FoxTalk

Апрель 2003
№ 4 (70)

русское издание

Solutions for Microsoft® FoxPro® and Visual FoxPro® Developers

Создание CHM-файлов из Word-документов, часть 2

Дуг Хенниг (Doug Hennig)

Во второй части этой серии публикаций, состоящей из двух статей, Дуг Хенниг рассматривает основы создания CHM-файлов, а затем, используя классы, представленные в прошлом месяце, автоматизирует процесс формирования CHM-документа из набора HTML-файлов, которые в свою очередь получены из Word-документов.

материал имеет отношение к соответствующей версии

материал имеет отношение к соответствующей платформе

исходные тексты программ можно скачать из Интернета

В прошлом месяце мы рассматривали процедуры, необходимые для того, чтобы преобразовать набор Word-документов в HTML-файлы, содержащие минимум HTML-кода, свободного от разного рода «излишеств». Такие HTML-файлы полезны и сами по себе, если вы размещаете их на Web-сайте или на компакт-диске. Однако, главное их назначение — это использование в файле справочной системы HTML Help (CHM-файле). В данной статье основное внимание уделяется тем шагам, которые необходимо предпринять для решения этой задачи. Мы продолжим процесс преобразования моих статей из журнала FoxTalk, публиковавшихся в нем начиная с 2000 года, в CHM-файл, хотя я также прибегну — там, где это будет уместно, — к некоторым примерам, связанным с подготовкой книги *The Hacker's Guide to Visual FoxPro 7*.

Рассказывая о процедурах, лежащих в основе создания CHM-файла, я рассмотрю различные файлы, которые используются для формирования этого справочного файла. Я буду обсуждать лишь некоторые вопросы, а не все доступные возможности. Чтобы получить более подробную информацию о CHM-файлах, загрузите документацию для справочной системы HTML Help с Web-сайта сети разработчиков MSDN (зайдите по адресу http://msdn.microsoft.com, выполните поиск строки "HTML Help", в полученном списке ссылок выберите страницу загрузки Microsoft HTML Help Downloads, а на этой странице выберите ссылку HTML Help Authoring Guide).

Есть только два дополнения к тем инструментам, которые мы будем рассматривать в этом месяце: программа CreateCHM.PRG, выполняющая всю необходимую работу, и таблица Contents.DBF, определяющая, как надо генерировать CHM-файл. Прежде чем мы изучим программу CreateCHM.PRG, давайте рассмотрим таблицу Contents.DBF.

Таблица 1. Структура таблицы содержания.

Имя поля	Тип/Размер	Описание
SECTION	I	Структура узлов самого верхнего уровня. Например, статьи из журнала FoxTalk могут быть организованы по годам, так что в этом поле могли бы храниться даты — 1999, 2000, 2001 и так далее.
ORDER	I	Порядок узлов в разделе.
LEVEL	N(2, 0)	Уровень узла: 0 — узел верхнего уровня, 1 — первый дочерний уровень нулевого уровня, 2 — следующий дочерний уровень и так далее.
NAME	C(120)	Заглавие в том виде, как оно должно быть представлено в элементе управления TreeView.
INDEX	M	Ключевые слова для CHM-индекса, каждое слово в отдельной строке. Если это memo-поле пусто, тогда для данного тематического раздела в качестве элемента индекса используется значение из поля NAME.
FILENAME	C(60)	Имя HTML-файла.
IMAGE	N(2, 0)	Номер CHM-изображения, используемого для данного узла, например, номер 5 определяет изображение папки, а номер 11 — документа. Список изображений и их номера ищите на вкладке Advanced в диалоге Table of Contents, предусмотренном для темы в приложении HTML Help Workshop.
HELPID	I	Идентификатор справочного контекста (help context ID) для данной темы.
INTOC	L	.T., если эта тема должна появиться в элементе управления TreeView.
ININDEX	L	.T., если эта тема должна появиться в индексе.
DEFAULT	L	.T., если эта тема автоматически выбирается при открытии CHM-файла.

Таблица 2. Эти записи описывают содержание CHM-файла для журнала FoxTalk.

Порядок	Уровень	Имя	Изображение	Intoc	Inindex
0	0	FoxTalk 2000	5	T	F
10	1	Persistence without Perspiration	11	T	T
20	1	Get the Message?	11	T	T
30	1	Spam, Wonderful Spam	11	T	T
40	1	Manage Your Applications 11	T	T	
75	1	Updating an Application over the Internet	5	T	F
80	2	Updating an Application over the Internet, Part I	11	T	T
90	2	Updating an Application over the Internet, Part II	11	T	T
100	1	File Your Collections	11	T	T

Таблица содержания

В таблице Contents.DBF определяются тематические разделы (topics), которые появятся в CHM-файле, иерархия древовидной структуры и порядок следования тематических разделов. Эта таблица имеет структуру, представленную в таблице 1.

В таблице 2 показаны несколько записей из таблицы содержания, предназначенной для CHM-файла, со статьями из журнала FoxTalk (для краткости представлены только отдельные записи и отдельные поля). Все записи имеют в поле SECTION значение 2000, поскольку эти статьи были опубликованы в 2000 году, а мы организуем CHM-файл по годам. Я использовал в поле ORDER значения, которые возрастают с шагом 10, что должно облегчить в будущем вставку новых записей: при этом устраняется необходимость перенумеровывать все последующие записи (как вы можете догадаться, я вставил запись с полем ORDER = 75 после того, как были созданы все остальные записи). Первая запись принадлежит уровню 0, имеет поле IMAGE = 5 (папка) и не появится в индексе, следовательно, понятно, что это узел верхнего уровня, который сам по себе не слишком содержателен. Следующие четыре записи являются «рядовыми» узлами, каждый из которых соответствует одной статье. Следующая запись, с полем ORDER = 75, аналогична первой записи (IMAGE = 5 и ININDEX = .F.), поскольку это родительский узел для статьи, состоящей из двух частей. Соответствующие этим статьям записи имеют в поле LEVEL значение 2, так что они будут появляться под родительским узлом. Последняя запись возвращается на уровень 1, следовательно, она снова окажется дочерним элементом для первой записи.

При желании вы могли бы создать и заполнить эту таблицу вручную, но это было бы скучным занятием, особенно, если имеется много документов. Один из обрабатывающих классов, который я не рассматривал в прошлый раз, — это класс UpdateContentsTable. Если вы добавляете запись для этого класса в таблицу Process.DBF (таблицу драйверов

процесса), то он займется созданием таблицы Contents.DBF (если это необходимо) и будет добавлять в нее запись для каждого сформированного HTML-документа. Впоследствии вам придется «чистить» эту таблицу, меняя, например, значения в полях SECTION, LEVEL и INDEX в соответствии с потребностями, но, тем не менее, класс UpdateContentsTable выполняет большой объем работы по созданию таблицы Contents.DBF.

Обратите внимание на то, что связи между HTML-файлами и тематическими разделами, перечисленными в таблице содержания, не обязательно являются связями «один-к-одному». Например, в некоторых из файлов, предназначенных для книги HackFox, рассматриваются не одна, а несколько команд или функций языка программирования FoxPro. Поскольку каждая команда или функция должна присутствовать в элементе управления TreeView CHM-файла, в таблице содержания на один и тот же файл указывают несколько записей, хотя они и имеют различные значения в полях ORDER и NAME.

CreateCHM.PRG

Как только вы сформировали HTML-файлы и заполнили таблицу содержания, вы готовы к исполнению программы CreateCHM.PRG, которая создает CHM-файл. Эта программа получает три параметра: имя, включая путь доступа создаваемого CHM-файла; имя используемой таблицы стилей (CSS-файл) (пропустите этот параметр, если вы не используете CSS-файл); и заголовок для CHM-файла (если этот параметр пропущен, используется заголовок «No title specified»). Программа CreateCHM.PRG исходит из предположения, что HTML-файлы и все связанные с ними файлы находятся в том каталоге, который вы указали для CHM-файла, поскольку в этом случае снимаются вопросы, связанные с определением пути доступа; при необходимости вы можете изменить эту настройку.

Для создания CHM-файла вам потребуется инсталлировать в своей системе приложение HTML Help Workshop. Если у вас еще нет этого приложения, можете скачать его последнюю версию со страницы Microsoft HTML Help Downloads на Web-сайте MSDN. Мы не будем использовать само по себе приложение HTML Help Workshop, а применим только входящий в его состав компилятор.

По сути, создание CHM-файла складывается из двух шагов: формирования файлов, необходимых компилятору HTML Help для выполнения его работы, и последующего обращения к этому компилятору для выполнения рутинной работы. Для работы компилятору необходимы, по крайней мере, четыре управляющих файла: файл проекта, включаемый файл (include file), файл содержания и индексный файл.

Создание файла проекта и включаемого файла

Аналогично тому, как в VFP файл исполняемого EXE-модуля формируется из файла VFP-проекта, CHM-файл создается из файла проекта HTML Help. Файл проекта, который имеет расширение HHP, по существу является INI-файлом с описанием настроек и файлов, образующих CHM-файл.

Вот фрагмент HHP-файла, предназначенного для CHM-файла журнала FoxTalk, который мы собираемся создать в этой статье.

```
[OPTIONS]
Compatibility=1.1
Compiled file=FoxTalk.chm
Contents file=FoxTalk.hhc
Index file=FoxTalk.hhk
Default topic=FOXTALK2000.HTML
Title=FoxTalk Articles
Default Window=HelpWindow
Auto index=Yes
Display compile progress=No
Full-text search=Yes
Language=0x409 English (United States)

[WINDOWS]
HelpWindow=,"FoxTalk.hhc","FoxTalk.hhk",
  "FOXTALK2000.HTML","FOXTALK2000.HTML",,,,,0x2520,,
  0x387e,,,,,0,,,

[FILES]
foxtalk.css
200001DHEN.HTML
200002DHEN.HTML

[ALIAS]
IDH_200001DHEN=200001DHEN.HTML
IDH_200002DHEN=200002DHEN.HTML

[MAP]
#include FoxTalk.h
```

Я не буду описывать все составные элементы этого файла, а расскажу только о самых важных. В секции Options параметры Contents File и Index File определяют имена файла содержания и индексного файла; мы рассмотрим эти файлы позже. Параметр Compiled File — это имя создаваемого CHM-файла. Параметр Default Topic содержит имя того HTML-файла, который автоматически выбирается при открытии пользователем CHM-файла; обычно это тематический раздел “welcome”, первый узел в CHM-дереве. Параметр Title — это заголовок для окна CHM-файла. Параметр Default Window — это имя окна, в котором отображается содержание; настройки для этого окна определяются в секции Windows. В секции Files перечисляются те файлы, которые будут компилироваться в CHM-файл. Все файлы, на которые есть ссылки во всех перечисленных в

этом списке файлах (например, графические файлы), также автоматически будут включены в результат, что очень здорово с точки зрения наших намерений, потому что в противном случае нам пришлось бы заниматься разбором всех HTML-файлов, чтобы определить какие еще файлы необходимо включить в состав CHM-файла. Секция Alias используется для определения идентификаторов справочного контекста (help context IDs) для каждого тематического раздела, но это делается окольным путем: вместо самого идентификатора ID указывается константа (например, "IDH_200001DHEN"), которая определена во включаемом файле как идентификатор справочного контекста ID для тематического раздела. Секция Map указывает на включаемый файл.

