

C# 7.0

Lesson 6: Evaluating Regular Expressions



## Lesson Objectives

- Introduction
- Forming Regular Expression
- Regex Class
- Regex Syntax
- IsMatch
- Matches
- Replace
- Split



## 6.1: Understanding Regular Expressions in C#



## Introduction to Regular Expressions

- Regular expressions are a pattern matching standard for string parsing and replacement and is a way for a computer user to express how a computer program should look for a specified pattern in text and then what the program is to do when each pattern match is found
- The regular expression engine in .NET is a powerful, full-featured tool that processes text based on pattern matches rather than on comparing and matching literal text
- In most cases, it performs pattern matching rapidly and efficiently. However, in some cases, the regular expression engine can appear to be very slow
- C# supports regular expressions through the classes in the System.Text.RegularExpressions namespace in the standard .NET framework

Regular expressions are a pattern matching standard for string parsing and replacement and is a way for a computer user to express how a computer program should look for a specified pattern in text and then what the program is to do when each pattern match is found.

The regular expression engine in .NET is a powerful, full-featured tool that processes text based on pattern matches rather than on comparing and matching literal text. In most cases, it performs pattern matching rapidly and efficiently. However, in some cases, the regular expression engine can appear to be very slow. In extreme cases, it can even appear to stop responding as it processes a relatively small input over the course of hours or even days.

### 6.1: Understanding Regular Expressions in C#

#### Forming Regular Expressions



- There are three important parts to a regular expression
- Anchors are used to specify the position of the pattern in relation to a line of text
- Character Sets match one or more characters in a single position
- Modifiers specify how many times the previous character set is repeated
  
- Example: `^#*`
  - `^` - Indicates beginning of line
  - `#` - Character set that matches single character
  - `*` - Modifier that specifies the how many time the previous character set will repeat



## 6.1: Understanding Regular Expressions in C#

### Forming Regular Expressions (Cont.)

- Anchor Characters (^ and \$):

Pattern	Matches
^A	"A" at the beginning of a line
A\$	"A" at the end of a line
A^	"A^" anywhere on a line
\$A	"\$A" anywhere on a line
^^	"^" at the beginning of a line
\$\$	"\$" at the end of a line



## 6.1: Understanding Regular Expressions in C#

## Forming Regular Expressions (Cont.)

## ▪ Character Set

Regular Expression	Matches
[]	The characters "["]
[0]	The character "0"
[0-9]	Any number
[^0-9]	Any character other than a number
[-0-9]	Any number or a "-"
[0-9-]	Any number or a "-"
[^0-9-]	Any character except a number or a "-"
[]0-9]	Any number or a "]"
[0-9]]	Any number followed by a "]"
[0-9-z]	Any number, or any character between "9" and "z"
[0-9\ -a\]]	Any number, or a "-", a "a", or a "]"

### 6.1: Understanding Regular Expressions in C# Forming Regular Expressions (Cont.)



- Example:

Quantifier	Description	Regex	Matches
*	Matches the preceding character zero or more times.	a*b	b, ab, aab, aaab, etc.
+	Matches the preceding character 1 or more times	a+b	ab, aab, aaab, etc.
?	Matches the preceding character zero or one time	a?b	b, ab
^	It is used to match the beginning of a string.	^ Capgemini	Capgemini holds the strength of more than 1 lakh employees in India.
\$	It is used to match the end of a string.	\$ Capgemini	I work with Capgemini
.	Matches any character only once.	C.P	CAP, CEP, COP

## 6.1: Understanding Regular Expressions in C# Forming Regular Expressions (Cont.)



- Example:

Quantifier	Description
\s*	Match zero or more white-space characters.
\s?	Match zero or one white-space character.
\d?	Match zero or more decimal digits.



## 6.1: Understanding Regular Expressions in C# Forming Regular Expressions (Cont.)



- Example:
- Literals and Special Characters:

**REGEX:** TX

**INPUT:** TX      **MATCH:** true

**INPUT:** AZ      **MATCH:** false

- Character Range:

**REGEX:** [013][FXB]

**INPUT:** 1X      **MATCH:** true

**INPUT:** 1Z      **MATCH:** false

### 6.1: Understanding Regular Expressions in C# Forming Regular Expressions (Cont.)



- Character Range: (Cont.)

