Implementacja wydajnego wzorca wstrzykiwania zależności dla złożonych grafów zależności

Adrian Mularczyk

Praca wykonana pod kierunkiem dr. Wiktora Zychli

Uniwersytet Wrocławski
Wydział Matematyki i Informatyki
Kierunek: Informatyka

Agenda

Przedstawienie problemu

SOLID

S	-	SRP	(Single responsibility principle)
O	-	OCP	(Open/closed principle)
L	-	LSP	(Liskov substitution principle)
I	-	ISP	(Interface segregation principle)
D	-	DIP	(Dependency inversion principle)

SOLID

S	-	SRP	(Single responsibility principle)
O	-	OCP	(Open/closed principle)
L	-	LSP	(Liskov substitution principle)
I	-	ISP	(Interface segregation principle)
D	-	DIP	(Dependency inversion principle)

Dependency Inversion Principal

```
public class Foo
{
    public void DoSomeWork()
    {
        //do some work
    }
}

public class Bar
{
    private readonly Foo _foo;

    public Bar()
    {
        _foo = new Foo();

    }

    public void DoSomething()
    {
        _foo.DoSomeWork();
    }
}
```

```
public interface IFoo
{
    void DoSomeWork();
}

public class Foo : IFoo
{
    public void DoSomeWork()
    {
        //do some work
    }
}

public class Bar
{
    private readonly IFoo _foo;

    public Bar(IFoo foo)
    {
        _foo = foo;
    }

    public void DoSomething()
    {
        _foo.DoSomeWork();
    }
}
```

Kontenery wstrzykiwania zależności

- Rejestracja
- Tworzenie obiektów

```
public class A
{
    public A(B b) { }
}

public class B
{
    public B() { }
}
```

```
container.Register<A>();
container.Register<B>();

container.Resolve<B>();
container.Resolve<A>();
```

Obserwacje

- W grafach zależności często powtarzają się typy
- Utworzenie instancji nowego obiektu zajmuje czas
- Nowe obiekty są często tworzone

Cel Pracy

Wydajny kontener wstrzykiwania zależności

Wstrzykiwanie zależności

Wstrzykiwanie zależności

- Wstrzykiwanie przez konstruktor
- Wstrzykiwanie przez metodę
- Wstrzykiwanie przez właściwość

```
public class SampleClass
{
    public SampleClass()
    {
    }

    [DependencyConstructor]
    public SampleClass(EmptyClass emptyClass)
    {
        EmptyClass = emptyClass;
    }

    public EmptyClass EmptyClass { get; }
}
```

```
public class SampleClass
{
    [DependencyMethod]
    public void SetEmptyClass(EmptyClass emptyClass)
    {
        EmptyClass = emptyClass;
    }

    public EmptyClass EmptyClass { get; private set; }
}
```

```
public class SampleClass
{
    [DependencyProperty]
    public EmptyClass EmptyClass { get; set; }
}
```

Implementacje przemysłowe

- Unity
- NInject
- Autofac
- StructureMap
- Windsor
- Grace
- Dryloc
- LightInject
- SimpleInjector

Implementacja

Implementacja

- CIL
- Reflection.Emit

```
IL_0000:  ldarg.1
IL_0001:  ldc.i4.0
IL_0002:  blt.s IL_0006
IL_0004:  ldarg.1
IL_0005:  ret
IL_0006:  ldarg.1
IL_007:  neg
IL_008:  ret
```

