



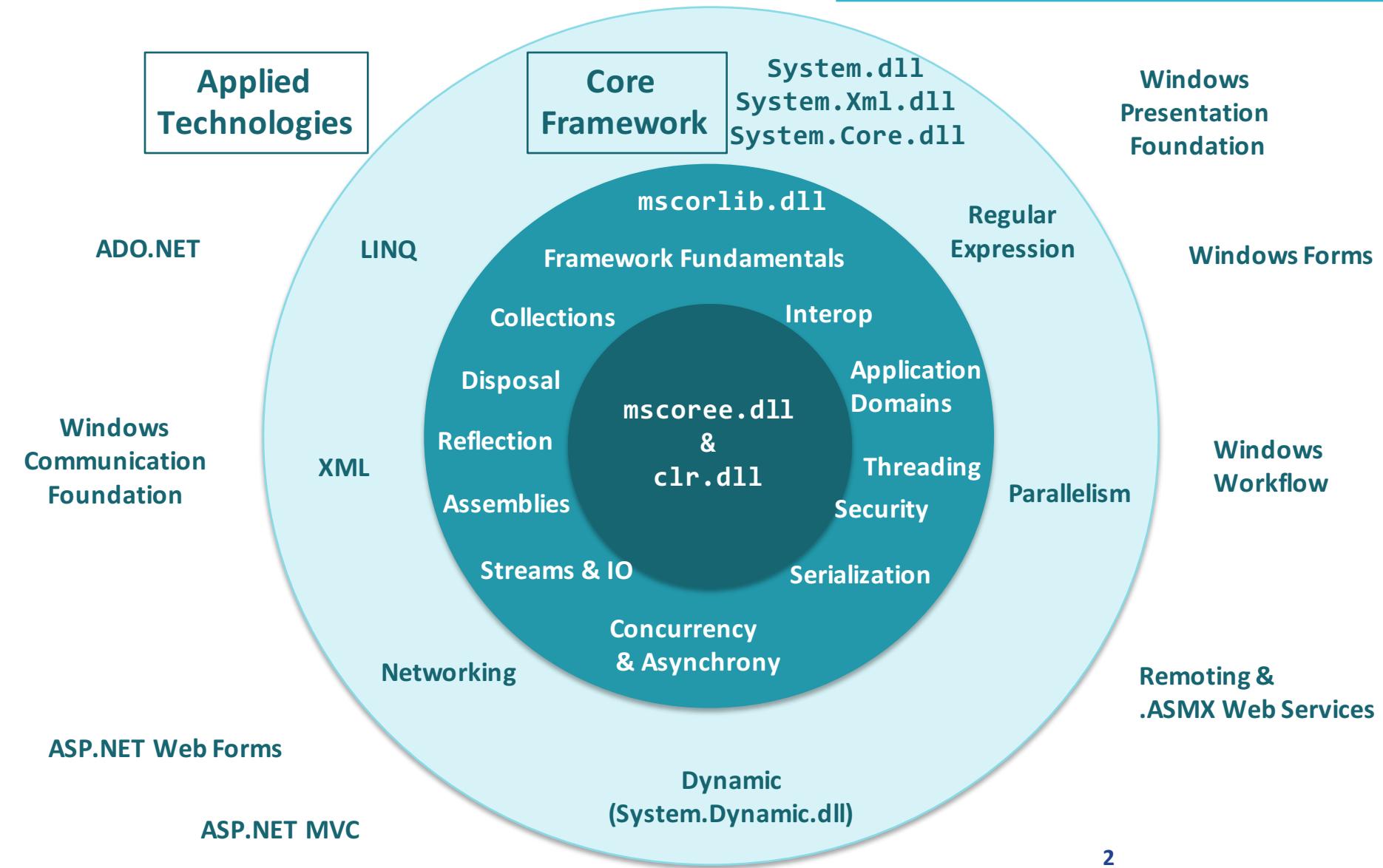
ВВЕДЕНИЕ В C# И .NET FRAMEWORK

.NET & JS LAB, MINSK

Анжелика КРАВЧУК

Платформа .NET Framework

<http://referencesource.microsoft.com/>



Платформа .NET Framework

- Microsoft VB .NET
- Microsoft VC++ .NET
- Microsoft C#
- Microsoft J#
- Microsoft Jscript
- APL
- ASNA Visual RPG.NET
- Fujitsu COBOL
- Micro Focus Cobol NetExpress
- F# (a mixed functional/imperative language based on Caml from Microsoft Research)
- Eiffel
- Delta Forth
- Lahey/Fujitsu Fortran for .NET
- Glasgow Haskell ...

Платформа .NET Framework

C# представляет собой унифицированный, безопасный к типам, объектно-ориентированный язык, не зависящий от платформы, но написанный для эффективной работы с платформой .NET Framework

Основные особенности языка

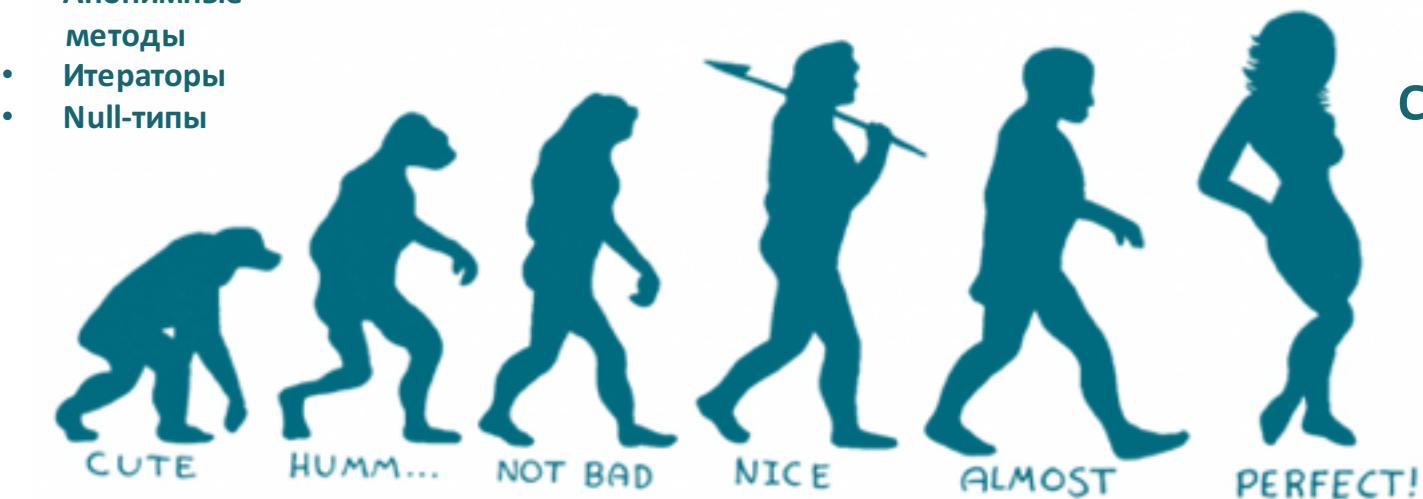
1. Объектная ориентация, унифицированная и расширенная (формальные синтаксические конструкции для классов, интерфейсов, структур, перечислений и делегатов) система типов; свойства, события
2. Безопасность в отношении типов, поддержка статической типизации
3. Управление памятью
4. Поддержка платформ

Эволюция C#, CLR и .NET Framework

Версия C#	Версия CLR	Версия Framework
1.0	1.0	1.0
1.2	1.1	1.1
2.0	2.0	2.0, 3.0
3.0	2.0 (SP1)	3.5
4.0	4.0	4.0
5.0	4.5 (Patched 4.0)	4.5 (including 4.5.1 и 4.5.2)
6.0	4.5 (Patched 4.0)	4.6

Эволюция языка C#

C# 1.0
Управляемый
код



- C# 2.0**
 - Обобщения
 - Смешанные типы
 - Анонимные методы
 - Итераторы
 - Null-типы
- C# 3.0**
 - Неявно типизируемые локальные переменные
 - Инициализаторы объектов и коллекций
 - Автоматическая реализация свойств
 - Анонимные типы
 - Методы расширения
 - Запросы
 - Лямбда-выражения
 - Деревья выражений

C# 5.0
Асинхронные
функции

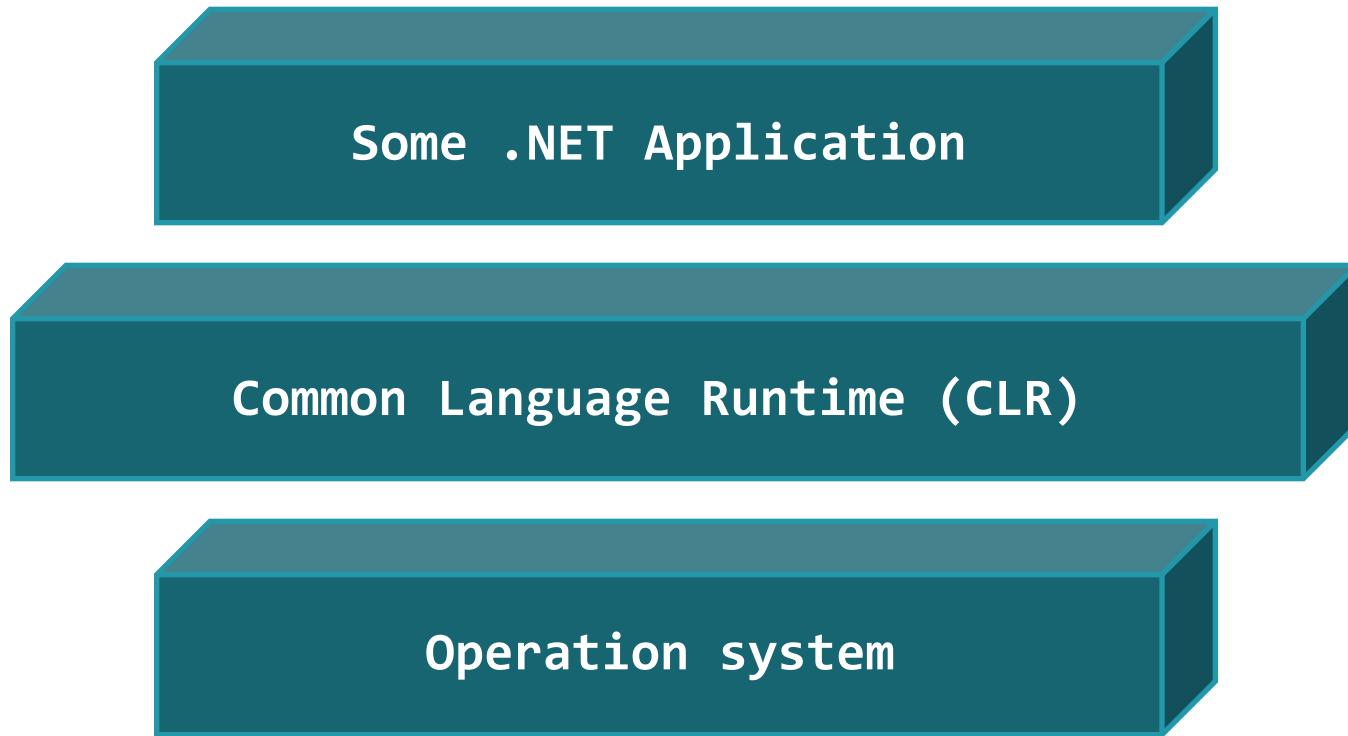
- C# 4.0**
 - Динамическое связывание
 - Именованные и дополнительные аргументы
 - Обобщенная ковариантность и контравариантность

- C# 6.0**
- Getter-only auto-properties
 - Auto-property initializers
 - Expression-bodied members
 - Null-conditional operators
 - Using static members
 - Index initializers
 - String interpolation
 - nameof operator
 - Await in catch/finally
 - Exception filters
 - Extension Add in collection initializers
 - Improved overload resolution

C# 7.0?

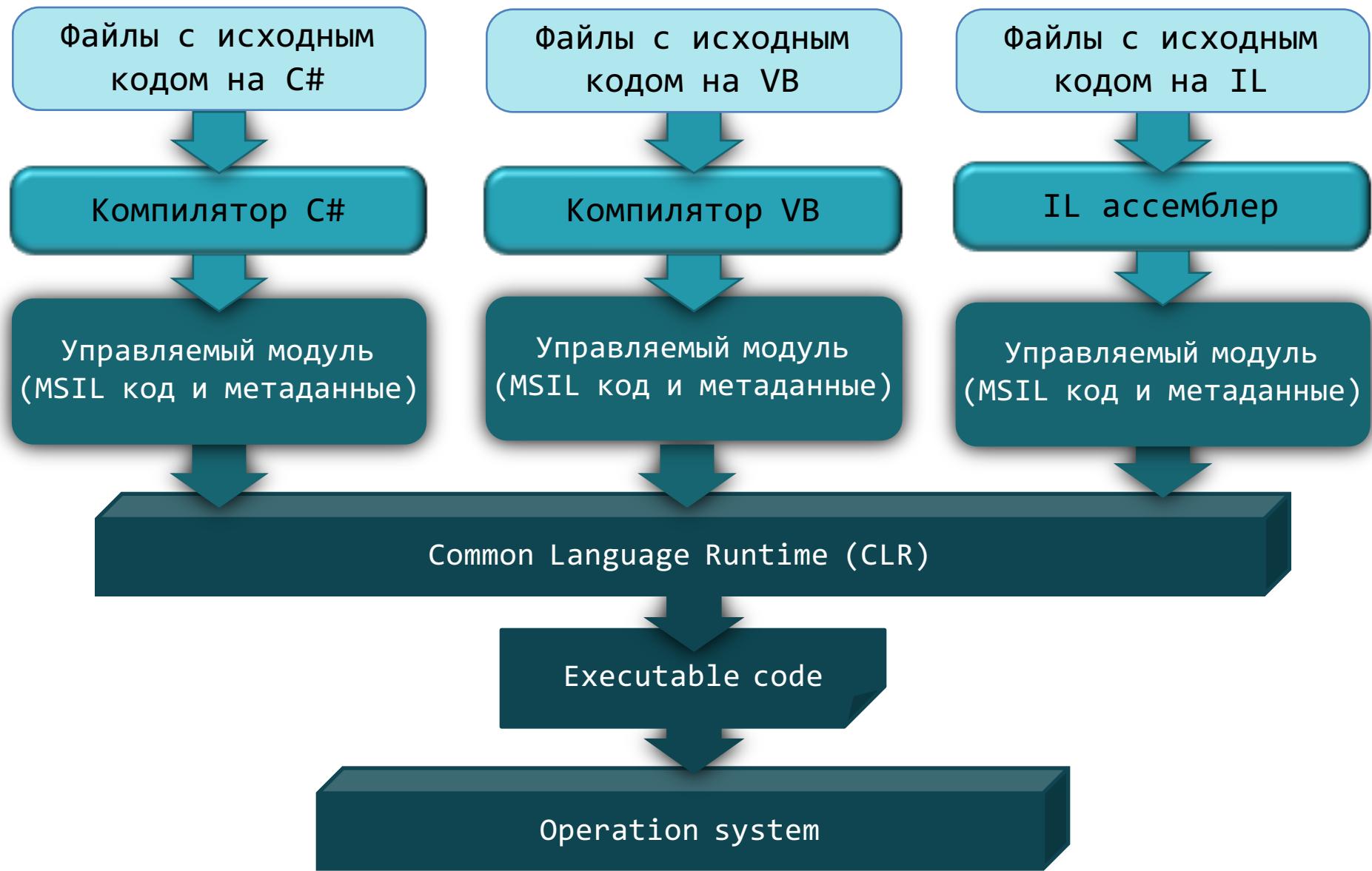
<http://www.ecma-international.org/publications/standards/Ecma-334.htm>

