Лабораторная работа по теме

«Регулярные выражения. Стандартные атрибуты валидации. Юнит-тестирование»

1. На основе представленного примера изучите основные возможности работы с регулярными выражениями на платформе .net.
2. Измените функциональность предыдущей лабораторной работы:
   a) добавьте в Меню пункт «Поиск» (по варианту). В поиске, кроме поиска на полное соответствие, реализовать поиск на основе регулярных выражений (диапазон, наличие букв на определенных позициях, число повторений символов и т.п.);
   b) добавьте в Меню пункт «Сортировка по». Для поиска, сортировки и модификаций существующих данных используйте LINQ to XML;
   c) добавьте панель инструментов с кнопками, дублирующими основные команды («поиск», «добавить», «очистить», «удалить» и т.д.);
   d) добавьте строку состояния с текстовыми сообщениями о текущем количестве объектов;
   e) замените разработанные вами ранее атрибуты на те, что описаны в пространстве имен **System.ComponentModel.DataAnnotations** (следует заменить только те атрибуты, функциональность которых совпадает с теми, что поставляются вместе с .net framework);
   f) при валидации вводимых данных используйте регулярные выражения;
   g) напишите unit-тесты для вашей логики. Уровень покрытия тестами должен быть более 50%.

Вариант	**Задание**
