**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ**

**Московский государственный технический университет**

**им. Н.Э. Баумана**

**(МГТУ им. Н.Э. Баумана)**

**Радиотехнический факультет (РТ)**

Лабораторная работа № 6

По дисциплине: «Базовые компоненты интернет технологий»

# Тема: «Разработать программу, использующую делегаты. Разработать программу, реализующую работу с рефлексией.»

Выполнил: Астанов Э.М,

студент группы РТ5-31

Проверил: Гапанюк Ю.Е.,

Преподаватель каф. ИУ5

г. Москва 2017 г.

Описание задания лабораторной работы:

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 классам. Выведите только те свойства, которым назначен атрибут

6. Вызовите один из методов класса с использованием рефлексии.

Текст программы:

Часть1:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Delegate

{

//объявление делегата

delegate string SQLParseQuery(string tableName, string queryParam);

class Program

{

//Методы, реализующие делегат

static string InsertQuery(string tableName, string queryParam)

{

return "INSERT INTO " + tableName + " VALUES (" + queryParam + ") ;\n";

}

static string SelectQuery(string tableName, string queryParam)

{

return "SELECT \* FROM " + tableName + " WHERE " + queryParam + " ;\n";

}

static string DeleteQuery(string tableName, string queryParam)

{

return "DELETE FROM " + tableName + " WHERE " + queryParam + " ;\n";

}

//Использование обобщенного делегата FUNC

static void SQLFunc(string comment, string tblName, string param, Func<string,string,string> Query){

string Result = Query(tblName, param);

Console.WriteLine(comment + Result);

}

/// <summary>

/// Использование делегата

/// </summary>

static void SQLQuery(string comment, string tableName, string queryParam, SQLParseQuery Query){

string Result = Query(tableName, queryParam);

Console.WriteLine(comment + Result);

}

static void Main(string[] args)

{

string tableName="users";

string param="username=admin";

//Вызов метода, передавай делегат в качестве параметра

SQLQuery("Вставка: ", tableName, param, InsertQuery);

SQLQuery("Выборка: ", tableName, param, SelectQuery);

SQLQuery("Удаление: ", tableName, param, DeleteQuery);

//На основе лямбда функции

SQLQuery("Изменение: ", tableName, param, (string tblName, string queryParam) =>

{

return "UPDATE " + tableName + " SET " + queryParam + " ;\n";

});

Console.WriteLine("\n Использование обобщенного делегата Func");

tableName="news"; param="hidden=1";

//На основе метода

SQLFunc("Удаление новости: ", tableName, param, DeleteQuery);

//На основе лямбда

SQLFunc("Выборка новости: ", tableName, param, (string tblName, string queryParam) =>

{

return "SELECT \* FROM " + tableName + " WHERE " + queryParam + "ORDER BY asc ;\n";

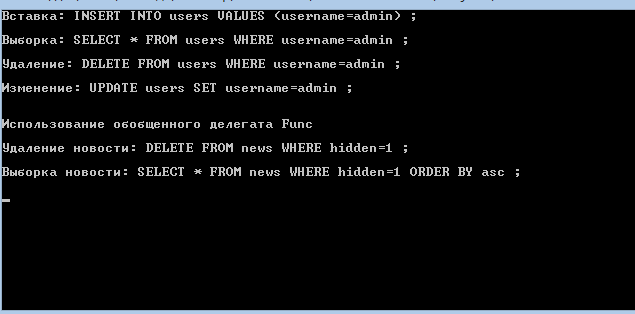
});

Console.ReadKey();

}

}

}



Часть2:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Reflection;

namespace Reflection

{

class Program

{

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 Main(string[] args)

{

Type t = typeof(TestingClass);

Console.WriteLine("Тип " + t.FullName + " унаследован от " +t.BaseType.FullName);

Console.WriteLine("Пространство имен " + t.Namespace);

Console.WriteLine("Находится в сборке " + t.AssemblyQualifiedName);

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);

}

Console.WriteLine("\nПоля данных (public):");

foreach (var x in t.GetFields())

{

Console.WriteLine(x);

}

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);

}

}

Console.WriteLine("\nВызов метода:");

TestingClass tc = (TestingClass)t.InvokeMember(null, BindingFlags.CreateInstance, null, null, new object[] { });

object[] parameters = new object[] { 3, 2 };

object Result = t.InvokeMember("Sum", BindingFlags.InvokeMethod,

null, tc, parameters);

Console.WriteLine("Sum(3,2)={0}", Result);

Console.ReadLine();

}

}

}

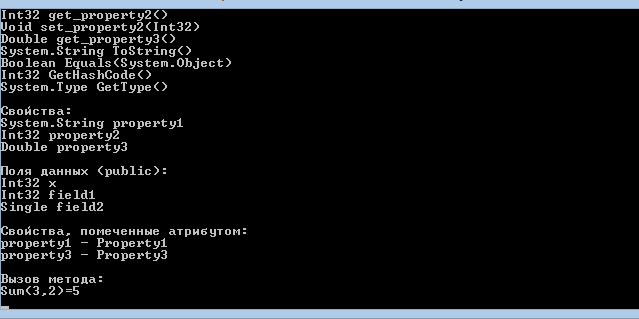
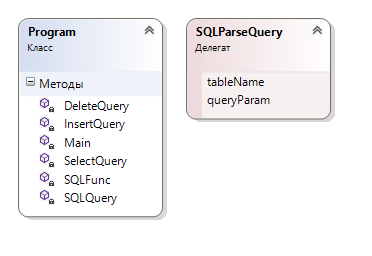


Диаграмма классов:

Часть 1:



Часть 2:

