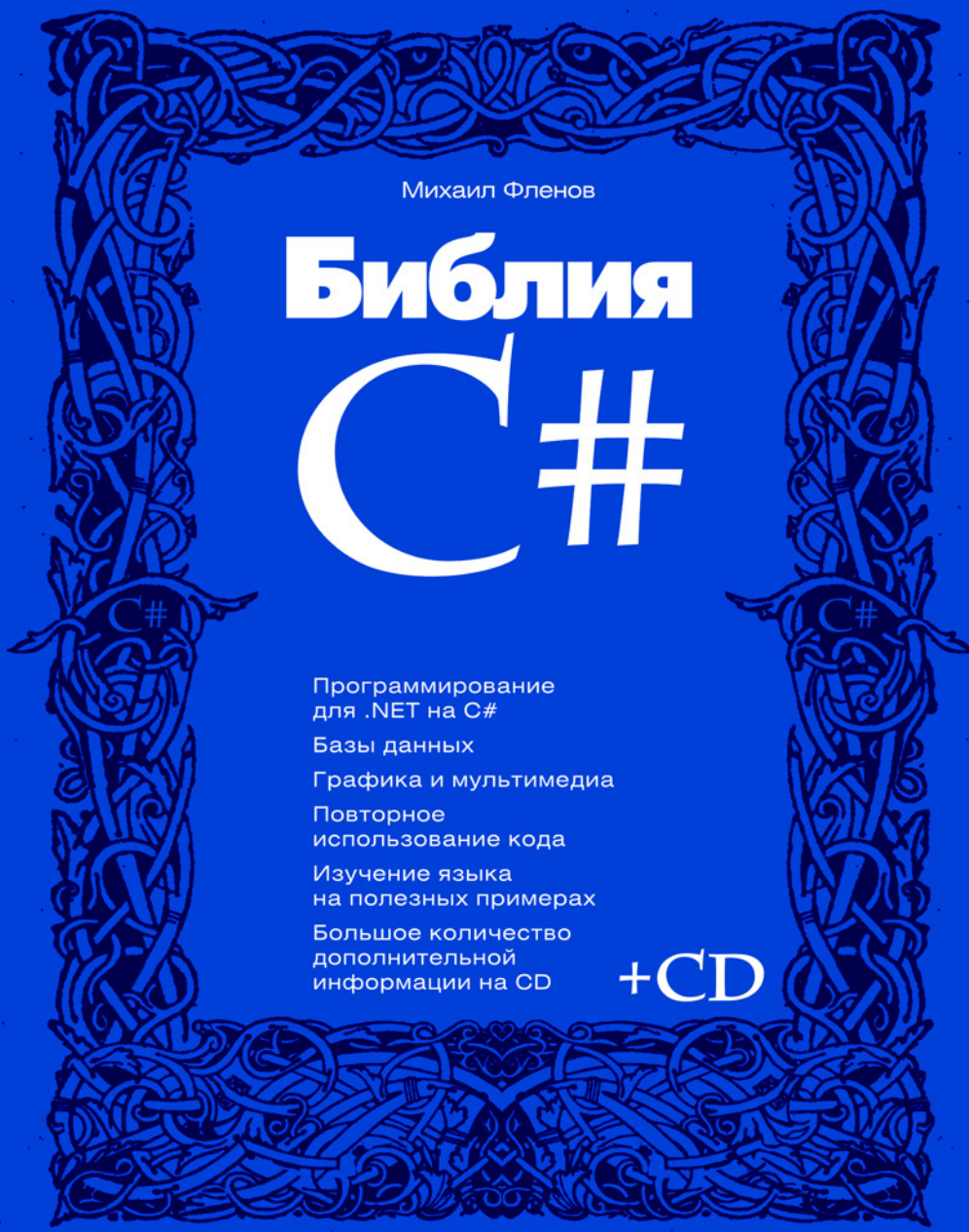

Михаил Фленов
Библия
C#
Программирование
для .NET на C#
Базы данных
Графика и мультимедиа
Повторное
использование кода
Изучение языка
на полезных примерах
Большое количество
дополнительной
информации на CD
+CD
bhv

Михаил Фленов

Библия C#

Санкт-Петербург
«БХВ-Петербург»
2009

УДК 681.3.068+800.92C#
ББК 32.973.26-018.1
Ф69

Фленов М. Е.

Ф69 Библия C#. — СПб.: БХВ-Петербург, 2009. — 560 с.: ил. + CD-ROM

ISBN 978-5-9775-0429-4

Книга посвящена программированию на языке C# для платформы Microsoft .NET, начиная с основ языка и разработки программ для работы в режиме командной строки и заканчивая созданием современных приложений различной сложности (баз данных, графических программ и др.). Материал сопровождается большим количеством практических примеров. Подробно описывается логика выполнения каждого участка программы. Уделено внимание вопросам повторного использования кода. Компакт-диск содержит примеры программ, дополнительную справочную информацию, а также готовые компоненты, тестовые программы и изображения.

Для программистов

УДК 681.3.068+800.92C#
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Нина Седых*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 30.06.09.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 45,15.
Тираж 2000 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.003650.04.08 от 14.04.2008 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0429-4

Оглавление

Введение

.NET — это новая платформа от компании Microsoft, которая состоит из полного набора инструментов для разработчиков (.NET Framework) и для пользователей. Сюда входят клиентская и серверная операционные системы (ОС), инструменты разработки, сервисы. В данной книге мы будем рассматривать .NET Framework, который нужен программистам для написания программ для платформы .NET, а также язык программирования C#, на котором мы и будем писать для .NET Framework. .NET Framework состоит из множества библиотек и классов, которые можно использовать для создания собственных приложений.

Меня очень часто спрашивают, как я отношусь к .NET. К первой версии .NET Framework я относился не очень хорошо, потому что не понял ее и не увидел преимуществ. Когда появилась вторая версия, я кардинально изменил свое мнение по поводу C#. Хотя нет, если быть точным, то мое отношение к .NET первой версии так и осталось скорее негативным, чем положительным или нейтральным. А вот ко второй версии и ко всем последующим я отношусь не то чтобы положительно, а даже с восхищением.

Разработка абсолютно нового языка позволила компании Microsoft избавиться от всего старого и создать что-то новое, чистое и светлое. Наверно, это слишком громкие слова, но просто захотелось сказать эту глупость.

Чтобы понять, почему мне нравится .NET и C#, давайте рассмотрим реальные преимущества, которые я бы выделил. Они уже есть в .NET и никуда не денутся. К основным преимуществам платформы .NET я бы отнес:

1. Универсальный API. На каком бы языке вы не программировали, вам предоставляются одни и те же имена объектов. Все языки для платформы .NET отличаются только синтаксисом, а классы используются из .NET Framework. Таким образом, все языки схожи, и вы выбираете только тот, который вам ближе именно по синтаксису. Я начинал изучать программи-

рование с Basic, затем Pascal, C, C++, Assembler, Delphi, Java и сейчас .NET. При переходе с языка на язык приходится очень много времени тратить на изучение нового API. На платформе .NET больше не будет такой проблемы.

И тут преимущество не только в том, что все языки одинаковы, а в том, что улучшается возможность взаимодействия программ, написанных на разных языках. Раньше для того, чтобы программа на C++ без проблем взаимодействовала с кодом на Visual Basic или Delphi, приходилось применять различные трюки и уловки. В основном, это было связано тем, что каждый язык по-своему обрабатывал и хранил строки. Сейчас такой проблемы нет, и все типы данных в C# абсолютно совместимы с Visual Basic .NET или другим языком платформы .NET.

Таким образом, программисты, использующие различные языки, могут работать над одним и тем же проектом и без швов сращивать модули на разных языках.

2. Защищенный код. Платформу Win32 очень часто ругали за ее незащищенность. В ней действительно есть очень слабое звено — незащищенность кода и возможность перезаписывать любые участки памяти. Самым страшным в Win32 являлась работа с массивами, памятью и со строками (последние являются разновидностью массива). С одной стороны, это предоставляет мощные возможности системному программисту, который умеет правильно распоряжаться памятью. С другой стороны, в руках неопытного программиста данная возможность превращается в уязвимость. Сколько раз нам приходилось слышать об ошибках переполнения буфера из-за неправильного выделения памяти? Уже и сосчитать сложно. На платформе .NET вероятность такой ошибки стремится к нулю, если вы используете управляемый код и если Microsoft не допустит ошибок при реализации самой платформы.

3. Полная ориентированность на объекты. Объектно-ориентированное программирование (ООП) — это не просто дань моде, это мощь, удобство и скорость разработки. Платформа Win32 все еще основана на процедурах и тормозит прогресс. Любые объектные надстройки, такие как Object Windows Library (OWL), Microsoft Foundation Classes (MFC) и др., решают далеко не все задачи.

4. Сборка мусора. Начинающие программисты очень часто путаются, когда нужно уничтожать объекты, а когда это делать не обязательно. Приходится долго объяснять, что такое локальные и глобальные переменные, распределение памяти, стек и т. д. Даже опытные программисты нередко путаются и допускают ошибки при освобождении памяти. А ведь если память освободить раньше, чем это можно сделать, то обращение к несуще-

ствующим объектам приведет к краху программы. На платформе .NET за уничтожение объектов отвечает сама платформа, хотя вы и можете повлиять на этот процесс, но только косвенно. Таким образом, у вас не будет утечек памяти, а вместо того, чтобы думать об освобождении ресурсов, вы можете заниматься более интересными вещами. А это приводит и к повышению производительности труда.

5. Визуальное программирование. Благодаря объектно-ориентированному подходу, стало проще создавать визуальные языки программирования. Если вы программировали на Visual C++, то, наверно, уже знаете, что этот язык далек от идеала, а визуальные возможности сильно ограничены по сравнению с такими языками, как Delphi и Visual Basic. Новый язык C# действительно визуален и по своим возможностям практически не уступает самой мощной (по крайней мере, до появления .NET) визуальной среде разработки Delphi. Визуальность упрощает создание графического интерфейса и ускоряет разработку, а значит, ваша программа сможет раньше появиться на рынке и захватить его.

6. Компонентное представление. Так как платформа имеет полностью объектную основу, появилась возможность компонентно-ориентированного программирования, как это сделано в Delphi. В платформе Win32 были попытки создания компонентной модели с помощью ActiveX, но развертывание подобных приложений являлось слишком сложной задачей. Большинство разработчиков старались не связываться с этой технологией (в том числе и я), а если жизнь заставляла, то для развертывания приходилось создавать инсталляционные пакеты. Самостоятельно создавать пакет — сложно и неинтересно, поэтому среди разработчиков быстро получил распространение пакет InstallShield, позволяющий упростить задачу создания инсталляционных пакетов и упростить развертывание ActiveX. На платформе .NET установка новых пакетов сводится к простому копированию без необходимости регистрации в реестре.

7. Распределенные вычисления. Платформа .NET ускоряет разработку приложений с распределенными вычислениями, что достаточно важно для корпоративного программного обеспечения. В качестве транспорта при взаимодействии используются технологии HTTP (HyperText Transfer Protocol, протокол передачи гипертекстовых файлов), XML (Extensible Markup Language, расширяемый язык разметки) и SOAP (Simple Object Access Protocol, простой протокол доступа к объектам).

8. Открытость стандартов. При рассмотрении предыдущего преимущества мы затронули открытые стандарты HTTP, XML и SOAP. Открытость — это неоспоримое преимущество, потому что предоставляет разработчику большую свободу.

9. Поддержка устройств. На мой взгляд, это основная причина появления платформы. Дело в том, что язык программирования Java оказался наиболее пригодным для различных устройств, и на данный момент этот язык поддерживает большинство мобильных телефонов и карманных устройств. Платформа .NET способна отобрать этот лакомый кусок, и мне кажется, что именно для этого она делалась независимой от аппаратной части. .NET называют межплатформенной, потому что она создавалась с возможностью работы на разных платформах, но сама компания Microsoft не спешит поддерживать конкурентные платформы, такие как Linux или Mac. Остается только наблюдать, что выберут производители устройств и программного обеспечения.

Список можно продолжать и дальше, но уже видно, что будущее у платформы есть. Какое это будет будущее — вот это большой и сложный вопрос. Но, глядя на деньги, которые были вложены в разработку и рекламную кампанию, можно говорить, что Microsoft не упустит своего и сделает все возможное для обеспечения долгой и счастливой жизни .NET Framework.

