bhv®
www.bhv.ru
www.bhv.kiev.ua

Игорь Шапошников

# ASP.NET

Самоучитель

Visual Studio .NET

Разработка Web-приложений

Взаимодействие с базами данных

XML Web-сервисы

*Мощное и гибкое средство создания многофункциональных Web-проектов*

**Игорь Шапошников**

# САМОУЧИТЕЛЬ ASP.NET

Санкт-Петербург

«БХВ-Петербург»

2002

УДК 681.3.06

**Шапошников И. В.**
Самоучитель ASP.NET. — СПб.: БХВ-Петербург, 2002. — 368 с.: ил.

ISBN 5-94157-171-2

Книга представляет собой руководство по созданию мощных полнофункциональных сайтов на основе технологии ASP (Active Server Pages). В книге рассмотрены вопросы установки средств разработки ASP.NET и настройки Web-сервера IIS. Описан процесс создания ASP-приложений, начиная самыми простыми и заканчивая многофункциональными ASP-сценариями. Отдельные главы книги посвящены обзору языков Visual Basic, XML и SOAP. Читатель познакомится с основами создания интернет-сервисов на основе ASP, достаточно новым направлением с многообещающими перспективами.

*Для Web-программистов*

УДК 681.3.06

**Группа подготовки издания:**

| | |
|---|---|
| Главный редактор | *Екатерина Кондукова* |
| Зам. главного редактора | *Евгений Рыбаков* |
| Зав. редакцией | *Анна Кузьмина* |
| Редактор | *Владислав Борисов* |
| Компьютерная верстка | *Натальи Караваевой* |
| Корректор | *Татьяна Звертановская* |
| Дизайн обложки | *Игоря Цырульникова* |
| Зав. производством | *Николай Тверских* |

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 29.03.02.
Формат 70×100 $^{1}/_{16}$. Печать офсетная. Усл. печ. л. 29,67.
Тираж 3000 экз. Заказ №
"БХВ-Петербург", 198005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар, № 77.99.1.953.П.950.3.99
от 01.03.1999 г. выдано Департаментом ГСЭН Минздрава России.

Отпечатано с готовых диапозитивов
в Академической типографии "Наука" РАН
199034, Санкт-Петербург, 9 линия, 12.

ISBN 5-94157-171-2

# Содержание

*Даниленко Ольге*

*"Если б не было тебя, я б шел по миру, как слепой,*
*В гуле сотен чужих голосов узнать пытаясь голос твой*
*И звук твоих шагов"*

# Благодарности

Огромное спасибо Вам за то, что Вы держите сейчас книгу в руках и читаете ее. Все книги пишутся именно для их, читателей. На самом деле, мало есть на свете вещей стимулирующих так, как отзывы читателей о книге.

Но автор никогда не пишет книгу один. Всегда есть люди, чей вклад в книгу является не менее весомым, чем авторский. Я, конечно, говорю о редакторском коллективе издательства "БХВ-Петербург". Спасибо нашему главному редактору Кондуковой Екатерине, отдельное спасибо принимающему редактору. На самом деле, упоминать надо всю группу подготовки издания, которая в очередной раз сделала работу просто замечательно.

Спасибо также всем моим друзьям в Сети: Буке, Умке, Лайке, Sweet-dream, Панку, Финисту и многим-многим другим. Ребята, я всех вас помню и люблю.

Огромное спасибо моей Ольге, которая всегда помогала и поддерживала меня во время работы. Спасибо тебе, милая, за терпение и понимание.

# Введение

О чем же эта книга? Что такое ASP.NET? По сути дела, в книге рассказывается о создании мощных полнофункциональных сайтов. Давно прошли те времена, когда сайт был просто набором статических Web-страниц. Теперь этого мало. Сейчас сайты все чаще создают с использованием новейших технологий программирования. Большая часть страниц генерируется автоматически, с использованием информации, получаемой из каких-либо баз данных. Естественно, для создания таких страниц одного языка HTML (Hypertext Markup Language, язык описания гипертекстовых документов), который все-таки является основой Web-страниц, явно недостаточно. Даже использование технологии динамического HTML, опирающегося на сценарии, выполняющиеся на стороне пользователя, не поможет в создании подобных сайтов.

Дело в том, что работа приложений, динамически формирующих самые разнообразные Web-страницы по запросу пользователя, может происходить только на самом сервере, который поддерживает функционирование искомого Web-сайта. А если в процессе работы с запросами удаленного пользователя приходится работать с содержимым базы данных, то здесь никак не обойтись без серверных приложений, так как база данных, естественно, может функционировать только на сервере.

Для создания приложений, действующих на стороне Web-сервера, есть достаточно много приемлемых технологий. Но технология ASP (Active Server Pages), о которой и рассказывается в этой книге, стоит особняком в этом ряду. Дело в том, что практически все технологии серверных приложений реализованы в виде дополнительных модулей, подключаемых к web-серверам. А программы ASP не нуждаются в дополнительных интерпретирующих модулях. Поддержка ASP изначально встроена в Web-сервер IIS (Internet Information Services). Этот достаточно мощный сервер в последнее время автоматически включается в состав операционных систем Windows 2000 и Windows NT.

Таким образом, если вы привыкли работать с продуктами корпорации Microsoft, и перед вами стоит задача создать полнофункциональный Web-сайт, то, пожалуй, выбор будет очевидным. Следует использовать связку IIS+ASP. Тем более что ASP-приложения отлично интегрируются с базами данных от Microsoft.

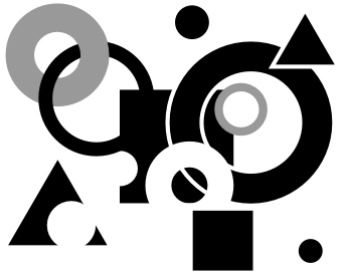
Технология ASP известна достаточно давно. В этой книге рассматривается ее последняя на данный момент модификация — ASP.NET. Разработка приложений ASP.NET производится с помощью пакета Visual Studio .NET.

Структура книги достаточно проста. В первой главе мы рассмотрим вопросы установки средств разработки программ ASP.NET и настройки web-сервера IIS. Во второй главе будет приведен обзор языка Visual Basic, на котором и разрабатываются приложения ASP. Третья глава является, пожалуй, основной в этой книге. Именно в ней будет рассказано о создании ASP-приложений. Начнем мы с самых простых вариантов и закончим созданием многофункциональных ASP-сценариев. В четвертой главе будут освещены вопросы связывания Web-приложений с системами управления базами данных. Затем будут рассмотрены языки XML и SOAP, с которыми очень тесно связана технология Microsoft .NET. А в последней главе мы рассмотрим создание интернет-сервисов на основе ASP. Это достаточно новое направление, но уже с многообещающими перспективами. Собственно говоря, сама концепция ".NET" строится на создании сервисов, которые будут доступны для работы пользователя с любой компьютерной платформой.

