C# code to validate email address

Ask Question

368

What is the most elegant code to validate that a string is a valid email address?

c# email email-validation

96

edited Jul 10 '13 at 15:48
Artemix
1,630 1 18 30

asked Sep 2 '09 at 1:01
leora
40.3k 292 764 1269

8 regular-expressions.info/email.html – Noon Silk Sep 2 '09 at 1:04

7 Have a look at Phil Haack's article: "I Knew How To Validate An Email Address Until I Read The RFC" – Luke Quinane Sep 2 '09 at 1:07

38 Answers

1 2 next

635

What about this?

```
bool IsValidEmail(string email)
{
    try {
        var addr = new System.Net.Mail.MailAddress(email);
        return addr.Address == email;
    }
    catch {
        return false;
    }
}
```

To clarify, the question is asking whether a particular string is a valid representation of an e-mail address, not whether an e-mail address is a valid destination to send a message. For that, the only real way is to send a message to confirm.

Note that e-mail addresses are more forgiving than you might first assume. These are all perfectly valid forms:

- cog@wheel
- "cogwheel the orange"@example.com
- 123@$.xyz

For most use cases, a false "invalid" is much worse for your users and future proofing than a false "valid". Here's an article that used to be the accepted answer to this question (that answer has since been deleted). It has a lot more detail and some other ideas of how to solve the problem.

Providing sanity checks is still a good idea for user experience. Assuming the e-mail address is valid, you could look for known top-level domains, check the domain for an MX record, check for spelling errors from common domain names (gmail.cmo), etc. Then present a warning giving the user a chance to say "yes, my mail server really does allow 🌮🍳🎁 as an email address."

As for using exception handling for business logic, I agree that is a thing to be avoided. But this is one of those cases where the convenience and clarity may outweigh the dogma.

Besides, if you do anything else with the e-mail address, it's probably going to involve turning it to a MailAddress. Even if you don't use this exact function, you will probably want to use the same pattern. You can also check for specific kinds of failure by catching different exceptions: null, empty, or invalid format.

Per Stuart's comment, this compares the final address with the original string instead of always returning true. MailAddress tries to parse a string with spaces into "Display Name" and "Address" portions, so the original version was returning false positives.

--- Further reading ---

Documentation for System.Net.Mail.MailAddress

Explanation of what makes up a valid email address

edited May 8 '18 at 20:37

answered Sep 3 '09 at 16:50
Cogwheel
17k 3 37 65

11 i don't think it works. i just tried it with simply a@a and it returned true. thats wrong. – Mouffette Oct 11 '09 at 5:59

19 Actually, that's not incorrect. a@a is a valid e-mail address. See haacked.com/archive/2007/08/21/… In fact, this method does return incorrect results if you use an address with quotes. – Cogwheel Oct 15 '09 at 13:37

42 +1: This is the best answer if you're using the `System.Net.Mail` classes

to send mail, which you probably are if you're using .NET. We made the decision to use this type of validation simply because there is no point in us accepting email addresses - even valid ones - that we cannot send mail to. – Greg Beech Feb 25 '10 at 17:18

20 I don't recommend. It returns true: `IsValidEmail("this is not valid@email$com");` – Kakashi Dec 11 '11 at 18:33

13 I like this; I don't get the compulsion to be more restrictive than the actual spec. False negatives are way worse than false positives ("what do you mean my e-mail is *invalid?*") – Casey Jul 30 '14 at 17:51

202

This is an old question, but all the answers I've found on SO, including more recent ones, are answered similarly to this one. However, in .Net 4.5 / MVC 4 you can add email address validation to a form by adding the [EmailAddress] annotation from System.ComponentModel.DataAnnotations, so I was wondering why I couldn't just use the built-in functionality from .Net in general.

This seems to work, and seems to me to be fairly elegant:

```
using System.ComponentModel.DataAnnotations;

class ValidateSomeEmails
{
    static void Main(string[] args)
    {
        var foo = new EmailAddressAttribute();
        bool bar;
        bar = foo.IsValid("someone@somewhere.com");         //true
        bar = foo.IsValid("someone@somewhere.co.uk");       //true
        bar = foo.IsValid("someone+tag@somewhere.net");     //true
        bar = foo.IsValid("futureTLD@somewhere.fooo");      //true

        bar = foo.IsValid("fdsa");                          //false
        bar = foo.IsValid("fdsa@");                         //false
        bar = foo.IsValid("fdsa@fdsa");                     //false
        bar = foo.IsValid("fdsa@fdsa.");                    //false
```

```
            //one-liner
            if (new EmailAddressAttribute().IsValid("someone@somewhere.co
                bar = true;
        }
    }
```

edited Jul 10 '13 at 15:36

answered May 6 '13 at 16:45

imjosh
3,241 1 12 20

4 Cool, although it's disappointing that MS can't make this agree with [their own documentation](). This rejects js@proseware.com9 – Casey Jul 30 '14 at 18:27

3 If you use [EmailAddress] in your view model (as I do), this is a slick way to ensure that you use the same validation logic -- for better or worse -- in your code. – Bob.at.Indigo.Health Nov 2 '14 at 16:33

11 Note that `EmailAddressAttribute` is less permissive than `System.Net.Mail.MailAddress` - for instance, `MailAddress` accepts an address for a TLD. Just something to keep in mind if you need to be as permissive as possible. – Aaroninus Feb 4 '15 at 17:11

4 @hofnarwillie: you did not read the comments, did you? – Cogwheel Dec 14 '15 at 18:29

4 @hofnarwillie: it is in the main body, but the comments elaborate. If your goal is to not anger your users then your validation shouldn't be any more restrictive than the spec. The only way to truly verify whether an e-mail is valid is to send a test message. – Cogwheel Dec 15 '15 at 23:40

I use this single liner method which does the work for me-

38

```
using System.ComponentModel.DataAnnotations;
public bool IsValidEmail(string source)
{
    return new EmailAddressAttribute().IsValid(source);
}
```

As per the comments, this will "fail" if the `source` (the email address) is null.

```
public static bool IsValidEmailAddress(this string address) => addre
EmailAddressAttribute().IsValid(address);
```

edited Dec 11 '18 at 9:40
Dave
4,931 9 51 93

answered Nov 26 '15 at 5:58
Manik Arora
3,397 16 41

1 This doesn't work for me; I get, "'Data Annotations' does not exist...are you missing an assembly reference?" Which reference do I need to add? – B. Clay Shannon Feb 12 '16 at 17:17

2 A better version: `public static Boolean IsValidMailAddress(this String pThis) => pThis == null ? false : new EmailAddressAttribute().IsValid(pThis);` – Sebastian Hofmann Mar 7 '18 at 13:04

4 Or even better: `public static bool IsValidEmailAddress(this string address) => address != null && new EmailAddressAttribute().IsValid(address);` – Patrik Melander Jun 19 '18 at 8:46

.net 4.5 added

37

System.ComponentModel.DataAnnotations.EmailAddressAttribute

You can browse the EmailAddressAttribute's source, this is the Regex it uses internally:

```
const string pattern = @"^((([a-z]|\d|[!#\$%&'\*\+\-\/=\?\^_`{\|}~]|
\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])+(\.([a-z]|\d|[!#\$%&'\*\+\-\/=\?\
\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])+)*)|((\x22)((((\x20|\x09)*(\x0d\x
(([\x01-\x08\x0b\x0c\x0e-\x1f\x7f]|\x21|[\x23-\x5b]|[\x5d-\x7e]|[\u0
\uFDCF\uFDF0-\uFFEF])|(\\([\x01-\x09\x0b\x0c\x0d-\x7f]|[\u00A0-\uD7F
\uFDCF\uFDF0-\uFFEF]))))*(((\x20|\x09)*(\x0d\x0a))?(\x20|\x09)+)?(\x
[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])|(([a-z]|\d|[\u00A0-\uD7FF
\uFFEF])([a-z]|\d|-|\.|_|~|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF]
\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])))\.)+(([a-z]|[\u00A0-\uD7FF\uF900
\uFFEF])|(([a-z]|[\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])([a-z]|\d
\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])*([a-z]|[\u00A0-\uD7FF\uF900-\uFDC
\uFFEF])))\.?$";
```

edited Mar 15 '16 at 13:22

Thomas Ayoub
23.2k 14 62 107

answered Jul 17 '13 at 23:57

Chad Grant
33.7k 8 50 71

1 Unfortunately the EmaillAddressAttribute allows Ñ which is not a valid character for email – B Z Sep 6 '13 at 18:06

13 @BZ Yes it is. Why do you think it isn't? – Casey Jul 30 '14 at 18:31

I took Phil's answer from #1 and created this class. Call it like this:

31

```
bool isValid = Validator.EmailIsValid(emailString);
```

Here is the class:

```
using System.Text.RegularExpressions;

public static class Validator
{

    static Regex ValidEmailRegex = CreateValidEmailRegex();

    /// <summary>
    /// Taken from http://haacked.com/archive/2007/08/21/i-knew-how-
email-address-until-i.aspx
    /// </summary>
    /// <returns></returns>
    private static Regex CreateValidEmailRegex()
    {
        string validEmailPattern = @"^(?!\.)(""([^""\r\\]|\\[""\r\\]
            + @"([-a-z0-9!#$%&'*+/=?^_`{|}~]|(?<!\.)\.)*)(?<!\.)"
            + @"@[a-z0-9][\w\.-]*[a-z0-9]\.[a-z][a-z\.]*[a-z]$";

        return new Regex(validEmailPattern, RegexOptions.IgnoreCase)
    }

    internal static bool EmailIsValid(string emailAddress)
    {
        bool isValid = ValidEmailRegex.IsMatch(emailAddress);

        return isValid;
    }
}
```

answered Dec 18 '11 at 20:54

David Silva Smith
6,597 10 55 84

4 @PeterKellner I ran it and it returned valid for the string jxxxx@linkedin.com. It seems strange you would guess it doesn't work instead of executing the code... – David Silva Smith Jul 30 '13 at 22:28

5 Just a small one, but I would use: return (!string.IsNullOrEmpty(emailAddress)) && ValidEmailRegex.IsMatch(emailAddress); – Marc Nov 15 '13 at 14:16

27

Personally, I would say that you should just make sure there is an @ symbol in there, with possibly a . character. There's many regexes you could use of varying correctness, but I think most of these leave out valid email addresses, or let invalid ones through. If people want to put in a fake email address, they will put in a fake one. If you need to verify that the email address is legit, and that the person is in control of that email address, then you will need to send them an email with a special coded link so they can verify that it indeed is a real address.

answered Sep 2 '09 at 1:09
Kibbee
51.7k 26 133 172

6 I personally think you should do a bit more validation than that. Someone's bound to try Bobby Table's email address or worse. – Luke Quinane Sep 2 '09 at 1:20

30 What's wrong with bobby table's email address if you're using prepared statements. We're talking about valid email addresses, not other things that have nothing to do with what constitues a valid email address, like how to properly do SQL queries so that you don't get SQL injection problems. – Kibbee Sep 2 '09 at 2:06

9 That's for the mail system to worry about. I wouldn't go around rejecting perfectly valid email addresses just because it could be a security issue with some other system. If all your email server needs is a malformed email address to cause security problems, you should probably switch to another server. – Kibbee Sep 2 '09 at 15:44

3 Defence in depth only works if each level of your security onion is not rotten. One rotten layer means you spoil the whole onion. Rejecting "foo@example.com.au" because you want to defend against vulnerabilities in Sun's μ-law encoding doesn't make sense, does it? Don't laugh, it's happened to me. The reason I am here commenting is that Medicare Australia doesn't allow ".au" addresses, only ".com". Also read Mark Swanson, "How not to validate email, ", mdswanson.com/blog/2013/10/14/… – ManicDee Nov 22 '13 at 5:21

14

I think the best way is as follow:

```
    public static bool emailIsValid(string email)
    {
        string expresion;
        expresion = "\\w+([-+.']\\w+)*@\\w+([-.]\\w+)*\\.\\w+([-.]\\
        if (Regex.IsMatch(email, expresion))
        {
            if (Regex.Replace(email, expresion, string.Empty).Length
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        else
        {
            return false;
        }
    }
```

You can have this static function in a general class.

edited Dec 12 '13 at 13:05
Benjamin
16.1k 30 126 236

answered Jul 7 '13 at 14:29
Poyson1
193 2 12

1 this is working charm – mzonerz Aug 16 '16 at 7:08

The most elegant way is to use .Net's built in methods.

13

These methods:

- Are tried and tested. These methods are used in my own professional projects.
- Use regular expressions internally, which are reliable and fast.
- Made by Microsoft for C#. There's no need to reinvent the wheel.
- Return a bool result. True means the email is valid.

For users of .Net 4.5 and greater

Add this Reference to your project:

> System.ComponentModel.DataAnnotations

Now you can use the following code:

```
(new EmailAddressAttribute().IsValid("youremailhere@test.test"));
```

Example of use

Here are some methods to declare:

```
protected List<string> GetRecipients() // Gets recipients from TextB
`TxtRecipients`
{
    List<string> MethodResult = null;

    try
    {
        List<string> Recipients = TxtRecipients.Text.Replace(",",";"
"").Split(';').ToList();

        List<string> RecipientsCleaned = new List<string>();

        foreach (string Recipient in RecipientsCleaned)
        {
            if (!String.IsNullOrWhiteSpace(Recipient))
            {
                RecipientsNoBlanks.Add(Recipient);

            }

        }

        MethodResult = RecipientsNoBlanks;

    }
    catch//(Exception ex)
    {
        //ex.HandleException();
    }

    return MethodResult;

}


public static bool IsValidEmailAddresses(List<string> recipients)
{
    List<string> InvalidAddresses = GetInvalidEmailAddresses(recipie

    return InvalidAddresses != null && InvalidAddresses.Count == 0;

}

public static List<string> GetInvalidEmailAddresses(List<string> rec
{
    List<string> MethodResult = null;

    try
    {
        List<string> InvalidEmailAddresses = new List<string>();

        foreach (string Recipient in recipients)
        {
            if (!(new EmailAddressAttribute().IsValid(Recipient)) &&
!InvalidEmailAddresses.Contains(Recipient))
            {
                InvalidEmailAddresses.Add(Recipient);

            }

        }

        MethodResult = InvalidEmailAddresses;
    }
    catch//(Exception ex)
    {
        //ex.HandleException();

    }

    return MethodResult;

}
```

...and code demonstrating them in action:

```
    List<string> Recipients = GetRecipients();

    bool IsValidEmailAddresses = IsValidEmailAddresses(Recipients);

    if (IsValidEmailAddresses)
    {
        //Emails are valid. Your code here

    }
    else
    {
        StringBuilder sb = new StringBuilder();

        sb.Append("The following addresses are invalid:");

        List<string> InvalidEmails = GetInvalidEmailAddresses(Recipients

        foreach (string InvalidEmail in InvalidEmails)
        {
            sb.Append("\n" + InvalidEmail);

        }

        MessageBox.Show(sb.ToString());

    }
```

In addition, this example:

- Extends beyond the spec since a single string is used to contain 0, one or many email addresses sperated by a semi-colon `;`.
- Clearly demonstrates how to use the IsValid method of the EmailAddressAttribute object.

Alternative, for users of a version of .Net less than 4.5

For situations where .Net 4.5 is not available, I use the following solution:

Specifically, I use:

```
public static bool IsValidEmailAddress(string emailAddress)
{
    bool MethodResult = false;

    try
    {
        MailAddress m = new MailAddress(emailAddress);

        MethodResult = m.Address == emailAddress;

    }
    catch //(Exception ex)
    {
        //ex.HandleException();

    }

    return MethodResult;

}

public static List<string> GetInvalidEmailAddresses(List<string> rec
{
    List<string> MethodResult = null;

    try
    {
        List<string> InvalidEmailAddresses = new List<string>();

        foreach (string Recipient in recipients)
        {
            if (!IsValidEmail(Recipient) && !InvalidEmailAddresses.C
            {
                InvalidEmailAddresses.Add(Recipient);

            }

        }

        MethodResult = InvalidEmailAddresses;

    }
    catch //(Exception ex)
    {
        //ex.HandleException();

    }

    return MethodResult;

}
```

edited Aug 24 '17 at 12:33

answered Mar 15 '16 at 13:10
Knickerless-Noggins
5,291 3 47 59

1 @ThomasAyoub I haven't seen an EmailAddressAttribute answer, and no answer that uses new MailAddress() with a check on input == .Address so I think its pretty useful. Without reading comments the accepted answer is pretty wrong. new MailAddress("Foobar Some@thing.com") proves that. – Jan Sep 18 '17 at 13:56

7

Short and accurate code

```
public static bool IsValidEmail(this string email)
        {
            const string pattern = @"^(?!\.)(""([^""\r\\]|\\[""\r\\]
9!#$%&'*+/=?^_`{|}~]|(?<!\.)\.)*)(?<!\.)" + @"@[a-z0-9][\w\.-]*[a-z0
[a-z]$";

            var regex = new Regex(pattern, RegexOptions.IgnoreCase);

            return regex.IsMatch(email);
        }
```

answered Jan 27 '18 at 13:33

Naveen Soni
247 4 27

1 Thanks for the solution – Jack Gajanan May 24 '18 at 13:39

6

To be honest, in production code, the best I do is check for an `@` symbol.