Вот предложения #DEFINE из включаемого файла (файл FOXTALK.H в данном случае), которые определяют идентификаторы справочного контекста для каждой темы. Итак, идентификатором темы для файла 200001DHEN.HTML будет значение 10, потому что такое значение указано в предложении #DEFINE для константы IDH_200001DHEN, а эта константа предусмотрена для данного HTML-файла в секции Alias файла проекта.

```
#define IDH_200001DHEN 10
#define IDH_200002DHEN 20
```

Чтобы создать файл проекта, программа CreateCHM.PRG начинает с того, что открывает таблицу Contents, используя для упорядочивания записей тэг Main. Этот тэг сортирует таблицу по полям SECTION и ORDER. Значение поля FILENAME из записи с полем DEFAULT = .T. запоминается в переменной lcDefault. Это то значение, которое послужит в качестве выходных данных в опции Default Topic. Нижеследующий фрагмент программного кода начнет затем процесс объединения текста для требуемого файла и сформирует секции Options, Windows и начало секции Files в HHP-файле.

```
set textmerge on to (lcHHPFile) noshow
 \\[OPTIONS]
 \Compatibility=1.1
 \Compiled file=<<justfname(lcCHMFile)>>
 \Contents file=<<justfname(forceext(lcCHMFile, 'hhc'))>>
 \Auto index=Yes
 \Default topic=<<lcDefault>>
 \Display compile progress=No
 \Full-text search=Yes
 \Index file=<<justfname(forceext(lcCHMFile, 'hhk'))>>
 \Language=0x409 English (United States)
 \Title=<<lcTitle>>
 \Default Window=HelpWindow
 \
 \[WINDOWS]
 \HelpWindow=,"<<justfname(forceext(lcCHMFile,'hhc'))>>",
 \\"<<justfname(forceext(lcCHMFile, 'hhk'))>>",
 \\"<<lcDefault>>","<<lcDefault>>",,,,,0x2520,,0x387e,,,,
 \\,0,,,
 \
 \[FILES]
 if not empty(lcCSSFile)
   \<<justfname(lcCSSFile)>>
 endif not empty(lcCSSFile)
```

Вслед за этим мы получим список уникальных имен файлов. Поскольку возможно, что один и тот же файл встречается в поле CONTENTS неоднократно (бесспорный факт в случае с книгой HackFox), мы создадим курсор с именами файлов и воспользуемся опцией GROUP BY 1 для создания списка уникальных значений.

```
select FILENAME from CONTENTS into cursor FILES ;
   group by 1
 scan
   \<<trim(FILENAME)>>
 endscan
 use
 select CONTENTS
```

Секции Alias и Map просты: константа, используемая для каждой темы в элементе управления TreeView, — это просто строка "IDH_", за которой следует имя файла, и эта константа ассоциирована с указанным файлом. H-файл имеет то же самое имя, что и файл проекта, но с расширением H. После этого мы закроем выходной файл.

```
\
\[ALIAS]
scan for INTOC
  lcFile = trim(FILENAME)
  \IDH_<<upper(juststem(lcFile)))>>=<<lcFile>>
endscan for INTOC
\
\[MAP]
\#include <<justfname(lcHFile)>>
set textmerge to
```

Программный код для создания включаемого файла очень прост: начать процесс объединения текста, выдать для каждой темы предложение #DEFINE с определением константы и идентификатора справочного контекста и закрыть файл.

```
set textmerge on to (lcHFile) noshow
 scan for INTOC
   lcFile = trim(FILENAME)
   \\#define IDH_<<upper(juststem(lcFile)))>> <<HELPID>>
   \
 endscan for INTOC
 set textmerge to
```

Формирование файла содержания

Тематические разделы для CHM-файла определяются в файле с расширением HHC. Этот файл использует формат "sitemap", который является HTML-кодом, но использует ограниченный набор тэгов. Отдельные элементы определяются как тэги Object с атрибутами, например такими, как текст и изображение для представления в элементе управления TreeView CHM-файла и HTML-файл, связанный с данной темой. Иерархия тематических разделов задается путем использования неупорядоченного списка

(UL и LI) тэгов. Очевидно, что этот формат является устаревшим; вы смело можете делать ставку на то, что если бы справочная система HTML Help разрабатывалась сегодня, вместо этого формата был бы использован XML-файл.

Вот пример HHC-файла. Он имеет три узла. Первый узел, «FoxTalk 2000», является узлом самого верхнего уровня; он указывает на файл FOXTALK2000.HTML и использует изображение с номером 5 (папка). Этому узлу принадлежат два дочерних узла, «Get the Message» и «Spam, Wonderful Spam», которые указывают на соответствующие HTML-файлы и используют изображение с номером 11 (документ).

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
 <HTML>
 <HEAD>
 <meta name="GENERATOR" content="Stonefield Systems
   Group Inc.">
 <!-- Sitemap 1.0 -->
 </HEAD>
 <BODY>
 <UL>
   <LI><OBJECT type="text/sitemap">
     <param name="Name" value="FoxTalk 2000">
     <param name="Local" value="FOXTALK2000.HTML">
     <param name="ImageNumber" value="5">
   </OBJECT>
   <UL>
     <LI><OBJECT type="text/sitemap">
       <param name="Name" value="Get the Message?">
       <param name="Local" value="200002DHEN.HTML">
       <param name="ImageNumber" value="11">
     </OBJECT>
     <LI><OBJECT type="text/sitemap">
       <param name="Name" value="Spam, Wonderful Spam">
       <param name="Local" value="200003DHEN.HTML">
       <param name="ImageNumber" value="11">
     </OBJECT>
   </UL>
 </UL>
 </BODY>
 </HTML>
```

Первая задача, которую надо решить при создании HHC-файла, — это выдача заголовка (header), содержащего только статичную информацию (хотя при желании вы могли бы использовать опцию textmerge применительно к тэгу META NAME).

```
set textmerge on to (forceext(lcCHMFile, 'hhc')) noshow
 \\<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
 \<HTML>
 \<HEAD>
 \<meta name="GENERATOR" content="Stonefield Systems
 \\Group Inc.">
 \<!-- Sitemap 1.0 -->
 \</HEAD>
 \<BODY>
 \<UL>
```

Затем мы получим в выходных данных тематические разделы. Мы будем «крутиться» в таблице Contents, обрабатывая записи, в которых поле INTOC имеет значение .T.. Хотя в форматировании, вообще говоря, нет необходимости, оно сделает файл более «удобочитаемым», поэтому мы предусмотрим для каждой строки отступ на соответствующее расстояние, в зависимости от значения поля LEVEL в данной записи. Если мы переходим к другому разделу или покидаем уровень (то есть для данной записи значение поля LEVEL меньше, чем в предыдущей записи), нам необходимо поместить в выходные данные тэг </UL>, чтобы завершить описание предыдущей группы. Если мы перемещаемся по уровню (данная запись является дочерним узлом предыдущей записи), мы помещаем в выходные данные тэг <UL>, чтобы начать описание группы. Наконец, мы помещаем в выходные данные информацию для текущей записи. Поскольку в поле NAME могут содержаться недопустимые HTML-символы (например, длинное тире, амперсанд или кавычки), мы обратимся к функции ConvertTitle, чтобы преобразовать заглавие в допустимую HTML-сроку.

```
lnSection = -1
 lnLevel   = 0
 scan for INTOC

 * Определить отступ.

   lcPrefix0 = replicate(ccTAB, LEVEL)
   lcPrefix1 = replicate(ccTAB, LEVEL + 1)
   lcPrefix2 = lcPrefix1 + ccTAB

 * если мы находимся в другом разделе или уровнем ниже,
 * вставить тэг </UL>.

   do case
     case SECTION <> lnSection and lnSection > 0
       \<<ccTAB>><</UL>
     case LEVEL < lnLevel
       \<<lcPrefix1>><</UL>

 * Если мы находимся уровнем выше, чем прежде,
 * вставить тэг <UL>.

     case LEVEL > lnLevel
       \<<lcPrefix0>><<UL>
   endcase
   lnSection = SECTION
   lnLevel   = LEVEL

 * Сформировать данные для текущего файла.
   \<<lcPrefix1>><<LI><OBJECT type="text/sitemap">
   \<<lcPrefix2>><<param name="Name" value="
   \\<<ConvertTitle(NAME)>>">
   \<<lcPrefix2>><<param name="Local" value="
   \\<<trim(FILENAME)>>">
   if not empty(IMAGE)
     \<<lcPrefix2>><<param name="ImageNumber" value="
     \\<<transform(IMAGE)>>">
   endif not empty(IMAGE)
   \<<lcPrefix1>><</OBJECT>
 endscan for INTOC
```

Последний шаг — поместить в выходные данные закрывающие тэги и закрыть HHC-файл:

```
\<<ccTAB>><</UL>
 \</UL>
 \</BODY>
 \</HTML>
 \
 set textmerge to
```

Вот программный код для функции ConvertTitle. Он использует функцию STRTRAN() для замены не-

допустимых HTML-символов на их допустимые эквиваленты (например, символ “&” заменяется на строку “&”). Эта подпрограмма обрабатывает не все такие символы, а только те, которые я использовал в заглавиях написанных мной документов. Не забудьте модифицировать эту подпрограмму в соответствии с требованиями ваших заглавий.

```
function ConvertTitle(tcTitle)
 local lcTitle
 lcTitle = strtran(trim(tcTitle), '&', '&amp;')
 lcTitle = strtran(lcTitle, '"', '&quot;')
 lcTitle = strtran(lcTitle, chr(151), '--')
 return lcTitle
```

Формирование индексного файла

Ключевые слова для индексного CHM-файла определяются в HHK-файле. Аналогично файлу содержания, этот файл представлен в формате “sitemap”. Каждое ключевое слово задается в тэге OBJECT с атрибутами, которые определяют ключевое слово и связанный с ним HTML-файл.

Вот пример HHK-файла. Обратите внимание на то, что его структура проще, нежели структура HHC-файла; в этом файле отсутствуют иерархические конструкции. Заметьте также, что для каждой записи предусмотрены два идентичных тэга PARAM NAME = "Name". Хотя для создания CHM-файла необходим только один тэг, появление второго тэга обеспечивает более ясное представление в приложении HTML Help Workshop; в отсутствие второго тэга вы увидите строку “Untitled” в записи для этого ключевого слова. В примере индексного файла есть четыре индексных записи. Первые две записи указывают на один и тот же файл, 200001DHEN.HTML. Две последние записи указывают на разные файлы, но они имеют одно и то же индексное ключевое слово, так что при выборе этого ключевого слова в CHM-файле открывается диалоговое окно, в котором перечислены заголовки двух документов, чтобы вы могли выбрать один из них.

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
 <HTML>
 <HEAD>
 <meta name="GENERATOR" content="Stonefield Systems
   Group Inc.">
 <!-- Sitemap 1.0 -->
 </HEAD>
 <BODY>
 <UL>
   <LI><OBJECT type="text/sitemap">
     <param name="Name" value="persist">
     <param name="Name" value="persist">
     <param name="Local" value="200001DHEN.HTML">
   </OBJECT>
   <LI><OBJECT type="text/sitemap">
     <param name="Name" value="form position">
     <param name="Name" value="form position">
     <param name="Local" value="200001DHEN.HTML">
   </OBJECT>
   <LI><OBJECT type="text/sitemap">
     <param name="Name" value="email">
     <param name="Name" value="email">
     <param name="Local" value="200002DHEN.HTML">
   </OBJECT>
   <LI><OBJECT type="text/sitemap">
     <param name="Name" value="email">
     <param name="Name" value="email">
     <param name="Local" value="200003DHEN.HTML">
   </OBJECT>
 </UL>
 </BODY>
 </HTML>
```

Первое, что надо сделать при создании HHK-файла, — это выдать заголовок (header). За исключением имени выходного файла, этот код идентичен коду, предназначенному для создания HHC-файла, поэтому я не буду утруждать себя его демонстрацией.

Затем мы помещаем в выходные данные информацию индексов. Мы будем обрабатывать только те записи из таблицы Contents, которые должны присутствовать в индексе (значение в поле ININDEX равно .T.). Если memo-поле INDEX пусто, будем использовать в качестве индексной записи заглавие темы. В противном случае используем каждое ключевое слово из поля INDEX. Для каждой записи индекса будем помещать в выходные данные ключевое слово (используя функцию ConvertTitle для гарантии того, что это допустимая HTML-строка) и связанный с ним файл.

```
scan for ININDEX
   lcFile = trim(FILENAME)
   if empty(INDEX)
     lnTopics = 1
     dimension laTopics[1]
     laTopics[1] = trim(NAME)
   else
     lnTopics = alines(laTopics, INDEX)
   endif empty(INDEX)
   lcPrefix1 = ccTAB
   lcPrefix2 = ccTAB + ccTAB
   for lnI = 1 to lnTopics
     lcKeyword = ConvertTitle(laTopics[lnI])
     \<<lcPrefix1>><LI><OBJECT type="text/sitemap">
     \<<lcPrefix2>><param name="Name" value="
     \\<<lcKeyword>>">
     \<<lcPrefix2>><param name="Name" value="
     \\<<lcKeyword>>">
     \<<lcPrefix2>><param name="Local" value="
     \\<<lcFile>>">
     \<<lcPrefix1>></OBJECT>
   next lnI
 endscan for ININDEX
```

Наконец, поместим в выходные данные закрывающие тэги и закроем HHK-файл. Я не буду демонстрировать этот код, поскольку он полностью аналогичен программному коду для HHC-файла.

