Разговоры о динамической кодогенерации, или «Тёмная сторона CIL-а»

dmitry.garavsky@devexpress.com

#Код. Такой разный вокруг.

Код не рождается сам… хотя, это как смотреть

Кодогенерация – это специализированный инструмент для эффективного решения определенного круга задач.

#Задача. Исходные данные.

```
// Scenario we NEED to process
Action someUserScenario = () => {
    Operations.One();
    Operations.Two();
    Operations.Three();
    Operations.Four(); // Throw Not Supported
    Operations.Five();
};
```

#Код как источник метаинформации.

Абсолютно новый уровень возможностей

Код, сам по себе является мощным источником данных. Особенно в том случае, когда он уже скомпилирован.

#Задача. Идеальный результат.

```
// Our dream ;-)
Action idealScenario = () => {
    Task.WaitAll(new Task[] {
        Task.Run(new Action(Operations.One)),
        Task.Run(new Action(Operations.Two)),
        Task.Run(new Action(Operations.Three)),
        Task.Run(new Action(Operations.Five)),
    });
};
```

#Подходы. Выберем подходящий.

Что надо учитывать при выборе подхода

- Способ преобразования.
- Момент выполнения преобразования.

#Подходы. Способы преобразования.

Что и как преобразуется

- `new HighLevelCode(dataOrMetadata)`
- `HighLevelCode.Inject(lowLevelCode)`
- `new LowLevelCode(dataOrMetadata)`

#Подходы. Момент преобразования.

Когда выполняется преобразование

- По требованию(статическая генерация)
- Compile-Time генерация
- Run-Time(динамическая генерация)

#Подходы. Статическая генерация.

Код генерируется по требованию

- Конвертация (C#-to-VB)
- CodeDom решения(Design-Time)
- T4 решения(DSL)

#Подходы. Compile-Time генерация.

Код генерируется по требованию

- Препроцессинг
- Постпроцессинг
- Внедрение в процесс компиляции

#Подходы. Compile-Time Preprocess.

Генерация платформенно-зависимого кода

```
// Model
public enum DiagramCommand {
    Undo,
    Redo,
    //...
}

// Common
public partial class DiagramCommands {
    // ...
}
```

#Подходы. Compile-Time Preprocess.

Генерация платформенно-зависимого кода

```
// WPF
partial class DiagramCommands {
    public DiagramCommands(DiagramControl diagram)
        : base(diagram) {
        Undo = CreateCommand(() => Undo(), () => CanUndo());
        Redo = CreateCommand(() => Redo(), () => CanRedo());
        //...
    }
    public ICommand Undo { get; private set; }
    public ICommand Redo { get; private set; }
}
```

#Подходы. Compile-Time Preprocess.

Чистка исходных файлов до компиляции

```
class Foo {
#if DEBUGTEST
    internal
#endif
    int value;
    [SomeAttribute(Value = 42)] // RELEASE_REMOVE (for debugging)
    public Foo(int value) {
        this.value = value; // some comment
    }
}
```

#Подходы. Compile-Time Preprocess.

Адаптация исходных файлов под целевую платформу

```
<!—WPF DynamicResource->
<ResourceDictionary
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
    <ResourceDictionary.MergedDictionaries>
        <ResourceDictionary Source="/xxx.v16.1;component/Themes/DeepBlue/Colors.xaml" />
        <ResourceDictionary Source="/xxx.v16.1;component/Themes/DeepBlue/Controls.xaml" />
        <!--Generic-->
        <ResourceDictionary Source="/xxx.v16.1;component/Themes/Generic/Toolbox.xaml" />
        <ResourceDictionary Source="/xxx.v16.1;component/Themes/Generic/Adorners.xaml" />
        <ResourceDictionary Source="/xxx.v16.1;component/Themes/Generic/Controls.xaml" />
    </ResourceDictionary.MergedDictionaries>
</ResourceDictionary>
```

#Подходы. Compile-Time Postprocess.

Аспектно-Ориентированное Программирование

```
// PostSharp (AOP)
// https://www.postsharp.net/

[NotifyPropertyChanged]
class CustomerViewModel {
    public Customer Customer { get; private set; }
}
```

#Подходы. Compiler-Intercept.

Внедрение в процесс сборки с целью трансформации кода

```
// CCICsharp (Common Compiler Infrastructure)
// http://ccimetadata.codeplex.com

class Foo {
    [WeakLazy]
    public object Value {
        get { return Environment.TickCount; }
    }
}
```

#Подходы. Runtime генерация.

Все дело в низкоуровневых манипуляциях

Основной объединяющий признак подходов этого типа:
- работа с **метаданными** сборки (Reflection)
- работа с **байт-кодом** сборки (Emit)

#Подходы. Runtime генерация.

Работа через Microsoft IL Generator API

```
// DynamicMethod/MethodBuilder

var ILGen = method.GetILGenerator();
ILGen.Emit(OpCodes.Ldarg_0);
ILGen.Emit(OpCodes.Ldarg_1);
ILGen.Emit(OpCodes.Add);
ILGen.Emit(OpCodes.Ret);
```

#Подходы. Runtime генерация.

Работа через Microsoft Expressions API

```
// Expression
Func<int, int, int> CreateSum() {
    var a = Expression.Parameter(typeof(int), "a");
    var b = Expression.Parameter(typeof(int), "b");
    return Expression.Lambda<Func<int, int, int>>(
            Expression.Add(a, b),
        a, b).Compile();
}
```

#Подходы. Runtime генерация.

Альтернативные реализации

```
// Mono.Cecil
var assembly = AssemblyDefinition.ReadAssembly(assemblyPath);
var writeLineRef = assembly.MainModule.Import(writeLineMethod);
//
method.Body.Instructions.Insert(0,
    Instruction.Create(OpCodes.Ldstr, "Hello!"));
method.Body.Instructions.Insert(1,
    Instruction.Create(OpCodes.Call, writeLineRef));
```

#Задача. Стратегия решения.

- Читаем CIL сценария и входящих в него элементов
- Выделяем и анализируем CIL используя паттерны
- Генерируем результат в виде CIL

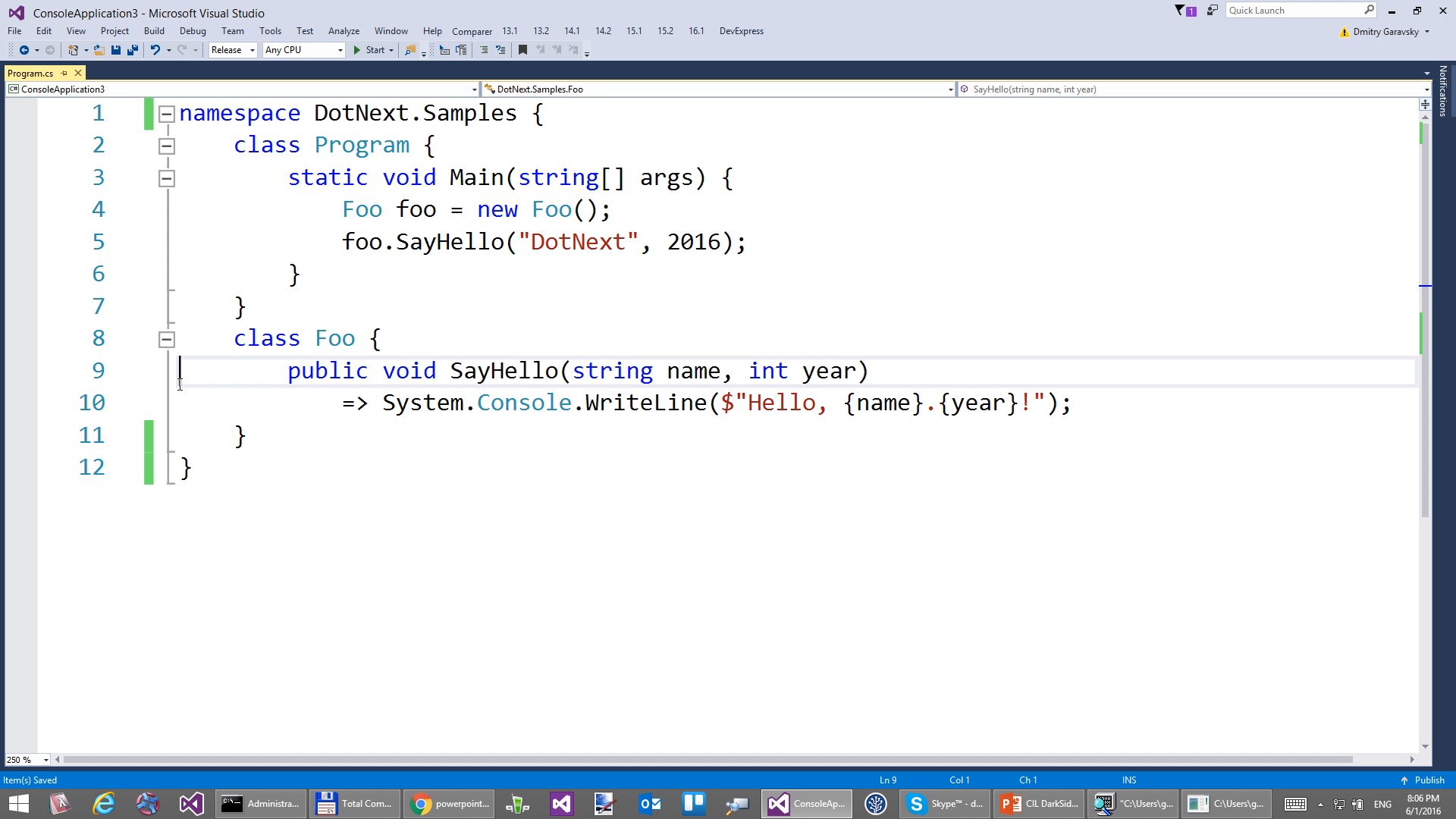
#Инструменты. Must Have

Хороший инструмент - половина работы.

- IL-декомпилятор
- Отладчик (WinDbg+SOS)
- Benchmarking tool.

#Инструменты. Учимся использовать.

```
// C# 6.0
// Bodied-method with string-interpolation

class Foo {
    void SayHello(string name, int year) =>
        Console.WriteLine($"Hello, {name}.{year}!");
}
```


ConsoleApplication3 - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help Comparer 13.1 13.2 14.1 14.2 15.1 15.2 16.1 DevExpress
Release
Any CPU
Start
Program.cs
ConsoleApplication3
DotNext.Samples.Foo
SayHello(string name, int year)
1 namespace DotNext.Samples {
2 class Program {
3 static void Main(string[] args) {
4 Foo foo = new Foo();
5 foo.SayHello("DotNext", 2016);
6 }
7 }
8 class Foo {
9 public void SayHello(string name, int year)
10 => System.Console.WriteLine($"Hello, {name}.{year}!");
11 }
12 }
250 %
Item(s) Saved
Ln 9
Col 1
Ch 1
INS
Publish
8:06 PM
6/1/2016

#Инструменты. DIY: IL-Reader.

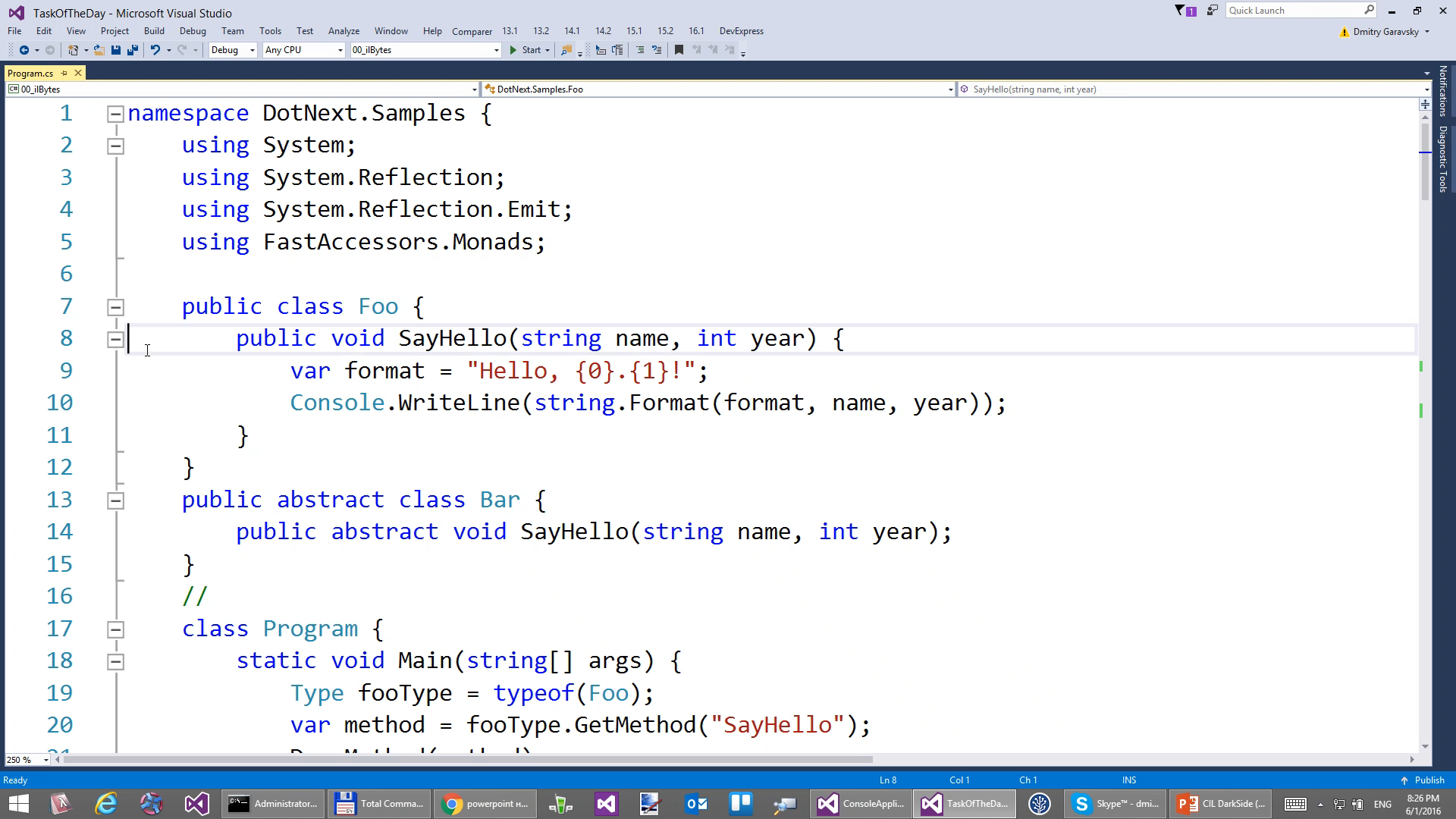
Как скрафтить уникальный инструмент

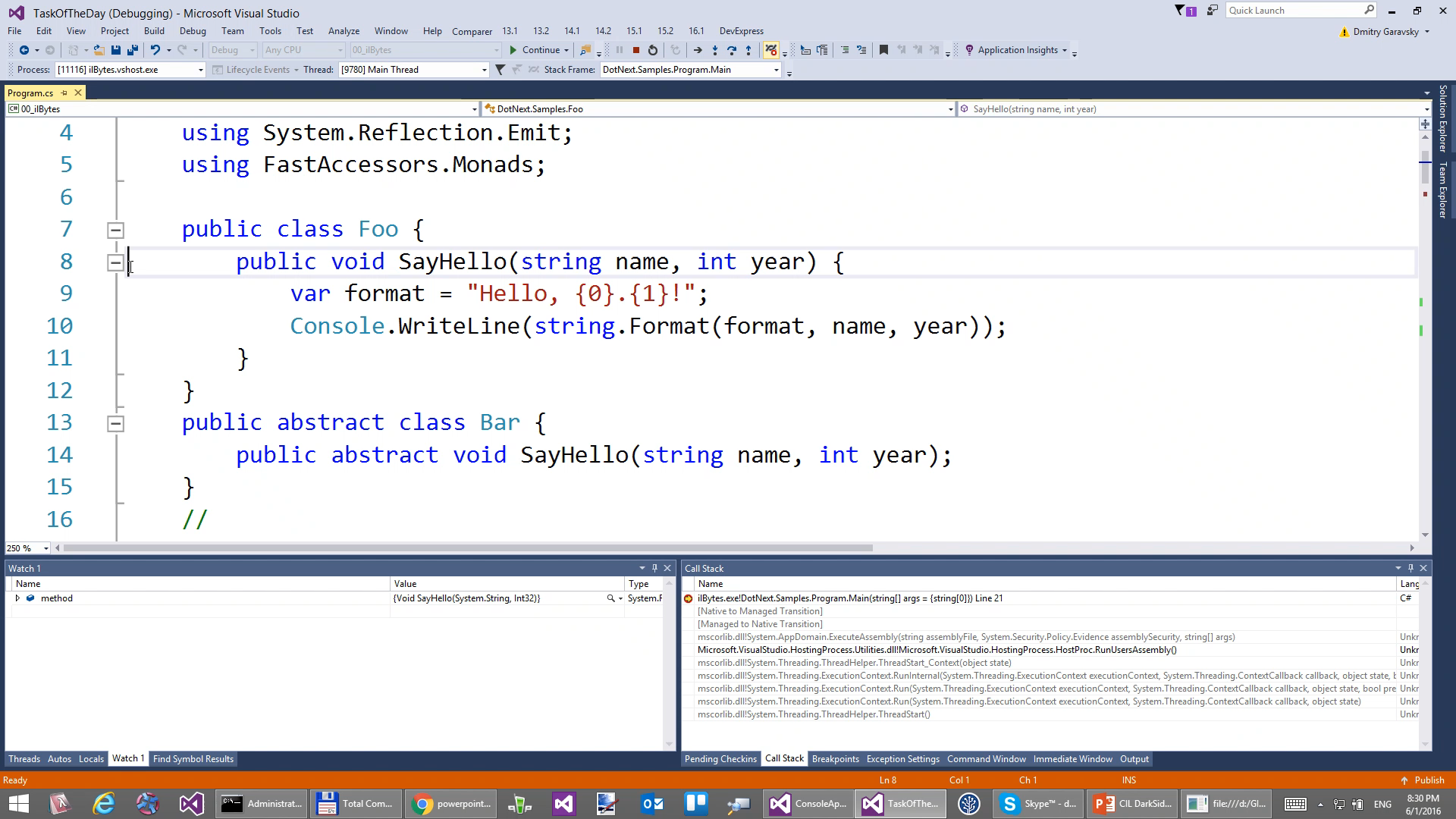
```
// Method
var mBody = method.GetMethodBody();
byte[] ilBytes = mBody.GetILAsByteArray();

int tSig = mBody.LocalSignatureMetadataToken;
byte[] locals = module.ResolveSignature(tSig);
```

#Инструменты. DIY: IL-Reader.

Как скрафтить уникальный инструмент

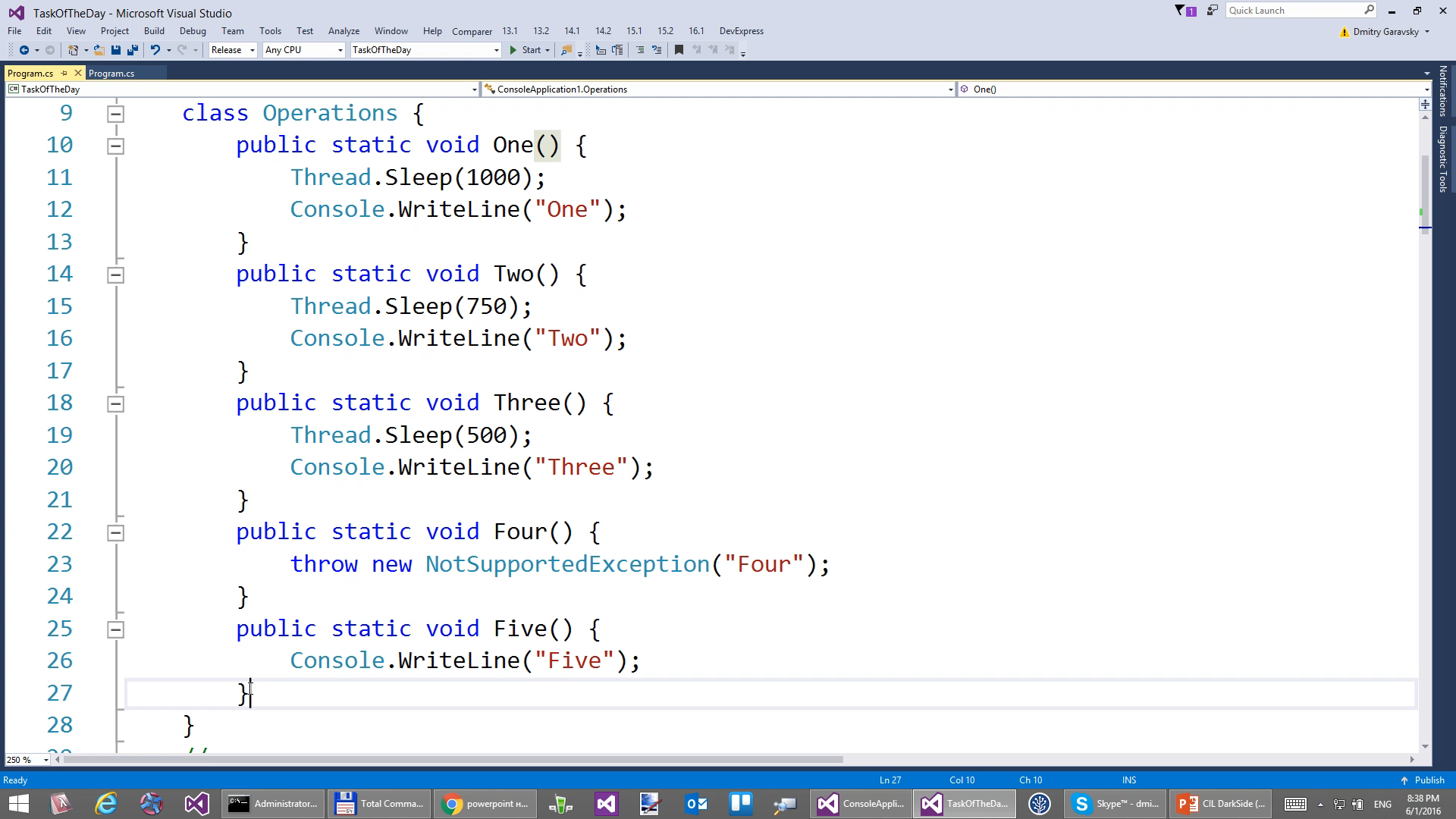
```
// DynamicMethod (.@f==GetFieldValue)
var resolver = method
    .@f("m_resolver");
byte[] ilBytes = (byte[])resolver
    .@f(resolver.GetType(), "m_code");
byte[] locals = (byte[])resolver
    .@f(resolver.GetType(), "m_localSignature");
```


TaskOfTheDay - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help Comparer 13.1 13.2 14.1 14.2 15.1 15.2 16.1 DevExpress
Debug Any CPU 00_ilBytes Start
Program.cs
00_ilBytes DotNext.Samples.Foo SayHello(string name, int year)
namespace DotNext.Samples {
using System;
using System.Reflection;
using System.Reflection.Emit;
using FastAccessors.Monads;
public class Foo {
public void SayHello(string name, int year) {
var format = "Hello, {0}.{1}!";
Console.WriteLine(string.Format(format, name, year));
}
}
public abstract class Bar {
public abstract void SayHello(string name, int year);
}
//
class Program {
static void Main(string[] args) {
Type fooType = typeof(Foo);
var method = fooType.GetMethod("SayHello");
250 %
Ready
Ln 8 Col 1 Ch 1 INS
Publish
Administrator...
Total Comma...
powerpoint н...
ConsoleAppli...
TaskOfTheDa...
Skype™ - dmi...
CIL DarkSide (...
ENG
8:26 PM
6/1/2016


TaskOfTheDay (Debugging) - Microsoft Visual Studio
Program.cs
00_ilBytes
DotNext.Samples.Foo
SayHello(string name, int year)
using System.Reflection.Emit;
using FastAccessors.Monads;
public class Foo {
public void SayHello(string name, int year) {
var format = "Hello, {0}.{1}!";
Console.WriteLine(string.Format(format, name, year));
}
}
public abstract class Bar {
public abstract void SayHello(string name, int year);
}
//
Watch 1
Name
Value
Type
method
{Void SayHello(System.String, Int32)}
Call Stack
ilBytes.exe!DotNext.Samples.Program.Main(string[] args = {string[0]}) Line 21
[Native to Managed Transition]
[Managed to Native Transition]
Threads
Autos
Locals
Watch 1
Find Symbol Results
Pending Checkins
Call Stack
Breakpoints
Exception Settings
Command Window
Immediate Window
Output
Ready
Publish

#Задача. Решение.

```
// https://github.com/DmitryGaravsky/ILReader
var reader = GetReader(scenario.Method);
while(Call.Instance.Match(reader)) {
    var nested = GetReader(call.Method);
    if(NotSupported.Instance.Match(nested))
        // Detected
    }
}
```


TaskOfTheDay - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help Comparer 13.1 13.2 14.1 14.2 15.1 15.2 16.1 DevExpress
Release Any CPU TaskOfTheDay Start
Program.cs Program.cs
TaskOfTheDay ConsoleApplication1.Operations One()
class Operations {
public static void One() {
Thread.Sleep(1000);
Console.WriteLine("One");
}
public static void Two() {
Thread.Sleep(750);
Console.WriteLine("Two");
}
public static void Three() {
Thread.Sleep(500);
Console.WriteLine("Three");
}
public static void Four() {
throw new NotSupportedException("Four");
}
public static void Five() {
Console.WriteLine("Five");
}
}
Ready Ln 27 Col 10 Ch 10 INS Publish

#Темная сторона. Основы.

Что можно делать в CIL, чего нельзя сделать в C#

- Полный контроль за вызовом методов.
- Полный контроль за исключениями.

#Темная сторона. Сила опкодов.

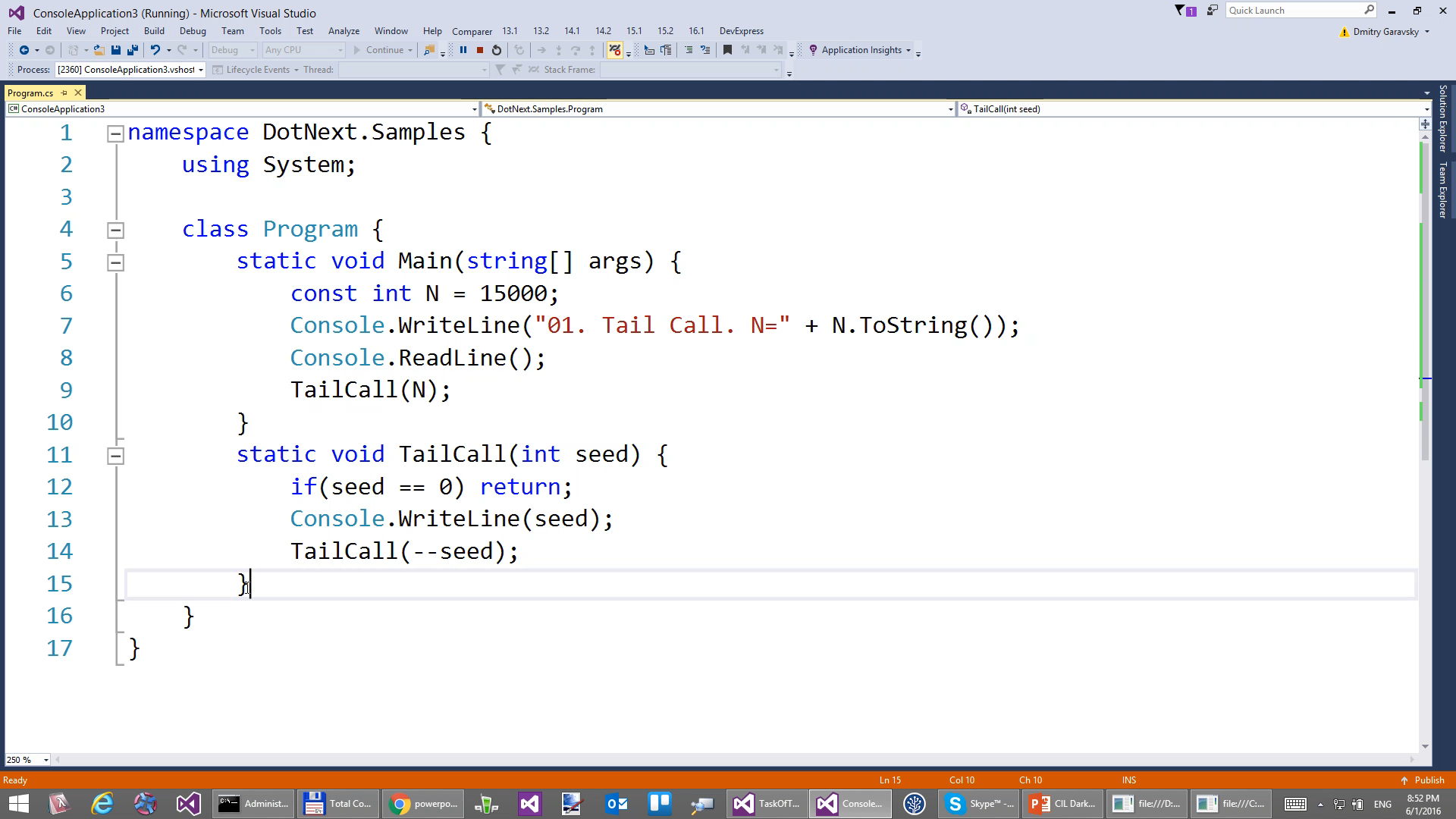
Контроль за вызовом методов

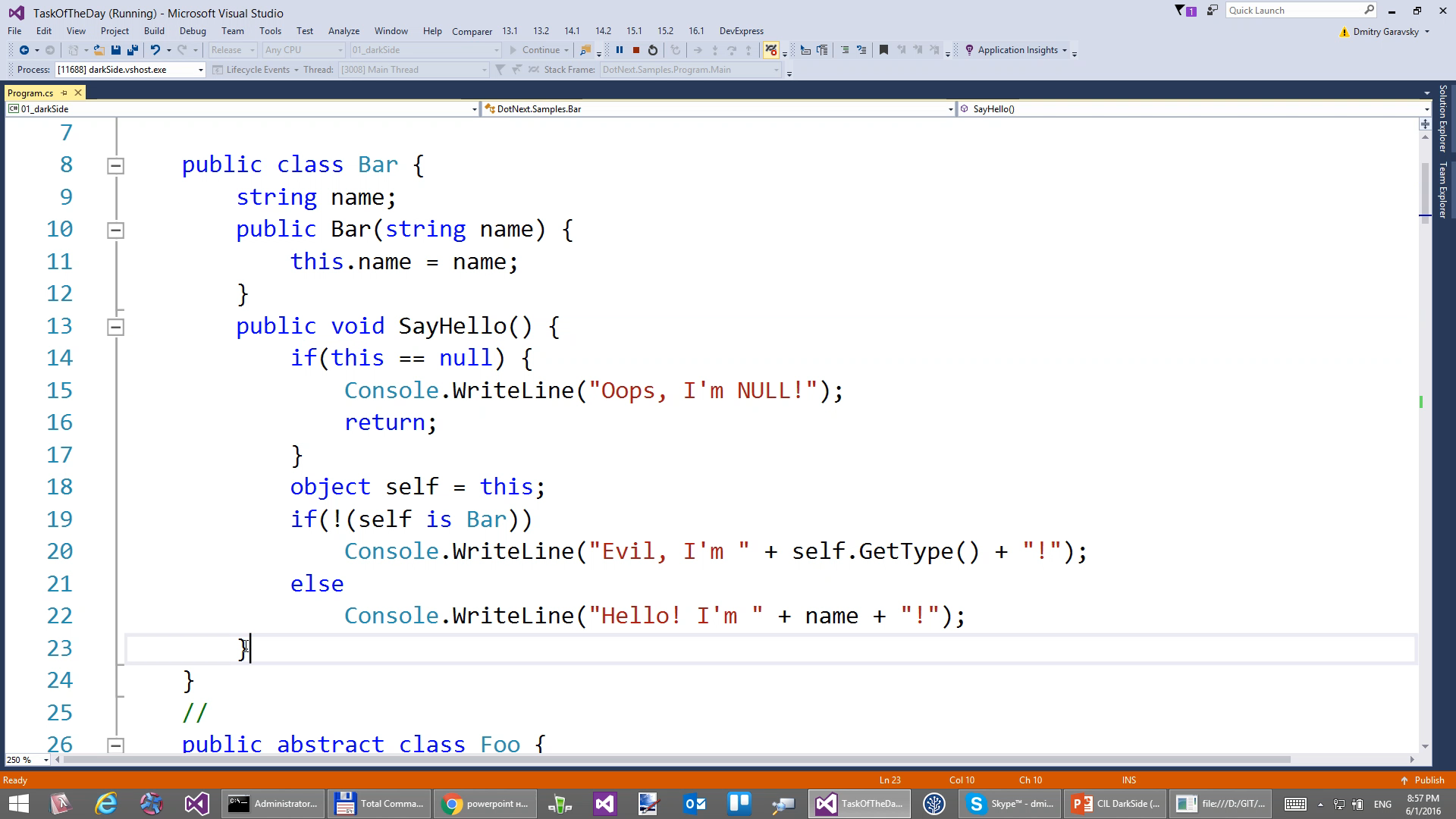
```
// Method Call (C#)
target.Method();

// Method Call (IL)
ILGen.Emit(OpCodes.Ldarg_1); // target
ILGen.Emit(OpCodes.Call, method);
```

#Темная сторона. Сила опкодов.

Контроль за вызовом методов

```
// Tail Call (C#)
static void TailCall(int seed) {
    if(seed > 0) TailCall(--seed);
}
// Tail Call (IL)
ILGen.Emit(OpCodes. Tailcall); // prefix
ILGen.Emit(OpCodes.Call, method);
```


ConsoleApplication3 (Running) - Microsoft Visual Studio
Program.cs
ConsoleApplication3
DotNext.Samples.Program
TailCall(int seed)
namespace DotNext.Samples {
using System;
class Program {
static void Main(string[] args) {
const int N = 15000;
Console.WriteLine("01. Tail Call. N=" + N.ToString());
Console.ReadLine();
TailCall(N);
}
static void TailCall(int seed) {
if(seed == 0) return;
Console.WriteLine(seed);
TailCall(--seed);
}
}
}


TaskOfTheDay (Running) - Microsoft Visual Studio
Program.cs
01_darkSide
DotNext.Samples.Bar
SayHello()
public class Bar {
string name;
public Bar(string name) {
this.name = name;
}
public void SayHello() {
if(this == null) {
Console.WriteLine("Oops, I'm NULL!");
return;
}
object self = this;
if(!(self is Bar))
Console.WriteLine("Evil, I'm " + self.GetType() + "!");
else
Console.WriteLine("Hello! I'm " + name + "!");
}
}
//
public abstract class Foo {
Ready
Ln 23
Col 10
Ch 10
INS

#Темная сторона. Сила опкодов.

Исключения под контролем

```
//.try
var @try = ILGen.BeginExceptionBlock();
//
ILGen.Emit(OpCodes.Leave_S, @ret);
//.fault
ILGen.BeginFaultBlock();
// only when exception thrown
ILGen.EndExceptionBlock();
```

#Темная сторона. Основы.

Что можно делать в CIL, чего нельзя сделать в C#

- Контроль за изготовлением типа.
- Работа с типами данных за рамками ограничений.

#Темная сторона. Практические уроки.

Пробуем извлекать практическую пользу

- Кастинг в обход иерархии.
- Сериализация структур «на стероидах».
- Generic вычисления.

#Темная сторона. Практические уроки.

Кастинг в обход иерархии

```
// Method Call (C#)
A a = Caster<A>.Default.As(new B());

// T Caster.As(object obj) (IL)
ilGen.Emit(OpCodes.Ldarg_1);
ilGen.Emit(OpCodes.Ret);
```

#Темная сторона. Практические уроки.

Кастинг в обход иерархии (практическая польза)

```
List<A> aList = new List<A>() { new B() };
// Impossible?
// List<B> bList = (List<B>)(object)aList

List<B> bList=Caster<List<B>>.Default.As(aList);
```

#Темная сторона. Практические уроки.

Кастинг в обход иерархии(производительность)

Method	Median	StdDev
As	5.2365 ns	0.2410 ns
Cast	5.2857 ns	0.1746 ns
CastDelegate	2.7783 ns	0.0912 ns
Caster	1.7604 ns	0.0581 ns

#Темная сторона. Практические уроки.

Сериализация структур «на стероидах».

```
 // Serialize/Deserialize(C#)
Point pt1 = new Point() { x = 5, y = 17 };
byte[] bytes = new byte[]
   { pt1.x, pt2.y };
Point pt2 = new Point()
   { x = bytes[0], y = bytes[1] };
```

#Темная сторона. Практические уроки.

Сериализация структур «на стероидах».

```
// Deserialize(IL)
ILGen.Emit(OpCodes.Ldarg_0);
ILGen.Emit(OpCodes.Ldc_I4_0);
ILGen.Emit(OpCodes.Ldelema, typeof(byte));
ILGen.Emit(OpCodes.Conv_I);
ILGen.Emit(OpCodes.Ldobj, typeof(T));
ILGen.Emit(OpCodes.Ret);
```

#Темная сторона. Практические уроки.

Сериализация структур «на стероидах».

```
// Serialize(IL)
ILGen.Emit(OpCodes.Ldc_I4_S, sizeof(T));
ILGen.Emit(OpCodes.Newarr, typeof(byte));
ILGen.Emit(OpCodes.Ldc_I4_0);
ILGen.Emit(OpCodes.Ldelema, typeof(byte));
ILGen.Emit(OpCodes.Ldarg_0);
ILGen.Emit(OpCodes.Stobj, typeof(T));
```

#Темная сторона. Практические уроки.

Generic вычисления

```
// Generic Sum (Expressions)
var a = Expression.Parameter(typeof(T), "a");
var b = Expression.Parameter(typeof(T), "b");
sum = Expression.Lambda<Func<T, T, T>>(
          Expression.Add(a, b), a, b
      ).Compile();
```

#Темная сторона. Тут не будет легко.

Как не бояться «минного поля»

- Проблемы со сложностью генерации IL-кода.
- Проблемы с отладкой.
- Ограничения и баги.

#Темная сторона. Сложность IL.

Теория Fluent IL подхода

```
// Fluent IL
// https://github.com/FluentIL/FluentIL
var method = IL.NewMethod()
    .Returns(typeof(void))
    .WriteLine("Hello!")
    .Ret();
```

#Темная сторона. Сложность IL.

Теория гибридного подхода

```
// Hybrid IL (POCO)
class ValueViewModel {
    public virtual T Value { get; set; }
    protected void OnValueChanged() { }
}
```

#Темная сторона. Сложность IL.

Теория гибридного подхода

```
// Hybrid (Bodied Expressons)

Expression<Action<string>> e =
  name => Console.WriteLine($"Hello, {name});
```

#Темная сторона. Отладка.

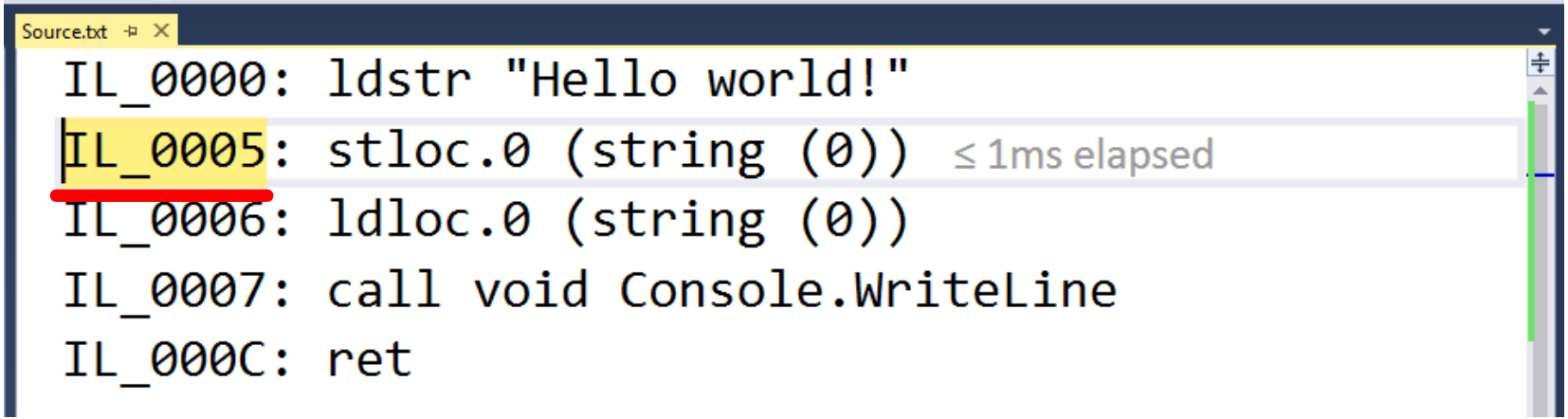
Генерация отладочной информации.

```
// Generate Source-File for Debugging
ISymbolDocumentWriter doc =
    module.DefineDocument(@"Source.txt",
        Guid.Empty, Guid.Empty, Guid.Empty);

// Generate Debug Points
ILGen.MarkSequencePoint(doc, 1, 1, 1, 8);
ILGen.Emit(OpCodes.Ldstr, "Hello world!");
```

#Темная сторона. Отладка.

Генерация отладочной информации.

```
ILGen.MarkSequencePoint(doc, 1, 1, 1, 8);
```



#Темная сторона. Ограничения и баги.

Это надо знать

- Ограничения Dynamic Method.
- Ограничения Expression API.
- Баги Expression API.
- Баги TypeBuilder.

#Темная сторона. Ограничения.

Dynamic Method API Restrictions

- Только статические методы.
- Не Generic.
- Не VarArgs.

#Темная сторона. Ограничения.

Dynamic Method not allowed API

```
// System.Reflection.Emit.DynamicILGenerator
public override void BeginFaultBlock() {
    var id = "InvalidOperation_NotAllowedInDynamicMethod";
    var msg = Environment.GetResourceString(id);
    throw new NotSupportedException(msg);
}
```

#Темная сторона. Ограничения.

Dynamic Method not allowed API

```
// System.Reflection.Emit.DynamicILGenerator
 - BeginExceptFilterBlock
 - BeginFaultBlock
 - UsingNamespace
 - MarkSequencePoint
 - BeginScope
 - EndScope
```

#Темная сторона. Ограничения.

Почему приходится опускаться до работы с байтами

- Emit API генерирует лишнее.
- Emit API не позволяет генерировать что-то.

#Темная сторона. Темнее некуда.

```
DynamicILInfo ilInfo = dm.GetDynamicILInfo();
byte[] code = {
/* ldstr */ 0x72, 0x00, 0x00, 0x00, 0x00,
/* call  */ 0x28, 0x00, 0x00, 0x00, 0x00,
/* ret   */ 0x2a
};
ilInfo.SetCode(code, 3);
```

#Темная сторона. Ограничения.

Expressions.Compile vs Expression.CompileToMethod

```
var lambda = Expression.Lambda<T>(/*...*/)
                       .Compile();
var delegate = (T)((DynamicMethod)dynMeth)
                       .CreateDelegate(typeo(T));
Assert.AreEqual(lambda, delegate);
```

#Темная сторона. Ограничения.

Expressions – ограничения bodied синтаксиса

- Переменные
- Присвоения/мутаторы
- События
- Null-coalescing (x?.Property)
- Блоки выражений

#Темная сторона. Баги Expression API.

Не все то работает, что компилируется

```
// Roslyn

DateTime? d;
Expression<Func<bool>> e = _ => d < DateTimeOffset.Now;
```

#Темная сторона. Баги Expression API.

Тип константы лучше указывать явно

```
// Func<Type> = () => typeof(Foo);

Type fooType = typeof(Foo);
var body = Expression.Constant(fooType);
var lambda = Expression.Lambda<Func<Type>>(body);
```

#Темная сторона. Баги TypeBuilder.

Разные базовые типы могут игнорироваться

```
// TypeBuilder
var tb1 = moduleBuilder.DefineType(
    "Foo1_Dyn", TypeAttributes.Public, typeof(Foo1));
var tb2 = moduleBuilder.DefineType(
    "Foo2_Dyn", TypeAttributes.Public, typeof(Foo2));
Assert.AreNotEqual(tb1.BaseType, tb2.BaseType);
// Runtime
tb1.CreateType().BaseType == tb2.CreateType().BaseType; //!!!
```

#Причины. Только реальные.

Зачем погружаться в пучины кодогенерации

- Производительность.
- Обход ограничений.
- Runtime взаимодействие.
- Возможность строить сложные системы (мета-, микро- и макро скаффолдинг)

DevAV - Employees

HOME VIEW

New Employee | New Group | New Items | Delete | Edit | Print | Meeting | Task | Mail Merge | Thank You Note | Service Excellence | Welcome To DevAV | Employee Award | Probation Notice | List | Card | Map It | Custom Filter | Getting Started | Get Free Support | Buy Now | About

New | Delete | Actions | Quick Letter | View | Find | DevExpress

Employees

Favorites
- All (51)
- Salaried (37)
- Commission (5)
- Contract (3)
- Terminated (4)
- On Leave (2)

Custom Filters
- On probation (4)
- Custom Filter 0

Groups

Search Employees (Ctrl + F)

DEPARTMENT

FULL NAME	ADDRESS	CITY	STATE	ZIP CODE	EMAIL
Engineering (Count=9)					
Brad Farkus	1635 Woods Drive	Los Angeles	CA	90069	bradf@dx-email.com
Bart Arnaz	13109 Old Myford Rd	Irvine	CA	92602	barta@dx-email.com
Leah Simpson	6755 Newlin Ave	Whittier	CA	90601	leahs@dx-email.com
Arnie Schwartz	125 W Elm St	Brea	CA	92821	arnolds@dx-email.com
Billy Zimmer	1325 Prospect Dr	Redlands	CA	92373	williamz@dx-email.com
Samantha Piper	200 E Ave 43	Los Angeles	CA	90031	samanthap@dx-email.com
Maggie Boxter	3101 W Harvard St	Santa Ana	CA	92704	margaretb@dx-email.com
Terry Bradley	4595 Cochran St	Simi Valley	CA	93063	terryb@dx-email.com
Stu Pizaro	200 N. Spring St	Los Angeles	CA	90012	stu@dx-email.com
Human Resources (Count=6)					
Greta Sims	1700 S Grandview Dr.	Alhambra	CA	91803	gretas@dx-email.com
Sandra Johnson	4600 N Virginia Rd.	Long Beach	CA	90807	sandraj@dx-email.com
Cindy Stanwick	2211 Bonita Dr.	Glendale	CA	91208	cindys@dx-email.com
Marcus Orbison	501 N Main St	Los Angeles	CA	90012	marcuso@dx-email.com
Sandy Bright	7570 McGroarty Ter	Tujunga	CA	91042	sandrab@dx-email.com
Ken Samuelson	12100 Mora Dr	Santa Fe Springs	CA	90670	kents@dx-email.com
…ount=8)					
Amelia Harper	527 W 7th St	Los Angeles	CA	90014	ameliah@dx-email.com
Brett Wade	1120 Old Mill Rd.	San Marino	CA	91108	brettw@dx-email.com
Taylor Riley	7776 Torreyson Dr	West Hollywood	CA	90046	taylorr@dx-email.com
Wally Hobbs	10385 Shadow Oak Dr	Chatsworth	CA	91311	wallyh@dx-email.com
Brad Jameson	1100 Pico St	San Fernando	CA	91340	bradleyj@dx-email.com
Karen Goodson	309 Monterey Rd	South Pasadena	CA	91030	kareng@dx-email.com
Morgan Kennedy	11222 Dilling St	San Fernando Valley	CA	91604	morgank@dx-email.com
Violet Bailey	1418 Descanso Dr	La Canada	CA	91011	violetb@dx-email.com
…agement (Count=4)					
John Heart	351 S Hill St.	Los Angeles	CA	90013	jheart@dx-email.com
Samantha Bright	5801 Wilshire Blvd.	Los Angeles	CA	90036	samanthab@dx-email.com
Arthur Miller	3800 Homer St.	Los Angeles	CA	90031	arthurm@dx-email.com
Robert Reagan	4 Westmoreland Pl.	Pasadena	CA	91103	robertr@dx-email.com
…s (Count=10)					

Enter text to search...

EMPLOYEES

- Amelia Harper — Network Admin
- Jackie Garmin — Support Assistant
- Lincoln Bartlett — Sales Assistant
- Brad Farkus — Engineer
- John Heart — CEO
- Samantha Bright — COO

Brad Farkus

Engineer

EVALUATIONS | TASKS

DUE DATE	SUBJECT
1/20/2013	Update address

I moved and I need to fill out an address change form for our insurance. Can I download a PDF online or can you email the form.

5/28/2014	Review Automation App

Can we set a time to go over the new changes? I need to be brought up to speed ASAP

Products

Enter text to search...

PRODUCTS

- ExcelRemote IR — Control all of your electronics with newfo...
- ExcelRemote Bluetooth — Control all of your electronics with ease u...
- ExcelRemote IP — Control all of your electronics with ease u...
- DesktopLED 21 — The 21" Brilliance LED Computer Monitor i...
- DesktopLED 19 — The 19" Brilliance LED Computer Monitor i...
- DesktopLCD 21 — The 21" Brilliance LCD Computer Monitor i...

Customers

Enter text to search...

CUSTOMERS

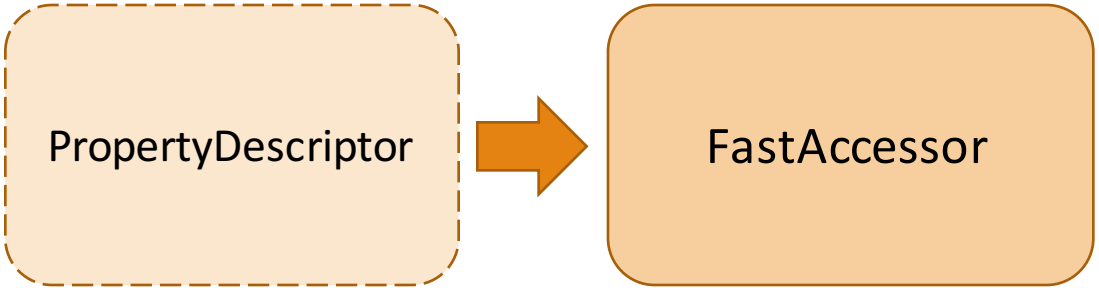
- ACME — (310) 536-0611
- Braeburn — (408) 996-1010
- Circuit Town — (800) 955-2929
- Clicker — (310) 884-9000
- E-Mart — (847) 286-2500
- EZ Stop — (626) 265-8632

Employees | Customers | Products | Sales | Opportunities | •••

RECORDS: 51

#Причины. Производительность

Узкоспециализированный код гораздо эффективнее



#Темная сторона. Fast Reflection.

Используя CIL, можно не платить…

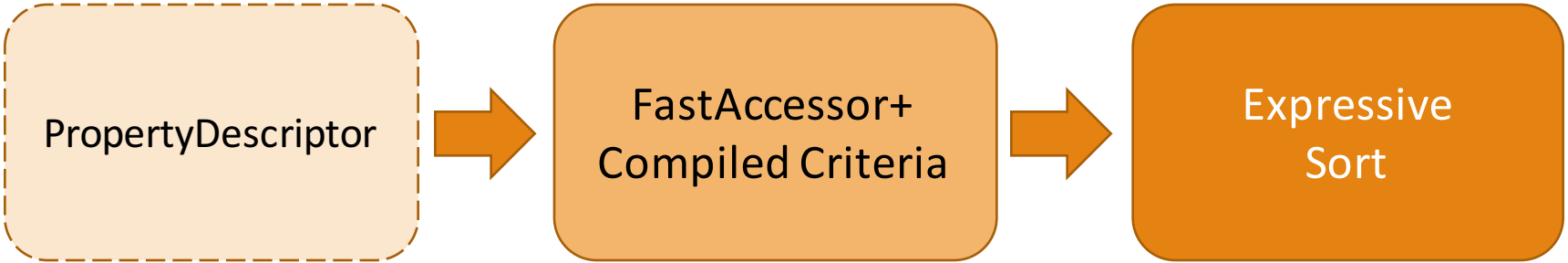
```
// Fast Accessor for Field(C#)
var value = target.@ƒ("m_private");

// Fast Accessor for Field(IL)
ILGen.Emit(OpCodes.Ldarg_0); // target
ILGen.Emit(OpCodes.Ldfld, field);
ILGen.Emit(OpCodes.Ret);
```

#Причины. Производительность

Узкоспециализированный код гораздо эффективнее

#Причины. Производительность

Узкоспециализированный код гораздо эффективнее



#Причины. Только реальные.

Зачем погружаться в пучины кодогенерации

- Производительность.
- Обход ограничений.
- Runtime взаимодействие.
- Возможность строить сложные системы (мета-, микро- и макро скаффолдинг)

#Причины. Обход ограничений

Возможность не писать инфраструктурный код руками

POCO:

```
public class CollectionViewModel {
    public virtual Entity SelectedEntity { get; set; }
    public IEnumerable<Entity> Entities { get; }
    public Task LoadEntitiesAsync() {
        //...
    }
}
```

#Причины. Обход ограничений

Возможность использовать недоступные для платформы возможности

```
var fluent = mvvmContext.OfType<CollectionViewModel>();
fluent.SetBinding(gridControl, g => g.DataSource,
   x => x.Entities);

fluent.WithEvent<RowEventArgs>(gridView,"FocusedRowChanged")
  .SetBinding(x => x.SelectedEntity, arg => arg.Row as Entity);

fluent.WithEvent(this,"Load")
    .EventToComand(x => x.LoadEntitiesAsync());
```

#Причины. Обход ограничений

Возможность использовать утиную типизацию

Is ICommand Needed?

```
class Command : ICommand {
    void Execute() {
        //...
    }
}
```

Absolutely NO!

```
class CustomAction {
    void Execute() {
        //...
    }
}
```

#Причины. Только реальные.

Зачем погружаться в пучины кодогенерации

- Производительность.
- Обход ограничений.
- Runtime взаимодействие.
- Возможность строить сложные системы (мета-, микро- и макро скаффолдинг)

#Причины. Runtime-взаимодействие

«Подогнать» тип под требования принимающей стороны

```
public class DialogService {
    public int ShowDialog(object viewModel) {
        //...
    }
}
```

#Причины. Только реальные.

Зачем погружаться в пучины кодогенерации

- Производительность.
- Обход ограничений.
- Runtime взаимодействие.
- Возможность строить сложные системы (мета-, микро- и макро скаффолдинг)

#Причины : Сложные системы.

Просто предоставьте данные. Все остальное можно сгенерировать.

```
public class FilteringModel {
    public decimal Price { get; set; }
    [FilterLookup("Trademarks", Top = 10)]
    public string Trademark { get; set; }
    [Display(Name = "Model Available"]
    public bool InStock { get; set; }
}
```

Land Rover

Range Rover Evoque

$42,040.00

Chrysler

Town & Country

$42,140.00

Audi

A6

$43,095.00

Toyota

Sequoia

$43,450.00

Trademark

- [ ] All
- [ ] Toyota
- [ ] Ford
- [ ] Nissan
- [ ] Honda
- [ ] Audi
- [ ] BMW
- [ ] Subaru
- [ ] Volkswagen
- [ ] Mitsubishi
- [ ] Mercedes-Benz

More...

- [ ] Model Available

Price

From $40,471.00 To $94,131.00

#Причины : Сложные системы.

Создайте свой DSL.

```
[DomainComponent]
public interface IPerson {
    string FirstName { get; set; }
    string LastName { get; set; }
    string FullName { get; }
}
```

#Причины : Сложные системы.

Создайте свой DSL.

```
[DomainLogic(typeof(IPerson))]
public class PersonLogic {
    public string Get_FullName(IPerson p) {
        return p.LastName + ", " + p.FirstName;
    }
}
```

#Причины. Just For Fun.

Надо попробовать! А вдруг, что-то хорошее получится?

#Вопросы.

Примеры+презентация:

https://github.com/DmitryGaravsky/DotNext.2016.Samples

Статьи:

http://dmitrygaravsky.github.io/

Projects:

https://github.com/DmitryGaravsky/ILReader

https://github.com/DmitryGaravsky/FastAccessors

DevExpress®

Спасибо за внимание!

Гаравский Дмитрий
dmitry.garavsky@devexpress.com

WWW.DEVEXPRESS.COM

support@devexpress.com