

# Dependency injection

Adrian Mularczyk

PGS Software

# Agenda

- 1 Description of the problem
- 2 Dependency injection
- 3 Code
- 4 Summary

# Description of the problem

# 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();
    }
}
```

# Dependency injection container

- Register
- Resolve

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

```
public interface IB
{
}
```

```
public class B : IB
{
}
```

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

```
container.Register<IB, B>();
```

```
container.Resolve<IB>();
```

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

# Dependency injection



# Dependency injection

- Injecting by the constructor
- Injecting by the method
- Injecting by the property

```
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; }
}
```

# Registration types

- Register as Singleton,
- Register as Transient,
- Register as Scope (Thread, HttpRequest),
- Register as FactoryMethod.

# Sample implementations

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

<http://www.palmmedia.de/blog/2011/8/30/ioc-container-benchmark-performance-comparison>

# Code

# Adding DI to our project

- .NET Framework
- .NET Core

# Sample implementation

## Demo

# Extensions

- List instead Dictionary
- Many input types
- Many destination types

# Summary



## Questions?

# Thank you!

e-mail: [amularczyk@pgs-soft.com](mailto:amularczyk@pgs-soft.com)