Компиляция CHM-файла

Теперь, когда у нас созданы файлы проекта справочной системы HTML Help, не трудно сформировать CHM-файл: мы просто запустим на исполнение компилятор HHC.EXE из приложения HTML Help Workshop и передадим ему в качестве параметра

имя HHP-файла. Вместо того, чтобы жестко программировать местонахождение этого EXE-модуля, мы будем считывать «адрес» приложения HTML Help Workshop из реестра Windows Registry с помощью класса Registry, который поставляется вместе с VFP. Компилятор будет посылать все сообщения в log-файл (LOG.TXT), так что после завершения компиляции мы можем просмотреть этот протокол. Один нюанс: если только вы не внесли изменений, файл FOXRUN, который находится в «домашнем» каталоге VFP и сообщает программе, что надо делать, если встретилась команда RUN, указывает на файл COMMAND.COM. Поскольку этот командный интерпретатор не может обрабатывать длинные имена файлов, нам придется преобразовать длинные имена в короткий формат (в стиле операционной системы DOS в формате “8.3”). Об этом позаботится функция ShortPath, которая использует функцию GetShortPathName интерфейса Windows API. После вызова компилятора HTML Help, мы выдадим на экран CHM-файл, воспользовавшись функцией ShellExecute интерфейса Windows API.

```
loRegistry = newobject('Registry', ;
   home() + 'FFC\Registry.vcx')
 lcPath     = ''
 llGotPath  = loRegistry.GetRegKey('Path', @lcPath, ;
   'Software\Microsoft\Windows\CurrentVersion\' + ;
   'App Paths\hhw.exe', HKEY_LOCAL_MACHINE) = 0
 if llGotPath
   lcPath = ShortPath(forcepath('hhc.exe', lcPath))
   erase log.txt
   run &lcPath &lcHHPFile > log.txt
   if file('log.txt')
     modify file log.txt nowait
   else
     erase log.txt
   endif file('log.txt')
 else
   messagebox('Cannot locate HTML Help Workshop')
 endif llGotPath

 * Если CHM-файл был создан, выдать его на экран.

 if file(lcCHMFile)
   declare integer ShellExecute in SHELL32.DLL ;
     integer nWinHandle, string cOperation, ;
     string cFileName, string cParameters, ;
     string cDirectory, integer nShowWindow
   ShellExecute(0, 'Open', lcCHMFile, '', '', 1)
 endif file(lcCHMFile)
```

Вот код для функции ShortPath:

```
function ShortPath(tcPath)
 local lcPath, ;
   lnLength, ;
   lcBuffer, ;
   lnResult
 declare integer GetShortPathName in Win32API ;
   string @lpszLongPath, string @lpszShortPath, ;
   integer cchBuffer
 lcPath   = tcPath
 lnLength = 260
 lcBuffer = space(lnLength)
 lnResult = GetShortPathName(@lcPath, @lcBuffer, lnLength)
 return iif(lnResult = 0, '', left(lcBuffer, lnResult))
```

Подводя итоги

Чтобы создать CHM-файл, содержащий мои статьи из журнала FoxTalk за 2000 год, запустите на исполнение программу FOXTALKMAIN.PRG. Сначала она создаст из Word-документов HTML-файлы (вы можете «закомментировать» эту строку, если этот шаг вы уже выполнили), а затем создаст CHM-файл и откроет его. Смотрите страницу Contents, чтобы разобраться с тем, как организован этот файл; его структура должна соответствовать компоновке, описанной в таблице содержания. Обратите внимание на ключевые слова, перечисленные на странице Index; они взяты из memo-поля INDEX таблицы содержания. Попробуйте выполнить поиск ключевого слова, скажем “e-mail”.

Сформировать CHM-файла из набора Word-документов несложно, хотя от вас потребуется выполнение некоторой предварительной работы. Вы должны решить, какая обработка Word-документов необходима при формировании HTML-файлов, и вы должны заполнить таблицу содержания с тем, чтобы эти утилиты знали, как следует формировать CHM-файл. Однако, как только это сделано, всего лишь несколько мгновений потребуется на то, чтобы проделать главную работу. Наслаждайтесь!

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

❑❑❑

Закон Бенфорда: фильтр для мошенников

Лорен Кларк (Lauren Clarke)

«Придумать» правдоподобные числа не так просто, как вы, возможно, думаете. Более того, придумать правдоподобные числа труднее, чем это представляется средней руки жулику. Люди чаще всего имеют ошибочное мнение о том, из чего состоит «случайный» набор чисел. Из этого факта можно извлечь пользу, если вашим занятием является поиск вероятных фальсификаций в наборах числовых данных. В публикуемой статье Лорен Кларк снабжает вас набором инструментов, позволяющих выявить необычные закономерности в данных и, может быть, вскрыть факт их фальсификации.

Последние события в мире финансов оказались тем прожектором, в свете которого стало видно сколь обманчивы могут быть числа. Хотя трудности фирмы Enron, как представляется, в большей степени связаны с сомнительной практикой ведения бухгалтерской отчетности, нежели с фабрикацией «липовых» данных, эта история служит примером того, что может произойти, если данные, которые используются как информационное обеспечение при принятии критичных решений, расходятся с реальностью. Те из нас, кто собирает и контролирует данные, поступают правильно, учитывая, какое влияние оказали бы «некачественная» информация на организации, с которыми мы работаем. В наших интересах не пожалеть усилий и обеспечить гарантию того, что данные являются достоверными и не содержат фальсификаций. В общем случае, вполне надежными методами выявления ошибок или проблем, которые связаны с файловой системой и могут привести к появлению ложных данных, являются контроль ссылочной целостности и обеспечение целостности файлов.

Но допустим, что вашим противником может оказаться не сбой в работе аппаратуры или программного обеспечения, а напротив, мыслящее, вынашивающее планы человеческое существо, у которого есть причины для разрушения тех данных, которые вы так заботливо храните. Какие разумные шаги вы предприняли бы, чтобы выявить такую попытку? Эта статья предоставит в ваше распоряжение набор инструментов для проведения углубленного анализа данных и автоматизации, в некоторых пределах, процесса защиты от ложных данных.

Цифровой анализ (ЦА) — это термин, который используется для обозначения исследования наборов числовых данных, проводимого с целью выявления в них неких закономерностей (patterns). Другими словами, высокая стоимость в 12 345,67 долларов сиденья для унитаза с точки зрения ЦА не является чем-то из ряда вон выходящим. Такая цена примечательна тем, что ее числовое выражение состоит из цифр “1234567”, расположенных в порядке возрастания: от младших цифр к старшим. Случайное совпадение? Может быть, но что если большинство цен в данном наборе состоит из цифр, которые расположены в точно таком же порядке? Этот факт может послужить основанием для проведения более тщательного анализа таких чисел.

Цифровой анализ способен оказать содействие в отслеживании таких вещей, как «округленные» числа, повторяющиеся числа и числа, в которых распределение цифр не соответствует ожидаемому. Цифровой анализ может быть использован для выявления обмана, ошибок, недостаточности данных и даже ошибок в программном обеспечении. Неотъемлемой частью ЦА является малоизвестный закон чисел, который носит название «закон Бенфорда». Закон Бенфорда предсказывает частоту появления определенных значащих цифр во множестве разнообразных наборов данных. Многие методики ЦА используют эти сведения в качестве инструмента выявления тех наборов чисел, поведение которых не соответствует ожидаемому.

Краткая история закона

В 1881 году астроном по имени Саймон Ньюком (Simon Newcomb) опубликовал статью, в которой описывалось некое явление, подмеченное им в таблицах логарифмов, применявшихся в то время для умножения больших чисел. В те дни, если вам необходимо было перемножить какие-то большие числа, вы не щелкали дважды мышью по пиктограмме табличного процессора Excel. Вместо этого, вы отправлялись в библиотеку и находили свои числа в таблицах логарифмов, выполняли сложение этих логарифмов, а затем искали антилогарифм полученной суммы.

Ньюком обратил внимание на то, что страницы, соответствующие числам, которые начинались с цифр 1, 2 и 3, были более затертыми, чем те, которые соответствовали числам, начинающимся со старших цифр. По сути, потрепанность страниц монотонно уменьшалась по мере того, как первая значащая

цифра возрастала с 1 до 9. (Заметьте, что оба числа, 876 и 0.0876, имеют одну и ту же первую значащую цифру — 8.) Поскольку этими таблицами логарифмов пользовалось множество самых разных людей, Ньюком пришел к выводу, что неодинаковая «зачитанность» страниц должна объясняться тем, что числа вообще имеют тенденцию чаще начинаться с младших цифр, нежели со старших. Он также выдвинул (без доказательства) следующую гипотезу: вероятность того, что любое полученное как результат измерений объектов во вселенной число начинается с цифры “d”, определяется по формуле:

$$P(\text{first significant digit} = d) = \log_{10}(1+1/d)$$

В таблице 1 перечислены некоторые значения вероятностей, предсказанные по этой формуле.

Таблица 1. Значения вероятностей, полученные для первых цифр чисел, согласно предсказанию закона Бенфорда.

Цифра	Вероятность
1	0.301
2	0.176
3	0.125
4	0.097
5	0.079
6	0.067
7	0.058
8	0.051
9	0.046

На первый взгляд, это вовсе не то, чего можно было бы ожидать. Учитывая, что числа могут начинаться с цифр 1-9, большая часть людей, скорее всего, предположили бы следующее: вероятность того, что конкретная цифра окажется первой, должна быть одинаковой для всех цифр, то есть она должна составлять 1/9 или приблизительно 11 процентов. Гипотеза Ньюкома утверждала, что первая цифра в тех числах, которые представляли собой измерения различных вещей, стремилась оставаться “1” в 30 процентах случаев — результат, определенно не являющийся очевидным.

Несколькими годами позже (точнее говоря, спустя 57 лет) инженер из фирмы GE по имени Фрэнк Бенфорд самостоятельно выявил ту же самую ситуацию в своих таблицах логарифмов. Он сделал еще один шаг и собрал данные из нескольких различных источников, диапазон которых простирался от бейсбольной статистики до площадей водных бассейнов рек и численности народонаселения штатов. Использованые Бенфордом 20229 различных источников данных очень хорошо укладывались в приведенную формулу. В таблице 2 представлены некоторые из полученных им результатов.

В 1938 году Бенфорд опубликовал полученные им данные в журнале *Proceedings of the American Philosophical Society*, и эта статья привлекла к себе такое большое внимание, что закон стал известен как «Закон Бенфорда», несмотря даже на то, что первым его открыл Ньюком. Между прочим, причина, по которой эта публикация привлекла к себе такое большое внимание читателей, заключалась, видимо, в том, что статья была напечатана именно в данном номере журнала и размещалась рядом со знаменитой теперь статьей о рассеивании электронов. Хотя Бенфорд не доказал справедливость этого закона, предоставленные им эмпирические свидетельства для многих людей были достаточно неотразимыми, чтобы они воспринимали выявленную закономерность как правомочное «эвристическое правило».

В последующие несколько десятилетий был опубликован ряд статей, которые добавили к этому закону несколько важных следствий. Пожалуй, наиболее значительным открытием стало то, что этот закон оказался масштабно-инвариантным. То есть, если набор чисел подчинялся закону Бенфорда, то числа из этого набора можно было бы умножить на любое скалярное кратное, и полученный в результате набор также был бы «набором Бенфорда». Это следствие важно, если вы примете во внимание, что оно означает следующее: пересчет валют из одной в другую или взаимные преобразования единиц измерения, например, дюймов в сантиметры, не повлияют на ожидаемые распределения первых цифр. Переходя далее, к более современной истории, в настоящее время этот закон статистически доказан Теодором Хиллом (Theodore Hill) в опубликованной им в 1996 году статье, озаглавленной «A Statistical Derivation of the Significant-Digit Law»,

Таблица 2. Распределение первых цифр в эмпирических данных, собранных Бенфордом.

	1	2	3	4	5	6	7	8	9
Площади бассейнов рек	31.0	16.4	10.7	11.3	7.2	8.6	5.5	4.2	5.1
Народонаселение	33.9	20.4	14.2	8.1	7.2	6.2	4.1	3.7	2.2
Статистика бейсбольной американской лиги	32.7	17.6	12.6	9.8	7.4	6.4	4.9	5.6	3.0
Цифры, встретившиеся на страницах издания Reader’s Digest	33.4	18.5	12.4	7.5	7.1	6.5	5.5	4.9	4.2

которую вы можете прочесть сами по адресу www.math.gatech.edu/~hill/publications.

Наглядное объяснение

Единственно на тот случай, если вы не склонны «продираться» через статью д-ра Хилла, вот краткое эвристическое объяснение того, в чем состоит суть закона Бенфорда. Коротко говоря, вещи начинаются с малого размера и имеют тенденцию расти со скоростью, пропорциональной их размеру. Рассмотрим, например, помесячный баланс депозитного счета, для которого исходная сумма депозита составляла $900, а годовая суммарная процентная ставка равна 6 процентам. При запрете любого рода операций снятия или вклада по этому счету потребуется всего лишь два года на то, чтобы сумма счета составила $1000 и, следовательно, изменилось значение первой цифры. В течение этого периода баланс вашего счета оставался бы в пределах $900: первой была бы цифра 9. После того, как баланс «перевалил» за тысячу, первой цифрой остатка будет 1 до тех пор, пока баланс не достигнет суммы в $2000. При годовой ставке дохода в 6 процентов на это ушло бы еще 12 лет. Вы можете видеть, что набор данных, представляющих ежемесячные балансы за 14-летний период, имел бы в своем составе гораздо больше чисел, начинающихся с цифры 1, чем с цифры 9.

Еще важнее то, что набор данных, представляющих все счета в некотором банке в некоторый конкретный момент времени, демонстрировал бы, вероятно, то же самое распределение (если только этот банк не был открыт совсем недавно). Этот факт не имеет ничего общего с экспоненциальным ростом как таковым, напротив, он связан с нашей системой чисел. Если некоторая сущность расширяется (или сжимается) с любой скоростью, она будет иметь тенденцию к более быстрому прохождению через числа, начинающиеся с цифр 7, 8 и 9 и к более медленному прохождению через числа, начинающиеся с цифр 1, 2 и 3. В общем, будь это банковские счета, реки, биржевой курс или галактики — вселенная наполнена измеримыми объектами — тенденция будет такова, что среди этого многообразия будут преобладать малые объекты, а те из объектов, которые возрастают, имеют тенденцию расти таким образом, при котором их измерения вынуждены подчиняться закону Бенфорда.

Дополнения к закону

В более общем виде закон Бенфорда может применяться ко всем цифрам числа, как показано в следующей формуле.

$$P(\text{starts with } d_n d_{n-1} d_{n-2}..d_0 = \\ = \log_{10}(1+1/(d_n\times 10^n + d_{n-1}\times 10^n + ... + d_0\times 10^0))$$

Пример этого вы можете видеть ниже:

$$P(\text{starts with } 678) = \log_{10}(1+1/678) = 0.00064$$

Сказанное означает, что мы можем взять первую цифру, вторую цифру, первую и вторую цифры и так далее и получить для сравнения ожидаемые распределения.

Решение, но не на все случаи жизни

Важно отметить, что закон Бенфорда применим не ко всем мыслимым наборам чисел. Это должно быть очевидным. Например, набор всех почтовых ZIP-кодов штатов западного побережья не включает ни одного кода, который начинается с цифры “1”. В общем, этот закон не применим к заданным числам. Числа должны представлять собой размеры некоторой физической или, по крайней мере, измеряемой сущности. Таким образом, не следует ожидать, что первичные ключи в таблице БД или номера из телефонной книги подчиняются закону Бенфорда. Вот некоторые общие правила для определения того, подчиняются ли числа закону Бенфорда:

- Данные должны описывать размеры набора подобных сущностей.
- Данные должны быть «не связанными между собой», то есть они не должны иметь никаких «встроенных» максимумов или минимумов (однако, нуль в качестве минимума допустим). Кроме того, не должно быть никаких «специальных» чисел. Например, в данных отчета о затратах, сумма допустимых ежедневных расходов может представлять собой искусственное ограничение, которое в состоянии исказить данные.
- Данные не должны быть заданными числами. Как оговорено первым правилом, данные должны быть связаны с измерениями некоторого рода размеров. Заданные числа не отвечают этому критерию.
- Сущности, описываемые данными, должны включать больше малых элементов, нежели больших.

Если ваши данные «не вписываются» в эти критерии, вы, тем не менее, можете использовать методы цифрового анализа. В некоторых случаях данные все-таки могут подчиняться закону Бенфорда. Даже если это не так, понимание того, почему это не так, и установление вашего собственного ожидаемого рас-

пределения цифр на основе исторических данных, также могут привести вас к эффективной процедуре проверки наличия аномальных данных.

Применение закона на практике

На первый взгляд, закон Бенфорда производит впечатление бесполезного математического пустяка. По сути, таково было общее мнение в течение определенного периода времени. Кроме одурачивания несведущих людей при заключении пари относительно того, какая цифра будет первой в первом числе, найденном на указанной странице альманаха фермера (Farmer's Almanac), от этой информации, казалось, мало было практической пользы. Однако, примем во внимание, что вряд ли люди, придумывающие числа, будут воспроизводить распределение Бенфорда в сфабрикованных ими числах. Вообще говоря, было показано, что цифры 6 и 7, по-видимому, являются «излюбленными» первыми цифрами в числах, которые «случайным образом» формирует человек. Раз так, набор фиктивных чисел не будет демонстрировать характеристику распределения Бенфорда, и этот факт может оказаться важным «фильтром» для идентификации ложных данных. Основная идея заключается в том, чтобы сравнить «данные отчета о еженедельных расходах, полученные от продавца xyz», с предполагаемым распределением Бенфорда и не оставить без внимания любые встретившиеся расхождения. Не забудьте, данные о расходах могут обладать такими свойствами, которые в состоянии «забраковать» числа даже для допустимых данных, но это должны быть известные и, вероятно, объяснимые факторы. Помните, цифровой анализ — это средство, позволяющее определить то место, где надо искать подлог; он не должен использоваться как единственный опознаватель искажения.

Учитывая, что закон Бенфорда дает нам некоторые предсказания относительно того, как распределяются в данных первая, вторая и другие цифры, каким образом мы можем сравнивать свои данные с результатами этого предсказания? Первый шаг этой процедуры — определить, как распределяются цифры в данном наборе чисел. Следующий шаг — функция, которая будет обрабатывать конкретные значащие цифры числа. Следует позаботиться о том, чтобы обеспечить корректную обработку денежных значений. Мы воспользуемся функцией xVal() Прадипа Ачарьи (Pradip Acharya) (публикация в ноябрьском номере FoxTalk за 2002 год) с тем, чтобы она принимала в качестве входных параметров нечисловые данные и разумно преобразовывала их в числовые значения.

```
function digit( tvNumber, tnSig , tnLen )
* возвращает tnLen значащих цифр числа,
* начиная с tnSig-ой значащей цифры
local lcNum, lnNum

if vartype( tnLen ) # "N"
  tnLen = 1
endif

*-- стандартизировать тип переменной vartype
lnNum = xVal( tvNumber )
*-- преобразовать тип currency
if vartype( lnNum ) ="Y"
  lnNum = mton(lnNum)
endif

*-- нормализовать мантиссу
if !empty(lnNum)
  lnNum = abs(lnNum / ;
          10^(round(log10(abs(lnNum))-1,0)))
endif

lcNum = strtran(transform(lnNum),".")
return val( substr( lcNum ,tnSig,tnLen) )

* Например:
? Digit( 1234.567,1,1 ) && prints 1
? Digit( 1234.567,2,4 ) && prints 2345
? Digit( 0.0876,2,2 )   && prints 76
```

С помощью этой функции (на самом деле, это метод класса BenfordCalc, который вы найдете на сопровождающей дискете) можно определить наличие конкретных цифр в определенных позициях для любого числа. Если функция используется применительно не к одной, а к набору записей, и при этом хранятся счетчики, предусмотренные для каждой цифры в каждой позиции, то распределение цифр может быть получено для любого набора чисел. В конечном итоге вы получите справочную таблицу относительных частот для набора данных, которая выглядит как таблица 1, но отражает распределение цифр в ваших данных. Вот пример того, как получить такое распределение, используя таблицу orders.dbf, поставляемую в наборе данных из примера TasTrade.

```
select digit( order_amt, 1, 1 ) as digit, ;
     cnt( digit( order_amt, 1, 1 )) / reccount("orders") ;
     from orders ;
     group by 1
```

Это предложение языка запросов вернет курсор, показанный на рис. 1, который при сравнении его с ожидаемым распределением из таблицы 1 обнаружит очень мало отличий. Следующий шаг — сравнить полученное в результате запроса распределение с предусмотренным законом Бенфорда ожидаемым распределением из таблицы 1. Это сравнение может быть выполнено несколькими различными способами и то, какой способ вы изберете, зависит от конкретных обстоятельств. Если тестируются несколько наборов чисел, предпочтительным окажется некоторого рода автоматизированный процесс, позволяющий обеспечить реальный просмотр данных в таблице или на графике. Давайте сначала оценим полученные результаты визуально, а затем исследуем

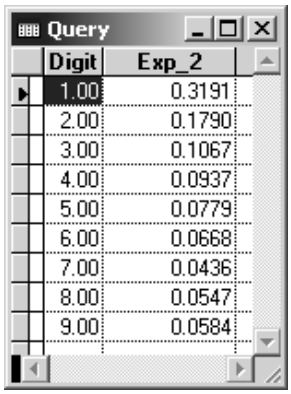

Рис. 1. Распределение первой цифры в таблице tastrade!orders.order_amt.

некоторые «невизуальные» способы сравнения. В качестве исходных данных мы будем использовать данные о заказах из примера TasTrade, которые находятся в подкаталоге samples каталога Visual FoxPro.

Обратимся теперь к графическому представлению наших результатов. Конечно, имеется множество доступных инструментальных средств графического представления данных. Среди наиболее распространенных инструментов — табличный процессор Excel, пакет Office Web Components и приложение MS Graph. В данной статье мы пойдем несколько менее исхоженным путем и используем для создания своих графиков векторный язык разметки VML. VML (Vector Markup Language: www.w3.org/tr/note-vml) — это приложение к стандарту XML, позволяющее определять графику в XML-коде с целью ее воспроизведения в VML-совместимых браузерах. Браузер Internet Explorer 5.0 и все его последующие версии будут воспроизводить VML-представление. Аналогичным и более распространенным (по крайней мере, в теории) стандартом является стандарт SVG, который в настоящее время поддерживается только браузером Amaya (www.w3.org/Amaya). Вот пример фрагмента VML-кода. В данном случае мы проводим штрихпунктирную линию красного цвета шириной в 2 пикселя из точки с координатами 0,0 (верхний левый угол) в точку с координатами 100,100. Истинное пространство используемых координат зависит от контейнера, в который помещается эта линия.

```
<v:line strokeweight='2.0px'
        strokecolor='red'
        from='0,0'
        to='100,100'>
        <v:stroke dashstyle="dashdot" />
</v:line>
```

Вы в состоянии представить себе, как можно было бы создать коллекцию графических элементов, например точек и линий, для построения графика. Небольшой класс, предназначенный для формирования графика по XML-описанию, представлен в исходном коде в приложении к данной статье. Для создания собственно VML-разметки этот класс использует шаблон strategy, так что можно было бы описать стратегию формирования SVG-представления и использовать ее вместо VML-стратегии тогда, когда стандарт SVG будет поддерживаться большим числом браузеров.

К тому же, вместо того, чтобы использовать для получения распределений команду SELECT, как это делалось раньше, мы воспользуемся классом BenfordStat, который работает с объектами из класса Dataset. Dataset — это очень простой итератор, который является «фасадом» для массивов и таблиц, что позволяет классу BenfordStat оперировать и с теми, и с другими объектами без внесения каких-либо изменений в программный код. Класс SMGraph также использует объекты dataset для воспроизведения VML-представления на экране. Конечный результат — это возможность более свободно обращаться с анализом данных за счет некоторых потерь в производительности. В зависимости от ваших потребностей, вы можете предпочесть непосредственное использование команды языка запросов SELECT или воспользоваться «объектным» методом.

График, показанный на рис. 2, был построен с помощью следующего кода:

```
local lcTableName, lcFieldName, loGraph, loTTOrders,;
      loTTFDist, loBenfordStat, loBenfordFD, lcStr

lcTableName = "orders"
lcFieldName = "order_net"

*-- Объект вычислений Бенфорда
loBenfordStat = createobject( "BenfordCalc" )

*-- Извлечь предопределенное распределение Бенфорда
select nvalue,np ;
  from benford ;
  where cposition=="FIRST" ;
  order by nvalue ;
  into array laTemp

*-- Определить ожидаемое распределение первой цифры
loBenfordFD = createobject( "DataSet" )
loBenfordFD.setData( @laTemp,1,2)

*-- Настроить опции вычерчивания графиков
loBenfordFD.cType  = [XY]
loBenfordFD.nLineWidth=0.5

*-- Выбрать набор данных для анализа
loTTOrders    = createobject( "DataSet" )

if !used(lcTableName)
  use _samples+"data\"+lcTableName+".dbf" in 0
endif
loTTOrders.setData( lcTableName, lcFieldName)

*-- Вычислить распределение первой цифры

loTTFDdist = loBenfordStat.digitDist( ;
             loTTOrders, "FIRST")

*-- Настроить опции вычерчивания графиков
with loTTFDdist
  .cType=[BAR]
  .nLineWidth=10
  .cLineColor="#4682B4"
endwith
```

Рис. 2. Точки, соединенные линиями, представляют ожидаемое распределение для первой цифры. Вертикальные столбцы представляют распределение первых цифр, выявленное в поле Order_Net для таблицы Orders из примера TasTrade.

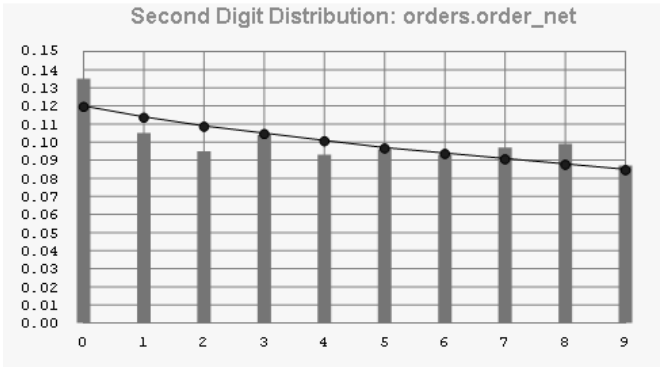

Рис. 3. На этом рисунке мы видим ожидаемое распределение второй цифры (линия) и распределение второй цифры в таблице Orders.Order_Net (столбцы). Похоже на то, что в данном конкретном наборе данных мы имеем слишком мало цифр 2 и слишком много цифр 8.

```
*-- генератор графики с VML-стратегией отображения
loGraph = createobject("SMGrapher","VMLArtist")

with loGraph
  .nMaxXcoord = 9.3
  .nMaxYcoord = 0.4
  .nHeight    = 150
  .nWidth     = 300
  .cBGColor   = "cornsilk"
  .nYlabelScale = 0.1
  .nYScale    = 0.1
  .nXScale    = 1
  .nXlabelScale = 1
  .cTitle     = [First Digit Distribution: ] + ;
                lcTableName + [.] + lcFieldName
  .addDataSet( loTTFDdist )
  .addDataSet( loBenfordFD )
endwith

*-- создать разметку
text to lcStr textmerge noshow

<head>
<style>
v\:* { behavior: url(#default#VML); }
.xlabel
{font-family: "Courier New", Courier, monospace;
font-weight: normal;
font-size: 8pt;
color: black;}

.ylabel
{font-family: "Courier New", Courier, monospace;
font-weight: normal;
font-size: 8pt;
color: black;}

.charttitle
{font-family: Arial,"Courier New", Courier, monospace;
font-weight: bold;
font-size: 12pt;
color: cadetblue;}

</style>
</head>
<html xmlns:v="urn:schemas-microsoft-com:vml">
<body>
<<loGraph.drawChart()>>
</body>
</html>

endtext
strtofile( lcSTR, "charts.htm")
```

Вышеприведенный программный код был взят из файла benford.prg, входящего в состав исходного кода, предусмотренного в качестве приложения к этой статье. Этот PRG-файл является простым сценарием, который создаст все демонстрируемые в этой статье графики и выдаст их на экран в окне вашего браузера.

При анализе результатов, представленных на рис. 2, складывается впечатление, что 1078 записей из таблицы Orders вполне соответствуют распределению Бенфорда. Давайте посмотрим на то, как распределяется вторая цифра (см. рис. 3).

Ни один из этих графиков не выглядит особенно плохим с точки зрения распределения Бенфорда. Тогда встает вопрос о степени соответствия. Как определить, когда набор данных следует «объявить» как несоответствующий закону? Этот вопрос связан с вопросом о том, как выявить отличающиеся наборы без визуального анализа графического представления распределений. Исчерпывающий ответ на эти вопросы выходит за пределы данной статьи. Я, однако, выскажу здесь пару идей.

Оценка отклонений

Если вы читали какую-либо статистическую литературу, то встречали, вероятно, упоминание о проверке гипотезы. В общих чертах, проверка гипотезы — это точный способ определения вероятности того, что данный набор данных соответствует (или не соответствует) предопределенному гипотетическому распределению. В нашем случае мы были бы заинтересованы в том, чтобы знать, до каких пределов распределение цифр в наших данных соответствует

ожидаемому распределению Бенфорда. Один очень распространенный тест, позволяющий определить степень согласованности — это тест «хи-квадрат», а другой — z-статистика. Оба этих теста вы можете найти почти в каждом учебнике статистики для первого года обучения.

В приложение к данной статье включен программный код получения обеих этих статистик. Выясняется, что сами по себе эти более мощные инструменты — это еще не все, что необходимо для работы с большими наборами данных: требуется выполнить ряд настроек, чтобы сделать их полезными при обращении с большим количеством результатов обработки данных. Более простым способом было бы вычисление для каждой вероятности значения среднего абсолютных разностей. Чтобы вычислить среднее абсолютной разности (Mean Absolute Difference — MAD), просто возьмите абсолютные значения разностей между ожидаемым значением и действительным значением, представляющими частоту появления каждой цифры, и суммируйте их. Затем разделите полученную сумму абсолютных разностей на общее количество цифр, присутствующих в результатах обработки данных. В таблице 3 показаны значения MAD для нашего теста распределения второй цифры.

Таблица 3. В этой таблице представлены ожидаемые распределения для второй цифры в сравнении с распределениями, обнаруженными в данных примера TasTrade. Чтобы вычислить среднюю абсолютную разность, абсолютные значения разностей усредняются для всех цифр.

Цифра	Benford	Actual	Diff	abs(Diff)
0	0.120	0.135	0.015	0.015
1	0.114	0.105	-.009	0.009
2	0.109	0.095	-.014	0.014
3	0.104	0.102	-.002	0.002
4	0.100	0.095	-.006	0.006
5	0.097	0.096	-.001	0.001
6	0.093	0.093	-.001	0.001
7	0.090	0.096	0.005	0.005
8	0.088	0.098	0.011	0.011
9	0.085	0.087	0.002	0.002

Сложение всех абсолютных разностей и деление полученной суммы на 10 (число измерений) дает число 0.0065, означающее, что в этом наборе данных действительная частота появления цифры отличается от предсказанного значения в среднем на 0.65%. Такое вычисление MAD легко может быть автоматизировано, и вы могли бы определить некоторое пороговое значение, при достижении которого можно было бы послать предупреждение аудитору по электронной почте. Программный код для функции, реализующей вычисление MAD, включен в исходный код.

Хотя это, вероятно, сверхмощное средство, z-статистика может оказаться достаточно полезной вещью, чтобы включить ее в свои графики, поскольку она в состоянии направить ваш взгляд в нужном направлении и тем самым способствовать идентификации важных отклонений. На рис. 4 изображен график распределения первых двух цифр с включенными в него z-границами. Если конкретный столбец лежит вне области, ограниченной пунктирными линиями, то существует лишь 5-процентная вероятность того, что эта ошибка обусловлена одной только случайностью. Еще раз, этот факт не доказывает и не опровергает присутствие ложных данных, но он может дать аудитору некоторые важные признаки того, что пора приступить к поиску проблем в наборах данных.

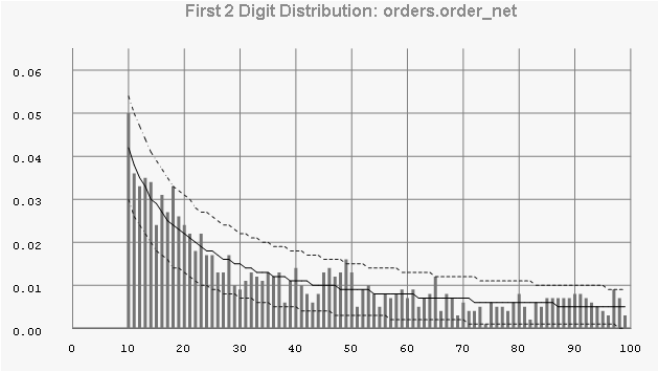

Рис. 4. На этом рисунке представлено распределение первых двух цифр (10-99). Сплошная линия — это распределение, ожидаемое согласно закону Бенфорда, столбцы — это распределение, выявленное в данных из таблицы orders.order_net, а пунктирные линии — это верхняя и нижняя границы для z-теста при 5-процентной значимости.

На рис. 4 мы анализируем распределение первых двух цифр в поле order_net. Пунктирные линии выше и ниже сплошной черной линии «ожидаемого распределения Бенфорда» демонстрируют верхнюю и нижнюю границы z-статистики при уровне значимости 5 процентов. Похоже на то, что числа 18, 49, 65, 74 и 97 могли выйти за эти границы. Это не является неожиданностью, принимая во внимание тот факт, что имеется 99 возможных чисел, состоящих из двух цифр; следовало бы ожидать, что пять или около того точек графика, представляющих данные, окажутся вне границ. Верхняя и нижняя границы предоставляют аналитику ценный «индикатор правдоподобия». Если более 5 процентов чисел постоянно не попадают в эти границы, или если в ошибках можно наблюдать какую-то закономерность, тогда эти факты могли бы указывать на некоторые манипуляции с данными, и более глубокое исследование могло бы иметь основания. По-настоящему хорошей

процедурой проведения цифрового анализа было бы сканирование входящих данных, исполнение различных тестов и сравнение полученных результатов с заранее определенными «запретными» значениями статистик, вроде MAD и хи-квадрат. Тогда, в случае обнаружения значительных отклонений, формировалось бы сообщение электронной почты со ссылками на графические представления полученных результатов для проверки их человеком. Такая процедура могла бы исполняться в фоновом режиме или как плановая задача, которая исполняется в соответствующие интервалы времени.

Не «Бенфордом» единым

Хотя закон Бенфорда является важной отправной точкой, существует много других, возможно, более подходящих тестов, которые можно было бы выполнить применительно к набору чисел. Применение многих из этих тестов было бы возможно даже для тех наборов, которые оказались бы вообще непригодными для их исследования с помощью тестов Бенфорда по причине подтасовок данных, таких как верхние и нижние границы. Вот некоторые из таких тестов:

- Две первые цифры.
- Две последние цифры.
- Последовательно расположенные цифры (например, $123.45).
- Округленные числа.
- Повторы чисел.
- Сравнение с историческими данными.

Последний пункт списка, возможно, является наиболее мощным средством. Сравнение с историческими данными важно, потому что при его проведении есть потенциальная возможность задействовать в качестве рычага другие тесты, которые оказались бы неприменимы при самостоятельном использовании. Например, если вы знаете, что определенный набор данных имеет нестандартную долю чисел, начинающихся с цифры 9, в соответствии с правилом бухгалтерского учета, согласно которому расходы, составляющие менее $1000, принудительно приписываются к этому же набору данных, тогда сравнение таких данных с законом Бенфорда было бы неуместно. Однако, не было бы и особых оснований думать, что по прошествии времени это распределение первых цифр (каково бы оно ни было) беспричинно изменилось. Раз так, может оказаться полезным сравнение исторического распределения первых цифр (или результатов любого другого теста) с распределением новых данных. Если новые данные не согласуются со старыми данными, тому, вероятно, есть причина, и этой причиной может оказаться мошенничество.

Наконец, не все ошибки в данных будут числовыми по природе, соответственно могут быть использованы иные подходы: посмотрите статистический состав ваших наборов данных и диаграмму их изменения во времени. Статья Джона Миллера (John Miller), опубликованная в февральском номере журнала FoxTalk за 2003 год, предлагает некоторые общие идеи и направления относительно поиска в данных аномалий всех сортов.

В заключение

Цифровой анализ должен присутствовать в инструментальном ящике всех менеджеров по работе с данными. При незначительных усилиях процедуру проведения такого анализа можно автоматизировать, и это обеспечит действенное оружие в арсенале средств борьбы с ложными данными. Для того, чтобы применить эти методики не только для обнаружения фактов элементарного подлога в бухгалтерской отчетности, требуется некоторое воображение. Рассмотрите возможности настройки рассмотренных инструментальных средств для проведения анализа закономерностей, встречающихся в цифрах IP-адресов, взятых из журналов обращений к Web-серверу, или выявления распределения цветов в графических изображениях (возможно, в интересах отделения фотографий от рисунков программным способом). Кроме того, в процессе дальнейшего перехода от закона Бенфорда к сравнению исторических или связанных наборов данных, появляются новые идеи, например, сравнить объемы продаж с итоговыми цифрами счетов поставщика, сопоставить денежные поступления с суммами, указанными в счетах-фактурах, и проанализировать итоговые данные ежедневных продаж в конце года и в начале следующего года (или в какие-то еще важные периоды времени).

В сравнении с другими мерами обеспечения безопасности и противодействия мошенничеству, цифровой анализ дает высокое качество результатов при относительной дешевизне реализации и предоставляет дополнительный уровень защиты для самого ценного нашего ресурса — информации.

Лорен Кларк (Lauren Clarke) возглавляет фирму Cornerstone Systems Northwest, специализирующуюся на предоставлении инструментальных средств на базе Web для сбора данных, формирования знаний и управления решениями. Кроме того, Лорен любит обучать других тому, как надо разрабатывать приложения, использующие Web-технологии.
lauren@cornerstonenw.com

❑❑❑

Дело было так, инспектор…

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

WIN 7.0 DOWNLOAD

В этом месяце Энди Крамек и Марсиа Акинз продолжают изучение модели событий версии 7.0, которое они начали в прошлый раз, рассмотрев последовательность событий для формы. На этот раз Марсиа и Энди переходят от текстового поля и командной кнопки к спискам combobox и listbox, которые оказались гораздо более сложными, чем они это себе представляли.

Марсиа: В прошлом месяце ты говорил, что мы могли бы рассмотреть события элементов управления grid. Так вот, начать мне хотелось бы с...

Энди: Не спеши. Я знаю, что grid — это твой любимый элемент управления, но я подозреваю, что тут и без grid довольно места для неразберихи.

Марсиа: О, тогда ладно. Базовая последовательность для элемента управления text box (или edit box) достаточно проста (см. рис. 1). Прежде всего наступает событие When(), чтобы определить, может ли данный элемент управления получать фокус или нет.

Энди: Следовательно, возвращение значения .F. из события When() полностью блокирует этот элемент управления?

Марсиа: Точно. Затем наступает черед события GotFocus(). Это одно из тех событий, которые имеют программный код в своем базовом классе, поскольку если ты помещаешь в событие GotFocus() базового класса textbox команду NODEFAULT, этот элемент управления просто не может получить фокус.

Энди: Следовательно, это еще один способ блокировать элемент управления. Одно преимущество от использования любого из этих событий вместо присваивания определенных значений свойствам Enabled или ReadOnly заключается в том, что при использовании событий не меняется внешний вид элемента управления.

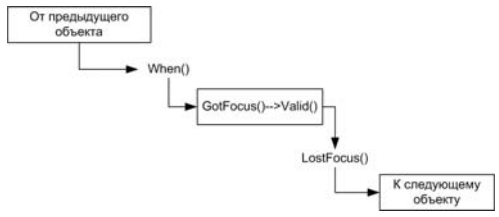

Рис. 1. Базовая последовательность событий для простого элемента управления.

Марсиа: Но мне необходимо, чтобы элемент управления визуально сигнализировал о том, что я не могу его использовать! Как бы там ни было, раз элемент управления получил фокус, после этого ничего больше не происходит (в смысле наступления событий) до тех пор, пока данный элемент не потеряет фокус. Позволены ли такие действия или нет зависит от того, какое значение возвращает событие Valid().

Энди: И этим объясняется, почему возвращение значения False из программного кода, размещенного в методе события Valid(), сохраняет фокус для этого элемента управления.

Марсиа: Последнее событие в рассматриваемой цепочке — это, конечно, событие LostFocus(), наступающее после того, как Valid() возвращает значение True, но раньше, чем наступает событие When() того элемента управления, которое следующим по очереди должно получить фокус.

Энди: И этот факт ведет к интересному отступлению от основной темы. Свойство формы ActiveControl содержит объектную ссылку лишь до тех пор, пока объект действительно имеет фокус. Таким образом, между завершением события LostFocus() одного элемента управления и завершением события GotFocus() следующего элемента управления просто не существует такого элемента управления, который был бы активен. Следовательно, в действительности ты не можешь проверить значение свойства ActiveControl формы в событии When() какого-либо элемента управления, размещенного в этой форме.

Марсиа: Это представляется разумным, поскольку задача события When() заключается в определении того, может ли элемент управления получить фокус. Однако, хотя все сказанное истинно до тех пор, пока ты просто переходишь от одного элемента управления к другому, перемещение фокуса щелчком мыши по элементу управления приводит к незначительному изменению.

Энди: Неужели, как же так? Не говоришь ли ты, что меняется базовая последовательность событий, а?

Марсиа: И да, и нет. Для конкретного элемента управления последовательность не меняется, но если

ты щелкаешь мышью по другому элементу управления, немедленно наступает событие When() этого другого элемента управления. Последовательность, которая обычно выглядит вот так:

```
Control1.When()
Control1.GotFocus()
Control1.Valid()
Control1.LostFocus()
Control2.When()
Control2.GotFocus()
```

меняется и выглядит следующим образом:

```
Control1.When()
Control1.GotFocus()
Control1.Valid()
Control2.When()
Control1.LostFocus()
Control2.GotFocus()
```

Энди: Хм-м. Я готов был предположить, что если бы мы в событии LostFocus() сохранили объектную ссылку на элемент управления в свойстве формы, то мы всегда знали бы, какой объект последним имел фокус. Однако, это означает, что нам следовало бы запомнить эту ссылку не позже, чем в событии Valid(), чтобы быть абсолютно уверенными в ее правильности при дальнейшем использовании. Собственно говоря, вероятно, это лучше всего сделать в конце события GotFocus() (к тому же, именно там это делает сама VFP).

Марсиа: Так много информации для элементов управления text box и edit box! Теперь как насчет элементов управления CommandButton, CheckBox и OptionButton? Все они имеют те же четыре события, что и текстовые элементы управления.

Энди: Да, имеют. Но когда ты просто переходишь от одного из них к другому с помощью клавиши Tab, не щелкая по ним мышью, событие Valid() этих элементов управления не наступает, хотя три других события наступают в той же последовательности, что и для элемента управления textbox. Для меня в этом есть смысл: все эти элементы управления выполняют некоторое действие, и если ты не инициируешь это действие каким-либо способом, им нет необходимости проверять что-либо. Поэтому у этих элементов управления вообще нет причины для наступления события Valid().

Марсиа: Но когда ты щелкаешь по ним мышью, событие Valid() не только наступает раньше события LostFocus(), как мы и ожидали бы, но наступает также и событие When(). Хотела бы я знать, почему так происходит.

Энди: Не имею ни малейшего понятия! Единственно, что я могу предположить: эти элементы управления часто блокируются и разблокируются, и, может быть, эти действия обрабатываются внутренне, в событии When(). Если бы причина заключалась в этом, имело бы некоторый смысл повторно инициировать наступление события When() после завершения события Valid(), чтобы проверить окончательное состояние элемента управления.

Марсиа: Очень неубедительно, но поскольку у меня нет другого объяснения, пока я принимаю такое. Теперь мы можем перейти к интересным для нас элементам управления — спискам combobox и listbox?

Энди: Одну секунду! Как насчет влияния специальных контейнеров, вроде OptionGroup и CommandGroup? Вообще говоря, они не имеют своих событий GotFocus() и LostFocus(), но они имеют собственные события Valid() и When(). Эти события наступают в промежутке между эквивалентными событиями содержащихся в контейнерах элементов управления button, хотя последовательность событий не меняется. Другими словами, за событием When() кнопки button следует событие When() ее родительского контейнера, а за «кнопочным» событием Valid() следует событие Valid() родительского контейнера этой кнопки. Итак, с чего ты хочешь начать рассмотрение списков combo и list?

Марсиа: Давай уточним о чем мы тут ведем речь. У нас есть три различных возможности: элемент управления Dropdown List (то есть элемент управления ComboBox, для которого свойство Style = 2), элемент управления Dropdown Combo (то есть элемент управления ComboBox, для которого свойство Style = 0) и элемент управления Scrolling ListBox.

Энди: Вполне достаточно. Сначала мы будем иметь дело с комбинированными списками. Чтобы следовать тому подходу, который мы применяли до сих пор, давай получим базовую последовательность событий для элементов управления, получающих и теряющих фокус. Затем мы можем углубиться в подробности того, что происходит при выборе элементов списка.

Марсиа: Как ни удивительно, на этом уровне все довольно просто. Элемент управления Dropdown List ведет себя точно таким же образом, что и командные кнопки, инициируя наступление одних только событий When(), GotFocus() и LostFocus(). С другой стороны, элемент управления Dropdown Combo ведет себя аналогично простым текстовым полям textbox и инициирует наступление событий When(), GotFocus(), Valid() и LostFocus().

Энди: Это представляется вполне очевидным. Конечно, как и в случае всех тех элементов управления, ко-

торые мы рассматривали до сих пор, использование мыши для перемещения фокуса, вместо простого перехода от одного элемента управления к другому по нажатию клавиши Tab, приводит к наступлению в промежутке между событиями Valid() и LostFocus() события When() выбранного элемента управления. Итак, что же происходит, когда мы пытаемся выбрать что-либо в комбинированном списке combo?

Марсиа: Самый простой случай, это когда список combo имеет фокус, а ты используешь мышь, чтобы открыть список и выбрать в нем некоторый элемент. При таком сценарии оба типа комбинированных списков ведут себя совершенно одинаково, и события наступают в следующей последовательности:

```
InterActiveChange()
Click()
Valid()
When()
```

Этот набор событий повторяется каждый раз, когда делается выбор какого-либо элемента списка, и значение свойства Value данного элемента управления обновляется в событии InterActiveChange(). Заметь также, что событие When() наступает после события Valid(), точно так же, как это происходит у командной кнопки.

Энди: Единственная странность заключается в том, что событие Click() не наступает до тех пор, пока не наступит событие InterActiveChange(). Это происходит из-за того, что событие InteractiveChange() «запускается» в результате обнаружения события MouseUp() для элемента списка, что означает следующее: ко времени наступления события Click(), свойство Value элемента управления уже получило обновление!

Марсиа: Правильно, и тут все немного усложняется, потому что способы интерпретации и обработки нажатий клавиш зависят от того, открыта ли часть списка или закрыта, а также от того, используешь ли ты клавиши управления курсором или алфавитно-цифровую клавиатуру. Лучше всего мы можем продемонстрировать это, организовав полученные сведения в виде строк и столбцов (см. таблицу 1).

Энди: Я понимаю, тебе необходимо было каким-то образом задействовать здесь элемент управления grid. Похоже на то, что события Keypress() и InteractiveChange() являются единственными событиями, на наступление которых мы можем рассчитывать во всех сценариях, но, как мы уже видели, свойство Value — это единственное свойство, которое обновляется при наступлении события InteractiveChange(). Этим закрывается вопрос о прокрутке списка. Но что делается, когда мы реально осуществляем выбор элемента в списке и закрываем выпадающий список, происходит еще что-то?

Марсиа: Ну, когда ты выполняешь прокрутку списка, используя клавиатуру, каждый элемент списка выбирается автоматически, но если используется мышь, тогда ты должен явно щелкнуть мышью по нужному элементу, чтобы осуществить его выбор. Что происходит в дальнейшем зависит от того, имеешь ли ты дело со списком, работающим в режиме Dropdown List или в режиме Dropdown Combo, от того, как ты выбираешь необходимый элемент, и от того, каким образом ты покидаешь область списка. Давай сначала разберемся со списком, работающим в режиме Dropdown List (см. таблицу 2).

Таблица 1. Последовательность событий при «прокрутке» элемента управления combo box.

	Keypress	InterActiveChange	Click	Valid	When
Режим Dropdown List, список List закрыт, клавиши управления курсором	x	x	x	x	x
Режим Dropdown List, список List закрыт, алфавитно-цифровая клавиатура	x	x		x	x
Режим Dropdown List, список List открыт, клавиши управления курсором/алфавитно-цифровая клавиатура	x	x			
Режим Dropdown Combo, список List открыт, клавиши управления курсором/алфавитно-цифровая клавиатура	x	x			

Таблица 2. Последовательность событий при выборе элемента в списке, работающем в режиме Dropdown List.

	Keypress	Click	Valid	When	LostFocus
Режим Dropdown List, список List открыт, клавиша Enter	x	x	x		
Режим Dropdown List, список List открыт, клавиша Tab	x	x	x	x	x
Режим Dropdown List, список List открыт, клавиша Spacebar	x	x	x	x	
Режим Dropdown List, список List открыт, щелчок мышью по другому элементу управления			x		x

Энди: Ого! Какое-то эксцентричное поведение. Почему это, если ты нажимаешь клавиши Tab или Spacebar, вслед за событием Valid() наступает событие When(), но этого не происходит при нажатии клавиши Enter или щелчке мышью по другому элементу?

Марсиа: Это выше моего понимания. Единственное, что остается неизменным в этих ситуациях, это то, что событие Valid() наступает в любом случае. Давай теперь посмотрим, что происходит, если вместо списка Dropdown List у нас есть список, работающий в режиме Dropdown Combo (см. таблицу 3).

Энди: Эта таблица почти идентична предыдущей за исключением того, что если ты нажимаешь клавишу Tab, чтобы покинуть область списка Dropdown Combo, наступает дополнительное событие Valid(). Вероятно, это событие Valid() принадлежит текстовому полю, которое является частью этого комбинированного списка, хотя кажется странным, что этого не происходит, если ты щелкаешь мышью по другому элементу управления.

Марсиа: Вероятно, это объясняется тем, что если ты щелкаешь мышью где-то за пределами списка, составное текстовое поле никогда не получает фокус, поэтому его событие Valid() не наступает повторно.

Энди: О да. Я уверен, что в этом ты права. А как насчет элементов управления ListBox? Пожалуйста, не говори мне, что они опять ведут себя по-разному.

Марсиа: Сожалею, но это так. Простейший случай: ListBox имеет фокус, а ты используешь мышь для прокрутки списка и выбора элемента. При таком сценарии наступают следующие события:

```
InterActiveChange()
Click()
When()
```

Энди: А! Это та же самая последовательность, что и для элементов управления combo, за исключением того, что отсутствует событие Valid().

Марсиа: В самом деле, единственный раз, когда для списка ListBox наступает событие Valid(), это когда ты выбираешь элемент, нажимая клавишу Enter или дважды щелкнув мышью по этому элементу. Поскольку элемент управления ListBox является очень простым элементом управления, мы можем свести все разнообразные операции с ним в одну таблицу. События, связанные с работой элемента управления ListBox, показаны в таблице 4.

Энди: Хорошо, мне понятно, что единственным отличием между прокруткой списка ListBox и прокруткой списка ComboBox является обязательное наступление событие When() элемента управления ListBox (для элементов управления ComboBox это событие наступает тогда, когда имеется список Dropdown List, и область списка закрыта). Предположительно, свойство Value обновляется при наступлении события InterActiveChange(), как и для списка combo?

Марсиа: Да, это так. Но помимо чисто академического интереса, который заключается в понимании того, как все эти события взаимодействуют друг с другом, реальная цель состоит в том, чтобы оказать помощь при принятии решения, где в этих элементах управления следует размещать программный код. Ключевым моментом, который надо взять на заметку относительно использования элемента управления ListBox, явля-

Таблица 3. Последовательность событий при выборе элемента в списке, работающем в режиме Dropdown Combo.

	Keypress	Click	Valid	When	Valid	LostFocus
Режим Dropdown Combo, список List открыт, клавиша Enter	X	X	X			
Режим Dropdown Combo, список List открыт, клавиша Tab	X	X	X	X	X	X
Режим Dropdown Combo, список List открыт, клавиша Spacebar	X	X	X	X		
Режим Dropdown Combo, список List открыт, щелчок мышью по другому элементу управления			X			X

Таблица 4. События элемента управления ListBox.

	Keypress	I/A Change	Click	DblClick	When	Valid	LostFocus
Прокрутка списка с помощью клавиш управления курсором	X	X	X		X		
Прокрутка списка с помощью клавиш алфавитно-цифровой клавиатуры	X	X			X		
Прокрутка списка с помощью клавиш Page Up/Down	X	X			X		
Выбор элемента списка по нажатию на клавишу Enter	X			X		X	
Выбор элемента списка по двойному щелчку мыши		X		X	X	X	
Покинуть область списка, нажав клавишу Tab	X						X
Покинуть область списка, щелкнув мышью на другом элементе управления							X

ется то, что в действительности нельзя полагаться на его событие Valid() для того, чтобы выполнить какое-либо действие при выборе элемента списка. Это объясняется тем, что это событие не наступает автоматически ни при изменении значения свойства Value, ни при потере фокуса списком ListBox.

Энди: Я вполне могу предугадать, в каких обстоятельствах это привело бы к возникновению проблемы. Как бы ты поступила в этих обстоятельствах?

Марсиа: Использовала событие InterActiveChange(). Оно наступает независимо от того, каким образом ты перебираешь элементы списка. Это событие не наступает лишь в одном случае: если ты используешь клавишу Enter (или двойной щелчок мыши), чтобы явно выбрать элемент или убрать фокус из элемента управления ListBox. Но в обоих этих случаях наступление события InterActiveChange() окажется уже свершившимся фактом как результат навигации к выбранному элементу.

Энди: Так значит этот подход применим также к элементам управления ComboBox?

Марсиа: Нет, в этом случае лучшим вариантом является использование события Valid(), поскольку ты не можешь покинуть элемент управления combobox без наступления этого события, но, в отличие от события InterActiveChange(), событие Valid() не наступает каждый раз, когда меняется выбранный элемент.

Энди: Фу-у! Придется мне все это тщательно систематизировать. Однако, мы исчерпали отведенное нам место и все еще не добрались до элементов управления grid, и к тому же — как я только что понял — мы не рассмотрели элементы управления pageframe или другие контейнеры.

Марсиа: Значит им придется подождать до публикации в следующем месяце.

❑❑❑

Интеграция браузера IE в VFP-приложения, часть 2

Реми Карон (Remi Caron)


В прошлый раз Реми Карон объяснил, как можно использовать браузер Internet Explorer в ваших собственных проектах. В опубликованной статье рассматривалась работа с готовыми Web-страницами и те действия, которые можно выполнять с внутренним содержимым этих страниц. В этом месяце все внимание уделяется использованию браузера IE в качестве средства, позволяющего создавать и редактировать содержимое этих Web-страниц; по ходу своих пояснений Реми Карон затрагивает некоторые важные компоненты объектной модели браузера Internet Explorer.

Прежде чем мы углубимся в ту часть статьи, которая посвящена редактированию содержимого Web-страниц, я хочу рассмотреть объектную модель браузера IE.

Самая большая проблема, с которой вы сталкиваетесь, приступая к использованию компонентов, созданных другой фирмой, — это получение сведений об объектной модели этих компонентов. Вам потребуются знания о том, какие свойства являются открытыми, когда наступают те или иные события, что делает каждый из представленных методов и где вы должны разместить свой собственный программный код, чтобы использовать этот элемент управления и расширить его возможности. В дополнение к чтению этой статьи вам также потребуется посетить Web-страницу, которую можно найти по адресу: www.msdn.microsoft.com/workshop/browser/default.asp.

Самый важный объект в модели браузера IE — это, разумеется, сам браузер. Когда вы будете просматривать те методы и команды, которые частично представлены в окне IntelliSense, вы увидите нечто схожее с тем, что изображено на рис. 1.

На этом рисунке курсор указывает на самое важное свойство — document. Это свойство обеспечивает вам доступ к содержимому документа — той самой странице, которая выдается на экран в окне браузера. Если вы теперь «оглядитесь вокруг» в этом окне, то увидите методы, в которых узнаются командные кнопки из инструментальной па-

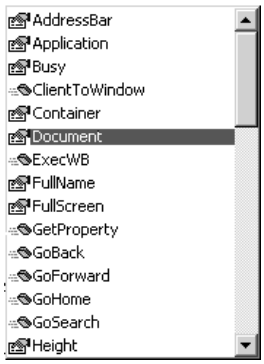

Рис. 1. Окно IntelliSense.

Рис. 2. Вот результат работы приведенного в примере программного кода, который зависит от того, какую страницу вы используете в качестве «домашней».

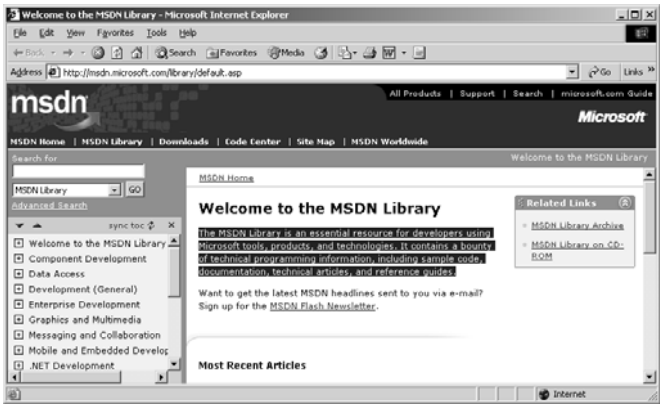

Рис. 3. Выбрать в браузере какой-то текст.

нели проводника Explorer, например, кнопки GoHome, GoForward и GoBack.

По мере того как вы будете знакомиться со структурой объекта document, вы обнаружите там массу знакомых свойств даже в том случае, если вы не являетесь опытным HTML-«программистом». Если вам необходима более подробная информация, обратитесь по адресу www.msdn.microsoft.com/webworkshop или воспользуйтесь справочными файлами HTMLRef.CHM и Iexplore.CHM Help, которые инсталлированы на вашей машине.

В этих файлах вы найдете сведения об объектных моделях и так далее. Если вы подумываете об использовании браузера IE в своих приложениях, я призываю вас к тому, чтобы вы прочли эти файлы и держали их открытыми во время работы.

Использовать браузер IE в качестве инструмента редактирования проще, чем это может показаться на первый взгляд. Ключевым методом, который вы будете использовать, является метод execCommand. Метод ExecCommand принадлежит уровню документа (см. рис. 2). Этот метод позволяет вам программно выполнять любого рода операции по редактированию. Среди возможностей, которые обеспечивает этот метод, использование шрифтов различного начертания, вставка изображений, создание гиперссылок и многое другое.

Во-первых, нам необходима ссылка на Internet Explorer, доступная в среде разработки. Создадим ее в следующей небольшой программе с помощью нескольких строк кода. Обратите внимание на то, что я создал экземпляр объекта Internet Explorer, заставив его отображать на экране мою домашнюю страницу, сделав его видимым и сохранив ссылку на документ в переменной с именем oDoc; все это уместилось в маленькой «пригоршне» кода.

Заметьте, что в первой части этой статьи для иллюстрации тех вещей, которые вам необходимо знать, чтобы заставить работать браузер IE, используется командное окно. Позже я продемонстрирую вам простую экранную форму, в которой сведу воедино рассматриваемый сейчас программный код с некоторыми приемами работы с браузером IE, которые я показал вам в прошлом месяце.

```
oIe = CREATEOBJECT ("InternetExplorer.application")
oIe.GoHome()
oIe.Visible = .t.
```

Как я уже упоминал ранее, метод execCommand функционирует на уровне документа. Поэтому, чтобы несколько облегчить себе жизнь, мы создаем прямую ссылку на документ, исполняя в командном окне следующее предложение:

```
oDoc = oIe.Document
```

Есть несколько вещей, о которых вы должны знать, прежде чем мы начнем использовать метод execCommand. Большая часть функциональных возможностей, которые обеспечиваются этим методом (например, использование для шрифта полужирного начертания или курсива), действует применительно к выбранному фрагменту документа, тогда как ряд других возможностей работают с учетом положения курсора (например, функция вставки изображения InsertImage). В таблице 1 перечислены все те команды, которые становятся доступны вам благодаря методу execCommand. Если вы обратитесь к базе знаний Knowledge Base, то обнаружите там список, в котором повторяются эти же самые команды, а также приводится ряд других команд, которые обозначены

Таблица 1. Команды, доступные благодаря методу execCommand.

