Введение

ЧАСТЬ

I

В этой части...

Глава 1

Знакомство со средой .NET

Глава 2

Отличия VB.NET от VB6

ГЛАВА

1

Знакомство со средой .NET

В этой главе...

Что такое .NET?

Знакомство со средой .NET

Язык CLR

Библиотеки классов

Что такое .NET? Этот вопрос все стали задавать, как только в 2000 году компания Microsoft анонсировала эту свою идею. Если вы интересовались данным вопросом ранее, вы, наверное, слышали про нечто, имеющее название NGWS (Next Generation Web Services). За год до этого ходили слухи, что Microsoft создает новый язык с названием Cool. Или, может, это была платформа? Я не уверен. Я тогда мало уделял этому внимания. Меня больше заботило то, как мои узлы Web растут с помощью компонент COM и ASP. Так как Windows DNA было всем, что нужно для построения устойчивых многоуровневых решений Web, я предполагал, что не может произойти ничего революционного, что заменит эту замечательную технологию.

Я ошибся.

Для меня стало очевидным, что среда .NET станет следующей вехой, когда друг мне дал книгу, посвященную чему-то, называемому ASP+. Хотя это было примерно за год до того, как ASP+ стал доступен массам, уже вышла книга, ему посвященная. Когда я прочитал предисловие к этой книге, написанное разработчиками ASP.NET из Microsoft, мне показалось, что они с самого начала знали наилучший способ написания приложений для Web.

Еще не просохли краски на предыдущем выпуске ASP, а они уже более 3-х лет как работали над новой версией, которую сегодня называют ASP.NET. Я подумал, что эти парни очень находчивы, так как они прислушивались и понимали все, чем были недовольны разработчики, и решили сделать что-нибудь для них.

Скорее всего, это стало началом рождения .NET. Но я в этом не уверен. Тяжело сказать, когда все это началось, но в одном можно быть уверенным: .NET стало результатом сотрудничества многих групп разработчиков Microsoft. От команды COM+ для Windows 2000 до средств разработки для сервера SQL — все, в той или иной степени, прошло через .NET.

Когда вы читаете о среде .NET, то встречаете упомина-

ния серверов .NET, языков .NET, платформы .NET, спецификации .NET и многое другое, что дополняется суффиксом .NET. В этой главе вы получите представление, что такое .NET и что эта среда значит для вас. В этой книге вы изучите язык VB.NET, увидите, как он соотносится со средой .NET и как вы можете использовать этот язык и его средства для трансформирования методологии, которой вы пользуетесь сегодня при напис ании программ.

Что такое .NET

Было много статей, книг, дискуссий о том, что в действительности представляет собой .NET, и, в зависимости от того, с кем вы имели дело, ответ на этот вопрос был разным. На самом деле, ответ прост: .NET является платформой Microsoft для построения Web-сервисов, основанных на XML.

Однако более важно то, что .NET представляет для вас. Неважно, какое определение .NET вы дадите, или что вы прочитаете об этой среде в журналах или Internet, конечной целью этой платформы является разработка и внедрение Web-сервисов простым, безопасным и содержательным способом. Однако это не значит, что теперь вы будете писать только Web-сервисы для всего, что вы создаете. Существуют технологические прорывы в .NET, которые простираются гораздо дальше возможности создавать и использовать Web-сервисы. В процессе чтения данной главы и книги в целом вам этот тезис станет более понятным.

Программное обеспечение как сервис

Программное обеспечение как пример сервиса стало более известным в течение последних нескольких лет. Я видел интервью с представителем компании Oracle на CNET летом 2000 года, где он упомянул, что имеющееся в наличии программное обеспечение остается в прошлом. Единственным путем распространения программ становится Internet.

Он был в чем-то прав, но я не знал, с чего он это взял. В последний раз я попытался скачать последнюю версию Oracle через Internet. Это заняло 27 часов даже на моей высокоскоростной (218 Кбит/с) линии ISDN. После окончания интервью я понял, что он имел ввиду онлайновые продажи услуг через Internet, нечто вроде услуг, которые предлагают порталы. Yahoo, Excite и другие главные портальные сервисы предоставляют их бесплатно. Поэтому необходимы технологии, которые позволят подобным компаниям зарабатывать деньги на подобных услугах. Я никогда не мог понять, как рекламные объявления на страницах могут обеспечить бюджет этих больших порталов, и, в конце концов, эта модель бизнеса могла бы дать сбой. Поэтому существовала потребность в способах организации предложения услуг и взимания денег за эти услуги.

Средства для разработки этого типа технологий существовали и раньше, но они не попадали в главную струю. Существовала потребность в общих методах такого взаимодействия между платформами и серверами посредством Internet или протокола HTTP, чтобы заказчики и провайдеры не были ограничены типами транзакций, которые можно осуществить на том оборудовании и в тех операционных системах, которые они использовали. Или, что еще хуже — чтобы это не зависело от типа используемого броузера.

Simple Object Access Protocol (протокол простого доступа к объектам), или SOAP, был первой попыткой в создании общего и состоятельного механизма для перемещения данных посредством протокола HTTP на любой тип компьютера. SOAP представляет собой набор спецификаций XML, которые описывают, как данные будут передаваться в Internet и приниматься из него. Сообщение SOAP содержит информацию о себе. Сообщение SOAP разбито на части, которые определяют содержание сообщения, цель сообщения и данные о том, как послать ответ на запрос SOAP назад отправителю.

Для того чтобы иметь общую платформу для построения сервисов Web, необходим общий эффективный способ связи через Internet. С помощью SOAP XML может быть использовано для посылки любого запроса, и так как XML — это текстовый файл, содержащий собственное описание, то любая операционная системы или броузер могут быть потребителями Web-сервисов.

Программа как пример сервиса получила свое воплощение с использованием SOAP в качестве общего протокола. Любой узел Web может предложить сервисы, а сервер на другом конце может принять запрос на этот сервис через стандартный порт 80, который использует HTTP. Затем он сможет переслать результаты назад клиенту в виде XML. А клиент уже впоследствии сможет манипулировать этими данными так, как ему это удобно.

Область применения .NET

Во время просмотра презентационного ролика среды .NET, созданного компанией Microsoft, вы видели лозунг: "Область применения .NET исходит из перспектив конечного пользователя". Это значит, что применять .NET на практике будут разработчики, такие, как вы, но, в конечном счете, .NET обеспечит доставку информации конечному пользователю в понятной, наиболее доступной форме и наискорейшим способом.

Когда появились карманные компьютеры, я думал, что это — самая удобная вещь в мире. Судя по анонсам, создавалась видимость беспроводного доступа в Internet, загрузки фильмов, просмотра информации в Outlook из офиса и т.п. Все эти вещи были настолько новыми, что не верилось, что это уже создано. На самом деле прошло два года, пока некоторые из этих вещей перестали быть игровой заставкой. Появление .NET сделало более очевидным то, что такие устройства, как карманные компьютеры, смогут стать действительно полезной вещью. До последнего времени я пользовался карманным компьютером только для чтения книг. Но с появлением Web-сервисов и ASP.NET Mobile SDK, Web-страницы, которые были разработаны для полноформатных броузеров, смогли приспосабливаться к размерам экранов карманных компьютеров и даже мобильных телефонов. При этом проводились только мелкие изменения в тексте программ. На рис. 1.1 наглядно представлены области применения .NET.

Как только доступ к полезному сервису обеспечивается с множества различных устройств, конечный пользователь начинает ощущать их полезность, и область распространения этих сервисов начинает стремительно расширяться. Если вы можете предложить заказчику одно и то же решение, которое он сможет использовать, находясь как в офисе, так и вне его (с помощью мобильного телефона), то, скорее всего, первым будет его вопрос не о стоимости заказа, а о его сроках выполнения.

Для разработчика применение .NET не менее важно, чем для пользователя. Если команда разработчиков хочет превратить процесс создания приложения в сплошную головную боль, она никогда не будет использовать .NET. Компания Microsoft реализовала технологии .NET и создала средства, которые разработчики могут использовать для построения приложений для Web и Windows. Они обладают большим быстродействием и в то же время более простые, чем те, которые вы когда-либо использовали раньше.

Visual Studio .NET предоставляет вам средства для воплощения в жизнь ваших знаний о создании приложений для .NET. Visual Basic всегда был известен благодаря своей высокоэффективной среде разработки приложений для Windows. Выпустив Visual InterDev, компания Microsoft попыталась создать подобный интерфейс и для разработки приложений для Web. Если вы когда-нибудь работали с InterDev, то знаете, что он очень далек от того средства быстрой разработки приложений (Rapid Application Development, или RAD) для Internet, которым должен был стать. VS.NET является действительно средством RAD. По сравнению с лучшими в мире средствами разработки, начиная с Visual Basic, InterDev и FrontPage, или любых

других средств, которые вы могли использовать, Visual Studio .NET является совокупностью всех лучших наработок компании Microsoft в области средств разработки приложений.

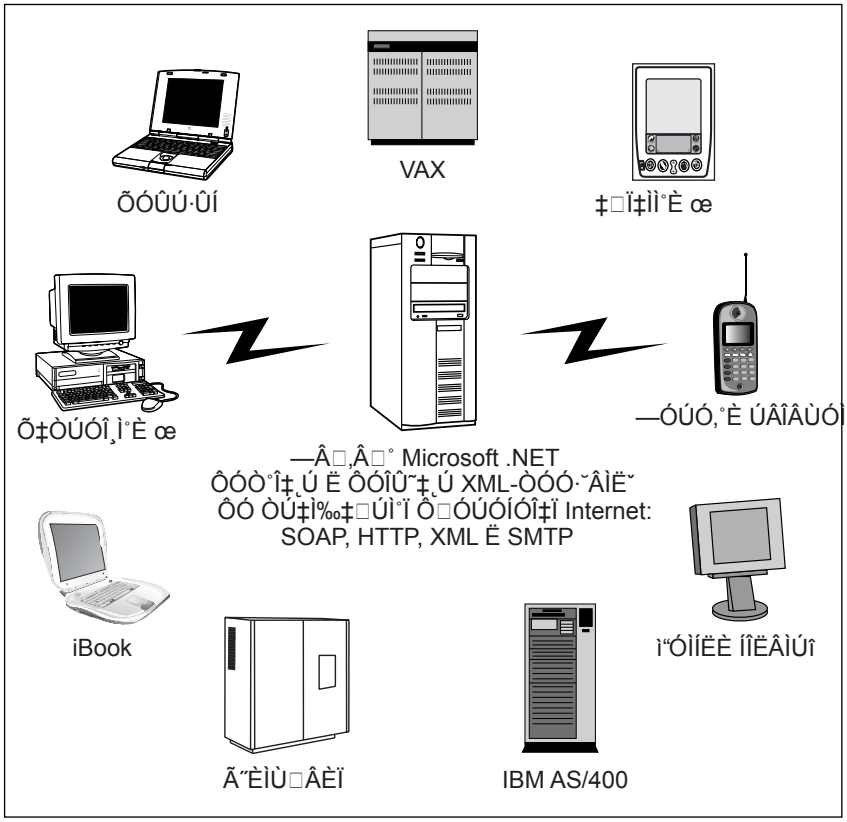


Рис. 1.1. Область применения .NET

Если вы такой же, как я, то у вас нет времени на изучение лишних подробностей. У вас и без этого достаточно работы, поэтому мы оставим в стороне изучение SOAP и способов его работы с .NET. Visual Studio .NET "прошита" XML. XML присутствует во всем, но вы не знаете, где и каким образом. Разработчику в этой среде понятно все: ему остается только одна задача — писать программу. Процесс пересылки и сортировки пакетов XML между клиентом и сервером скрыт. Я упомянул RAD для Internet, но ничего не сказал о том, что VS.NET — это также и RAD для сервера. Это — универсальная среда для разработки приложений и сервисов с технологией клиент-сервер, и, независимо от того, какой язык вы выберете для программирования, вы выполните работу легче и быстрее, чем когда-либо раньше. И что лучше всего — он основывается на уже действующих сегодня стандартах — XML, SOAP, HTTP и HTML.

Давайте рассмотрим некоторые детали, которые делают .NET такой выдающейся средой, и посмотрим, как вы можете ими воспользоваться.

Среда .NET

Среда .NET предлагает необходимые услуги для разработки и использования приложений для слабосвязанной или отсоединенной от Internet среды. На рис. 1.2 показаны ключевые компоненты этой среды.

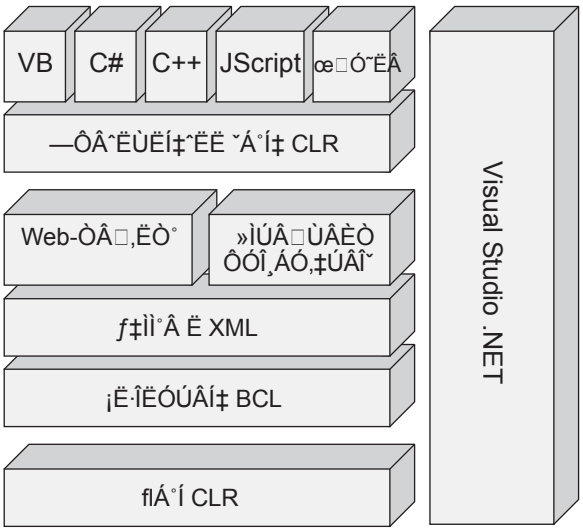


Рис. 1.2. Среда .NET

Два главные компонента, являющиеся фундаментом среды, — Common Language Runtime (CLR) и Base Class Library (BCL). Все, что описано в этой книге, основано на BCL. Как разработчик вы оперируете с наборами классов, производных от классов BCL. В будущем вы сможете также использовать библиотеки классов, созданные третьими лицами и не являющиеся частью этих базовых классов, но все они должны основываться на спецификации CLR.

Остальными сервисами ядра являются межъязыковое взаимодействие, безопасность, управление выполнением, единая система типов (Common Type System). В совокупности все эти сервисы образуют среду .NET.

Common Language Runtime

CLR является фундаментом среды. Цели CLR:

- безопасная и устойчивая среда выполнения;
- упрощенный процесс разработки;
- многоязыковая поддержка;
- упрощенное управление и внедрение.

Как упоминалось ранее, я всегда думал, что Windows DNA был вершиной концепций программирования. Так как я работал только с Windows, передо мной никогда не стояли задачи взаимодействия между многими операционными оболочками, но в реальном мире это самая большая трудность в технологиях COM. COM предоставляет хороший способ интеграции приложений, но каждое приложение должно поддерживать собственную инфраструктуру, и

объекты не имеют непосредственного взаимодействия между собой. Это не подходит для глобальной концепции. В нашем случае для использования всех типов сервисов существует необходимость в лучшем способе реализации связей между отдельными процессами и платформами.

Безопасная и устойчивая среда выполнения

CLR является средой, которая управляет выполнением программы. Программа, которая выполняется внутри этой среды (управляемая программа), при выполнении следует некоторым правилам, установленным CLR. Управляемая программа предоставляет метаданные (данные о данных), необходимые для того, чтобы CLR мог предоставить свои сервисы, такие, как управление памятью, межъязыковое взаимодействие, обеспечение безопасности программы, автоматическое управление периодами существования объектов. Программы, основанные на Microsoft Intermediate Language (MSIL), выполняются как управляемые программы. Управляемая программа — это основное понятие среды. Используя управляемую программу, специфичные для данного процессора компиляторы могут построить языково независимый запрос. При таком типе сценария модель COM выглядит устаревшей.

MSIL является выходным продуктом компиляции приложения .NET. Эта концепция частично знакома разработчикам VB. В прошлом вы имели возможность компилирования программы либо в исполняемый код (который, на самом-то деле, и не являлся исполняемым), либо в P-код, который обрабатывался интерпретатором VB при выполнении программы. MSIL является языком, в который выполняется компиляция из любого языка .NET. После того, как программа будет представлена в этом опосредованном языке, процесс, называемый JIT-компиляцией (Just-In-Time), вызывается каждый раз, когда требуется использование определенных ресурсов в вашем приложении, запущенном на выполнение. JIT позволяет выполняться составным частям вашего приложения тогда, когда в них возникает потребность. Это означает, что, если в какой-либо части не возникнет необходимость, она никогда не будет скомпилирована в PE-файл (portable executable), являющийся уже реальным исполняемым кодом. Используя JIT, CLR может кэшировать фрагменты программы, которые выполняются многократно, и использовать их в последующих вызовах, не повторяя процедуру компиляции. На рис. 1.3 показан процесс JIT.

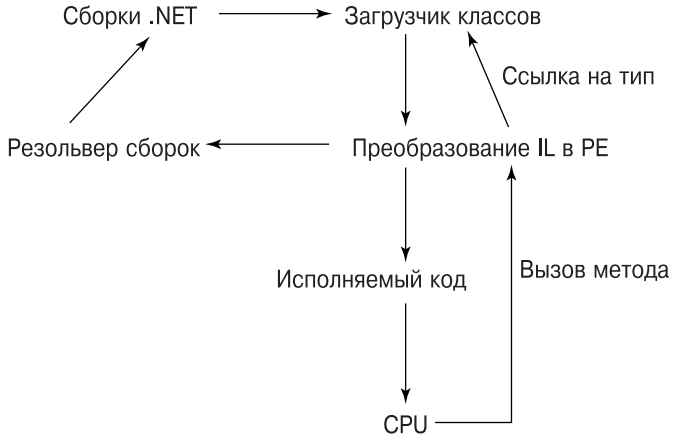


Рис. 1.3. Процесс компиляции JIT

Процесс JIT инициирует среду безопасности, которая предполагает следующее:

- ссылки на типы совместимы с типами, к которым обращаются;

- операции с объектом выполняются только в том случае, если они содержатся в аргументах выполнения этого объекта;
- все сущности в приложении уникальны.

Следуя этим правилам, управляемое выполнение может гарантировать безопасность выполнения в том смысле, что оно будет производиться только в разрешенной к доступу оперативной памяти. Это становится возможным благодаря процессу проверки, который проводится при конвертировании MSIL в предназначенный для процессора код. Во время этой проверки код проверяется на искажения, внутреннюю безопасность и невмешательство в политику безопасности, реализуемую системой.

Обработка исключений

Среда поддерживает структурированную обработку исключений (Structured Exception Handling или SEH) во всех процессах и языках. Когда вы компилируете свое приложение, создаются таблицы, основанные на сопоставлении методов в классах и ошибок, которые могут возникнуть, — обработчикам вызовов этих методов. В неуправляемом окружении ошибки передаются посредством возвращаемых значений типов `HRESULT` и `Boolean`, и не существует общих методов обработки ошибки при ее возникновении. В .NET обработка ошибок интегрирована в саму среду и не является запоздалой реакцией системы.

Удаление ненужных данных

Сроки службы объектов управляются процессом, называемым Garbage Collection (GC). Посредством этого процесса ссылки на удаленные объекты перераспределяются автоматически операционной системой. В VB6 вы должны были явным образом присваивать ссылке значение `Nothing` для того, чтобы гарантировать высвобождение задействованной объектом памяти. В C++ пропуск высвобождения объектов приводил к утечкам в свободной памяти. В .NET управление памятью автоматизировано. Память перераспределяется каждый раз, когда интерпретатор решает, что ссылка на объект больше не используется.

Упрощенная разработка

Понятие упрощенной разработки для различных групп людей может иметь разное значение. В некоторых случаях это значит, что компьютер читает ваши мысли и частично избавляет от набивки текста программы. В других случаях это может значить выигрыш в лотерею отдыха на острове в Тихом океане или даже путешествие на орбитальную космическую станцию Alpha, с*т*оящее 20 миллионов долларов. Упрощенная разработка .NET значит гораздо больше, чем сказанное.

Одно из самых больших изменений в среде — отказ от использования реестра. Реестр являлся врагом всех разработчиков. GUID, файлы IDL, HRESULT и многие другие кошмары, связанные с COM, уже не присутствуют в .NET. Однако сами компоненты COM вы можете продолжать использовать и в .NET.

Подобно добавлению ссылки на библиотеку DLL в VB6, вы можете добавить ссылку на COM DLL в .NET. При этом среда создаст оболочку для DLL, при помощи которой станет возможным использование элементов DLL в управляемом окружении. Вы также можете использовать наборы .NET из неуправляемой среды, такой, как VB6. Все эти возможности не требуют вашей дополнительной работы, поэтому эта среда является очень гибким средством для использования существующих программ в приложении .NET или наборов .NET в среде VB6.

Объектно-ориентированные возможности

Новой концепцией для разработчиков VB является объектно-ориентированное программирование (ООП). ООП упрощает многократное использование и взаимодействие между отдельными компонентами. Классы в среде .NET — полностью объектно-ориентированные. Так как BCL является полностью объектно-ориентированной, то вы можете использовать возможности ООП (такие, как полиморфизм и наследование) во многих языках программирования. Этот ключевой фактор упрощает разработку в больших группах программистов, где часть из них использует VB.NET, в то время как остальные — C++ или COBOL .NET. Независимо от выбора языка вы сможете использовать одни и те же возможности.

Visual Studio .NET

Среда разработки Visual Studio .NET — самая лучшая часть внедрения концепции упрощенной разработки. Средства, доступные в VS.NET, позволяют быстро и легко разрабатывать большие распределенные приложения. Глава 17 углубляется в возможности среды VS.NET, и я уверен, что вы получите большое впечатление, когда начнете применять их на практике при разработке приложений.

Многоязыковая поддержка

На сегодняшний день среда .NET обеспечивает поддержку приблизительно 18 языков программирования. Независимо от того, используете ли вы Pascal, JScript или COBOL, вы получаете полную поддержку средств системы для разработки своего приложения. Так как CLR обеспечивает большую доступность, существует уверенность, что в этот список будут добавлены и другие языки и что эту работу кроме Microsoft будут осуществлять и другие компании.

Кроме пакета VS.NET компания Microsoft поставляет компиляторы для JSript .NET, Visual Basic .NET и Managed C++. Все эти языки полностью поддерживаются средой разработки VS.NET, и в ней существуют компиляторы командной строки для этих языков. Около 15 языков созданы третьими компаниями и имеют либо свои среды разработки, либо должны быть включены в среду VS.NET.

Как это возможно? Среда .NET определяет множество правил, которые устанавливают, как язык может быть обработан в CLR. Этот набор правил называется Common Language Specification (CLS) (Общая спецификация языка). CLS позволяет третьим лицам создавать языки, которые могут внедряться в среду . NET, если их спецификация соответствует CLS.

Благодаря CLS стало возможным межъязыковое взаимодействие. Компоненты, написанные на VB.NET, можно использовать в программах C++ или Managed C++ без создания дополнительных средств. Передача строк из Managed C++ в VB.NET не требует функций конверторов, позволяющих VB.NET использовать эти данные. В среде .NET строка — это всегда строка, одинаковая во всех языках. Это стало возможным благодаря Common Type System (CTS) (Общая система типов), которая внедрена в среду .NET.

CTS определяет, какие типы данных допускаются в среде. Тип может определяться как значение или как ссылка. Типы значений хранятся в представлении их значений (в соответствии с типом значения). Типы ссылок хранятся как ссылки на тип, в частности на объект. Ссылочные типы в .NET основываются на типе System.Object и далее могут разбиваться на классы, управляемые из него. На рис. 1.4 описана спецификация CTS, используемая в среде .NET.

Отладка, трассировка и профилирование доступны для всех языков и всех машин. Так как эти процессы основываются на том, что происходит во время выполнения, то для этого может использоваться всего один отладчик для всех языков, так как он взаимодействует с MSIL независимо от специфики конкретного языка программирования.

Все языки, независимо от тех возможностей, которые предлагают, компилируются в MSIL и затем интерпретируются механизмом выполнения, зависящим от типа процессора. Это значит, что все языки используют одни и те же правила игры. Все они поддерживают правила среды .NET, в противном случае они не будут рассматриваться как языки . NET.

Существуют особенные для каждого языка возможности, включающие встроенные функции, ключевые слова и семантику языка. Однако файл всегда перестраивается в MSIL. Компилятор языка гарантирует, что программа обеспечивает безопасность типов (type safe) и запускает процесс проверки правильности MSIL. Это совсем не значит, что некоторые правила нельзя обойти, поэтому следует тщательно исследовать спецификации CLS и CTS для того, чтобы быть уверенным, что вы используете CLS-совместимые типы.

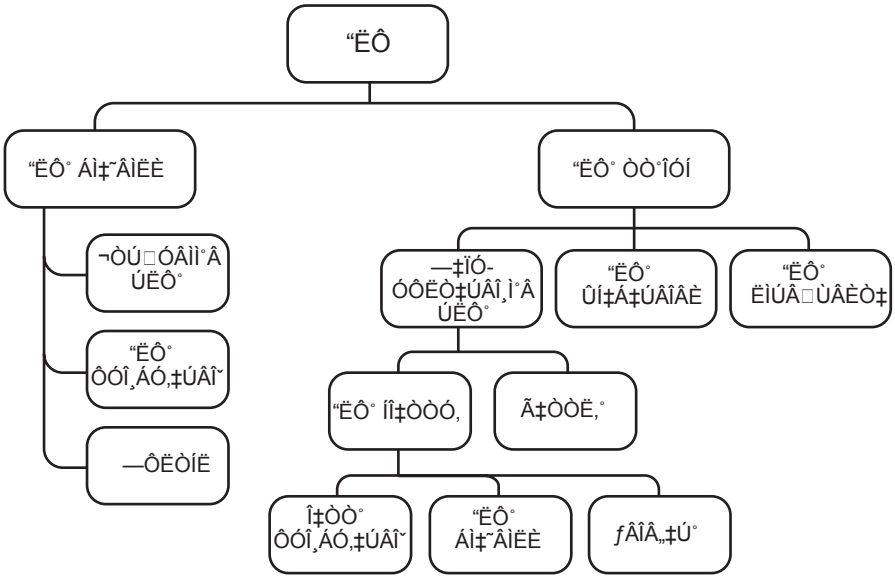


Рис. 1.4. Общая система типов (CTS)

VB.NET является примером языка, который имеет собственные специфические возможности, которыми другие языки не обладают. Так как в мире около 12 миллионов разработчиков использовали VB, и каждый из них имеет десятки тысяч строк написанных программ, то Microsoft присоединила средства модификации для существующих приложений. Они включают в себя библиотеку совместимости, содержащую большинство ключевых слов и функций, использовавшихся раньше в VB6. Когда вы пишете программу в VB.NET, функция `MsgBox` остается приемлемой, независимо от того, что в BCL имеется класс более подходящий MessageBox, который следовало бы использовать. В сущности, функция `MsgBox` является оболочкой, созданной для совместимости, которая в действительности вызывает оригинальный член класса `MesageBox` из библиотеки BCL.

Упрощенное внедрение и управление

Основной единицей размещения в .NET является сборка (assembly). Несмотря на то, что сборки имеют расширение DLL (а иногда — EXE), они не являются стандартными библиотеками COM DLL. Сборка имеет манифест, представляющий собой метаданные, которые предъявляются объекту, вызвавшему данную сборку.

Эти метаданные включают имя сборки, версию, национальную привязку и (необязательно) общедоступный (public) ключ сборки. Остальная информация сборки состо-

ит из данных о том, какие типы экспортируются, на какие типы передается ссылка, и прав доступа к каждому типу.

Сборки делятся на 2 разновидности: *частные* (private) и *разделяемые* (shared).

Частные сборки задействуются посредством инсталляции нулевого влияния (zero impact install). Это значит, что вам не требуется даже шевелить пальцем, чтобы задействовать приложение, так как установка приложения не изменяет состояние компьютера. Так как реестр не задействован, вы просто копируете файлы, которые ваше приложение должно выполнить, в тот каталог, из которого будет производиться их запуск. Этот процесс называется размещение `XCopy`. Все правильно, это хорошо знакомая функция `XCopy` из времен DOS. Вы просто копируете файлы для того, чтобы они работали.

Разделяемые сборки копируются в глобальный кэш сборок (Global Assembly Cache или GAC). GAC — это хранилище файлов, которые могут использоваться несколькими процессами. Когда сборки перемещаются в GAC, они разграничиваются по версии и политикам, определяемым автором сборки.

Side-by-side DLL, введенные с выходом Windoew 2000, полностью внедрены в .NET. На одной и той же машине, в одном процессе библиотеки DLL с одинаковыми именами, но разными версиями, они могут выполняться в одно и то же время.

Библиотеки основных классов

Среда .NET представляет множества иерархических объектов, разбитых по своему назначению, которые называются библиотеками основных классов (base class library или BCL). В этих классах реализовано множество общих объектно-ориентированных интерфейсов, доступ к которым можно получить из любого языка . NET.

BCL разделяется на пространства имен, которые определяют схемы имен для классов, таких, как классы для Web, для данных, для форм Windows, для XML, для сервисов и системные классы. Внедряя схему имен, можно легко определить по названию, какой набор функций она должна обеспечить. Например, классы данных разделяются на следующие пространства имен верхнего уровня:

- `System.Data`;
- `System.Data.Common`;
- `System.Data.OLEDB`;
- `System.Data.SQLClient`;
- `System.Data.SQLTypes`.

Каждое из пространств имен разбивается, в свою очередь, дальше — на более подробные классы, определяющие методы, поля, структуры, перечисления и интерфейсы, которые представляются каждым типом.

Класс `System` осуществляет основные сервисы, которые должны содержать все языки, включая функции ввода-вывода, массивы, наборы, безопасность и глобализацию.

Так как система классов унифицирована для всех языков, то не имеет значения, какой язык пытается получить доступ к базовым классам. Все возможности доступны для всех языков, и обращение к ним в текстах программ осуществляется одинаковым образом. Это способствует легкому пониманию текстов программ, написанных на других языках. Так как используется одна и та же библиотека классов, то все различия в программах сводятся к различию в семантике языков. Если вы разберетесь в этой семантике, то полностью поймете все, что выполняется в программе на другом языке. Рассмотрим эти различия на примере языков VB.NET и C++:

```
' VB.NET
```

```
CodePublic Sub ReadMyData (strCN AS String)
Dim strQuery as String="SELECT * FROM Orders"
Dim cn As New SqlConnection(strCN)
Dim cmd As New SqlCommand(strQuery,cn)
Cn.Open()
Dim rdr As SqlDataReader
Rdr=cmd.ExecuteReader()
While rdr.Read()
     Console.Write(rdr.GetString(1))
End While
Rdr.Close()
Cn.Close()
End Sub 'ReadMyData
```

```
' C++
CodePublic void ReadMyData (string strCN) {
String strQuery="SELECT * FROM Orders";
SqlConnection cn=New SqlConnection(strCN);
SqlCommand cmd=New SqlCommand(strQuery,cn);
Cn.Open();
SqlDataReader rdr ;
rdr=cmd.ExecuteReader();
While rdr.Read(){
     Console.Write(rdr.GetString(1));}
Rdr.Close();
Cn.Close();
}'ReadMyData
```

Оба фрагмента программы полностью идентичны. Исключение составляют только символ точки с запятой в конце строки, обрамление фигурными скобками блоков текста и способ объявления переменных в языке C++.

Visual Basic .NET

Итак, вы получили представление, что такое среда .NET и каковы те функции, которые она выполняет. Что это значит для вас как разработчика программ VB.NET? Ключевыми компонентами в VB.NET, которые облегчат вашу работу как разработчика, являются нововведения в языке, средства RAD, новые модели форм для приложений Web и Windows, и, что наиболее важно, — возможность создавать Web-сервисы.

Нововведения в языке

VB.NET стал в конце концов первоклассным языком. Все возможности, которые обеспечиваются средой, доступны в нем. Он стал полностью объектно-ориентированным языком, реализуя полиморфизм, наследование, перегрузку и переопределение. Структурированная обработка исключений является прозрачным и состоятельным способом обработки ошибок не только в самом методе, но и в цепочке последовательно вызываемых методов и даже в компонентах, написанных на других языках.

Функции RAD

Средства VS.NET осуществляют большинство функций RAD, разработанных для вашей поддержки в процессе написания программы. Усовершенствованные конструкторы данных и

XML, а также обозреватель серверов являются внешними средствами, которые по мановению пальца облегчают создание серверных и клиентских приложений. И все это — без необходимости изучать что-то новое, так как интерфейс среды разработки Visual Studio .NET должен быть вам знаком, если вы работали с предыдущими версиями Visual Basic или даже Visual InterDev. Такие средства, как автозавершение и автоперечисление членов класса, были усовершенствованы, что практически свело задачу написания текста к его проверке на правильность.

Формы Web

Web-формы позволяют разрабатывать приложения для Web в среде разработки, подобной VB. С такими возможностями, как минимизация спагеттиподобного, сгенерированного с помощью Visual InterDev кода и поддержка броузера на нижнем уровне, нет необходимости писать программы для каждого отдельного броузера или его версии. Среда разработки не отличается от аналогичной для приложений Windows, поэтому все возможности, которые доступны при создании приложений для Window, доступны также и при создании Web-приложений.

Web-сервисы

Добавляя к методу приставку-идентификатор `<webmethod>`, являющуюся примером программирования, основанного на атрибутах, вы создаете вызываемый метод Web-сервиса. Взаимодействие с ним обрабатывается средой, а размещение сводится к копированию файла на Web-сервер. Создание Web-сервиса не сложнее создания любого другого типа приложения.

Формы Windows

Формы Windows представляют собой переписанные генераторы форм, предназначенные специально для платформы .NET. Те же классы, которые используются в VB.NET, разделяются также и с другими языками, и формы Windows могут работать как компоненты броузера. Кому теперь нужны апплеты?

Резюме

Как видим, среда .NET может предложить нам многое не только с позиций самой среды, но и с позиций средств разработки. Беря на вооружение возможности Common Language Runtime, вы можете создавать приложения, которые являются объектно-ориентированными вплоть до ядра и могут взаимодействовать с другими приложениями .NET, написанными на любом из 18 языков .NET. .NET произвела революцию в методах написания текстов приложений разработчиками, вооружив их устойчивым набором средств, разработанных компанией Microsoft. Это очень знаменательный момент: читая эту книгу, вы создаете себе прочный фундамент, на котором будут стоять ваши будущие приложения .NET.

ГЛАВА

2

Отличия VB.NET от VB6

В этой главе...

Изменения в типах данных

Изменения в массивах

Изменения операторов

Область видимости переменных

Изменения в процедурах и свойствах

Управление потоком

Изменения в формах

Изменения в типах приложений

Visual Basic .NET является самым большим прорывом в языке Basic со времен скачка GW-Basic в Visual Basic 1.0. Как и в любом принципиально новом продукте, в нем произошли изменения, направленные на облегчение работы, однако в первое время делающие ее сложнее. Это вы почувствуете, когда будете переходить на VB.NET. Одной из причин стало то, что переход осуществляется не только к новому языку, но и к новой платформе среды .NET. Все, что вы изучали в последние 4–5 лет (Windows DNA, COM+, ASP), также осуществило переход на рельсы .NET. Когда я впервые увидел VB.NET, он мне показался странным и лишенным смысла. Я продолжал мыслить категориями VB6. Я не мог понять, зачем все модули должны быть консольными приложениями — я никогда этого раньше не делал в VB6. Было очень тяжело осознать новую идею классов и понятие CodeBehind, покуда вся моя работа над программой не тронулась с места. Я боялся изучать SOAP и XML, так как они всюду присутствуют в среде .NET. Однако, как и во всем, нужно было с чего-то начать. Поэтому я вначале посмотрел на изменения, которые произошли в сравнении с VB6, затем написал несколько небольших приложений и постепенно понял, что VB.NET является самой удивительной вещью в мире. С этого мы и начнем в этой главе.

В процессе чтения этой книги вы видите, что произошли некоторые изменения во внешнем виде программ VB. Когда вы начнете писать программу, то увидите, что они не так уж и велики. Изменения в синтаксисе имели смысл, и с такими возможностями, как автодополнение, автоперечисление членов класса и динамическая помощь, писать программы стало легко как никогда. Первое, что вы должны полностью усвоить, — это те устаревшие и неиспользуемые операторы и функции, которые хоть и выглядят, как и раньше, но имеют несколько другое значение.

Изменения в типах данных

Среда .NET содержит классы, определенные в общей системе типов (Common Type System), позволяющие типам данных применяться во всех приложениях, написанных на разных языках .NET. Из-за этого VB и был вынужден изменить используемые типы данных и диапазоны, ограничивающие их значения. В следующих разделах описаны эти изменения.

Все типы данных и диапазоны их значений приведены в главе 5.

Тип Variant не поддерживается

В VB6 универсальным типом данных по умолчанию являлся тип `Variant`. В VB.NET он был заменен на тип `Object`. Значением по умолчанию для типа `Object` является `Nothing`, в то время, как для этой цели в типе `Variant` служило значение `Empty`.

```
Dim var1 As Variant
```

изменилось на

```
Dim var1 As Object
```

Integer и Long

В VB6 тип `Integer` представлял 16-битное целое число в диапазоне от 32 767 до -32 767. Теперь для этого представления используется тип `Short`, а `Integer` стал представлять 32-битные целые числа, варьирующиеся от -2 147 483 648 до 2 147 483 647. Тип `Long` теперь представляет 64-битные целые числа. Для 32-битных операций тип `Integer` является самым эффективным.

```
Dim X As Integer
Dim Y As Long
```

изменилось на

```
Dim X As Short
Dim Y As Integer
```

Тип Currency не поддерживается

Тип `Currency` в VB.NET изменился на десятичный (`Decimal`). Этот новый тип более точен в операциях округления чисел, поэтому и был создан для поддержки денежных операций. Тип `Currency` представлял 64-битное число с 4-мя десятичными знаками. Новый тип данных `Decimal` — это знаковое 64-битное число, которое может иметь до 28 десятичных знаков.

```
Dim X As Currency
```

изменилось на

```
Dom X As Decimal
```

Изменения в типе даты

Тип данных `Date` — теперь 64-битное целое число, в то время, как в VB6 был 64-битным числом с плавающей точкой. Для преобразования типов `double` и `Date` между собой используются функции `ToOADate` и `FromOADate`.

```
Dim X As Double
```

изменено на

```
Dim X As Double, Y As Date
Y=X.ToOADate
```

Строки

Строки фиксированной длины более не поддерживаются. В документации SDK говорится, что этот тип появится в последующих версиях. Существует уровень совместимости, который допускает строки фиксированной длины, но это не поддерживается напрямую CLR.

DefType не поддерживается

Оператор `DefType`, который по умолчанию присваивал тип переменным, для которых не был явно указан тип, — более не поддерживается, равно как и аналогичные операторы `DefInt`, `DefStr`, `DehBool` и `DefLng`.

VarPtr, StrPtr, ObjPtr

Эти функции, которые возвращают целочисленный адрес переменных в вызовах API, более не поддерживаются. Метод `AddrOfPinnedHandle` класса `Ghandle` обеспечивает выполнение тех же задач.

Массивы

Одним из самых значительных изменений, которые Microsoft анонсировала в VB.NET, является изменение нижней границы массива. С самого начала своего существования VB всегда позволял устанавливать нижний индекс массива в 0 или 1. Это осуществлялось с помощью оператора `Option Base`, который боле не поддерживается в VB.NET. Microsoft приняла решение: всегда считать базисом индексов массивов 0. Это значит, что все программы, в которых базисом считалась единица, должны быть проанализированы на предмет недопущения нарушений в данных, связанных с изменением размеров массива.

Массивы начинаются с нулевого элемента

Теперь при создании массива нельзя указывать нижнюю и верхнюю границы индексов. Можно задавать только верхнюю границу; нижняя же принимает значение по умолчанию — ноль.

```
Dim x(0 to 5) as String 'массив из 6 элементов
```

изменился на

```
Dim x(5) as String 'массив из 6 элементов
```

или

```
Dim x as String=New String(5) 'массив из 6 элементов
```

Option Base не поддерживается

Оператор `Option Base` для определения нижнего индекса массивов во всем модуле или форме более не поддерживается. Нижний индекс всех массивов считается равным нулю.

ReDim изменился

Оператор `ReDim` не может быть использован при определении переменной массива. В VB.NET сначала нужно определить массив оператором Dim, и только затем использовать оператор `ReDim` для изменения размеров массива.

Значение True

В VB6 значением истины (`True`) было минус единица. То же самое осталось и в VB.NET. Однако в CLR значение `True` — это единица. Когда булевские переменные передаются в среде .NET, то значением `True` считается для VB — минус единица, а для всех остальных языков — единица.

Операторы

Самые значительные изменения в языке VB произошли в категории операторов. В таблице 2.1 перечислены новые операторы присвоения, которые значительно сокращают ввод текста при их использовании. Внимательно прочитайте главу 5, где подробно описаны все эти операторы и приведены удачные примеры их использования.

EQV

Оператор `EQV` заменен на оператор присвоения `=`.

Укороченная проверка

Операторы `AndAlso` и `OrElse` были добавлены для укороченной проверки логических выражений. Все остальные операторы не изменились. При укороченной проверке если первое из выражений имеет значение `False`, то остальная часть выражения игнорируется и в качестве результата возвращается `False`.

```
Dim x as Integer = 5
Dim y As Integer = 6
Dim z As Integer = 7
Ret = x>y AndAlso z>y 'возвращается False, т.к. 5 не больше, чем 6
```

Присваивание

В языке VB.NET предлагается несколько новых удобных операторов присваивания.

```
Ddim intX As Integer
IntX=intX+1
```

можно заменить на

```
Dim intX As Integer
IntX+=1
```

В табл. 2.1 перечислены новые операторы присваивания, которые сокращают текст программы.

Таблица 2.1. Новые операторы присваивания

Оператор	Действие
`+=`	Сложение и конкатенация
`-=`	Вычитание
`*=`	Умножение
`/= и \=`	Деление
`^=`	Возведение в степень
`&=`	Конкатенация строк

Типы, определяемые пользователем

Типы, определяемые пользователем, больше не поддерживаются в VB.NET. Они заменены структурами, которые имеют сходный синтаксис, но больше возможностей и гибкости.

```
Public Type MyCust
     StrName As String
     StrEmail As String
End Type
```

заменено на

```
Public Struct MyCust
     Private StrName As String
     Private StrEmail As String
End Type
```

Значение Null

Объявление `Null` более не поддерживается. Значения типов, которые равны `null`, передаются в функцию как типы `DBNull`. Для проверки таких значений больше не используется оператор `IsNull` — он заменен на оператор `IsDBNull`.

```
If IsNull(field) then …
```

заменен на

```
If IsDBNull(field) then …
```

Больше не поддерживается оператор `IsEmpty`. В VB6 значения `NULL` и `EMPTY` использовались для проверки переменных на предмет их инициализации или на предмет наличия в них данных. `Null` и `Empty` больше не имеют значения при проверке переменных. Вместе с ними утратило значение выражение `IsEmpty`. Для проверки этого условия теперь служит ключевое слово `nothing`.

Область действия переменных

Переменные могут быть определены в блоках операторов, в пределах которых и могут использоваться. В следующем примере переменная `intX` доступна только внутри блока `If … End If`, а переменная `intY` — в пределах всей процедуры.

```
Private Sub Test ()
     Dim intY as Integer
     If intY>5 Then
        Dim intX as Integer
        'выполняем действия
     End If
     IntX=5 'вызовет ошибку, т.к. intX здесь уже не определен
     IntY=10 'правильное выражение, т.к. intY определен во всей про-
цедуре
End Sub
```

Инициализация переменных

Переменным могут теперь присваиваться начальные значения в строке их объявления.

```
Dim intX as Integer
IntX=5
```

теперь можно заменить на

```
Dim intX as Integer = 5
```

Инициализация объекта может быть также проведена с помощью ключевого слова `New`, помещенного в строку объявления. Это поведение в корне противоположно принятому в VB6. Там это не одобрялось из-за способа, которым COM убеждалось, что объект вначале создается, а впоследствии используются его свойства и методы.

```
Dim cn as Connection
Set cn=New Connection
```

можно изменить на

```
Dim cn as SQLConnection = New SQLConnection 'это более эффективно
```

Теперь можно объявлять в одной строке несколько переменных одного типа. Рассмотрим следующий пример:

```
Dim Str1, Str2, Str3 as String
```

В VB6 переменные `str1` и `str2` имели бы тип `Variant`, и только `str3` имела бы тип `String`. В VB.NET все три переменные будут иметь тип `String`.

Переменные ParmArray

Переменные `ParmArray` больше не передаются по ссылке (`ByRef`). Они теперь передаются по значению, а принимающая функция может изменять значения `ParmArray` так, как будто это обычный массив.

Соглашения языка

Особенное внимание при переходе к языку VB.NET следует уделить изменениям в ранее существовавших функциях. В этом разделе рассмотрены специальные функции, синтаксис которых изменился. В главе 8 рассмотрены подробности работы с типами дат и времени, которые очень важно понять, так как они являются наиболее часто используемыми функциями, с которыми приходится постоянно работать.

IsMissing

Функция `IsMissing` более не поддерживается. И должна быть заменена оператором `IsNothing`.

Date$ и Time$

Функции `Date$` и `Time$`, которые в VB6 возвращали строчные представления текущей даты и времени, заменены в VB.NET методами `DateString` и `TimeString`.

Atn, Sgn и Sqr

Функции `Atn`, `Sgn` и `Sqr` заменены методами пространства имен `System.Math` — `Atan`, `Sign` и `Sqrt`.

MsgBox

Функция `MsgBox` заменена методом `Show` класса `MessageBox`.

```
MsgBox "VB6 is great"
```

изменилось на

```
MessageBox.Show "VB.NET is greater"
```

Хотя функция `MsgBox` и продолжает поддерживаться библиотекой совместимости, все же в новых программах вам следует использовать класс `MessageBox`, так как функция `MsgBox` является только оболочкой вокруг метода нового класса.

Процедуры

Так как VB.NET полностью поддерживает объектно-ориентированные возможности, произошли критические изменения в написании процедур, возвращаемых ими значениях и типах процедур, которые можно использовать. Кроме изменений, которые упоминаются здесь, главы 8 и 14 полностью ответят на вопрос, что можно делать с процедурами в VB.NET.

Вызов процедур

Вызов процедур (равно как и функций) теперь требует наличия скобок, обрамляющих список аргументов, даже если он пустой.

```
Sub Test()
     'code
end Sub
call Test
```

изменилось на

```
call Test()
```

Статические процедуры

Статические процедуры более не поддерживаются. Если вы использовали статические процедуры, то следует их переделать в обычные подпрограммы и определить все переменные внутри них в `Static`.

```
Static Sub Test()
     Dim X as Integer
End Sub
```

изменилось на

```
Sub Test()
     Static X as Integer
End Sub
```

ByRef, ByVal и As Any

В VB.NET изменилось на противоположное значение по умолчанию для способа передачи параметров в процедуру. В VB6 параметры передавались по умолчанию по ссылке (`ByRef`), а в VB.NET теперь таким механизмом является передача по значению (`ByVal`). Оператор `As Any` в выражении `Declare` для вызовов API не поддерживается.

Свойства

Синтаксис описания процедур свойств изменился для того, чтобы соответствовать объектно-ориентированным возможностям VB.NET. В главе 14 приводятся реальные примеры использования свойств и определяются особенности синтаксиса, которые следует применять.

Let, Get и Set

Свойство `Let` более не поддерживается. Когда используется `Get` или `Set`, указание области их действия должно остаться таким же и в VB.NET: блок `Get` определялся как `Private`, а блок `Set` — как `Public`. Это можно явно обнаружить в новом синтаксисе определения этих свойств.

```
Friend Property strName() as String
     Get
     End Get
     Set(ByVal value as String)
```

```
    End Set
End Property
```

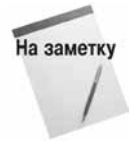
Свойства по умолчанию

Свойства по умолчанию более не поддерживаются в объектах. В VB6 свойства по умолчанию имели определяемые пользователем объекты, а также визуальные объекты, такие, как текстовые поля, списки и комбинированные списки.

```
StrName=Text1 'Text1 — это имя текстового поля формы
Me.Caption="Hello World"
```

изменилось на

```
StrName=Text1.Text 'используется свойство Text
Me.Text="Hello World"
```



Свойства по умолчанию можно продолжать создавать. В главе 14 описывается, как это может быть выполнено.

Управление порядком выполнения

Изменения в операторах порядка следования выполнения не такие уж страшные. Самое большое изменение коснулось ключевого слова `GoSub`, но так как разработчики его не использовали со времен VB 2.0, оно не имеет такого значения. Оператор `Return` — это новый способ возвращения значения из функций.