1, 8	Поиск по ФИО и специальности. Сортировка по стажу и зарплате.
2, 9	Поиск по номеру счета, ФИО пользователя. Сортировка по типу вклада и дате создания счета.
3, 10	Поиск по названию дисциплины, ФИО лектора. Сортировка по количеству лекций и темам курсового проекта.
4, 11	Поиск по названию и ФИО автора. Сортировка по количеству страниц, имени издательства.
5, 12	Поиск по названию, ФИО первооткрывателя, и стране произрастания. Сортировка по классу и виду.
6, 13, 15	Поиск по производителю и модели процессора. Сортировка по частоте работы процессора, размеру ОЗУ.
7, 14, 16	Поиск по имени авиакомпании и ФИО пилотов. Сортировка по классу самолета и производителю.

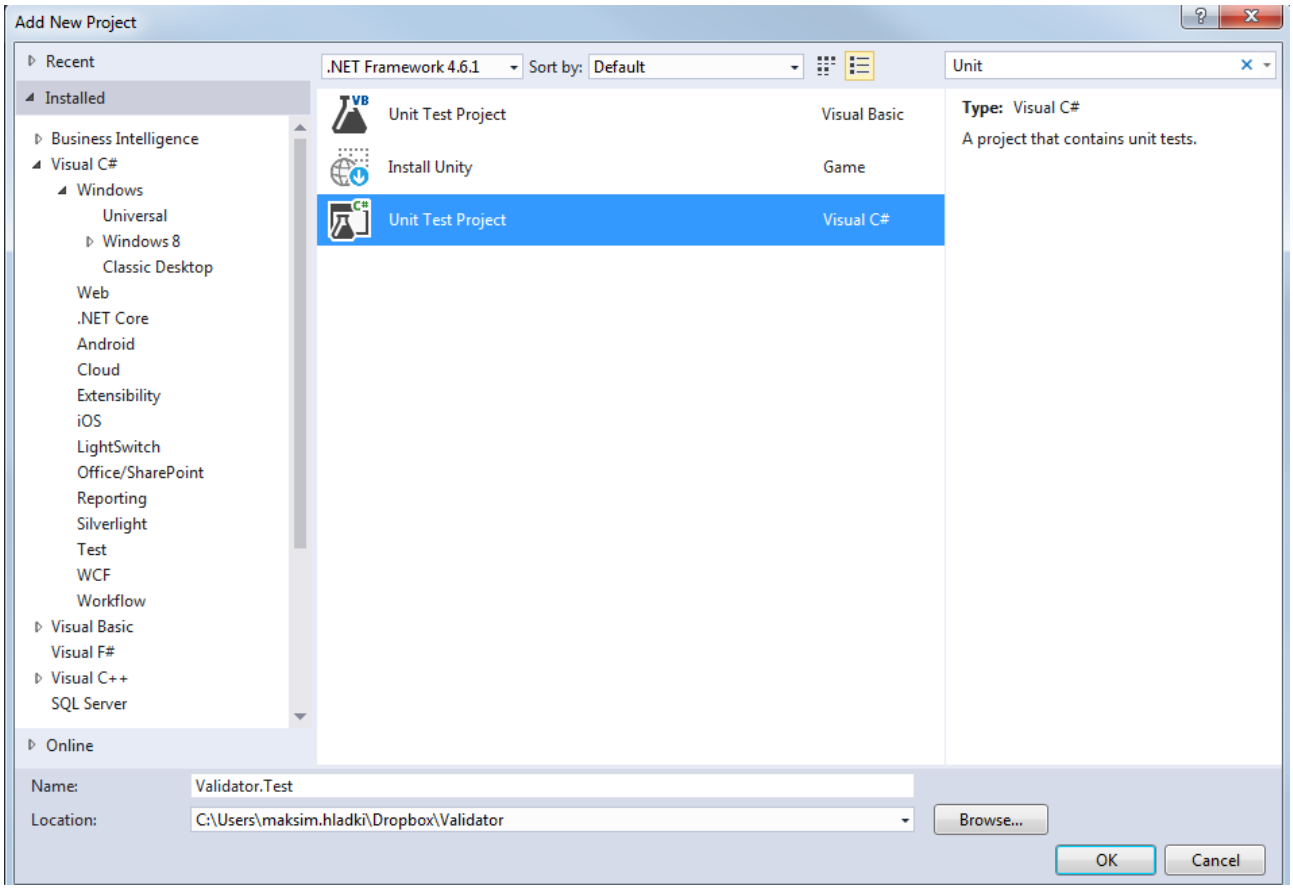
Пример кода работы с регулярными выражениями:

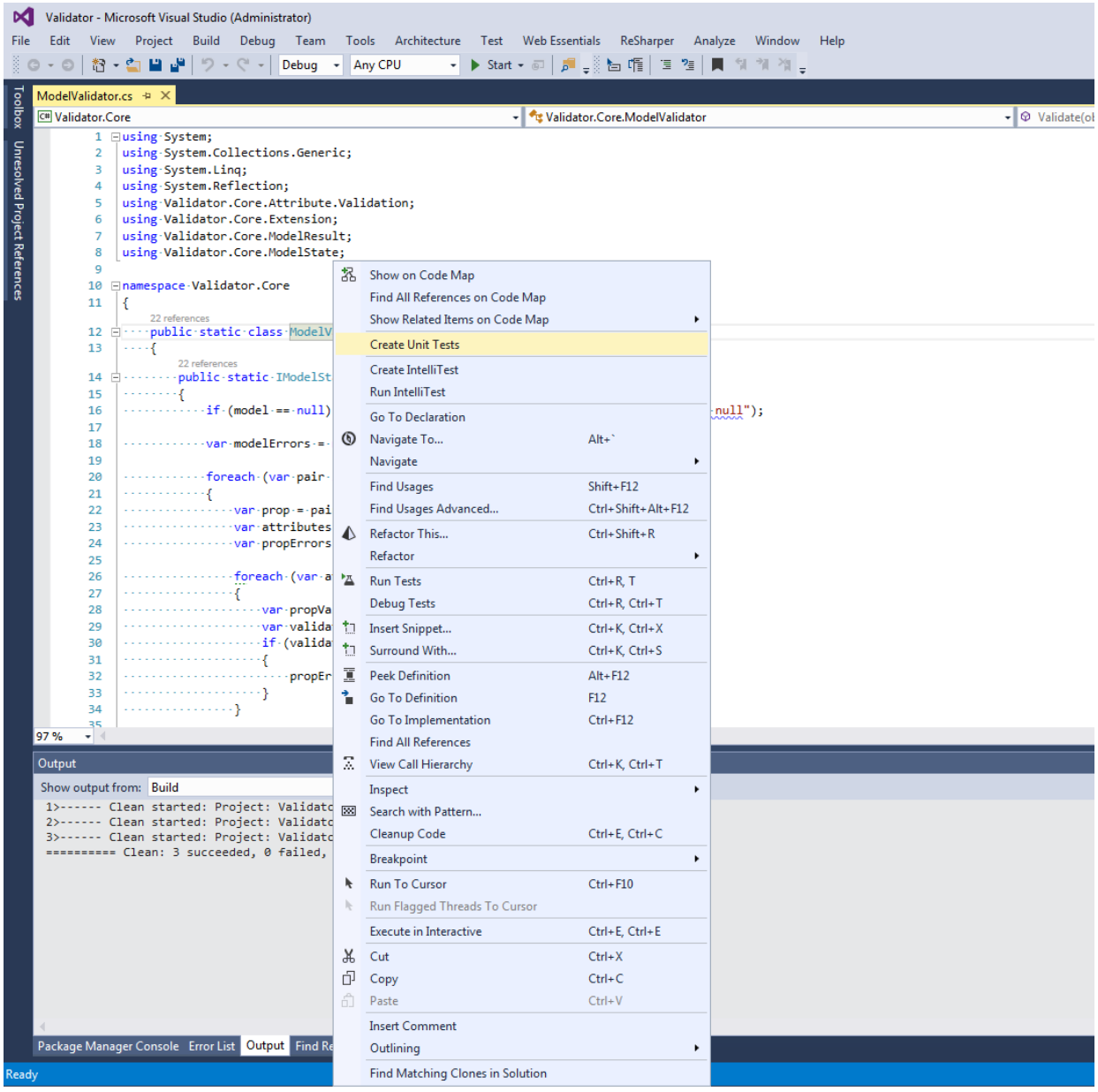
```
using System;
using System.Text.RegularExpressions;

namespace Expressions
{
    class Program
    {
        static void Main()
        {
            //RegexExample.FindSimple();
            //RegexExample.FindWords();
            //RegexExample.FindPhones();
            //RegexExample.Replace();
            //RegexExample.FindCosts();
            //RegexExample.SplitByWordBoundary();
            //RegexExample.IsMatch();
        }
    }

    /// <summary>
    /// изучите следующую информацию
    /// https://msdn.microsoft.com/ru-ru/library/az24scfc(v=vs.110).aspx
    /// </summary>
    public static class RegexExample
    {
        /// <summary>
        /// простой поиск всех вхождений подстроки "рук"
        /// </summary>
        public static void FindSimple()
        {
            var text = "Рука руку моет, а две руки – лицо.";
            var pattern = "рук";
            var matches = Regex.Matches(text, pattern);
            Console.WriteLine($"Исходный текст: {text}");

            foreach (Match match in matches) match.Print();
        }

        /// <summary>
        /// поиск всех слов подстроки "рук"
        /// использование специального метасимвола \w
        /// использование квантификатора *
        /// </summary>
        public static void FindWords()
        {
            var text = "Рука руку моет, а две руки – лицо.";
            var pattern = @"рук\w*";
            var matches = Regex.Matches(text, pattern, RegexOptions.IgnoreCase |
RegexOptions.Compiled);
            Console.WriteLine($"Исходный текст: {text}");

            foreach (Match match in matches) match.Print();
        }

        /// <summary>
        /// поиск телефонных номеров по маске +XX{X}-XX{X}-XXX-XX-XX или XX{X}-XX{X}-XXX-
XX-XX
        /// использование квантификатора ?
        /// использование специального метасимвола \d
        /// использование квантификаторов {n}, {n, m}
        /// </summary>
        public static void FindPhones()
        {
```

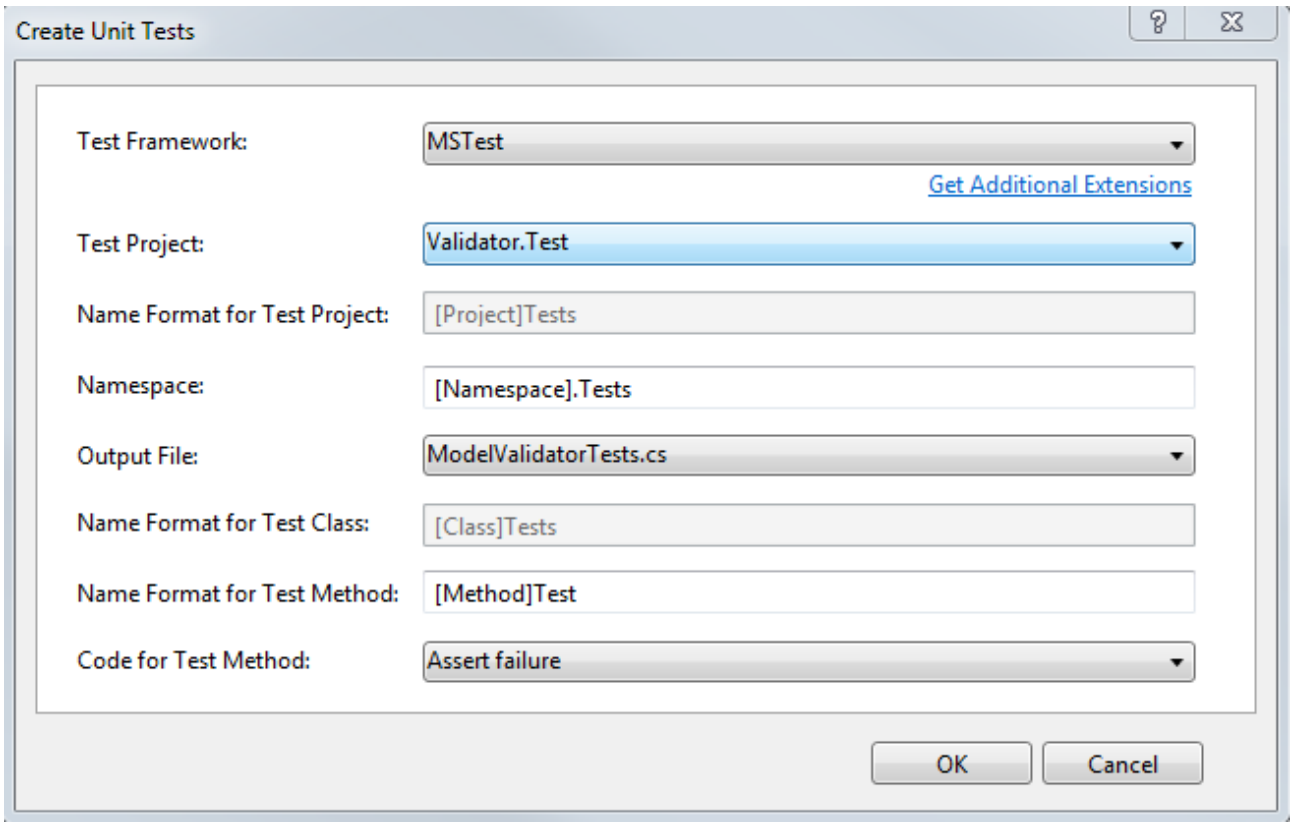
```
            var text = "Первый номер: +275-29-801-09-98; Второй номер: +19-190-232-34-00;
Третий номер: 19-210-031-34-70";
            var pattern = @"\+?\d{2,3}-\d{2,3}-\d{3}-\d{2}-\d{2}";
            var matches = Regex.Matches(text, pattern, RegexOptions.Compiled);
            Console.WriteLine($"Исходный текст: {text}");

            foreach (Match match in matches) match.Print();
        }

        /// <summary>
        /// разбиение строки на подстроки
        /// использование метасимвола \b
        /// использование диапазона символов []
        /// использование квантификатора +
        /// </summary>
        public static void SplitByWordBoundary()
        {
            var text = "Богатую взять\t—\tстанет попрекать. Умную взять — не даст слова
сказать.\nЗнатную взять — не сумеет к работе пристать. (1963 год).";
            //var regexPattern = @"\b(?<!-_)\w+";
            var regexPattern = @"\b[а-яА-Я0-9]+";
            var matches = Regex.Matches(text, regexPattern);
            Console.WriteLine($"Исходный текст: {text}");

            foreach (Match match in matches) match.Print();
        }

        /// <summary>
        /// поиск всех цен в строке
        /// исспользование групп ()
        /// использование констукции группирования ?:
        /// использование конструкции изменения |
        /// использование квантификаторов * и +
        /// использование метасимволов
        /// </summary>
        public static void FindCosts()
        {
            var text = "Сегодня в магазине я купил 1 кг. яблок за 200,50 рублей, гречку -
150 руб, 2 кг. апельсинов (230 руб.) и шоколадку за 13.5 рублей.";
            //var pattern = @"\b(\d+(\.|,)?\d+\Wруб[а-я]*)";
            var pattern = @"\b(\d+(?:\.|,)?\d+\Wруб[а-я]*)";
            Match match = Regex.Match(text, pattern, RegexOptions.IgnoreCase);
            Console.WriteLine($"Исходный текст: {text}");

            while (match.Success)
            {
                match.Print();
                match = match.NextMatch();
            }
        }

        /// <summary>
        /// замена подстрок в строке
        /// использование групп
        /// использование квантификатора ?
        /// использование констукции группирования ?:
        /// </summary>
        /// <returns></returns>
        public static void Replace()
        {
            var text = @"Сервером называется компьютер, выделенный из группы персональных
компьютеров (или рабочих станций) для выполнения какой-либо сервисной задачи без
непосредственного участия человека. Сервер и рабочая станция могут иметь одинаковую
аппаратную конфигурацию, так как различаются лишь по участию в своей работе человека за
консолью.";
```

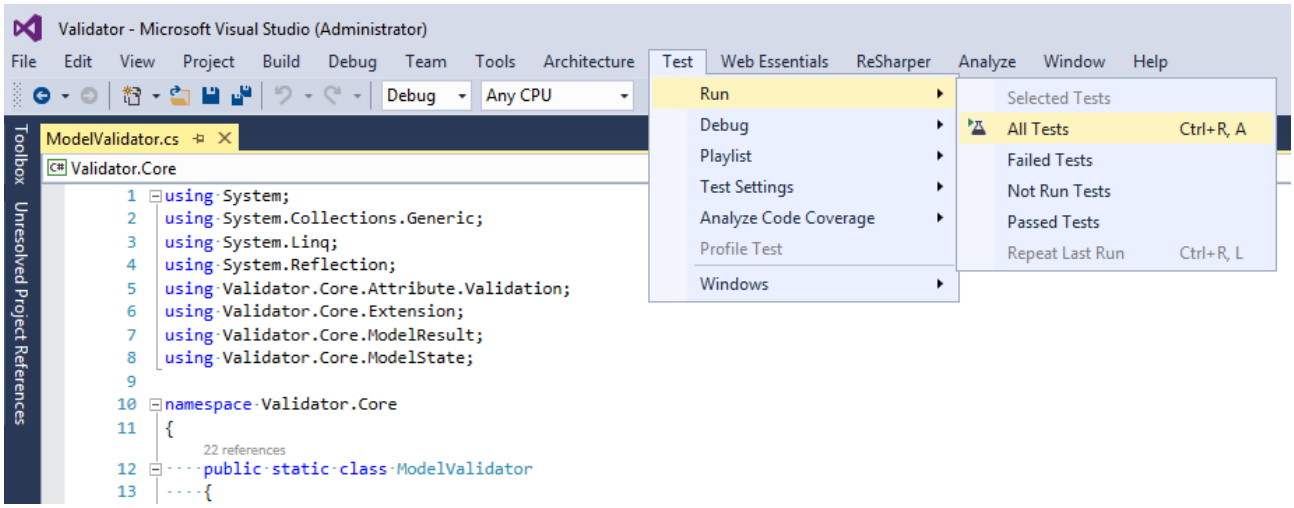
```
            var pattern = "Сервер(?:ом|у|е)?";
            var result = Regex.Replace(text, pattern, "Server", RegexOptions.IgnoreCase);

            Console.WriteLine($"Исходный текст: {text}\n");
            Console.WriteLine($"Измененный текст: {result}\n");
        }

        /// <summary>
        /// Проверяет, обнаружено ли в указанной входной строке соответствие заданному
регулярному выражению
        /// использование привязок ^ и $
        /// использование диапазонов символов []
        /// использование отрицания для диапазона сиволов [^]
        /// использование квантификатора *
        /// </summary>
        /// <returns>true/false</returns>
        public static void IsMatch()
        {
            var text = "Этот текст не должен содержать числа, двоеточие или
восклицательный знак.";
            var pattern = @"^[^0-9:!]*$";
            var result = Regex.IsMatch(text, pattern);

            Console.WriteLine($"Исходный текст: {result}");
            Console.WriteLine($"Текст не содержит число, двоеточие или восклицательный
знак: {result}");
        }

        /// <summary>
        /// метод расширения для класса Match
        /// </summary>
        /// <param name="match"></param>
        private static void Print(this Match match)
        {
            Print((Group)match);
        }

        /// <summary>
        /// метод расширения для класса Group
        /// </summary>
        /// <param name="group"></param>
        private static void Print(this Group group)
        {
            Console.WriteLine($"Index: {group.Index}\tLength: {group.Length}\tValue:
{group.Value}");
        }
    }
}
```

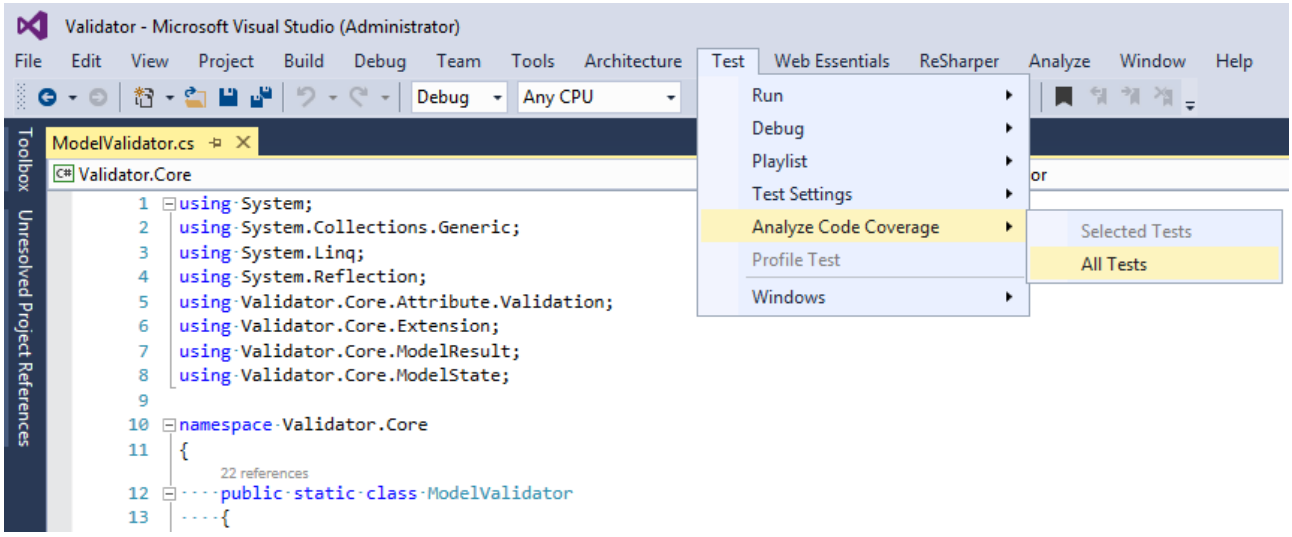
Пример работы с MS Test:

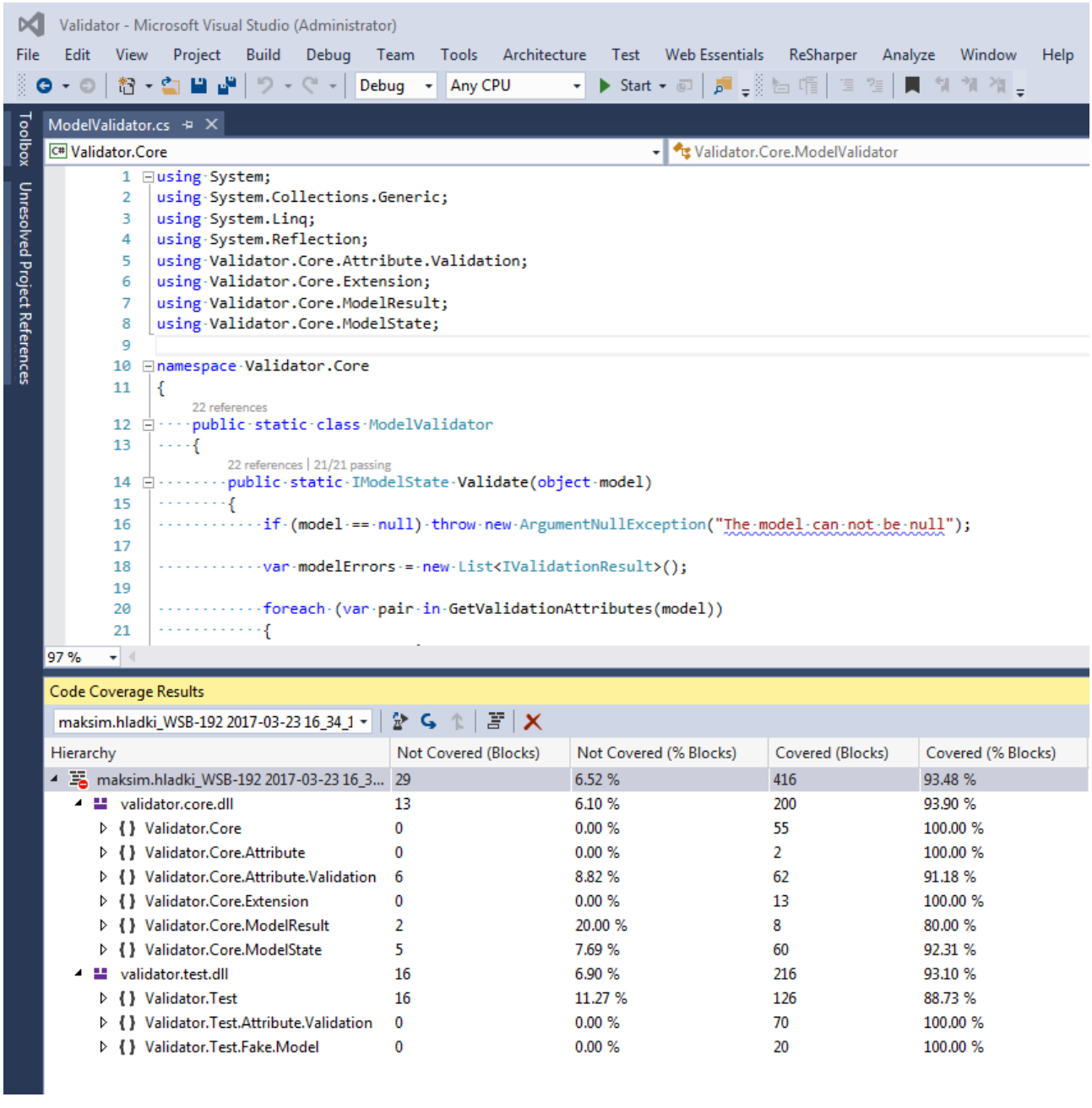



Validator - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Team Tools Architecture Test Web Essentials ReSharper Analyze Window Help
Debug
Any CPU
Start
ModelValidator.cs
Validator.Core
Validator.Core.ModelValidator
Validate(ob
Toolbox
Unresolved Project References
