

.NET code hot reload



Асадуллин

Тимур
Directum,

Уфа

Что за «Code hot reload»?

- Обновление кода приложения без перезапуска этого приложения.
- .NET Framework (4.6 +)
- Надеюсь такие слова вас не пугают: CLR, MSIL, Assembly, указатель.

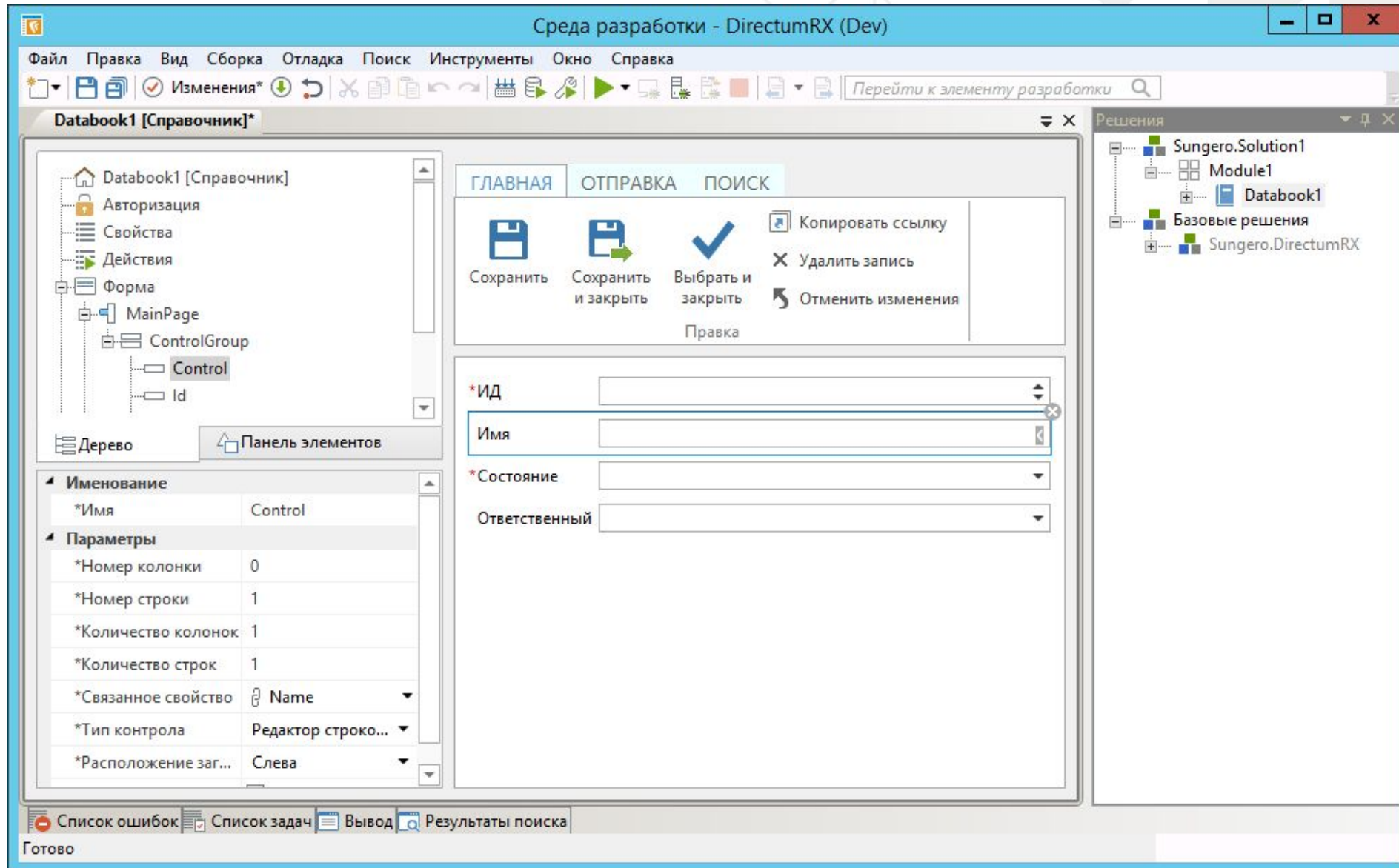
DirectumRX – ECM-система

The screenshot displays the DirectumRX web application interface. The browser address bar shows the URL `https://localhost/Client/#/Folder/2`. The application header includes the DirectumRX logo, a search bar with the text "Искать документы, задачи, прочее...", and user profile icons. The main navigation menu on the left lists various sections: "Искать в модулях...", "Входящие", "Исходящие", "Избранное", "Недавние документы" (highlighted), "Недавние задания", "Общие папки" (with sub-items "Проекты"), and "Модули" (with sub-items "Делопроизводство", "Договоры", "Проекты", "Совещания", "Финансовый архив").

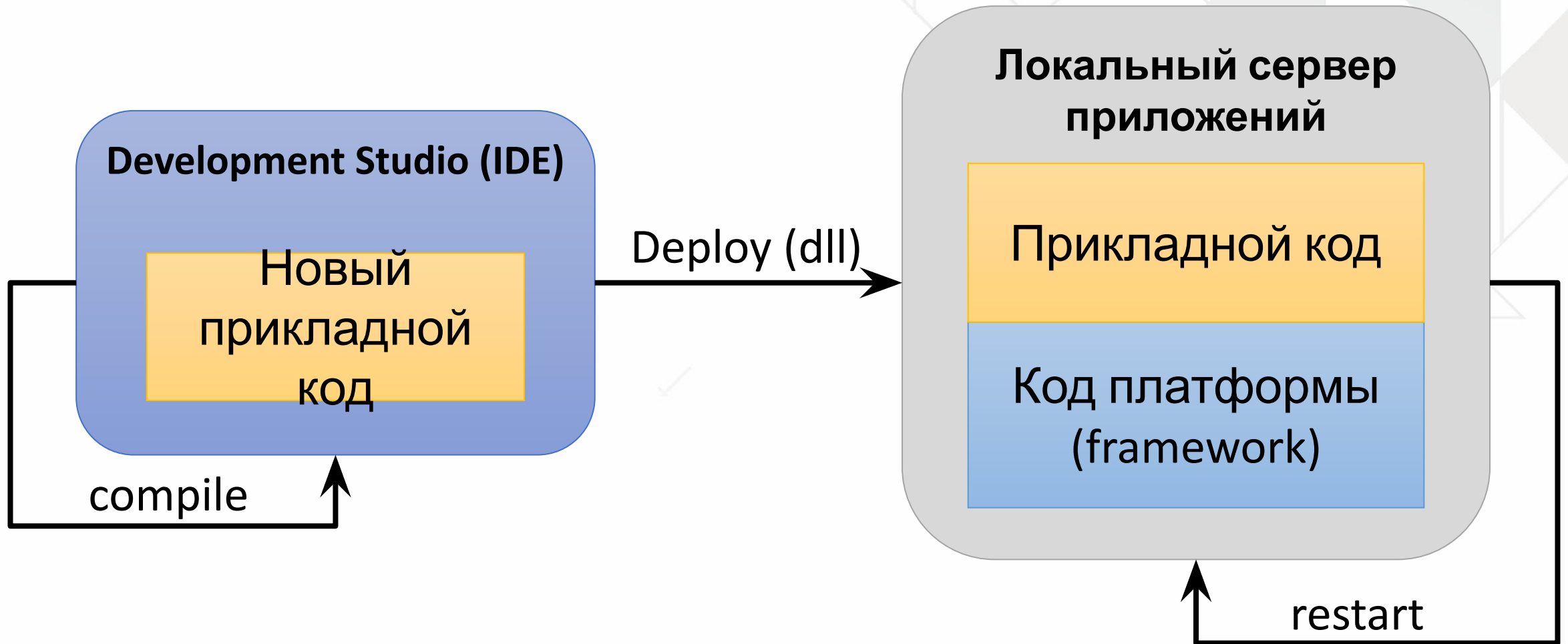
The "Недавние документы" section is active, showing a list of documents. The header of this section includes a search input "Искать в списке...", a refresh button, and filters for "Сегодня", "За 3 дня" (selected), and "За 7 дней". The document list is sorted by name and contains the following items:

@	Имя ↓
✓	Схема публикации DT
	Реализация режима черновика редакторе схемы WF
	Предпросмотр документов в вебе
	Предпросмотр документов в вебе
	Подписка на события типов сущностей в модуле
	Sungero overview
	DirectumRX 3.2 анализ спринта 3

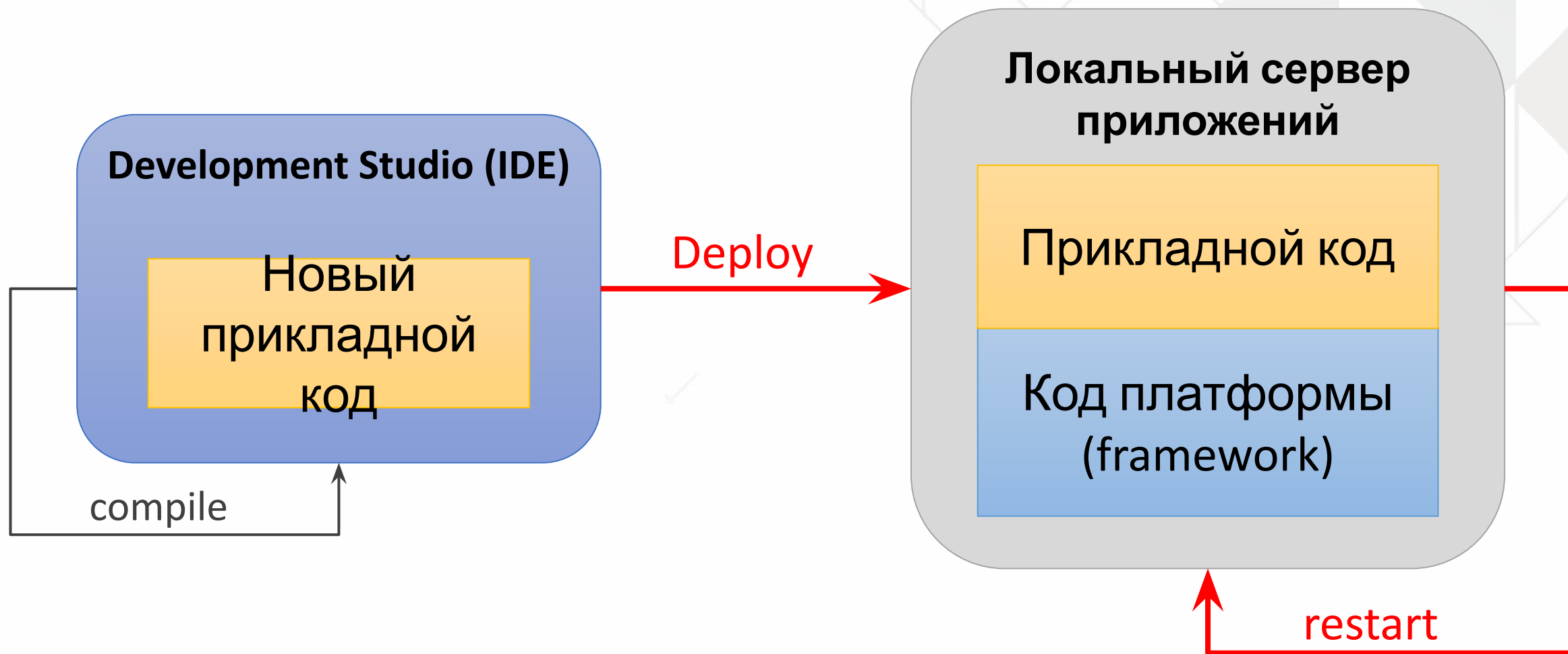
DirectumRX Development Studio



Прикладная разработка DirectumRX



Зачем нам HotReload

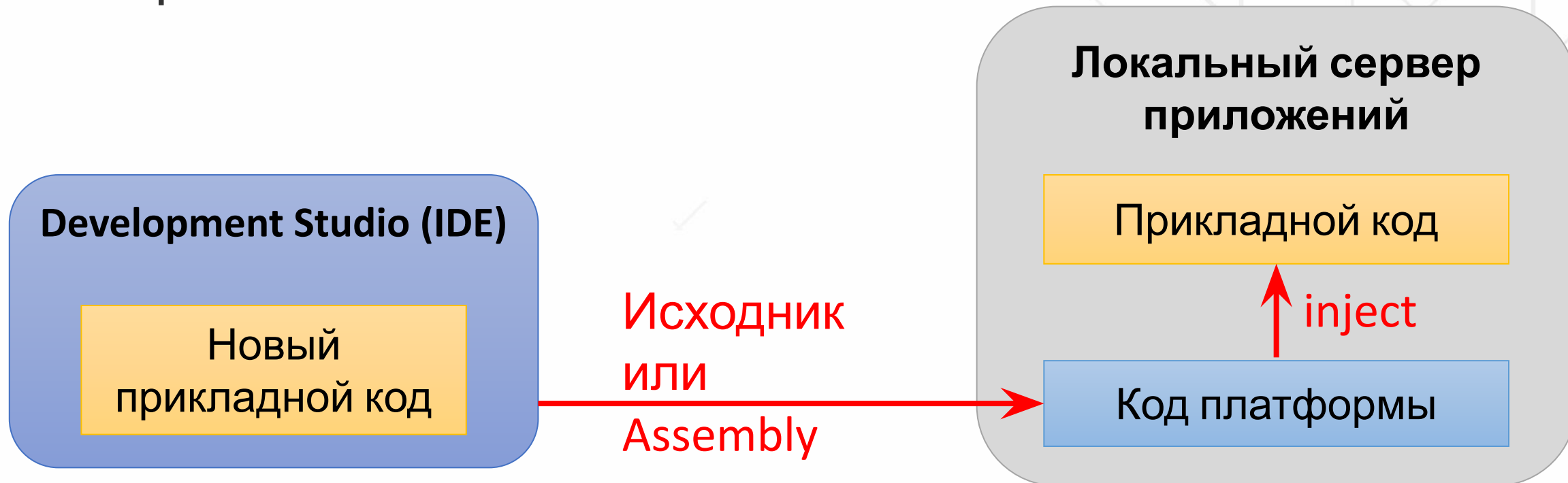


Дополнительные ограничения и пожелания

- Если поменялась структура БД, одним HotReload не обойтись.
- HotReload и отладка в Dev Studio (breakpoints, watch).
- Подходит и для серверного, и для клиентского кода
(у нас есть Desktop-клиент).

Основная идея

- Избавиться от перезапуска сервера.
- Встраивать код напрямую в работающее приложение.

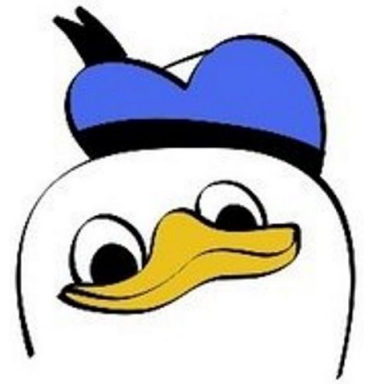




Решения в лоб

- Reflection – LoadAssembly (Shadow Copy Assemblies)
- Managed Extensibility Framework (MEF, VS-MEF)
- Mono.Cecil

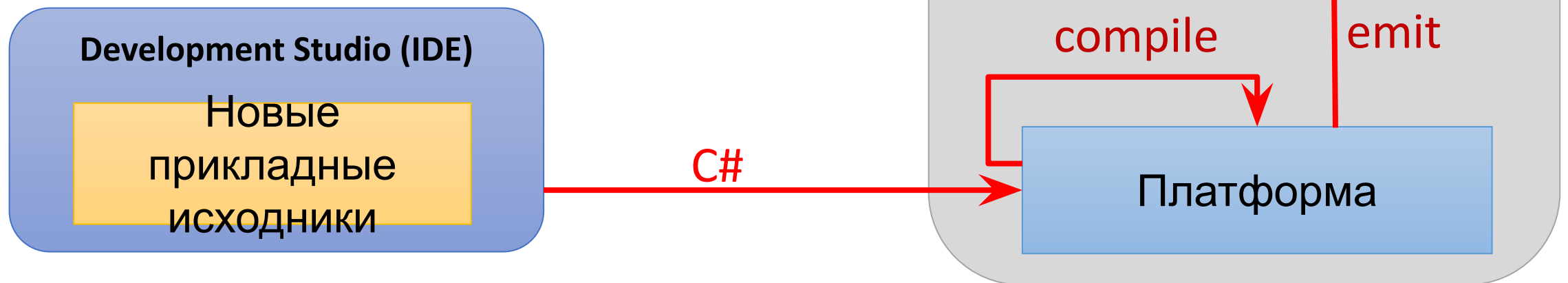
Решения в лоб. Проблемы



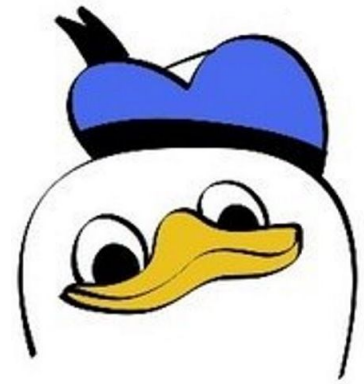
- Не повлияем на уже существующие в памяти объекты.
- Новые объекты, создаваемые через `new` будут ссылаться на старые типы.
- В случае с MEF прикладной код придётся переписать особым образом (атрибуты, интерфейсы).

Emit

- CodeDom – компиляция C# в Assembly сразу в память в AppDomain.
- Emit – генерация IL-кода.



Emit. Проблемы



- Emit работает только с `DynamicAssembly`.
А при компиляции `CodeDom`'а мы получаем обычную `Assembly`.
- Чтобы «завести» динамический класс, нужно вызвать у него `CreateType()`.
Это блокирует его дальнейшие модификации.

Edit and Continue

- Встроенный в Visual Studio хитрый механизм, генерирующий некоторые дельты.
- Общедоступного API нет.
- Даже в самой VS механизм не работает в ряде случаев.

Google

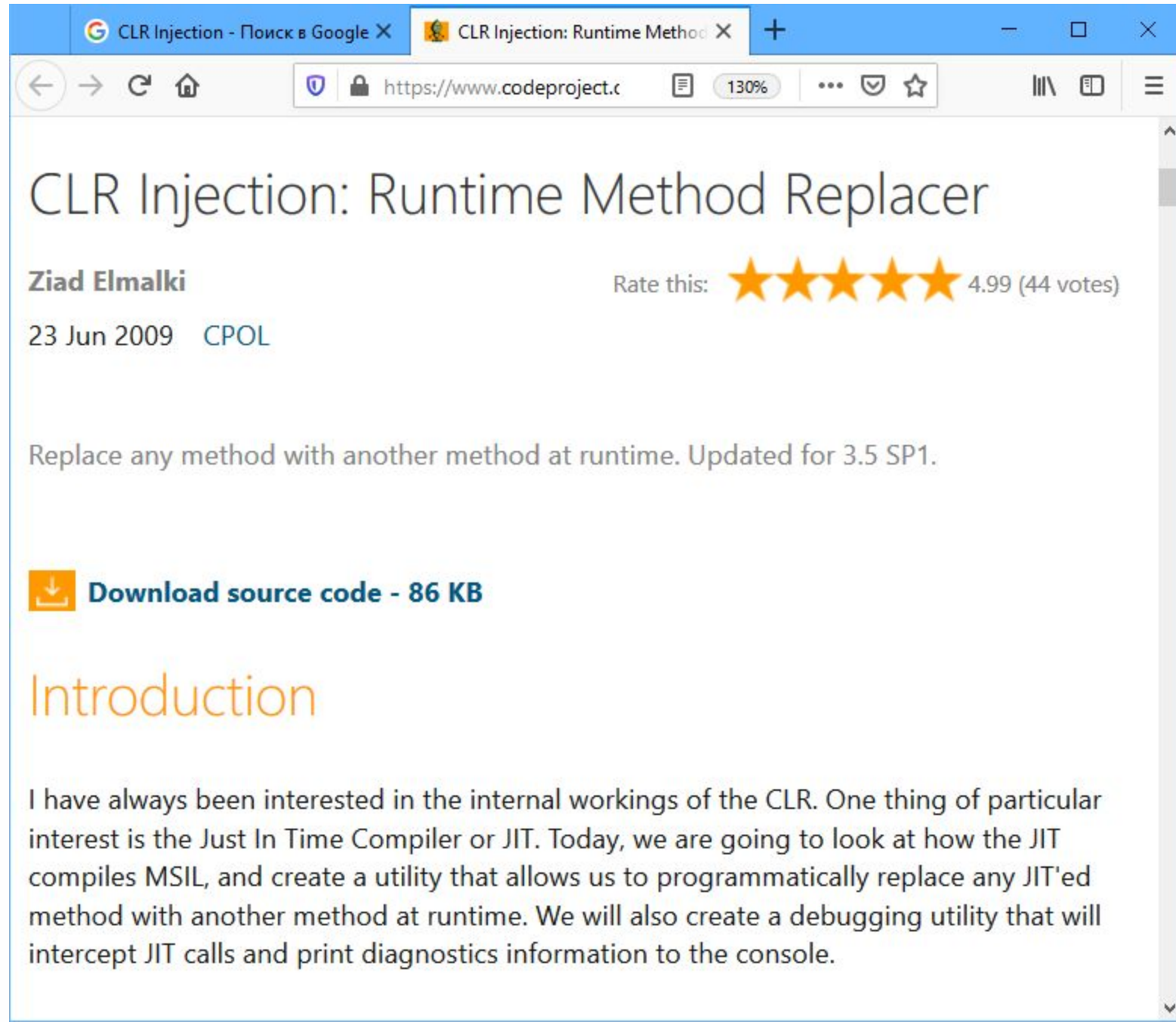


stack
overflow



GitHub

Method Inject v1



The screenshot shows a web browser window with two tabs. The active tab is titled "CLR Injection: Runtime Method Replacer" and the address bar shows the URL "https://www.codeproject.c". The page content includes the title "CLR Injection: Runtime Method Replacer", the author "Ziad Elmalki", a rating of 4.99 (44 votes) with five stars, and the date "23 Jun 2009" with the license "CPOL". A download button is visible with the text "Download source code - 86 KB". The main heading "Introduction" is displayed in orange, followed by a paragraph of text.

CLR Injection - Поиск в Google X CLR Injection: Runtime Method X +


← → ↻ 🏠 🔒 https://www.codeproject.c 📄 130% ⋮ 🛡️ ☆ 🗑️ 📄 ☰

CLR Injection: Runtime Method Replacer

Ziad Elmalki Rate this: ★★★★★ 4.99 (44 votes)

23 Jun 2009 CPOL

Replace any method with another method at runtime. Updated for 3.5 SP1.

 **Download source code - 86 KB**

Introduction

I have always been interested in the internal workings of the CLR. One thing of particular interest is the Just In Time Compiler or JIT. Today, we are going to look at how the JIT compiles MSIL, and create a utility that allows us to programmatically replace any JIT'ed method with another method at runtime. We will also create a debugging utility that will intercept JIT calls and print diagnostics information to the console.

Method Inject v1

Суть – замена указателя на метод.

```
MethodInfo methodToReplace = ... ;  
MethodInfo methodToInject = ... ;
```

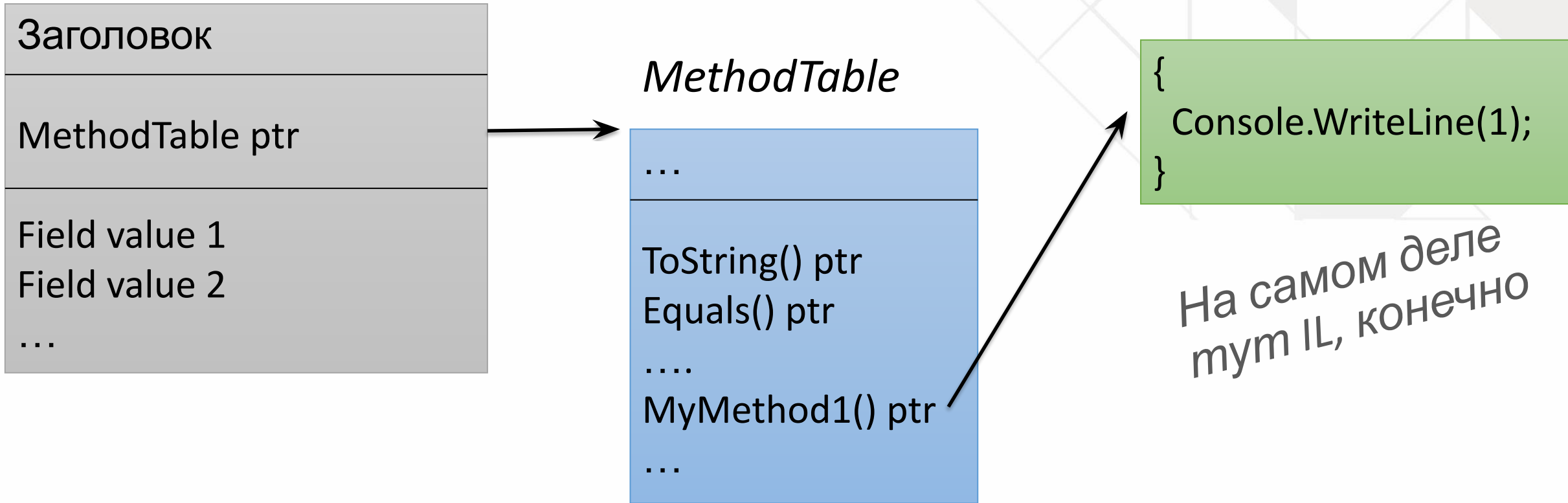
```
unsafe
```

```
{  
    long* target = (long*) methodToReplace.MethodHandle.Value.ToPointer();  
    long* inject = (long*) methodToInject.MethodHandle.Value.ToPointer();  
    *target = *inject;  
}
```

В реальности чуток сложнее, потому что надо учесть x86/x64, Debug/Release.

