Системное программирование для Microsoft.NET на языке C#
Лекция 1

Оцоков Шамиль Алиевич,
Московский энергетический институт,
Email: otsokovShA@mpei.ru

Литература

1. Ali Asad, Hamza Ali. The C# Programmer’s Study Guide (MCSD), 2017.
2. Джуст Виссер. Разработка обслуживаемых программ на языке C#, 2017
3. Зиборов В. В. Visual C# 2010 на примерах. − СПб.: БХВ-Петербург, 2011. – 432 с.
4. Svetlin Nakov. Fundamentals of computer programming with C#., 2013
6. **https://stepik.org/course/3944/promo#toc**

Некоторые виды современного программного обеспечения

- *Клиент-серверные системы*
- *Internet (Web) - приложения*
- *Интегрированные решения*
- *Встроенные системы*
- *Программное обеспечение мобильных устройств*

Характерные черты современных программных систем

- Ориентированность на использование в Internet и WWW
- Универсальное представление программных моделей (UML) и данных (XML)
- Повышенные требования к безопасности и надежности (TWC – Trustworthy Computing)
- Интеграция различных языков программировани, инструментальных средств, баз данных и знаний, в *единую инфраструктуру*
- Проектирование и разработка программных компонент многократного использования (reusable)

Инициатива безопасных и надежных вычислений (Trustworthy Computing Initiative) - Microsoft, 2002

- *Безопасность (Security)* – Пользователь может ожидать, что система устойчива к атакам, и что конфиденциальность, целостность и доступность системы и ее данные защищены
- *Конфиденциальность (Privacy)* – Пользователь может управлять использованием системой своих личных данных, и эти данные используются, в соответствии с принципами корректности
- *Надежность (Reliability)* – Пользователь может быть уверен в том, что продукт выполнит требуемые от него функцииd
- *Оперативность и корректность бизнеса (Business integrity)* – Разработчик продукта оперативно реагирует на обращения пользователей и ведет себя со всей ответственностью
- *Основная идея: Требования TWC необходимо иметь в виду постоянно в ходе разработки и сопровождения продукта*

Современные платформы для разработки программ

- **Java (*Sun Microsystems, 1995*) – платформа для разработки программ на объектно-ориентированном языке *Java*, программы на котором компилируются в Java байт-код (до сих пор имеет статус ведомственного стандарта Sun)**
- **.NET (*Microsoft, 2000*) – *многоязыковая* объектно-ориентированная платформа для разработки программ с общим промежуточным языком (CIL), общей инфтаструктурой языков (CLI) и единым представлением данных на основе XML (стандарты *ISO/ ECMA)*. Язык C# - наиболее удобный язык программирования для .NET, *но не единственный и не обязательный для использования***
- **Обе платформы уделяют особое внимание надежности и безопасности, на основе *исполнения управляемого кода* и *динамического контроля топов.* И Java, и .NET – наиболее безопасные платформы**

Инструменты Microsoft .NET

- ***Microsoft.NET Framework SDK*** **– базовый инструментарий, используемый в режиме командной строки, бесплатно доступный через Web. Входит в состав новых версий Windows 2003, Windows Vista.**
- ***Microsoft Visual Studio .NET*** **(другие названия -** ***Microsoft Visual Studio 7.0 (.NET 1.0),*** **8.0** ***(.NET 2.0*****) – мощная интегрированная среда (IDE) для .NET; включает в себя реализации языков** ***C#.NET, VB.NET, C++.NET, JScript.NET и J#.NET***
- **Всего в данный момент более 30 языков уже реализованы для .NET, включая** ***Oberon и Eiffel*** **- объектно-ориентированные языки;** ***ML – язык функционального программирования; Python и многие другие*****.**

Основные идеи и принципы .NET

- **Компилятор с любого языка (например, C# или VB) транслирует исходный код приложения в общий для всех входных языков формат - *CIL (Common Intermediate Language; другое название - MSIL – Microsoft Intermediate Language);***
- **Кроме CIL-кода, компилятор также генерирует *метаданные* – общую для всех языков информацию о типах и других именованных сущностях (классах, методах, полях и др.), определенных и использованных в данном приложении;**
- **CIL + метаданные + манифест = *сборка*, состоящая из одного или более PE (*Portable Executable)* файлов**
- **Во время выполнения CIL-код *динамически компилируется (“на лету” - on-the-fly) в объектный код конкретной целевой платформы* (x86, Macintosh, IA-64, и др.) с помощью *Just-in-Time (JIT) - компилятора***
- **Все данные, передаваемые через Internet/Intranet, а также все конфигурационные файлы представлены в формате XML**

Преимущества подхода .NET

- .NET Portable Executable (PE) – файлы, генерируемые компиляторами NET, и данные в XML-формате могут быть переданы через Internet (например, по протоколу *HTTP*) и JIT-компилированы или интерпретированы на клиентских компьютерах
- Использование метаданных во время выполнения обеспечивает *исполнение управляемого кода* (*managed code execution)*, с *динамическим контролем типов, проверками безопаснности во время выполнения и т.д.*
- Программист может разработать свое приложение на любом удобном ему языке, реализованном для .NET, и включить в приложение любые модули, написанные на других языках, также реализованных для .NET
- Повышенная степень безопасности: ролевая безопасность (role-based security), безопасность доступа к коду (code access security), безопасность на основе информации о сборках (evidence-based security), “прогулка по стеку” (security stack walk)
- XML Web-сервисы и ASP.NET – особенно важны для современного программирования; только для их одного использования стоит изучить .NET

Основные концепции и аббревиатуры .NET (1/2)

- *.NET* – общая инфраструктура языков, соответствующая стандартам ISO/ECMA; например: Microsoft.NET (работающая в среде Windows 2000/XP/2003/Vista); SSCLI (Rotor) работающая для Windows 2000/XP/2003, UNIX FreeBSD, MacOS)
- *CLI* (Common Language Infrastructure) – ECMA-стандарт на общую инфраструктуру языков .NET
- *CTS* (Common Type System) – ECMA-стандарт на общую систему типов .NET
- *CLR* (Common Language Runtime) – ECMA-стандарт на общее окружение времени выполнения .NET - библиотека, состоящая из нескольких тысяч классов
- *CIL* – ECMA- стандарт на единый промежуточный язык .NET, основанный на постфиксной записи

  NB: В документации по Microsoft Phoenix "CIL" означает промежуточный язык компилятора Visual C++ !
- *VES* (Virtual Execution System) – виртуальная машина CLR, используемая для исполнения управляемого кода
- *COR* (Common Object Runtime) – предшественник .NET. Данное сокращение до сих пор используется в именах системных файлов .NET, например, *msCORee.dll* – главная DLL .NET

Основные концепции и аббревиатуры .NET (2/2)

- *Метаданные* – единое (табличное) представление информации о типах и других именованных сущностях, определелеенных и используемых в .NET-приложении. По структуре очень близки к реляционной СУБД
- *Манифест* – содержимое (инвентарный список) сборки, в частности, включающей PE-файл, представляющий данную сборку
- *PE* (Portable Executable) – файл переносимого исполняемого кода, содержащий CIL-код, метаданные, ресурсы и манифест
- *Сборка* (*Assembly)* – представление программы в форме PE-файла или набора PE-файлов, генерируемых компилятором .NET

Сравнение платформ .NET и Java

- .NET – *многоязыковое программирование*; Java – программирование *только на Java* (модули, написанные *не* на Java, играют подчиненную, зависимую роль – "*платформно-зависимые*" (*native) методы, написанные на* C/C++)
- .NET – *универсальная система типов и универсальное представление данных*; в Java-технологии все ориентировано *только на язык Java*
- Языки Java и C#: авторы C# взяли все лучшее из Java, но, в дополнение, C# содержит еще целый ряд удобных возможностей, которых нет в Java: свойства (properties) – будут введены только в Java 7; индексаторы (indexers), атрибуты (attributes) – в Java только в версии 5 введены *аннотации;*, синонимы (aliases), дополнительные виды операторов и др.
- В свою очередь, Java взяла многое от .NET: boxing/unboxing. Java Web-сервисы и т.д. (*обратное влияние .NET на Java)*
- *Резюме*: обе платформы весьма интересны и заслуживают изучения, но платформа .NET – более общая, следовательно, по-видимому, более перспективная. Хотя Java до сих пор широко используется для мобильных телефонов (JME) и распределенных решений (JEE)
- ***".NET – платформа следующего десятилетия"; ".NET – наиболее надежная и безопасная платформа"* (Microsoft)**
- **Java 6: *Tango* (элемент платформы JEE) – поддержка "сосуществования" .NET CLR и JVM в одном приложении**
- **Фактически Java и .NET до сих пор конкурируют, хотя, выражаясь официально, Sun и Microsoft в 2004 г. заключили соглашение на 10 лет в области интеллектуальной собственности**

Этапы разработки программного обеспечения

1. Анализ бизнес-требований и разработка технического задания
2. Разработка архитектуры программы
3. Разработка программы
4. Тестирование
5. Ввод в эксплуатацию программы у заказчика
6. Техническая поддержка

Этапы разработки программного обеспечения

Разработка программного обеспечения - это больше, чем просто кодирование. Как мы видели, разработка программного обеспечения - это гораздо больше, чем просто кодирование (написание кода), и она включает в себя ряд других процессов, таких как анализ требований, проектирование, планирование, тестирование и поддержка, которые требуют самых разных специалистов, называемых инженерами-программистами.

Программирование - это всего лишь небольшая, но очень важная часть разработки программного обеспечения.

Первая программа

```
class HelloCSharp
{
static void Main(string[] args)
{
System.Console.WriteLine("Hello C#!");
}
}
```

Чисто объектно-ориентированный язык программирования

Понятие класса и объекта

Регистрозависимый язык программирования

Оформление текста программы

```
class HelloCSharp
{
   static void Main()
   {
      System.Console.WriteLine("Hello C#!");
   }
}
```

Основные правила форматирования

Если мы хотим, чтобы наш код был правильно отформатирован, мы должны соблюдать несколько важных правил относительно отступов:

- Методы имеют отступ внутри определения класса (переместитесь вправо на один или несколько символов [Tab]);
- Содержимое метода находится внутри определения метода с отступом;
- Открывающая фигурная скобка {должна находиться на отдельной строке и находиться точно под методом или классом, к которому она относится;
- Закрывающая фигурная скобка} должна находиться на отдельной строке, расположенной строго вертикально под соответствующей открывающей скобкой (с таким же отступом);
- Все названия классов должны начинаться с заглавной буквы;
- Имена переменных должны начинаться с строчной буквы;
- Имена методов должны начинаться с заглавной буквы;

Основные правила форматирования

Имена файлов соответствуют именам классов

Каждая программа на C # состоит из одного или нескольких определений классов. Принято, что каждый класс определяется в отдельном файле с именем, соответствующим имени класса, и с расширением .cs. Когда эти требования не выполняются, программа будет работать, но с навигацией по коду.

C # и Java

Первая версия C # была разработана Microsoft между 1999 и 2002 годами и официально выпущена для широкой публики в 2002 году как часть платформы .NET. Платформа .NET призвана упростить разработку программного обеспечения для Windows, обеспечивая новый качественный подход к программированию, основанный на концепциях «виртуальной машины» и «управляемого кода». В то время язык и платформа Java достигли огромного успеха во всех областях разработки программного обеспечения; C # и .NET были естественным ответом Microsoft на технологию Java.

Язык C

C # - это современный объектно-ориентированный язык программирования высокого уровня общего назначения для программирования. Его синтаксис аналогичен синтаксису C и C ++, но многие функции этих языков не поддерживаются в C #, чтобы упростить язык, что упрощает программирование.

Программы на C # состоят из одного или нескольких файлов с расширением .cs, которые содержат определения классов и других типов. Эти файлы компилируются компилятором C # (csc) в исполняемый код, и в результате создаются сборки, которые представляют собой файлы с тем же именем, но с другим расширением (.exe или .dll). Например, если мы скомпилируем HelloCSharp.cs, мы получим файл с именем HelloCSharp.exe

Некоторые ключевые слова

abstract	as	base	bool	break	byte
case	catch	char	checked	class	const
continue	decimal	default	delegate	do	double
else	enum	event	explicit	extern	false
finally	fixed	float	for	foreach	goto
if	implicit	in	int	interface	internal
is	lock	long	namespace	new	null

Некоторые ключевые слова

object	operator	out	override	params	private
protected	public	readonly	ref	return	sbyte
sealed	short	sizeof	stackalloc	static	string
struct	switch	this	throw	true	try
typeof	uint	ulong	unchecked	unsafe	ushort
using	virtual	void	volatile	while	

Автоматическое управление памятью

Одно из самых больших преимуществ .NET Framework - встроенное автоматическое управление памятью. Он защищает программистов от сложной задачи ручного выделения памяти для объектов и последующего ожидания подходящего момента для ее освобождения. Это значительно увеличивает продуктивность разработчика и качество программ, написанных на C #.

В .NET Framework есть специальный компонент среды CLR, отвечающий за управление памятью. Это называется «сборщиком мусора» (автоматизированная система очистки памяти). Сборщик мусора выполняет следующие основные задачи: проверять, когда выделенная память для переменных больше не используется, освобождать ее и делать доступной для размещения новых объектов.

Автоматическое управление памятью

Важно отметить, что не совсем ясно, в какой момент память очищается от неиспользуемых объектов (например, локальных переменных). Согласно спецификациям языка C #, это происходит в какой-то момент после того, как заданная переменная выходит за пределы области видимости, но не указано, происходит ли это мгновенно, через некоторое время или когда доступной памяти становится недостаточно для нормальной работы программы.

Независимость от окружающей среды и языка программирования

Одним из преимуществ .NET является то, что программисты, использующие разные языки .NET, могут легко обмениваться своим кодом. Например, программист на C # может использовать код, написанный другим программистом на VB.NET, Managed C ++ или F #. Это возможно, потому что программы, написанные на разных .NET.
языки имеют общую систему типов данных, инфраструктуру выполнения и единый формат скомпилированного кода (сборок).
Большим преимуществом технологии .NET является возможность запускать код, который написан и скомпилирован только один раз, на разных операционных системах и аппаратных устройствах. Мы можем скомпилировать программу на C # в среде Windows, а затем запустить ее в Windows, Windows Mobile, Windows RT или Linux.

