсиа: Ну что ж, рада, что мы с этим разобрались. Можем ли мы приступить к проектированию метаданных, которые должны содержать определения классов? Очевидно, нам потребуются поля для ключевого слова, имен класса и библиотеки класса. Я всегда добавляю в таблицу memo-поле properties, чтобы иметь возможность обрабатывать любую дополнительную информацию, вроде имени разработчика, истории изменений и так далее.

Энди: Вообще говоря, я предпочел бы использовать это memo-поле properties для хранения используемых по умолчанию значений свойств, а для дополнительной информации добавить еще одно memo-поле notes.

Марсиа: Зачем? Если тебе необходимы значения, используемые по умолчанию, почему бы не задать их в определении класса? Или, если это значения инициализации, они могут быть просто переданы как параметры.

Энди: Хорошо, и тут я рассуждаю теоретически, предположим, у тебя есть класс, которому необходим набор значений, используемых по умолчанию, и таких наборов у него может быть несколько. Вместо того, чтобы передавать кучу параметров, мы могли бы определить различные ключевые слова и связать их не только с самим классом, но также с набором используемых по умолчанию значений.

Марсиа: Конечно, мы можем это сделать. Я полагаю, что при этом мы извлекаем дополнительную выгоду, состоящую в том, что если свойства или их значения по умолчанию меняются, нам не надо разыскивать их в коде. Все они находятся в метаданных. Да, мне нравится такое решение. Структура метаданных представлена в таблице 1.

Обратите внимание на то, что это свободные таблицы. Поэтому в поле Properties не используется префикс (это поле уже имеет длину в 10 символов), и поэтому в этих таблицах нет первичных ключей.

Энди: Но зачем тебе нужны две идентичные таблицы? Я думаю, нам потребуется только одна.

Таблица 1. Метаданные Factory-класса (таблицы classes.dbf и wipclasses.dbf).

Поле	Тип	Размер	Описание
cKey	C	20	Ключевое слово, используется для поиска подробного описания класса. Определено как Candidate Key.
cClassname	C	50	Имя класса, экземпляр объекта которого надо создать.
cLibrary	C	50	Имя библиотеки класса, которую надо использовать. (Замечание: предполагается, что пути доступа указываются в приложении, следовательно здесь хранится только имя библиотеки, и оно должно включать расширение файла.)
Properties	M	4	Список имен свойств и тех значений, которые должны использоваться. Данные для этого поля вводятся в формате: “имя свойства = значение”. Например: cName = Testing Application lSetFlag = .T. nValue = 123.45 Замечание: этот класс использует функцию TYPE() для того, чтобы определить тип данных свойства, поэтому все свойства должны быть соответственным образом инициализированы в определении класса.
mNotes	M	4	Свободное текстовое поле для записи любой дополнительной информации, связанной с этим классом, например, история внесения изменений или сведения о разработчике.

Марсиа: О да, нам нужна только одна таблица, но я предпочитаю иметь таблицу classes.dbf для хранения прошедшего тестирование, готового к использованию программного кода, работа над которым завершена, а другую таблицу использовать в процессе разработки. Код из factory-класса всегда сначала проверяет таблицу wipclasses, так что мне, собственно, не надо что-то модифицировать в этой таблице до тех пор, пока я не буду уверена в правильности новой версии. Это также имеет тот смысл, что при коллективной разработке каждый разработчик может иметь собственную локальную таблицу и используемую в режиме «только для чтения» копию основной библиотеки classes.dbf. Таким образом, когда каждый программист разрабатывает класс, он совершенно не влияет на чей-то еще программный код, пока не отладит свои изменения.

