Крейг Атли

VB .NET для программистов

Разработка Internet-приложений

VB.NET

SAMS

ДМК ПРЕСС

для программистов

УДК 004.438 VB.NET
ББК 32.973.26-018.1
А92

Атли К.
А92 Visual Basic. NET для программистов: Пер. с англ. – М.: ДМК Пресс. - 304 с.: ил. (Серия «Для программистов»).

ISBN 5-94074-110-X

Книга посвящена языку Visual Basic. NET. Особое внимание уделяется отличиям новой версии от предыдущих, в том числе революционным для этого языка нововведениям: наследованию реализации, многопоточности, пространствам имен, новой интегрированной среде разработки Visual Studio. NET. Освещаются вопросы, связанные с применением новой технологии доступа к данным ADO. NET и технологии создания Web-приложений ASP. NET. Также не обойдены вниманием новые виды проектов, которые можно создавать на языке Visual Basic: NT-сервисы, Web-сервисы и консольные приложения. Подробно рассказывается о процессе конвертирования старых программ на новый язык. По ходу изложения автор раскрывает возможности принципиально нового каркаса для разработки приложений .NET Framework и подчеркивает его неразрывную связь с программированием на VB.NET. Специальная глава посвящена совместной работе технологий .NET и COM.

Издание предназначено для программистов, работавших с предыдущими версиями языка Visual Basic и желающих познакомиться с особенностями новой платформы .NET.

ISBN 0-672-32264-1 (англ.)

ISBN 5-94074-110-X (рус.)

Содержание

Глава 2. Первое приложение на VB.NET

Настало время перейти от теории к практике. Для начала познакомимся с некоторыми особенностями новой интегрированной среды разработки (IDE). На первый взгляд она не сильно отличается от той, к которой вы привыкли. Однако внесенные в нее изменения могут разочаровать опытных программистов на VB, поскольку стали другими «горячие» клавиши, поменялись названия окон и по-другому стали работать средства отладки. VB.NET стал частью интегрированной среды Visual Studio.NET, которая консолидирует системы разработки для всех языков: VB.NET, C++.NET и C#. В ней даже можно создать одно решение, содержащее несколько проектов, написанных на разных языках.

Страница Start Page

Впервые открывая Visual Studio.NET, вы видите экран, позволяющий сконфигурировать IDE. Это страница **My Profile** (Мой профиль). После первого посещения все остальные обращения к Visual Studio.NET начинаются со страницы **Start Page**, показанной на рис. 2.1. Она состоит из нескольких разделов, о чем свидетельствуют ссылки, расположенные вдоль левого края. Перечислим эти разделы:

- **Get Started** (Начнем) – позволяет повторно открыть недавно использовавшийся или уже существующий проект, а также создать новый проект. Имена нескольких недавно открывавшихся проектов представлены на рис. 2.1. При создании проектов в VB.NET на этой странице показывается четыре последних варианта. Кроме того, в ней есть ссылки для открытия существующего и создания нового проекта, а также сохранения отчета об ошибке. Последняя ссылка, наверное, исчезнет в окончательной версии продукта;
- **What's new** (Что нового) – здесь описываются новые возможности Visual Studio.NET, касающиеся как отдельных языков, так и интегрированной среды в целом. Информация представлена в виде ссылок на файлы справки по языкам, поддерживаемым VS.NET, и по .NET SDK. Имеется даже ссылка, позволяющая проверить появление обновлений VS.NET;
- **Online Community** (Сетевое Сообщество) – здесь находятся ссылки на сетевые конференции Microsoft. Их можно открыть с помощью любой программы

чтения новостей, но обслуживаются они специальным сервером Microsoft (msnews.microsoft.com), а не обычным сервером Usenet;

- **Headlines** (Главные новости) – здесь размещаются ссылки, по которым можно получить новую информацию о платформе .NET. Среди них – ссылка на сайт MSDN Online, на раздел технических статей, раздел базы знаний (Knowledge Base) и ряд других ресурсов;
- **Search Online** (Поиск в сети) – поиск в библиотеке MSDN Online;
- **Downloads** (Загрузить) – доступ к загрузке различных продуктов. Это могут быть новые инструменты, плановые обновления программ (service packs), инструментарий для мобильного Internet (Mobile Internet Toolkit), примеры кода и документация;
- **Web Hosting** (Web-хостинг) – обеспечение связи с провайдерами, которые предоставляют хостинг для размещения Web-сервисов и Web-приложений, созданных для платформы .NET. Одна-единственная команда позволяет вам опубликовать свои приложения или сервисы на серверах этих компаний. По крайней мере, одна из них – Brinkster – предоставляет бесплатный хостинг, так что вы сможете протестировать свое приложение;
- **My Profile** (Мой профиль) – настройка Visual Studio.NET. Например, можно установить ту же раскладку клавиатуры, какая была в предыдущей версии Visual Studio для VB6, так же расположить окна или автоматически фильтровать оперативную справку. Далее будем считать, что установлен профиль разработчика (с именем Visual Studio Developer) и сохранены настройки по умолчанию.

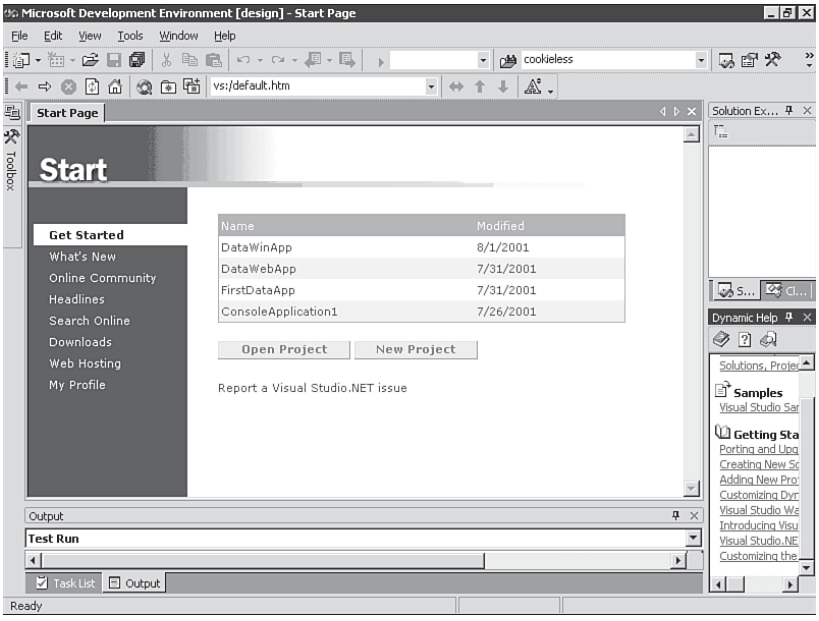

*Рис. 2.1. Начальная страница **Start Page** в Visual Studio.NET*

Примечание *Start Page – это HTML-страница, которая к моменту выхода окончательной версии VS.NET, вероятно, будет выглядеть по-другому.*

Система справки теперь более тесно интегрирована с Visual Studio. Чтобы убедиться в этом, щелкните по ссылке **What's New** на начальной странице. В разделе **What's New** есть три вкладки. Щелкните по вкладке **Product Information** (Информация о продукте), а затем по ссылке **What's New in Visual Basic** (Что нового в Visual Basic). Обратите внимание, текст справки загрузился в то же окно, где была начальная страница (рис. 2.2). Позже вы увидите, что справка может автоматически меняться по ходу работы в соответствии с тем, что вы в данный момент делаете.

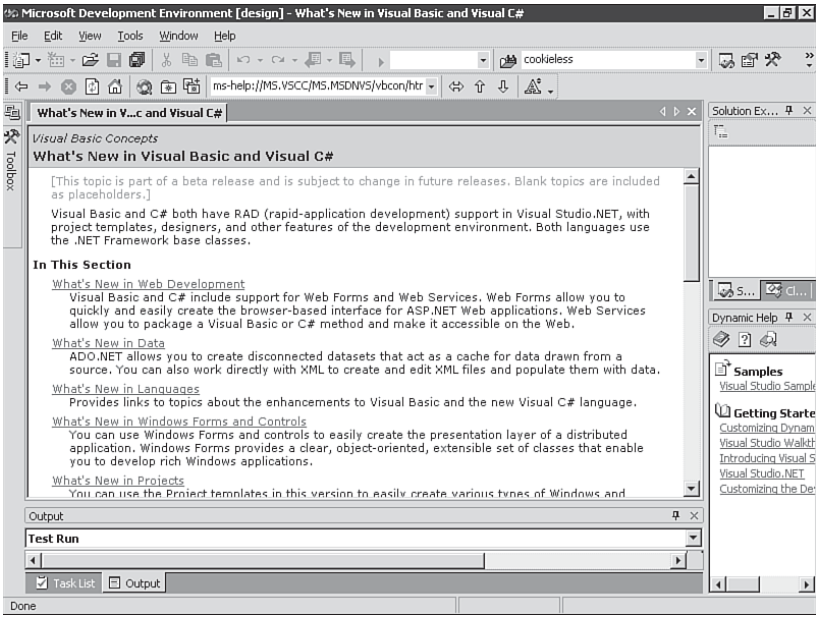

Рис. 2.2. Текст справки отражается в интегрированной среде, а не в отдельном окне

Создание нового проекта

Вернитесь на начальную страницу, щелкнув по иконке **Back** или **Home** на панели инструментов. Затем щелкните по ссылке **New Project** (Новый проект). В результате откроется диалоговое окно **New Project**, показанное на рис. 2.3. Обратите внимание, что система предлагает несколько языков для создания приложений в Visual Studio.NET. В этой книге мы будем говорить только о проектах на языке Visual Basic.

Многие проекты на Visual Basic отличаются от тех, что были в VB6. Перечислим основные виды проектов:

- **Windows Application** (Приложение Windows). В терминологии VB6 – *стандартный исполняемый файл* (standard executable). Этот вид проекта предназначен для создания приложений со стандартным интерфейсом Windows, состоящим из форм и элементов управления;
- **Class Library** (Библиотека классов). Такой проект позволяет создавать классы, которые будут использоваться в приложениях. Можете считать его аналогом проекта для создания COM-компонентов в VB6, где он назывался ActiveX DLL, или ActiveX EXE;

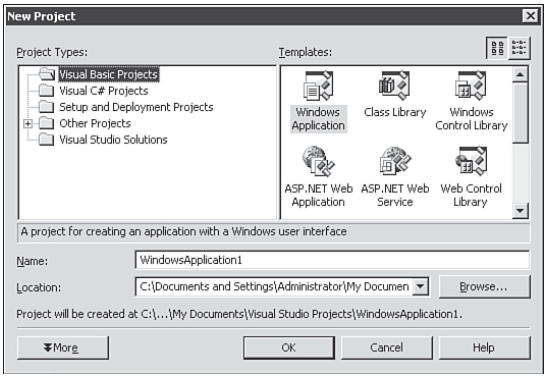

*Рис. 2.3. Диалоговое окно **New Project***

- **Windows Control Library** (Библиотека элементов управления). Такие проекты предназначены для создания аналога ActiveX Controls. Это новые элементы управления, которые можно будет использовать в приложениях для Windows;
- **ASP.NET Web Application** (Web-приложение для ASP.NET). Нет необходимости в использовании Visual InterDev и интерпретируемых сервером языков сценариев для ASP. Теперь в Visual Basic.NET есть проект для создания динамических Web-приложений, работающих под управлением ASP.NET. Такой проект может включать HTML-страницы, ASP.NET-страницы и код на языке VB.NET. Отныне Web-приложения будут строиться на основе событийно-управляемой модели, а не модели запрос—ответ. На самом-то деле парадигма запрос—ответ, конечно, никуда не делась, но программа пишется так же, как для Windows-приложений, то есть представляет собой набор обработчиков событий;
- **ASP.NET Web Service** (Web-сервис для ASP.NET). Если вам приходилось писать COM-компоненты на VB6, а затем предоставлять к ним доступ по протоколу SOAP, то с идеей Web-сервиса вы уже знакомы. Проект вида Web Service позволяет создавать компоненты, доступные другим приложениям через Web. Но запросы и ответы передаются по протоколу HTTP, а не DCOM, и представлены в формате XML. Одно из преимуществ Web-сервисов – следование общепринятым стандартам и платформенная независимость. В от-

личие от DCOM, который был тесно связан с технологией COM и, следовательно, с инфраструктурой Windows, Web-сервис можно разместить на любой платформе, поддерживающей .NET, и вызывать из другого приложения, пользуясь только протоколом HTTP;

- **Web Control Library** (Библиотека элементов управления Web). У этого вида проектов, равно как и у предыдущего, нет прямых аналогов в VB6. В проектах Web Service элементы управления можно размещать на Web-страницах точно так же, как на стандартных приложениях Windows, но .NET во время исполнения преобразует их в HTML-код. Вы можете создавать свои собственные элементы управления для использования в Web-приложениях;
- **Console Application** (Консольное приложение). Многие административные утилиты Windows все еще являются консольными приложениями (или командными, или DOS). Раньше у вас не было удобного способа создания таких программ в VB и приходилось прибегать к C++. Теперь VB.NET полностью поддерживает консольные приложения;
- **Windows Service** (Сервис Windows). В предыдущих версиях VB точно так же нельзя было создать сервис Windows. Так называются программы, которые работают в фоновом режиме и автоматически запускаются при загрузке операционной системы, даже если нет ни одного активного пользователя. У сервисов Windows обычно не бывает пользовательского интерфейса, а информацию они выводят в файл протокола. Поскольку сервис, как правило, должен функционировать постоянно, нужно уделять особое внимание обработке ошибок.

Мы перечислили основные виды приложений, которые можно писать на VB.NET. Можно также создать пустой проект (Empty Project) для приложений Windows, библиотек классов и сервисов или пустой Web-проект (Empty Web Project) – для Web-приложений. И наконец, при выборе варианта **New Project in Existing Folder** (Новый проект в существующей папке) будет создан новый пустой проект в папке, где уже есть один или несколько проектов.

Интегрированная среда разработки

Если вы еще не закрыли диалоговое окно **New Project**, выберите проект Windows Application. Назовите его Learning VB и щелкните по кнопке **OK**. Через некоторое время появится новый проект. Обратите внимание, что в главном окне при этом появляется вкладка **Form1.vb [Design]**, а в ней – пустая форма. Вы видите перед собой так называемый Дизайнер форм (Form Designer). На самом деле есть несколько видов дизайнеров, которые можно загрузить в рабочую область главного окна. Пока что для программиста на VB нет ничего нового.

Но кое-что все же произошло, хотя, скорее всего, осталось незамеченным: созданные файлы уже сохранены на диске. В VB можно было создать проект, что-то быстро написать и выйти без сохранения, не оставляя на диске никаких следов. В VB.NET файлы сохраняются в момент создания проекта, так что от каждого проекта что-то остается, даже если вы ничего не сохраняли. Пользователи Visual

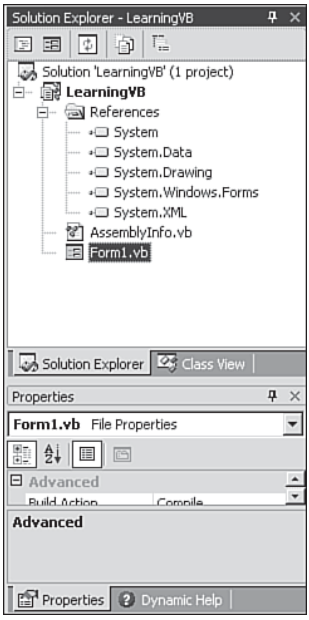

*Рис. 2.4. Окно **Solution Explorer***

InterDev к этому привыкли, но для программистов на VB такое поведение может оказаться неожиданным.

В правой части IDE вы видите окно **Solution Explorer** (Обозреватель решения). Оно, как и окно **Project Explorer** в VB6, содержит проекты и файлы, входящие в состав текущего *решения* (в VB6 оно называлось *группой*). Сейчас в этом окне (рис. 2.4) показано имя решения, имя проекта, а равно все формы и модули. На данный момент есть только одна форма с именем **Form1.vb**. Помимо нее в окне представлен файл AssemblyInfo.vb, содержащий метаданные, которые будут включены в состав сборки. Кроме того, в списке есть еще узел **References** (Ссылки). Если раскрыть его, вы увидите все ссылки, включенные в проект еще до начала программирования. Но пока ссылки вам не понадобятся.

В нижней части окна Solution Explorer находится еще одна вкладка – **Class View** (Классы). Если щелкнуть по ней, появится перечень классов, входящих в состав проекта LearningVB. Раскрыв узел проекта, вы увидите список пространств имен. На данный момент в списке есть только одно пространство, и по умолчанию его имя совпадает с именем проекта. Разверните узел пространства имен

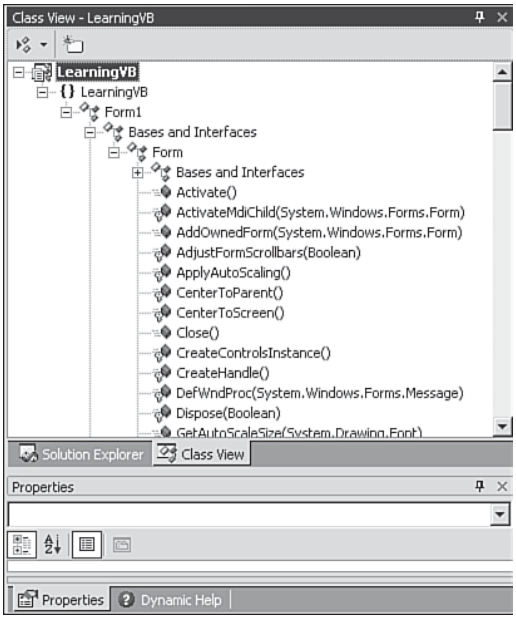

*Рис. 2.5. Новое окно **Class View***

LearningVB, и вы увидите под ним только форму `Form1`. Раскрыв узел **Form1**, вы обнаружите некоторые методы формы, а также узел **Bases and Interfaces** (Базовые классы и интерфейсы). Развернув его, а также узел **Form** ниже, вы найдете длинный список свойств, методов и событий формы. Малая часть этого перечня показана на рис. 2.5. Пока он вам не потребуется. Просто отметьте, что все это сильно отличается от того, к чему вы привыкли в VB6.

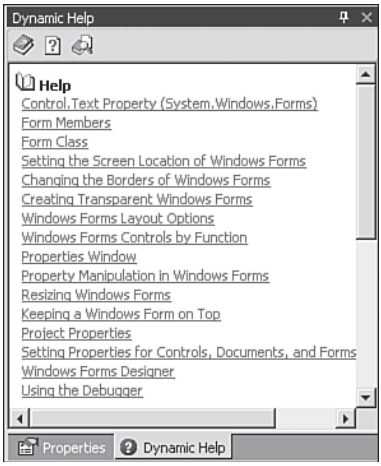

*Рис. 2.6. Окно **Object Browser** стало вкладкой в рабочей области главного окна*

Узнать о назначении свойств и методов вам поможет Object Browser (Инспектор объектов). Прокрутите список в узле **Bases and Interfaces**, пока не появится событие `Load`. Щелкните по нему правой клавишей мыши и выберите из меню пункт **Browse Definition** (Показать определение). Инспектор объектов откроет в рабочей области главного окна вкладку, на которой приведено определение события `Load`. Вы видите, что оно возвращает объект `System.EventHandler`. На рис. 2.6 показано, как это выглядит в IDE. Поскольку **Object Browser** – это всего лишь вкладка, закрывать ее необязательно. Если вы все же хотите это сделать, щелкните по иконке **X** в правом верхнем углу окна Инспектора, но не закрывайте окно Visual Studio.NET.

Ниже окна **Solution Explorer/Class View** расположено уже знакомое окно свойств **Properties**. Если вы закроете окно Инспектора объектов и вернетесь к вкладке **Form1.vb [Design]**, то увидите свойства формы `Form1`. Надо только щелкнуть по форме, чтобы передать ей фокус. Большинство свойств вам наверняка знакомо, но есть и кое-что новое. Для чего нужны новые свойства, вы узнаете из следующей главы.

Примечание *В окне Properties свойства по умолчанию отсортированы по категории, а не по алфавиту. Поэтому найти нужное свойство бывает трудно. На панели инструментов окна Properties есть иконки, позволяющие менять порядок сортировки.*

Рядом с окном свойств есть вкладка **Dynamic Help** (Динамическая справка). Это новшество Visual Studio.NET, позволяющее автоматически обновлять выводимую справочную информацию по ходу работу. Система следит за вашими действиями и предлагает список тем справки, относящихся к текущей работе. Например, открыв и сделав активным окно формы, щелкните по вкладке **Dynamic Help**. Вы получите перечень тем, касающихся Дизайнера форм: как поместить в форму элементы управления и т.п. На рис. 2.7 показано, как выглядит такой список. Попробуйте пощелкать по разным окнам IDE и понаблюдайте, как меняется содержимое окна динамической справки. Но не забывайте, что для работы динамической

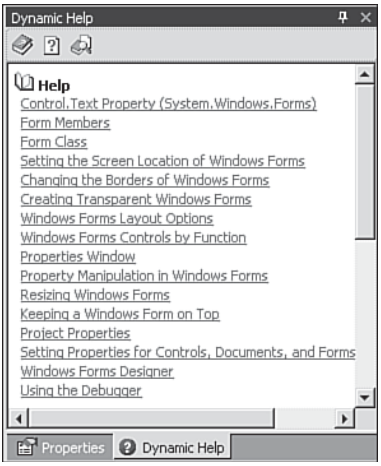

Рис. 2.7. Окно динамической справки

справки нужны ресурсы. На мощных машинах проблемы не возникает, но если на вашем компьютере медленный процессор или мало памяти, вы почувствуете замедление работы.

Вдоль левого края IDE есть две вертикально расположенных вкладки. Сейчас вы видите только одну метку. Это потому, что одна вкладка представлена меткой, а другая – кнопкой. Первая вкладка/кнопка относится к окну **Server Explorer** (Обозреватель серверов), а вторая – к набору инструментов (Toolbox). Чтобы появилось нужное окно, задержите курсор над вкладкой/кнопкой или сразу щелкните по ней.

Обозреватель серверов – это новая возможность IDE. Он позволяет обнаруживать сервисы на различных серверах. Например, если вы хотите найти машины, на которых работает Microsoft SQL Server, то в дереве для каждого сервера имеется узел SQL Server Databases. На рис. 2.8 показано, что в обозревателе серверов зарегистрирован один сервер: culaptop. На этом сервере работает SQL Server, поэтому вы видите список баз данных. В окне Server Explorer вы можете выполнять действия, которые раньше выполнялись в окне **Data Window**: просматривать данные в таблицах; удалять или создавать таблицы; удалять, создавать и редактировать хранимые процедуры и т.д. Если вы работали в Visual InterDev, то эти инструменты вам хорошо знакомы.