Как хранятся описания классов в .NET

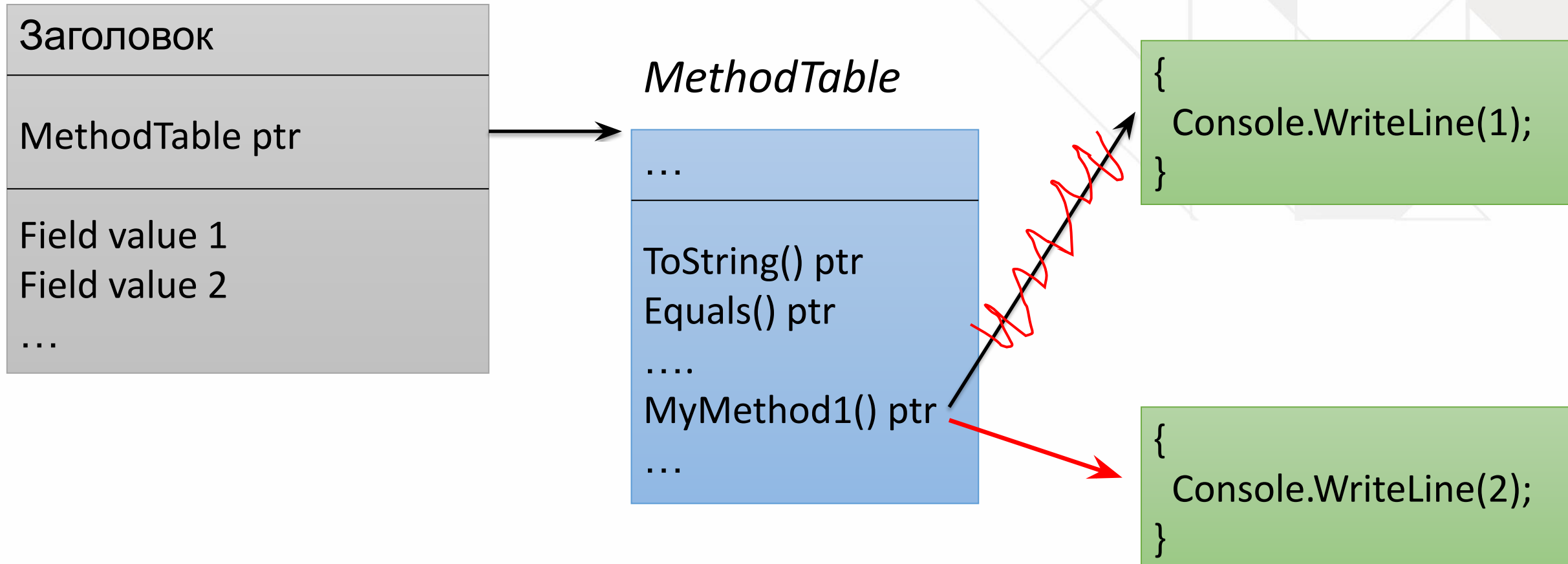
ObjectInstance



На самом деле
тут IL, конечно

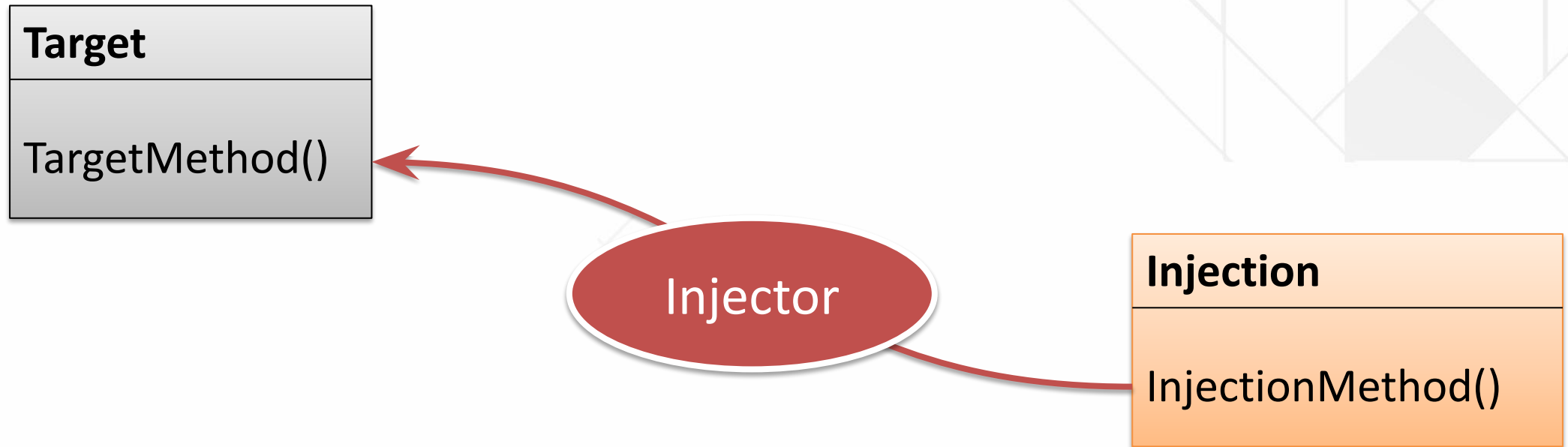
Method Inject v1. Суть

ObjectInstance



Method Inject v1. Проверяем на практике

- Есть два класса...



Method Inject v1.

Проверяем на практике

WinDbg + SOS →

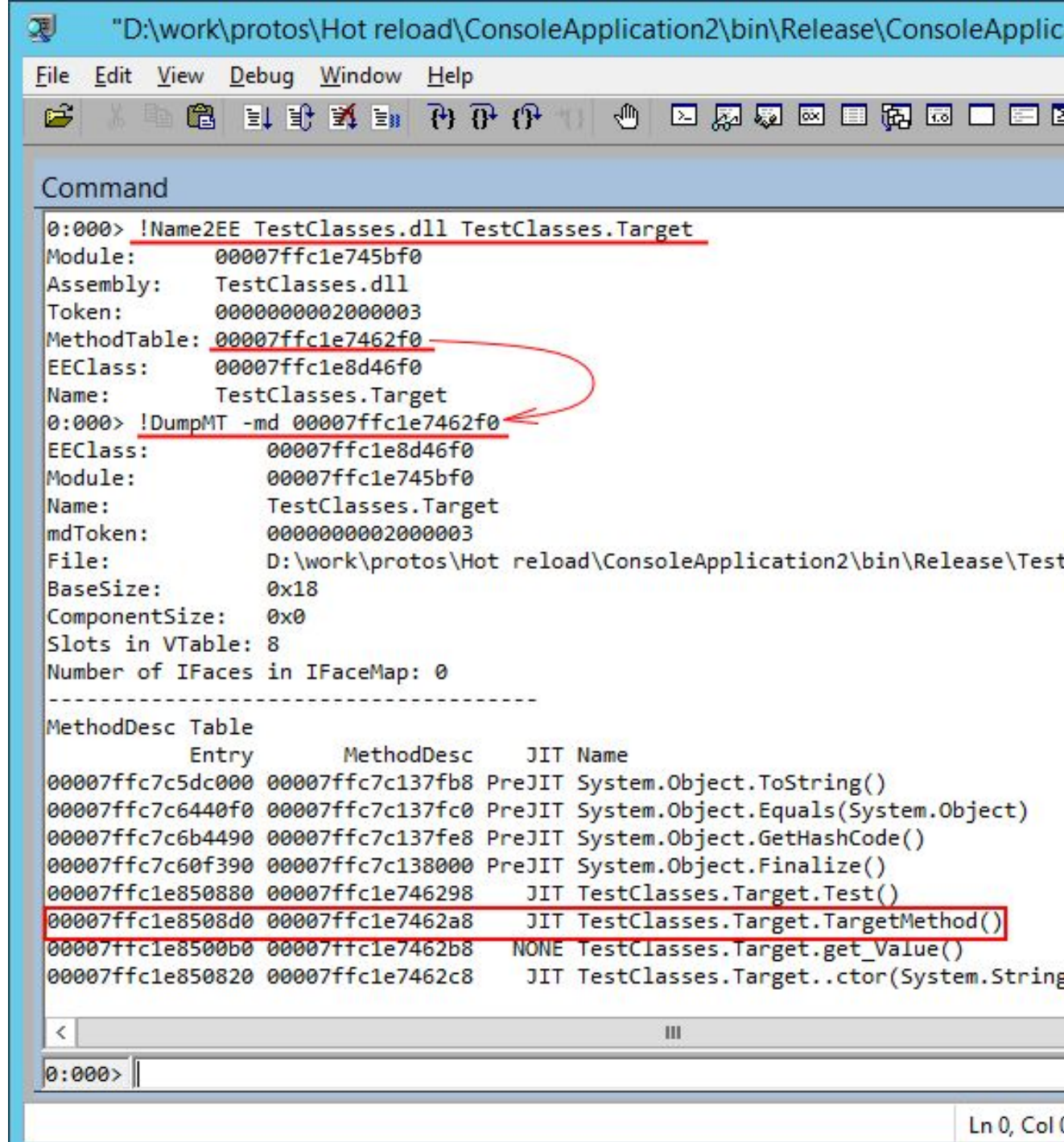
Исходный MethodTable

TestClasses.Target.Test()

TestClasses.Target.TargetMethod()

TestClasses.Target.get_Value()

TestClasses.Target..ctor (System.String)



```
"D:\work\protos\Hot reload\ConsoleApplication2\bin\Release\ConsoleApplic
File Edit View Debug Window Help
Command
0:000> !Name2EE TestClasses.dll TestClasses.Target
Module:      00007ffc1e745bf0
Assembly:    TestClasses.dll
Token:       0000000002000003
MethodTable: 00007ffc1e7462f0
EEClass:     00007ffc1e8d46f0
Name:        TestClasses.Target
0:000> !DumpMT -md 00007ffc1e7462f0
EEClass:     00007ffc1e8d46f0
Module:      00007ffc1e745bf0
Name:        TestClasses.Target
mdToken:     0000000002000003
File:        D:\work\protos\Hot reload\ConsoleApplication2\bin\Release\Test
BaseSize:    0x18
ComponentSize: 0x0
Slots in VTable: 8
Number of IFaces in IFaceMap: 0
-----
MethodDesc Table
          Entry      MethodDesc      JIT Name
00007ffc7c5dc000 00007ffc7c137fb8 PreJIT System.Object.ToString()
00007ffc7c6440f0 00007ffc7c137fc0 PreJIT System.Object.Equals(System.Object)
00007ffc7c6b4490 00007ffc7c137fe8 PreJIT System.Object.GetHashCode()
00007ffc7c60f390 00007ffc7c138000 PreJIT System.Object.Finalize()
00007ffc1e850880 00007ffc1e746298 JIT TestClasses.Target.Test()
00007ffc1e8508d0 00007ffc1e7462a8 JIT TestClasses.Target.TargetMethod()
00007ffc1e8500b0 00007ffc1e7462b8 NONE TestClasses.Target.get_Value()
00007ffc1e850820 00007ffc1e7462c8 JIT TestClasses.Target..ctor(System.String)
0:000> |
```


Method Inject v1.

Проверяем на практике

MethodTable после Inject

TestClasses.Target.Test()

TestClasses.Injection.InjectionMethod()

TestClasses.Target.get_Value()

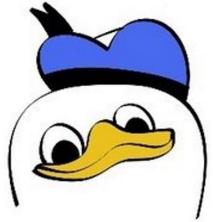
TestClasses.Target..ctor (System.String)

```
"D:\work\protos\Hot reload\ConsoleApplication2\bin\Release\ConsoleApplica
File Edit View Debug Window Help
00007ffc1e850880 00007ffc1e746298 JIT TestClasses.Target.Test()
00007ffc1e8508d0 00007ffc1e7462a8 JIT TestClasses.Target.TargetMethod()
00007ffc1e8500b0 00007ffc1e7462b8 NONE TestClasses.Target.get_Value()
00007ffc1e850820 00007ffc1e7462c8 JIT TestClasses.Target..ctor(System.String)
0:000> g
Breakpoint 0 hit
00007ffc`1e85093e 90 nop
0:000> !DumpMT -md 00007ffc1e7462f0
EEClass: 00007ffc1e8d46f0
Module: 00007ffc1e745bf0
Name: TestClasses.Target
mdToken: 0000000002000003
File: D:\work\protos\Hot reload\ConsoleApplication2\bin\Release\TestC
BaseSize: 0x18
ComponentSize: 0x0
Slots in VTable: 8
Number of IFaces in IFaceMap: 0
-----
MethodDesc Table
Entry MethodDesc JIT Name
00007ffc7c5dc000 00007ffc7c137fb8 PreJIT System.Object.ToString()
00007ffc7c6440f0 00007ffc7c137fc0 PreJIT System.Object.Equals(System.Object)
00007ffc7c6b4490 00007ffc7c137fe8 PreJIT System.Object.GetHashCode()
00007ffc7c60f390 00007ffc7c138000 PreJIT System.Object.Finalize()
00007ffc1e850880 00007ffc1e746298 JIT TestClasses.Target.Test()
00007ffc1e850b50 00007ffc1e746500 JIT TestClasses.Injection.InjectionMethod()
00007ffc1e8500b0 00007ffc1e7462b8 NONE TestClasses.Target.get_Value()
00007ffc1e850820 00007ffc1e7462c8 JIT TestClasses.Target..ctor(System.String)
0:000> |
```

Method Inject v1. Дополнительные работы

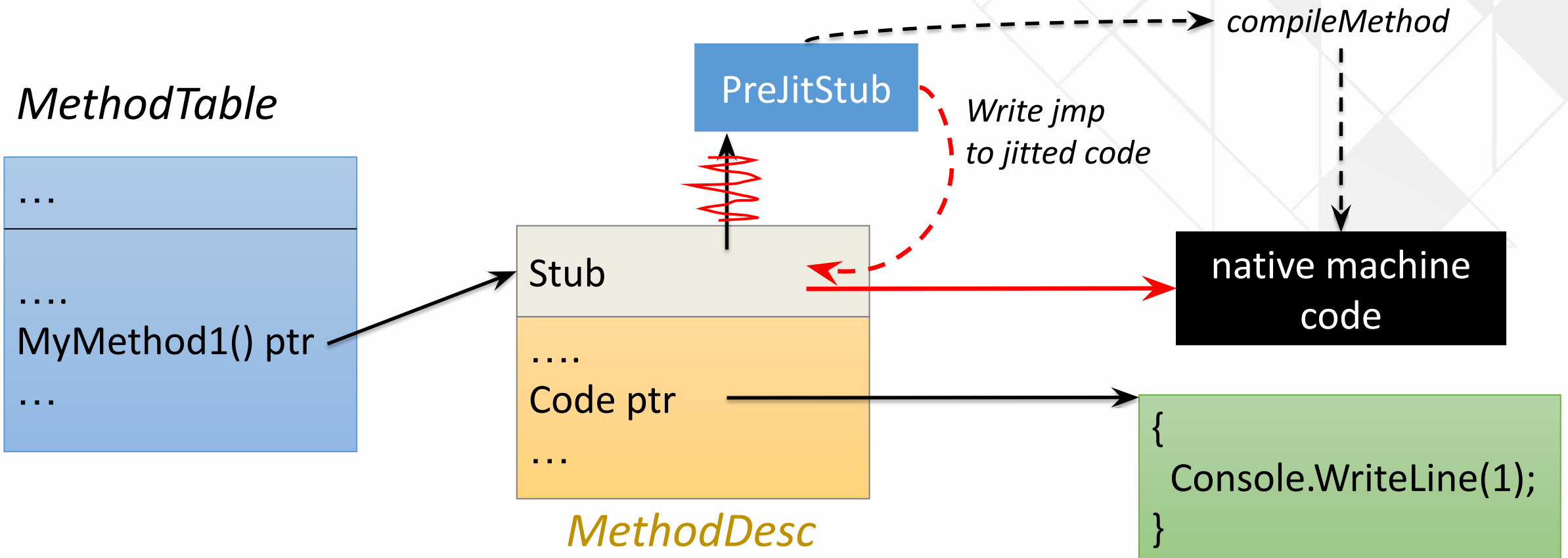
- Сравнить прикладные исходники и найти изменившиеся методы.
- Передать исходники методов на сервер.
- Создать новый класс, засунуть в него методы и скомпилировать в память.
- Найти старый метод и сделать MethodInject на **НОВЫЙ**.

Method Inject v1. ПРОБЛЕМА!

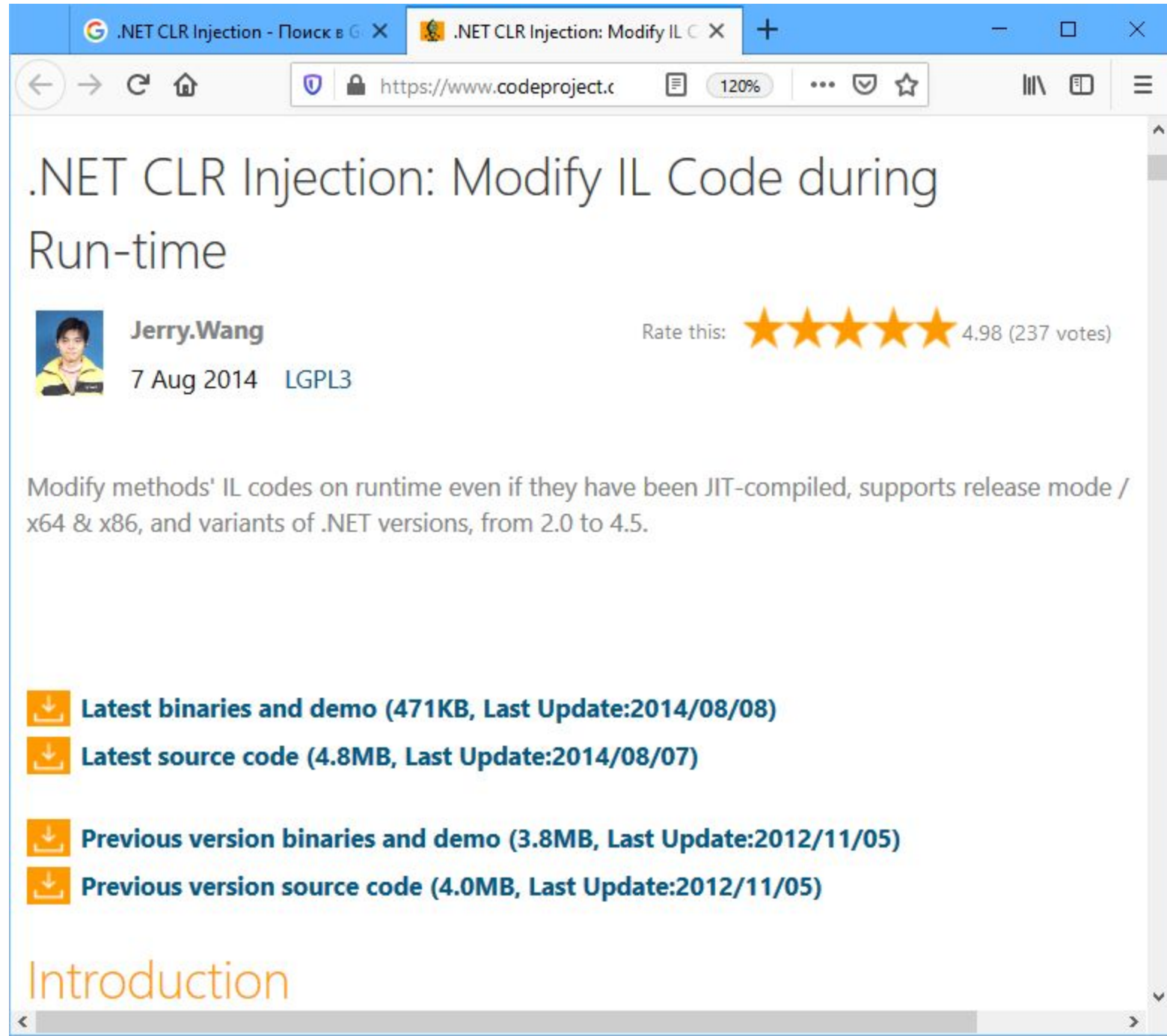


Прекрасно работает
пока мы не вызвали подменяемый метод.