Энди: Это полезное дополнение. Я не додумался об использовании этого класса в условиях коллективной разработки! Итак, как тогда насчет программного кода? Очевидно, мы собираемся использовать в качестве базового класс Session, чтобы factory-класс мог открыть свои метаданные в частной сессии данных и можно было работать в режиме EXACT. Это означает, что мы должны дать определение этого класса в программе PRG (поскольку мы не можем использовать базовый класс Session в визуальном конструкторе классов). Насколько я понимаю, для factory-класса потребуется всего лишь один открытый метод New, и у него нет вообще никаких свойств.

Таблица 2. Интерфейс Factory-класса.

Метод	Описание
Init	Встроенный метод Init(). Открывает таблицу Classes.dbf. Если она найдена, открывает таблицу WipClasses.dbf и присваивает свойству lWipTable соответствующее значение.
New	Открытый метод, который получает ключевое слово и до пяти параметров. Если ключевое слово найдено в одной из таблиц с определениями классов, создается экземпляр объекта этого класса, и ссылка на этот объект возвращается в вызывающую программу. В противном случае, возвращается значение NULL.
ChkLibType	Защищенный метод, вызываемый из метода New(), который гарантирует, что библиотека классов имеет расширение. Если расширение не было указано, метод пытается найти файл типа VCX и, если попытка провалилась, пытается найти файл типа PRG. Если все попытки закончились неудачей, возвращает пустую строку.
GetProperties	Защищенный метод, вызывается из метода New() в том случае, если поле свойств не пустое. Выполняет проверку существования указанных имен свойств в переданном объекте, и, если возможно, присваивает значения.
Str2Exp	Защищенный метод, вызывается из метода GetProperties() с представлением значения в виде символьной строки и требуемым типом данных. Возвращает значение указанного типа данных.

Марсиа: Почти, но не совсем, верно. Нам необходимо одно свойство lWipTable, которое получает свое значение в методе Init() factory-класса в зависимости от того, существует ли таблица wipclasses. Понятно, что при эксплуатации мы не хотели бы использовать таблицу wipclasses.dbf и разрушить программный код.

Энди: Это представляется в высшей степени разумным. Следовательно, теперь мы можем определить метод для factory-класса, как показано в таблице 2.

Марсиа: Кажется, все правильно. Этот программный код в действительности достаточно прямолинеен, кроме двух исключений. Во-первых, код, непосредственно выполняющий создание объекта, должен быть запрограммирован по-разному, в зависимости от того, собираемся ли мы использовать функцию CREATEOBJECT() или функцию NEWOBJECT(). Вот соответствующая часть этого кода:

```
*** Теперь посмотрим, что следует
*** использовать, prg-файл или vcx-библиотеку
 IF lcLibType = 'PRG'
   llCreate = ( lcLibrary $ SET( 'PROCEDURE' ) ) OR ;
     (FORCEEXT( lcLibrary, 'FXP' ) $ SET( 'PROCEDURE' ))
 ELSE
   llCreate = lcLibrary $ SET( 'CLASSLIB' )
 ENDIF

 IF llCreate
   lcCommand = 'CreateObject( "' ;
             + lcClassName ;
             + '"'
 ELSE
   lcCommand = 'NewObject( "' ;
             + lcClassName ;
             + '", "' + lcLibrary + '"'
 ENDIF

 *** дописать все параметры в конец команды
 lnParmCount = PCOUNT()-1
 IF  lnParmCount > 0
   *** Добавить только третий параметр, если мы
   *** используем функцию NewObject()
   IF NOT llCreate
     lcCommand = lcCommand + ', ""'
   ENDIF
   FOR lnParm = 1 TO lnParmCount
     lcCommand = lcCommand + ', tuParam' ;
       + TRANSFORM( lnParm )
   ENDFOR
 ENDIF
 lcCommand = lcCommand + ' )'

 *** Идем дальше и создаем экземпляр объекта
 loObject = &lcCommand
```

Энди: Да, стоит, однако, сделать это, чтобы избежать накладных расходов, неизбежных при постоянном использовании одной только функции NEWOBJECT(). Камнем преткновения является то, что даже если требуемая библиотека классов уже открыта, функция NEWOBJECT() прежде всего запоминает установки, выполненные командами SET("CLASSLIB") и SET("PROCEDURE"), а затем закрывает все открытые библиотеки, открывает ту, которая необходима для создания именно этого нового объекта, а затем возвращает все библиотеки в их исходное состояние. Не то, чтобы эти накладные расходы были очень уж велики, но их можно избежать, если мы сами проверим, открыта ли необходимая библиотека классов.