Common Language Runtime

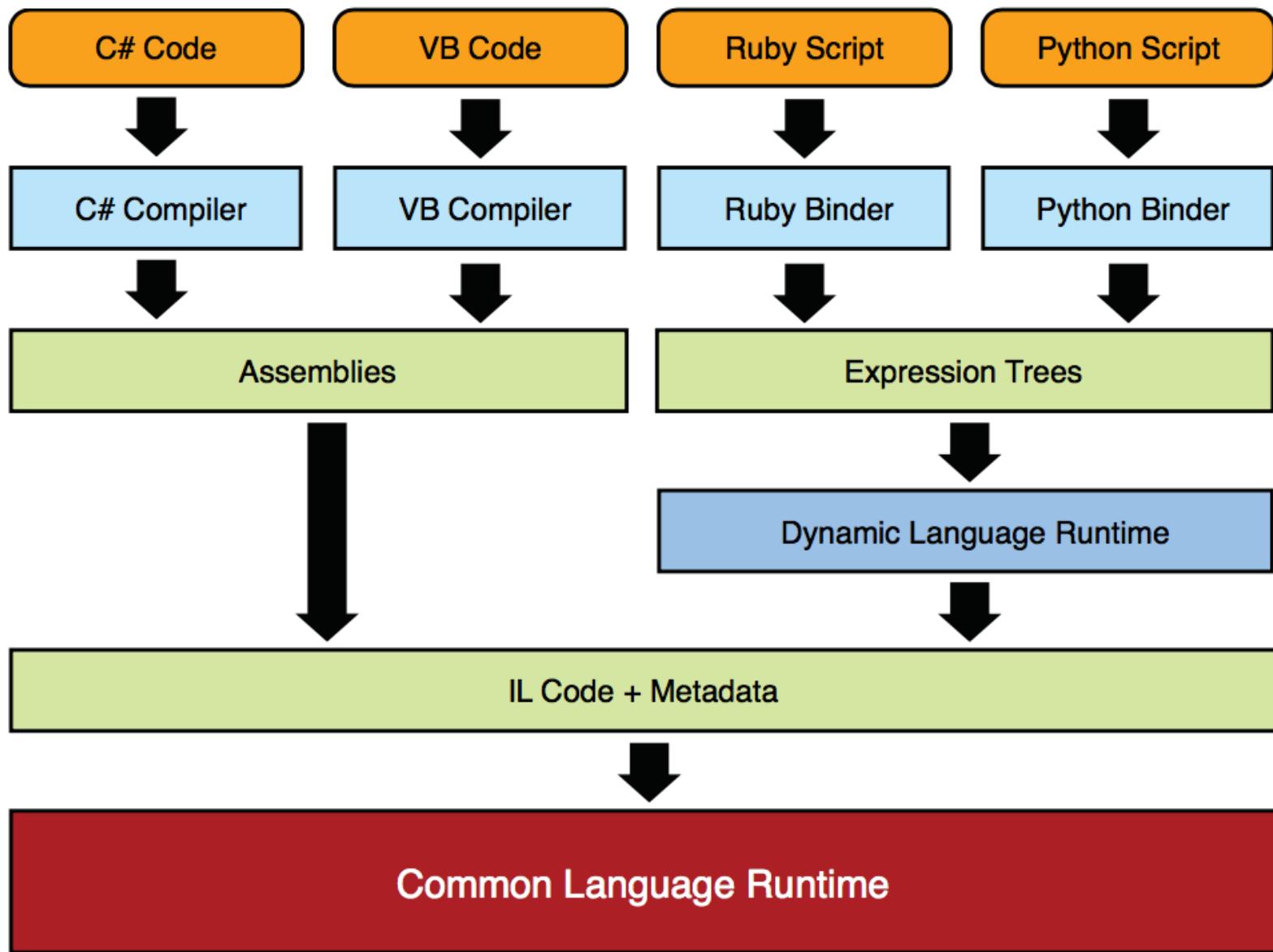


Общеязыковая среда выполнения (Common Language Runtime, CLR) – среда выполнения, которая подходит для разных языков программирования. Основные возможности CLR (управление памятью, загрузка сборок, безопасность, обработка исключений, синхронизация) доступны в любых языках программирования, использующих эту среду.

Управляемые модули, MSIL код и метаданные



Управляемые модули, MSIL код и метаданные



Управляемые модули, MSIL код и метаданные

Метаданные устраняют необходимость в заголовочных и библиотечных файлах при компиляции

Visual Studio .NET использует метаданные для облегчения написания кода

В процессе верификации кода CLR использует метаданные, чтобы убедиться, что код совершает только «безопасные» операции

Метаданные позволяют сериализовать поля объекта в блок памяти на удаленной машине и затем десериализовать, восстановив объект и его состояние на этой машине

Метаданные позволяют сборщику мусора отслеживать жизненный цикл объектов

IL и верификация

IL является стековым языком – все его инструкции заносят операнды в стек вычислений (evaluation stack) и извлекают результаты из стека. IL не содержит инструкций для работы с регистрами (абстрагирует разработчика от конкретного процессора), что упрощает создание новых языков и компиляторов, генерирующих код для CLR.

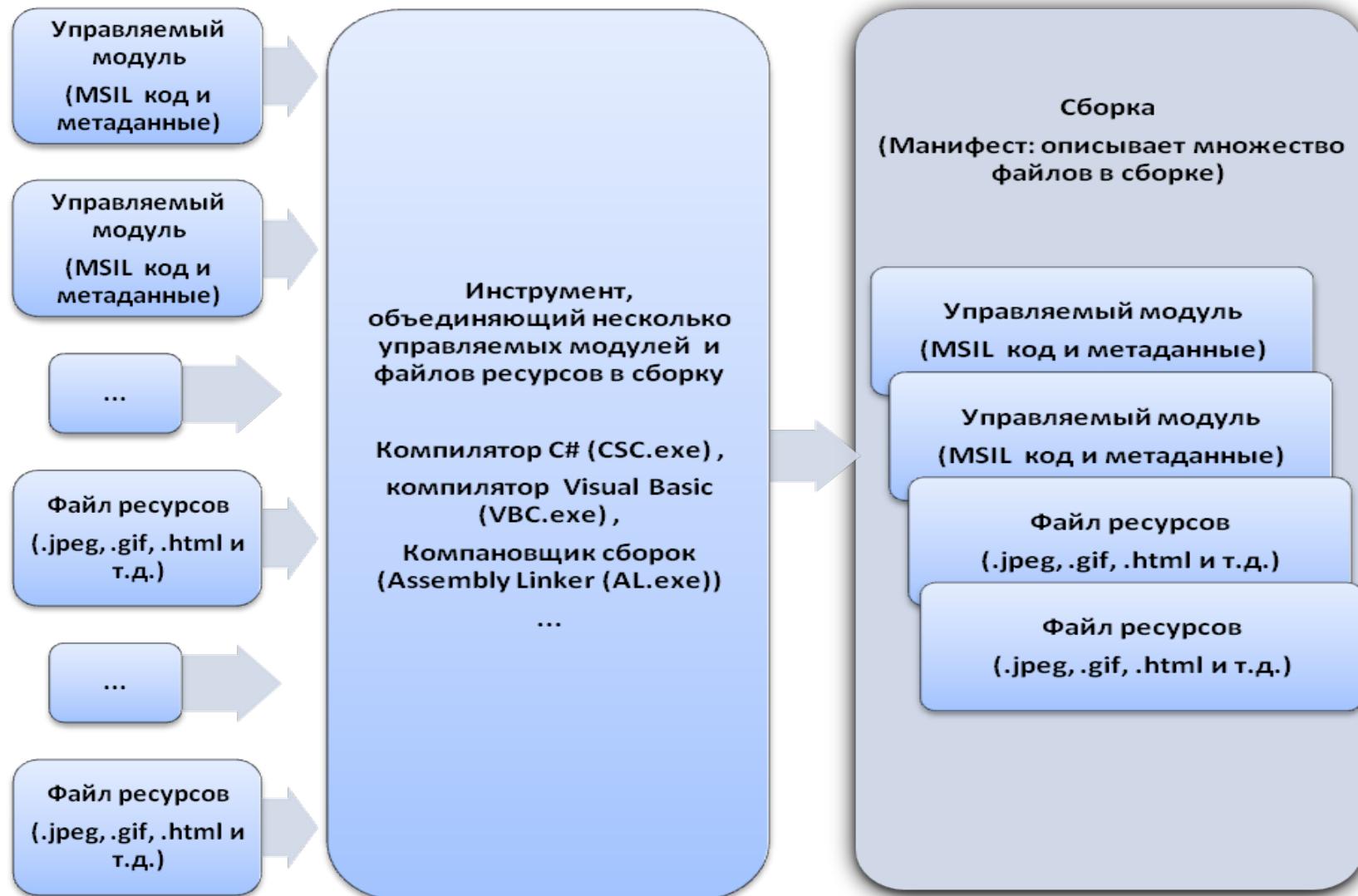
IL-код обеспечивает безопасность приложения и его устойчивость перед ошибками – в процессе компиляции IL в машинные инструкции CLR выполняется процедура, называемая *верификацией* — анализ высокоуровневого кода IL и проверка безопасности всех операций.

Сборки в .NET

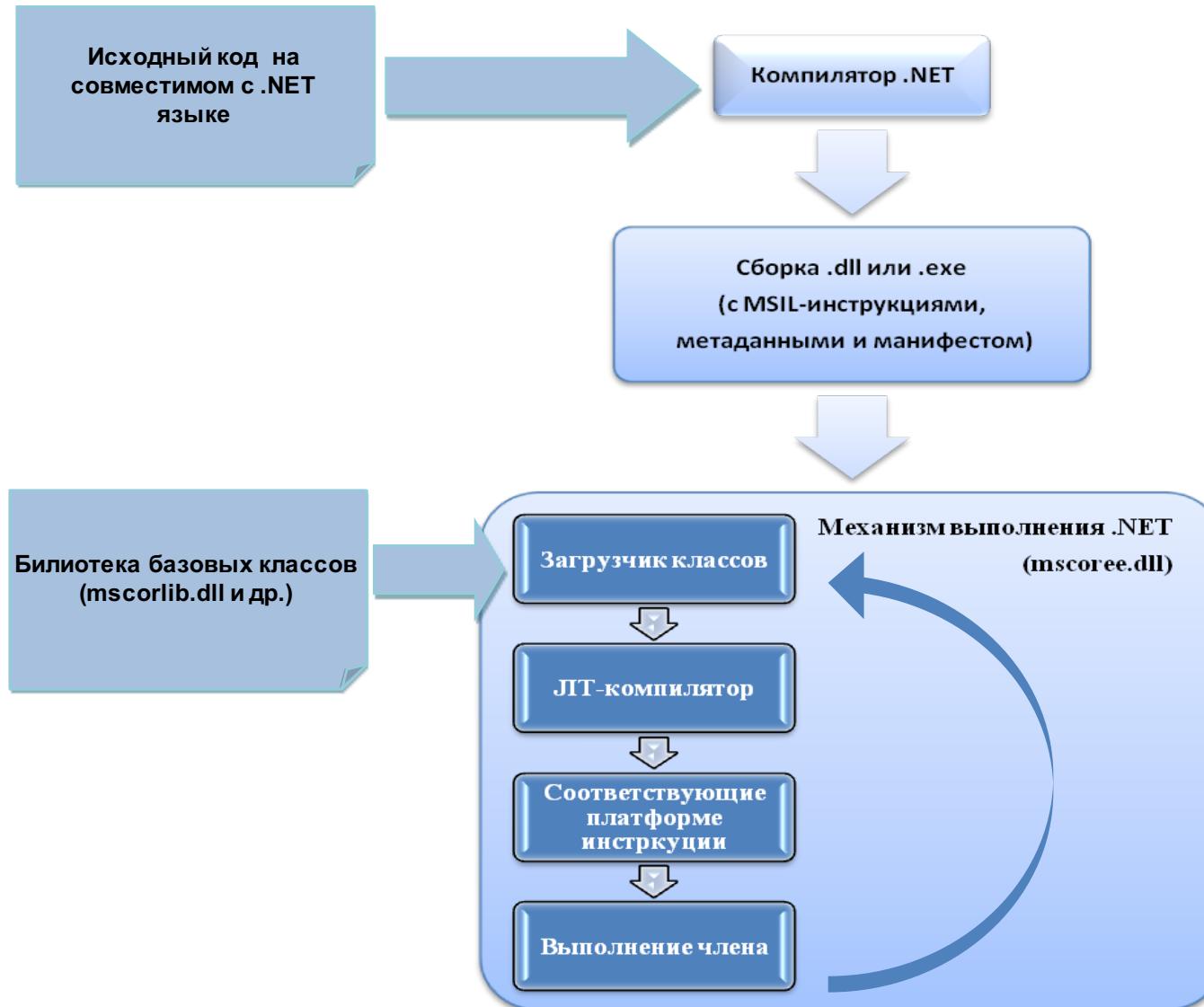
Логическая группировка одного или нескольких управляемых модулей и файлов ресурсов

Самая маленькая единица с точки зрения повторного использования, безопасности и управления версиями

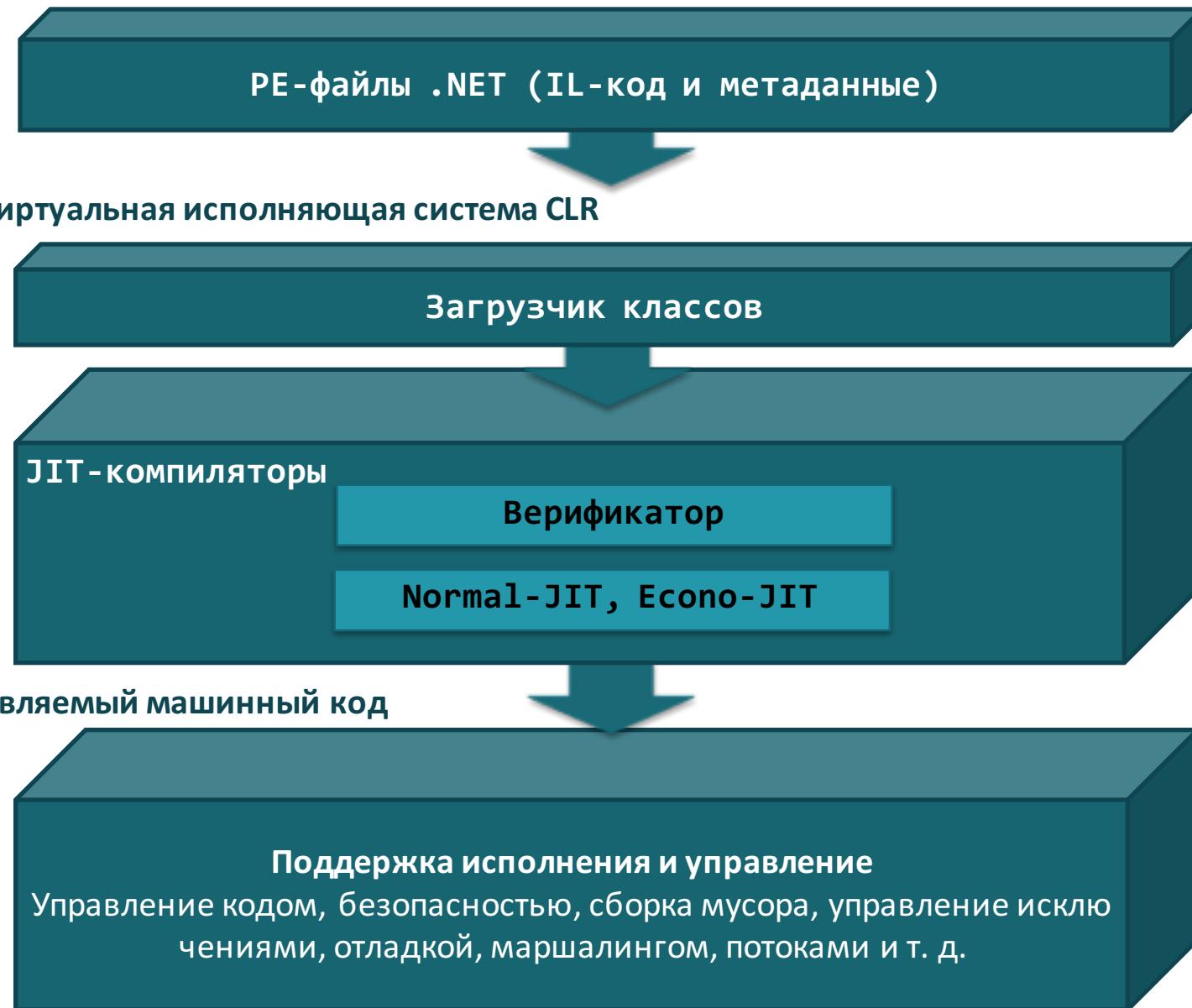
Сборки в .NET



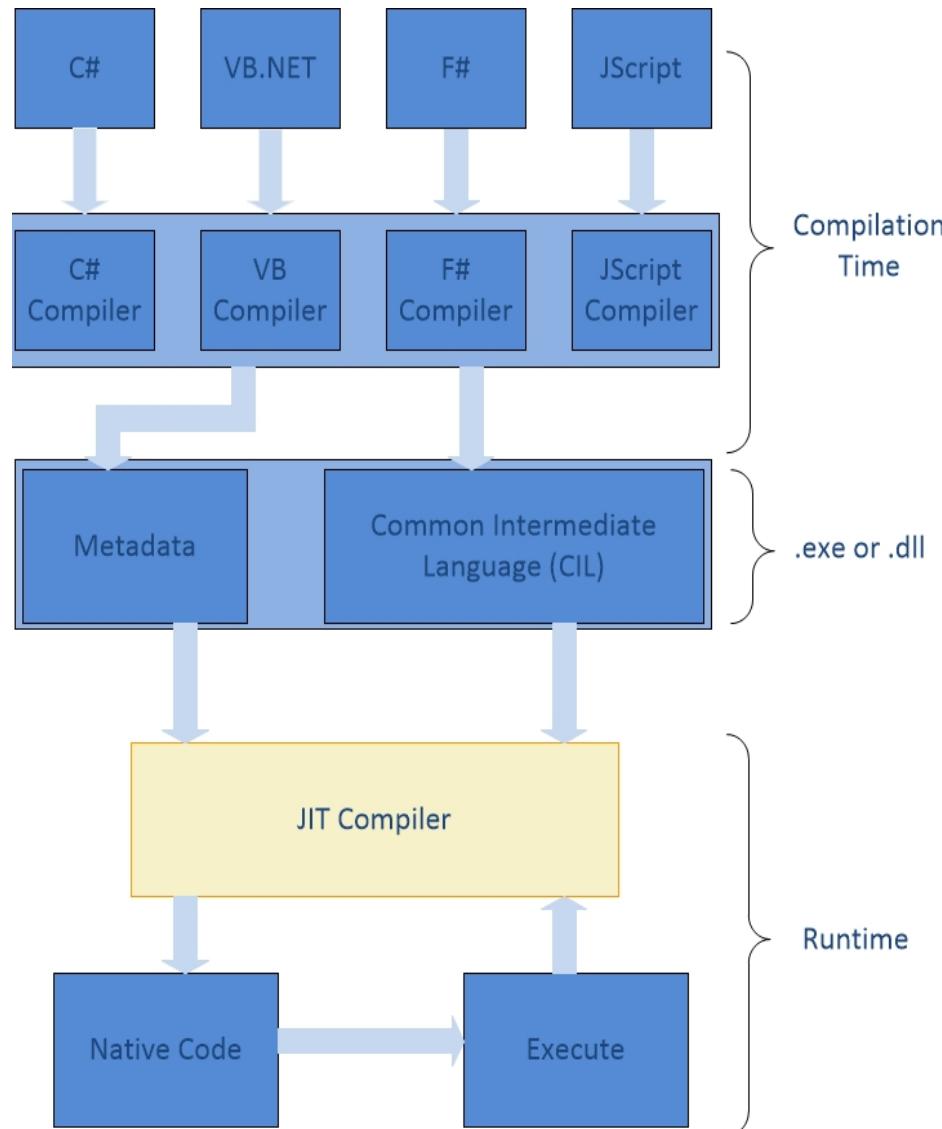
Как CLR загружает, компилирует и запускает сборки