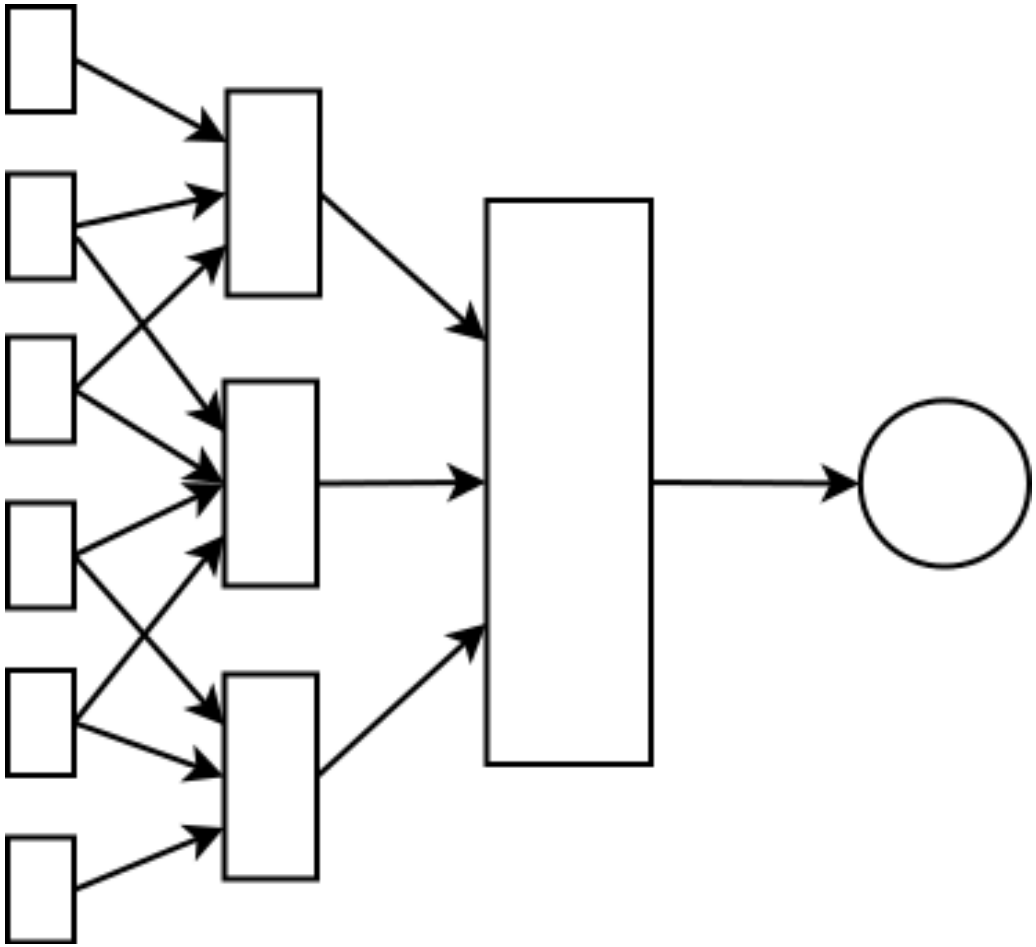
```
public int Abs(int x)
{
    if (x >= 0)
        return x;
    else
        return -x;
}
```

```
var lower = ilgen.DefineLabel();
ilgen.Emit(OpCodes.Ldarg_0);
ilgen.Emit(OpCodes.Ldc_I4_0);
ilgen.Emit(OpCodes.Blt_S, lower);
ilgen.Emit(OpCodes.Ldarg_0);
ilgen.Emit(OpCodes.Ret);
ilgen.MarkLabel(lower);
ilgen.Emit(OpCodes.Ldarg_0);
ilgen.Emit(OpCodes.Neg);
ilgen.Emit(OpCodes.Ret);
```