Марсиа: Еще одна «хитрость» присутствует в методе GetProperties() для указания значений, используемых по умолчанию. Нам придется рассчитывать на функцию TYPE() при определении (по свойству нового объекта) какого типа данные представлены в memo-поле (поскольку все значения в действительности являются символьными строками). Это означает, что если в определении класса свойства были инициализированы некорректно, мы могли бы получить сообщения об ошибках. Ты уверен, что за решение этой задачи отвечает именно factory-класс? Не можем ли мы просто скопировать содержимое поля Properties в свойства нового объекта и предоставить ему право улаживать все это?

Энди: Нет, я действительно думаю, что эта работа должна выполняться в factory-классе. Поскольку «фабрика» несет ответственность за создание объекта, именно она должна определить значения, используемые по умолчанию, если в них есть необходимость. В противном случае, нам пришлось бы добавить метод в родительский класс, чтобы разрешить эту проблему, и обращаться к этому методу после того, как «фабрика» вернет созданный объект.

Марсиа: Принимаю! Даже с учетом сказанного, программный код для factory-класса не должен быть слишком уж длинным, но такой подход только в очередной раз доказывает насколько просто организовать управление с помощью данных средствами VFP. Кроме того, эту «фабрику» так легко использовать! Все, что необходимо сделать, — это создать экземпляр объекта класса при загрузке приложения и обеспечить гарантию того, что этот объект находится в области видимости. Например, ты мог бы сохранить ссылку на этот объект в частной переменной в программе загрузки, или в свойстве твоего «крестного» объекта (ой! Я имею в виду объект приложения) или в свойстве объекта _Screen.

```
PRIVATE poFactory
poFactory = NEWOBJECT( 'Factory', 'Factory.prg' )
```

Энди: А чтобы создать объект, необходимо только добавить запись с соответствующими данными в таблицу классов, а затем вызвать метод New() factory-класса, передав ему ключевое слово.

```
loForm = Factory.New( 'Test' )
```

Видишь, до какой степени универсальным становится твой код, если ты используешь «фабрику»? Код класса и соответствующие таблицы метаданных можно найти в файле-приложении к данной статье.

❑❑❑

Передача данных с уровня на уровень, часть 2

Эрик Мур (Erik Moore)

Одним из наиболее запутанных вопросов разработки n-уровневой (n-tier) модели является передача данных с уровня на уровень. В прошлом месяце Эрик Мур рассмотрел одну из двух основных возможностей решения этой задачи — использование спецификации XML. В этом месяце он рассматривает другой вариант.

Прежде чем у вас явится мысль о том, что вся дискуссия по поводу n-уровневой архитектуры сводится к тому, как использовать курсоры, созданные на сервере, в клиентском приложении, вспомните, что курсоры — это не единственное средство для работы с данными в VFP-клиенте.

Это правда, что при формировании HTML-кода для представления Web-страницы конечный результат остается одним и тем же, независимо от того, что является источником данных: курсор, строка или объект. Отличается только программный код, используемый для формирования HTML-кода, а первейшей заботой является производительность.

«Толстые» клиенты, например VFP-приложения, усугубляют эту картину.

За исключением элемента управления grid, представленные в VFP элементы управления, предназначенные для ввода данных, могут быть связаны (bound) практически с любым значением, включая поля в курсоре, открытые (public) или частные (private) переменные или свойства некоторого объекта. Элементу управления это безразлично. Ввиду сказанного, нет необходимости хранить данные на стороне клиента непременно в курсоре. Вы могли бы использовать для этого метод формы, который получает бизнес-объект, загружаемый вместе со своими свойствами, соответствующими полям связанной с этим объектом таблицы.