Разумеется, сами сервисы будут функционировать на Windows-серверах, но удаленные пользователи могут использовать их вне зависимости от того, на какой платформе они работают. Для связи клиентской системы и .NET-сервиса обычно используется язык XML с его потрясающей адаптивностью и расширяемостью. Естественно, для того чтобы строить подобные сервисы, следует знать основные правила работы с языком XML. Именно поэтому целая глава будет посвящена рассмотрению языка XML.

А теперь перейдем к работе.

# Глава 1



# Инсталляция и настройка

## Среда разработки

Создание приложений ASP.NET производится при помощи средства разработки Visual Studio .NET. Это огромнейший пакет средств и технологий для создания самых разнообразных приложений. У меня просто не хватает слов, чтобы вкратце описать все его возможности. Разработчик может использовать один из трех языков программирования, входящих в состав пакета: Visual Basic .NET, Visual C++ .NET и Visual C# .NET. Все они объединены общим интерфейсом, поэтому разработчик для реализации каждой задачи может выбирать тот язык, который максимально хорошо подойдет к данной ситуации. При этом внешний вид среды разработки, ее основные функции и механизмы не будут изменяться, и в любом случае, разработчик будет действовать в рамках знакомой среды программирования. Подобное единообразие значит очень много, так как удобство разработки является одним из ключевых факторов быстроты создания и качества программного продукта.

Дистрибутив Visual Studio .NET занимает семь дисков CD-ROM. Естественно, на жесткий диск переносится не все их содержимое, но так или иначе, придется выделить не менее 3 Гбайт на основном жестком диске и около 300 Мбайт на системном логическом диске.

Как и все системное программное обеспечение, производимое корпорацией Microsoft, среда разработки Visual Studio .NET является достаточно требовательным приложением. В качестве минимальных системных требований, помимо уже упоминавшегося необходимого объема свободного места на дисках системы, необходимо наличие процессора не ниже уровня Pentium II — 450 и оперативная память от 96 Мбайт. На практике, для более приемлемой скорости работы, следует нарастить объем оперативной памяти до 128 Мбайт. Да и более мощный процессор явно не помешает.

Также стоит обратить внимание на используемую операционную систему. Рекомендуется использовать операционные системы семейства Windows, основанные на технологии NT. Это сама Windows NT, Windows 2000 (желательно оригинальная версия) или Windows XP. Однако последняя в настоящий момент имеет не слишком долгую историю эксплуатации, поэтому, теоретически, может быть несколько нестабильна. На первые две перечисленные системы надо установить самые новые пакеты обновлений. Иначе говоря, машину, на которой будет производиться разработка, необходимо поднять на максимально высокий уровень.

Впрочем, на пятом диске дистрибутива Visual Studio .NET находятся все дополнительные компоненты, необходимые для функционирования среды разработки и среды выполнения программ. И первым действием программы установки будет проверка наличия необходимых дополнительных компонентов в составе операционной системы. Кстати, одним из таких дополнительных компонентов является набор серверных расширений FrontPage, который отказывается корректно устанавливаться на русскую версию Windows 2000. Поэтому я и упоминал, что следует использовать именно оригинальную версию. Впрочем, данный компонент не является критичным для функционирования Visual Studio .NET. И уж тем более, он не нужен для разработки ASP-приложений. Данные расширения позволяют обеспечивать работу некоторых активных элементов, которые FrontPage позволяет встраивать в разрабатываемые HTML-документы. Мы же будем сами разрабатывать подобные активные элементы, поэтому наличие специализированных расширений нам абсолютно не нужно.

Изначально скрипты ASP создавались при помощи языка Visual Basic. Однако концепция MSIL (Microsoft Intermediate Language), являющаяся частью Microsoft .NET, позволяет писать ASP-приложения на любом языке, поддерживаемом средой разработки Visual Studio .NET. В ее состав входят языки Visual Basic .NET, Visual C++ .NET и Visual C# .NET. Приложения ASP можно разрабатывать на любом из этих языков, но традиционно они разрабатывались при помощи языка Visual Basic, и мы тоже будем использовать именно этот язык.

Но вернемся к рассмотрению концепции MSIL. Для того чтобы разработчик мог выбирать язык программирования в зависимости от своих предпочтений или поставленной задачи, необходимо нечто большее, чем единая среда разработки. Необходима общая основа для всех приложений. И она была создана. Как мы помним, при установке Visual Studio .NET были инсталлированы дополнительные элементы. Одним из таких элементов была среда .NET Framework. Этот компонент можно рассматривать как некоторое дополнение к операционной системе, которое несколько расширяет ее возможности. Вместе с .NET Framework технология разработки приложений разбивается на два этапа. Сначала разработчик пишет исходный код на выбранном языке программирования. После этого компилятор Visual

Studio .NET переводит его на язык MSIL. А на втором этапе приложение, записанное на языке MSIL при помощи виртуальной машины CLR (Common Language Runtime), переводится в код, присущий той компьютерной платформе, на которой приложение будет функционировать. При этом приложение получает возможность использовать всю функциональность, описанную в Microsoft .NET Framework.

Естественно, подобная структура компиляции полностью копирует идеологию Java. Как мы помним, Java-компиляторы переводили исходный текст программы не в машинные коды, а в некий байт-код, который затем компилировался виртуальной Java-машиной. Конечно виртуальную Java-машину надо было писать для каждой специфичной платформы. Точно также обстоит дело и с Microsoft .NET Framework. Точнее, с ее основной частью — так называемым *"компилятором по требованию"*, JIT-компилятором, чаще называемым *джиттером*, который и отвечает за перевод кода MSIL в код, специфичный для конкретной платформы. На момент написания книги джиттер был разработан только для операционной системы Microsoft Windows 2000 и Windows XP.

# Администрирование IIS

Но вернемся к ASP. Данная технология позволяет создавать приложения, обеспечивающие интерактивность и иные дополнительные возможности для сайтов. ASP-приложения действуют не на стороне удаленного пользователя, а на самом WWW-сервере. Поддержка технологии ASP встроена в WWW-сервер IIS (Internet Information Services), который входит в состав дистрибутива Windows NT 4 и Windows 2000. Впрочем, на диске с дополнительными компонентами, входящими в состав дистрибутива Visual Studio.NET, также находится этот компонент, и если он не установлен в операционной системе, то инсталлятор Visual Studio.NET попробует установить IIS самостоятельно. Но все-таки стоит установить его с самого начала еще перед установкой Visual Studio.NET, чтобы правильно его настроить.

Для установки IIS следует открыть окно **Control Panel** (Панель управления) операционной системы и запустить утилиту **Add/Remove Programs** (Установка и удаление программ). Нам потребуется утилита **Add/Remove Windows Components** (Установка компонентов Windows). Эта утилита активизирует диалоговое окно **Windows Components Wizard** (Мастер установки компонентов Windows), показанное на рис. 1.1.

Компонент Internet Information Services (IIS) состоит из нескольких разнородных частей. Список их можно увидеть, если выделить в диалоговом окне **Windows Component Wizard** (Мастер установки компонентов Windows) строку **Internet Information Services (IIS)** и нажать кнопку **Details** (Состав). В результате будет отображено диалоговое окно **Internet Information Services (IIS)**, показанное на рис. 1.2.

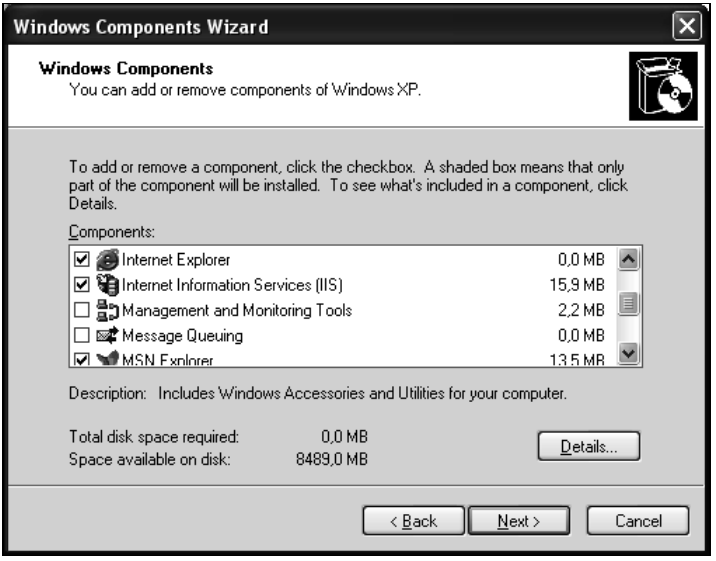



**Рис. 1.1.** Диалоговое окно **Windows Components Wizard**

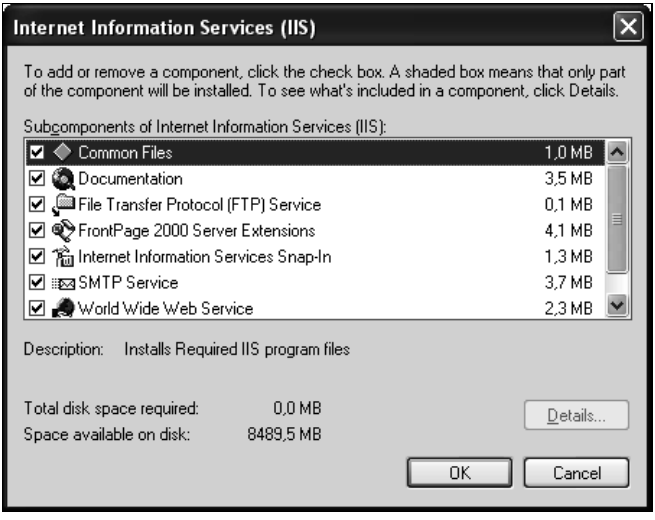



**Рис. 1.2.** Диалоговое окно **Internet Information Services (IIS)**

В диалоговом окне **Internet Information Services (IIS)** мы можем указать, какие именно функциональные части элемента IIS требуется установить в операционную систему. Для создания полноценного сервера нам потребуются следующие компоненты:

- **Common Files** (Общие файлы). Общие файлы, необходимые для функционирования практически всех элементов IIS;

- **File Transfer Protocol (FTP) Server** (Служба FTP). Сервер приема и передачи файлов по протоколу FTP;
- **Internet Information Services Snap-In** (Оснастка IIS). Система администрирования сервера при помощи утилиты MMC (Microsoft Management Console);
- **Personal Web Manager** (Управление личным Web-сервером). Графический интерфейс администрирования сервера;
- **SMTP service** (Служба SMTP). Служба отправки исходящих сообщений электронной почты по протоколу SMTP;
- **World Wide Web Server** (Служба WWW). Собственно, сам WWW-сервер, который и поддерживает функционирование сайтов.

Для того чтобы установить все эти элементы, необходимо поставить флажки в независимых переключателях, находящихся рядом с наименованиями компонентов, и нажать кнопку **OK**. После этого компонент IIS будет автоматически установлен. Использование остальных элементов не является критичным для создания полноценных сайтов с использованием ASP-приложений, поэтому их установка необязательна.

В процессе установки IIS будет создана структура каталогов, в которых размещается содержимое серверов. По умолчанию, корневой каталог элемента носит наименование `Inetpub`. В каталоге `FTPROOT` размещается содержимое FTP-сервера, каталог `MAILROOT` предназначен для почтового SMTP-сервера, а в каталоге `WWWROOT` находится содержимое WWW-сервера.

Нас будет интересовать администрирование именно WWW-сервера. При установке, автоматически создается некий локальный сайт, в терминологии Microsoft часто называемый Web-узлом. Доступ к нему можно получить, если в строке **Адрес** браузера набрать адрес **http://localhost**.

Необходимо сделать некоторое техническое отступление. В подавляющем большинстве, Web-сайты обладают доменным именем, как, например, **www.bhv.ru**. В том случае, если сайт не имеет подобного имени, доступ к нему осуществляется по IP-адресу того сервера, на котором он установлен. Однако, использование IP-адресов вместо доменных имен явно невыгодно, так как запомнить IP-адрес достаточно трудно, чего нельзя сказать о правильно подобранном доменном имени. Сопоставление доменных имен и IP-адресов серверов, на которых размещены сайты с этими именами производится DNS-серверами при помощи DNS-таблиц. Служба DNS (Domain Name Service) как раз и существует для того, чтобы пользователь WWW мог набрать обычное символьное доменное имя сайта, вместо IP-адреса сервера.

IP-адрес, как известно, представляет собой комбинацию из четырех положительных целых чисел, находящихся в промежутке от 0 до 255, и разделенных точками. При этом все адреса, начинающиеся с числа 127

(обозначается, как 127.*.*.*), являются локальными. Эти адреса нельзя назначить серверу, реально находящемуся в Интернете. Они могут быть назначены только локальным системам. Проще говоря, если необходимо на локальной машине разработать сервер, используя при этом некое доменное имя, например, **www.mysite.com**, то надо сопоставить это имя одному из таких локальных IP-адресов. Казалось бы, для этого необходимо запускать на локальной машине еще и DNS-сервер, чтобы браузер мог в поисках сервера, на котором содержится искомый сайт, обратиться к этому серверу без выхода в Интернет, но на самом деле все проще. Все Windows-системы перед тем, как обратиться к DNS-серверу, проверяют наличие файла hosts, находящегося в системном каталоге Windows. Это текстовый файл, в котором указывается список доменных имен и соответствующих им сайтов. Поэтому, если мы хотим, чтобы на нашем сервере действовал локальный сайт с доменным именем **www.mysite.com**, необходимо в этот файл добавить следующую строку:

```
127.0.0.1 www.mysite.com
```

Теперь, для того чтобы связаться с этим сайтом, браузер не будет обращаться к DNS-серверам, а всего лишь воспользуется файлом hosts. Впрочем, справедливости ради надо сказать, что сайт с доменным именем `localhost` система всегда ищет в корневом каталоге функционирующего WWW-сервера даже без обращения к файлу hosts.

Мы уже говорили, что основной сайт размещается в каталоге Inetpub/WWWROOT. Но в этом каталоге находятся не только HTML-документы и скрипты. Любой сайт, это еще и структура дополнительных каталогов. Всегда удобнее структурировать данные по различным каталогам. Отдельно держать графические файлы, отдельно информацию для различных разделов, и так далее. Поэтому URL графического файла с наименованием picture1.gif может выглядеть следующим образом:

**http://www.mysite.com/image/picture1.gif**

При этом подразумевается, что графический файл picture1.gif находится в каталоге images, который в свою очередь размещен в корневом каталоге WWW-сервера. На самом деле, в URL не обязательно указывается точное наименование каталога. В нем указывается наименование так называемого "виртуального каталога", чье наименование не обязательно будет совпадать с именем настоящего физического каталога. Так, каталог, в котором находится графический файл picture1.gif, мог бы на самом деле называться img, а не image, как указано в URL. То есть, виртуальные каталоги это всего лишь названия реальных, физических каталогов.

Процедура создания виртуальных каталогов достаточно проста. В группе органов управления, располагающихся в **Control Panel** (Панель управления), существует инструмент настройки основных сервисов системы —

**Administrative Tools** (Администрирование). Нас будет интересовать один из его компонентов — **Internet Services Manager** (Управление службами Интернета). Двойной щелчок по иконке этого компонента отображает диалоговое окно **Internet Information Services** (Информационные службы Интернета), предназначенное для настройки IIS, внешний вид которого показан на рис. 1.3.

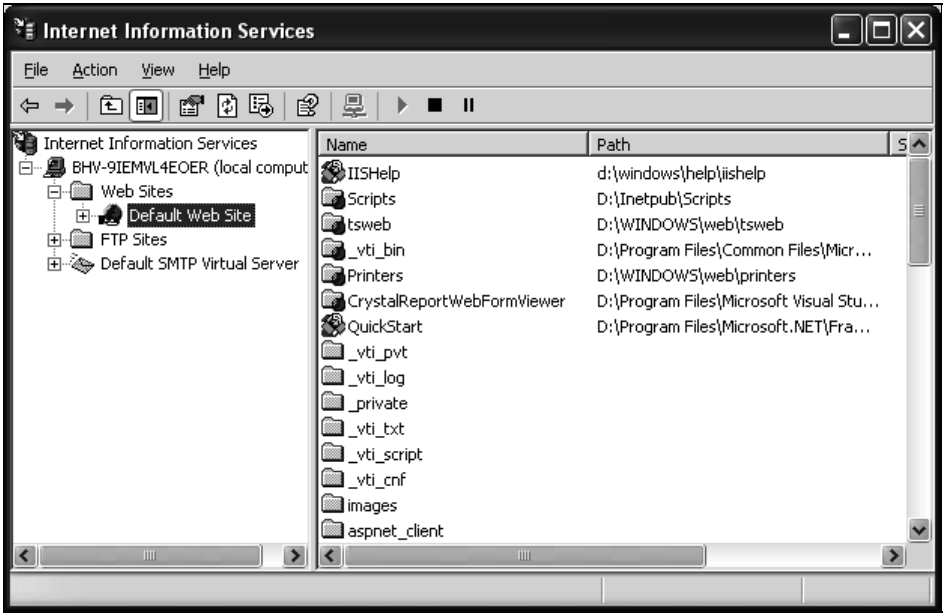



**Рис. 1.3.** Диалоговое окно настройки **Internet Information Services**

Для того чтобы создать новый виртуальный каталог, необходимо выделить строку с наименованием действующего Web-сайта, чаще всего она носит наименование **Default Web Site** (Веб-сайт по умолчанию), и нажать кнопку **Action** (Действие). В появившемся меню нам следует выбрать пункт **New | Virtual Directory** (Создать | Виртуальный каталог). После этого будет запущен мастер создания виртуальных каталогов. На первом этапе работы мастера будет отображено диалоговое окно **Virtual Directory Creation Wizard** (Мастер создания виртуальных каталогов) с полем текстового ввода, куда необходимо будет внести наименование виртуального каталога, как это показано на рис. 1.4.

После того, как мы ввели в поле **Alias** (Псевдоним) наименование виртуального каталога, следует нажать кнопку **Next** (Далее). Будет отображено диалоговое окно второго шага мастера установки виртуальных каталогов, показанное на рис. 1.5.

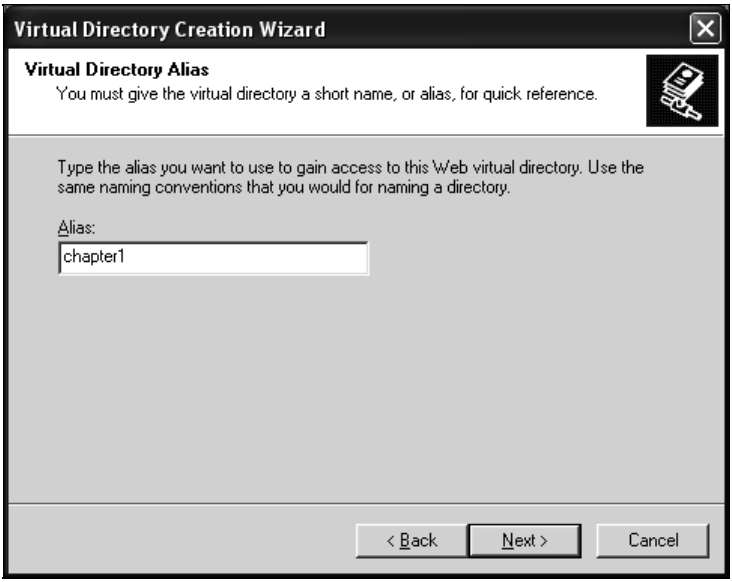



**Рис. 1.4.** Диалоговое окно настройки **Virtual Directory Creation Wizard**, отображаемое на первом этапе работы мастера

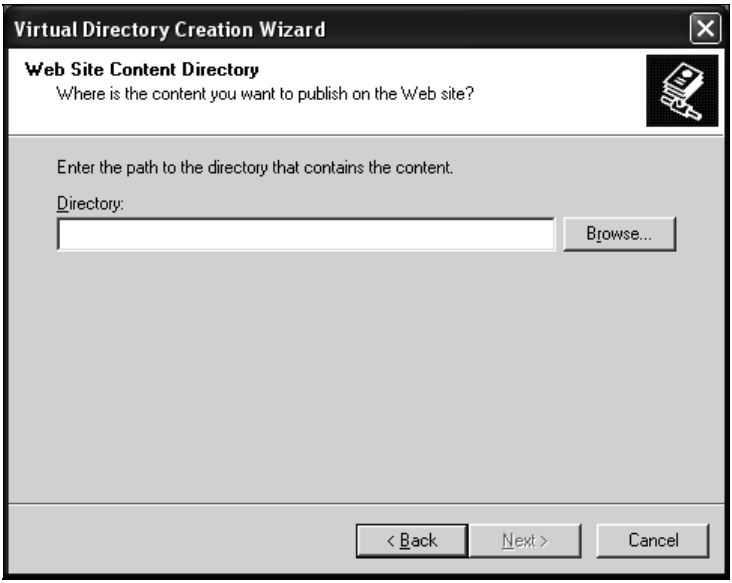



**Рис. 1.5.** Диалоговое окно настройки **Virtual Directory Creation Wizard**, отображаемое на втором этапе работы мастера

Полный путь к физическому каталогу, который будет связан с создаваемым виртуальным каталогом, следует внести в поле ввода **Directory** (Каталог),

или воспользоваться кнопкой **Browse** (Обзор). После указания пути к физическому каталогу еще раз нажать кнопку **Next** (Далее). Третий этап работы мастера позволяет установить назначение виртуального каталога. Как известно, мы можем гибко устанавливать правила работы с виртуальными каталогами. Можно разрешать чтение файлов из них, запись в файлы данного каталога, исполнение скриптов или приложений, просмотр содержимого каталога. Также возможно использование любой комбинации из этих разрешений. Итак, после того, как мы, указав путь к физическому каталогу, нажали кнопку **Next** (Далее), будет отображено диалоговое окно третьего этапа работы мастера создания виртуальных каталогов, показанное на рис. 1.6.


**Рис. 1.6.** Диалоговое окно настройки **Virtual Directory Creation Wizard**, отображаемое на третьем этапе работы мастера

Рассмотрим возможные установки разрешений для виртуального каталога. Установка или отключение того или иного разрешения производится при помощи независимых переключателей, связанных с описаниями разрешений. Всего предусмотрено пять разрешений.

- **Read** (Чтение). Разрешение на чтение файлов из создаваемого виртуального каталога.
- **Run scripts (such as ASP)** (Запуск сценариев (например ASP)). Позволяет удаленному пользователю запускать на выполнение скрипты из данного виртуального каталога.
- **Execute (such as ISAPI applications or CGI)** (Выполнение (например приложений ISAPI и CGI)). Разрешение выполнять Web-приложения, осно-

ванные на технологии ISAPI или CGI, находящиеся в данном виртуальном каталоге.

- **Write** (Запись). Разрешение на запись в файлы, размещенные в данном виртуальном каталоге.
- **Browse** (Обзор). Означает, что удаленный пользователь может просматривать содержимое данного каталога.

Так как подключение или отключение любого разрешения производится при помощи независимых переключателей, мы можем использовать любую комбинацию из предлагаемых вариантов. Однако не следует без особой нужды подключать все разрешения к создаваемым виртуальным каталогам, так как это будет являться нарушением элементарных правил безопасности WWW-сервера. Следует использовать для каждого каталога минимально необходимый набор разрешений. То есть, если каталог предназначается только для хранения графических изображений, применяемых в оформлении Web-страниц, входящих в состав сайта, нам потребуется установить лишь разрешение на чтение. Все остальные разрешения нам в данном случае не нужны.

Но основная настройка WWW-сервера производится при помощи его диалогового окна настройки свойств. Для того чтобы получить возможность работать с параметрами сервера, необходимо в диалоговом окне **Internet Information Services** (Информационные службы Интернета) выделить строку, обозначающую Web-сервер, и щелкнуть правой кнопкой мыши. В появившемся контекстном меню следует выбрать команду **Properties** (Свойства). Результатом выполнения этой команды меню будет отображение диалогового окна **Default Web Site Properties** (Свойства веб-узла по умолчанию) с открытой вкладкой **Web Site** (веб-узел), как это показано на рис. 1.7.

Это диалоговое окно позволяет управлять практически всеми свойствами WWW-сервера. Но для наших целей все они не понадобятся. Разберемся лишь с основными параметрами. Начнем мы с вкладки **Web Site** (Веб-узел). Органы управления, собранные в группе **Web Site Identification** (Идентификация веб-узла) идентифицируют сайт, который мы разрабатываем. Общее наименование сайта (но не его доменное имя) указывается в поле **Description** (Описание). В текстовых полях **IP Address** (IP-адрес) и **TCP Port** (TCP-порт) указываются IP-адрес и порт, на которых будет функционировать WWW-сервер. В поле ввода **Connection Timeout** (Время ожидания) мы указываем время тайм-аута, в течение которого сервер может ожидать ответа от удаленного пользователя на посланные запросы. Если по истечении этого времени браузер удаленного пользователя не ответил, WWW-сервер будет считать, что удаленный пользователь сбросил соединение, и разорвет его со своей стороны.

В том случае, если необходимо вести log-файлы, в которых будут записываться сообщения обо всех соединениях удаленных пользователей с WWW-сервером, следует поставить флажок в переключателе **Enable Logging** (Вести

журнал). Правильная установка службы log-файлов может сделать очень многое. С ее помощью можно узнавать IP-адреса, пользователей, их путь по сайту, страницы, с которых они пришли на сайт, время и дату их запросов к серверу и многое другое. Все зависит от того, как настроить log-файлы. Для тонкой настройки службы ведения log-файлов следует нажать кнопку **Properties** (Свойства). При этом будет показано диалоговое окно **Extended Logging Properties** (Раширенные свойства ведения журнала).

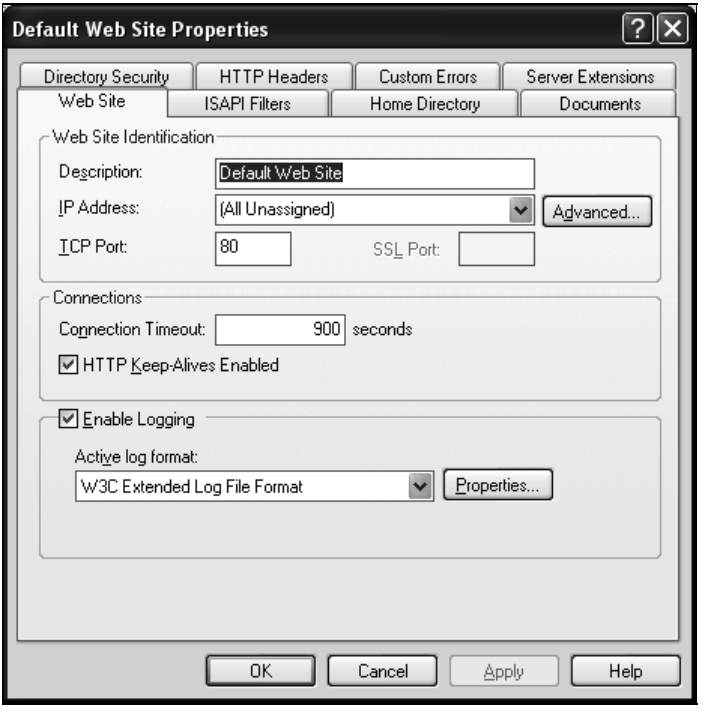



**Рис. 1.7.** Вкладка **Web Site** диалогового окна **Default Web Site Properties**

Это диалоговое окно состоит из двух вкладок — **General Properties** (Общие свойства) и **Extended Properties** (Расширенные свойства). Вкладка **General Properties** (Общие свойства) содержит органы управления, регулирующие свойства самих файлов, в которых будут содержаться сообщения о подключениях. А формат сообщений о подключениях удаленных пользователей и их действиях регулируется на вкладке **Extended Properties** (Расширенные свойства).