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reflection;
using Validator.Core.Attribute.Validation;
using Validator.Core.Extension;
using Validator.Core.ModelResult;
using Validator.Core.ModelState;
namespace Validator.Core
{
22 references
public static class ModelV
{
22 references
public static IModelSt
{
if (model == null)
var modelErrors =
foreach (var pair
{
var prop = pai
var attributes
var propErrors
foreach (var a
{
var propVa
var valida
if (valida
{
propEr
}
}
null");
97 %
Output
Show output from: Build
1>------ Clean started: Project: Validatc
2>------ Clean started: Project: Validatc
3>------ Clean started: Project: Validatc
========== Clean: 3 succeeded, 0 failed,
Package Manager Console
Error List
Output
Find Re
Ready
Show on Code Map
Find All References on Code Map
Show Related Items on Code Map
Create Unit Tests
Create IntelliTest
Run IntelliTest
Go To Declaration
Navigate To... Alt+`
Navigate
Find Usages Shift+F12
Find Usages Advanced... Ctrl+Shift+Alt+F12
Refactor This... Ctrl+Shift+R
Refactor
Run Tests Ctrl+R, T
Debug Tests Ctrl+R, Ctrl+T
Insert Snippet... Ctrl+K, Ctrl+X
Surround With... Ctrl+K, Ctrl+S
Peek Definition Alt+F12
Go To Definition F12
Go To Implementation Ctrl+F12
Find All References
View Call Hierarchy Ctrl+K, Ctrl+T
Inspect
Search with Pattern...
Cleanup Code Ctrl+E, Ctrl+C
Breakpoint
Run To Cursor Ctrl+F10
Run Flagged Threads To Cursor
Execute in Interactive Ctrl+E, Ctrl+E
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Insert Comment
Outlining
Find Matching Clones in Solution


Create Unit Tests
Test Framework:
MSTest
Get Additional Extensions
Test Project:
Validator.Test
Name Format for Test Project:
[Project]Tests
Namespace:
[Namespace].Tests
Output File:
ModelValidatorTests.cs
Name Format for Test Class:
[Class]Tests
Name Format for Test Method:
[Method]Test
Code for Test Method:
Assert failure
OK
Cancel


Validator - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Team Tools Architecture Test Web Essentials ReSharper Analyze Window Help
Debug
Any CPU
Run
Debug
Playlist
Test Settings
Analyze Code Coverage
Profile Test
Windows
Selected Tests
All Tests Ctrl+R, A
Failed Tests
Not Run Tests
Passed Tests
Repeat Last Run Ctrl+R, L
Toolbox
Unresolved Project References
ModelValidator.cs
Validator.Core
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reflection;
using Validator.Core.Attribute.Validation;
using Validator.Core.Extension;
using Validator.Core.ModelResult;
using Validator.Core.ModelState;
namespace Validator.Core
{
22 references
public static class ModelValidator
{


Validator - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Team Tools Architecture Test Web Essentials ReSharper Analyze Window Help
Debug
Any CPU
Run
Debug
Playlist
Test Settings
Analyze Code Coverage
Profile Test
Windows
Selected Tests
All Tests
ModelValidator.cs
Validator.Core
Toolbox
Unresolved Project References
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reflection;
using Validator.Core.Attribute.Validation;
using Validator.Core.Extension;
using Validator.Core.ModelResult;
using Validator.Core.ModelState;
namespace Validator.Core
{
22 references
public static class ModelValidator
{


Validator - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Team Tools Architecture Test Web Essentials ReSharper Analyze Window Help
Debug
Any CPU
Start
ModelValidator.cs
Validator.Core
Validator.Core.ModelValidator
Toolbox
Unresolved Project References
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reflection;
using Validator.Core.Attribute.Validation;
using Validator.Core.Extension;
using Validator.Core.ModelResult;
using Validator.Core.ModelState;
namespace Validator.Core
{
22 references
public static class ModelValidator
{
22 references | 21/21 passing
public static IModelState Validate(object model)
{
if (model == null) throw new ArgumentNullException("The model can not be null");
var modelErrors = new List<IValidationResult>();
foreach (var pair in GetValidationAttributes(model))
{
97 %
Code Coverage Results
maksim.hladki_WSB-192 2017-03-23 16_34_1
Hierarchy
Not Covered (Blocks)
Not Covered (% Blocks)
Covered (Blocks)
Covered (% Blocks)
maksim.hladki_WSB-192 2017-03-23 16_3... 29 6.52 % 416 93.48 %
validator.core.dll 13 6.10 % 200 93.90 %
Validator.Core 0 0.00 % 55 100.00 %
Validator.Core.Attribute 0 0.00 % 2 100.00 %
Validator.Core.Attribute.Validation 6 8.82 % 62 91.18 %
Validator.Core.Extension 0 0.00 % 13 100.00 %
Validator.Core.ModelResult 2 20.00 % 8 80.00 %
Validator.Core.ModelState 5 7.69 % 60 92.31 %
validator.test.dll 16 6.90 % 216 93.10 %
Validator.Test 16 11.27 % 126 88.73 %
Validator.Test.Attribute.Validation 0 0.00 % 70 100.00 %
Validator.Test.Fake.Model 0 0.00 % 20 100.00 %