Лабораторная работа № 6.2 по курсу
“Базовые компоненты интернет-технологий”

Алексеев А.В.
РТ5-31
МГТУ им. Баумана

Описание задания лабораторной работы.

Разработать программу, использующую делегаты.

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 классам. Выведите только те свойства, которым назначен атрибут.
6. Вызовите один из методов класса с использованием рефлексии.

Код программы:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Reflection;
namespace Лаб6._2
{
    class Program
    {
        class Exmpl
        {
            float width;
            float length;

            public Exmpl(float width,float length)
            {
                this.width = width;
                this.length = length;
            }
            [NewAttribute("Описание для длины")]
            public float Length { get; set; }
            [NewAttribute("Описание для ширины")]
            public float Width { get; set; }
            public float Area() { return width * length; }
            public float Perimeter() { return (width + length) * 2; }

        }
        static void AssemblyInfo()
        {
            Console.WriteLine("Текущая сборка");
            Assembly i = Assembly.GetExecutingAssembly();
            Console.WriteLine("Текущее имя сборки" + i.FullName);
            Console.WriteLine("Исполняемый файл" + i.Location);
        }
        [AttributeUsage(AttributeTargets.Property, AllowMultiple = false, Inherited =
false)]
        public class NewAttribute : Attribute
        {
            public NewAttribute() { }
            public NewAttribute(string DP)
            {
                Description = DP;
            }

            public string Description { get; set; }
        }

        static void TypeInfo()
        {
            Type t = typeof(Exmpl);
            //Type t = obj.GetType();

            Console.WriteLine("Вывод информации о типе ");
            Console.WriteLine("Тип" + t.FullName + "Унаследован от " +
t.BaseType.FullName);
            Console.WriteLine("Пространство имён" + t.Namespace);


            Console.WriteLine("\nКонструкторы:");
            foreach (var x in t.GetConstructors())
            {
                Console.WriteLine(x);
            }
```

```
            Console.WriteLine("\nМетоды:");
            foreach (var x in t.GetMethods())
            {
                Console.WriteLine(x);
            }

            Console.WriteLine("\nСвойства:");
            foreach (var x in t.GetProperties())
            {
                Console.WriteLine(x);


            }
        }
        public static bool GetPropertyAttribute(PropertyInfo checkType, Type
attributeType, out object attribute)
        {
            bool Result = false;
            attribute = null;

            //Поиск атрибутов с заданным типом
            var isAttribute = checkType.GetCustomAttributes(attributeType, false);
            if (isAttribute.Length > 0)
            {
                Result = true;
                attribute = isAttribute[0];
            }

            return Result;
        }
        static void AttributeInfo()
        {
            Type t = typeof(Exmpl);
            Console.WriteLine("\nСвойства, помеченные атрибутом:");
            foreach (var x in t.GetProperties())
            {
                object attrObj;
                if (GetPropertyAttribute(x, typeof(NewAttribute), out attrObj))
                {
                    NewAttribute attr = attrObj as NewAttribute;
                    Console.WriteLine(x.Name + " - " + attr.Description);
                }
            }
        }


        static void Main(string[] args)
        {
            AssemblyInfo();
           TypeInfo();
           AttributeInfo();
            Console.WriteLine("Вызвать метод.");
            Console.WriteLine("Введите имя метода: ");
            string name = Console.ReadLine();

            Exmpl q = new Exmpl(3, 5);
            Type e = q.GetType();
            object[] parameters = new object[] { };
            Console.WriteLine(e.InvokeMember(name, BindingFlags.InvokeMethod, null, q,
parameters));

        }
    }
}
```

Пример консольного вывода: