УРОК Программирование для Интернета на Visual Basic

- Использование элемента WebBrowser
- Разработка приложений Winsock
- Проектирование документов ActiveX
- Использование DHTML

В Visual Basic существуют различные средства, облегчающие разработку Интернет-приложений. Вы можете создавать мощные приложения, произвольно сочетая HTML-код (статический и динамический), документы ActiveX и элементы ActiveX, ориентированные на Интернет.

В этом уроке мы познакомимся со средствами разработки Интернет-приложений и воспользуемся ими для создания чат-комнаты WebComm. Наше приложение будет состоять из нестандартного Web-броузера, который запускает страницу, написанную на DHTML. Эта страница, в свою очередь, запускает клиента WebComm, который на самом деле является документом ActiveX, написанным на Visual Basic. Клиент WebComm использует элемент WinSock для обмена информацией с централизованным сервером чат-комнаты. Такое приложение легко создать и модифицировать для обслуживания дополнительных пользователей или расширения его возможностей. К концу этого урока вы научитесь писать на Visual Basic собственные Интернет-приложения!

Использование элемента WebBrowser

Как нетрудно догадаться по названию, элемент WebBrowser наделяет приложения, написанные на Visual Basic, возможностями Web-броузера. Разве не приятно вывести на экран Web-страницу, не запуская броузер? Хорошим примером может служить новая справочная система Microsoft на базе HTML. Хотя Microsoft Internet Explorer 4.01 должен быть установлен на компьютере, справочная система не запускает его. Вместо этого запускается приложение, в интерфейсе которого присутствует элемент WebBrowser (рис. 18.1).

Пример из этого раздела показывает, как элемент WebBrowser применяется для создания нестандартных приложений с функциями броузера. Вероятно, у вас возник вопрос — зачем создавать новый броузер, для которого вдобавок нужен еще один, уже установленный на вашем компьютере? Ответ прост: это позволяет интегрировать определенные Web-узлы, документы ActiveX и другие средства способом, не предусмотренным в стандартном броузере. Сейчас вы поймете, о чем я говорю.

1. Создайте новый проект клавишами Ctrl+N. В диалоговом окне New Project выберите тип проекта Standard EXE и нажмите кнопку OK.
2. Задайте свойству `Name` проекта значение `WebIndex`.
3. Задайте свойству `Name` формы `Form1` значение `frmMain`, а свойству `Caption` — значение `WebIndex`.
4. Щелкните правой кнопкой мыши на панели элементов и выполните команду Components из контекстного меню. На экране появляется диалоговое окно Components, изображенное на рис. 18.2.
5. Установите флажок рядом со строкой Microsoft Internet Controls, чтобы добавить элемент WebBrowser на панель элементов.
6. Установите флажки рядом со строками Microsoft Windows Common Controls 6.0 и Microsoft Windows Common Controls-3 6.0.

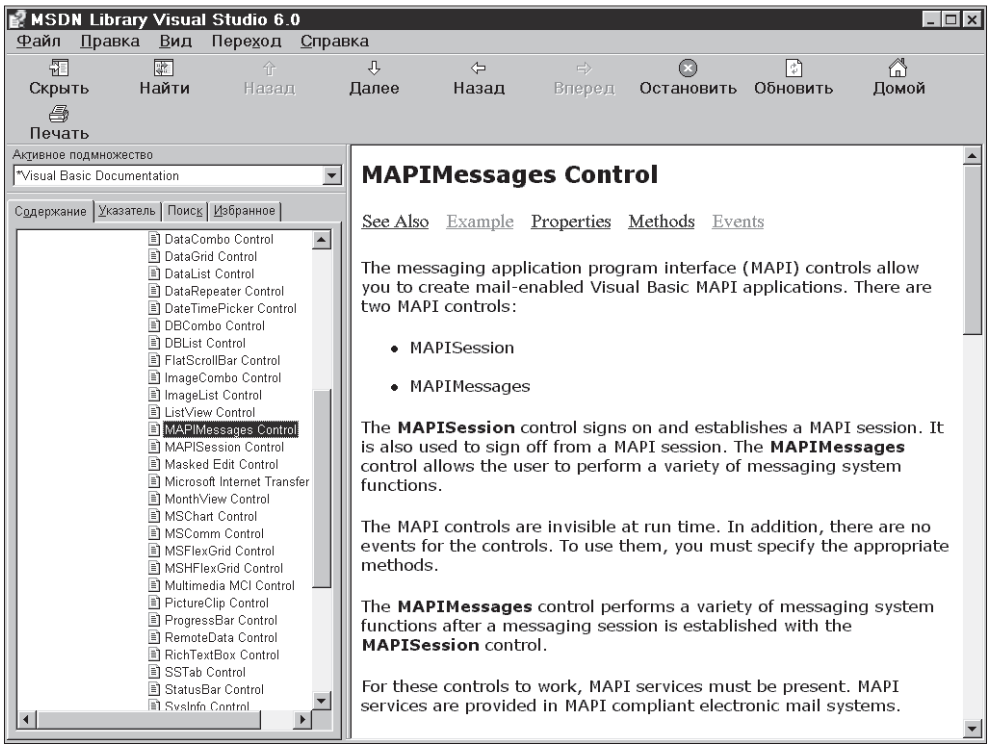


Рис. 18.1. *В справочной системе на базе HTML использован интерфейс WebBrowser*

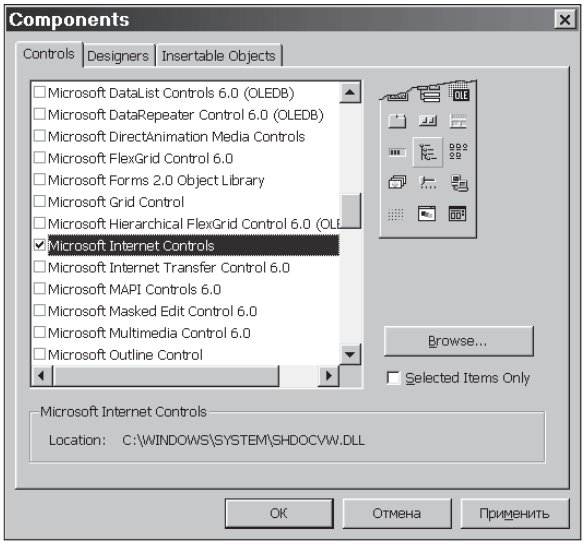


Рис. 18.2. *Диалоговое окно Components*

7. Закройте диалоговое окно Components кнопкой OK. Добавленные компоненты появляются на панели элементов.
8. Создайте усовершенствованную панель (CoolBar) в верхней части формы frmMain и задайте ее свойству Name значение cbrCoolbar.
9. Поместите на cbrCoolbar другой элемент — панель инструментов. Проследите за тем, чтобы панель инструментов была создана именно в cbrCoolbar, а не на самой форме.
10. Задайте свойству Name панели инструментов значение tbrToolbar.
11. Создайте список в левой части формы. Растяните его примерно на середину ширины формы и задайте свойству Name значение lstSites.
12. Создайте элемент WebBrowser в правой части формы, под панелью инструментов. Задайте его свойству Name значение WebBrowser.
13. Создайте на форме список изображений, задайте его свойству Name значение imlToolbar.
14. Сохраните проект, чтобы не потерять внесенные изменения. Примерный вид формы показан на рис. 18.3.

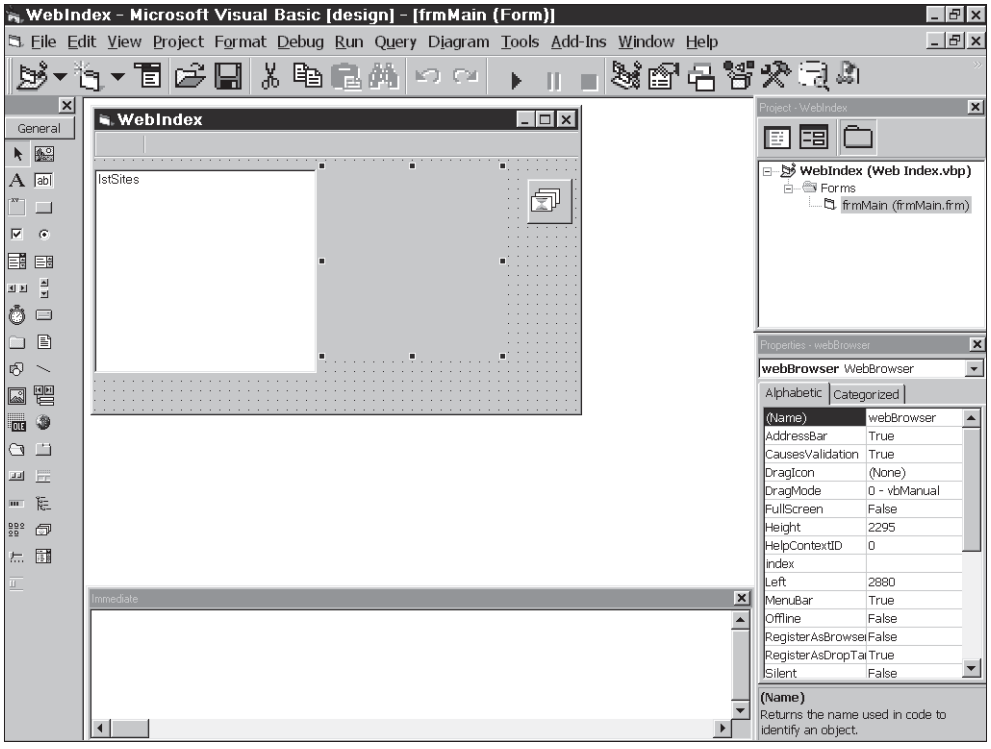


Рис. 18.3. *Форма WebIndex в режиме конструирования*

15. Щелкните на элементе ImageList, затем дважды щелкните в строке (Custom) окна свойств. Открывается диалоговое окно со страницами свойств элемента.

16. Перейдите на вкладку Images.
17. Нажмите кнопку Insert Image, чтобы добавить изображение в список.
18. В открывшемся диалоговом окне Select Picture выберите файл Undo.bmp из каталога \Common\graphics\Bitmaps\Tlbr_Win95 и нажмите кнопку Open, чтобы добавить изображение в список.
19. Снова нажмите кнопку Insert Image, чтобы добавить другое изображение. Выберите файл Redo.bmp из того же каталога и снова нажмите кнопку OK.
20. Еще раз нажмите кнопку Insert Image. Выберите файл Find.bmp из того же каталога. Нажмите кнопку OK — последнее изображение заносится в список. Примерный вид страницы показан на рис. 18.4.

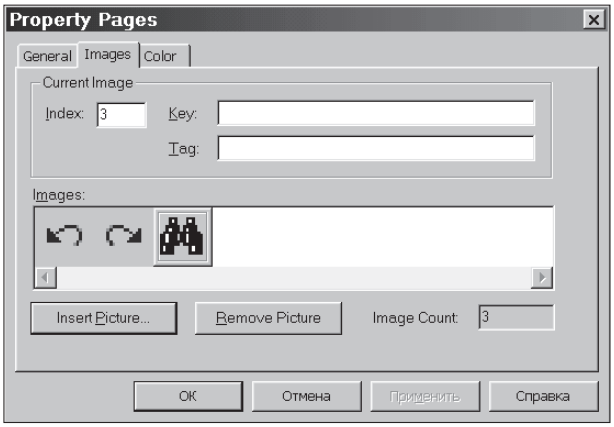


Рис. 18.4. *Список с изображениями*

21. Закройте страницы свойств кнопкой OK.
22. Щелкните на панели инструментов tbrToolbar в конструкторе форм, чтобы активизировать ее. Дважды щелкните в строке (Custom) окна свойств — открывается диалоговое окно со страницами свойств панели.
23. На вкладке General выберите в списке Image List строку imlToolbar, а в списке Style — строку 1 – tbrFlat. Когда это будет сделано, перейдите на вкладку Buttons.

ПОДСКАЗКА В панелях с большим количеством кнопок изображения часто подключаются из кода программы в режиме выполнения. Это упрощает добавление и удаление изображений при внесении в приложение новых возможностей.

24. Нажмите кнопку Insert Button, чтобы добавить новую кнопку на панель инструментов. Введите в поле Key значение Back, а в поле Image — значение 1. Подтвердите внесенные изменения кнопкой Apply.

25. Снова нажмите кнопку Insert Button и добавьте на панель еще одну кнопку. Введите в поле Key значение Forward, а в поле Image — значение 2.
26. Снова нажмите кнопку Insert Button и добавьте на панель следующую кнопку. Введите в поле Style значение 3 - tbrSeparator.
27. Добавьте еще одну кнопку. Введите в поле Key значение Search, а в поле Image — значение 3. Закройте диалоговое окно со страницами свойств кнопкой OK.
28. Щелкните на усовершенствованной панели. Откройте ее страницы свойств, дважды щелкнув в строке (Custom) в окне свойств.
29. Перейдите на вкладку Bands.
30. Нажимайте кнопку Remove Band до тех пор, пока в поле Index не появится 1. В усовершенствованной панели остается одна полоса.
31. Задайте свойству Child значение tbrToolbar и нажмите кнопку OK. Форма должна выглядеть так, как показано на рис. 18.5.

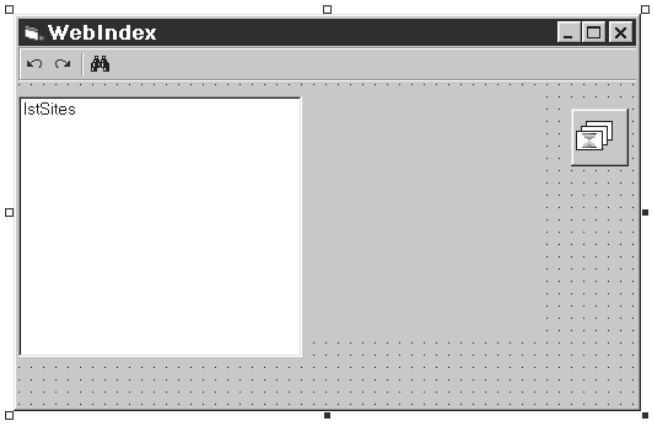


Рис. 18.5. *Форма приложения WebIndex*

Визуальное конструирование приложения завершено. В последующих шагах мы добавим код в форму:

1. Дважды щелкните на форме frmMain — открывается окно программы.
2. Вставьте следующий фрагмент в процедуру события Load формы frmMain:

```
Private Sub Form_Load()
    lstSites.AdItem "www.piter-press.ru"
    lstSites.AdItem "www.microsoft.com"
    lstSites.AdItem "www.myle.com"
End Sub
```

3. Вставьте следующий фрагмент в процедуру события Form_Resize:

```
Private Sub Form_Resize()
    If WindowState <> vbMinimized Then
```

```
        ' Растянуть панель
        cbrCoolbar.Move 0, 0, ScaleWidth

        ' Изменить размер списка узлов
        With lstSites
            .Move 0, cbrCoolbar.Height, .Width, _
                (ScaleHeight - cbrCoolbar.Height)
        End With

        ' Изменить размер элемента WebBrowser
        With webBrowser
            .Move .Left, cbrCoolbar.Height, _
                 (ScaleWidth - .Left), _
                 (ScaleHeight - cbrCoolbar.Height)
        End With
    End If
End Sub
```

Элемент WebBrowser загружает Web-страницы с помощью метода `Navigate`. При вызове этого метода компоненты Internet Explorer извлекают содержимое заданного URL и отображают его в элементе WebBrowser.

4. Чтобы наше приложение обладало навигационными возможностями, вставьте следующий код в процедуру события `Click` элемента `lstSites`:

```
Private Sub lstSites_Click()
    ' Перейти к выбранному узлу
    webBrowser.Navigate = Trim$(lstSites.Text)
End Sub
```

5. Давайте расширим навигационные средства броузера. Вставьте следующий код в процедуру события `ButtonClick` панели инструментов `tbrToolbar`:

```
Private Sub trToolbar_ButtonClick(ByVal Button As _
    ComctlLib.Button)
    ' Пропустить проверку ошибок
    On Error Resume Next

    ' Какая кнопка была нажата?
    Select Case UCase$(Trim$(Button.Key))
        Case Is = "BACK"
            webBrowser.GoBack
        Case Is = "FORWARD"
            webBrowser.GoForward
        Case Is = "SEARCH"
            webBrowser.GoSearch
    End Select
End Sub
```

6. Сохраните проект, чтобы не потерять внесенные изменения. Затем запустите приложение клавишей F5.

Щелкните в строке www.piter-press.ru — появляется домашняя страница издательства «Питер» (рис. 18.6). Если щелкнуть на любой ссылке данной страницы, элемент WebBrowser перенесет вас в заданное место. Вы даже можете щелкнуть на странице правой кнопкой мыши, просмотреть ее HTML-код или вывести на печать — точно так же, как это делается в Internet Explorer!

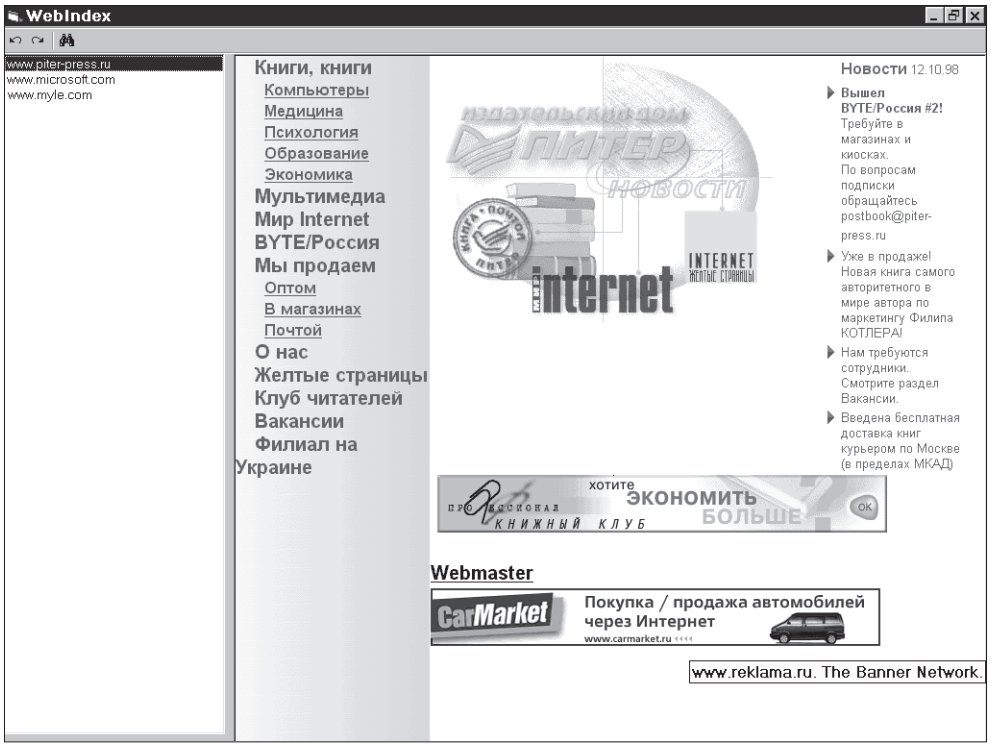


Рис. 18.6. *Web-узел издательства «Питер» в элементе WebBrowser*

Если ваши странствия в Web зайдут слишком далеко, вы всегда можете вернуться назад с помощью кнопки Back на панели инструментов. Кроме того, можно перемещаться в прямом направлении или перейти на поисковую Web-страницу Microsoft для поиска нужного узла. Завершив просмотр страниц, закройте приложение и вернитесь в Visual Basic.

Перед тем как переходить к следующему примеру, обратите внимание на ряд моментов. Во-первых, список узлов был заполнен в процедуре события `Form_Load`. Вы можете добавить в него новые узлы с помощью метода `AddItem`.

Вся основная работа выполняется в процедурах события `Click` элемента `lstSites` и `ButtonClick` элемента `tbrToolbar`. Как говорилось ранее, для загрузки конкретного

URL в элемент WebBrowser применяется метод `Navigate`. Кроме того, в процедуре события `ButtonClick` вызываются два метода.

Метод `GoBack` используется для обратного перемещения среди посещенных страниц. Вызов этого метода кнопкой **Back** избавит пользователя от долгих странствий в лабиринте Web. Как нетрудно догадаться, метод `GoForward` используется для прямого перемещения в списке недавно посещенных URL.

Наконец, метод `GoSearch` используется для перехода к стандартной поисковой странице, указанной в Internet Explorer.

Теперь вы хорошо представляете, как создать клиентское приложение с функциями Web-броузера. Перейдем к элементу Winsock, предназначенному для работы с функциональными средствами более низкого уровня.

Разработка приложений Winsock

Элемент Winsock упрощает работу с протоколами TCP (Transmission Control Protocol) и UDP (User Datagram Protocol). Вы можете использовать оба протокола в своих приложениях для разработки профессиональных коммуникационных программ и утилит. Для TCP необходимо существование *сеанса*, тогда как протокол UDP *не ориентирован на соединение*.

Сеансом (session) называется изолированный логический канал между двумя приложениями, по которому осуществляется их взаимодействие. Сеанс можно рассматривать как аналог телефонного звонка. Чтобы два приложения могли взаимодействовать друг с другом, приложение-клиент передает приложению-серверу запрос на соединение (подобно тому, как вы набираете телефонный номер своего друга). Приложение-сервер оповещается о поступившем запросе (раздается телефонный звонок) и решает, будет оно отвечать или нет. Если запрос принят, создается сеанс, и между приложениями устанавливается двухсторонняя связь. Взаимодействие продолжается до тех пор, пока одна из сторон не прервет сеанс (не повесит трубку).

Протокол UDP *не ориентирован на соединение*. По принципу действия он напоминает двухстороннюю радиосвязь. Вместо того чтобы создавать конкретный сеанс, приложения узнают о существовании друг друга. В UDP для этого указывается имя хоста или IP-адрес и номер порта другого приложения. После того как «заочное знакомство» состоится, приложения могут общаться друг с другом.

В этом уроке мы воспользуемся протоколом TCP для создания приложения чат-комнаты, которое будет называться WebComm. С помощью этого приложения в интрасети или в Интернете можно организовать переговоры, совместное обсуждение проектов или просто дружескую беседу. Впрочем, для какой бы цели вы ни использовали приложение, описанный далее процесс демонстрирует подробности организации связи на основе протокола TCP/IP.

Начнем с построения сервера чат-комнаты, использующего элемент Winsock. Следующие действия закладывают фундамент работы приложения WebComm:

1. Создайте новый проект командой File ➤ New Project. В диалоговом окне New Project выберите значок Standard EXE и нажмите кнопку OK.
2. Задайте свойству Name проекта значение WebCommServer.
3. Задайте свойству Name формы Form1 значение frmMain, а свойству Caption — значение Сервер WebComm.
4. Дважды щелкните в строке Icon в окне свойств. Выберите файл W95mbx04.bmp из каталога \Common\Graphics\Icons\Computer. Нажмите кнопку Open, чтобы поместить значок на форму.
5. Запустите редактор меню командой Tools ➤ Menu Editor.
6. Создайте меню верхнего уровня. Задайте его свойству Name значение mnuFile, а свойству Caption — значение &File.
7. Создайте в меню mnuFile команду с именем mnuFileExit. Задайте свойству Caption значение E&xit.
8. Закройте редактор меню кнопкой OK.
9. Создайте на форме список с именем lstMessages.
10. Поместите на форму таймер, задайте его свойству Name значение tmrTimer, а свойству Interval — значение 1000.
11. Щелкните правой кнопкой мыши на панели элементов и выполните команду Components из контекстного меню.
12. Установите флажки рядом со строками Microsoft Windows Common Controls 6.0 и Microsoft Winsock Control 6.0. Закройте диалоговое окно кнопкой OK. На панели появляются новые элементы.
13. Поместите на форму элемент Winsock. Для простоты задайте его свойству Name значение wsk, а свойству Index — значение 0. Позднее мы воспользуемся им для создания массива элементов. Не беспокойтесь об остальных свойствах, они будут заданы во время выполнения программы.
14. Поместите на форму frmMain строку состояния. Задайте ее свойству Name значение stsStatus и проследите за тем, чтобы свойство Align имело значение 2 - vbAlignBottom.
15. Дважды щелкните в строке (Custom) окна свойств, чтобы открыть диалоговое окно со страницами свойств строки состояния.
16. В открывшемся окне перейдите на вкладку Panels.
17. Введите в поле Text строку Сеансы: 0.
18. Нажмите кнопку Insert Panels, чтобы добавить в строку еще одну панель. Когда панель будет добавлена, введите в поле Index значение 2 — дальнейшие изменения будут относиться ко второй панели.
19. Выберите из списка Bevel строку 0 – sbrNobevel, а из списка AutoSize — строку 1 – sbrSpring.
20. Снова нажмите кнопку Insert Panel. Когда появится следующая панель, введите в поле Index значение 3 (работа с третьей панелью). Выберите из списка Style строку 6 – sbrDate.

21. Добавьте еще одну панель и введите в поле Index значение 4 (работа с четвертой панелью). Выберите из списка Style строку 5 – sbrTime.
22. После добавления всех панелей закройте диалоговое окно кнопкой OK. Примерный вид формы показан на рис. 18.7.

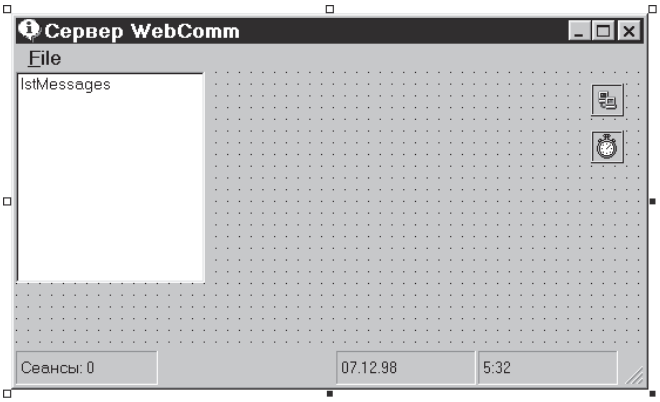


Рис. 18.7. *Сервер WebComm в режиме конструирования*

23. Сохраните проект, чтобы не потерять внесенные изменения.

Перед нами лежат фрагменты головоломки. Давайте соберем их воедино и добавим в приложение необходимый код:

1. Дважды щелкните на форме frmMain, чтобы открыть окно программы.
2. Вставьте следующий фрагмент в секцию (General)(Declarations) формы:

```
Option Explicit

Private Const WSK_LOCAL_PORT = 12345
Private Const MAX_SESSIONS = 11
```

Первая константа сообщает серверу о том, что он должен использовать локальный порт с номером 12345. К этому порту подключаются клиенты WebComm, желающие вступить в разговор. Вторая константа определяет максимальное количество одновременно открытых сеансов — 11. На самом деле это число на 1 превышает количество пользователей, которые могут одновременно общаться друг с другом, — первый сеанс используется сервером для прослушивания запросов на подключение. При получении запроса сервер WebComm перенаправляет сеанс на один из 10 элементов Winsock и продолжает ждать поступления следующего запроса.

3. Каждый раз, когда на сервере что-нибудь происходит (например, подключается или отключается пользователь), обновляется строка состояния формы. Для этой цели мы используем процедуру UpdateStatus. Вставьте следующий фрагмент в секцию (General)(Declarations) формы frmMain:

```
Private Sub UpdateStatus()
    Dim i As Integer
```

```
    Dim actsess As Integer

    ' Подсчитать количество активных сеансов
    For i = 1 To MAX_SESSIONS
        With wsk(i)
            If .State = sckConnected Then
                ' Увеличить счетчик сеансов
                actsess = actsess + 1
            End If
        End With
    Next

    ' Обновить строку состояния
    With stsStatus
        .Panels(1).Text = "Сеансы: " & _
            Trim$(Str$(actsess))
    End With
End Sub
```

Процедура `UpdateStatus` проверяет текущее состояние всех элементов Winsock в массиве. Если элемент в настоящий момент подключен к клиенту, она увеличивает переменную-счетчик `actsess`. После завершения цикла значение `actsess` отображается в первой панели строки состояния.

4. Следующий фрагмент создает массив элементов Winsock и настраивает протокол для каждого элемента. Вставьте следующий фрагмент в процедуру события `Form_Load`:

```
Private Sub Form_Load()
    Dim i As Integer

    ' Задать параметры получателя Winsock
    With wsk(0)
        .Protocol = sckTCPProtocol
        .LocalPort = WSK_LOCAL_PORT
        .Listen
    End With

    ' Загрузить еще 10 элементов Winsock
    For i = 1 To MAX_SESSIONS
        Load wsk(i)
        wsk(i).Protocol = sckTCPProtocol
        wsk(i).LocalPort = WSK_LOCAL_PORT
    Next

    ' Обновить строку состояния
    UpdateStatus
End Sub
```

5. Вставьте следующий фрагмент в процедуру события Form_Resize:

```
Private Sub Form_Resize()
    ' Растянуть список сообщений
    If WindowState <> vbMinimized Then
        lstMessages.Move 0, _
            0, _
            ScaleWidth, _
            (ScaleHeight - stsStatus.Height)
    End If
End Sub
```

6. Вставьте следующий фрагмент в процедуру события Click меню mnuFileExit:

```
Private Sub mnuFileExit_Click()
    ' Завершить приложение
    Unload Me
End Sub
```

7. Строка состояния должна обновляться каждую секунду. Вставьте следующий фрагмент в процедуру события tmrTimer_Timer():

```
Private Sub tmrTimer_Timer()
    ' Обновить строку состояния
    UpdateStatus
End Sub
```

Все основные функции сервера WebComm выполняются массивом элементов Winsock. Как будет видно из текста программы, мы используем многие события массива wsk.

Первое из событий, связанных с элементами Winsock, — ConnectionRequest. Следующий фрагмент отвечает за получение запроса, проверку наличия открытых сеансов Winsock и перенаправление запроса соответствующему сеансу.

8. Вставьте следующий фрагмент в процедуру события ConnectionRequest() массива элементов wsk:

```
Private Sub wsk_ConnectionRequest(Index As Integer, _
       ByVal requestID As Long)
    Dim msg As String
    Dim i As Integer

    ' Если это прослушивающий порт...
    If Index = 0 Then
        ' Ищем открытый сеанс
        For i = 1 To MAX_SESSIONS
            With wsk(i)
                If .State = sckClosed Then
                    ' Подключиться к сеансу i
```

```
                    .Accept requestID

                    ' Выйти из цикла
                    Exit For
                End If
            End With
        Next
    End If
End Sub
```

Параметр `Index` определяет, какой из элементов массива вызвал данное событие. Поскольку нам известно, что `wsk(0)` занят прослушиванием, программа проверяет, можно ли перенаправить запрос на соединение другому элементу. Параметр `RequestID` представляет собой уникальный идентификатор клиента, запросившего соединение. Он используется для подтверждения запроса на соединение конкретным элементом Winsock. После того как элемент Winsock согласится принять сеанс, инициируется событие `Connect` для данного элемента.

9. Вставьте следующий фрагмент в процедуру события `Connection()` массива элементов `wsk`. В нем происходит принудительное обновление строки состояния при подключении к серверу нового пользователя:

```
Private Sub wsk_Connect(Index As Integer)
    ' Обновить строку состояния
    UpdateStatus
End Sub
```

Вся основная работа происходит в следующем фрагменте. Он получает сообщение от клиента и перенаправляет его всем активным сеансам. Таким образом, приложение наделяется функциональными возможностями чат-комнаты. Когда от сеанса поступают данные, они заносятся в буфер и инициируется событие `DataArrival()`, которое сообщает элементу Winsock о наличии данных, подлежащих обработке. После этого следует написать обработчик для получения данных методом `GetData`.

10. Вставьте следующий фрагмент в процедуру события `wsk_DataArrival`:

```
Private Sub wsk_DataArrival(Index As Integer, _
       ByVal bytesTotal As Long)
    Dim msg As String
    Dim rc As Integer
    Dim i As Integer

    ' Получить сообщение
    wsk(Index).GetData msg,  , bytesTotal

    ' Занести в список
    lstMessages.AddItem msg
```

```
        ' Передать сообщение остальным клиентам
        For i = 1 To MAX_SESSIONS
            With wsk(i)
                If .State = sckConnected Then
                ' Передать сообщение текущему клиенту
                .SendData msg

                ' Подождать, пока сообщение будет доставлено
                DoEvents
            End With
        Next
    End Sub
```

Этот фрагмент работает весьма прямолинейно. Строка

```
wsk(Index).GetData msg,  , bytesTotal
```

извлекает из буфера данные, предназначенные для элемента `wsk(Index)`. Используя метод `GetData`, она извлекает количество байт данных, не превышающее `bytesTotal`, и помещает их в строку `msg`. После извлечения данных из буфера сообщение заносится в список — администратор увидит, что говорят пользователи. При желании можно добавить код, который будет сохранять каждое сообщение в ASCII-файле или базе данных с помощью ADO. После того как сообщение будет зарегистрировано, сервер передаст его всем активным сеансам (получается своего рода широковещательная рассылка).

11. Последнее событие, происходящее во время сеанса, — событие `Close()`. При отключении пользователя мы просто обновляем строку состояния. Вставьте следующий фрагмент в процедуру события `Close`:

```
Private Sub wsk_Close(Index As Integer)
    ' Обновить строку состояния
    UpdateStatus
End Sub
```

12. Сохраните проект.

Запустите приложение клавишей F5 — необходимо убедиться в том, что оно работает без ошибок. Если приложение запускается успешно, откомпилируйте проект в EXE-файл (команда **File ➤ Make WebCommServer.exe** в меню Visual Basic). EXE-файл понадобится нам в следующем разделе.

ПРИМЕЧАНИЕ О компиляции приложений и построении выполняемых файлов (EXE-файлов) рассказано в уроке 14.

Хотя наше приложение запускается и в локальном режиме, оно выглядит более эффектно, если запустить его на отдельном компьютере и наблюдать за поступлением и регистрацией сообщений в списке. С точки зрения клиента, происхо-

дит нечто замечательное — данные пересылаются по сети на другой компьютер и обратно. Но давайте продолжим работу над нашим примером. Когда он будет готов, вы сможете запустить серверный и клиентский компоненты WebComm и начать общение!

Проектирование документов ActiveX

Как вы узнали из урока 17, с помощью документов ActiveX вы сможете распространить возможности Visual Basic на Интернет или вашу интрасеть. Документы ActiveX представляют собой приложения Visual Basic, в которых функции контейнера выполняет Microsoft Internet Explorer 3 (или более новая версия).

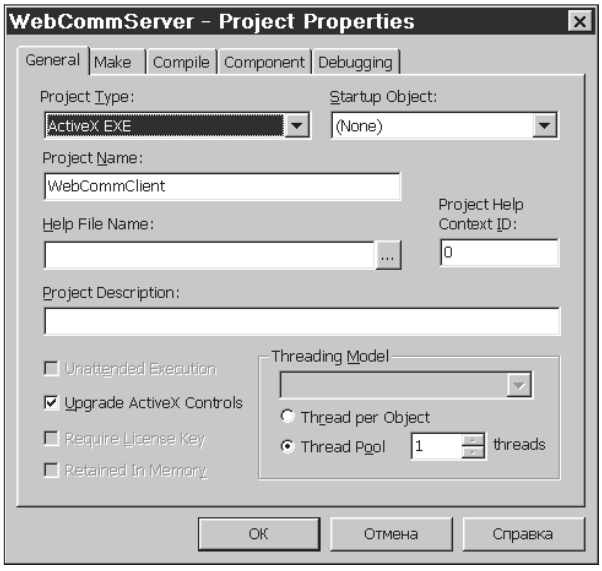
Используя документы ActiveX в своих проектах, вы сможете создавать переносимые варианты приложений и работать с ними на портативных компьютерах, в дальних офисах и даже из дома. Приложение работает прямо из Web-броузера. Тем не менее, документ ActiveX не является Web-страницей — это вполне самостоятельное приложение. Кроме того, пользователи могут свободно переключаться между документами ActiveX и Web-страницами в броузере.

Чтобы наглядно продемонстрировать, как это происходит, мы продолжим работу над приложением WebComm и создадим клиента WebComm в виде документа ActiveX (вместо стандартного EXE-файла):

1. Создайте новый проект командой **File ➤ New Project**.
2. В диалоговом окне **New Project** выберите значок **ActiveX Document EXE**.
3. Задайте свойству `Name` проекта значение `WebCommClient`.
4. Откройте диалоговое окно свойств проекта — для этого щелкните правой кнопкой мыши на имени проекта в окне проекта и выполните команду **WebCommServer properties** из контекстного меню.
5. Перейдите на вкладку **Make** и введите в поле **Title** строку `Клиент WebComm`.
6. Закройте диалоговое окно кнопкой **OK**.
7. Задайте свойству `Name` документа `UserDocument1` значение `docWebComm`.
8. Щелкните правой кнопкой мыши на панели элементов и выполните команду **Components** из контекстного меню.
9. Установите флажки рядом со строкой **Microsoft Winsock Control 6.0**. Закройте диалоговое окно кнопкой **OK** — элемент Winsock появляется на панели.
10. Добавьте элемент Winsock в `docWebComm` и задайте его свойству `Name` значение `wsk`.
11. Создайте элемент-надпись (Label) в левой верхней части документа. Задайте его свойству `Name` значение `lblName`, а свойству `Caption` — значение `Введите имя:`.
12. Создайте под надписью текстовое поле. Задайте его свойству `Name` значение `txtName`. Удалите текущее содержимое свойства `Text`.

Совместная потоковая модель

В открывшемся диалоговом окне Project Properties (см. ниже) обратите внимание на рамку Threading Model вкладки General. Работа с потоковыми моделями — одна из новых возможностей Visual Basic 6. Точнее, эта возможность была выпущена для Visual Basic 5 в составе дополнения (service pack), но в версии 6 она является встроенной. Некоторые компоненты ActiveX могут быть многопоточными — это означает, что они «хорошо ведут себя» с другими объектами и не мешают работе друг друга.



В Visual Basic разделение потоков между объектами основано на совместной потоковой модели (apartment-model threading). При таком подходе каждый поток имеет свою собственную копию глобальных данных, необходимых для работы объекта. При создании нового потока создается новая копия данных, а внесенные в нее изменения не влияют на глобальные данные других потоков. Упрощенно говоря, в результате приложения Visual Basic могут поддерживать ограниченную многозадачность.

Если вам захочется больше узнать о проблемах многопоточности, обращайтесь к справочной системе Visual Basic.

13. Создайте еще одну надпись рядом с полем `txtName`. Задайте ее свойству `Name` значение `lblTx`, а свойству `Caption` — значение `Введите сообщение:`. Свойство `Enabled` должно быть равно `False`.

14. Создайте текстовое поле под `lblTx`. Задайте его свойству `Name` значение `txtTx`, удалите текущее содержимое свойства `Text`. Кроме того, задайте свойству `Enabled` значение `False`.
15. Создайте еще одно текстовое поле, расположенное на пару линий сетки ниже `txtTx`, и задайте его свойству `Name` значение `txtRx`. Задайте свойству `Enabled` значение `False` и удалите текущее содержимое свойства `Text`.
16. Задайте свойству `MultiLine` элемента `txtRx` значение `True`, а свойству `ScrollBars` — значение `2 - Vertical`.

После создания всех элементов документ будет выглядеть так, как показано на рис. 18.8.

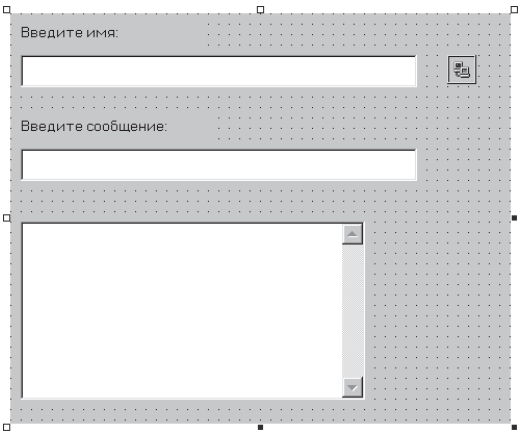


Рис. 18.8. *Клиент WebComm в режиме конструирования*

17. Сохраните проект.

Вероятно, вы уже поняли, что визуальное конструирование составляет лишь незначительную часть процесса разработки приложений Visual Basic. К счастью, программный код в данном случае не так уж сложен. Он добавляется в приложение следующим образом:

1. Дважды щелкните на документе в окне конструктора, чтобы открыть окно программы.
2. Вставьте следующий фрагмент в процедуру события `Resize` документа:

```
Private Sub UserDocument_Resize()
    ' Растянуть текстовое поле
    With txtRx
        .Move 0, .Top, ScaleWidth, (ScaleHeight - .Top)
End Sub
```

3. Вставьте следующий код в процедуру события `Terminate`:

```
Private Sub UserDocument_Terminate()
    ' Закрыть соединение
```

```
    wsk.Close
End Sub
```

Чтобы другие собеседники знали, кто участвует в разговоре, пользователь должен ввести свое имя. Мы обеспечиваем выполнение этого требования, устанавливая соединение с сервером лишь после ввода данных в поле Name. После ввода имени и нажатия клавиши Enter документ подключается к серверу.

4. Вставьте следующий фрагмент в процедуру события KeyPress поля txtName:

```
Private Sub txtName_KeyPress(KeyAscii As Integer)
    ' Переслать данные, когда пользователь нажимает Enter
    If (KeyAscii = 13) And _
        (txtName.Text <> "") Then
        ' Подключиться к серверу
        With wsk
            .Protocol = sckTCPProtocol
            .RemoteHost = "10.0.0.1"
            .RemotePort = 12345
            .Connect
        End With

        ' Сделать доступным к текстовым полям
        lblTx.Enabled = True
        txtTx.Enabled = True
        txtRx.Enabled = True

        ' Запретить возможное изменение имени
        lblName.Enabled = False
        txtName.Enabled = False
    End If
End Sub
```

Обратите внимание: текстовое поле передачи (txtTx) и окна разговора (txtRx) остаются заблокированными до тех пор, пока в поле Name не появится какой-нибудь текст.

ПРИМЕЧАНИЕ Перед тем как запускать приложение, необходимо заменить свойство RemoteHost именем хоста или IP-адресом компьютера, на котором работает сервер WebComm. В нашем примере свойство RemoteHost равно 10.0.0.1, что соответствует первому свободному адресу в тестовом пуле TCP/IP.

