МГТУ имени Н. Э. Баумана

Базовые компоненты интернет технологий

Отчет по лабораторной работе №6

Кочетков Михаил Дмитриевич
Группа ИУ5-31б

10 октября 2018 г.

Москва
2018

Задание

Часть 1. Разработать программу, использующую делегаты.

1. Программа должна быть разработана в виде консольного приложения на языке C#
2. Определите делегат, принимающий несколько параметров различных типов и возвращающий значение произвольного типа
3. Напишите метод, соответствующий данному делегату
4. Напишите метод, принимающий разработанный Вами делегат, в качестве одного из входных параметров. Осуществите вызов метода, передавая в качестве параметра-делегата:
   - метод, разработанный в пункте 3
   - лямбда-выражение
5. Повторите пункт 4, используя вмето разработанного Вами делегата, обобщенный делегат Func< > или Action< >, соответствующий сигнатуре разработанного Вами делегата

Часть 2. Разработать программу, реализующую работу с рефлексией.

1. Программа должна быть разработана в виде консольного приложения на языке C#
2. Создайте класс, содержащий конструкторы, свойства, методы
3. С использованием рефлекции выведите информацию о конструкторах, свойствах, методах
4. Создайте класс атрибута (унаследован от класса System.Attribute)
5. Назначьте атрибут некоторым свойствам класса. Выведите только те свойства, которым назначен атрибут
6. Вызовите один из методов класса с использованием рефлексии

Код

Program.cs

```
using System;
using System.Reflection;

namespace Lab_6
{
    class Program
    {
        public delegate string Stringing(int x, double y);

        static void Main(string[] args)
        {
            // Delegate Testing
            Apply(MakeStringSum, 10, 15.2);
            Apply((x, y) => $"{x} - {y} = {(x - y)}", 10, 15.2);

            // Func<> Testing
            ApplyFunc(MakeStringSum, 10, 15.2);
            ApplyFunc((x, y) => $"{x} - {y} = {(x - y)}", 10, 15.2);


            Console.WriteLine();
            // Reflection Testing

            var aInfo = typeof(A);
            var propertyInfos = aInfo.GetProperties();
            var methodInfos = aInfo.GetMethods();
            var constructorInfos = aInfo.GetConstructors();

            Console.WriteLine("Properties:");
            foreach (var info in propertyInfos)
            {
                Console.WriteLine(info);
            }
            Console.WriteLine();

            Console.WriteLine("Methods:");
            foreach (var info in methodInfos)
            {
                Console.WriteLine(info);
            }
            Console.WriteLine();

            Console.WriteLine("Constructors:");
            foreach (var info in constructorInfos)
            {
                Console.WriteLine(info);
            }
            Console.WriteLine();
```

```
49
50              Console.WriteLine("Attributed Properties:");
51              foreach (var info in propertyInfos)
52              {
53                  if (Attribute.IsDefined(info, typeof(MyAttribute)))
54                  {
55                      Console.WriteLine(info);
56                  }
57              }
58              Console.WriteLine();
59
60              Console.WriteLine("Call MethodString:");
61              Console.WriteLine(aInfo.GetMethod("MethodString").Invoke(new A(10, 10.10,
                ↪  "10"), null));
62          }
63
64          public static void ApplyFunc(Func<int, double, string> f, int param1, double
            ↪  param2)
65          {
66              Console.WriteLine(f(param1, param2));
67          }
68
69          public static void Apply(Stringing f, int param1, double param2)
70          {
71              Console.WriteLine(f(param1, param2));
72          }
73
74          public static string MakeStringSum(int x, double y)
75          {
76              return $"{x} + {y} = {(x + y)}";
77          }
78      }
79  }
```