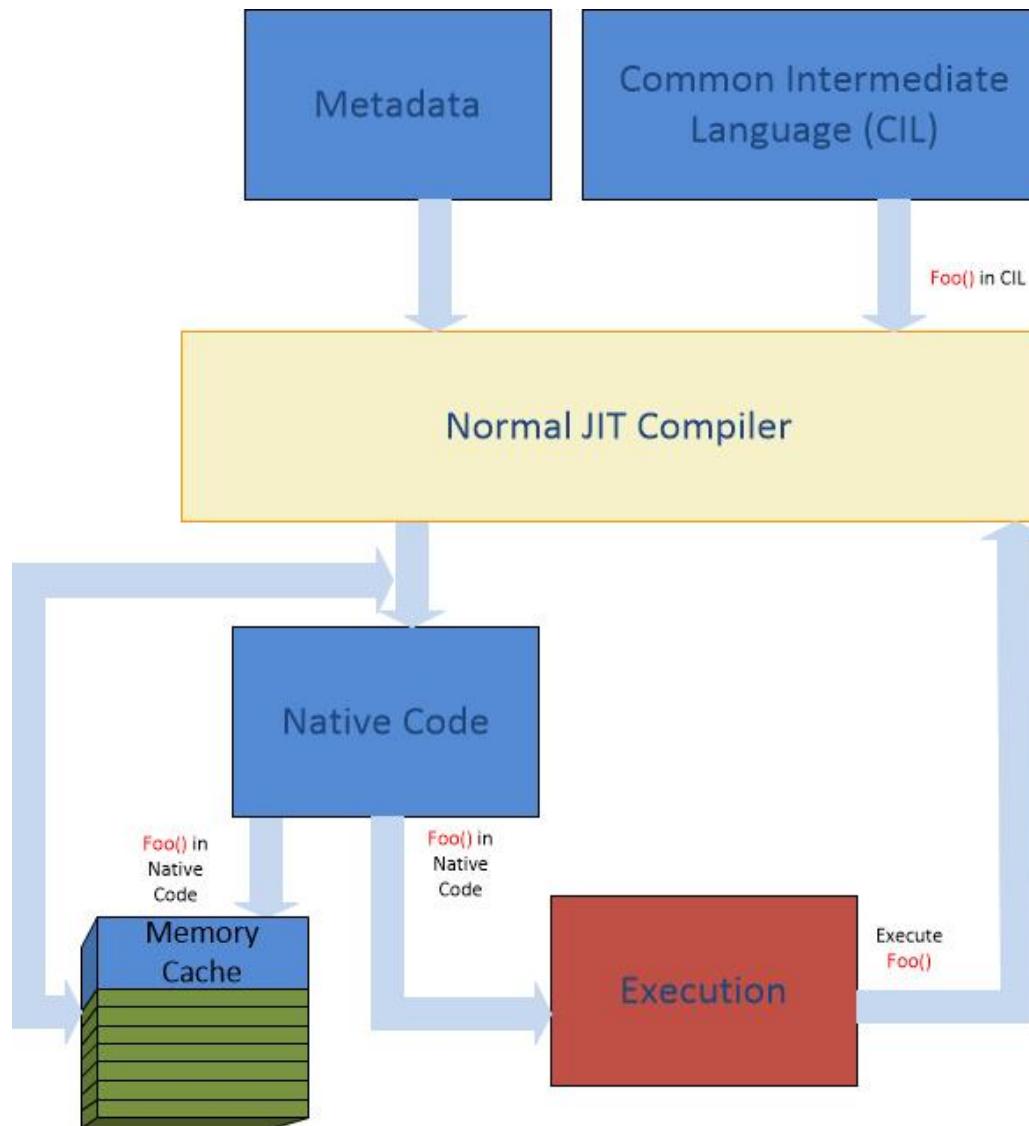
Основные компоненты CLR



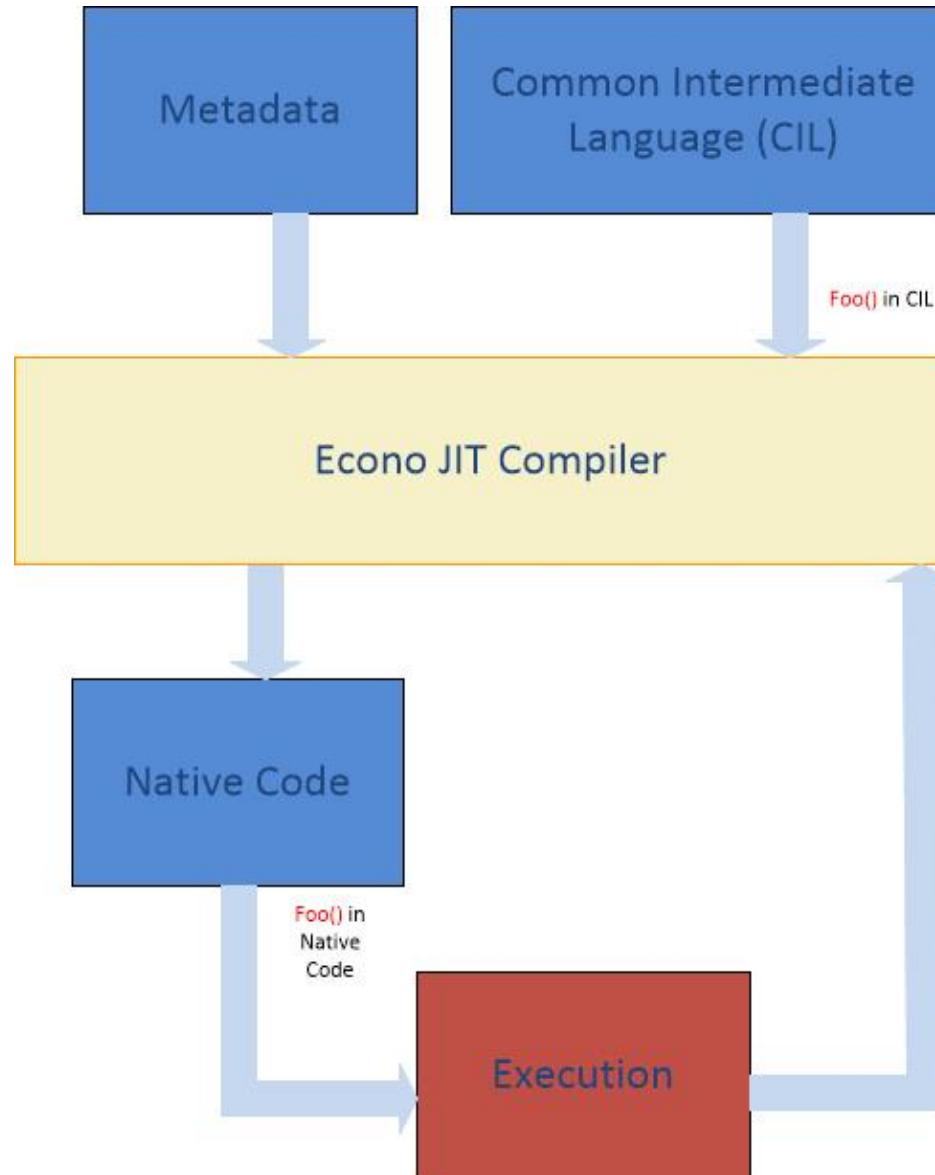
JIT-компиляция



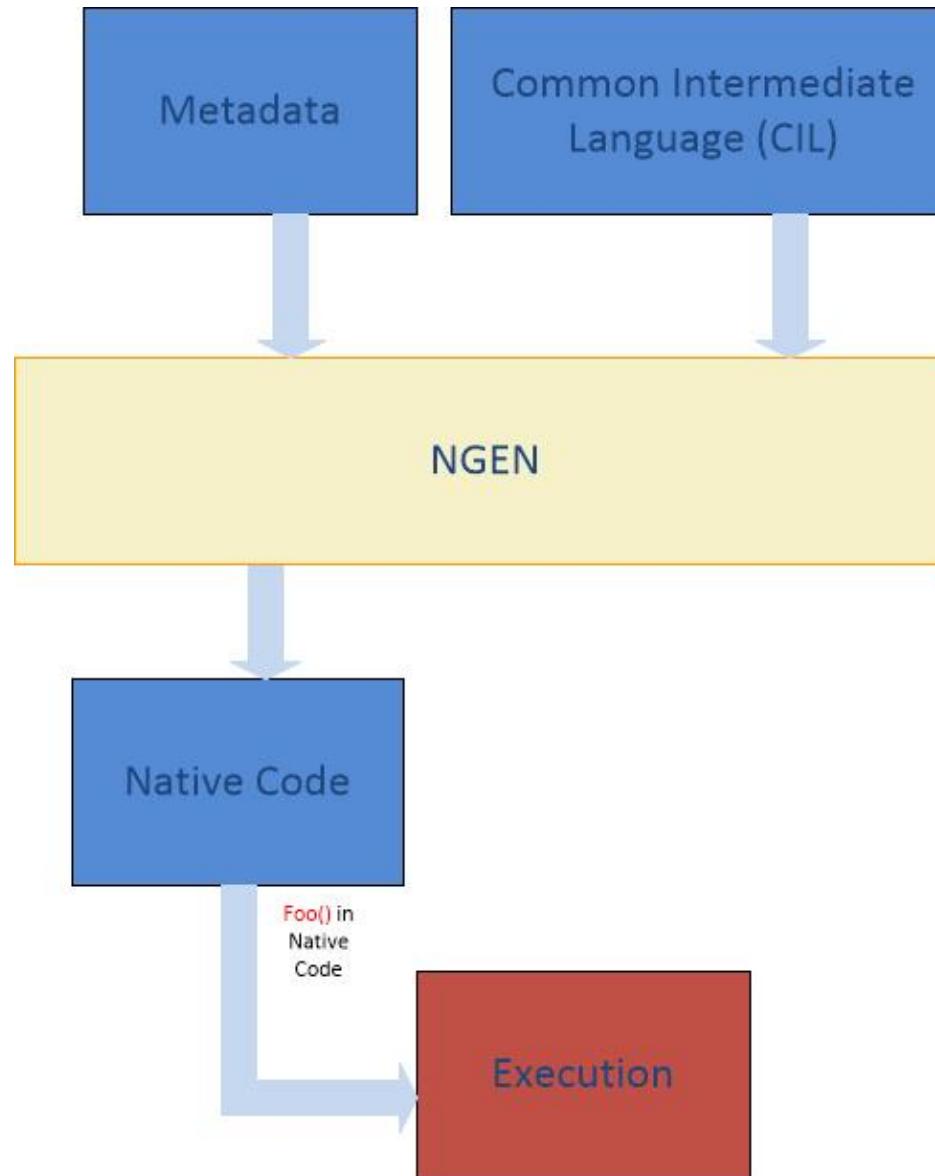
Виды компиляции. Normal JIT-компиляция



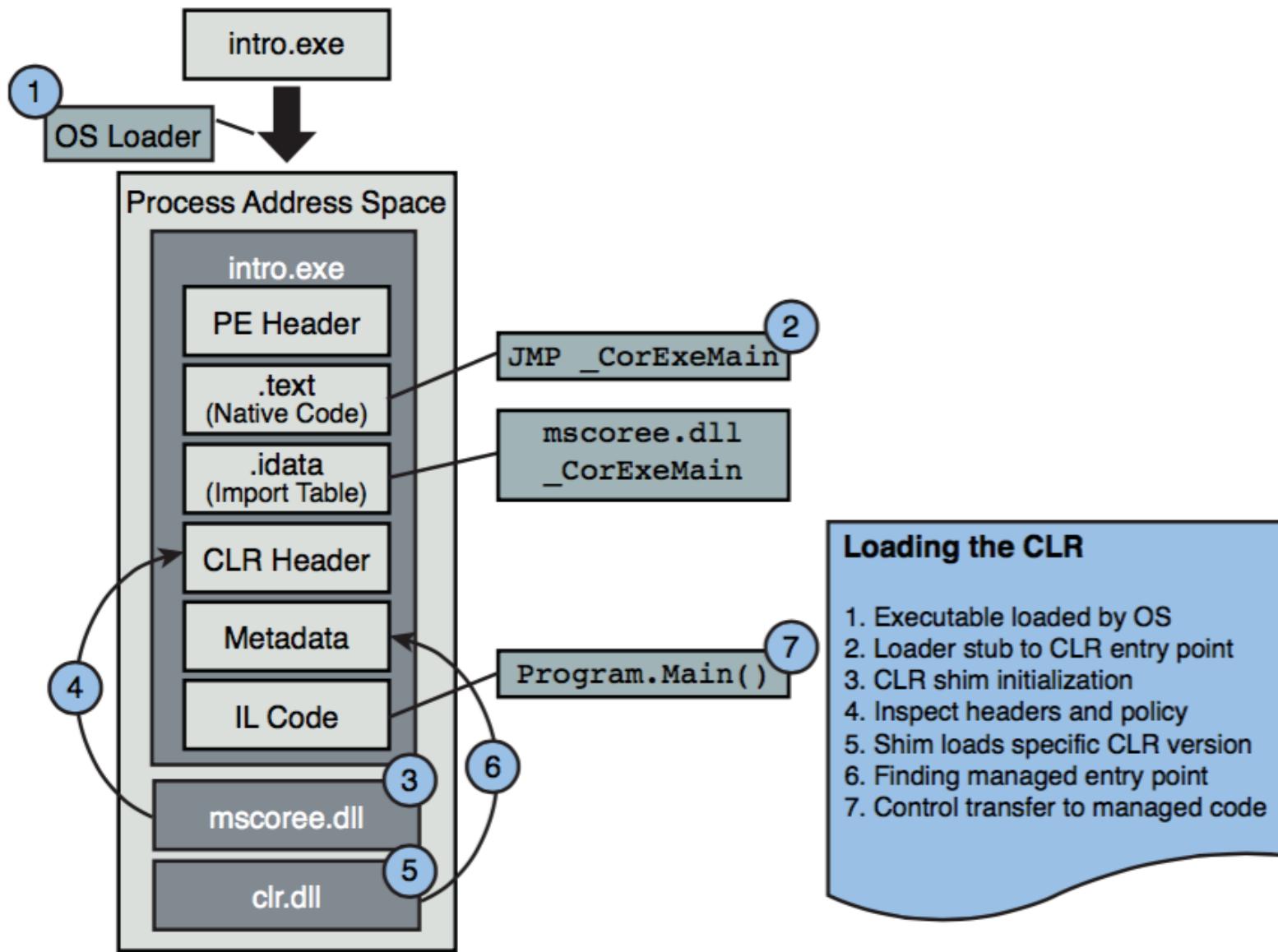
Виды компиляции. Econo-JIT-компиляция



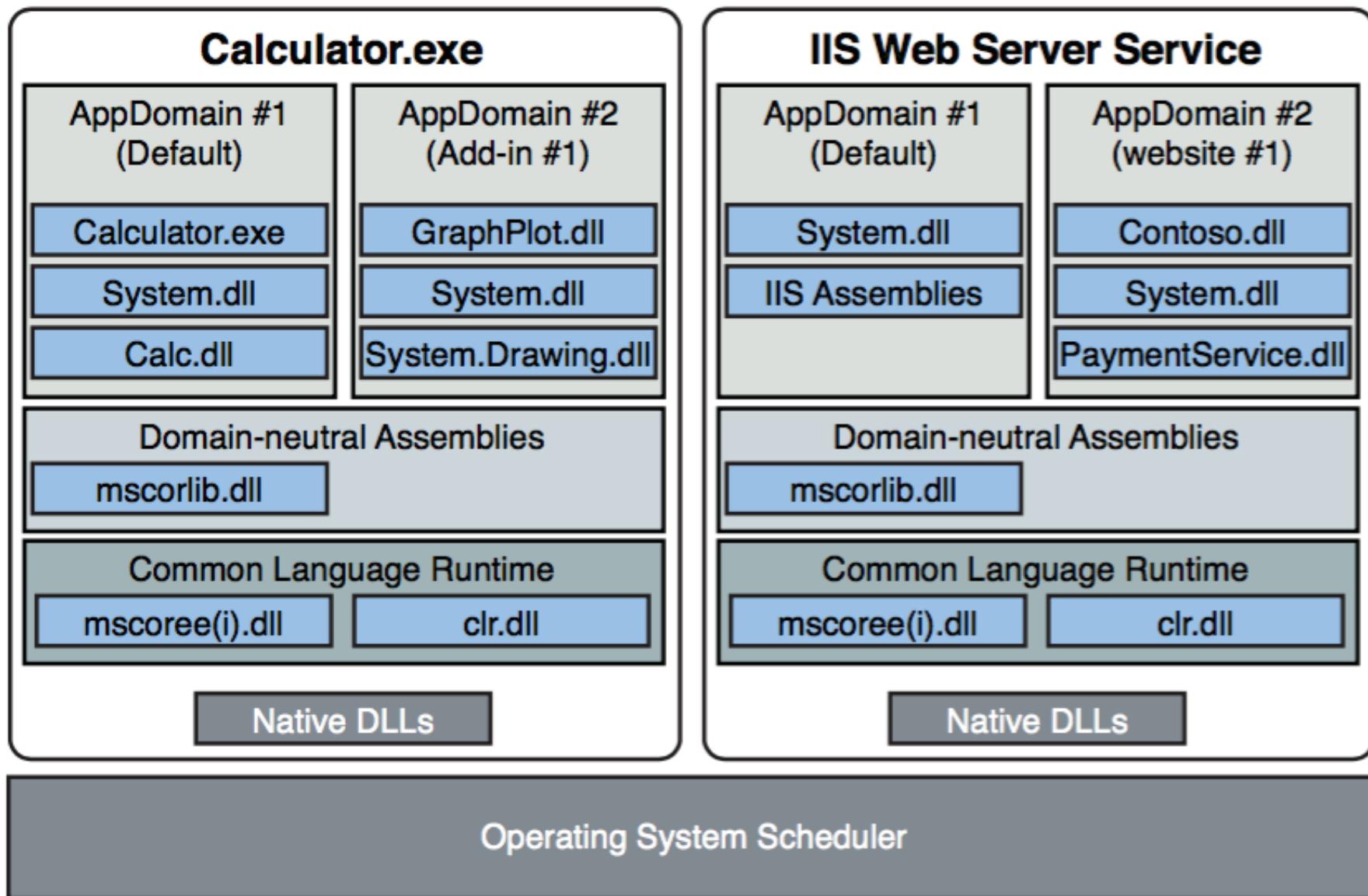
Виды компиляции. Pre-JIT-компиляция



Как CLR загружает, компилирует и запускает сборки



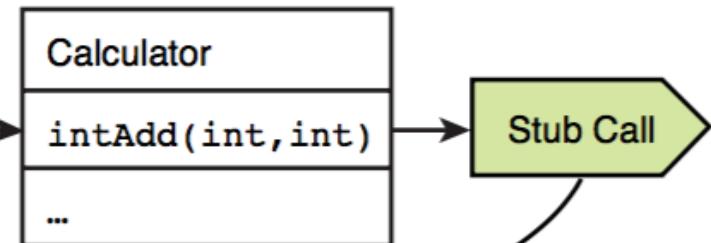
Домены приложения



Как CLR загружает, компилирует и запускает сборки

First Call to the Calculator's Add Method

```
static void Main()
{
    var calc = new Calculator();
    ➔ int four = calc.Add(1, 3);
    int five = calc.Add(2, 3);
    Console.WriteLine(sum);
}
```