Промежуточный язык Microsoft (MSIL)

Промежуточный язык Microsoft (MSIL)

Идея независимости от окружения была заложена на самых ранних этапах создания платформы .NET и реализуется с помощью небольшого трюка. Выходной код не компилируется в инструкции для конкретного микропроцессора и не использует функции конкретной операционной системы; он скомпилирован на так называемом промежуточном языке Microsoft (MSIL). Этот MSIL не выполняется напрямую микропроцессором, а из виртуальной среды, называемой Common Language Runtime (CLR).

Common Language Runtime (CLR)

Common Language Runtime (CLR) - сердце .NET

В самом центре платформы .NET бьется ее сердце - Common Language Runtime (CLR) - среда, которая контролирует выполнение управляемого кода (кода MSIL). Он обеспечивает выполнение программ .NET на различных аппаратных платформах и операционных системах.

CLR - абстрактная вычислительная машина (виртуальная машина). Подобно физическим компьютерам, он поддерживает набор инструкций, реестров, доступа к памяти и операций ввода-вывода. CLR обеспечивает контролируемое выполнение программ .NET с использованием всех возможностей процессора и операционной системы. CLR также выполняет управляемый доступ к памяти и другим ресурсам компьютера, соблюдая правила доступа, установленные при выполнении программы.

Платформа .NET

Платформа .NET содержит язык C #, среду CLR и множество вспомогательных инструментов и библиотек, готовых к использованию. В зависимости от целевой группы пользователей существует несколько версий .NET:

- .NET Framework - самая распространенная версия среды .NET из-за ее общего назначения. Он используется при разработке консольных приложений, приложений Windows с графическим пользовательским интерфейсом, веб-приложений и многого другого.

- .NET Compact Framework (CF) - это «облегченная» версия стандартной .NET Framework, которая используется при разработке приложений для мобильных телефонов и других КПК, использующих Windows Mobile Edition.

- Silverlight также является «облегченной» версией .NET Framework, предназначенной для выполнения в веб-браузерах с целью реализации мультимедийных и многофункциональных Интернет-приложений.

- .NET для приложений Магазина Windows - это подмножество .NET Framework, предназначенное для разработки и выполнения приложений .NET в среде Windows 8 и Windows RT (так называемые приложения Магазина Windows).

Программные технологии

Программные технологии - это наборы классов, модулей, библиотек, моделей программирования, инструментов, шаблонов и передовых методов решения некоторых конкретных проблем в разработке программного обеспечения. Существуют общие программные технологии, такие как веб-технологии, мобильные технологии, технологии компьютерной графики и технологии, связанные с некоторыми платформами, такими как .NET или Java.

Существует множество технологий .NET, которые используются в различных областях разработки .NET. Типичными примерами являются веб-технологии (такие как ASP.NET и ASP.NET MVC), позволяющие быстро и легко создавать динамические веб-приложения, и мобильные технологии .NET (например, WinJS), которые позволяют создавать мультимедийные приложения с широким пользовательским интерфейсом, работающие над интернет.

Интерфейс прикладного программирования (API)

Каждая библиотека или технология .NET используются путем создания объектов и вызова их методов. Набор общедоступных классов и методов в библиотеках программирования называется интерфейсом прикладного программирования или просто API. В качестве примера мы можем взглянуть на сам .NET API; это набор библиотек классов .NET, расширяющих возможности языка и добавляющих функциональность высокого уровня. Все технологии .NET предлагают общедоступный API. Технологии часто называют просто API, что добавляет определенные функции. Например: API для работы с файлами, API для работы с диаграммами, API для работы с принтерами, API для чтения и создания документов Word и Excel, API для создания документов PDF, API веб-разработки и т. д.

Документация NET

Очень часто необходимо задокументировать API, потому что он содержит много пространств имен и классов. Классы содержат методы и параметры. Их цель не всегда очевидна и требует пояснений. Между отдельными классами также существуют внутренние зависимости, которые необходимо объяснить для правильного использования. Эти объяснения и технические инструкции по использованию данной технологии, библиотеки или API называются документацией. Документация состоит из набора документов с техническим содержанием.

.NET Framework также имеет документацию, официально разработанную и поддерживаемую Microsoft. Он общедоступен в Интернете, а также распространяется вместе с платформой .NET в виде набора документов и инструментов для просмотра и поиска.

Домашнее задание к лекции 1

- Скачайте и инсталлируйте на Visual Studio
- Создайте простейшее консольное приложение и приложение Windows Forms.
- Дайте сравнительную оценку Java и Microsoft.NET
- Добавить себя в группу вконтакте «Системное программирование 2020»:
  club198345138