A.cs

```cs
namespace Lab_6
{
    public class A
    {
        [My("Attributed")]
        public int PropertyInt { get; set; }
        public double PropertyDouble { get; set; }
        [My("Attributed")]
        public string PropertyString { get; set; }

        public A(int propertyInt, double propertyDouble, string propertyString)
        {
            PropertyInt = propertyInt;
            PropertyDouble = propertyDouble;
            PropertyString = propertyString;
        }

        public int MethodInt()
        {
            return 10;
        }

        public double MethodDouble()
        {
            return 10.10;
        }

        public string MethodString()
        {
            return "10";
        }
    }
}
```

MyAttribute.cs

```
namespace Lab_6
{
    public class MyAttribute : System.Attribute
    {
        public string Role { get; set; }

        public MyAttribute(string role)
        {
            Role = role;
        }
    }
}
```

Тесты

```
10 + 15.2 = 25.2
10 - 15.2 = -5.2
10 + 15.2 = 25.2
10 - 15.2 = -5.2
```

Рис. 1: Проверка делегатов

```
Properties:
Int32 PropertyInt
Double PropertyDouble
System.String PropertyString
```

Рис. 2: Рефлексия. Свойства

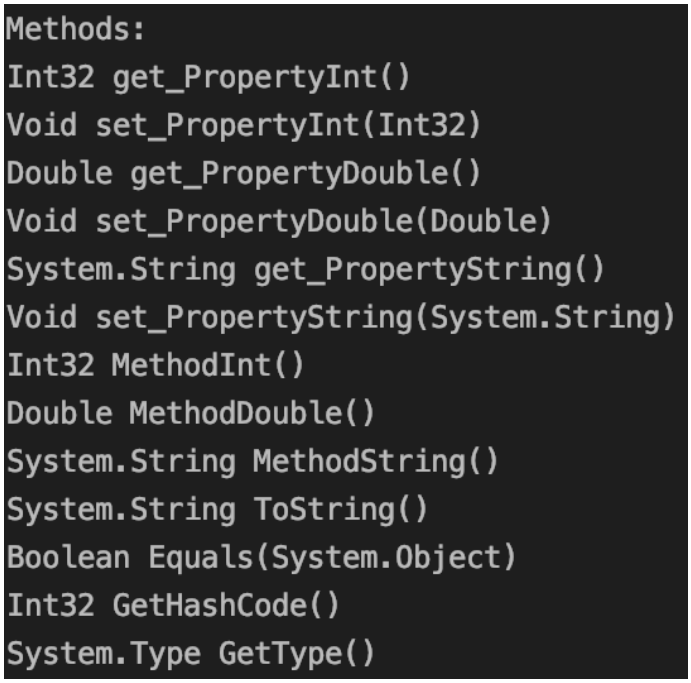

Рис. 3: Рефлексия. Методы

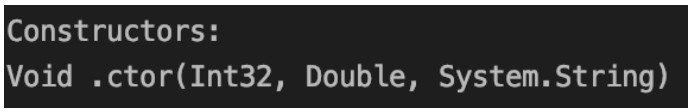

Рис. 4: Рефлексия. Конструкторы

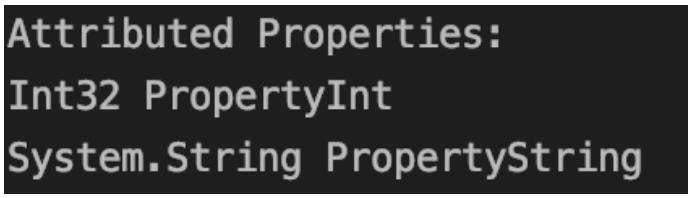

Рис. 5: Рефлексия. Свойства с атрибутом

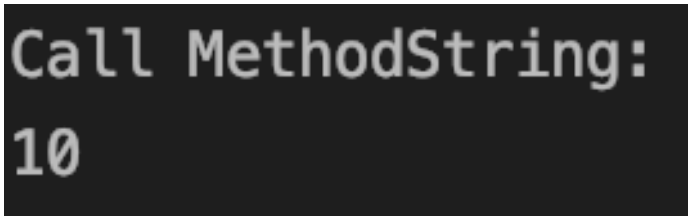

Рис. 6: Рефлексия. Вызов метода