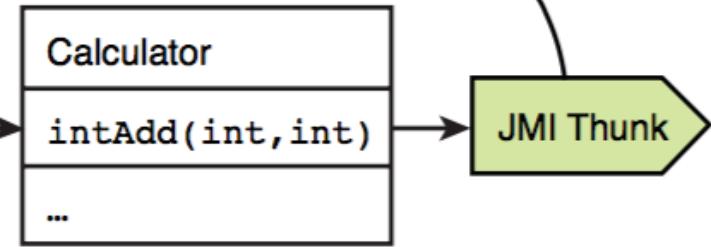
```
ldarg.1  
ldarg.2  
add  
ret
```

➔ JIT Compiler

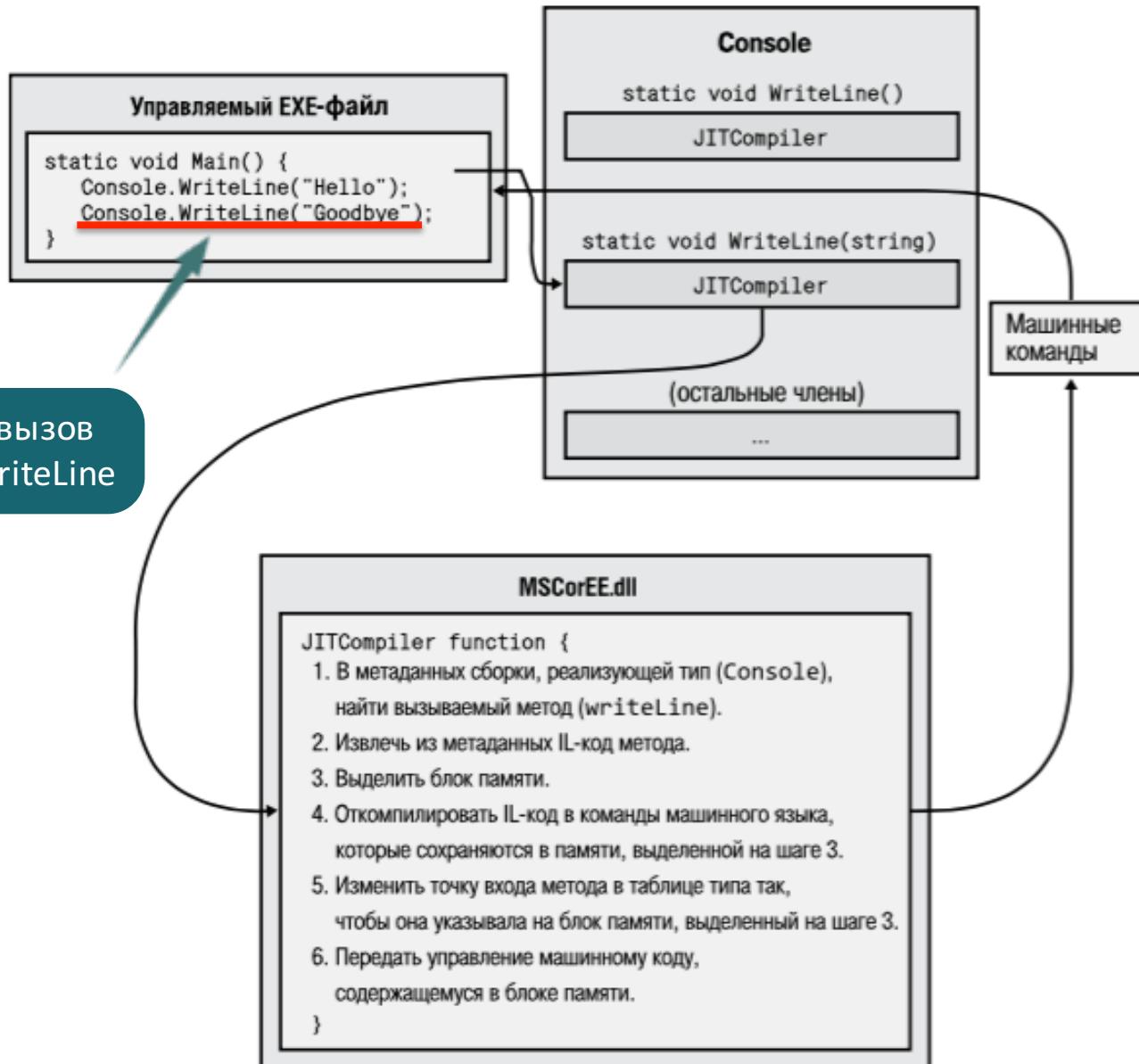
```
// x86 assembler fragment
mov eax, dword ptr [ebp-40h]
add eax, dword ptr [ebp+8]
mov dword ptr [ebp-44h], eax
jmp 00000044
```

Subsequent Calls to the Calculator's Add Method

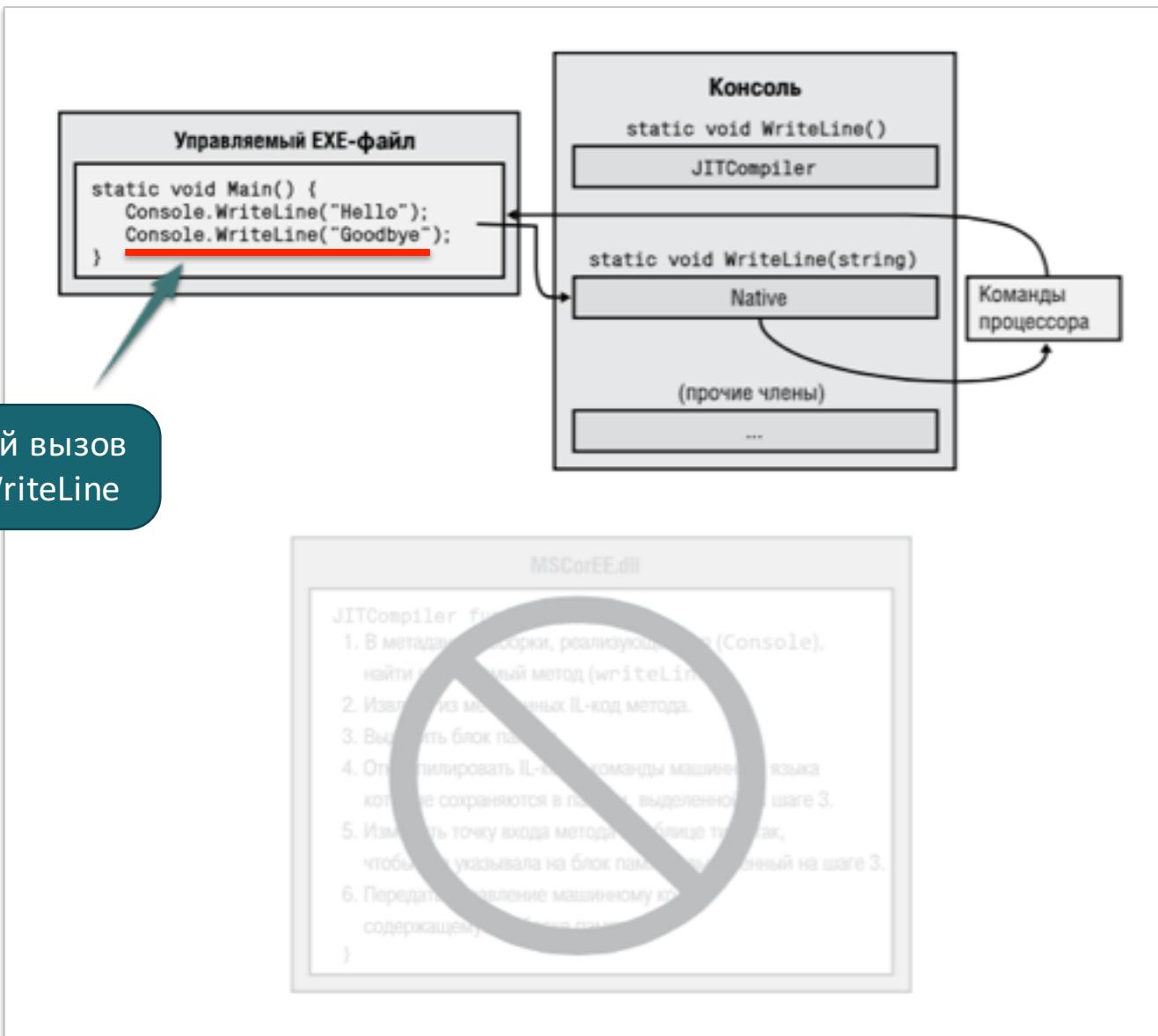
```
static void Main()
{
    var calc = new Calculator();
    int four = calc.Add(1, 3);
    ➔ int five = calc.Add(2, 3);
    Console.WriteLine(sum);
}
```



Как CLR загружает, компилирует и запускает сборки



Как CLR загружает, компилирует и запускает сборки

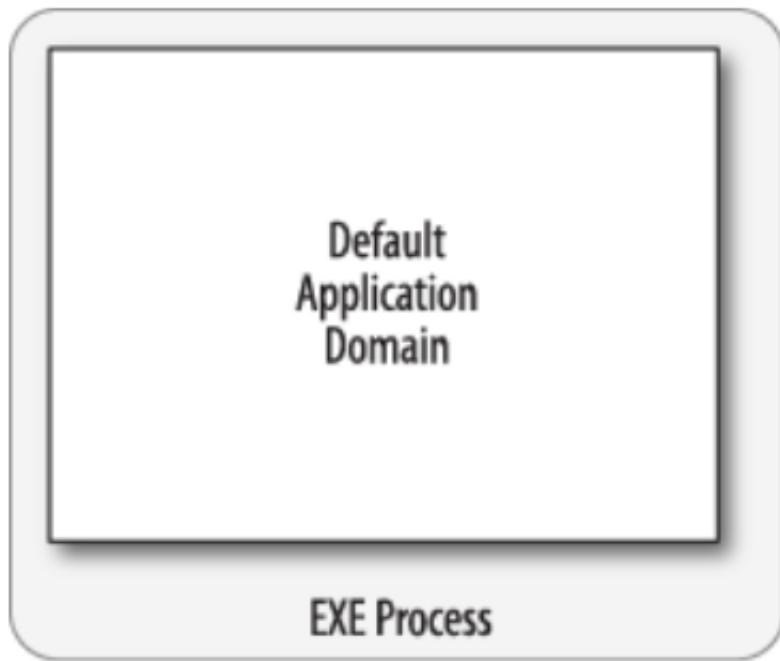


Как CLR загружает, компилирует и запускает сборки

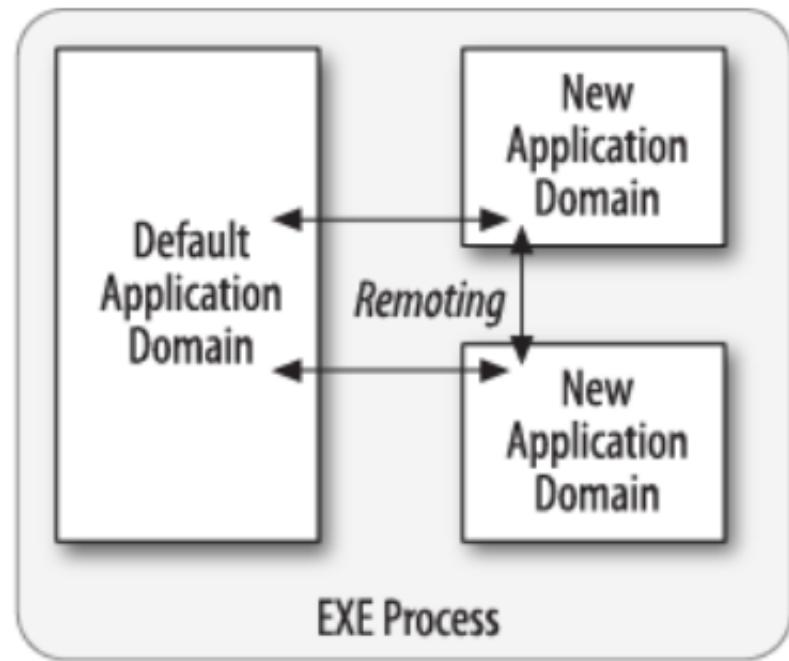
CLR содержит несколько компонентов, которые при запуске .NET Framework приложений выполняют следующие задачи:

- **Загрузчик классов (Class Loader)** находит и загружает все сборки, которые требуются приложению. Сборки к этому моменту будут уже скомпилированы в MSIL
- **MSIL-to-native компилятор** проверяет MSIL код, а затем компилирует все сборки в машинный код, готовый к исполнению
- **Code Manager** загружает исполняемую сборку и запускает метод Main
- **Garbage Collector** обеспечивает автоматическое управление памятью жизни всех объектов, которые создает приложение
- **Exception Manager** предоставляет структурированную обработку исключений для .NET приложений, которая интегрирована с структурированной обработкой исключений Windows

Домены приложения

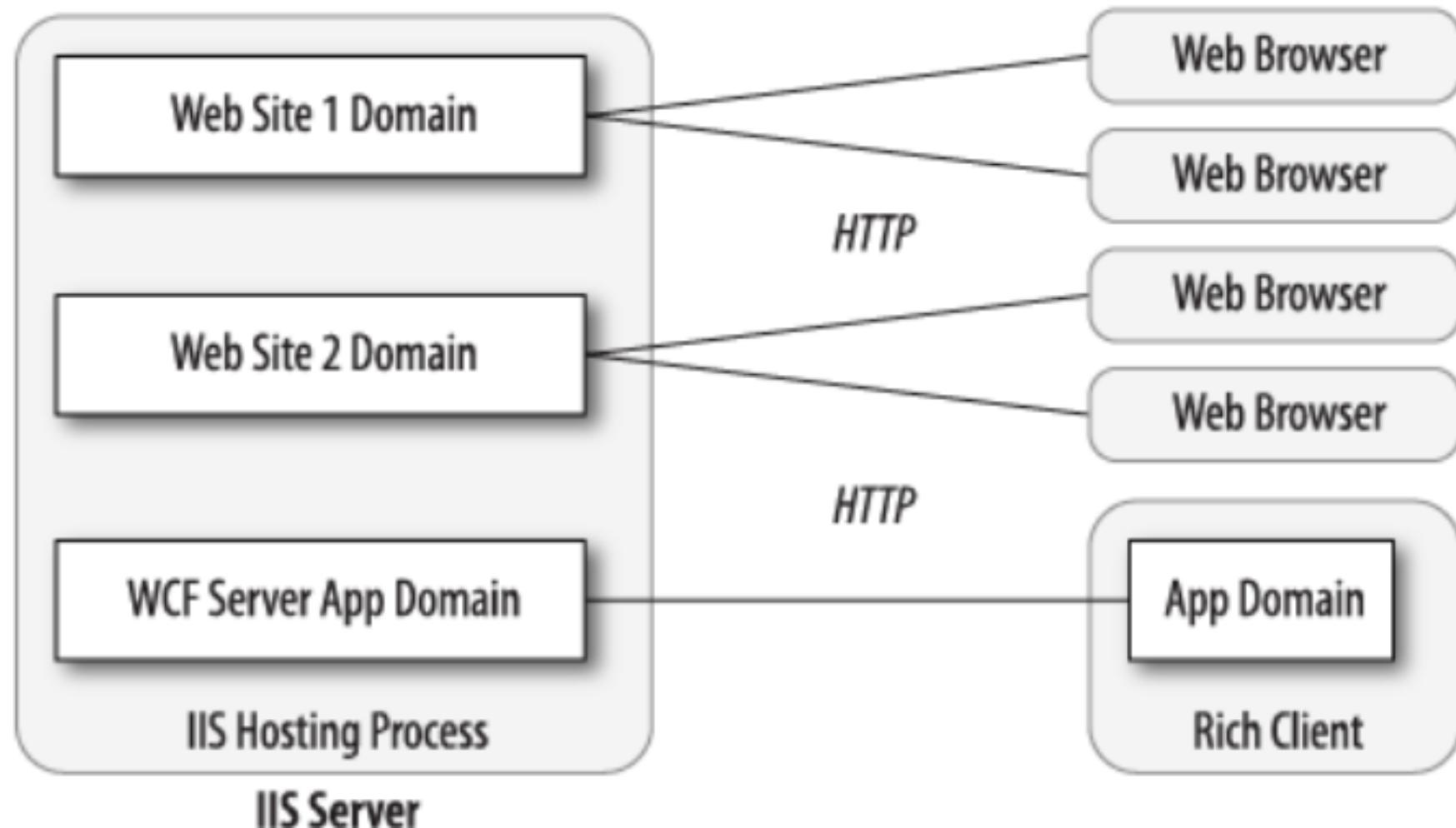


Single-Application-Domain Program



Multi-Application-Domain Program

Домены приложения



Спецификации

Спецификация CTS описывает способ определения и поведение типов

Common Type System

Стандарт CTS вместе с другими частями .NET Framework (форматы файлов, метаданные, IL, механизм вызова P/Invoke и т. д.) называется CLI (Common Language Infrastructure) и определяется спецификацией ECMA-335