Команда	Описание
2D-Position	Позволяет перемещать элементы, положение которых дается в абсолютных координатах, путем перетаскивания.
AbsolutePosition	Определяет координаты как абсолютные.
BackColor	Задает или считывает цвет заднего фона для выбранного в данный момент фрагмента документа.
Bold	Включает/выключает для выбранного фрагмента документа режим полужирного начертания шрифтов.
Copy	Копирует выбранный фрагмент документа в буфер обмена clipboard.
CreateBookmark	Создает анкер закладки (bookmark anchor) или запрашивает имя анкера закладки для выбранного в данный момент фрагмента документа или точки вставки.
CreateLink	Преобразует выбранный фрагмент документа в гиперссылку или выдает на экран диалоговое окно, позволяющее пользователю указать URL-локатор для его использования в качестве такой гиперссылки.
Cut	Копирует выбранный фрагмент документа в буфер обмена clipboard с последующим удалением этого фрагмента.
Delete	Удаляет выбранный фрагмент документа.
FontName	Задает или считывает шрифт для выбранного фрагмента документа.
FontSize	Задает или считывает размер шрифта для выбранного фрагмента документа.
ForeColor	Задает или считывает цвет переднего фона (текст) для выбранного фрагмента документа.
FormatBlock	Задает тэг текущего блока форматирования.
Indent	Увеличивает отступ для выбранного текста на одну единицу увеличения отступа.
InsertButton	Заменяет выделенный фрагмент элементом управления button.
InsertFieldset	Заменяет выделенный фрагмент полем box.
InsertHorizontalRule	Заменяет выделенный фрагмент горизонтальной линией.
InsertIFrame	Заменяет выделенный фрагмент встроенной рамкой (inline frame).
InsertImage	Заменяет выделенный фрагмент изображением.
InsertInputButton	Заменяет выделенный фрагмент элементом управления button.
InsertInputCheckbox	Заменяет выделенный фрагмент элементом управления check box.
InsertInputFileUpload	Заменяет выделенный фрагмент элементом управления file upload.
InsertInputHidden	Вставляет скрытый элемент управления в выбранный фрагмента документа.
InsertInputImage	Заменяет выделенный фрагмент элементом управления image.
InsertInputPassword	Заменяет выделенный фрагмент элементом управления password.
InsertInputRadio	Заменяет выделенный фрагмент элементом управления radio.
InsertInputReset	Заменяет выделенный фрагмент элементом управления reset.
InsertInputSubmit	Заменяет выделенный фрагмент элементом управления submit.
InsertInputText	Заменяет выделенный фрагмент элементом управления text.
InsertMarquee	Заменяет выделенный фрагмент документа пустой помеченной областью (marquee).
InsertOrderedList	Переключает для выбранного фрагмента документа режимы между ordered list и обычным форматируемым блоком.
InsertParagraph	Заменяет выбранный фрагмент документа разрывом строки (line break).
InsertSelectDropdown	Заменяет выбранный фрагмент документа элементом управления drop-down selection.
InsertSelectListbox	Заменяет выбранный фрагмент документа элементом управления list box selection.
InsertTextArea	Заменяет выбранный фрагмент документа элементом управления multiline text input.
InsertUnorderedList	Включает/отключает для выбранного фрагмента документа режим ordered list.
Italic	Включает/отключает для выбранного фрагмента документа начертание шрифта курсив.
JustifyCenter	Выравнивает по центру блок форматирования, в котором находится выбранный фрагмент документа.
JustifyLeft	Выравнивает по левому краю блок форматирования, в котором находится выбранный фрагмент документа.
JustifyRight	Выравнивает по правому краю блок форматирования, в котором находится выбранный фрагмент документа.
LiveResize	Заставляет редактор MSHTML Editor постоянно обновлять элементы во время изменения размеров или перемещения, вместо того, чтобы выполнить обновление только после завершения этих операций.
MultipleSelection	Позволяет выбрать несколько разрешенных для выбора элементов сайта одновременно, когда пользователь удерживает в нажатом состоянии клавиши Shift или Ctrl.
Outdent	Уменьшает на одну единицу отступ блока форматирования, в котором находится выбранный фрагмент.
OverWrite	Переключает режим ввода текста между insert и overwrite.
Paste	Перезаписывает выбранный фрагмент документа содержимым буфера обмена clipboard.
Print	Открывает диалоговое окно печати, чтобы пользователь мог напечатать текущую страницу.
Refresh	Обновляет текущий документ.
RemoveFormat	Удаляет тэги форматирования из выбранного фрагмента документа.
SaveAs	Сохраняет текущую Web-страницу в файле.
SelectAll	Выбирает весь документ.
UnBookmark	Удаляет все закладки из выбранного фрагмента документа.
Underline	Включает/выключает для выбранного фрагмента документа режим начертания шрифта с подчеркиванием.
Unlink	Удаляет все гиперссылки из выбранного фрагмента документа.
Unselect	Отменяет выбор данного фрагмента документа.

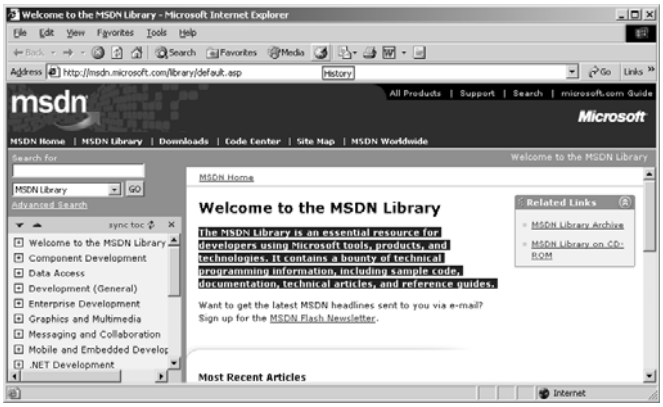

Рис. 4. Результат выполнения предложения execCommand.

как «not supported» — «не поддерживается». Возможно, поддержка этих команд будет обеспечена в будущем, но никто не знает, когда именно это произойдет.

Для иллюстрации работы метода execCommand проделайте следующее:

- 1. Выберите некоторый текст на странице, отображаемой в окне объекта браузера, который мы активизировали из окна Command Window (как показано на рис. 3).
- 2. Вернитесь назад в окно Command Window.
- 3. Введите строку: oDoc.ExecCommand("Bold").
- 4. Отмените выбор текста в браузере и обратите внимание на то, что теперь для его содержимого использован полужирный шрифт (см. рис. 4).

Еще один тип функционирования основывается на том, что курсор находится в определенном месте, например, команда InsertImage. Команда InsertImage может выдавать на экран дополнительное диалоговое окно, чтобы облегчить выбор изображения, предназначенного для вставки в документ. Полный синтаксис этой команды таков:

```
oDoc.ExecCommand("InsertImage", .T.)
```

Второй параметр обеспечивает появление экрана, представленного на рис. 5. Есть еще несколько других команд, которые также имеют возможность открытия диалогового окна, которую вы можете инициировать, передав значение .T. в качестве второго параметра.

Вы, вероятно, заметили, что все наши действия — это манипуляции с содержимым готовых страниц, но в начале этой статьи я обещал, что мы будем использовать проводник Explorer в качестве инструмента редактирования! Сейчас мы займемся этим, используя те возможности, о которых я уже рассказал.

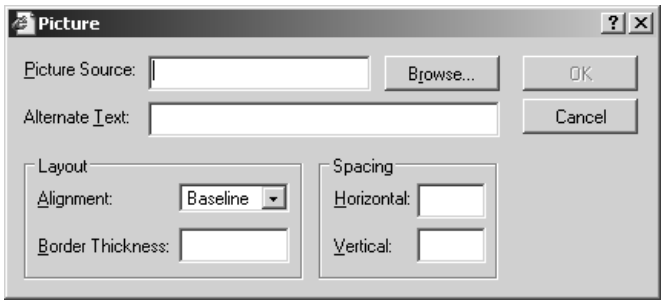

Рис. 5. Метод ExecCommand вызывает дополнительную экранную форму в браузере Internet Explorer.

Нам потребуется HTML-страница с редактируемым тэгом <DIV> внутри, чтобы мы могли вводить текст в форму, а не только просматривать его. Редактируемый тэг <DIV> — это используемой мной пример, и он работает следующим образом. Задав для свойства ContentEditable тэга <DIV> значение .T., мы можем получить доступ к информации, хранящейся в тэге <DIV>, прямо из браузера, и таким образом определить или изменить содержимое этого тэга. Существуют дополнительные HTML-элементы управления, которые также демонстрируют такое свойство.

Я создал вот такой простой HTML-файл:

```
<html>

<head>
<meta http-equiv="Content-Language"
content="en-us">
<meta http-equiv="Content-Type"
content="text/html;
charset=windows-1252">
<meta name="GENERATOR" content=
  "Microsoft FrontPage 4.0">
<meta name="ProgId" content=
  "FrontPage.Editor.Document">
<title>Content Management Tool</title>
</head>

<body>

<p align="center"><b><font
  color="#0000FF" size="6">
  Internet Explorer editor</font></b></p>
<p align="center">

<DIV id="oEditRegion" CONTENTEDITABLE STYLE="height:
  100%; width: 100%;z-index:0; visibility:visible;
  background-color:#FFFFFF">Here's an
  edit region </DIV>

</body>

</html>
```

В первой статье я объяснил, как использовать элемент управления браузера в форме, так что здесь не буду снова повторять это объяснение. На рис. 6 я демонстрирую форму, где размещены браузер и несколько командных кнопок button, которые обраща-

ются к методу execCommand для того, чтобы выполнить форматирование определенного типа.

Программный код инициализации этой экранной формы выглядит вот так:

```
* Код инициализации
This.obrowser.Navigate(CURDIR() ;
  + "CONTENT_MANAGEMENT_TOOL.HTM")
```

Объект oBrowser — это элемент управления MS Web browser, в который загружается приведенный ранее HTML-код. Теперь у вас есть редактируемая HTML-страница в FoxPro-форме, которая использует Web-браузер в качестве редактора.

Код для командных кнопок выглядит вот так:

```
* Полужирный шрифт:
   ThisForm.oBrowser.Document.ExecCommand("Bold")

* Курсив:
   ThisForm.oBrowser.Document.ExecCommand("Italic")

* Подчеркивание:
   ThisForm.oBrowser.Document.ExecCommand("underline")

* Вставка гиперссылки:
   ThisForm.oBrowser.Document.ExecCommand("CreateLink",;
    .T.)

* Вставка изображения:
   ThisForm.oBrowser.Document.ExecCommand("InsertImage",;
    .T.)
```

В своей предыдущей статье я продемонстрировал вам, как извлечь данные из Web-страницы и сохранить их в таблице для последующего использования. Если вы объедините эту идею с функциями редактирования, которые я продемонстрировал в этом месяце, то можете комбинировать эти две возможности и использовать их в любом необходимом вам сочетании для того, чтобы создавать отличные инструменты для своих заказчиков.

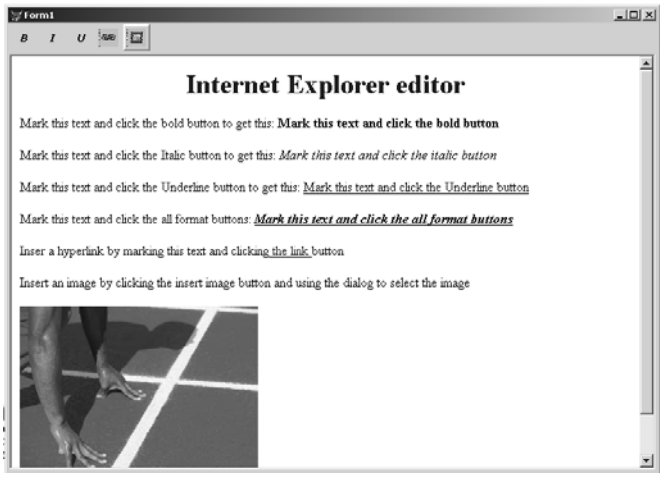

Рис. 6. Использование браузера Internet Explorer в качестве редактора текста.

Как вы и сами можете видеть, все остальное — это не полет в космос, но всего лишь повторное использование готового программного кода. Надеюсь, эти две статьи, посвященные использованию браузера Internet Explorer, продемонстрировали вам, как почти без труда можно использовать этот прекрасный компонент в ваших собственных приложениях. Воспользуйтесь им.

❑❑❑

FoxTalk *русское издание*

Печатается ежемесячно

Учредитель и издатель:

ООО Эдэль.

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

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

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

Журнал зарегистрирован комитетом Российской Федерации по печати.

Регистрационное свидетельство № 015520 от 17.12.1996

Подписано в печать 20/04/03. Формат 60x90 1/8. Тираж 330 экз.