Когда пользователь в качестве URL указывает только доменное имя сайта, WWW-сервер отсылает браузеру стартовую страницу. Обычно HTML-файл с данной страницей носит наименование **index.htm**. Но по умолчанию WWW-сервер IIS отображает страницу **default.htm**. Впрочем, наименование

файла, который будет посылаться удаленному пользователю, естественно, можно изменить. Для этого необходимо активировать вкладку **Documents** (Документы) диалогового окна **Default Web Site Properties** (Свойства веб-узла по умолчанию), показанную на рис. 1.8.

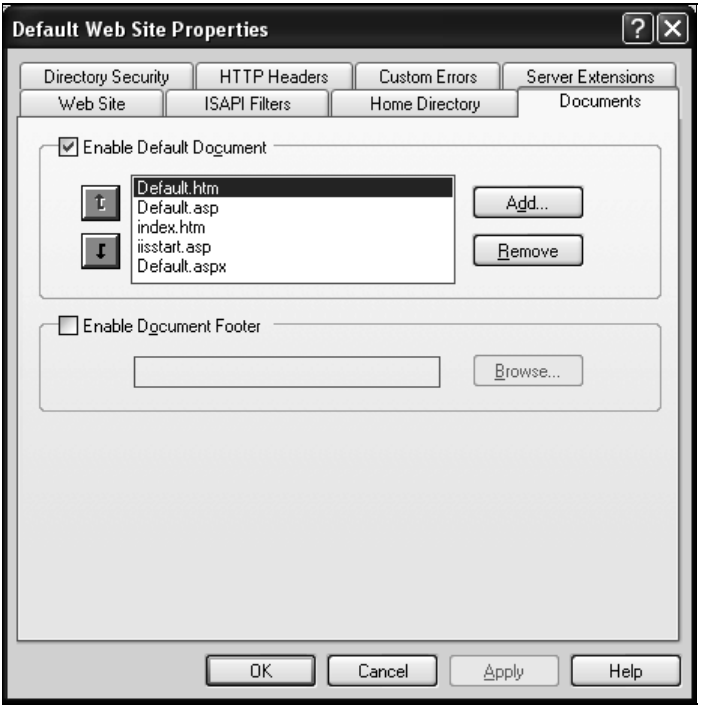



**Рис. 1.8.** Вкладка **Documents** диалогового окна **Default Web Site Properties**

Для того чтобы при указании удаленным пользователем только доменного имени сайта WWW-сервер выдавал браузеру какую-либо Web-страницу, необходимо поставить флажок в независимом переключателе **Enable Default Document** (Задать документ, используемый по умолчанию). При этом будет активизирован список, входящий в одноименную группу органов управления. В списке находятся наименования файлов, которые будет пытаться отыскать в корневом каталоге сайта WWW-сервер, чтобы обработать их и передать браузеру удаленного пользователя. При этом учитывается порядок расположения наименований файлов в данном списке. Сначала сервер будет пытаться найти файл, находящийся на первой позиции списка. Если этот файл не будет найден в корневом каталоге сайта, то сервер попытается отыскать следующий файл из списка, и так далее. На иллюстрации видно, что мы можем использовать не только чистые HTML-файлы, но и ASP-сценарии. Для изменения порядка следования наименований файлов в списке, слева от списка размещены две кнопки с изображением стрелок,

направленных вверх и вниз. Если необходимо добавить в список наименование еще одного файла, можно использовать кнопку **Add** (Добавить). А для удаления файла из списка необходимо воспользоваться кнопкой **Remove** (Удалить).

В качестве последнего штриха настройки сервера можно использовать страницы обработки ошибок. Как известно, WWW-сервер может обрабатывать ошибки, вызванные действиями удаленного пользователя или локальных сценариев и Web-приложений. Одной из наиболее распространенных ошибок является ошибка с кодом 404, когда пользователь запрашивает документ, которого в действительности нет на сервере. При возникновении какой-либо ошибки, сервер возвращает браузеру удаленного пользователя Web-страницу с описанием возникшей ситуации. Так вот, мы можем самостоятельно создавать подобные Web-страницы с сообщениями об ошибках. Для того чтобы сопоставить созданные Web-страницы с кодами возможных ошибок, следует воспользоваться вкладкой **Custom Errors** (Специальные ошибки) диалогового окна **Default Web Site Properties** (Свойства веб-узла по умолчанию), показанной на рис. 1.9.

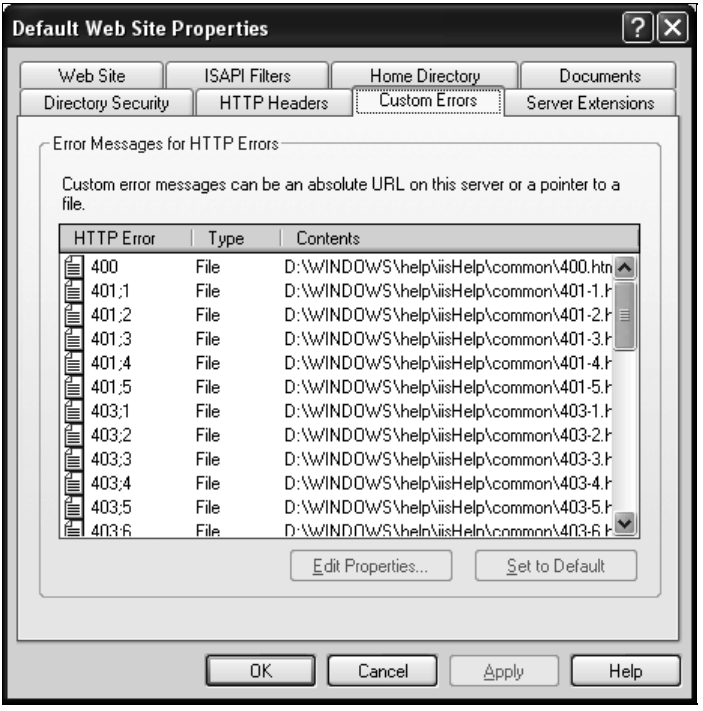


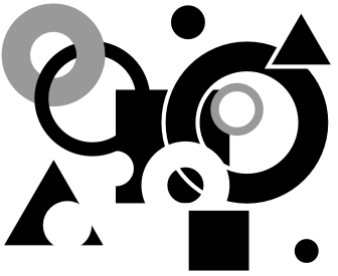

**Рис. 1.9.** Вкладка **Custom Errors** диалогового окна **Default Web Site Properties**

Основу рассматриваемой нами вкладки диалогового окна настройки свойств сервера составляет список, строки которого состоят из кода ошибки и имени

файла, который будет выводиться сервером при возникновении этой ошибки. Если мы хотим изменить стандартные предупреждения сервера, у нас есть два пути. Мы можем либо самостоятельно изменить те Web-страницы, которые указаны в этом списке, либо создать свои собственные Web-страницы с описанием возникшей критической ситуации и привязать их к кодам ошибок. Для привязки вновь созданных Web-страниц к ошибкам, следует выделить строку с кодом интересующей нас ошибки и нажать кнопку **Edit Properties** (Изменить свойства). После этого будет отображено дополнительное диалоговое окно с полем ввода для полного наименования HTML-файла, который будет посылаться в браузер удаленного пользователя при возникновении искомой ошибки.