ECMA-335

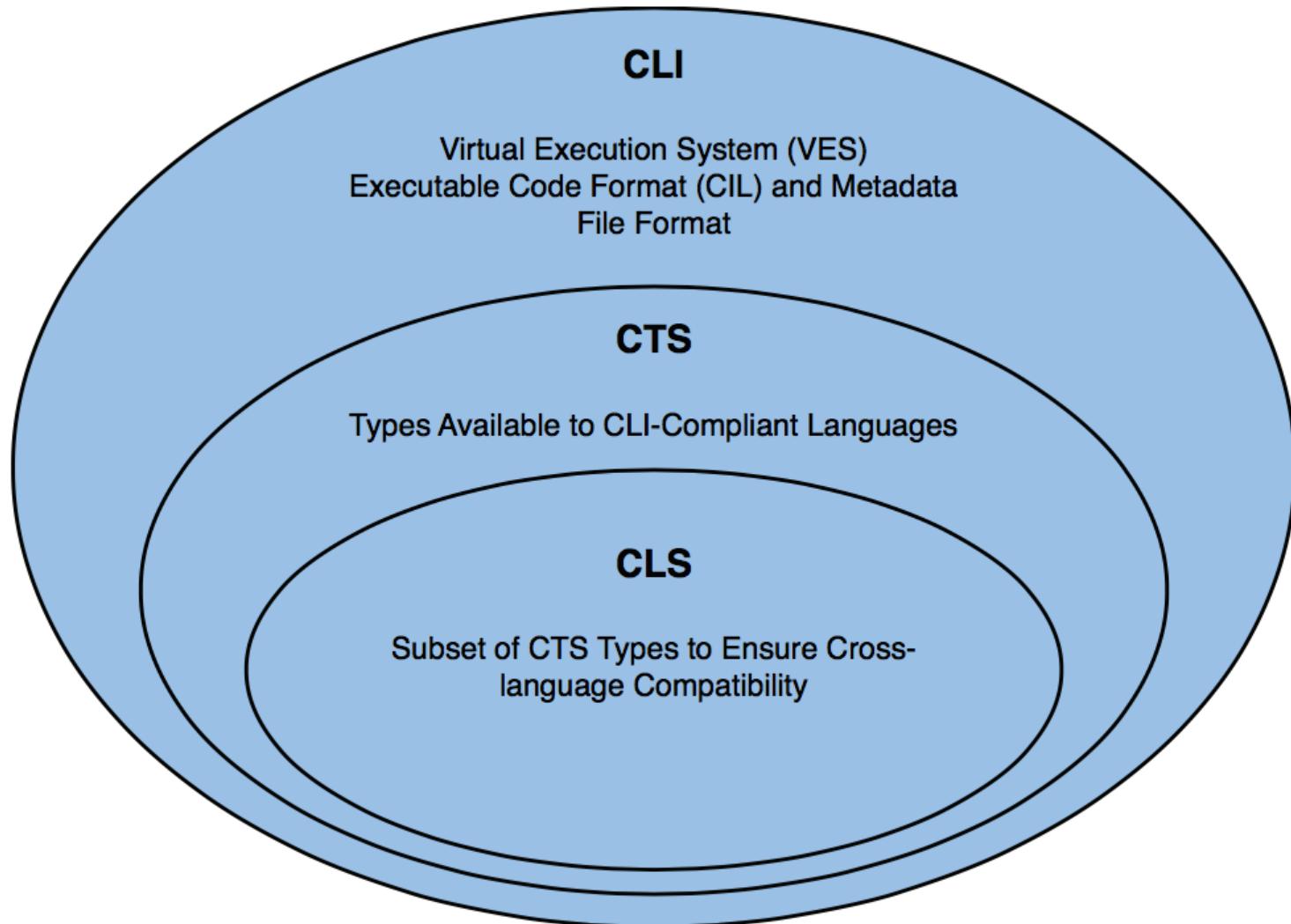
Спецификация CLS (Common Language Specification) перечисляет минимальный набор возможностей, которые должны поддерживаться компилятором для генерирования типов, совместимых с другими компонентами, написанными на других CLS-совместимых языках на базе CLR

ECMA-334

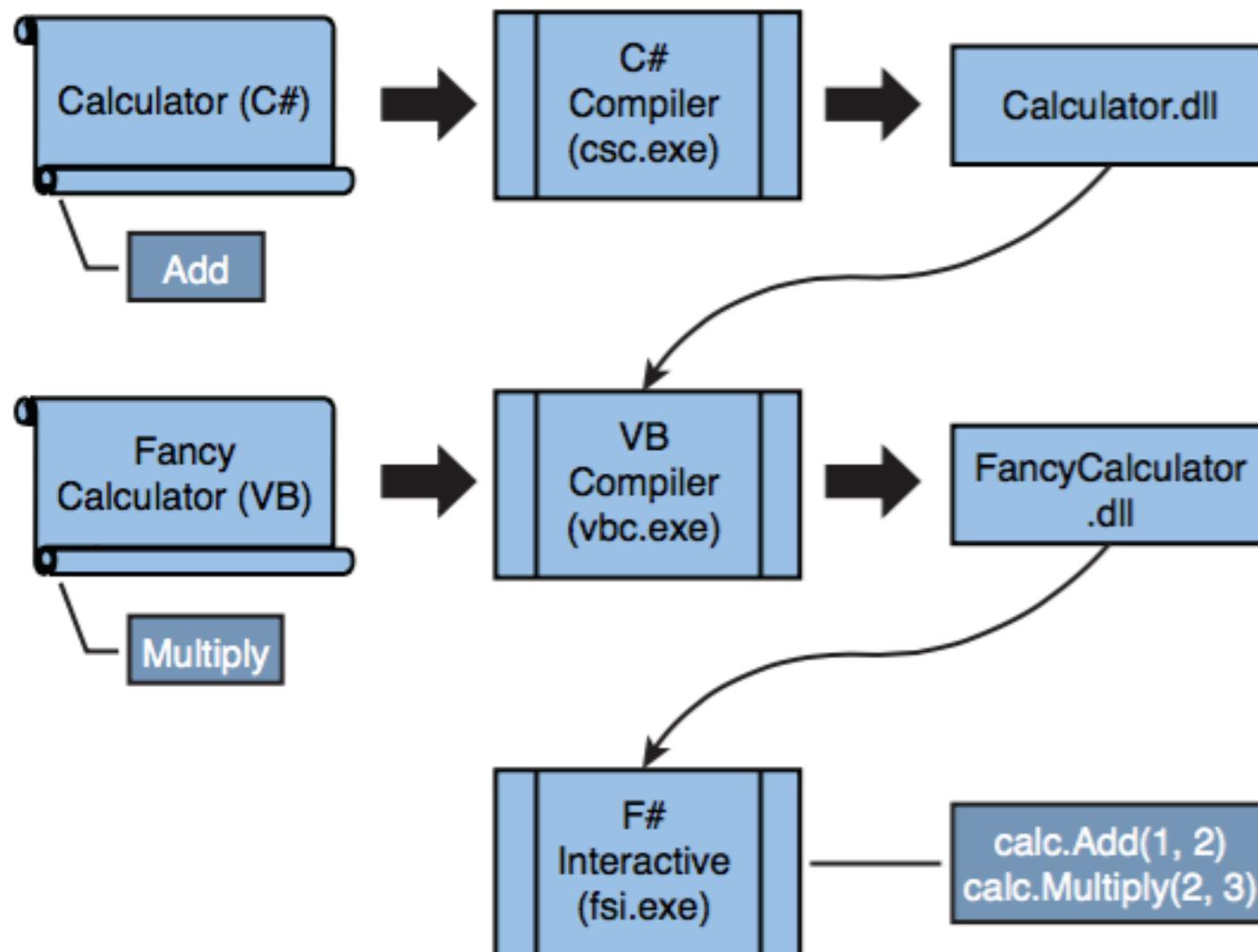
Common Language Specification

<http://www.ecma-international.org/>

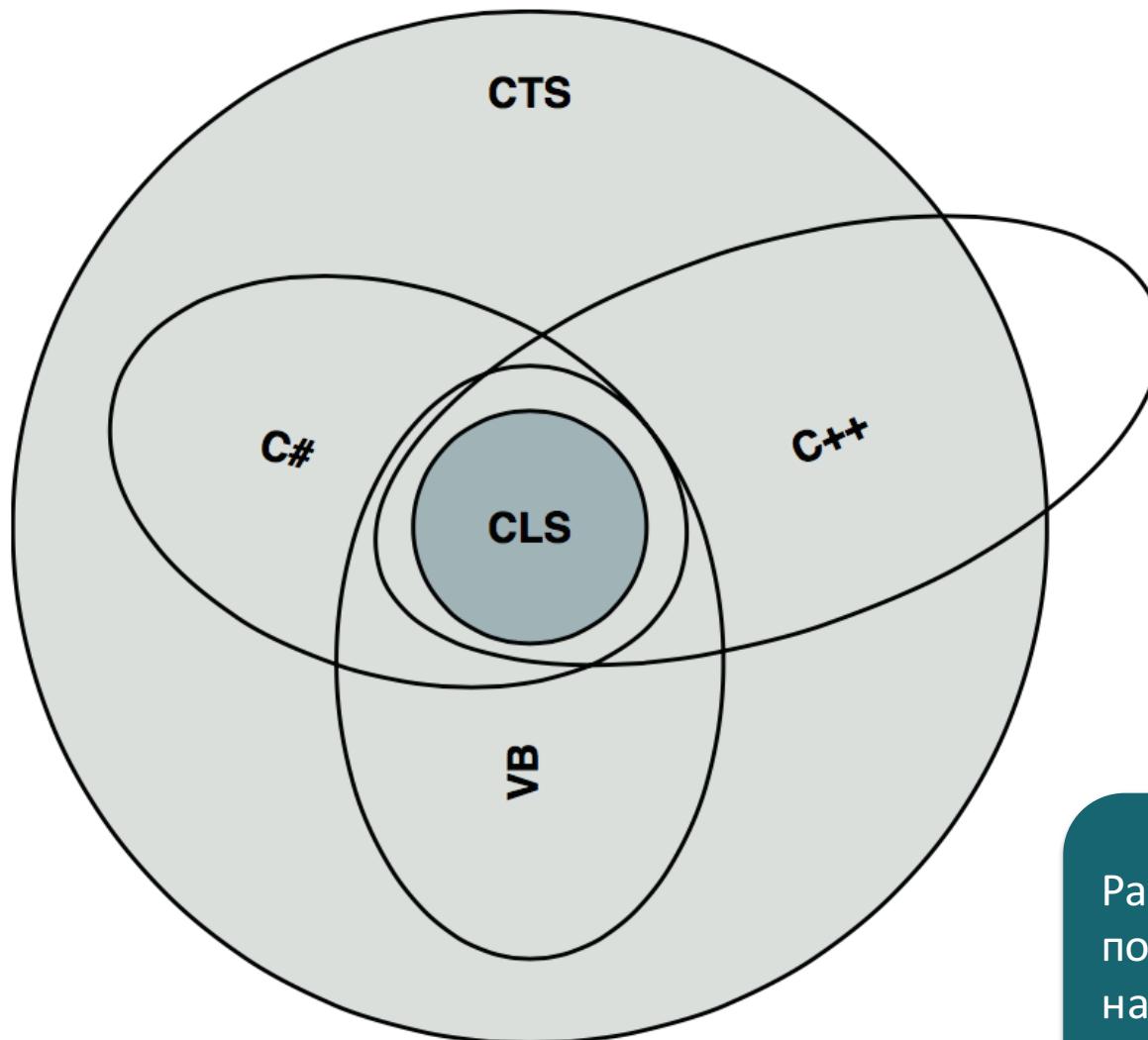
Введение в C# и .NET Framework



Введение в C# и .NET Framework



Введение в C# и .NET Framework



Разные языки поддерживают подмножество CLR/CTS и надмножество CLS (возможно, разные подмножества)

Некоторые характеристики сборки:

В сборке определены повторно используемые типы

Сборка помечена номером версии

Со сборкой может быть связана информация безопасности

Сборка определяет границы типов

Сборки являются самоописываемыми

Сборки поддаются конфигурированию

Номер версии сборки состоит из следующих компонент:

Главный номер версии (Major version number)

Второстепенный номер версии (Minor version number)

Номер сборки (Build number)

Номер редакции (Revision number)

NameAssembly, Version=1.1.0.0, Culture=en, PublicKeyToken=null

Подписание сборки:

Защищает сборки от модификаций

Позволяет включать подписанную сборку в глобальный кэш сборок (GAC), позволяя ее использование другими приложениями

Гарантирует, что имя сборки является уникальным

Тип сборки	Закрытое развертывание	Глобальное развертывание
Сборка с нестрогим именем	Да	Нет
Сборка со строгим именем	Да	Да