Данная книга призвана показать вам преимущества новой платформы и научить вас создавать собственные законченные приложения. Мы будем не просто рассматривать процесс расстановки компонентов на форме, а окунемся в глубину платформы и увидим жизнь ваших программ с разных сторон. Это позволит вам создавать более эффективные приложения.

Напоследок хочу выразить собственное мнение о платформе, которое не опирается на какие-либо факты. Как мы выбираем в нашей жизни какие-либо вещи? В основном, опираясь на личный опыт. Да, качество товара тоже играет свою роль, но если вам не нравиться, как выглядит майка, вы никогда ее не наденете.

Я знаком с .NET с момента появления первой версии и пытался изучать ее, несмотря на то, что синтаксис и возможности не вызывали восторга. Да, язык C# вобрал в себя все лучшее от C++ и Delphi, но все равно удовольствия было мало. Тем не менее я продолжал его изучать в свободное время, ибо люблю находиться на гребне волны, а не стоять в очереди за догоняющими.

Возможно, мое первоначальное отрицательное отношение было связано со скудными возможностями самой среды разработки Visual Studio, которая является стандартом, потому что после выхода финальной версии Visual Studio .NET и версии .NET Framework 2.0 мое отношение к языку начало меняться в противоположную сторону. Язык и среда разработки стали намного удобнее, а код читабельным, что очень важно при разработке больших проектов.

На данный момент мое личное отношение к языку и среде разработки Visual Studio .NET более чем положительное. Для меня эта платформа стала номером один в моих личных проектах (на работе я использую то, что требует ра-

ботодатель). Я удивляюсь, как компания смогла разработать за такой короткий срок целую платформу, которая позволяет быстро решать широкий круг задач.

Благодарности

Я хочу в очередной раз поблагодарить свою семью, которая отпустила меня на три месяца ради работы над этой книгой. На новый год и по завершении работы над книгой я постараюсь отплатить им за терпение.

Я благодарю всех тех, кто помогал мне в работе над книгой в издательстве — редакторов и корректоров, дизайнеров и верстальщиков — всех, кто старался сделать эту книгу интереснее для читателя.

Спасибо всем моим друзьям, кто меня поддерживал, и всем тем, кто уговорил меня все же написать эту книгу, а это в большинстве своем посетители моего сетевого журнала **www.flenov.info** и сайта для программистов **www.hackishcode.com**, где вы всегда можете встретить меня на форуме.

Бонус

Я долго не хотел браться за этот проект, потому что люди в наше время не покупают книги, а качают их из Интернета, да и тратить свободное время, отрываясь от семьи, тяжеловато. Очень жаль, что люди не покупают книг. Издательство "БХВ-Петербург" устанавливает не такие высокие цены, и можно было бы отблагодарить всех тех людей, которые старались для читателя, небольшой суммой. Если вы скачали книгу, и она вам понравилась, то я прошу вас купить полноценную версию.

Я же со своей стороны постарался сделать книгу максимально интересной, а на прилагаемый компакт-диск выложил дополнительную информацию в виде статей и исходных кодов для дополнительного улучшения и совершенствования своих навыков. Обязательно убедитесь при покупке книги, что она содержит компакт-диск, потому что на нем очень много полезного и интересного. Это как бы дополнительный бонус для тех, кто покупает полноценную книгу.

Структура книги

Раньше в своих предыдущих книгах я старался как можно быстрее начать показывать визуальное программирование в ущерб начальным теоретическим знаниям. В этой книге я решил потратить немного времени на теоретические

знания, чтобы потом рассмотрение визуального программирования шло как по маслу. Чтобы теория вводной части не была слишком скучной, я постарался писать как можно интереснее и по возможности придумывал интересные примеры.

Первые четыре главы мы будем писать программы, но они будут без графического интерфейса. Информация будет выводиться в консоль. В мире, где властвует графический интерфейс, окна, меню, кнопки и панели, консольная программа может выглядеть немного дико. Но командная строка еще жива и, наоборот, набирает популярность, и ярким примером тут является PowerShell (это новая командная строка, которая поставляется в Windows Server 2008 и может быть установлена в Windows Vista).

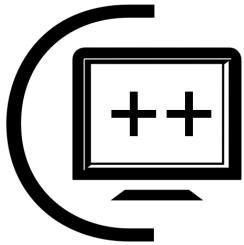
Визуальное программирование мы начнем изучать, как только познакомимся с основами .NET и объектно-ориентированным программированием. Не всегда примеры будут интересными, но я постарался придумать что-то познавательное, полезное и максимально приближенное к реальным задачам, которые вам придется решать в будущем, если вы свяжете свою работу с программированием или просто будете создавать что-то для себя.

Рассмотрев работу с компонентами, мы перейдем к графике и к наиболее важному вопросу для тех, кто работает профессионально в компаниях, — к базам данных. Да, профессиональные программисты чаще всего работают именно с базами данных, потому что в компаниях компьютеры нужны в основном для управления данными и информацией, которая должна быть структурированной и легкодоступной.

Я не пытался переводить файл справки, потому что Microsoft в этом году уже сделала это для нас. Моя задача — научить вас понимать программирование и уметь строить код. А за подробным описанием классов, методом и т. д. всегда можно обратиться к MSDN (Microsoft Development Network, обширная библиотека технической информации, которая располагается на сайте **www.msdn.com**, теперь доступна и на русском языке).

Мир программ и программирования очень интересен и познавателен, особенно для тех, кто любит изучать что-то новое. Никакая книга не способна охватить сразу абсолютно все области программирования даже на каком-то одном языке, поэтому вам может быть придется покупать что-то для дальнейшего изучения. Но я надеюсь, что материала данной книги вам хватит как минимум на год или даже на написание собственного большого проекта.

ГЛАВА 1



Введение в .NET

Платформа Microsoft .NET Framework состоит из набора базовых классов и CLR (Common Language Runtime, общеязыковая среда выполнения). Базовые классы, которые входят в состав .NET Framework, поражают своей мощью, универсальностью, удобством использования и разнообразием. В данной главе мы познакомимся с платформой .NET более подробно. Если вы сможете понять эти основы, то проще будет изучать последующий материал и понимать, почему и зачем разработчики что-то сделали именно так, а не иначе.