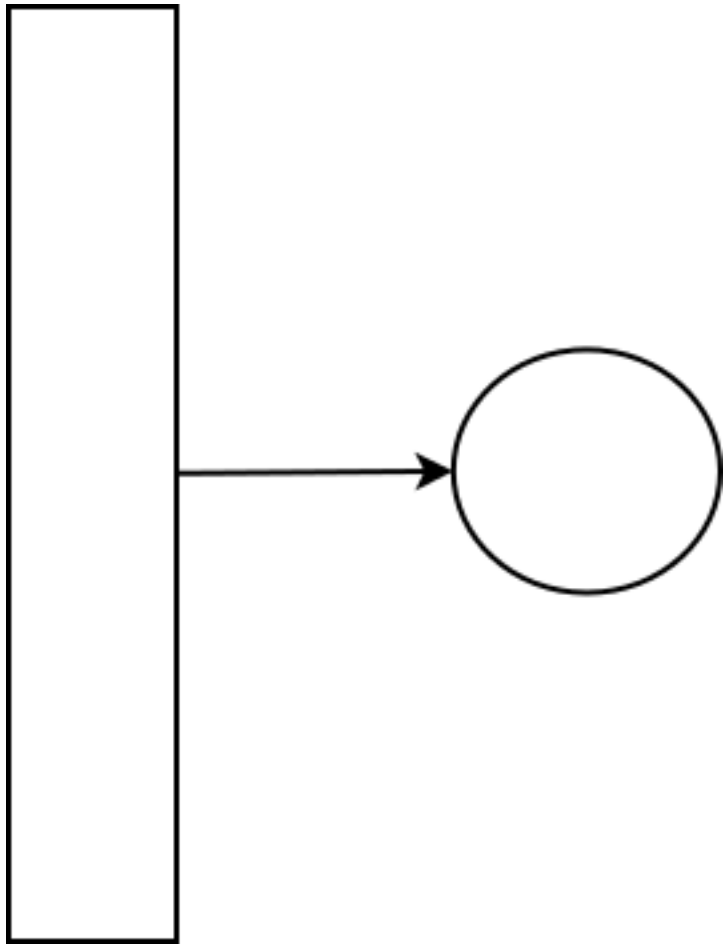
Dwa rozwiązania

- Partial Emit Function
- Full Emit Function

Partial Emit Function



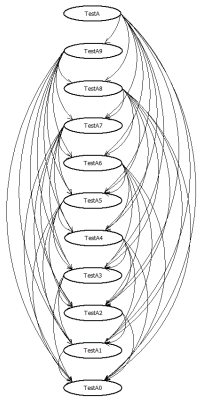
Full Emit Function



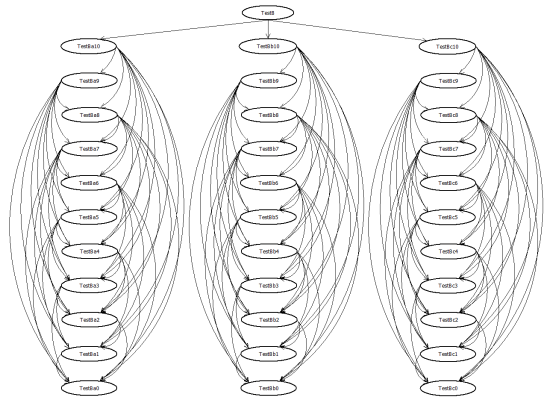
Wyniki

Przypadki testowe

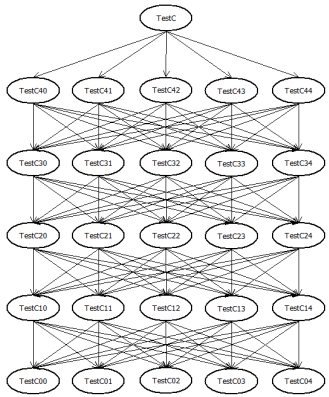
A



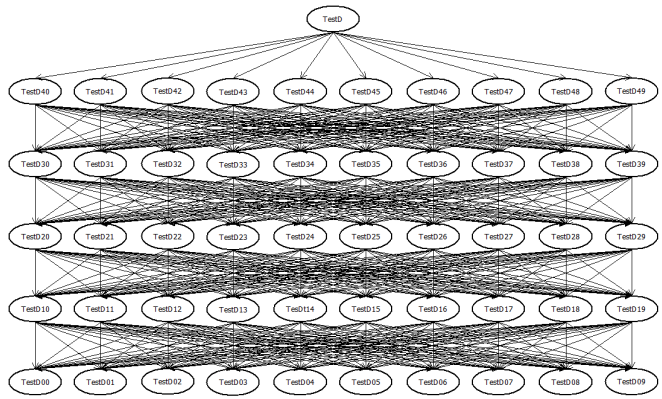
B



C



D



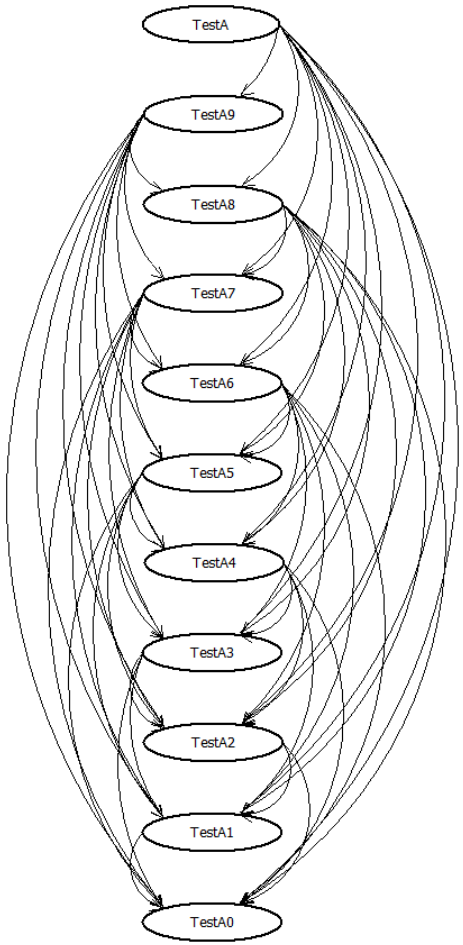
Typy rejestracji

- Register as Singleton,
- Register as Transient,
- Register as TransientSingleton,
- Register as PerThread (PerScope),
- Register as FactoryMethod.

Typy rejestracji

- Register as Singleton,
- Register as Transient,
- Register as TransientSingleton,
- Register as PerThread (PerScope),
- Register as FactoryMethod.

Przypadek testowy A - graf zależności



Liczba obiektów w 1 iteracji: 1 024

Przypadek testowy A - Transient

Liczba iteracji: 1 i 10

Autofac	0
NiquloCPartial	**1**
Windsor	1
NiquloCFull	**8**
Unity	8
LightInject	10
StructureMap	10
Ninject	11
SimpleInjector	13
Dryloc	14
Grace	15

NiquloCPartial	**3**
Autofac	6
NiquloCFull	**8**
LightInject	10
SimpleInjector	14
StructureMap	14
Dryloc	15
Grace	16
Unity	16
Windsor	16
Ninject	90

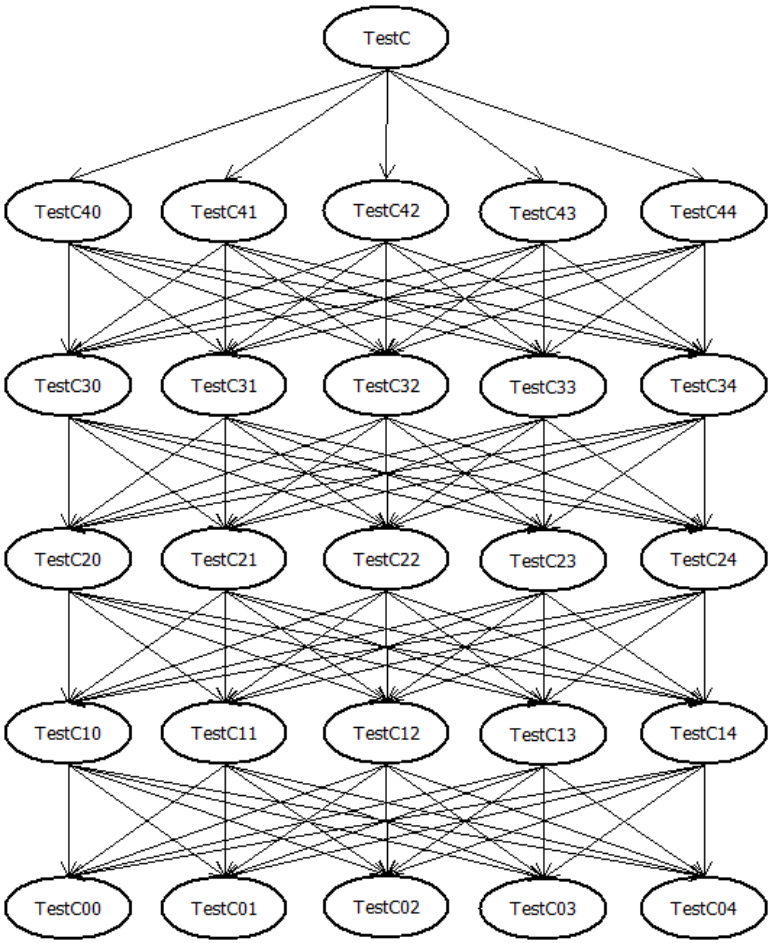
Przypadek testowy A - Transient

Liczba iteracji: 100 i 1000

NiquloCFull	**9**
LightInject	11
SimpleInjector	15
Dryloc	16
Grace	18
NiquloCPartial	**19**
StructureMap	54
Autofac	59
Unity	88
Windsor	155
Ninject	882

NiquloCFull	**18**
LightInject	19
SimpleInjector	29
Dryloc	29
Grace	37
NiquloCPartial	**173**
StructureMap	417
Autofac	587
Unity	813
Windsor	1529
Ninject	8934

Przypadek testowy C - graf zależności



Liczba obiektów w 1 iteracji: 3 906

Przypadek testowy C - Transient

Liczba iteracji: 1 i 10

Autofac	2
NiquloCPartial	**4**
Windsor	7
Unity	19
StructureMap	26
NiquloCFull	**31**
LightInject	36
Dryloc	39
Ninject	39
SimpleInjector	41
Grace	61

NiquloCPartial	**10**
Autofac	24
NiquloCFull	**32**
LightInject	37
Dryloc	40
StructureMap	40
SimpleInjector	41
Unity	47
Grace	62
Windsor	62
Ninject	353

Przypadek testowy C - Transient

Liczba iteracji: 100 i 1000

NiquloCFull	**38**
LightInject	44
Dryloc	47
SimpleInjector	49
Grace	73
NiquloCPartial	**74**
StructureMap	181
Autofac	231
Unity	318
Windsor	608
Ninject	3511

NiquloCFull	**82**
LightInject	86
Dryloc	99
SimpleInjector	116
Grace	155
NiquloCPartial	**690**
StructureMap	1540
Autofac	2288
Unity	3015
Windsor	6037
Ninject	37642

Podsumowanie


2
1
3