Uniwersytet Wrocławski
Wydział Matematyki i Informatyki
Kierunek: Informatyka

Adrian Mularczyk

Stworzenie wydajnego wzorca wstrzykiwania zależności dla złożonych grafów zależności

Praca wykonana pod kierunkiem dr. Wiktora Zychli

Wrocław, 2016

Spis treści

1 Wstęp

1.1 Cel pracy

Wstrzykiwanie zależnośc jest wzorcem projektowym, który pozwala na tworzenie kodu o luźniejszych powiązaniach, łatwiejszego w testowaniu i modyfikacji. Najbardziej popularnymi implementacjami tego wzorca w języku C# są Autofac, StructureMap, Unity i Windsor, a najbardziej wydajnymi DryIoc, LightInject i SimplyInjector. Celem niniejszej pracy magisterskiej jest stworzenie wydajnej implementacji tego wzorca dla złożonych grafów zależności. Do tego celu zostanie wykorzystana funkcjolanosci z przestrzeni nazw Reflection.Emit. W tej pracy zostaną przedstawione dwa rozwiązania.

2 Wstrzykiwanie zależnosci

Jest to zbiór zasad projektowania oprogramowania i wzorców, które pozwalają nam rozwijać luźno powiązany kod [1] (str. 4).
Jakiemu celowi ma służyć wstrzykiwanie zależności? Wstrzykiwanie zależnści nie jestem celem samym w sobie, raczej jest to środek do celu. Ostatecznie celem większości technik programowania jest dostarczenie jak najwydajniej działającego oprogramowania. Jednym z aspektów tego jest napisanie utrzymywalnego kodu.
O ile nie pisze się prototypu lub aplikacji, które nigdy nie mają kolejnych wersji (kończą się na wersji 1), to wkrótce będzie trzeba zająć się utrzymaniem i rozwijaniem istniejącego kodu. Aby być w stanie pracować wydajnie z takim kodem bazowym, musi on być jak najlepiej utrzymywalny.
Wstrzykiwanie zależności jest niczym więcej niż techniką, która umożliwia luźne powiązania, a luźne powiązania sprawiają, że kod jest rozszerzalny i łatwy w utrzymaniu. [1] (str. 5)

3 Implementacja

<Wyjaśnić jak działa Reflection.Emit i do czego zostanie użyte>

3.1 Rozwiązanie 1

<opis PartialEmitFunction>

3.2 Rozwiązanie 2

<opis FullEmitFunction>

4 Testy wydajnościowe

<wyniki testów i ich opis>

5 Podsumowanie

<parę słów na koniec>

Literatura

[1] Dependency Injection in .NET, Mark Seemann