После вводной информации в саму платформу мы не будем тянуть время, а сразу начнем изучать язык программирования C# и знакомиться с объектно-ориентированным программированием. Я люблю делать упор на практику и считаю ее наиболее важной в нашей жизни. А по ходу практических занятий будем ближе знакомиться с теорией.

Для чтения этой и последующих глав рекомендуется иметь установленную среду разработки. Желательно повторять все, что описывается в книге, чтобы ощутить это собственными руками. Для большинства примеров достаточно даже бесплатной версии среды разработки Visual C# Express Edition, которую можно скачать с сайта компании Microsoft (**www.microsoft.com/express/ru/**).

Мы рассмотрим только самые основы .NET Framework, чтобы те, кто не читал об этой платформе, смогли получить необходимые для начала знания. Мы изучаем программирование, поэтому на него и сделаем упор.

1.1. Платформа .NET

Во введении мы уже затронули основные преимущества .NET, а сейчас рассмотрим платформу более подробно. На рис. 1.1 показана схема изменений платформы за последние годы. В основе всего, конечно же, стоит операцион-

ная система (ОС). Самое интересное, что Microsoft упоминает только Windows-системы (Vista, XP, 2000 и т. д.). В этом списке нет других платформ, хотя существует реализация .NET для Linux, которая называется Mono.

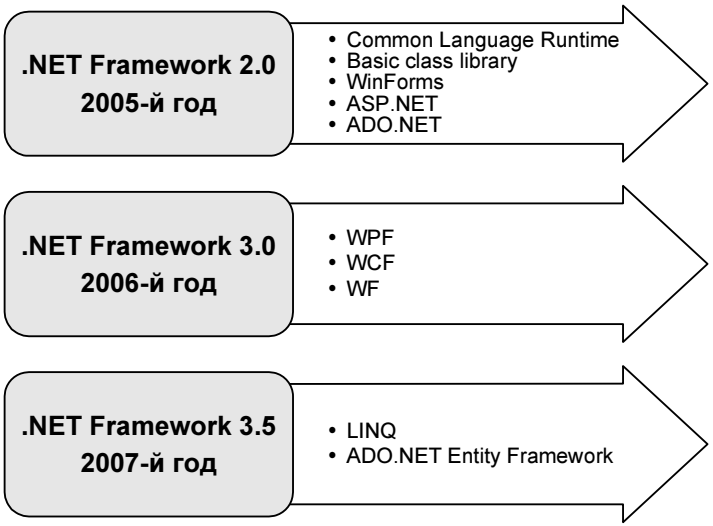


Рис. 1.1. Основа платформы .NET

Платформа .NET состоит из множества различных технологий, серверов и постоянно расширяется. Стоит только посмотреть на составляющие .NET Framework, которые постоянно появляются, и можно понять, что платформа не стоит на месте и будет расширяться в будущем, надеюсь, в таком же темпе. Я специально указал на схеме годы выпуска, чтобы была видна скорость развития.

1.1.1. Кубики .NET

Если отбросить всю рекламу, которую нам предлагают, и взглянуть на проблему глазами разработчика, то .NET описывается следующим образом: с помощью среды разработки Visual Studio .NET вы можете разрабатывать приложения любой сложности с использованием .NET Framework, которые очень просто интегрируются с серверами и сервисами от Microsoft.

Основа всего, центральное звено платформы — это .NET Framework. Давайте посмотрим на главные составляющие платформы:

1. Операционная система. Понятное дело, что все должно работать поверх ОС, но теоретически может работать на специализированном железе, которое умеет выполнять код .NET. На данный момент .NET Framework —

это как бы виртуальная машина, которая является промежуточным звеном между ОС и приложением. Но мне кажется, что недалек тот день, когда все перевернется, и .NET станет основой, а Win32-приложения начнут выполняться в виртуальной машине, как сейчас работают MS-DOS-приложения.

2. CLR (Common Language Runtime). Поверх ОС работает среда выполнения CLR. Это и есть виртуальная машина, которая обрабатывает IL-код (Intermediate Language, промежуточный язык) программы. Код IL — это аналог бинарного кода для платформы Win32 или байт-кода для виртуальной машины Java. Во время запуска приложения IL-код на лету компилируется в машинный код под то железо, на котором запущена программа. Да, сейчас все работает на процессорах Intel, но все может быть, никто не запрещает реализовать платформу на процессорах других производителей.
3. Базовые классы .NET Framework. Как и библиотеки на других платформах, здесь нам предлагается обширный набор классов, которые упрощают создание приложения. С помощью этих компонентов вы можете строить свои приложения как бы из блоков. Когда я услышал это выражение в отношении MFC, то долго смеялся, потому что построение приложений с помощью их классов больше похоже на копание нефтяной скважины лопатой. Компоненты .NET реально упрощают программирование, и разработка приложений с помощью расстановки компонентов действительно стала похожа на строительство домика из готовых блоков.
4. Расширенные классы .NET Framework. В предыдущем пункте говорилось о базовых классах, которые реализуют базовые возможности. Также выделяют более сложные компоненты доступа к базам данных, XML и др.
5. WEB-сервисы, WEB-формы, Windows-формы. Это основа любого приложения. Если вы программировали на Delphi, то аналогом может служить класс `TForm`, который реализует все необходимые окну программы возможности.

.NET не является переработкой Java, у них общее только то, что они являются виртуальными машинами и выполняют не машинный код, а байт-код. Да, они оба произошли из C++, но "каждый пошел своей дорогой, а поезд пошел своей" (из песни Макаревича).

1.1.2. Сборки

Термин "сборка" в .NET переводят и преподносят по-разному. Я встречал много различных описаний и переводов, например, "компоновочные блоки" или "бинарные единицы". На самом деле, если не углубляться в терминологию, а посмотреть на сборки глазами простого программиста, то окажется, что это просто файлы, являющиеся результатом компиляции. Именно конеч-

ные файлы, потому что среда разработки может сохранить на диске после компиляции множество промежуточных файлов.

Наиболее распространены два типа сборки — библиотеки, которые сохраняются в файлах с расширением dll, и исполняемые файлы, которые сохраняются в файлах с расширением exe. Несмотря на то, что расширения файлов такие же, как и у Win32-библиотек и приложений, это все же совершенно разные файлы по своим внутренностям. Программы .NET содержат не инструкции процессора, как в классических Win32-приложениях, а IL-код (Intermediate Language — промежуточный язык, но можно также встретить термины CIL (Common Intermediate Language) или MSIL (Microsoft Intermediate Language)). Этот код создается компилятором и сохраняется в файле. Когда пользователь запускает программу, то она на лету компилируется в машинный код и выполняется на процессоре.

Благодаря тому, что IL-код не является машинным, а интерпретируется JIT-компилятором (Just-In-time Compiler, компилятор периода выполнения), то говорят, что кодом "управляет JIT-компилятор". Машинный код выполняется напрямую процессором, и ОС не может управлять этим кодом. А вот IL-код выполняется на .NET-платформе, и она уже решает, как выполнять, какие процессорные инструкции использовать, а также берет на себя множество рутинных вопросов безопасности и надежности выполнения.

Тут нужно сделать еще одно замечание, чтобы понять, почему программы .NET внешне не отличаются от классических приложений и имеют те же расширения. Это сделано для того, чтобы скрыть сложности реализации от конечного пользователя. Зачем ему знать, как выполняется программа и на чем она написана, это совершенно не имеет значения. Как я люблю говорить, главное — это качество программы, а как она реализована, нужно знать только программисту и тем, кто заинтересован.

Помимо кода в сборке хранится информация об используемых типах данных. Метаданные сборки очень важны с точки зрения описания объектов и их использования. В файле есть метаданные не только данных, но и самого исполняемого файла, где хранится версия сборки и ссылки на используемые внешние сборки. В последнем утверждении кроется одна интересная мысль — сборки могут ссылаться на другие сборки, что позволяет нам создавать многомодульные сборки.

Чаще всего код делят на сборки по принципу логической завершенности. Допустим, что наша программа реализует работу автомобиля. Все, что касается работы двигателя, можно поместить в отдельный модуль, а все, что касается трансмиссии, — в другой. Помимо этого, каждый модуль будет разбивать двигатель и трансмиссию на более мелкие составляющие с помощью классов. Код, разбитый на модули, проще сопровождать, обновлять, тестировать и за-

гружать пользователю. Пользователю нет необходимости загружать все приложение, можно предоставить возможность обновления через Интернет, и программа будет сама скачивать только те модули, которые были обновлены.

Теперь еще на секунду хочу вернуться к JIT-компиляции и сказать об одном сильном преимуществе этого метода. Когда пользователь запускает программу, то она компилируется так, чтобы максимально эффективно выполняться на железе и ОС компьютера, где сборка была запущена на выполнение. Для персональных компьютеров существует множество различных процессоров, и, компилируя программу в машинный код, чтобы он выполнялся на всех компьютерах, программисты очень часто не используют современные инструкции процессоров, чтобы не сужать рынок продаж. JIT-компилятор может использовать эти инструкции и оптимально скомпилировать программу под текущий процессор. Кроме того, разные ОС обладают разными функциями, и JIT-компилятор также может использовать возможности ОС максимально эффективно.

За выполнение сборки отвечает среда выполнения CLR. Основа CLR реализована в библиотеке mscoree.dll, которую называют Microsoft .NET Runtime Execution Engine и которая отвечает за выполнение сборок. Основные типы данных .NET реализованы в библиотеке mscorlib.dll.

1.1.3. Язык программирования

Несмотря на то, что платформа .NET является независимой от языка, и для нее можно писать на многих различных языках, самым популярным является C#, потому что он разрабатывался именно для этой платформы и создавался с нуля. У языка нет какого-то старого наследия, которое нужно сохранить, поэтому он вобрал в себя все лучшее из других языков программирования.

Я также предпочитаю использовать именно C# и вам рекомендую.

На данный момент этих знаний нам будет достаточно. Перейдем к рассмотрению среды разработки Visual Studio .NET и нового языка C#, параллельно с практическими знаниями будем углубляться в платформу. Таким образом, закрепим теоретические знания на практике, а я не раз говорил, что практика лучше всего помогает освоить любую теорию.

1.2. Обзор среды разработки Visual Studio .NET

Познакомившись с основами платформы .NET, переходим непосредственно к практике. Для этого мы познакомимся с новой средой разработки и посмотрим, что она предоставляет нам, как программистам.

Итак, запускаем Visual Studio .NET (рис. 1.2). На момент написания этой книги я использовал последнюю вышедшую версию — 2008. Перед нами открывается главное окно, которое состоит из нескольких панелей.

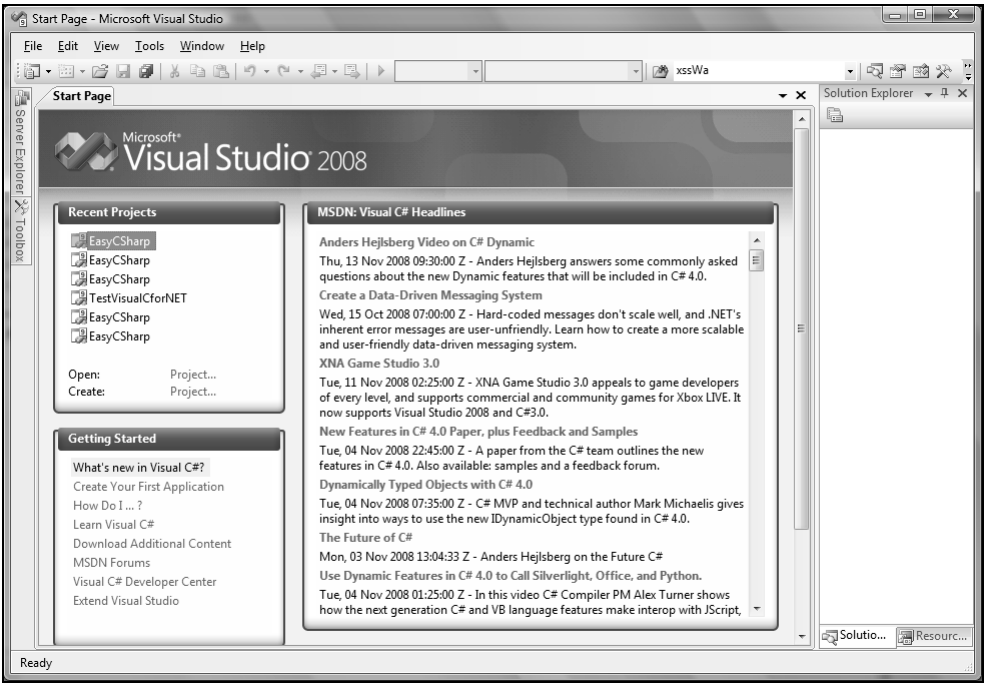


Рис. 1.2. Главное окно Visual Studio .NET

В центре окна расположена рабочая область, в которой в виде вкладок будут открываться файлы. На данный момент в рабочей области открыта HTML-страничка, в которой можно произвести основные настройки поведения среды разработки. Остальные панели мы рассмотрим по мере надобности. Сейчас нет смысла заострять на них внимания, потому что не видно, для чего они нужны.

1.2.1. Работа с проектами и решениями

В Visual Studio любая программа заключается в проект. Проект — это как папка для файлов. Он обладает определенными свойствами (например, платформа и язык, для которого создан проект) и может содержать файлы с исходным кодом программы, который необходимо скомпилировать в исполняемый файл. Проекты могут объединяться в решения (Solution). Более подробно о решениях и проектах можно почитать в файле Documents/Visual Studio 2008.docx на компакт-диске.

Для создания нового проекта выберите в меню **File | New | Project** (Файл | Новый | Проект). Перед вами откроется окно **New Project** (Новый проект), как на рис. 1.3. Слева расположено дерево **Project types** (Типы проектов). Здесь можно выбрать тип проекта. Все типы проектов объединены по папкам:

- **Visual C#** — проекты на языке C#;
- **Visual C++** — проекты на языке C++;
- **Visual Basic** — проекты на языке Visual Basic;
- **Other Project Types** — типы проектов, которые не связаны с определенным языком, например, проект базы данных.

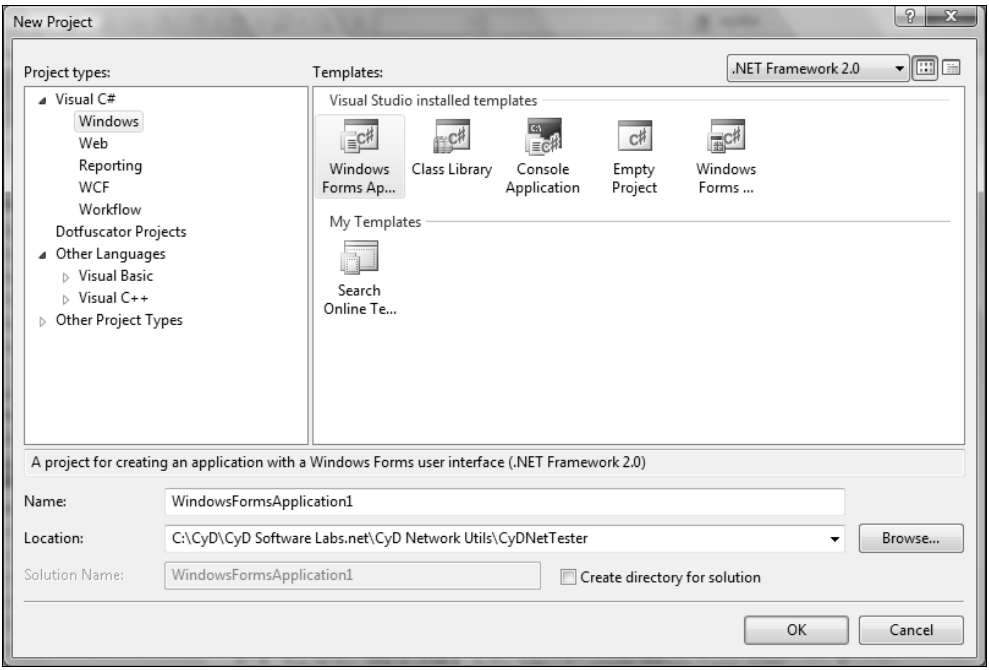


Рис. 1.3. Окно создания нового проекта

В последней версии Visual Studio во время первой записи проекта среда разработки запрашивает, какой язык программирования вы предпочитаете. Я выбрал C#, поэтому этот язык в окне создания проектов у меня находится на первом месте. Остальные языки попадают в раздел **Other Languages**. Проекты для создания пакетов инсталляций и развертывания могут находиться в **Other Project Types**. Почему я говорю "могут"? Дерево разделов изменяется от версии к версии и от типа среды разработки, у меня это Visual Studio 2008.

Среда разработки Visual Studio может работать и компилировать проекты на нескольких языках: Visual C++, Visual C#, Visual Basic и т. д. Для каждого языка есть своя папка в дереве **Project types**. Бесплатные версии среды разработки Express Edition рассчитаны для работы с определенным языком программирования.

В списке **Templates** (Шаблоны) справа появляются типы проектов выбранного раздела. Например, если в дереве **Project types** выбрать тип проекта **Visual C++**, то в списке **Templates** появятся ярлыки для создания проектов этого типа.

Внизу окна есть два поля ввода:

- **Name** — здесь вы указываете имя будущего проекта;
- **Location** — расположение папки проекта.

При задании имени и расположения будьте внимательны. Допустим, что в качестве пути вы выбрали **C:\Projects**, а имя **MyProject**. Созданный проект будет находиться в папке C:\Projects\MyProject. То есть среда разработки создаст папку с именем проекта, указанным в поле **Name**, в папке, указанной в поле **Location**.

В Visual Studio 2008 Express Edition можно создавать проекты без указания их расположения. Проект будет создан в памяти (а точнее, во временном хранилище), и вы можете начинать писать код и даже компилировать и выполнять его. После создания проекта вам будет предложено выбрать папку и имя файла проекта, ведь во время создания вы не указали этой информации, и среда разработки не знает, где нужно сохранить файл.

Файл проекта, в котором находятся все настройки и описания входящих в проект файлов, имеет расширение csproj (для рассмотренного примера это будет файл C:\Projects\MyProject\MyProject.csproj). На самом деле, этот файл имеет формат XML, и его легко просмотреть в любом текстовом редакторе или специализированной программе.

Внизу окна создания нового проекта могут располагаться два переключателя (если у вас уже был открыт какой-то проект в среде разработки):

- **Add to Solution** — добавить в текущее решение;
- **Close Solution** — закрыть решение. Текущее решение будет закрыто и создано новое.