Method Inject v1. ПРОБЛЕМА!



Method Inject v2



The screenshot shows a web browser window with two tabs. The active tab is titled ".NET CLR Injection: Modify IL Code during Run-time" and the address bar shows the URL "https://www.codeproject.c". The page content includes the project title, the author's name "Jerry.Wang" with a profile picture, the date "7 Aug 2014", the license "LGPL3", and a rating of 4.98 (237 votes) represented by five stars. Below this, there is a description: "Modify methods' IL codes on runtime even if they have been JIT-compiled, supports release mode / x64 & x86, and variants of .NET versions, from 2.0 to 4.5." There are four download links, each with a download icon: "Latest binaries and demo (471KB, Last Update:2014/08/08)", "Latest source code (4.8MB, Last Update:2014/08/07)", "Previous version binaries and demo (3.8MB, Last Update:2012/11/05)", and "Previous version source code (4.0MB, Last Update:2012/11/05)". At the bottom, the word "Introduction" is visible in orange text.

.NET CLR Injection: Modify IL Code during Run-time

Jerry.Wang
7 Aug 2014 LGPL3

Rate this: ★★★★★ 4.98 (237 votes)

Modify methods' IL codes on runtime even if they have been JIT-compiled, supports release mode / x64 & x86, and variants of .NET versions, from 2.0 to 4.5.

- Latest binaries and demo (471KB, Last Update:2014/08/08)
- Latest source code (4.8MB, Last Update:2014/08/07)
- Previous version binaries and demo (3.8MB, Last Update:2012/11/05)
- Previous version source code (4.0MB, Last Update:2012/11/05)

Introduction

Method Inject v2

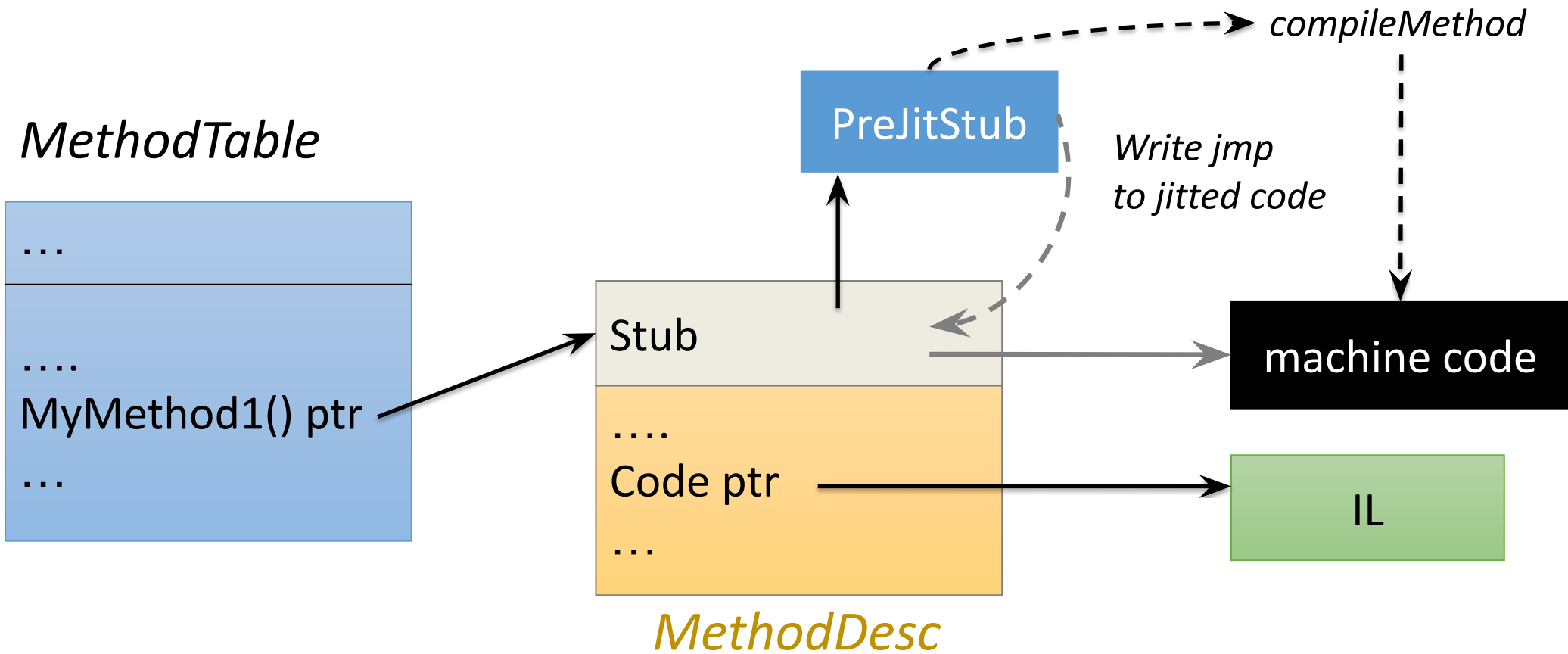
```
public static class InjectionHelper
{
    ...

    public static void Initialize()

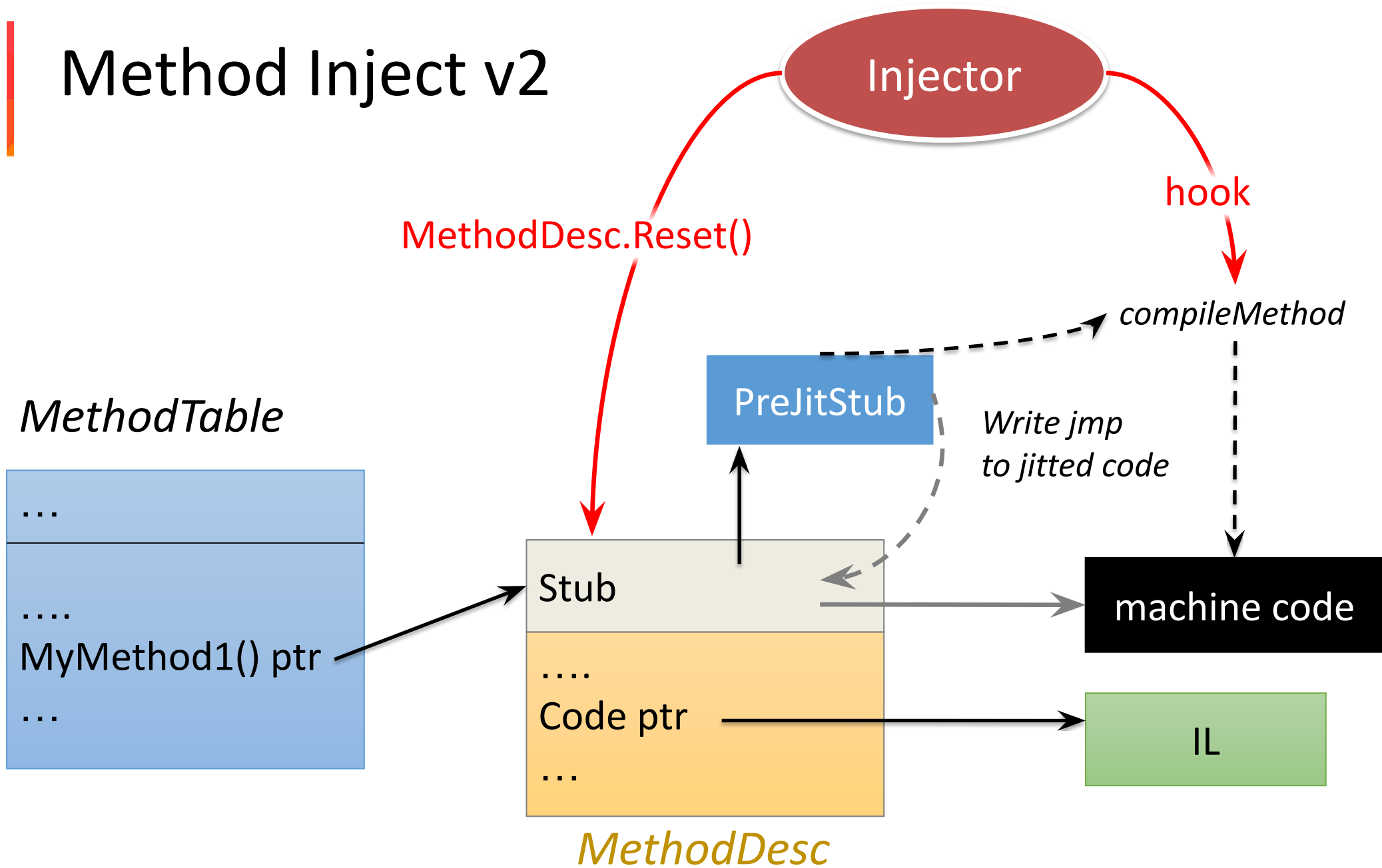
    public static void UpdateILCodes(MethodInfo method, byte[] ilCodes)

    ...
}
```

