Assembly & Refleksje I.

C#

Wydział Matematyki i Informatyki Stosowanej
Ćwiczenia: Michał Mnich

Refleksje.

Refleksja służy do uzyskania informacji o typie w trakcie wykonywania programu. Klasy, które mają dostęp do metadanych działającego programu są zdefiniowane w przestrzeni nazw System.Reflection.

Przestrzeń nazw System.Reflection zawiera klasy, które pozwalają na uzyskanie informacji o aplikacji oraz pozwalają na dynamiczne dodawanie typów, wartości i obiektów do aplikacji.

Wykorzystanie refleksji pozwala na:

- podgląd atrybutów w trakcie wykonywania programu;
- sprawdzenie różnych typów danych w danej bibliotece oraz utworzenie ich instancji;
- wykonanie późnego wiązania do metod i właściwości (późne wiązanie oznacza, że np. docelowa metoda jest poszukiwana w trakcie wykonywania programu. Wiązanie takie ma zwykle wpływ na wydajność. Poszukiwanie takie wymaga dopasowania w trakcie wykonywania programu, oznacza to, że wywołania metod są wolniejsze. Przeciwieństwem jest wczesne wiązanie, tj. docelowa metoda jest znana już w trakcie kompilacji kodu);
- tworzenie nowych typów w trakcie wykonywania programu a następnie wykonywanie różnych zadań przy użyciu tych typów.

Przykładowe użycie:

Przykład 1.

```
using System;
namespace Refleksja
{
   class Program
   {
      static void Main(string[] args)
      {
         System.Reflection.MemberInfo info = typeof(MyClassToGetAttributeInfo);
         // pobranie listy atrybutów
         object[] attributes = info.GetCustomAttributes(true);
         for (int i = 0; i < attributes.Length; i++)
         {
            // Wypisujemy wszystkie atrybuty
            Console.WriteLine(attributes[i]);
            // Dodatkowo uzyskamy dostęp do opisu naszego atrybutu
            ExampleAttribute ea = (ExampleAttribute)attributes[i];
            Console.WriteLine("Info: {0}", ea.message);
         }
         Console.ReadKey();
         // Wynik działania programu
         // Refleksja.ExampleAttribute
         // Info: Informacja o mojej klasie
      }
   }
   [AttributeUsage(AttributeTargets.All)]
   public class ExampleAttribute : Attribute
   {
      public readonly string message;
      private string topic;
      public ExampleAttribute(string Message)
      {
         this.message = Message;
      }
      public string Topic
      {
         get
         {
            return topic;
         }
         set
         {
            topic = value;
         }
      }
   }
   [ExampleAttribute("Informacja o mojej klasie")]
   class MyClassToGetAttributeInfo
   {
   }
}
```

Przykład 2

```
//bez refleksji
Foo foo = new Foo();
foo.hello();

//z użyciem refleksji
var type = Type.GetType("namespace.Foo");   //string powinien zawierać namespace naszej klasy
var foo = Activator.CreateInstance(type); //inicjalizacja obiektu określonego typu
MethodInfo inf = type.GetMethod("hello");
inf.Invoke(foo); // jako drugi parametr metoda Invoke przyjmuje tablicę Object[] są to parametry metody hello.
```

Więcej informacji znajdą państwo w wykładzie.

Zadanie 1:

Proszę przy użyciu refleksji przeprowadzić pełna analizę poniższej klasy.

```
public class Customer
{
    private string _name;
    protected int _age;
    public bool isPreferred;


    public Customer(string name)
    {
        if (string.IsNullOrEmpty(name)) throw new ArgumentNullException("Customer name!");
        _name = name;
    }

    public string Name
    {
        get
        {
            return _name;
        }
    }
    public string Address { get; set; }
    public int SomeValue { get; set; }

    public int ImportantCalculation()
    {
        return 1000;
    }

    public void ImportantVoidMethod()
    {
    }

        public enum SomeEnumeration
    {
        ValueOne = 1
        , ValueTwo = 2
    }

    public class SomeNestedClass
    {
        private string _someString;
    }
}
```

poprzez analizę rozumiem ćwiczenie, w którym należy wypisać za pomocą refleksji następujące elementy powyższej klasy do każdej prosze przy nazwie danego pola wypisać także jakiego jest ona typu

```
Console.WriteLine("Fields: ");
//lista pól w klasie Pogrupowane względem dostępu
Console.WriteLine("-- Public: ");
//publiczne
Console.WriteLine("-- Non Public: ");
//niepubliczne
//Przykład:
//Type: “string”; name: “_name”

Console.WriteLine("Methods: ");
//Lista metod

Console.WriteLine("Nested types: ");
//typy zagnieżdżone
```

```
Console.WriteLine("Properties: ");
//propercje

Console.WriteLine("Members: ");
//Członkowie
```

Zadanie 2:

Za pomocą metody **GetProperty** prosze ustawić wszytkie propercje w klasie **Customer** a następnie prosze je wyswietlić.