Сборки в .NET

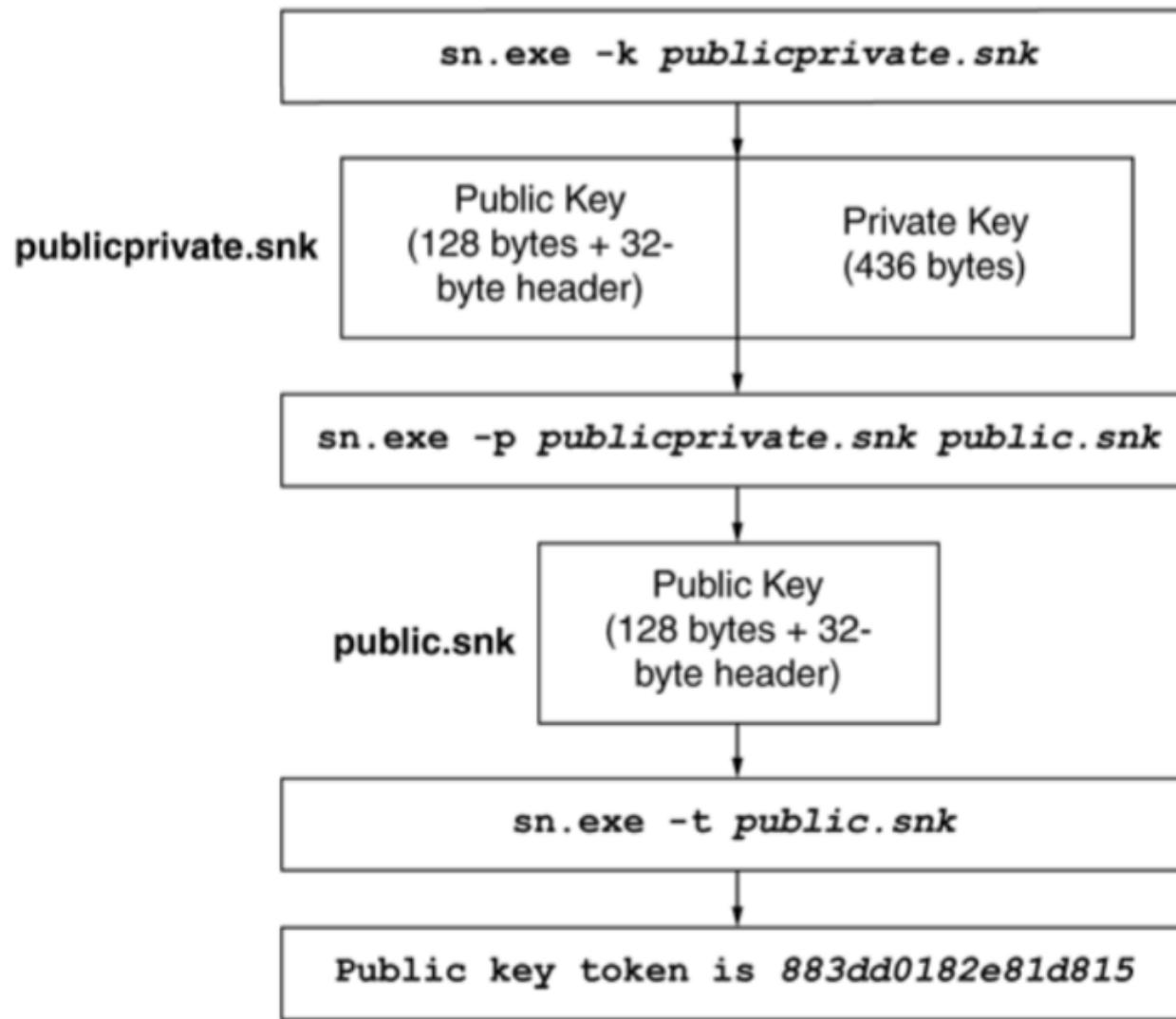
```
sn -k keyPair.snk
```

```
sn -p keyPair.snk publicKey.snk
```

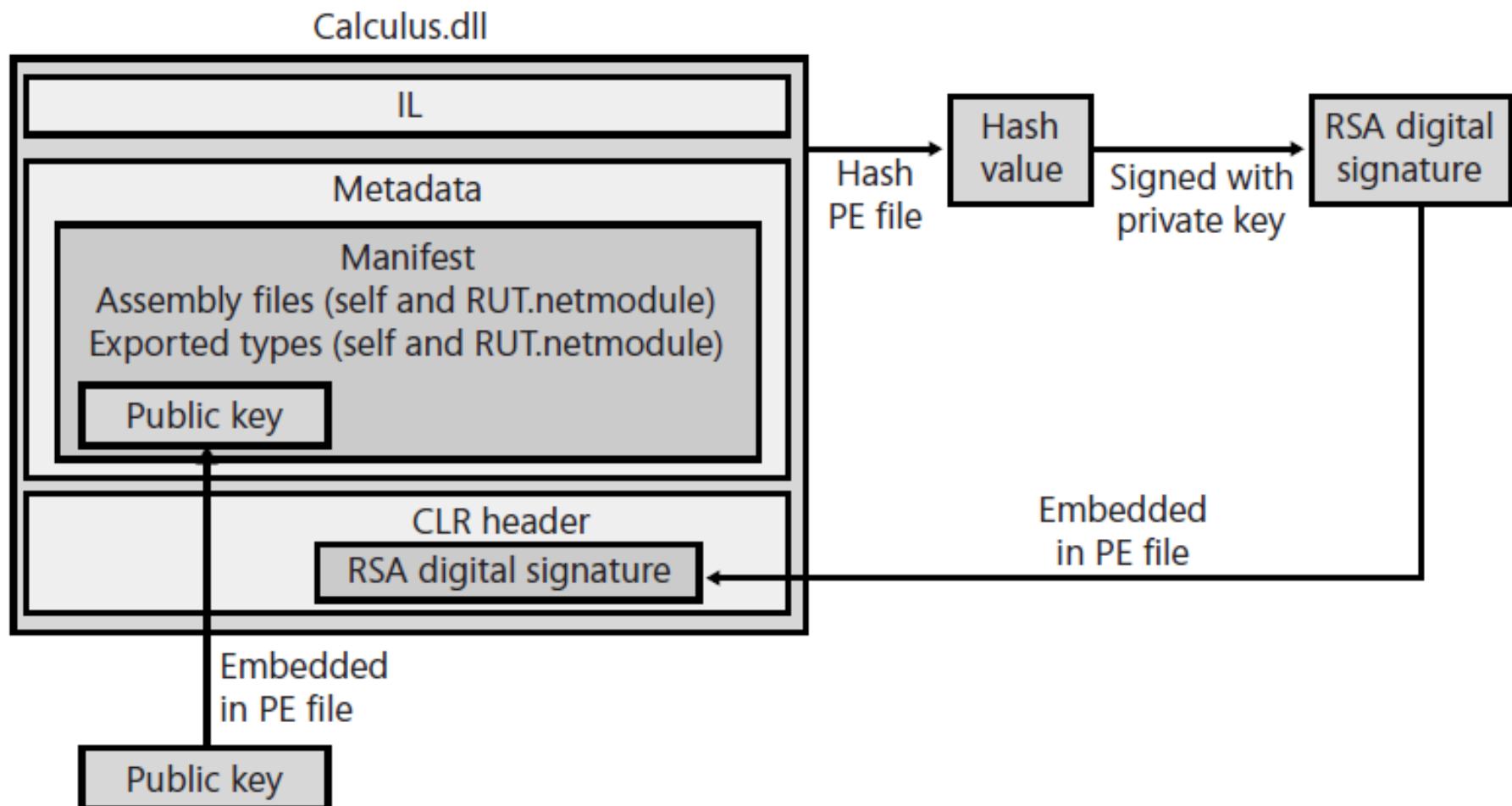
```
sn -tp keyPair.snk
```

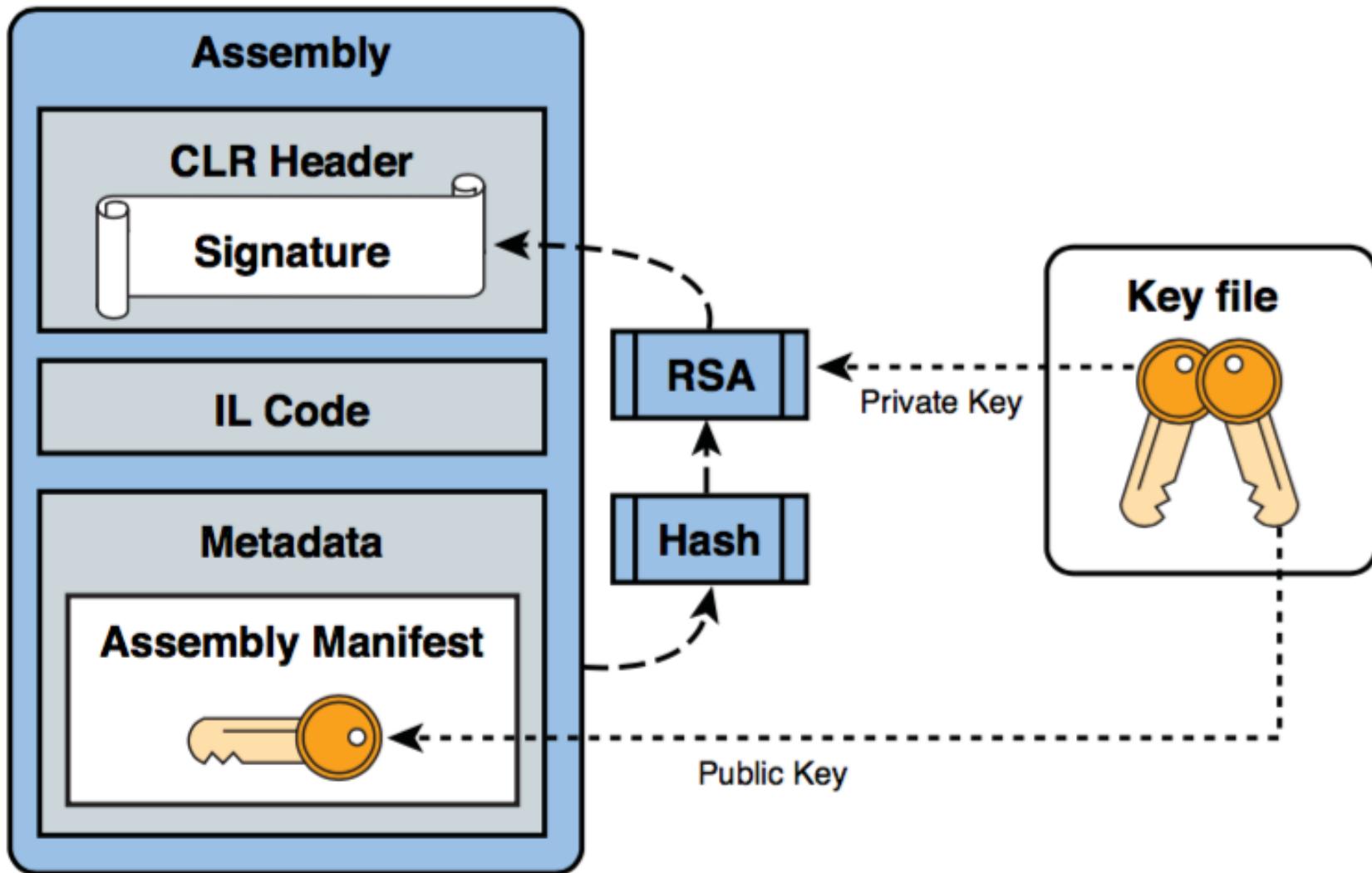
```
gacutil.exe -i NameAssambly.dll
```

Сборки в .NET



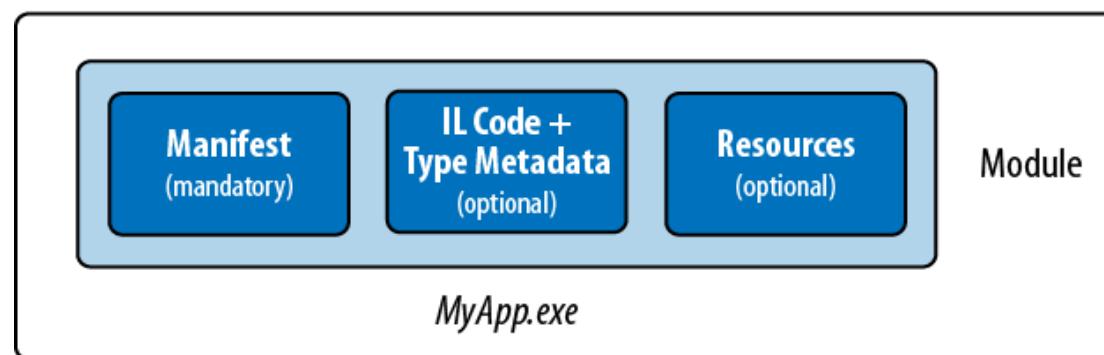
Сборки в .NET



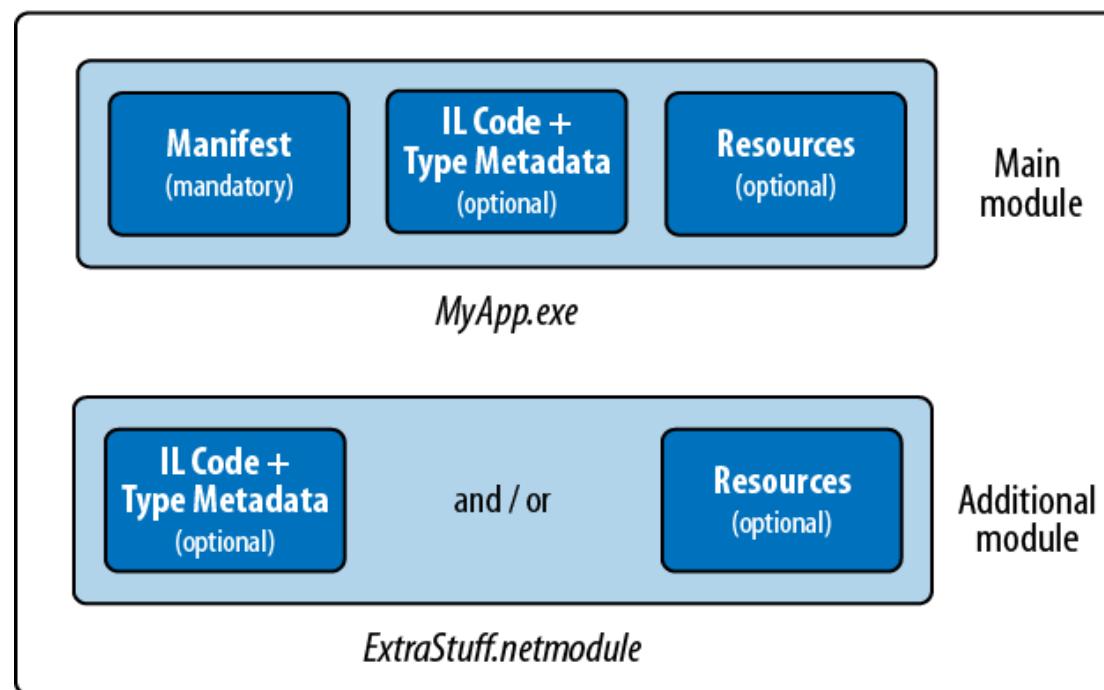


Сборки в .NET

Assembly



Assembly



Изучение сборок с помощью утилит ildasm.exe и Reflector

Утилита ildasm.exe, поставляемая в составе пакета .NET Framework 4 SDK, позволяет загружать любую сборку .NET и изучать ее содержимое, в том числе ассоциируемый с ней манифест, CIL-код и метаданные типов

NET Reflector – платная утилита для Microsoft .NET, комбинирующая браузер классов, статический анализатор и декомпилятор

<http://www.red-gate.com/products/dotnet-development/reflector/>

Free .NET Decompiler and Assembly Browser <https://www.jetbrains.com/decompiler/>
ILSpy is the open-source .NET assembly browser and decompiler <http://ilspy.net/>

Инструменты, предоставляемые .NET Framework

Caspol.exe

Makecert.exe

Ildasm.exe

Gacutil.exe

Ngen.exe

Sn.exe

[https://msdn.microsoft.com/ru-ru/library/d9kh6s92\(v=vs.110\).aspx](https://msdn.microsoft.com/ru-ru/library/d9kh6s92(v=vs.110).aspx)

Инструменты, предоставляемые .NET Framework

Инструмент	Описание
Global Assembly Cache Tool (Gacutil.exe)	Позволяет пользователям просматривать содержимое глобального кэша сборок и кэша загрузки, а также управлять ими. С помощью этого инструмента можно добавлять и удалять сборки в GAC, для того, чтобы приложения могли получать к ним доступ.
Native Image Generator (Ngen.exe)	Генератор образов в машинном коде (Native Image Generator) — это средство повышения быстродействия управляемых приложений. Ngen.exe создает образы в машинном коде, представляющие собой файлы, содержащие компилированный специфический для процессора машинный код, и устанавливает их в кэш образов в машинном коде на локальном компьютере. Среда выполнения может использовать образы в машинном коде, находящиеся в кэше, вместо использования JIT-компилятора для компиляции исходной сборки.

