Основы платформы Microsoft .NET

Тема:

Введение в технологию Microsoft.NET

Введение

Даже небольшая по длительности история существования компьютерной техники позволяет утверждать, что программирование относится к числу наиболее сложных областей человеческой деятельности. Трудоемкость создания программных систем управления критическими производствами, корпоративных систем большого размера, систем решения крупных научно-технических проблем, систем поддержки процессов принятия решений в разных областях приложений сопоставима (а порой и превышает) сложность разработки крупных материальных объектов. Для преодоления возникающих трудностей при разработке такого рода программных систем теория и практика в каждый конкретный исторический момент времени формирует вполне определенный перечень проблем, решение которых будет способствовать выходу из очередного "кризиса программирования". Выделим этот критический список проблем (абсолютно не претендуя на полноту и однозначность), который существовал в программировании в конце прошлого столетия, и на разрешение которого было направлено создание корпорацией Microsoft своего флагманского на данный момент времени продукта – технологии Microsoft.Net:

- **Разнообразие частных решений для решения задач разработки крупномасштабного программного обеспечения** – как результат, явно ощущалась потребность в разработке некоторого общего подхода, в котором бы критически учитывались все имеющиеся решения, и в рамках которого с единых позиций можно было бы разрешать многие проблемы информационной индустрии,

- **Сложность интеграции существующих решений в рамках единых программных систем** – различие аппаратно-программных платформ, предлагаемых корпоративных решений, вариантность (версионность) программных компонент выводит проблему интеграции разрабатываемого ПО в число наиболее острых задач программирования,
- **Трудоемкость разработки распределенных программных систем** – возникающие при разработке распределенных систем проблемы обеспечения надежности, безопасности и масштабируемости требовали создания более общих средств решения, определения признаваемых подходов и стандартов,
- **Широкое распространение Интернет технологий** – мир Интернета требовал осмысления накопленных после появления Java решений и ожидал промышленного перехода на технологии сервис-ориентированного программного обеспечения и др.

Приведенный перечень не дает, конечно, исчерпывающего представления о всех проблемах программирования конца XX столетия, но и данного списка достаточно для понимания – мир программирования нуждался в некотором новом общем и промышленном подходе к разработке крупномасштабного программного обеспечения. И такое решение было представлено корпорацией Microsoft летом 2000 г. как *новая платформа Microsoft.Net. для разработки и исполнения программного обеспечения.*

Понятие платформы MS.NET

Итак, под платформой Microsoft.NET следует понимать интегрированную систему (*инфраструктуру*) средств разработки, развертывания и выполнения сложных (как правило, распределенных) программных систем [1].

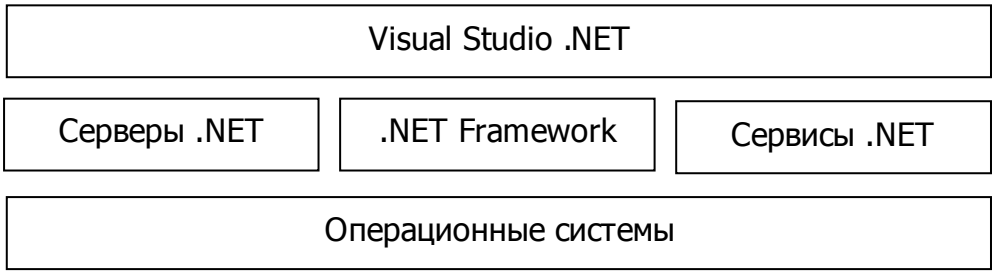

Рис. 1. Платформа Microsoft.NET

Платформа .NET состоит из нескольких основных компонентов (см. рис. 1):

– *операционные системы* корпорации Microsoft (Windows 2000/XP/ME/CE), представляющие собой базовый уровень платформы MS.Net,

– *серверы MS.Net* (.Net Enterprise Servers) являются программными продуктами корпорации Microsoft, использование которых позволяет снизить сложность разработки сложных программных систем. В числе готовых для применения серверы Application Center 2000, Exchange Server 2000, SQL Server и др.,

– *сервисы MS.Net* (.Net Building Block Services) представляют собой готовые "строительные блоки" сложных программных систем, которые могут быть использованы через Интернет как сервисные услуги. Набор таких сервисов MS.Net планируется последовательно расширять. Примером имеющегося сервиса платформы MS.Net является Microsoft Passport, позволяющий установить единое имя пользователя и пароль на всех сайтах, поддерживающих аутенфикацию через Passport,

– *интегрированная среда разработки* приложений Visual Studio.NET (VS.Net) – верхний уровень MS.Net - обеспечивает возможность создания сложного ПО на основе платформы и продолжает в этом плане ряд разрабатываемых корпорацией Microsoft средств разработки профессионального программного обеспечения.

Центральной частью платформы MS.Net, как можно заметить по рис. 1, является *подсистема Microsoft.Net Framework*.

Структура Microsoft.NET Framework

Как уже отмечалось выше, подсистема MS.NET Framework является ядром платформы MS.Net, обеспечивая возможность построения и исполнения .Net приложений [1].

На верхнем уровне рассмотрения в составе MS.NET Framework могут быть выделены (см. рис. 2) *общеязыковая среда выполнения* (Common Language Runtime или CLR) и *библиотеки классов* подсистемы MS.NET Framework. По своему функциональному назначению в составе библиотек классов могут быть выделены:

– набор *базовых классов*, обеспечивающих, например, работу со строками, ввод-вывод данных, многопоточность и т.п.,

– набор *классов для работы с данными*, предоставляющих возможность использования SQL-запросов, ADO.Net и обработки XML данных,

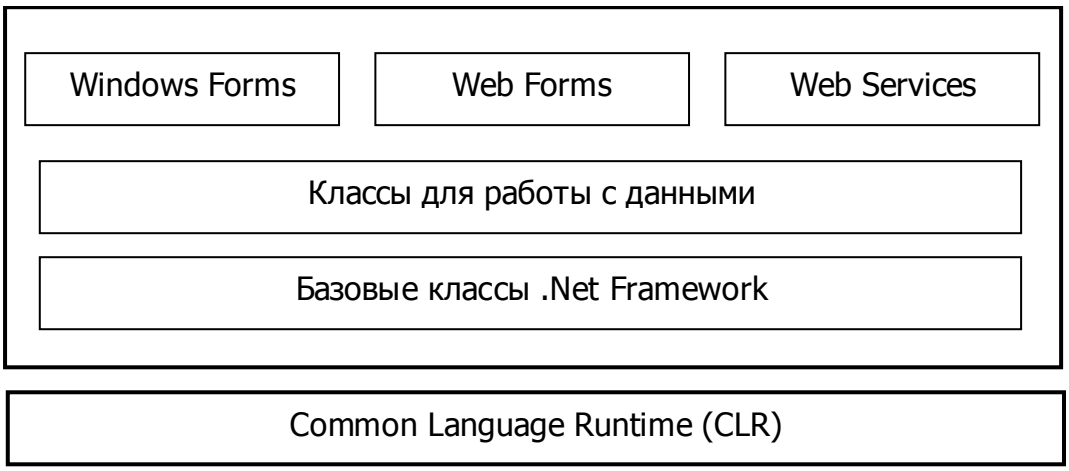

Рис. 2. Архитектура MS.NET Framework

– набор *классов Windows Forms*, позволяющих создавать обычные Windows-приложения, в которых используются стандартные элементы управления Windows,

– набор *классов Web Forms*, обеспечивающих возможность быстрой разработки Web-приложений, в которых используется стандартный графический интерфейс пользователя,

– набор *классов Web Services*, поддерживающих создание распределенных компонентов-сервисов, доступ к которым может быть организован через Интернет.

Базовый уровень подсистемы MS.NET Framework (см. рис. 2) составляет *общеязыковая среда выполнения* (Common Language Runtime или CLR).

Введение в среду Common Language Runtime

Как следует из самого названия, общеязыковая среда выполнения CLR обеспечивает исполнение программного кода и является в этом плане основой платформы MS.Net [1,2]. Среда CLR активизирует исполняемый код, выполняет для него проверку безопасности, располагает этот код в памяти и исполняет его. Важной частью работы среды CLR является управление свободной памятью, автоматически обеспечивая использование освобождающейся при работе программ памяти (*сборку мусора*).

Для представления общей схемы функционирования среды CLR дадим ряд важных для платформы MS.Net понятий и определений:

- Для обеспечения возможности многоязыковой разработки ПО программный код, получаемого после компиляции программы на одном из алгоритмических языков платформы MS.Net, представляется на специально разработанным в корпорации Microsoft *общем промежуточном языке* (Common Intermediate Language или CIL). Этот язык, с одной стороны, достаточно близок к машинно-зависимым языкам – ассемблерам, с другой

стороны, CIL обеспечивает некоторый более высокий уровень представления различных компьютерных платформ. Как результат, программа на языке CIL остается платформенно-независимой, однако требует некоторой дополнительной настройки (компиляции) перед началом своего выполнения,

- Программные файлы на языке CIL, получаемые после компиляции программ на алгоритмических языках платформы MS.Net, называются *сборками* (assembly), другое более технические наименование сборок - *переносимые исполняемые файлы* (Portable Executable или PE). Сборки являются основными структурными элементами *(.Net компонентами*) разрабатываемого в рамках .Net программного обеспечения и, относятся, тем самым, к числу наиболее важных понятий платформы MS.Net. В конструктивном плане, сборки являются файлами с расширениями exe или dll и состоят из непосредственно программного кода на языке CIL и дополнительных служебных данных, именуемых в MS.Net *метаданными* (в составе метаданных необходимая информация о сборке – сведения о типах, данные о версии, ссылки на внешние сборки и т.п.),
- Как уже отмечалось выше, сборки перед своим исполнением должны пройти определенную настройку для работы в условиях конкретной выбранной платформы – для выполнения таких настроек в составе среды CLR имеется ряд *JIT-компиляторов* (Just-In-Time compilers), вызываемых для перевода программного кода на промежуточном языке (CIL-кода) в машинный (native) код платформы исполнения.

С учетом введенных понятий представим общую схему исполнения сборок в среде CLR (см. рис. 3):

– При запуске сборки загрузчик распознает .Net приложение и передает управление среде CLR, далее загрузчик классов находит и загружает класс, в котором содержится точка начала работы сборки (обычно функция Main), затем CLR передает управление сборке,

– Загрузчик классов выполняется каждый раз при первом обращении к необходимым для выполнении сборки классам. Загрузчик находит нужный класс, размещает информацию о типах класса в кэше и загружает класс для выполнения (при этом в методах загружаемого класса делается отметка, что они не являются еще готовыми для выполнения, поскольку не прошли через JIT-компиляцию),

Рис. 3. Среда CLR: общая схема исполнения сборок

– Верификатор является частью JIT-компилятора и отвечает за проверку корректности CIL-кода и метаданных. Подобный контроль повышает надежность работы программ, с другой стороны, верификация может и не осуществляться (например, для доверенного кода),

– JIT-компиляторы вызываются при обращении к CIL-коду, который ранее при работе программы еще не компилировался. В результате компиляции получается машинный код, оптимизированный для выбранной платформы, для откомпилированного метода устанавливается адрес полученного машинного кода и управление передается исполняемой сборке. Как результат, компиляция методов классов осуществляется только в момент первого к ним обращения (и, тем самым, не используемые методы классов останутся неоткомпилированными). Подобная схема компиляции по мере необходимости может существенно снизить размеры порождаемого программного кода и сократить время подготовки сборки для выполнения, вместе с этим, в среде CLR может быть выполнена и полная компиляция сборки перед началом исполнения (pre-JITing),

– Непосредственное исполнение машинного кода также происходит при активном взаимодействии со средой CLR. Функции среды выполнения (см. рис. 3) состоят в управлению свободной памятью, обработке исключений, поддержке безопасности и др.

Примеры программ для платформы MS.Net

Первая программа на управляемом C++

Начнем с простой программы, написанной на управляемом C++ (Managed C++ - расширение языка C++ для платформы Microsoft.NET):

```
// Первая программа на Managed C++
#using <mscorlib.dll>
using namespace System;
void main()
{
  Console::WriteLine(”C++ Hello, World!”);
}
```

Как можно увидеть, это практически обычная программа на C++ - отличие состоит лишь в том, что вместо директивы препроцессору #include используется конструкция #**using**. С помощью директивы **#using** можно подключать модули, написанные на любом языке программирования, имеющимся в .NET. **#using** подключает модуль, а затем с помощью **using** импортируется нужное пространство имен из этого модуля. Например, пространство имен System, содержащее некоторые базовые системные классы, такие, как Console.

В данной программе методе main() содержит единственный оператор, который является вызовом статического метода WriteLine() класса Console, предназначенным для вывода переданной строки на устройство вывода.

Затем программу необходимо скомпилировать. Сделать это можно как с помощью среды разработки Visual Studio .NET, так и с помощью компилятора командной строки:

cl hello.cpp /CLR /link /entry:main

После выполнения компилятор создаст файл hello.exe, и, если запустить его, CLR загрузит, проверит и выполнит его.

Первая программа на C#

Новый язык программирования C# разработан компанией Microsoft с учетом основных положений технологий Microsoft.Net и вбирает в себя положительный опыт практического использования языков C++ и Java. После своего создания язык C# прошел серьезную проверку, поскольку Microsoft использовал C# для разработки большей части базовых классов и утилит платформы MS.Net..

Рассмотрим пример простой программы на языке C#

```
// Первая программа на C#
using System;
class MainApp
{
  public static void Main()
```

```
  {
    Console.WriteLine(“C# Hello, World!”);
  }
}
```

Код C# отличается от C++ тем, что заголовочных файлов нет совсем – и определение классов и их реализация располагаются в одном файле с расширением .cs. Директива using аналогична по смыслу той же директиве в Managed C++, но при этом нет необходимости указывать файл, где содержится такое пространство имен.

Компиляция программы на C# выполняется при помощи команды:

csc hello.cs

Первая программа на VB.Net

Наконец, приведем примерHello, World на Visual Basic .NET

```
Rem Первая программа на VB.Net
Imports System
Public Module modmain
 Sub Main()
      Console.WriteLine(“VB Hello, World!”)
 End Sub
End Module
```

Как можно увидеть из примера, язык Visual Basic немного изменился и стал похож на C#, сохранив при том все свои отличительные черты (практически каждая строка программы может быть перетранслирована в аналогичную строку на C#).

Получение исполняемого PE-файла может быть выполнено при помощи команды:

vbc /t:exe /out:hello.exe hello.vb

Рассматривая приведенные примеры, можно обнаружить замечательное свойство - на всех трех языках программирования класс Console и его метод WriteLine() остался неизменным. Это означает, что опыт программирования на каком-либо языке может быть использован и при разработке программ на другом языке программирования, переходы между языками для программистов становятся более безболезненными и значительно более простой становится интеграция программ из модулей, разработанных на разных языках программирования.

Преимущества платформы MS.Net

В ходе последовательного развития новых методов, средств и подходов разработки сложного обеспечения регулярно возникали *моменты обобщения и интеграции*, когда появлялись решения, органично вбирающие в себя последние достижения в области науки и практики программирования. Эти знаковые решения (заслуженно иногда называемые

революционными открытиями) приводили к подъему мира программирования на качественно иной уровень состояния дел. Так происходило при разработке языков программирования Pascal и C, создании операционных систем Unix и Windows, отработке принципов объектно-ориентированного программирования и их реализации в языке программирования C++, применении интегрированных и визуальных средств разработки, появлении Интернет и Java, и т.д. Так произошло и при создании новой платформы Microsoft.Net для разработки, развертывания и выполнения сложного программного обеспечения.

Таким образом, первое, что характеризует новое предложение корпорации Microsoft, - это современность используемых в рамках платформы решений. Платформа .Net, наряду с наличием многих замечательных новаторских решений, вбирает в себя самые передовые технологии разработки масштабного ПО. Можно сказать, что для получения MS.Net была выполнена переплавка всей лучшей "руды" информационной индустрии, в результате чего удалось получить надежную современную основу производства и использования сложных программных систем.

Следует отметить и ряд других ключевых моментов, характеризующих значимость появления платформы MS.Net (для понимания приводимых далее характеристик в целом достаточно изложенного ранее материала, для обоснования же отдельных утверждений необходимо более глубокое проникновение в технологию MS.Net):

- **Современные средства разработки** - платформа MS.Net включает в себя как готовые компоненты для построения ПО, так и интегрированную среду разработки, обеспечивающая возможность *многоязыковой разработки* программных систем с использованием разных языков программирования (C#, C++, VBasic.Net, Java# и др.). Как результат, разработчик программ уже не ограничивается выбором одного какого-либо языка программирования, а может варьировать средства разработки с учетом собственного опыта и свойств разрабатываемых программ даже в пределах одной программной системы. Следует отметить также, многие общие части (типы данных, обработка исключений, библиотеки) разных языков программирования являются одинаковыми,
- **Компонентное представление ПО –** MS.Net развивает существующие подходы к основному способу снижения сложности ПО - *компонентному представлению программных систем* - предлагая более простой, удобный и надежный метод формирования программных компонент,

- **Распределенные вычисления** – использование платформы MS.Net в значительной степени снижает сложность современной формы разработки ПО в виде *распределенных программных систем* или клиент-серверных приложений,
- **Интернет технологии** - платформа MS.Net содержит большинство существующих *Интернет технологий*, обеспечивающую возможность быстрой разработки как обычных Web-приложений, так и Web-сервисов, выступающих как доступные через Интернет "строительные блоки" современного сервис-ориентированного программного обеспечения и др.

Подводя итог, можно сказать, что в мире программирования после появления технологии MS.Net появилась новая знаковая отметка – индустрия разработки ПО до и после появления MS.Net.

Дополнительная информация по общей характеристике платформы MS.Net может быть получена в большом ряде изданий как начального (см., например [1-2]), так и профессионального [3-4] уровней рассмотрения.

Литература

1. Пономарев В. Программирование на C++/C# в Visual Studio .NET 2003. – СПб.: БХВ-Петербург, 2004.
2. Тай Т., Хонг К. Лем Платворма .NET. Основы, 2-е издание. – СПб.: Символ-Плюс, 2003.
3. Рихтер Д. Программирование на платформе Microsoft .NET Framework. – М.: "Русская Редакция", 2002.
4. Просиз Д. Программирование для Microsoft .NET – М.: "Русская Редакция", 2003.