How to validate a phone number

Ask Question

12

3

A valid phone number contains:

less than 9 characters, a "+" at the start, and only digits.

Im trying to use regular expressions but i've only started using them and im not good at it. The code i have so far is:

```
static void Main(string[] args)
{
    Console.WriteLine("Enter a phone number.");
    string telNo = Console.ReadLine();

    if (Regex.Match(telNo, @"^(\+[0-9])$").Success)
        Console.WriteLine("correctly entered");

    else
        Console.WriteLine("incorrectly entered");

    Console.ReadLine();
}
```

But i don't know how to check the length of the string this way. Any help is appreciated.

c# regex

edited Apr 30 '15 at 15:13

Jake
1,211 2 11 32

asked Apr 30 '15 at 14:19

Adam Higgins
153 1 2 14

3 possible duplicate of A comprehensive regex for phone number validation – sab669 Apr 30 '15 at 14:21

2 You want to use your regex in server code (c#) or in java script? I'm not sure but there could be litte differences – Jacek Apr 30 '15 at 14:52

thanks for the help everyone, the one that worked for me in the end was @"^(\+\d[0-9]{1,8})$" – Adam Higgins Apr 30 '15 at 17:54

1 Does a valid phone number contain less than 9 characters, does this not depend on the country you are in? – JsonStatham May 11 '18 at 18:34

Gah. This question is horrible; it provides a definition of "valid phone number" that just straight up is not the true definition of what a valid phone number is (I don't know of a single country where a phone number including country code is less than 9 characters, actually - it's at least not true for the US or UK), but the current title gives no hint of this and the answers take the weird definition at face value. A title edit designed to anti-SEO this question and prevent Googlers from landing here seems to be in order... – Mark Amery Nov 14 '18 at 15:00

Home

PUBLIC

Stack Overflow

Tags

Users

Jobs

8 Answers

16

Jacek's regex works fine

```
public class Program
{
    public static void Main()
    {
        Console.WriteLine("Enter a phone number.");
        string telNo = Console.ReadLine();
```

```
            Console.WriteLine("{0}correctly entered", IsPhoneNumber(telN
            Console.ReadLine();
        }

        public static bool IsPhoneNumber(string number)
        {
            return Regex.Match(number, @"^(\+[0-9]{9})$").Success;
        }
    }
```

answered Apr 30 '15 at 14:44

Greg
740 1 5 16

16

Your regex should look like this, you need information about char counter

```
@"^(\+[0-9]{9})$"
```

answered Apr 30 '15 at 14:20

Jacek
4,842 16 45 95

it says incorrectly entered in every scenario – Adam Higgins Apr 30 '15 at 14:23

2 @Adam Higgins There regex101.com find more help. This regex look ok – Jacek Apr 30 '15 at 14:31

1 What is the setting at that site for C# regexes? I only see PHP, Javascript, and Python. – krillgar Apr 30 '15 at 15:18

@Jacek What about just phone code? (+1, +880, +965 etc) – FaizanRabbani Oct 13 '15 at 6:38

@FaizanRabbani Do you want give input starting with plus? – Jacek Oct 13 '15 at 7:37

2

Something like this could work:

^+\d{0,9}

But I would suggest playing around with a regex tester to learn more about how regular expressions work. I still like to use them heavily myself, as I don't write regular expressions often. Here is one example but there are many more out there.

https://regex101.com/

answered Apr 30 '15 at 14:25

maniak1982

632 2 6 23

1

Simple function for Valid USAPhoneNumber or not.

```
    /// <summary>
     /// Allows phone number of the format: NPA = [2-9][0-8][0-9] Nxx
[0-9] Station = [0-9][0-9][0-9][0-9]
    /// </summary>
    /// <param name="strPhone"></param>
    /// <returns></returns>
    public static bool IsValidUSPhoneNumber(string strPhone)
    {
        string regExPattern = @"^[01]?[- .]?(\([2-9]\d{2}\)|[2-9]\d{
\d{4}$";
        return MatchStringFromRegex(strPhone, regExPattern);
    }
    // Function which is used in IsValidUSPhoneNumber function
    public static bool MatchStringFromRegex(string str, string regex
    {
        str = str.Trim();
        System.Text.RegularExpressions.Regex pattern = new
System.Text.RegularExpressions.Regex(regexstr);
```

```
        return pattern.IsMatch(str);
    }
```