Давайте создадим новый пустой C#-проект, чтобы увидеть, из чего он состоит. Выделите в **Project types** раздел **Visual C#** (этого достаточно, но можно выбрать вложенный в **Visual C#** элемент **Windows**, как на рис. 1.3), а затем в списке **Templates** выберите **Windows Forms Application**. Укажите имя и расположение проекта и нажмите кнопку **OK**.

Вот теперь панели, расположенные справа, заполнились информацией, и имеет смысл рассмотреть их более подробно. Но сначала посмотрим, что появилось слева окна. Это панель **Toolbox** (Инструментальные средства). У вас она может выглядеть в виде тонкой полоски. Наведите на нее указатель мыши, и выдвинется панель, как на рис. 1.4. Чтобы закрепить **Toolbox** на экране, щелкните по кнопке с изображением иголки в заголовке панели (слева от кнопки закрытия панели).

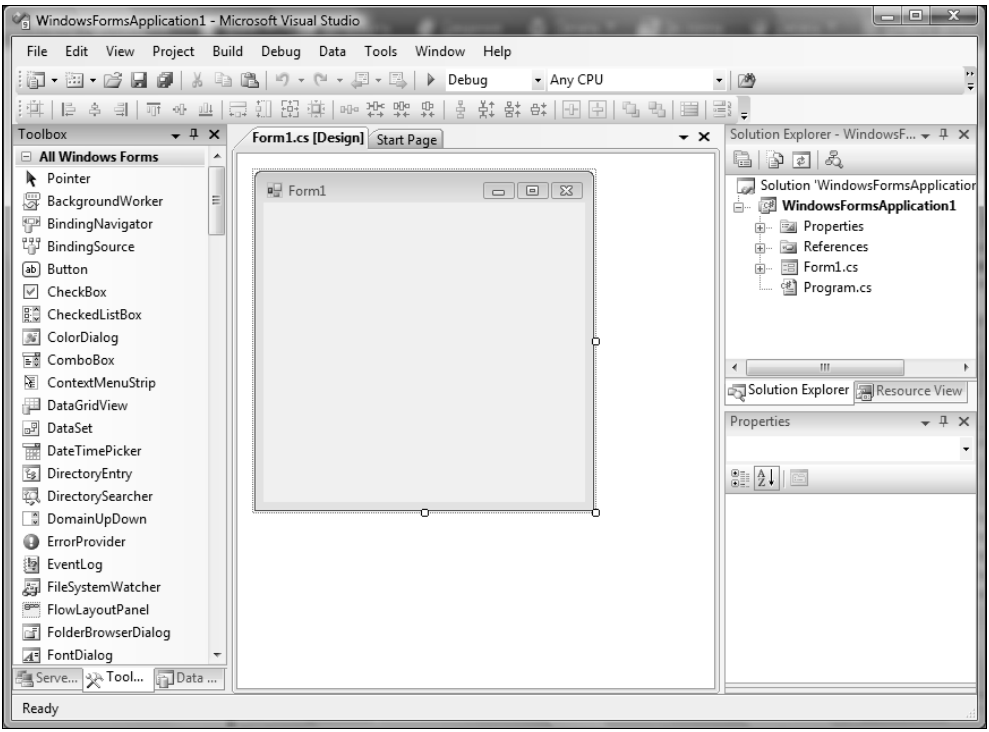


Рис. 1.4. Окно Visual Studio с открытым проектом

Слева внизу окна находятся вкладки для быстрого перехода между тремя панелями:

- **Server Explorer** — панель позволяет просматривать соединения с базами данных и редактировать данные в таблицах прямо из среды разработки;
- **Toolbox** — на этой панели располагаются компоненты, которые вы можете устанавливать на форме;
- **Data Sources** — источники данных.

Любая из этих панелей может отсутствовать, а могут присутствовать и другие панели, потому что среда разработки Visual Studio настраиваема, и пане-

ли можно закрывать, а можно и отображать на экране, выбирая их имена в меню **View** (Вид). Давайте познакомимся с этими и другими панелями, которые стали нам доступны.

1.2.2. Панель *Server Explorer*

Панель **Server Explorer** (рис. 1.5) представляет собой дерево, состоящее из двух основных разделов:

- **Data Connections** — в этом разделе можно создавать и просматривать соединения с базами данных;
- **Servers** — зарегистрированные серверы. По умолчанию зарегистрирован только ваш компьютер.

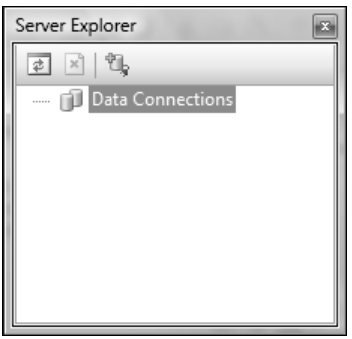


Рис. 1.5. Панель **Server Explorer**

Щелкните правой кнопкой мыши по разделу **Database Connections**. В контекстном меню можно увидеть два очень интересных пункта: **Add Connection** (Добавить соединение) и **Create New SQL Server Database** (Создать новую базу данных). Мы этого делать не будем, а попробуем создать соединение с уже существующей таблицей или базой данных.

Выберите пункт контекстного меню **Add Connection**, и перед вами появится стандартное окно соединения с базой данных (рис. 1.6).

Раскройте ветку созданного соединения, и перед вами появится содержимое этой базы данных. Для Access это будут разделы:

- **Tables** — таблицы, которые есть в базе данных;
- **Views** — представления. Я их часто называю хранимыми запросами выборки, потому что это запросы `SELECT`, которые хранятся в самой базе данных;
- **Stored Procedures** — хранимые процедуры.

Рис. 1.6. Окно настройки подключения с базой данных

Вы можете просматривать содержимое таблиц и хранимых запросов прямо из среды разработки. Для этого дважды щелкните по имени таблицы, и ее содержимое появится в отдельном окне рабочей области.

Теперь переходим к рассмотрению раздела **Servers**. Щелкните правой кнопкой мыши по имени раздела. В контекстном меню вы можете выбрать **Add server** для добавления нового сервера. В моей домашней версии Standard Edition этого пункта в контекстном меню нет, но на работе в Visual Studio 2005 Professional он есть.