В большинстве чат-комнат перед текстом сообщения указывается имя участника. Это позволяет другим собеседникам следить за тем, от кого пришло то или иное

сообщение. Конечно, наша чат-комната должна следовать этому неписаному правилу. Проблема решается просто: мы берем имя, введенное в поле Name, приписываем к нему текст сообщения и отсылаем полученную строку.

5. Вставьте следующий фрагмент в процедуру события txtTx_KeyPress:

```
Private Sub txtTx_KeyPress(KeyAscii As Integer)
    Dim msg As String

    ' Переслать данные, когда пользователь нажимает Enter
    If (KeyAscii = 13) And _
        (wsk.State = sckConnected) Then

        ' Включить в сообщение имя отправителя
        msg = "[" & _
            UCase$(Trim$(txtName.Text))& _
            "] - " & txtTx.Text

        ' Отправить сообщение
        wsk.SendData msg

        ' Очистить поле с отправленным текстом
        txtTx.Text = ""
    End If
End Sub
```

Кроме того, нам понадобится функция, которая будет обрабатывать поступающие сообщения и отображать их в окне разговора.

6. Вставьте следующий фрагмент в процедуру события wsk_DataArrival:

```
Private Sub wsk_DataArrival(ByVal bytesTotal As Long)
    Dim msg As String

    ' Получить сообщение из буфера
    wsk.GetData msg, , bytesTotal

    ' Вывести сообщение в текстовом поле
    With txtRx
        .Text = msg & _
                vbCrLf & _
                vbCrLf & _
                .Text
    End With
End Sub
```

7. Сохраните проект!

Как видите, клиент устроен намного проще, чем сервер, хотя оба приложения во многом похожи. Каждое из них должно создать свою сторону сеанса и обрабатывать поступающие данные. Полученные данные обрабатываются в процедуре события `DataArrival` методом `GetData`, а отправляются — методом `SendData`.

Впрочем, оставим теорию! Наверняка вам не терпится посмотреть, как работает наша чат-комната!

Использование сервера WebComm

Сервер WebComm можно запустить на том же компьютере, на котором он был создан. Впрочем, чтобы понаблюдать за ним во всем блеске, желательно запустить его на другом компьютере, входящем в сеть. Выберите тот вариант, который лучше подойдет для ваших условий.

Запуск сервера WebComm на отдельном компьютере

Если вы хотите запустить сервер WerbComm на другом компьютере, выполните описанные ниже действия; в противном случае пропустите этот раздел и переходите к следующему:

1. Создайте программу установки с помощью мастера Package and Deployment Wizard (см. урок 14).
2. Скопируйте готовую программу установки на тот компьютер, с которого она должна устанавливаться. Кроме того, программу можно скопировать в общий каталог файлового сервера, к которому имеют доступ оба компьютера.
3. Запустите программу установки на компьютере, который должен выполнять функции сервера.
4. После успешного завершения установки запустите сервер WebComm из меню Пуск или Проводника Windows.
5. Пропустите следующий раздел.

Запуск сервера WebComm на том же компьютере

1. Запустите сервер WebComm из Проводника Windows или меню Пуск.
2. Убедитесь в том, что окно сервера WebComm открылось, а в строке состояния выводится текст `Сеансы: 0`.

Использование клиента WebComm

Перед тем как запускать клиента WebComm, проследите за тем, чтобы свойство `RemoteHost` из процедуры события `txtName_KeyPress` совпадало с именем компьютера, на котором работает сервер WebComm. Если сервер работает на том же компьютере, что и клиент, укажите в свойстве `RemoteHost` IP-адрес вашего компьютера.

1. Запустите клиента WebComm клавишей F5 из Visual Basic. Поскольку приложение представляет собой документ ActiveX, для его работы необходим контейнер. В нашем случае контейнером является Microsoft Internet Explorer. Если все работает нормально, окно клиента будет выглядеть так, как показано на рис. 18.9.

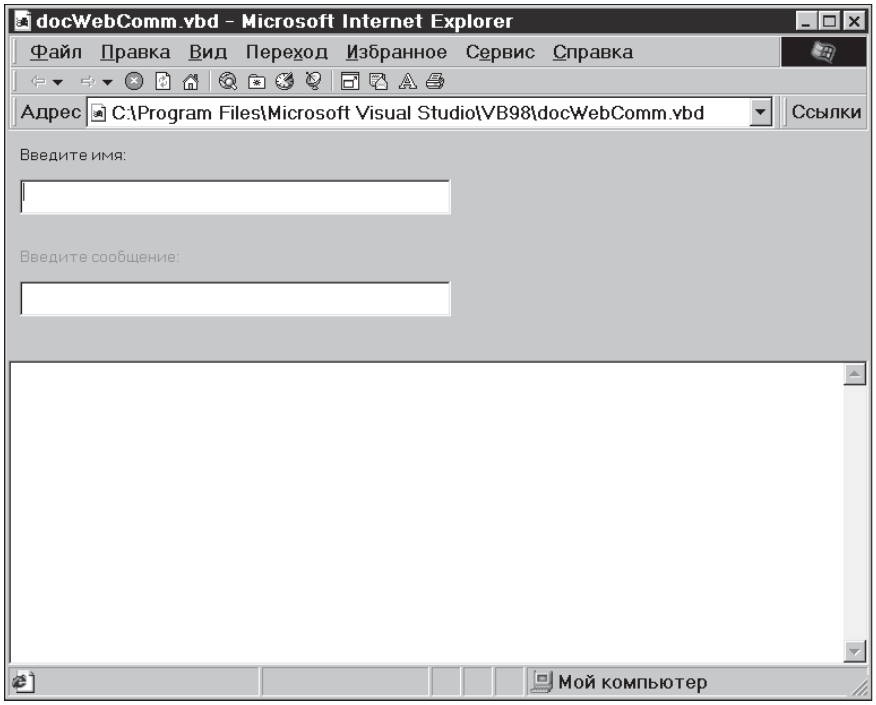


Рис. 18.9. *Клиент WebComm работает в Internet Explorer*

2. Введите свое имя в поле `Name` и нажмите клавишу Enter. Если свойство `RemoteHost` задано правильно, поле `txtTx` становится доступным, а поле `Name` — наоборот, блокируется. Подключение к серверу WebComm состоялось!
3. Введите текст `Кто-нибудь меня слышит?` в поле рядом с надписью `Введите сообщение:`.
4. Нажмите клавишу Enter, чтобы отправить сообщение. Если все работает правильно, сообщение появляется в основном поле (рис. 18.10).

Вы можете продолжить разговор и проследить за тем, как сообщения появляются в окне клиента. При желании загляните в окно сервера — вы увидите список отправленных сообщений, а если вы еще не отключились — количество сеансов, увеличившееся до 1. Когда эксперименты будут закончены, закройте окно Internet

Explorer. После этого не забудьте остановить работу проекта в Visual Basic командой Run ➤ End.

Как видите, на Visual Basic можно писать весьма нетривиальные программы, ведь коммуникации всегда считались одной из сложнейших тем программирования. Visual Basic упростил эту задачу. В приложении WebComm осталось немало возможностей для усовершенствования, например, пользователю можно предложить выбрать сервер, к которому он хочет подключиться. Вместо простых текстовых полей можно воспользоваться элементами RichText и раскрасить сообщения в разные цвета. А если вам захочется настоящих приключений, можно сделать так, чтобы приложение воспроизводило WAV-файлы на компьютерах ваших собеседников!

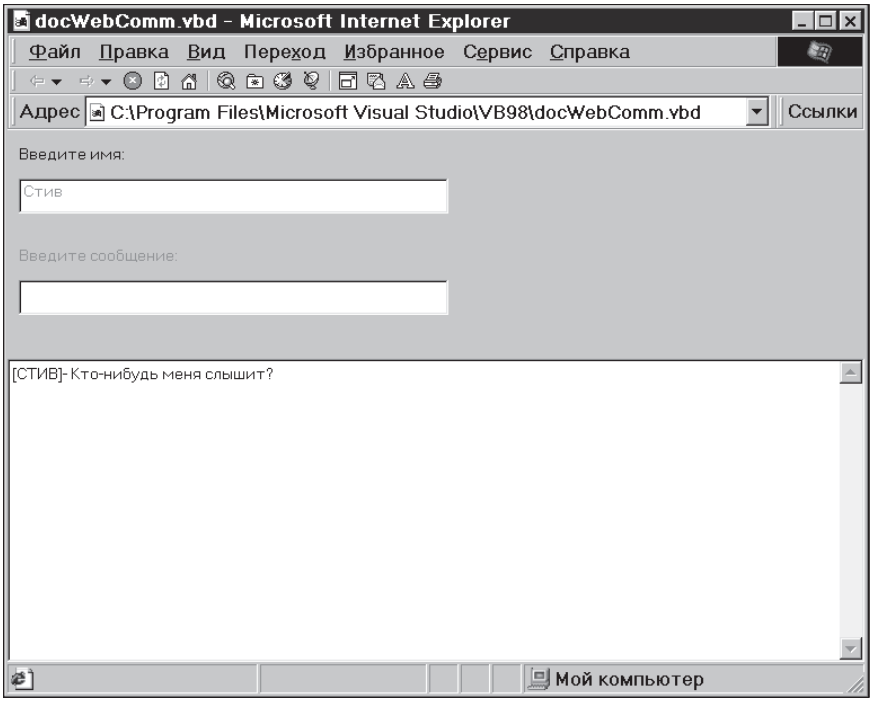


Рис. 18.10. *Разговор начался!*

ПРИМЕЧАНИЕ Обязательно посетите Web-узел издательства «Питер» (www.piterpress.ru). Программа WebComm постоянно обновляется и включает все больше замечательных возможностей. Вы не только обзаведетесь собственной чат-комнатой, но и увидите, как она реализована. Если у вас появятся интересные идеи по поводу этого приложения, поделитесь с нами!

Использование DHTML

DHTML, или Dynamic HTML (динамический язык гипертекстовой разметки), позволяет вдохнуть новую жизнь в обычные Web-страницы. Visual Basic вошел в ту область, которая всегда считалась достоянием текстовых редакторов и специализированных HTML-редакторов, — его средства визуального конструирования и IDE помогают в разработке ваших собственных DHTML-приложений.

Поскольку DHTML является самостоятельным языком, его подробное описание выходит за рамки данной книги. Подробную информацию об использовании DHTML можно найти в справочной системе Visual Basic. А мы лишь посмотрим, как Visual Basic применяется для создания базовой структуры проекта DHTML, создадим страницу с несколькими элементами и напишем код, который «оживит» наш документ.

DHTML и Visual Basic

Язык DHTML основан на модели документного объекта (Document Object Model), которая представляет собой иерархию элементов Web-страниц. Элементы в DHTML играют ту же роль, что и управляющие элементы в Visual Basic. Страница DHTML является аналогом объекта-формы в Visual Basic.

Такая парадигма проектирования позволяет без особого труда воспользоваться знанием Visual Basic для разработки интеллектуальных Web-страниц. Код Visual Basic можно использовать в событиях элементов DHTML — почти так же, как это делается в обычных проектах.

Если вы умеете создавать Web-документы на языке HTML, возможно, вам стоит продолжить работу в своем HTML-редакторе и импортировать результаты в проект DHTML. Последующая модификация элементов наделит ваше Web-приложение необходимым блеском.

Создание проекта DHTML

Недавно мы видели, как WebComm работает в Internet Explorer. Теперь давайте создадим DHTML-страницу, которая будет запускать клиентское приложение WebComm:

1. Создайте новый проект командой File ➤ New Project.
2. Выберите в диалоговом окне New Project значок DHTML Application и нажмите кнопку OK, чтобы создать проект.
3. Задайте свойству `Name` проекта значение `WebCommStart`.
4. Дважды щелкните в строке `DHTMLPage1` в окне проекта, чтобы открыть страницу в режиме конструктора. Примерный вид окна показан на рис. 18.11.

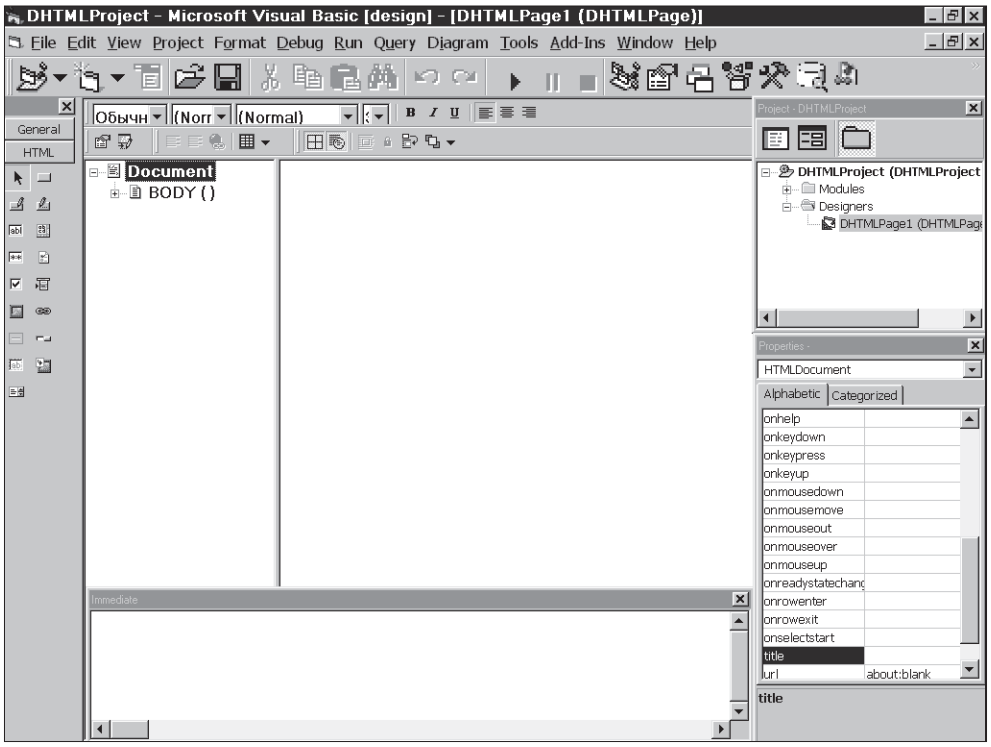


Рис. 18.11. *Конструктор DHTML*

Как видите, окно конструктора разделено на две панели. В левой панели отображается список компонентов (в DHTML они называются *элементами*). Прокручивая панель, вы сможете просмотреть свойства каждого элемента. В правой панели показано, как выглядит страница в броузере. В ней можно ввести текст или добавить более сложные элементы (например, кнопки и текстовые поля). Элементы, помещаемые на страницу, отображаются в левой панели; вы можете выделить их и изменить значения свойств.

5. Задайте свойству `ID` страницы `DHTMLPage1` значение `htmWebCommStart`.
6. Щелкните в правой панели, введите текст `Добро пожаловать в WebComm!` и нажмите клавишу Enter.
7. Выделите текст *Добро пожаловать в WebComm!* и присвойте ему шрифтовые атрибуты Arial, Bold, 6 с помощью панели инструментов, расположенной в верхней части окна конструктора. Задайте его свойству `ID` значение `pWelcome`.
8. Введите текст `Издательство` и дважды щелкните на элементе Hyperlink (гиперссылка) в панели элементов. После введенного текста создается гиперссылка.
9. Замените текст ссылки `Hyperlink1` на `Питер!`

10. Выделите текст `Издательство Питер!` и присвойте ему шрифтовые атрибуты Arial, Italic, 3. Задайте его свойству `ID` значение `pPiter`.
11. Щелкните на гиперссылке правой кнопкой мыши и выберите команду Properties из диалогового окна Property Pages.
12. Введите текст `www.piter-press.ru` в поле Link. В поле Pop-up Text введите текст `Питер – любые книги о компьютерах!` Подтвердите изменения кнопкой OK.
13. В окне свойств задайте свойству `ID` значение `lnkPiter`.
14. Дважды щелкните на элементе Button (кнопка) в панели элементов, чтобы поместить его на страницу.
15. Задайте свойству `ID` значение `cmdStart`, а свойству `Value` — значение `Начать работу`.

После добавления всех элементов страница должна выглядеть так, как показано на рис. 18.12. На левой панели отображаются свойства всех элементов. Поведение элементов, выделенных жирным шрифтом, можно запрограммировать (поскольку для них было задано свойство `ID`).

