**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:
   1. 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.);
   2. добавьте в Меню пункт «Сортировка по». Для поиска, сортировки и модификаций существующих данных используйте LINQ to XML;
   3. добавьте панель инструментов с кнопками, дублирующими основные команды («поиск», «добавить», «очистить», «удалить» и т.д.);
   4. добавьте строку состояния с текстовыми сообщениями о текущем количестве объектов;
   5. 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);
   6. при валидации вводимых данных используйте регулярные выражения;
   7. 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:











