

# Container Dependency Injection

Adrian Mularczyk

# Agenda

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

# Description of the problem

# Dependency injection

# Dependency injection

```
public class Foo
{
    public void DoSomeWork()
    {
        Bar bar = new Bar();
        bar.DoSomething();
    }
}
```

# Dependency injection

```
public class Foo
{
    public void DoSomeWork()
    {
        Bar bar = new Bar();
        bar.DoSomething();
    }
}
```

# Dependency injection

```
public class Foo
{
    public void DoSomeWork(Bar bar)
    {
        bar.DoSomething();
    }
}
```

# Dependency injection

```
public class Foo
{
    Bar _bar;
    public Foo(Bar bar)
    {
        _bar = bar;
    }
    public void DoSomeWork()
    {
        _bar.DoSomething();
    }
}
```



# Dependency injection kinds

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

# Dependency injection kinds

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

# Container of dependency injection

# Basic operations

- Register,
- Resolve.

# Registration types

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

# Container

```
public class Container
{
    private List<Type> RegisteredTypes;

    public Container()
    {
        RegisteredTypes = new List<Type>();
    }

    ...
}
```

# Register

```
public void Register(Type type)
{
    RegisteredTypes.Add(type);
}
```

# Resolve

```
public object Resolve(Type type)
{
    if(!RegisteredTypes.Contains(type))
        throw new TypeNotRegisteredException();

    return CreateObjectOfType(type);
}
```



# RegisteredTypes

```
public class Container
{
    private List<RegisteredType>
RegisteredTypes;

    public Container()
    {
        RegisteredTypes = new
List<RegisteredType> ();
    }

    ...
}
```

# RegisteredType

```
public class RegisteredType
{
    public Type RegisteredType;
    public RegistrationKind RegistrationKind;
    public object Value;
    public List<Scope, object> Scopes;
    ...
}
```

# Summary

## Questions?

Thank you!