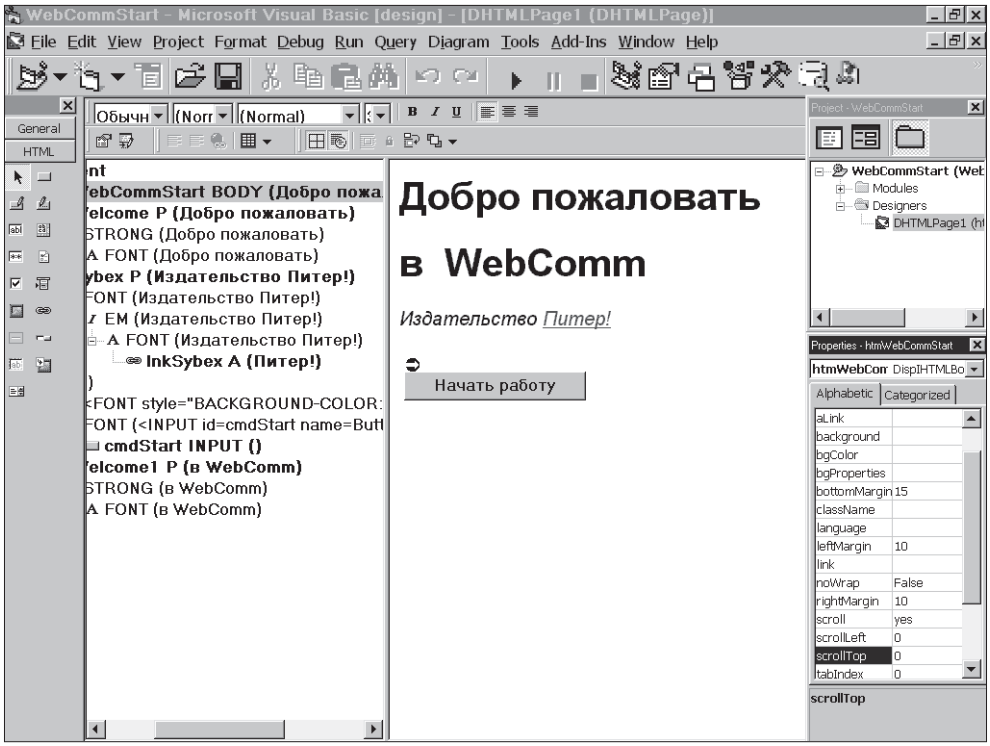


Рис. 18.12. *Страница с добавленными элементами*

16. Дважды щелкните на странице, чтобы открыть окно программы.
17. Вставьте следующую строку в секцию (General)(Declarations):

```
Private Const LNK_WEBCOMM_CLIENT = ""
```

18. Введите полный путь к файлу **WebComm.vbd**, созданному в предыдущем примере. Страница будет просматривать указанный каталог при нажатии кнопки cmdStart.

Приложения DHTML имеют событие onLoad, аналогичное событию Load для форм Visual Basic. Включая свой код в процедуру этого события, вы можете настроить страницу перед ее отображением в броузере.

19. Вставьте следующий фрагмент в процедуру события onLoad объекта Base Window:

```
Private Sub BaseWindow_onLoad()
    ' Задать свойства документа
    With Document
        .bgColor = "lightyellow"
        .linkColor = "blue"
        .vlinkColor = "blue"
        .alinkColor = "blue"
    End With

    ' Настроить приветствие
    pWelcome.Style.Color = "blue"
    pPiter.Style.Color = "black"
End Sub
```

Возможно, вы заметили, что цвета определяются в текстовом виде (вместо шестнадцатеричных констант). Дело в том, что переменные DHTML по умолчанию имеют тип Variant. Кроме того, в HTML вместо констант используются текстовые обозначения цветов.

Также следует заметить, что программа изменяет цвет свойства Style объекта. Поскольку в страницах DHTML широко используется стилевое оформление, вы можете изменить внешний вид (стиль) объекта через его свойство Style.

20. Чтобы кнопка на форме выполняла полезные действия, вставьте следующий фрагмент в функцию события cmStart_onClick:

```
Private Function cmdStart_onClick() As Boolean
    ' Запустить клиента WebComm
    BaseWindow.navigate LNK_WEBCOMM_CLIENT
End Function
```

21. Чтобы гиперссылка смотрелась более эффектно, мы будем окрашивать ее в красный цвет, когда над ней задерживается указатель мыши. Для этого вставьте в процедуру события `onmouseover()` гиперссылки следующий фрагмент:

```
Private Sub lnkPiter_onmouseover()
    ' Перекрасить ссылку в красный цвет
    lnkPiter.Style.Color = "red"
End Sub
```

22. Конечно, когда указатель мыши отходит от гиперссылки, необходимо восстановить прежний цвет. Для этого вставьте следующий фрагмент в процедуру события `onmouseout()` ссылки `lnkPiter`:

```
Private Sub lnkPiter_onmouseout()
    ' Перекрасить ссылку в синий цвет
    lnkPiter.Style.Color = "blue"
End Sub
```

23. Сохраните и запустите проект.

На экране появляется окно Internet Explorer с нашей страницей DHTML (рис. 18.13).

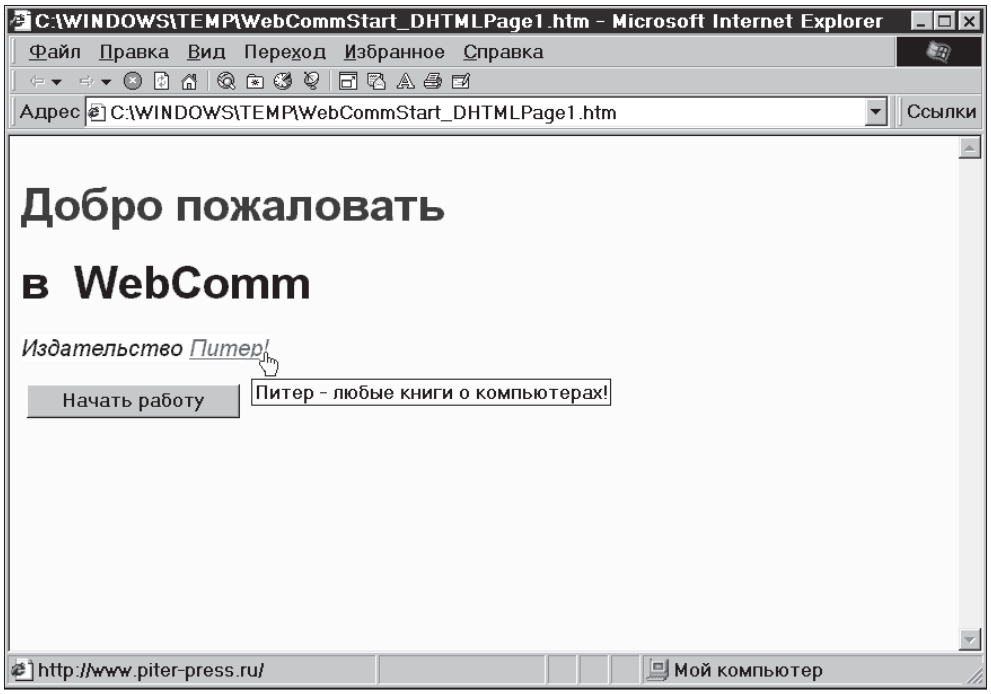


Рис. 18.13. *Страница запуска WebComm*

Если провести указатель мыши над гиперссылкой, она окрашивается в красный цвет. Если убрать его в сторону, ссылка снова становится синей. Если задержать мышь над гиперссылкой, появляется рекламная подсказка. Когда все будет готово к запуску клиентского приложения WebComm, нажмите кнопку на странице.

1. Закройте Internet Explorer и остановите приложение командой Run ➤ End в меню Visual Basic.
2. Откомпилируйте приложение командой File ➤ Make WebCommStart.dll.
3. Откройте проект WebIndex, созданный в начале урока.
4. Вставьте в список lstSites новую строку. Для этого используется команда следующего вида (указанный каталог следует заменить тем, в котором был сохранен проект WebCommStart):

```
lstSites.AddItem "d:\piter\webcomm start page\" & _
    "WebCommStart_DHTMLPage1.htm"
```

Сохраните и запустите проект, чтобы проверить, как работает наша страница.

Перед тем как активизировать клиента WebComm, следует предварительно запустить сервер WebComm. Если выбрать из списка строку WebCommStart, появляется страница DHTML в элементе WebBrowser. Она будет работать точно так же, как и во время тестирования в Internet Explorer. Если нажатием кнопки запустить клиента WebComm, он тоже будет отображаться в элементе WebBrowser!

Шаблон приложения IIS

Остается сказать несколько слов о последнем средстве разработки Internet-приложений — шаблоне приложения IIS (IIS Application). Он помогает создать базовую структуру DHTML-приложения, работающего с Microsoft Internet Information Server (IIS). В нем DHTML, Web-классы, страницы ASP (Active Server Pages) и сценарные языки (такие, как VBScript и JScript) объединяются для построения сложных серверных приложений.

К сожалению, разработка таких приложений — занятие не для слабонервных, и данная тема выходит за рамки книги. Если вам захочется заняться разработкой серверных приложений, почитайте Microsoft Developer Network или купите хорошую книгу, посвященную ISAPI, Web-классам и ASP.

Что нового мы узнали?

В этом уроке мы научились:

- ☑ Создавать приложения-броузеры с помощью элемента WebBrowser.
- ☑ Использовать элемент Winsock для разработки коммуникационных программ.

- ☑ Использовать технологию ActiveX для создания распределенных Интернет-приложений.
- ☑ Создавать приложения DHTML.