

МГТУ имени Н. Э. Баумана

Базовые компоненты интернет технологий

Отчет по лабораторной работе №6

Мелисов Арсен Ермекович

Группа ИУ5-31б

24 декабря 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 вместо 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 рефлексии 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

```
1 using System;
2 using System.Reflection;
3
4 namespace Lab_6
5 {
6     class Program
7     {
8         public delegate string Stringing(int x, double y);
9
10        static void Main(string[] args)
11        {
12            // Delegate Testing
13            Apply(MakeStringSum, 10, 15.2);
14            Apply((x, y) => $"{x} - {y} = {(x - y)}", 10, 15.2);
15
16            // Func<> Testing
17            ApplyFunc(MakeStringSum, 10, 15.2);
18            ApplyFunc((x, y) => $"{x} - {y} = {(x - y)}", 10, 15.2);
19
20
21            Console.WriteLine();
22            // Reflection Testing
23
24            var aInfo = typeof(A);
25            var propertyInfos = aInfo.GetProperties();
26            var methodInfos = aInfo.GetMethods();
27            var constructorInfos = aInfo.GetConstructors();
28
29            Console.WriteLine("Properties:");
30            foreach (var info in propertyInfos)
31            {
32                Console.WriteLine(info);
33            }
34            Console.WriteLine();
35
36            Console.WriteLine("Methods:");
37            foreach (var info in methodInfos)
38            {
39                Console.WriteLine(info);
40            }
41            Console.WriteLine();
42
43            Console.WriteLine("Constructors:");
44            foreach (var info in constructorInfos)
45            {
46                Console.WriteLine(info);
47            }
48            Console.WriteLine();
```

```

49
50 Console.WriteLine("Attributed Properties:"); foreach
51 (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

```
1 namespace Lab_6
2 {
3     public class A
4     {
5         [My("Attributed")]
6         public int PropertyInt { get; set; }
7         public double PropertyDouble { get; set; }
8         [My("Attributed")]
9         public string PropertyString { get; set; }
10
11         public A(int propertyInt, double propertyDouble, string propertyString)
12         {
13             PropertyInt = propertyInt;
14             PropertyDouble = propertyDouble;
15             PropertyString = propertyString;
16         }
17
18         public int MethodInt()
19         {
20             return 10;
21         }
22
23         public double MethodDouble()
24         {
25             return 10.10;
26         }
27
28         public string MethodString()
29         {
30             return "10";
31         }
32     }
33 }
```

MyAttribute.cs

```
1 namespace Lab_6
2 {
3     public class MyAttribute : System.Attribute
4     {
5         public string Role { get; set; }
6
7         public MyAttribute(string role)
8         {
9             Role = role;
10        }
11    }
12 }
```

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

```
Methods:
Int32 get_PropertyInt()
Void set_PropertyInt(Int32)
Double get_PropertyDouble()
Void set_PropertyDouble(Double)
System.String get_PropertyString()
Void set_PropertyString(System.String)
Int32 MethodInt()
Double MethodDouble()
System.String MethodString()
System.String ToString()
Boolean Equals(System.Object)
Int32 GetHashCode()
System.Type GetType()
```

Рис. 3: Рефлексия. Методы

```
Constructors:
Void .ctor(Int32, Double, System.String)
```

Рис. 4: Рефлексия. Конструкторы

```
Attributed Properties:
Int32 PropertyInt
System.String PropertyString
```

Рис. 5: Рефлексия. Свойства с атрибутом

```
Call MethodString:
10
```

Рис. 6: Рефлексия. Вызов метода