edited Aug 22 '16 at 13:29

answered Aug 22 '16 at 13:22

Amit Gorvadiya

11 3

1

If you're looking for a country specific regex, try this expression which works for all Australian (+61-) numbers. I've put comments on how to go about varying it for other uses.

```
public static bool IsValidPhoneNumber(string phoneNumber)
{
    //will match +61 or +61- or 0 or nothing followed by a nine digi
    return Regex.Match(phoneNumber,
        @"^([\+]?61[-]?|[0])?[1-9][0-9]{8}$").Success;
    //to vary this, replace 61 with an international code of your ch
    //or remove [\+]?61[-]? if international code isn't needed
    //{8} is the number of digits in the actual phone number less on
}
```

answered Jun 2 '17 at 9:54

Ash

3,122 4 12 36

This solution validates every test criteria for validating a phone number, it also leverages from the Regex API. Criteria includes spacing, any non numeric values, area codes (which you specify),

0

number of values (digits) the phone number should have, and also includes error messaging as well as phone number old and new state.

Here is the source code:

```
public class PhoneNumberValidator
{
    public string ErrorMessage { get; set; }
    public int PhoneNumberDigits { get; set; }
    public string CachedPhoneNumber { get; set; }

    private Dictionary<int, string> VaildAreaCodes()
    {
        return new Dictionary<int, string>
        {
            [3] = "0",
            [4] = "27"
        };
    }

    private bool IsInteger(string value)
    {
        return int.TryParse(value, out int result);
    }

    private string GetConsecutiveCharsInPhoneNumberStr(string phoneN
    {
        switch (PhoneNumberDigits)
        {
            case 0:
            case 10:
                PhoneNumberDigits = 10;
                return phoneNumber.Substring(phoneNumber.Length - 7)

            case 11:
                return phoneNumber.Substring(phoneNumber.Length - 8)

            default:
                return string.Empty;
        }
    }

    private bool IsValidAreaCode(ref string phoneNumber, string area
    {
```

```
            if (!IsInteger(areaCode))
            {
                ErrorMessage = "Area code characters of Phone Number valu
contain integers.";
                return false;
            }

            var areaCodeLength = areaCode.Length;
            var invalidAreaCodeMessage = "Phone Number value contains inv
            switch (areaCodeLength)
            {
                case 2:
                    phoneNumber = string.Concat("0", phoneNumber);
                    return true;

                case 3:
                    if (!areaCode.StartsWith(VaildAreaCodes[3]))
                        ErrorMessage = invalidAreaCodeMessage;
                    return string.IsNullOrWhiteSpace(ErrorMessage) ? tru

                case 4:
                    if (areaCode.StartsWith(VaildAreaCodes[4]))
                    {
                        phoneNumber = string.Concat("0", phoneNumber.Rem
replace first two charaters with zero
                        return true;
                    }
                    ErrorMessage = invalidAreaCodeMessage;
                    return false;

                default:
                    ErrorMessage = invalidAreaCodeMessage;
                    return false;
            }
        }

        public bool IsValidPhoneNumber(ref string phoneNumber)
        {
            CachedPhoneNumber = phoneNumber;

            if (string.IsNullOrWhiteSpace(phoneNumber))
            {
                ErrorMessage = "Phone Number value should not be equival
                return false;
            }

            phoneNumber = Regex.Replace(phoneNumber, " {2,}", string.Emp
```

```
whitespaces
        phoneNumber = Regex.Replace(phoneNumber, "[^0-9]", string.Em
non numeric characters

        var lastConsecutiveCharsInPhoneNumberStr =
GetConsecutiveCharsInPhoneNumberStr(phoneNumber);

        if (string.IsNullOrWhiteSpace(lastConsecutiveCharsInPhoneNuml
        {
            ErrorMessage = "Phone Number value not supported.";
            return false;
        }

        if (!IsInteger(lastConsecutiveCharsInPhoneNumberStr))
        {
            ErrorMessage = "Last consecutive characters of Phone Numl
only contain integers.";
            return false;
        }

        var phoneNumberAreaCode =
phoneNumber.Replace(lastConsecutiveCharsInPhoneNumberStr, "");

        if (!IsValidAreaCode(ref phoneNumber, phoneNumberAreaCode))
        {
            return false;
        }

        if (phoneNumber.Length != PhoneNumberDigits)
        {
            ErrorMessage = string.Format("Phone Number value should
characters instead of {1} characters.", PhoneNumberDigits, phoneNumb
            return false;
        }

        return true;
    }
}
```

The solution is highly configurable and may be used for any digits phone number as well as area code.

edited Mar 6 '18 at 6:32

answered Jan 18 '18 at 13:09

Tech
162 1 9

0

Expanding upon one of the answers provided above, the method I came up with to also handle a few phone number delivery styles as well as international phone number is

```
internal static bool IsValidPhoneNumber(this string This)
{
    var phoneNumber = This.Trim()
        .Replace(" ", "")
        .Replace("-", "")
        .Replace("(", "")
        .Replace(")", "");
    return Regex.Match(phoneNumber, @"^\+\d{5,15}$").Success;
}
```

answered Mar 8 at 11:10

Bolorunduro Winner-Timothy
65 1 12

0

DON'T USE A REGULAR EXPRESSION!!

There are too many variables for a regex to be of any use. Instead, just remove all characters from your string that are not 0-9, and then check to see if you have the correct number of digits left. Then it doesn't matter what extra stuff the user includes or doesn't include... ()x-+[] etc etc, as it just strips them all out and only counts the characters 0-9.

I've got a string extension that works great, and allows for a wide range of formats. It accepts an `IsRequired` parameter. So, you can validate a phone number like this:

```
string phone = "(999)999-9999"
bool isValidPhone = phone.ValidatePhoneNumber(true) // returns true

string phone ="1234567890"
bool isValidPhone = phone.ValidatePhoneNumber(true) // returns true

string phone = ""
bool isValidPhone = phone.ValidatePhoneNumber(false) // not required

string phone = ""
bool isValidPhone = phone.ValidatePhoneNumber(true) // required, so

string phone ="12345"
bool isValidPhone = phone.ValidatePhoneNumber(true) // returns false

string phone ="foobar"
bool isValidPhone = phone.ValidatePhoneNumber(true) // returns false
```

Here's the code (assumes a 10-digit American phone number. Adjust accordingly):

```
public static class StringExtensions
{

    /// <summary>
    /// Checks to be sure a phone number contains 10 digits as per A
numbers.
    /// If 'IsRequired' is true, then an empty string will return Fa
    /// If 'IsRequired' is false, then an empty string will return T
    /// </summary>
    /// <param name="phone"></param>
    /// <param name="IsRequired"></param>
    /// <returns></returns>
    public static bool ValidatePhoneNumber(this string phone, bool I
    {
        if (string.IsNullOrEmpty(phone) & !IsRequired)
            return true;

        if (string.IsNullOrEmpty(phone) & IsRequired)
            return false;
```

```
            var cleaned = phone.RemoveNonNumeric();
            if (IsRequired)
            {
                if (cleaned.Length == 10)
                    return true;
                else
                    return false;
            }
            else
            {
                if (cleaned.Length == 0)
                    return true;
                else if (cleaned.Length > 0 & cleaned.Length < 10)
                    return false;
                else if (cleaned.Length == 10)
                    return true;
                else
                    return false; // should never get here
            }
        }

        /// <summary>
        /// Removes all non numeric characters from a string
        /// </summary>
        /// <param name="phone"></param>
        /// <returns></returns>
        public static string RemoveNonNumeric(this string phone)
        {
            return Regex.Replace(phone, @"[^0-9]+", "");
        }
    }
```

answered May 31 '17 at 11:18

Casey Crookston

3,320 7 34 78