Projektowanie obiektowe oprogramowania
Zestaw B

Inversion of Control (3)

2024-05-21

Liczba punktów do zdobycia: **6/74**
Zestaw ważny do: 2024-06-04

Uwaga! Kontynuacja/zakończenie pracy nad silnikiem Inversion of Control na identycznych zasadach. W szczególności obowiązkową częścią każdego zadania są testy jednostkowe, nawet jeśli nie wspomina się o tym w treści zadań.

1. (**2p**) (**Wstrzykiwanie właściwości**)

   Silnik IoC rozbudować o możliwość wstrzykiwania właściwości/metod (do wyboru).

   Ściślej - kontener w trakcie rozwikływania zależności obiektu dodatkowo analizuje właściwosci i metody i te, które są opatrzone atrybutami (anotacjami) `[DependencyProperty]` / `[DependencyMethod]` i spełniają warunki wstrzykiwania (na przykład: metoda ma określoną sygnaturę, typu `setXYZ( X x, Y y, ... )`) próbuje rozwikływać i wstrzykiwać tak samo, jak robi to z parametrami konstruktora.

   Warunki wstrzykiwania są następujące.

   Dla właściwości (C# i inne języki z właściwościami, ale nie Java), "wstrzykiwalne" są właściwości które mają publiczny akcesor `set`.

   Dla metod (wszystkie języki), "wstrzykiwalne" są metody które nie zwracają żadnego wyniku `void` i mają niepustą listę argumentów.

   Kolejność wstrzykiwania właściwości/metod nie ma znaczenia.

```
public class A {

  ...

  // wstrzykiwanie przez konstruktor (to już mamy)
  public A( B b ) { }

  // wstrzykiwanie przez właściwość
  [DependencyProperty]
  public C TheC { get; set; }

  // wstrzykiwanie przez metodę
  [DependencyMethod]
  public void setD( D d )
  {
     this.d = d;
  }
}

public class B {}
public class C {}
```

```
public class D {}

SimpleContainer c = new SimpleContainer();
A a = c.Resolve<A>();

// tworzy nową instancję A z B wstrzykniętym przez konstruktor,
// C wstrzykniętym przez właściwość i D wstrzykniętym przez metodę.
```

2. (**2p**) (**Uzupełnianie zależności istniejącego obiektu**)

   Silnik IoC rozbudować o możliwość uzupełaniania zależności istniejących obiektów, zbudowanych poza kontenerem.

```
public class SimpleContainer
{
    ...
    public void BuildUp<T>( T Instance );
}

SimpleContainer c = new SimpleContainer();

A theA = ....; // obiekt theA SKĄDŚ pochodzi, ale nie z kontenera
c.BuildUp( theA );

// wstrzykuje do theA zależności przez właściwości, tu: TheC
```

3. (**2p**) (**Przemysłowy framework IoC**)

   W ramach wybranej przez siebie platformy technologicznej/języka, zademonstrować przykłady użycia wybranego przez siebie frameworka IoC (innego niż Unity dla .NET, prezentowanego na wykładzie).

   Przykłady (do wyboru):

   - .NET
     - kontener wbudowany w .NET5 (i wyższe)
     - Autofac
     - Ninject
   - Java
     - Spring DI
     - Google Guice
     - Dagger
   - node.js
     - InversifyJS
     - Noder.io
     - Awilix

   W ramach wybranego frameworka zademonstrować następujące elementy (lub powiedzieć że nie są dostępne):

   - rejestracja typu
   - rejestracja typu na interfejs
   - rejestracja typu na klasę abstrakcyjną
   - rejestracja funkcji fabrykującej

- rejestracja instancji
- uzyskanie instancji wskazanego typu
- uzyskanie instancji implementującej wskazany interfejs
- uzyskanie instancji implementującej klasę abstrakcyjną
- zarządzanie czasem życia - nowa instancja za każdym uzyskaniem lub singleton
- konfiguracja imperatywna (w kodzie)
- konfiguracja deklaratywna (w pliku XML/json)

Wiktor Zychla