Раскрыв ветку своего компьютера, вы можете увидеть список его событий, установленных сервисов, серверов и т. д. Если у вас установлен Microsoft SQL Server, то его вы также увидите в этом списке. В нем вы сможете создавать новые базы данных, как говорится, не отходя от кассы.

1.2.3. Панель *Toolbox*

Это наиболее интересная панель. В ней по разделам расположены компоненты, которые можно устанавливать на форму. Для каждого типа проекта количество и виды доступных на панели компонентов могут отличаться. Мы создали простое приложение Windows Forms, и основные компоненты для этого типа приложений находятся в разделе **All Windows Forms** (рис. 1.7).

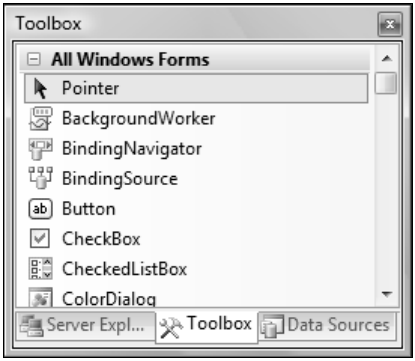


Рис. 1.7. Панель **Toolbox**

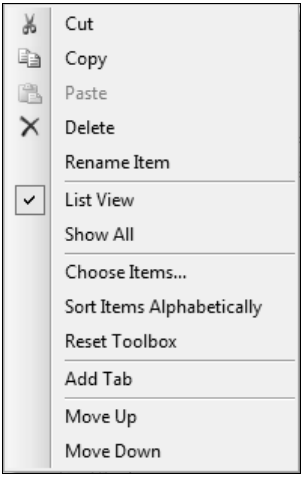


Рис. 1.8. Контекстное меню панели **Toolbox**

Есть несколько способов установить компонент на форму:

1. Перетащить компонент на форму, удерживая левую кнопку мыши. При этом компонент будет создан в той точке, где вы его бросили, а размеры будут установлены по умолчанию.
2. Дважды щелкнуть по кнопке компонента. При этом компонент будет создан в произвольной точке, а размеры будут установлены по умолчанию.
3. Щелкнуть на нужном компоненте, чтобы выделить его. Щелкнуть на форме, чтобы установить компонент. При этом компонент будет создан в той точке, где вы щелкнули, а размеры будут установлены по умолчанию.
4. Щелкнуть на нужном компоненте, чтобы выделить его. Нажать левую кнопку мыши на форме и протянуть курсор до нужных размеров, чтобы установить компонент. При этом компонент будет создан в той точке, где вы щелкнули, а размеры будут установлены в соответствии с обрисованным на форме прямоугольником.

Вы можете настраивать панель **Toolbox** на свой вкус и цвет. Щелкните правой кнопкой мыши в пустом пространстве или по любому компоненту в панели **Toolbox**. Перед вами откроется контекстное меню, как на рис. 1.8.

Рассмотрим его пункты:

- **Cut** — вырезать компонент в буфер обмена;
- **Copy** — скопировать компонент в буфер обмена;
- **Paste** — вставить компонент из буфера обмена;
- **Delete** — удалить компонент;

- **Rename Item** — переименовать компонент;
- **List View** — просмотреть компоненты в виде списка с именами. Если отключить этот пункт, то на панели **Toolbox** будут отображаться только иконки;
- **Show All** — по умолчанию показываются только разделы, содержащие элементы, которые можно устанавливать на данный тип формы, но с помощью этого пункта меню можно отобразить все компоненты;
- **Choose Items** — показать окно добавления/удаления компонентов;
- **Sort Items Alphabetically** — сортировать элементы по алфавиту;
- **Reset Toolbox** — вернуть настройки панели по умолчанию;
- **Add Tab** — добавить вкладку;
- **Move Up** — выделить предыдущий компонент;
- **Move Down** — выделить следующий компонент.

Наиболее интересным является пункт **Choose Items**, который позволяет с помощью удобного диалогового окна **Choose Toolbox Items** (рис. 1.9) создавать и удалять компоненты. Для добавления нового компонента поставьте галочку

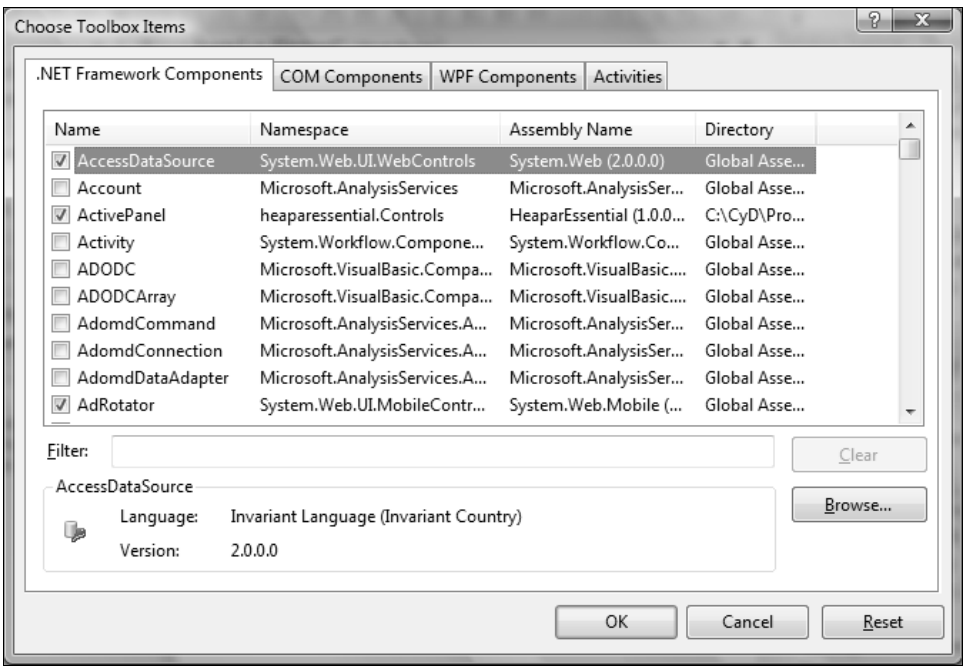


Рис. 1.9. Окно добавления и удаления компонентов