Итак, мы настроили свой WWW-сервер. Теперь для того чтобы начать с ним работать в локальном режиме, необходимо его запустить. Обычно запуск WWW-сервера происходит автоматически при загрузке операционной системы. Проверить его функционирование можно в диалоговом окне контроля IIS. Контекстное меню, появляющееся при щелчке правой кнопкой мыши на строке **Default Web Site** (Веб-узел по умолчанию) в диалоговом окне **Internet information Services** (Информационные службы Интернет), содержит три пункта — **Start** (Пуск), **Stop** (Остановить) и **Pause** (Приостановить), которые позволяют принудительно запустить, отключить и приостановить работу сервера соответственно. Нет нужды напоминать, что ASP-сценарии обрабатываются сервером, поэтому при их тестировании необходимо, чтобы сервер был запущен.

# Глава 2



# Основы языка Visual Basic

## Управляющая логика

Почти все программы имеют одинаковую структуру. В хорошей программе явно выделены области объявления переменных, функций и процедур, везде присутствуют одни и те же циклы и управляющие логические конструкции, всегда можно найти операторы сравнений. Разница лишь в ключевых словах, которые используются в том или ином языке программирования. В данной главе мы рассмотрим основные конструкции языка Visual Basic.

Почему мы рассматриваем именно Visual Basic? Дело в том, что именно на этом языке изначально писались сценарии для ASP, поэтому стоит рассматривать именно этот язык в качестве рабочей лошадки для ASP-сценариев. Если быть более точным, надо заметить, что при разработке сценариев ASP.NET мы используем язык Visual Basic .NET, который несколько отличается от канонического Visual Basic. Однако эти отличия не так уж и кардинальны, и там, где они будут важны для создания сценариев ASP.NET, мы упомянем об этих изменениях.

Естественно, мы не будем рассматривать язык Visual Basic .NET полностью. Дело в том, что для нужд создания сценариев ASP.NET нужны далеко не все его возможности. Но все, что действительно необходимо для ASP, мы рассмотрим.

Основа любой программы — это ее управляющая логика, т. е. те самые циклы, операторы сравнений и прочие конструкции, обуславливающие порядок действий программы.

**Примечание**

Яростные приверженцы объектно-ориентированной модели программирования могут заметить, что основой программы является именно объектная модель.

Но ради объективности надо признать, что это не так. Программы писались еще в те времена, когда об объектах никто и не задумывался. Все-таки управляющая логика это основа, альфа и омега любой программы.

Начнем мы с основного оператора управляющей логики — условного оператора. В языке Visual Basic он имеет предельно детализированный вид и объявляется следующим образом:

```
If условие [ Then ]
   [ блок кода  ]
[ ElseIf условие [ Then ]
   [ блок кода ] ]
[ Else
   [ блок кода ] ]
End If
```

Детально рассмотрим это формальное объявление оператора. Обязательным ключевым словом является, естественно, слово `If`. После него мы указываем либо логическое условие (или их комбинацию), либо переменную булева (логического) типа. После ключевого слова `Then` размещается блок кода, который будет выполнен в том случае, если логическое условие истинно. Или булева переменная имеет значение `True`. Блок, начинающийся с ключевого слова `ElseIf`, позволяет задать еще одно логическое условие, которое будет проверяться на истинность в том случае, когда первое условие, находящееся после ключевого слова `If`, является ложным. Как и в первом блоке оператора, код, выполняемый в случае истинности условия, размещается после ключевого слова `Then`. Если же оба логических условия оказываются ложными, мы можем использовать третий раздел оператора, начинающийся с ключевого слова `Else`. Сразу после него размещается блок кода, выполняемый в том случае, если все логические условия оказались ложными. Завершается весь условный оператор комбинацией ключевых слов `End If`. Необходимо также отметить, что второй и третий блоки оператора не являются обязательными.

Приведем пример использования условного оператора. Рассмотрим следующую конструкцию:

```
If i>5  Then
   i=i*2
 ElseIf i<0 Then
   i=i*(-1)
[ Else
   i=1
End If
```

Все операции в этом блоке кода производятся с переменной `i`. Если значение этой переменной больше пяти, то оно удваивается. Если оно меньше

нуля, то у этого значения принудительно меняется знак. Во всех остальных случаях, т. е. когда значение переменной находится в промежутке от нуля до пяти, это значение становится равным единице.

Но, как мы уже говорили, последние два блока оператора являются необязательными, поэтому без них можно обойтись, если необходимо просто проверить некоторое условие. В этом случае условный оператор будет выглядеть следующим образом:

```
If i>5 Then
   i=i*2
End If
```

Легко увидеть, что реализация условного оператора в Visual Basic .NET явно несколько утяжелена по сравнению с остальными распространенными языками программирования. Как мы увидим позже, подобная избыточность свойственна практически всем управляющим конструкциям Visual Basic .NET.

Логическим продолжением условного оператора является оператор выбора `Select`. Его формальное объявление выглядит следующим образом:

```
Select [ Case ] testexpression
   [ Case expressionlist
      [ блок кода ] ]
   [ Case Else
      [ блок кода ] ]
End Select
```

В теле этого оператора при помощи ключевого слова `Case` мы можем задавать логические условия и сопоставлять им блоки кода. Причем количество ключевых слов `Case` и связанных с ними логических условий ограничивается только требованиями программиста. Более того, в операторе присутствует ключевое слово `Else`. Блок кода, связанный с ним, выполняется в том случае, если ни одно из вышеперечисленных логических условий не было истинным.

Условия, связанные с ключевым словом `Case`, задаются несколько шире, нежели в случае с обычным условным оператором. Можно задавать целый интервал допустимых значений при помощи ключевого слова `To`. Пример такого условия показан в следующем фрагменте кода:

```
Select i
  Case 1 To 10 . . .
  . . . .
End Select
```

Оператор `Select` связывается с переменной `i`. А в первом условии проверяется, входит ли значение этой переменной в промежуток между единицей и десятью.

В логических условиях оператора `Select` мы можем использовать ключевое слово `Is`, применяемое в связке с операторами логического сравнения. Так, например, для того, чтобы выполнить некий блок кода в том случае, если значение основной переменной оператора больше пяти, следует использовать следующий код:

```
Select i
  Case Is>5 . . .
  . . . .
End Select
```

Завершается оператор, как видно из объявления, комбинацией ключевых слов `End Select`.