Method Inject v2



Method Inject v2



Method Inject v2. Проблемки

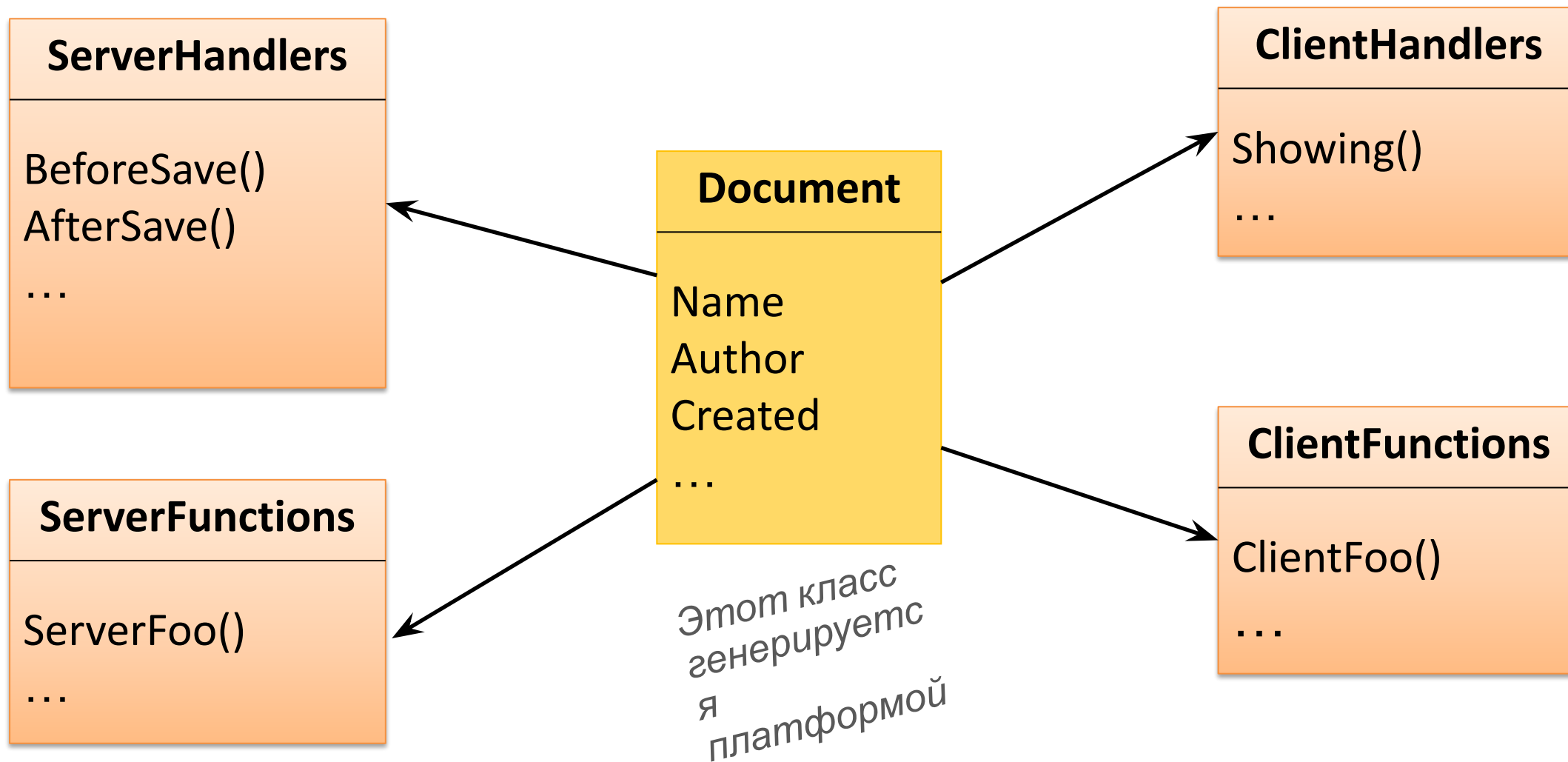
- Нормално заработало только на .NET Framework 2.0 - 4.0
- Inline маленьких методов.
- Лезем внутрь CLR.
Слишком рискованно для продакшена.

Давайте вспомним, с чего мы начали

- Есть платформа.
- Она исполняет прикладной код.
- Мы хотим подменять прикладной код.

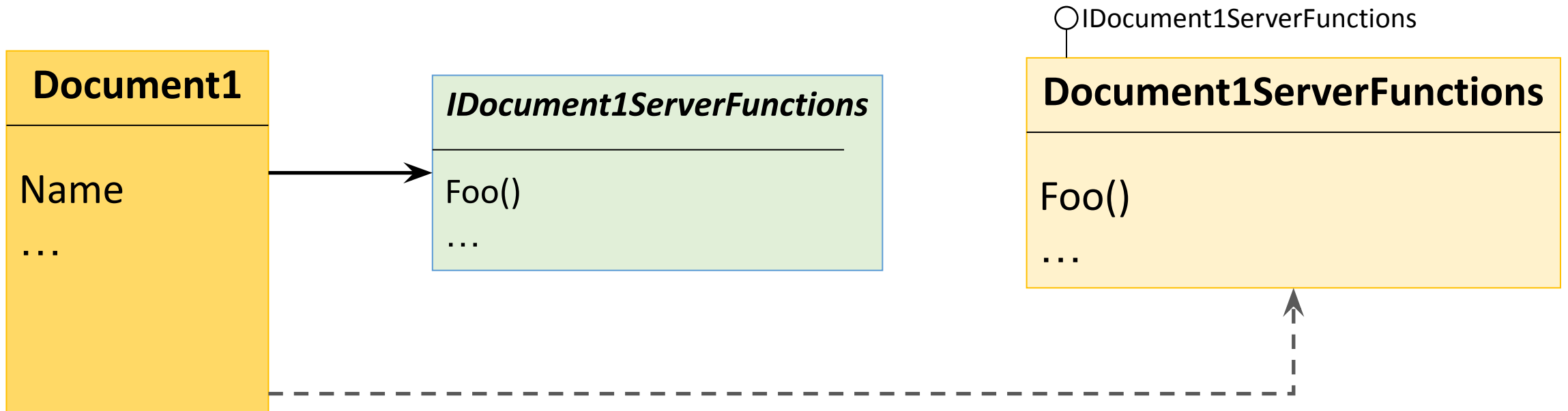


Что за прикладной код?