Инструменты, предоставляемые .NET Framework

Инструмент	Описание
MSIL Disassembler (Ilasm.exe)	MSIL Disassembler является парным инструментом к ассемблеру MSIL (Ilasm.exe). Ilasm.exe принимает входной исполняемый файл (PE-файл). Содержащий код на языке MSIL, и создает на его основе текстовый файл, который может служить входным для программы Ilasm.exe. Можно использовать Ilasm.exe для просмотра промежуточного языка MSIL в файле. Если анализируемый файл является сборкой, то эти данные могут включать в себя атрибуты сборки, а также ссылки на другие модули и сборки. Эти данные полезны для определения того, является ли файл сборкой или частью сборки и имеет ли он ссылки на другие модули и сборки.
Strong Name Tool (Sn.exe)	Позволяет пользователям подписывать сборки строгими именами. Strong Name Tool включает в себя команды для создания новой пары ключей, извлечения открытого ключа из пары ключей и верификации сборки.

Документирование приложений. XML комментарии

В Visual Studio можно добавить комментарии к исходному коду, который будет обработан в XML файл

XML файл может быть включен в процесс создания справочной документации по классу или использован для поддержки IntelliSense

```
/// <summary> The Hello class prints a greeting on the screen
/// </summary>
public class Hello
{
    /// <summary> We use console-based I/O. For more information about
    /// WriteLine, see <seealso cref="System.Console.WriteLine()" />
    /// </summary>
    public static void Main()
    {
        Console.WriteLine("Hello World");
    }
}
```

Документирование приложений. XML комментарии

Тег	Назначение
<summary>	Предоставляет краткое описание. Для более подробного описания используются теги <remarks>.
<remarks>	Содержит подробное описание. Этот тег может содержать вложенные разделы (пункты), списки и другие типы тегов.
<example>	Предоставляет пример того, как метод, свойство или другой член библиотеки должен быть использован. Этот тег часто связано с использованием вложенных тегов <code>.
<code>	Указывает, что прилагаемый текст является кодом приложения.
<returns>	Документирует возвращаемое значение и тип метода.
<exception>	Документирует класс исключения (синтаксис проверяется компилятором)
<param>	Помечает параметр метода (синтаксис проверяется компилятором)
<value>	Описывает свойство

<https://msdn.microsoft.com/ru-ru/library/b2s063f7.aspx>

Создание документации из XML комментариев



Sandcastle - Documentation Compiler for Managed Class Libraries - генератор документации (открытая часть некоторого внутреннего инструмента) компании Microsoft, который позволяет автоматически получить техническую документацию в стиле MSDN по заданной .NET-сборке с управляемым кодом.

Sandcastle состоит из набора утилит, которые получают метаинформацию из сборки, и затем рядом операций преобразуют ее в конечный вид. В процессе преобразования информация представлена в формате XML, часть преобразований выполняется с помощью шаблонов XSLT.

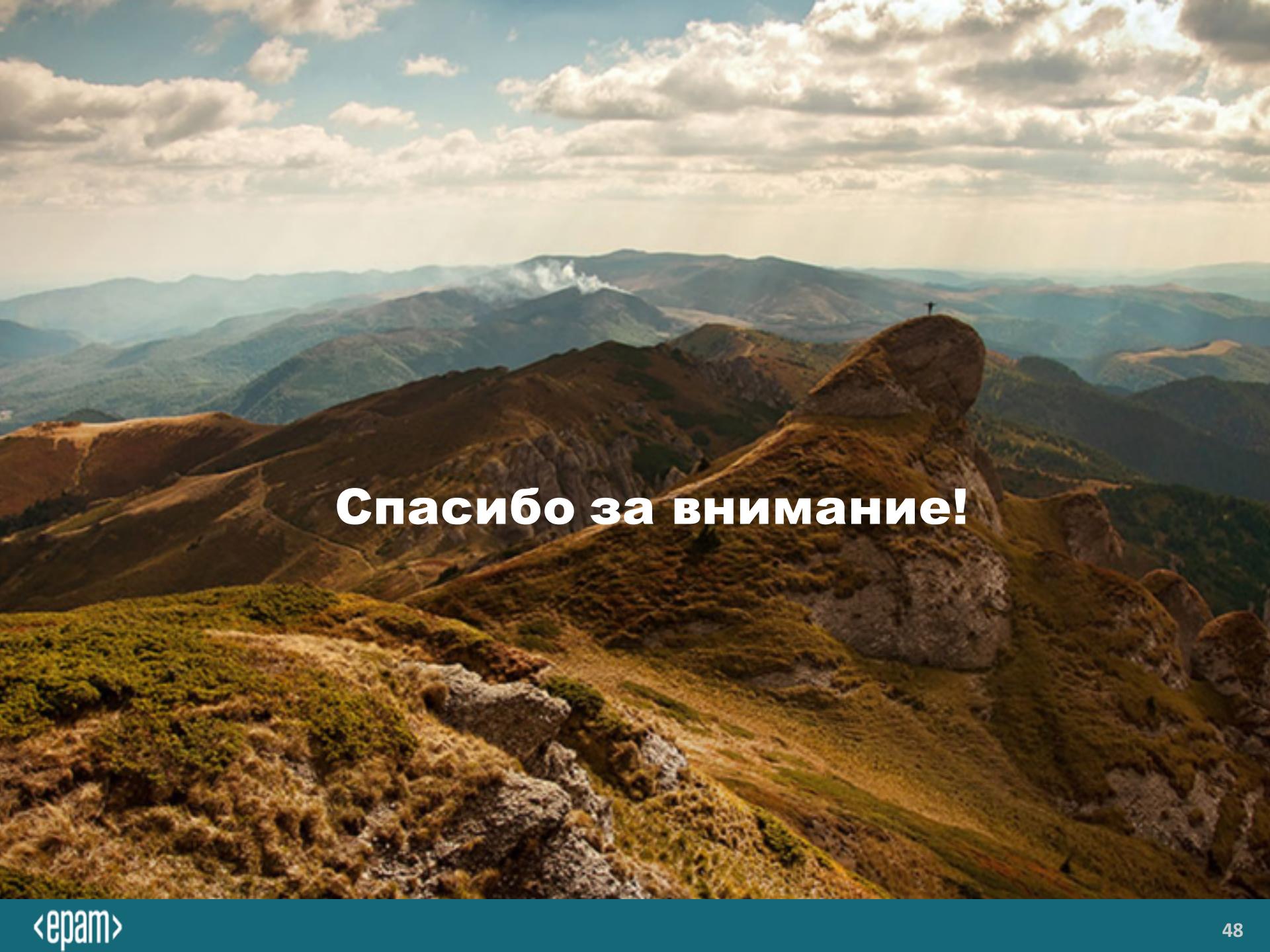
Sandcastle используется внутри Microsoft для получения документации на Visual Studio и .NET Framework.

<http://sandcastle.codeplex.com/>

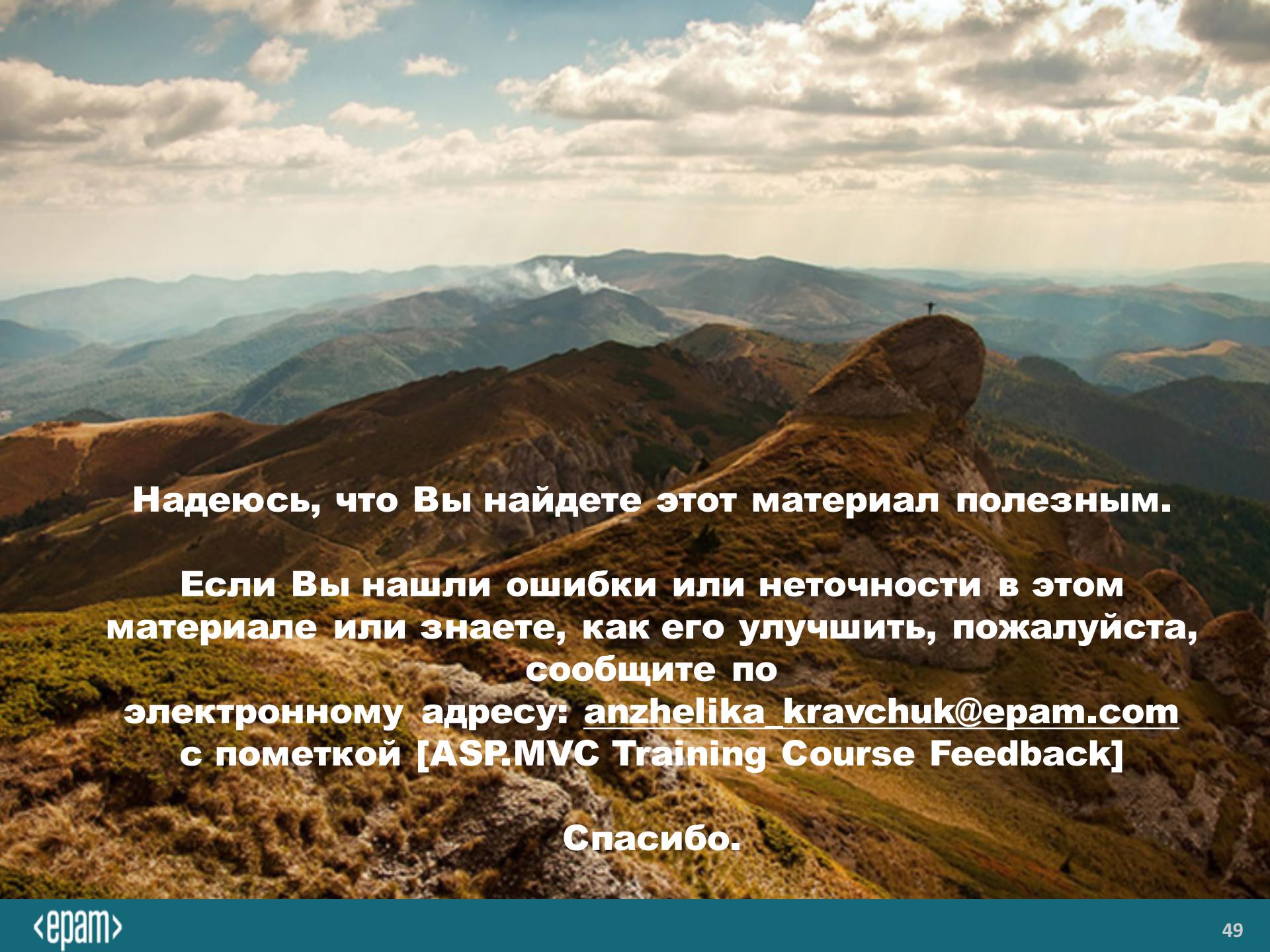
Создание документации из XML комментариев

На основании существующего XML файла, содержащего комментарии, которые были извлечены из проекта, можно создать .chm файл с помощью такого инструмента как Sandcastle Help File Builder. Для этого нужно выполнить следующие действия

- Нажать кнопку Start, пункт All Programs, выбрать Sandcastle Help File Builder, а затем нажать Sandcastle Help File Builder GUI.
- В Sandcastle Help File Builder, в меню File выбрать команду New Project.
- В диалоговом окне Save New Help Project As выполнить следующие действия, а затем нажать Save:
 - Выбрать путь, по которому следует сохранить проект.
 - Указать имя для Sandcastle проекта.
- В окне Project Explorer щелкнуть правой кнопкой мыши Documentation Sources, а затем нажать кнопку Add Documentation Source.
- В диалоговом окне Select the documentation source(s) перейти к папке, содержащий XML файл, а затем нажать кнопку Open.
- В меню Documentation, выбрать Build Project. Подождать, пока проект успешно построится. Это займет некоторое время.

A wide-angle photograph of a mountain range under a dramatic sky. In the foreground, rocky terrain and green slopes are visible. A lone figure stands on a prominent peak in the middle ground. The background features multiple layers of mountains, with a large plume of white smoke or steam rising from one of the peaks in the distance.

Спасибо за внимание!

A wide-angle photograph of a mountain range under a dramatic sky. In the foreground, a rocky mountain ridge slopes down towards the viewer. On the right side of the ridge, a small figure of a person stands on a prominent rock, looking out over the vast landscape. The background features several layers of mountains, with the furthest ones appearing as dark silhouettes against a bright, cloudy sky.

Надеюсь, что Вы найдете этот материал полезным.

**Если Вы нашли ошибки или неточности в этом
материале или знаете, как его улучшить, пожалуйста,
сообщите по**

**электронному адресу: anzhelika_kravchuk@epam.com
с пометкой [ASP.MVC Training Course Feedback]**

Спасибо.