Вдаваясь в детали, вы могли бы поместить некоторый программный код в событие Init формы для обращения к COM-объекту, который возвращает объект Customer, и сохранить объект Customer в свойстве этой формы. Затем связать свойства объекта Customer с источниками данных (control sources) различных элементов управления, размещенных в этой форме. Предполагая, что у нас есть форма, для которой определено пользовательское свойство (custom property), где будет храниться ссылка на запись customer, мы можем использовать вот такой программный код для «настройки» формы с элементами управления text box, привязанными к объекту record:

```
* Создать экземпляр объекта класса бизнес-объекта
oBiz = CREATEOBJECT("MyApplication.Customer")
* Вызвать метод GetById, передав ему в качестве параметра
* первичный ключ записи в таблице customer
THISFORM.oRecord = oBiz.GetByID(12345)
* Связать размещенные в форме элементы управления
* со свойствами объекта record
THISFORM.txtFName.ControlSource = "THISFORM.oRecord.Fname"
THISFORM.txtLName.ControlSource = "THISFORM.oRecord.Lname"
```

При наличии в форме элементов управления text box, связанных со свойствами объекта record, изменение значения в текстовом поле приведет к изменению значений свойств объекта. После того, как пользователь закончил редактирование полей формы, он щелкает мышью по кнопке Save, чтобы вызвать метод Save формы, возвращающий объект обратно в метод Save COM-объекта:

```
oBiz = CREATEOBJECT("MyApplication.Customer")
IF oBiz.Save(THISFORM.oRecord)
  =MESSAGEBOX("Update succeeded")
ELSE
  =MESSAGEBOX("Update failed")
ENDIF
```

Многие n-уровневые приложения были написаны с использованием аналогичных схем (более вероятно, на других языках программирования, нежели на VFP). Это простое и изящное решение до тех пор, пока вы не начнете понимать, что ситуация существенно усложняется с введением понятия дочерних записей. Если вы пытаетесь аккуратно «упаковать» свои бизнес-элементы предметной области в пользовательские объекты, тогда естественным способом реализации дочерней таблицы было бы представление дочерних элементов как членов коллекции родительского объекта. Попытка реализации такой схемы в VFP наталкивается на две проблемы.

Во-первых, созданные в VFP объекты не обеспечивают встроенной поддержки коллекций. Разумеется, вы можете использовать объект Container для создания объекта верхнего уровня и с помощью метода AddObject добавить к нему всех «детей», но это будут «дети», а не члены коллекции. Если у объекта есть «дети» двух разных типов (скажем, счета и платежи), тогда все эти дочерние записи оказались бы перепутаны между собой. Способы «имитировать» коллекции в VFP есть, но они не особенно удачны.

Вторая проблема — это вопрос о том, как представить дочерние объекты на экране в форме. Даже

если бы VFP-объекты поддерживали коллекции, это не отменяет того, что принадлежащий VFP элемент управления grid понятия не имеет о том, как отображать эти коллекции на экране. Чтобы заставить grid отображать на экране ваши объекты, пришлось бы «разнести» каждый объект по записям курсора, и тогда мы возвращаемся к тому, с чего начали, — использование курсора в клиентском приложении.

Непроизводительные издержки — это еще одна проблема, требующая рассмотрения перед тем, как принимать решение об использовании объектов на стороне клиента. Хотя создание и подключение встроенного объекта, как правило, реализованы эффективно, возможны значительные потери производительности в том случае, если COM-объекты замещаются бизнес-объектами. Создание объектов и однократная проверка свойств проблем не создает, но любая операция с циклическим перебором множества записей либо вызовом множества методов или заданием значения для множества свойств, может оказаться тяжким бременем, поскольку каждое обращение должно пересекать границы COM-объектов. Не забывайте, впрочем, о том, что производительность — понятие относительное.

Идеальное решение обеспечить возможность использования объекта для представления данных на стороне клиента — это такое решение, которое естественным образом поддерживает множественные «записи» и может быть представлено на экране в некоторого рода элементе пользовательского интерфейса UI без преобразования.

ADO

И такое решение у нас есть! Возможно, самой распространенной альтернативой курсору в клиентском приложении является объект Recordset модели ADO. Объектная модель ADO (ActiveX Data Objects) — это попытка фирмы Microsoft обеспечить универсальное средство обмена данными. Основная идея, лежащая в основе модели ADO, сводится к тому, что данные прекрасно можно упаковывать в объектах, в которых они могут передаваться, обрабатываться и даже сохраняться автоматически. Имеется пара причин, по которым модель ADO является темой, имеющей отношение к данной статье:

- Объектная модель ADO — это рекомендованная фирмой Microsoft стратегия доступа к данным.
- Объект Recordset модели ADO — это отличная упаковка для обмена данными между объектами.

По существу, модель ADO служит оболочкой интерфейса OLEDB той же фирмы Microsoft. OLEDB — это набор интерфейсов на базе COM-модели, которые обеспечивают последовательный унифицированный доступ к данным, хранящимся в разнообразных источниках данных. Разработчик может использовать модель ADO для доступа к любому хранилищу данных, для которого существует провайдер OLEDB или ODBC-драйвер, поскольку имеется провайдер OLEDB для ODBC-источников и в эту категорию попадает большинство баз данных.

Модель ADO состоит из нескольких различных объектов. Описание некоторых из наиболее важных объектов приводится далее.

Объект Connection

Объект Connection используется для того, чтобы задать значения свойств соединения, таких как имя OLEDB-провайдера, служащая источником база данных, имя пользователя и пароль. Эти параметры могут быть определены явно как свойства соединения или в строке соединения, определяемой в свойстве ConnetionString. После того, как соответствующие свойства получили свои значения и был вызван метод Open, объект Connection может быть использован объектом Command для выполнения каких-либо действий с базой данных.

Объект Command

Объект Command — это рабочая лошадка для объектов модели ADO, и он используется для выполнения команд применительно к базе данных. У него есть коллекция параметров, которая может быть использована для определения значений входных параметров хранимых процедур и чтения возвращаемых значений и выходных параметров. Метод Execute этого объекта посылает команду на сервер (back end) и может возвращать объект Recordset.

Объект Recordset

Табличное представление данных, возвращаемых в качестве результата обработки запроса или как результат работы хранимой процедуры; реализовано в виде набора записей. Эквивалент VFP-курсора в модели ADO — объект recordset — имеет методы, которые аналогичны командам и функциям VFP: SKIP, GO TOP, GO BOTTOM, LOCATE, APPEND BLANK, DELETE, EOF() и TABLEUPDATE().

Пример использования модели ADO

Чтобы продемонстрировать, как применять модель ADO в VFP, я воспользуюсь объектами Connection, Command и Recordset для организации доступа к данным, хранящимся в БД Northwind (пример, который инсталлируется по умолчанию вместе с SQL Server).

```
* Создать объект Connection
 oConnection = CREATEOBJECT("ADODB.Connection")
 * Указать провайдера OLEDB для SQL Server
 oConnection.Provider = "SQLOLEDB.1"
 * Указать сервер на локальной машине
 oConnection.Properties("Data Source").Value = "(local)"
 * Указать базу данных, используемую по умолчанию
 oConnection.Properties("Initial Catalog").Value ;
    = "Northwind"
 * Указать имя пользователя и пароль
 oConnection.Properties("User ID").Value = "sa"
 oConnection.Properties("password").value = ""
 * Открыть соединение
 oConnection.Open()

 * Создать объект Command
 oCommand = CREATEOBJECT("ADODB.Command")
 * Сообщить ему, что для связи необходимо использовать
 * объект connection
 oCommand.ActiveConnection = oConnection
 * Выбрать всех поставщиков, чьи названия фирм
 * начинаются с буквы 'E'
 oCommand.CommandText = "Select * FROM suppliers ;
   WHERE CompanyName LIKE 'E%'"
 oRS = oCommand.Execute()

 * Перейти к первой записи полученного в результате
 * набора записей
 oRS.MoveFirst()
 * Циклический перебор всех записей.
 * Эта конструкция цикла аналогична конструкции SCAN в VFP
 DO WHILE !oRS.EOF()
   * Получить значение поля 'CompanyName'
   lcCompanyName = oRS.Fields("CompanyName").Value
   =MESSAGEBOX("Company: " + lcCompanyName)
   * Перейти к следующей записи
   oRS.MoveNext()
 ENDDO
```

Использование строки соединения

Большая часть работы в предыдущем фрагменте программного кода приходится на настройку объекта Connection. Я решил указать значения конкретных свойств объекта Connection, чтобы продемонстрировать вам, как это делается, но я мог бы просто указать строку соединения, в которой определены все эти значения:

```
oConnection.ConnectionString = ;
   "Provider=SQLOLEDB.1; User ID=sa;
   Initial Catalog=Northwind; DataSource=(local)"
```

Строка соединения может выглядеть пугающе, но есть несложная уловка, с помощью которой можно позволить операционной системе Windows сформировать для вас строку соединения на этапе проектирования.

- 1. Создайте пустой файл в любом каталоге и определите для него расширение UDL.
- 2. Щелкните правой кнопкой мыши по этому файлу и выберите в контекстном меню пункт Open.
- 3. Воспользуйтесь редактором Data Link, чтобы визуально выбрать провайдера, базу данных и информацию для регистрации в системе.
- 4. Щелкните мышью по кнопке Close.
- 5. Откройте созданный UDL-файл в блокноте Notepad.

UDL-редактор создал для вас строку соединения и поместил ее в этот файл, следовательно вы можете прибегнуть к операции «вырезать и вставить», чтобы использовать эту строку для своих нужд. Кроме того, для конфигурации соединения можно использовать непосредственно UDL-файл, указав путь доступа к этому файлу в строке соединения:

```
oConnection.Open("File Name=c:\mydatalink.udl")
```

Исполнение хранимых процедур

Объект Command был использован в данном примере для выполнения на сервере команды языка запросов SQL, но с его помощью можно также исполнять хранимые процедуры. Если бы у нас была хранимая процедура CompanyGetByName, получающая название фирмы в качестве параметра, мы могли бы использовать объект Command для обращения к этой процедуре следующим образом:

```
oCommand.CommandText = "Exec CompanyGetByName ;
  @CompanyName='Microsoft'"
oRS = oCommand.Execute()
```

Кроме того, есть очень мощный объект Parameter, который может быть использован для задания входных и получения выходных параметров и возвращаемых значений хранимых процедур. Вы могли бы использовать его вместо параметров в свойстве CommandText.

```
WITH oCommand
   .CommandType = 4 && adCmdStoredProc
   .CommandText = "CompanyGetByName"
   .Parameters("@CompanyName").Value = 'Microsoft'
   oRS = .Execute()
ENDWITH
```

Такой способ назначения параметров может оказаться очень удобным в том случае, когда у вас есть набор классов, назначающий параметры автоматически, и вам необходимо выполнить итеративный перебор параметров, чтобы получить доступ к ним по имени.

Если вызываемая вами хранимая процедура создает выходные параметры, или вы хотите выполнить проверку возвращаемого значения, использование объекта Parameter — это правильный путь. Предположим, что у вас есть хранимая процедура OrderCountByCustomer, которая подсчитывает количество заказов, сделанных заказчиком. Эта процедура могла бы иметь выходной параметр @OrderCount. Вы вызывали бы эту процедуру и выполняли проверку выходного параметра следующим образом:

```
WITH oCommand
   .CommandType = 4 && adCmdStoredProc
   .CommandText = "OrderCountByCustomer"
   .Parameters("@CustomerID").Value = 2
   .Execute()
   lnOrderCount = .Parameters("@OrderCount").Value
ENDWITH
```

Таблица 1. Многие, часто используемые, методы и свойства модели ADO эквивалентны знакомым VFP-командам.

Метод модели ADO	Описание	Эквивалент в VFP
MoveFirst	Перемещает указатель записи к первой записи в наборе.	GO TOP
MoveNext	Перемещает указатель записи к следующей записи в наборе.	SKIP 1
MovePrevious	Перемещает указатель записи к предыдущей записи в наборе.	SKIP -1
MoveLast	Перемещает указатель записи к последней записи в наборе.	GO BOTTOM
AddNew	Добавляет новую запись в набор записей.	APPEND BLANK
Delete	Удаляет запись из набора записей.	DELETE
BOF	Возвращает значение true, если указатель записи находится в начале набора.	BOF()
EOF	Возвращает значение true, если указатель записи находится в конце набора.	EOF()
Move	Перемещает указатель записи на указанное количество записей.	SKIP
Find	Перемещает указатель к первой записи, которая отвечает заданному критерию.	LOCATE
Seek	Использует текущий индекс для поиска записи.	SEEK
Save	Сохраняет изменения, внесенные в набор записей, в связанной с этим набором таблице.	TABLEUPDATE()
Requery	Использует запрос (query), который создает набор записей для обновления их содержимого данными из источника данных.	REQUERY()

Кстати, если вам необходимо выполнить проверку значения, возвращаемого хранимой процедурой, вы просто проверяли бы 0-й параметр:

```
lnReturnValue = oCommand.Parameters(0).Value
```

Объект recordset может показаться вам знакомым

Для тех запросов и хранимых процедур, которые возвращают результаты в табличном виде, «упаковкой» является набор записей — recordset. Как я уже намекал ранее, объект Recordset — это аккуратное и универсальное транспортное средство. Его использование должно быть отдаленно знакомо VFP-программисту, в конце концов, объект Recordset модели ADO проектировался с использованием механизма VFP-курсора в качестве модели. Объект recordset имеет массу свойств, и данная статья не претендует на то, чтобы быть исчерпывающим руководством, поэтому я собрал некоторые ключевые методы и свойства в таблице 1 вместе с указанием эквивалентных встроенных команд VFP.

Итак, что же это будет: объект или строка?

Теперь, когда я рассказал о двух транспортных средствах, которые вы можете использовать для передачи данных с одного уровня на другой, вам придется принять решение о том, какое из этих средств является наиболее подходящим для вашего приложения. Какого-то одного самого лучшего формата данных, пригодного для всех приложений, не существует, поскольку и модель ADO, и спецификация XML имеют, каждая, свои недостатки и достоинства.

Хотя модель ADO, безусловно, обеспечивает инкапсуляцию большого объема функциональных возможностей и сравнительно проста в использовании, ей присущи некоторые ограничения. Во-первых, объект Recordset модели ADO — это COM-объект. Вспомните о том, что в n-уровневой архитектуре нам необходимо иметь возможность разместить различные уровни на разных машинах в целях обеспечения масштабируемости, а передача COM-объекта с одной машины на другую означает использование модели DCOM. Тем самым на условия эксплуатации приложения накладывается ряд ограничений:

- Необходимо, чтобы на обеих машинах исполнялись COM-совместимые платформы. Как правило, это означает использование операционной системы Windows.
- Эти машины должны быть постоянно подключены к сети.
- Права доступа к серверной машине должны иметь такую конфигурацию, которая позволяет обращаться к этому объекту. Поскольку модель DCOM использует свой собственный набор TCP-портов, она не сможет шагнуть за брандмауэр (firewall), если только этот брандмауэр не будет целенаправленно сконфигурирован так, чтобы позволить трафик через эти порты.
- В контролируемой корпоративной среде, эти барьеры, как правило, можно преодолеть. Но, зачастую, это потребует расходов, связанных с инсталляцией и устранением конфликтов. Такого рода приложения вряд ли можно определить термином «plug-n-play».