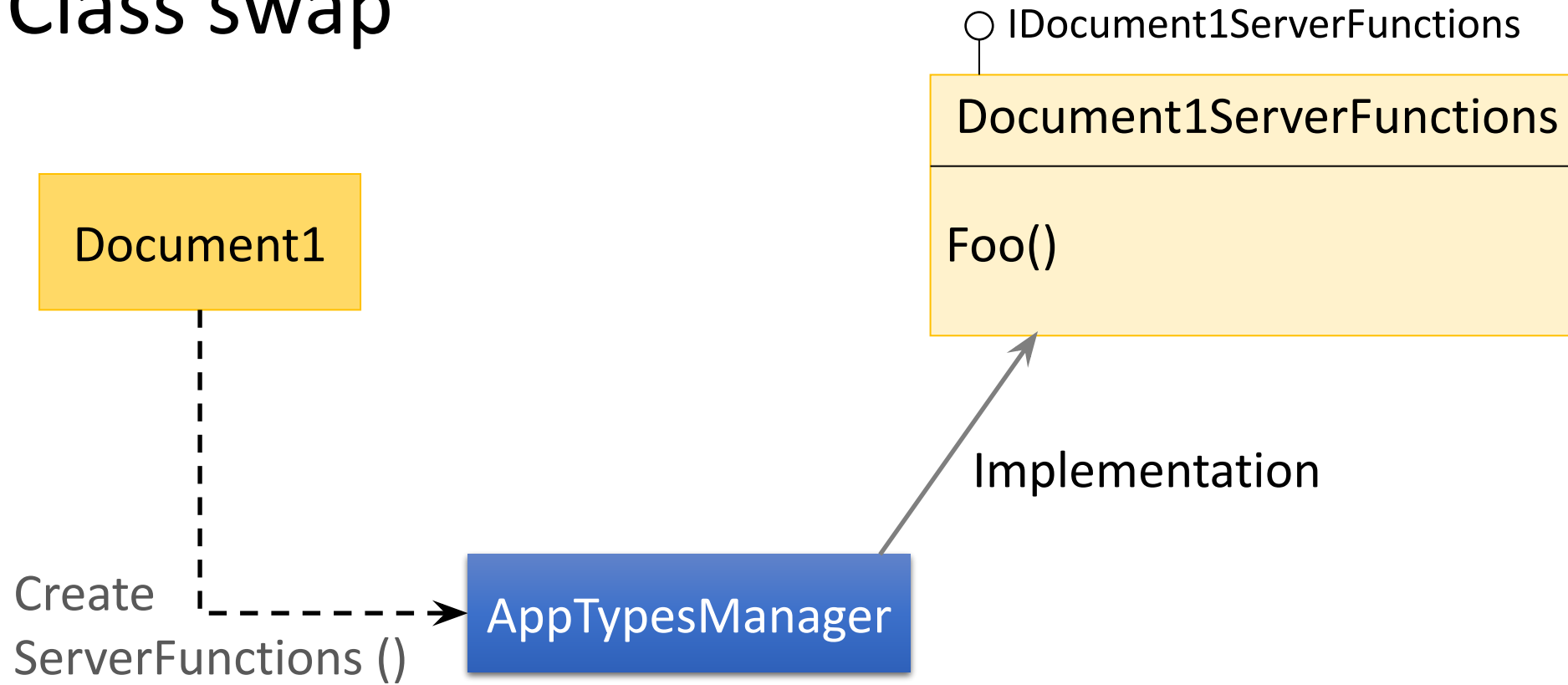


Class swap

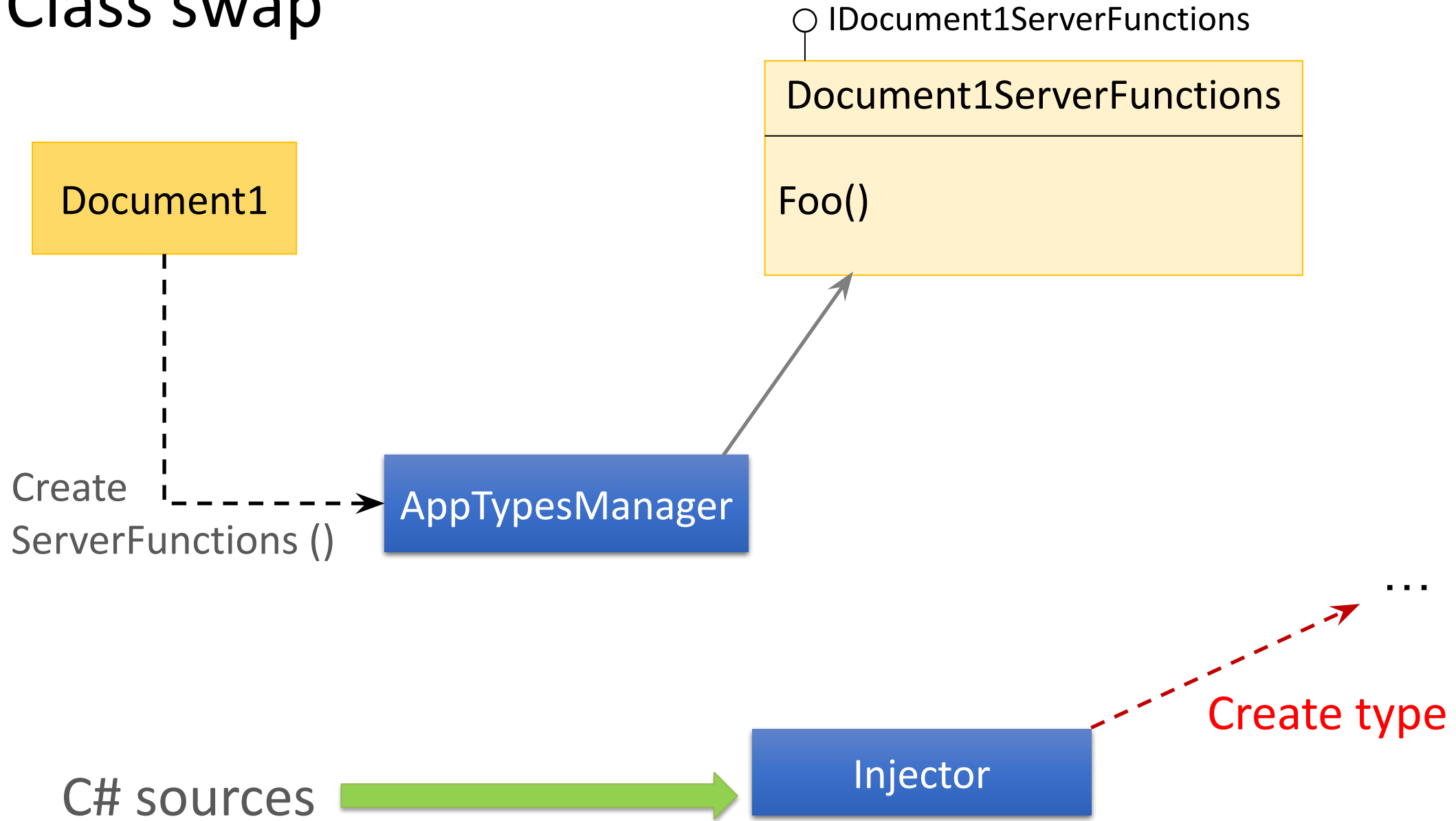
- Платформенный код создаёт приложенияльные объекты (Dependency Injection).
- Не возимся с методами – заменяем сразу весь объект.



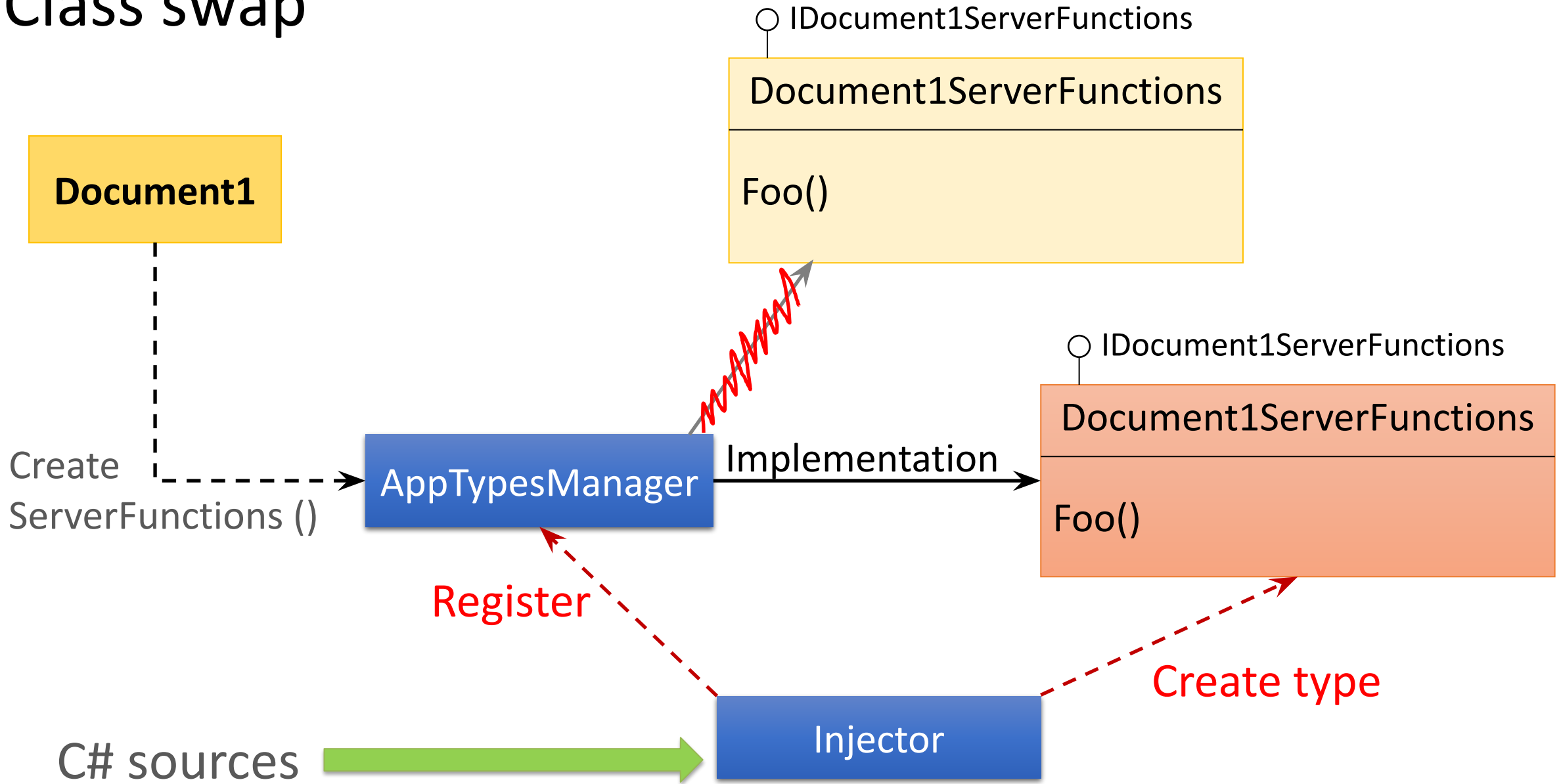
Class swap



Class swap



Class swap



Заключение

- В продакшен это не пошло :(Деплой мы ускорили иначе – оптимизации.
- Исследования – круто.
- Погружение в детали инструмента – интересно.



()

- .net core
- JIT disable
- Custom .NET CLR





THE END



Асадуллин

Тимур
Directum,

Уфа
timonik@bk.ru



ССЫЛОЧКИ

- MethodInject v1
<https://www.codeproject.com/Articles/37549/CLR-Injection-Runtime-Method-Replacer>
- MethodInject v2
<https://www.codeproject.com/Articles/463508/NET-CLR-Injection-Modify-IL-Code-during-Run-time>
- EasyHook
<https://easyhook.github.io>
- Отладка WinDbg + SOS
<https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/debugging-managed-code>
- .net core
«How to use and debug assembly unloadability in .NET Core»
<https://docs.microsoft.com/ru-ru/dotnet/standard/assembly/unloadability>