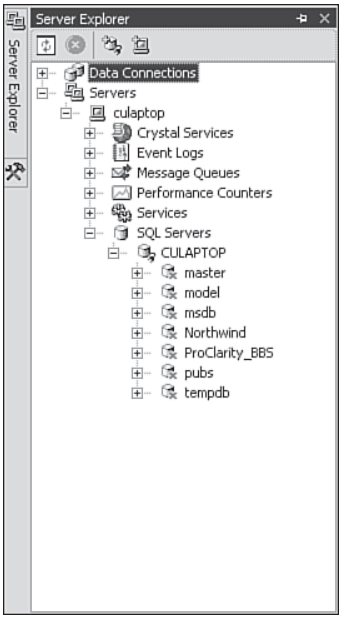

*Рис. 2.8. Новое окно **Server Explorer***

Обозреватель серверов можно использовать также для обнаружения очередей сообщений и соединения с ними, мониторинга работы удаленной машины с помощью показателей производительности и нахождения Web-сервисов на других машинах. Это удобный инструмент, обеспечивающий централизованный доступ к множеству административных функций.

Вторая вкладка – **Toolbox** – представляет собой набор инструментов, именно отсюда берутся элементы управления, которые вы хотите поместить в форму. Однако новая вкладка отличается от того, что было в VB6, и скорее похожа на набор инструментов в Visual InterDev. Окно **Toolbox** представляет собой набор расположенных друг под другом вкладок. На рис. 2.9 раскрыта вкладка **Windows Forms**, но

кроме нее есть еще вкладки **Data** (Данные), **Components** (Компоненты), **Clipboard Ring** (Связка буферов обмена) и **General** (Общие). Пусть их количество вас не смущает.

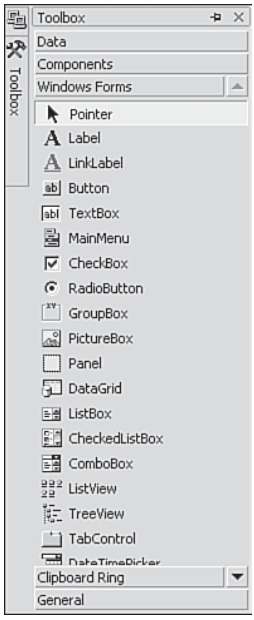

*Рис. 2.9. Вкладка **Toolbox***

Создание первого приложения на VB.NET

Итак, у вас есть открытый проект с одной формой. Давайте напишем классическую программу Hello, World. Не так давно Microsoft с восторгом демонстрировала всему миру, как написать такое приложение в VB с помощью всего одной строчки кода. Сейчас это так же легко, хотя выглядит все, конечно, по-другому.

Сделайте окно Дизайнера форм текущим и откройте набор инструментов. Щелкните по инструменту «кнопка» и перетащите ее в любое место формы. Пока все идет точно так же, как в VB. А теперь дважды щелкните по кнопке.

При этом открывается окно кода, как и в VB. Однако в VB.NET окно кода стало вкладкой в рабочей области главного окна. Новая вкладка называется **Form1.vb**, и рядом с заголовком находится звездочка, показывающая, что код еще не сохранен. В этом окне содержится код, который вы раньше не встречали. Часть кода скрыта, поскольку Редактор кода в VS.NET может сворачивать и раскрывать блоки текста. Если раскрыть секцию кода, сгенерированную Дизайнером форм, появится много разной информации. Таким образом, еще не написав ни одной строчки самостоятельно, вы получили код, показанный на рис. 2.10. Я включил нумерацию строк, чтобы было проще ссылаться. Если вы хотите сделать то же самое, выберите из меню **Tools** (Инструменты) пункт **Options** (Настройки). Раскройте узел **Text Editor** (Редактор текста) и выберите **Basic**. Установите флажок **Line Numbers** (Нумерация строк).

Первая строка кода показывает, что `Form1` в действительности является классом. Это одно из наиболее существенных изменений в VB.NET: формы являются настоящими классами. Почему? Потому что они есть и всегда были классами, хотя, возможно, вы не рассматривали их с этой точки зрения. При отображении формы вы создавали экземпляр класса.

Следующая строка – это предложение `Inherits`. Любая создаваемая вами форма Windows наследует базовому классу `Form`. Вы обращаетесь к классу в одном из пространств имен .NET – в данном случае это класс `Form` в пространстве имен `System.Windows.Forms`. Именно этот класс предоставляет в ваше распоряжение всю функциональность форм, то есть методы и события, которыми вы пользуетесь в своей программе. Предложение `Inherits` – это первый признак наличия в VB.NET истинного наследования. Вы можете унаследовать поведение базового класса, а затем при необходимости расширить его.

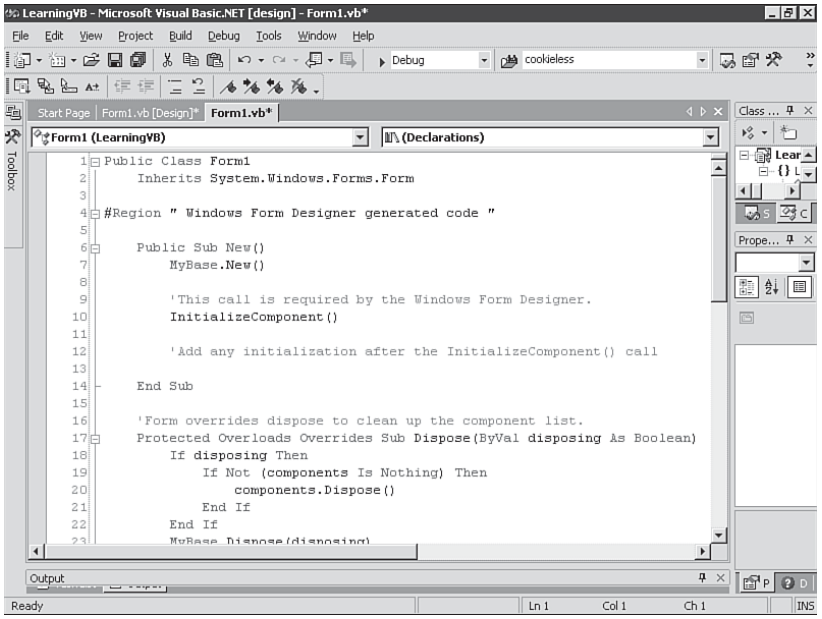

Рис. 2.10. В окне кода присутствует много текста еще до того, как вы набрали первую строку

Код внутри обычно свернутой секции, сгенерированной Дизайнером форм Windows, показан в листинге 2.1[1].