While … Wend

Конструкция `While … Wend` более не поддерживается. Она заменена аналогичной конструкцией `While … End While`.

```
While X<5
     'code
Wend
```

изменилось на

```
While X<5
     'code
End While
```

GoSub … Return

Переход к другой процедуре `GoSub` в теле функции более не поддерживается. Следующий фрагмент будет ошибочным.

```
Sub Test
     Return
End Sub
Sub Test2
```

```
     GoSub Test
End Sub
```

Return

Оператор `Return` теперь передает управление из функции вызывающей программе. Если функция имеет определенный тип, он может также вернуть в вызывающую программу ее возвращаемое значение.

```
Private Function GetNames () as String
     Dim StrName as String
     StrName="Seven of Nine"
     GetNames=strName
End Function
```

изменено на:

```
Private Function GetNames () as String
     Dim StrName as String
     StrName="Seven of Nine"
     Return strName
End Function
```

Изменения в приложениях, основанных на формах

Приложения Windows Forms имеют полностью отличную подсистему, в которой форма выводится на страницу, что делает необходимым убрать некоторые функции, предназначенные для отображения информации на форме. Одной из новых возможностей в VB.NET является интерфейс для внедрения меню в формы и элементы управления печатью, которые обеспечивают предварительный просмотр печати в приложениях Windows Forms.

PrintForm

Метод `PrintForm` более не поддерживается в VB.NET. Среда .NET имеет новую подсистему для печати с такими возможностями, как предварительный просмотр, который позволяет более точно настроить слепки экрана и печать форм.

Circle, Cls, Pset и Point

Эти методы более не поддерживаются в VB.NET. Графические методы VB6 полностью заменены пространством имен `System.Drawing`, которое использует новые классы GDI+ для рисования.

Свойство Caption

Свойство `Caption` в элементах меток и форм больше не поддерживается — его заменило свойство `Text`.

```
Label1.Caption="VB.NET"
Form1.Caption="My Form"
```

изменилось на:

```
Label1.Text="VB.NET"
Form1.Text="MyForm"
```

Twips на формах

Единица измерения для форм `Twips` изменена на пикселы.

Массивы управления

Множество элементов, имеющих одно и то же имя, но различные значения свойства `Index`, могли использовать одни и те же процедуры событий в VB6. В VB.NET элементы не могут быть сгруппированы в массивы, однако одна и та же процедура может использоваться для нескольких элементов. Для этого подойдет ключевое слово `Handles`.

Контекстное меню и главное меню

В VB6 меню, добавляемое в форму, могло использоваться и как главное меню, и как контекстное, раскрывающееся по правому щелчку мыши. В VB.NET вы можете определить меню как главное или как контекстное, но не в двух качествах одновременно.

DDE

DDE более не поддерживается. Предложен другой подход к организации связей между приложениями с использованием OLE, Web-сервисов или компонент процесса.

Объект Clipboard

Объект `Clipboard` был заменен классом `Clipboard`.

Изменения в элементах

Когда вы создаете новое приложение Windows Forms, вы обнаруживаете некоторые новые впечатляющие элементы в панели инструментов. Элементы, перечисленные в этом разделе, более не поддерживаются, и поэтому не содержатся в панели инструментов.

Элемент OLE

Элемент контейнера OLE, позволявший добавлять в форму элементы OLE, более не поддерживается.

Элемент Image

Элемент `Image` заменен в VB.NET элементом `PictureBox`.

Элемент Line

Элемент `Line` был заменен методом `Draw` из GDI+.

Элемент Shape

Элемент `Shape` был заменен методом `Draw` из GDI+.

Типы приложений

Многие из шаблонов, которые вы могли использовать в предыдущих версиях VB, более не существуют. Некоторые типы приложений даже не имеют перспектив к развитию.

Webclass

Приложения типа WebClass более не поддерживаются. Для написания приложений для Web используются Web Forms. Мощь и функции, которые обеспечиваются Web-сервисами и приложениями ASP.NET, дадут вам бо́льшую гибкость, чем программирование приложений Webclass, которым вы раньше занимались.

Документы ActiveX

Приложения документов `ActiveX` более не поддерживаются. Тем не менее, вы можете писать приложения Windows Forms, которые запускаются броузером как безопасная программа, имитирующая документ `ActiveX`. В дополнение вы получите полный объект Windows Forms с полной поддержкой его элементов.

Приложения DHTML

Приложения DHTML более не поддерживаются. Для написания приложений для Web используются Web Forms.

Элементы пользователя

Все элементы пользователя, созданные в VB6, можно использовать в VB.NET, однако в нем не существует поддержки конструирования для изменения или редактирования этих элементов.

Страницы свойств

Страницы свойств более не поддерживаются, и у них нет будущего. Однако это ничего не меняет, так как они использовались только в приложениях элементов `ActiveX`, и в VB.NET вы можете создать элементы пользователя взамен элементов `ActiveX`.

Доступ к данным

ADO.NET представляет собой новую библиотеку для доступа к данным из любой программы. Она в корне отличается от предыдущей версии ADO, которую вы раньше могли использовать. Главное место в переводе приложений на новые рельсы занимает исключение подсоединения данных (data binding). Когда переход на новую версию осуществляется с помощью Upgrade Wizard, вы будете предупреждаться обо всех фрагментах программ, которые

не могут быть автоматически переведены, если вы использовали подсоединение данных любого вида.

Присоединение данных, ADO, RDO

Присоединение данных к источнику данных ADO или RDO более не поддерживается. Следующий фрагмент программы является устаревшим:

```
Text1.DataField="Customer_Name"
Set Text1.DataSource=rs
```

В VB.NET присутствует существенно более мощный и передовой механизм присоединения данных, и его применение в корне отличается от предыдущих версий.

DAO

Data Access Objects более не поддерживаются в VB.NET. Так как вы получаете доступ к данным с помощью ADO.NET, потерю механизма DAO вы не ощутите. Это значит, что вы должны конвертировать текст, использующий DAO, в текст, применяющий механизмы ADO.NET.

Отладка

Visual Basic всегда обеспечивал передовую поддержку процесса отладки. Метод `Debug.Print` был заменен, но это — мизерная потеря по сравнению с теми новыми возможностями, которые делают процесс отладки легче, чем когда-либо ранее. В главах 18 и 42 описаны техники отладки как для приложений Windows, так и для Web.

Debug.Print

Метод `Debug.Print` заменен методами `Debug.Write` и `Debug.Writeline`. Класс `Debug` содержит полный набор методов и свойств, которые помогают в отладке программ.

```
Debug.Print Err.number
```

заменено на

```
Debug.Writeline Err.number 'включает перевод строки
```

или

```
Debug.Write Err.number 'не включает перевод строки
```

Debug.Assert

Старый метод `Debug.Assert` более не поддерживается. Он заменен методами `Assert` и `Fail` класса `Debug`.

Резюме

В этой главе рассмотрены главные изменения в VB.NET по отношению к VB6. Большинство из них относятся к редко используемым вызовам функций, однако при переходе на новую версию вам необходимо знать причины, почему ваша программа изменена.

VB.NET содержит пространство имен Microsoft.VisualBasic, которое содержит большинство функций и ключевых слов VB, которые продолжают поддерживаться в .NET. Однако лучше избегать его использования в противовес новым системным пространствам имен `System` для того, чтобы сделать вашу программу совместимой со всеми остальными языками .NET.