Приведем пример использования оператора `Select`. Рассмотрим следующий фрагмент кода:

```
Select i
  Case 1 To 5
     s="От единицы до пяти"
  Case 6, 7
     s="Или шесть или семь"
  Case Is>8
     s="Больше восьми"
  Case Else
     s="Судя по всему, меньше единицы"
End Select
```

Рассмотрим работу этого оператора. Все логические условия завязаны на переменную `i`. Первое условие проверяет вхождение значения основной переменной в интервал от единицы до пяти. В том случае, если это условие является истинным, то переменной `s`, очевидно имеющей строковый тип, присваивается значение `От единицы до пяти`. Второе условие проверяет, не равно ли значение основной переменной оператора шести или семи. Легко заметить, что в случае перечисления сравниваемых значений, они просто разделяются запятой. Соответственно, если это условие выполняется, то переменной `s` присваивается другое значение. Третье и последнее логическое условие будет истинным, если значение основной переменной оператора больше восьми. В том случае, если ни одно из перечисленных условий не выполняется, управление переходит к блоку кода, указанному после комбинации ключевых слов `Case Select`. Впрочем, этот блок не является обязательным.

Настало время рассмотрения циклов. Стандартный цикл `For`, связанный с переменной-счетчиком, объявляется следующим образом:

```
For counter = start To end [ Step step ]
   [ блок кода ]
```

```
[ Exit For ]
   [ блок кода ]
Next [ counter ]
```

Процесс выполнения такого цикла связан не с каким-либо логическим условием, а с переменной-счетчиком, на основе которой вычисляется количество проходов цикла. Простейший пример подобного цикла выглядит следующим образом:

```
For i=0 To 9
 my_array(i)=i*2
Next
```

В этом цикле мы каждому элементу массива присваиваем значение, в два раза больше его порядкового номера. Следует отметить, что нумерация массива начата с нуля, что нетипично для Visual Basic. Это нововведение Visual Basic .NET, которое мы рассмотрим в разделе, посвященном созданию массивов и работе с ними.

По умолчанию, после каждого прохода цикла значение переменной-счетчика увеличивается на единицу. Цикл будет выполняться до тех пор, пока значение переменной-счетчика не превысит числа, указанного после ключевого слова `To`. Однако могут возникать случаи, когда необходимо при прохождении цикла изменять значение счетчика не на единицу. В этом случае следует использовать необязательное ключевое слово `Step`, после которого указывается величина изменения счетчика с каждым проходом цикла, т. е. если мы хотим модифицировать цикл из вышеприведенного примера так, чтобы операции присваивания производились только с четными элементами массива, следует использовать следующий фрагмент кода:

```
For i=0 To 9 Step 2
  my_array(i)=i*2
Next
```

После ключевого слова `Next` мы можем указать наименование переменной-счетчика цикла. Это указание переменной не является обязательным, но может помочь в создании более понятного кода, когда применяются встроенные циклы. Посмотрите на такой пример:

```
For i=0 To 9
 For j=0 To 9
  my_array(i,j)=i*2+j
 Next j
Next i
```

В данном случае, дополнительное указание наименований счетчиков в конце цикла помогает читать код.

На основе приведенных примеров может сложиться впечатление, что переменная-счетчик для данного цикла может обладать только целочисленным

типом. На самом деле это не так. Для создания переменных-счетчиков может применяться любой численный тип, для которого поддерживаются арифметические операции сложения и вычитания, а также логические операции "больше" и "меньше".

Существует и еще одна вариация цикла `For`. Она применяется в тех случаях, когда необходимо обработать все элементы некоей группы. Это могут быть элементы коллекций, списков, различных стеков и прочих групп. Пример подобного цикла выглядит следующим образом:

```
For Each myitem In mylist
 myitem.qwant=1
Next
```

Как видно, структура подобного цикла достаточно прозрачна. Начало цикла формируется при помощи комбинации ключевых слов `For Each`, после которых указывается наименование элементов, по которым будет проходить цикл. Затем следует ключевое слово `In`, после которого мы указываем наименование списка, в который входят обрабатываемые элементы. В нашем случае мы производим обработку всех элементов `myitem`, которые являются вложенными частями объекта `mylist`. Естественно, в данном случае нам не нужна переменная-счетчик, для контроля над количеством проходов цикла. Легко заметить, что в приведенном примере мы успешно обошлись без нее. Завершается цикл при помощи знакомого нам уже ключевого слова `Next`.

Также часто применяются циклы, основанные на каких-либо логических условиях. То есть тело цикла будет выполняться до тех пор, пока заранее оговоренное условие не станет истинным или ложным (в зависимости от того, какую разновидность цикла использует разработчик). Подобные циклы объявляются следующим образом:

```
Do { While | Until } условие
   [ блок кода ]
[ Exit Do ]
   [ блок кода ]
Loop
```

Это объявление так называемого цикла с предусловием. То есть, перед выполнением очередного прогона цикла проверяется логическое условие, и в зависимости от результата проверки действие цикла может быть выполнено или нет. Ключевое слово `Until` применяется в тех случаях, когда цикл должен выполняться до тех пор, пока логическое условие является истинным. Примером подобного цикла является следующий фрагмент кода:

```
i=0
Do While i<10
 my_array(i)=i*2
```

```
 i+=1
Loop
```

Следует заметить, что в данном конкретном случае мы все же использовали переменную-счетчик, пусть и опосредованную. Но все зависит от логики программы. Часто подобные циклы применяются и без счетчиков.

Легко увидеть, что в приведенном примере цикл будет выполняться до тех пор, пока условие не станет ложным. Если же мы будем использовать ключевое слово `Until` вместо `While`, то цикл будет выполняться до тех пор, пока условие не станет истинным.

Мы уже говорили, что подобная модификация циклов называется циклом с предусловием, т. е. соответствие логического условия проверяется перед выполнением тела цикла. Таким образом, может возникнуть ситуация, когда еще перед самым первым проходом цикла окажется, что логическое условие не выполняется. В этом случае цикл не будет выполнен. Однако не стоит забывать о циклах с постусловием. В этой разновидности циклов логическое условие проверяется уже после выполнения тела цикла. Другими словами, при любом раскладе можно быть уверенным, что цикл будет пройден хотя бы один раз. Определяется подобный цикл следующим образом:

```
Do
   [ блок кода ]
[ Exit Do ]
   [ блок кода ]
Loop { While | Until } условие
```

Как видно, отличие от первого варианта цикла очень невелико — условие продолжения работы вместе с ключевым словом `Until` или `While` перенесено в конец цикла.

Итак, мы рассмотрели практически все управляющие конструкции языка Visual Basic .NET. Без внимания остались некоторые элементы управляющей логики, которые автор не счел нужным приводить. Одним из таких пропущенных элементов является ключевое слово `GoTo`, обеспечивающее прямую передачу управления какой-либо строке программы. Понятно, что этот пережиток прошлого должен быть как можно быстрее забыт, так как эта разновидность управления ходом выполнения программы устарела даже в момент появления парадигмы модульного программирования. Сейчас это просто дополнительный источник ошибок, которых и без того немало в наших программах, средство ухудшения читабельности кода и просто признак плохого стиля программирования.

# Типы данных

Какими бы сложными объектами, массивами или коллекциями мы ни пользовались, следует помнить, что основой их являются единичные переменные.