**REGEX:** [A-Za-z0-9][0-9]

**INPUT:** i5      **MATCH:** true

**INPUT:** 1X      **MATCH:** false

- REGEX: [^AEIOU]

**INPUT:** X      **MATCH:** true

**INPUT:** E      **MATCH:** false

## 6.1: Understanding Regular Expressions in C# Forming Regular Expressions (Cont.)



- Quantifiers:

**REGEX:** [A-Z][A-Z][A-Z]

**INPUT:** YCA     **MATCH:** true

**REGEX:** [A-Z]{3}

**INPUT:** YCA     **MATCH:** true

**REGEX:** [0-9]{3}-[0-9]{3}-[0-9]{4}

**INPUT:** 470-127-7501     **MATCH:** true

**INPUT:** 75663-2372     **MATCH:** false

## 6.1: Understanding Regular Expressions in C#

## Forming Regular Expressions (Cont.)



**REGEX:** [A-Za-z0-9]{2,}

**INPUT:** YZ1    **MATCH:** true

**INPUT:** YZSDjhfhSBH2342SDFSDfsw123412    **MATCH:** true

**REGEX:** [0-3]+[XYZ]\*

**INPUT:** 34    **MATCH:** true

**INPUT:** 34YYXZZ    **MATCH:** true

### 6.1: Understanding Regular Expressions in C# Forming Regular Expressions (Cont.)



- Alternation:

**REGEX:** [0-9]{3}(35|75)

**INPUT:** 75035 **MATCH:** true

**INPUT:** 75062 **MATCH:** false

### 6.1: Understanding Regular Expressions in C#

#### IsMatch() Function



- `public bool IsMatch(string input):`
  - Indicates whether the regular expression specified in the `Regex` constructor finds a match in a specified input string.
- `public bool IsMatch(string input, int startat) :`
  - Indicates whether the regular expression specified in the `Regex` constructor finds a match in the specified input string, beginning at the specified starting position in the string.
- `public static bool IsMatch(string input, string pattern) :`
  - Indicates whether the specified regular expression finds a match in the specified input string

```
class IsMatchDemo
{
    public static bool IsValid(string value)
    {
        return Regex.IsMatch(value, @"^[a-zA-Z0-9]*$");
    }

    static void Main()
    {
        Console.WriteLine(IsValid("DotNetLearningAndDevelopment2019"));
        Console.WriteLine(IsValid("DotNet Learning And Development 2019"));
    }
}
```

## 6.1: Understanding Regular Expressions in C#

## Matches(), Replace(), Split() Function



- `public MatchCollection Matches(string input)`
  - Searches the specified input string for all occurrences of a regular expression
- `public string Replace(string input, string replacement)`
  - In a specified input string, replaces all strings that match a regular expression pattern with a specified replacement string
- `public string[] Split(string input)`
  - Splits an input string into an array of substrings at the positions defined by a regular expression pattern specified in the `Regex` constructor

```
class MatchCollectionDemo
{
    private static void showMatch(string text, string expression)
    {
        Console.WriteLine("The Expression: " + expression);
        MatchCollection matchCollection = Regex.Matches(text, expression);
        foreach (Match match in matchCollection)
        {
            Console.WriteLine(match);
        }
    }

    static void Main(string[] args)
    {
        string input = "The Sun Sets in South-west during winter";

        Console.WriteLine("Matching words that start with 'S': ");
        showMatch(input, @"^\bS\S*");
        Console.ReadKey();
    }
}
```

## Demo



- Implementing Regular Expressions in C#





## Summary



- In this lesson you have learnt:
  - Introduction to Regular Expressions in C#
  - Forming regular expressions in C#
  - Introduction to IsMatch(), Matches(), Replace(), Split() Functions



### Review Question



#### Question 1:

- What are Regular Expressions in C#?

#### Question 2:

Which of the following functions does the job of replacing all strings that match a regular expression pattern with a specified replacement string

- ReplaceAll()
- Replace()
- ReplaceString()
- All of the above