Листинг 2.1. Код, сгенерированный Дизайнером форм Windows

```
#Region " Windows Form Designer generated code "

Public Sub New()
   MyBase.New()

   ' Этот код необходим Дизайнеру форм Windows
   InitializeComponent()

   ' Добавьте свою инициализацию после вызова
InitializeComponent()

End Sub

' Класс Form переопределяет метод Dispose, чтобы очистить список
компонентов.
Protected Overloads Overrides Sub Dispose(ByVal disposing As
Boolean)
   If disposing Then
      If Not (components Is Nothing) Then
         components.Dispose()
      End If
```

[1] Здесь и далее автоматически сгенерированные комментарии переводятся для удобства чтения, хотя в VS.NET вы, естественно, увидите английский текст. – *Прим. переводчика.*

```
    End If
    MyBase.Dispose(disposing)
End Sub
Friend WithEvents Button1 As System.Windows.Forms.Button

' Необходим Дизайнеру форм Windows.
Private components As System.ComponentModel.Container

'ПРИМЕЧАНИЕ: Следующая процедура необходима дизайнеру форм
Windows.
' Модифицировать ее может только сам Дизайнер форм.
' Не изменяйте текст в Редакторе кода.
<System.Diagnostics.DebuggerStepThrough()> Private Sub
InitializeComponent()
    Me.Button1 = New System.Windows.Forms.Button()
    Me.SuspendLayout()
    '
    'Button1
    '
    Me.Button1.Location = New System.Drawing.Point(64, 56)
    Me.Button1.Name = "Button1"
    Me.Button1.TabIndex = 0
    Me.Button1.Text = "Button1"
    '
    'Form1
    '
    Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
    Me.ClientSize = New System.Drawing.Size(292, 273)
    Me.Controls.AddRange(New System.Windows.Forms.Control() {Me.
Button1})
    Me.Name = "Form1"
    Me.Text = "Form1"
    Me.ResumeLayout(False)

End Sub

#End Region
```

Внутри автоматически сгенерированного кода находится открытая процедура `Sub New`. Обратите внимание, что она вызывает процедуру `InitializeComponent`, которую создает VS.NET. Процедура `New` выполняет те же функции, что и обработчик события `Form_Load` в VB6. После вызова `InitializeComponent` вы можете вставить собственный код.

Сразу за процедурой `New` следует процедура `Dispose`, выполняющая те же функции, что и процедура `Form_Unload` в VB6. В ней очищается все, что было создано внутри формы. Значение ключевых слов `Overloads` и `Overrides` будет пояснено при рассмотрении наследования.

В этой же секции находится процедура `InitializeComponent`, которая конфигурирует элементы управления, присутствующие в форме. Обратите внимание, что после задания свойств кнопки, в частности ее положения и размера, она добавляется в набор элементов управления.

За рассмотренной секцией кода идет процедура обработки события щелчка по кнопке. Она также отличается от того, к чему вы привыкли. Если открыть новый проект Standard EXE в VB6, добавить в форму кнопку и дважды щелкнуть по ней, в окне кода будет только такой текст:

```
Private Sub Command1_Click()

End Sub
```

Но VB.NET генерирует намного больше кода. Обработчик события `Click` появляется только в строке 57. Объявление процедуры обработки события выглядит совершенно иначе, чем в VB6:

```
Private Sub Button1_Click()(ByVal sender As System.Object, _
   ByVal e As System. EventArgs) Handles Buton1.Click
End Sub
```

Синтаксис этой строки разъясняется позже. А пока наберите следующую строку в теле процедуры:

```
msgbox "Hello, World"
```

Как только эта строка перестанет быть текущей, VB.NET автоматически заключит аргумент в скобки:

```
msgbox("Hello, World")
```

Это еще одно важное изменение: аргументы процедур и функций должны быть заключены в скобки. `MsgBox` – это функция, но в VB6 скобки вокруг параметров можно было опускать, если возвращаемое значение игнорировалось. В VB.NET скобки нужны всегда, так что привыкайте ставить их или хотя бы не удивляйтесь, когда VB.NET сделает это за вас.

Рис. 2.11. Запущенное приложение Hello, World

Теперь настало время запустить эту программу и посмотреть, будет ли она работать. Можете щелкнуть по иконке **Start** на панели инструментов (она выглядит так же, как раньше) или меню **Debug** и выбрать пункт **Start**.

Должна загрузиться форма `Form1`. Щелкните по кнопке – появится окно сообщения с текстом `Hello, World`. На рис. 2.11 видно, что заголовок окна сообщения совпадает с именем проекта, как это было и в VB6. Закройте приложение и вернитесь в среду разработки.

Обратите внимание, что в нижней части экрана появилось новое окно **Output** (Вывод). В нем есть выпадающий список, содержащий разную информацию. Сейчас вы видите весь отладочный вывод. Вы не вставляли в программу предложений `debug.print`, но некоторые действия компилятора автоматически выводят сообщения в окно отладки.

Прежде чем двигаться дальше, вы, возможно, захотите изменить заголовок окна сообщения. Вернитесь к написанной вами строке кода и модифицируйте ее:

```
msgbox("Hello, World", , "My first VB.NET App")
```

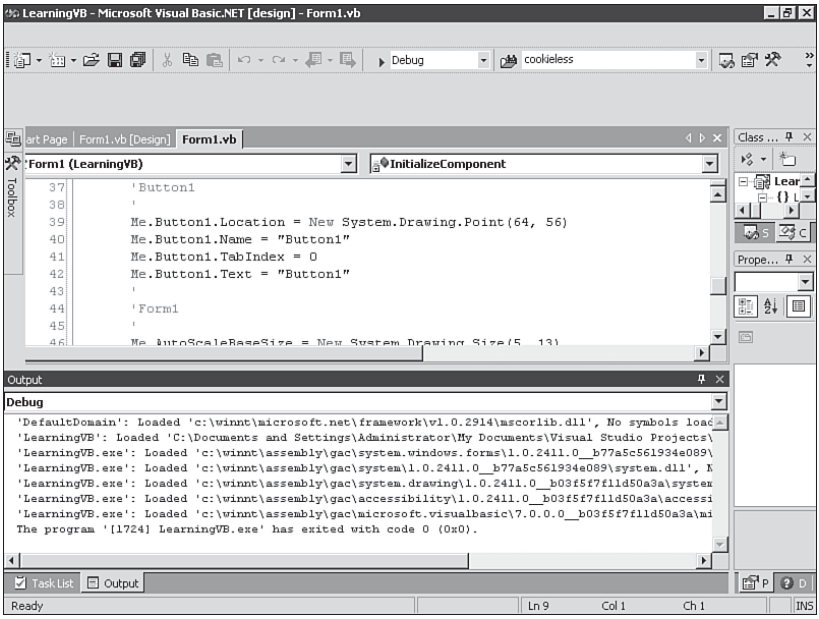

Рис. 2.12. Новое окно отладки

Снова запустите проект – заголовок окна будет содержать текст `My first VB.NET App`. Если никаких мыслей по дальнейшему улучшению программы не приходит в голову, пойдем дальше.

Усовершенствования при разработке приложений Windows

В Visual Studio.NET появилось много новшеств, позволяющих создавать более мощные и удобные формы. Усовершенствована поддержка для построения меню, элементы управления теперь автоматически изменяют размер при увеличении/уменьшении отображаемого в них текста, стала другой стратегия позиционирования элементов при изменении размеров окна, улучшен механизм, задающий порядок обхода элементов.

Автоматическое изменение размера элементов управления

Размер некоторых элементов управления можно сделать автоматически изменяемым в зависимости от длины отображаемого текста. Продемонстрировать это проще всего на примере метки. Откройте набор инструментов и перетащите в форму метку. Расположите ее ближе к левому краю и не в той же строке, в которой находится кнопка.

Теперь измените свойства метки. Пусть свойство `BorderStyle` будет равно `FixedSingle`, а свойство `AutoSize` – `True`. В свойство `Text` запишите строку `This is a test`. Заметьте, что вы изменили именно свойство `Text`, а не `Capti-`

on, как в VB версий 1–6. Вот и еще одно небольшое изменение, которое поначалу может вас смутить.

Теперь модифицируйте код обработчика события Button1_Click. Уберите вызов MsgBox и добавьте такой код:

```
Label1.Text = "Put your hand on a hot stove " & _
   "for a minute, and it seems like an hour. " & _
   "Sit with a pretty girl for an hour, and " & _
   "it seems like a minute. THAT'S relativity."
```

Прежде чем запускать этот пример, обратите внимание еще на одну вещь. В VB6 можно было написать просто

```
Label1 = "Put your hand on a hot stove... "
```

Этот код работал, так как свойством по умолчанию для элемента Label1 было Caption. Так вот, в VB.NET свойств по умолчанию больше нет. Поэтому имя свойства всегда надо указывать.

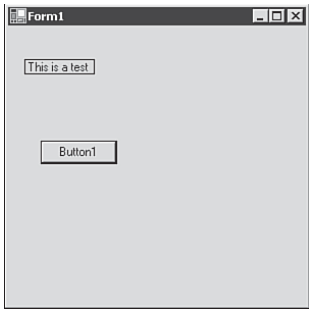

Рис. 2.13. Форма, содержащая метку с коротким текстом

Запустите проект. Появится форма, изображенная на рис. 2.13. Щелкните по кнопке **Button1**, текст метки, как и следовало ожидать, изменился. Обратите внимание, что теперь метка растянулась при попытке показать как можно больше текста. Если вы увеличите ширину формы, то в метке действительно появится весь текст. Это видно на рис. 2.14.

Если вам кажется, что это небольшое достижение, то вспомните, как приходилось выполнять подобное раньше. Нужно было вычислить длину строки и принять во внимание используемый шрифт. Одна и та же группа символов, выведенная шрифтами Arial и Courier, занимает разную площадь на экране. Поэтому для вычисления размера приходилось строить догадки. Свойство AutoSize помогает решить эту проблему.

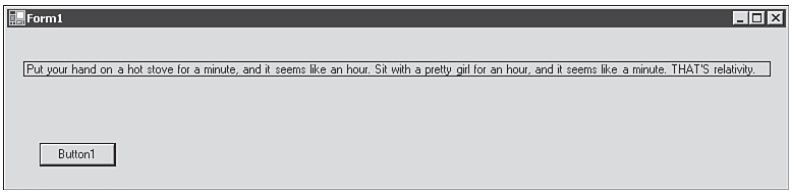

Рис. 2.14. Та же форма, в которой метка растянулась, чтобы вместить весь текст

Привязка элементов управления к краям формы

Сколько раз вам приходилось создавать в VB и присваивать свойству BorderStyle значение Fixed, чтобы нельзя было изменить размер формы? Например,

вы не хотели, чтобы после изменения размера пользователь увидел, что кнопки, расположенные вдоль нижнего края формы, вдруг оказались в середине.

VB.NET позволяет привязывать элементы управления к одному или нескольким краям формы. При изменении ее размера такой элемент будет передвигаться, сохраняя свое относительное положение.

Вернитесь к форме `Form1` и удалите только что добавленную метку и относящийся к ней код в обработчике события `Button1_Click`. Немного увеличьте форму и расположите кнопку **Button1** в правом нижнем углу. Добавьте в форму еще элемент `TextBox`. Измените его свойство `Multiline` на `True` и сделайте так, чтобы поле ввода заняло почти всю оставшуюся в форме площадь, кроме строки, в которой находится кнопка. Форма должна выглядеть так, как показано на рис. 2.15.

Запустите проект. Когда появится окно, измените его размер, сдвинув правый нижний угол ниже и правее. В результате форма примет вид, как на рис. 2.16.

Выглядит это некрасиво, но именно так все происходило в VB6. Если не считать приобретения элемента ActiveX от сторонних фирм, то единственное, что вы могли сделать, – это написать довольно длинный код, срабатывающий при возникновении события формы `Resize`. Назначение такого кода – вычислить новое положение и размеры поля ввода. В VB.NET для этого есть средства получше. Чтобы познакомиться с ними, закройте приложение и вернитесь в среду разработки.

Сделайте текущей вкладку **Form1.vb [Design]** и щелкните по кнопке **Button1**. В окне **Properties** найдите свойство `Anchor`. Раскрыв связанный с ним список, вы увидите серый прямоугольник, от которого в разные стороны отходят четыре

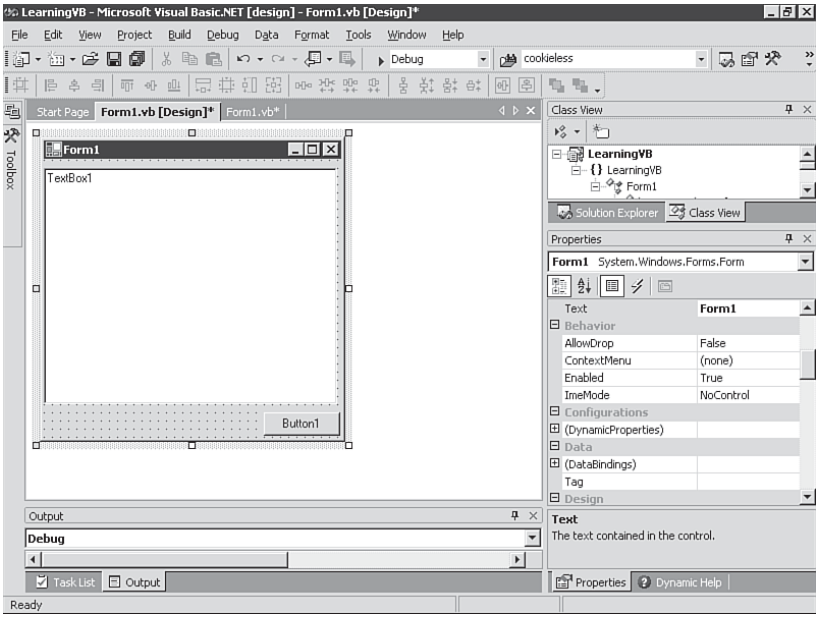

Рис. 2.15. Форма на этапе проектирования

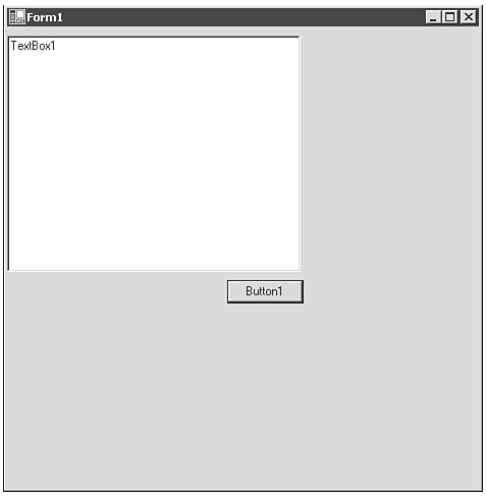

Рис. 2.16. Форма во время выполнения. Так она выглядит после изменения размера

луча. По умолчанию верхний и левый луч темные, а правый и нижний – светлые. Темные лучи говорят о том, что в настоящий момент кнопка привязана к верхнему и левому краям, то есть при изменении размера формы будет сохраняться неизменное расстояние от кнопки до этих краев. Щелкните по верхнему и левому лучам, чтобы они стали светлыми, а затем сделайте темными нижний и правый лучи, также щелкнув по ним. Свойство `Anchor` должно измениться — см. рис. 2.17. После закрытия выпадающего списка значением этого свойства будет `Bottom, Right`.

Далее щелкните по элементу `TextBox1` и выберите его свойство `Anchor`. Щелкните по нижнему и правому лучам, но верхний и левый также оставьте выбранными. Поле ввода теперь привязано ко всем четырем краям, а значение свойства `Anchor` для него равно `Top, Bottom, Left, Right`.

Снова запустите проект. Когда форма появится, измените ее размер. Обратите внимание, что кнопка теперь остается в правом нижнем углу, а поле ввода

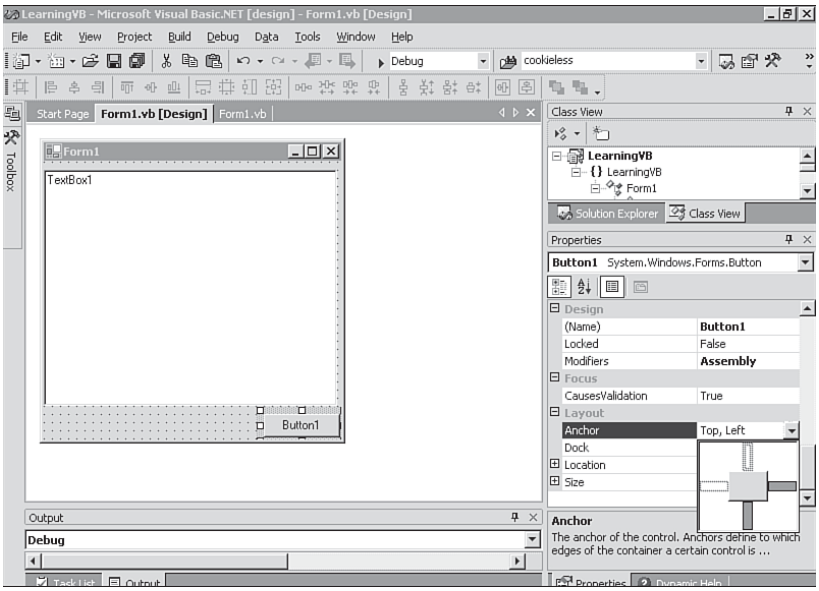

Рис. 2.17. Изменение свойства `Anchor`