С другой стороны, текст (и, следовательно, спецификация XML) вполне дружелюбен по отношению к брандмауэрам, поскольку он может передаваться через те порты, которые используют Web-браузеры

(порт 80 для протокола HTTP и порт 443 для протокола SSL). Практически все корпоративные брандмауэры оставляют эти порты открытыми с тем, чтобы находящиеся за ними пользователи могли иметь доступ к Web. Если ваше приложение использует формат данных, который является дружественным по отношению к протоколу HTTP, вам не придется беспокоиться и обращаться к «ребятам-железячникам» с просьбой «пробить брешь» в брандмауэре для вашего приложения.

Настольное приложение, спроектированное для работы с Internet в качестве сети, открывает массу возможностей. Путешествующий торговый агент в гостиничном номере в Берлине легко может получить доступ к базе данных, размещенной на сервере в Нью-Йорке. Служащие, работающие в режиме дистанционного присутствия на рабочем месте (telecommuting employees), получают доступ к данным фирмы из дома, используя то же самое приложение, с которым они работают в офисе. Функциональность такого типа просто невозможна в условиях модели DCOM.

Спецификация XML также имеет свои ограничения. Во-первых, она многословна. Каждый элемент данных в XML-документе обрамляется парой тэгов, описывающих эти данные. Для набора, объемом в 100 записей, имеющих по 15 полей, это минимум 3 000 открывающих и закрывающих тэгов, большинство из которых повторяется. (Впрочем, именно эта характеристика делает XML-формат исключительно сжимаемым; в зависимости от спецификации XML и используемого алгоритма сжатие может составлять 3-15% от их исходного размера).

Кроме того, использование спецификации XML требует ресурсы для выполнения парсинга. Загрузка большого документа в DOM-парсер может оказаться заметной нагрузкой на машину. Если ваше приложение использует в качестве транспортного средства спецификацию XML и имеет дело с наборами данных, пусть даже средних размеров, вы увидите снижение производительности, которое прямо можно отнести на счет парсинга. Парсер фирмы Microsoft достаточно быстр, и он становится все быстрее и быстрее с появлением каждой новой версии, но нельзя обойти стороной тот факт, что он должен загрузить в память каждую вершину документа как объект, прежде чем можно будет осуществить разбор этого документа или его фрагмента.

Для обычного приложения баз данных, наборы данных большого объема не являются проблемой, особенно, если это приложение с самого начала проектировалось с учетом концепций клиент/серверной архитектуры. Форма для ввода данных в типичном приложении могла бы загружать одну первичную запись, два или три дочерних курсора с парой дюжин записей в каждом и, может быть, пару справочных списков. Такая форма, которая получает свои данные в виде XML-документа, не загрузится в мгновение ока, но вы и не сможете никуда отлучиться, чтобы перекусить. Другими словами, время ее загрузки будет приемлемым.

Отсюда следует вывод, что обе рассмотренные технологии являются достаточно ценными инструментальными средствами, чтобы иметь их в своем арсенале.

❑❑❑

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

FoxTalk (русское издание) индекс 72495

Объединенный каталог индекс 45007

Журнал для FoxPro-программистов.

FoxTalk (русское издание) индекс 72496

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

FoxTalk (русское издание) индекс 72497

Подписка на старые номера журнала FoxTalk.

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

Книги компьютерной тематики по последним версиям популярных программных продуктов.

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

Подписано в печать 10/02/03. Формат 60x90 1/8. Тираж 330 экз.