I'm never in a place to be completely validating emails. You know how I see if it was really valid? If it got sent. If it didn't, it's bad, if it did, life's good. That's all I need to know.

answered Sep 2 '09 at 2:53

Noon Silk
47.4k 6 77 98

6

I find this regex to be a good trade off between checking for something more than just the @ mark, and accepting weird edge cases:

```
^[^@\s]+@[^@\s]+(\.[^@\s]+)+$
```

It will at least make you put something around the @ mark, and put at least a normal looking domain.

edited Sep 21 '16 at 2:32

answered Sep 2 '09 at 3:38

Matthew Lock
8,052 7 65 109

4

Email address validation is not as easy as it might seem. It's actually theoretically impossible to fully validate an email address using just a regular expression.

Check out my blog post about it for a discussion on the subject and a F# implementation using FParsec. [/shameless_plug]

answered Sep 2 '09 at 1:10

Mauricio Scheffer
87.4k 18 178 266

1 I liked you list of alternative approaches; very interesting. – Luke Quinane Sep 2 '09 at 1:14

1 Interesting article. But seriously who's putting comments, and nested ones at that, in email addresses? – Matthew Lock Sep 2 '09 at 3:44

3 @matthew I don't know why the IETF has even allowed that, but it's *possible*, so a thorough validation has to take it into account. – Mauricio Scheffer Sep 2 '09 at 12:21

Here's my answer -- Phil's solution fails for single letter domains like "someone@q.com". Believe it or not, that's used =) (goes to

4

centurylink, for instance).

Phil's answer is also going to work only with PCRE standard... so C# will take it, but javascript is going to bomb. It's too complex for javascript. So you can't use Phil's solution for mvc validation attributes.

Here's my regex. It'll work nicely with MVC validation attributes.
- Everything before the @ is simplified, so that at least javascript will work. I'm okay relaxing validation here as long as exchange server doesn't give me a 5.1.3. - Everything after the @ is Phil's solution modified for single letter domains.

```
public const string EmailPattern =
        @"^\s*[\w\-\+_']+(\.[\w\-\+_']+)*\@[A-Za-z0-9]([\w\.-]*[A-Za
[A-Za-z\.]*[A-Za-z]$";
```

For people suggesting using system.net.mail MailMessage(), that thing is WAY to flexible. Sure, C# will accept the email, but then exchange server will bomb with 5.1.3 runtime error as soon as you try to send the email.

edited May 13 '14 at 21:05

answered Feb 22 '14 at 1:06

Ralph N
2,237 5 21 34

3

If you really and I mean really want to know if an email address is valid...ask the mail exchanger to prove it, no regex needed. I can provide the code if requested.

General steps are as follows: 1. does email address have a domain name part? (index of @ > 0) 2. using a DNS query ask if domain has a mail exchanger 3. open tcp connection to mail exchanger 4. using the smtp protocol, open a message to the server using the email address as the reciever 5. parse the server's response. 6. quit the message if you made it this far, everything is good.

This is as you can imagine, very expensive time wise and relies on smtp, but it does work.

answered Sep 2 '09 at 4:26
Joe Caffeine
788 4 10

1 That won't work. The smtp protocol to verify an email address has LONG been deprecated/not used. It is considered bad practice to enable that on mail servers since spammers use that functionality. – Chad Grant Sep 9 '13 at 20:15

2

Generally speaking, a regular expression to validate email addresses is not an easy thing to come up with; at the time of this writing, the syntax of an email address must follow a relatively high number of standards and implementing all of them within a regular expression is practically unfeasible!

I highly suggest you to try our EmailVerify.NET, a mature .NET library which can validate email addresses following **all** of the current IETF standards (RFC 1123, RFC 2821, RFC 2822, RFC 3696, RFC 4291, RFC 5321 and RFC 5322), tests the related DNS records, checks if the target mailboxes can accept messages and can even tell if a given address is disposable or not.

Disclaimer: I am the lead developer for this component.

edited Feb 6 '12 at 11:41

answered Sep 22 '11 at 5:59

Efran Cobisi
4,315 13 20

1 @OhadSchneider Yes: we are offering a free version of our email validation technology by way of Verifalia, our SaaS email verification service. Verifalia comes with free and open-source SDKs for the major software development platforms, including .NET. – Efran Cobisi Jul 16 '17 at 14:26

2

Check email string is right format or wrong format by `System.Text.RegularExpressions` :

```
public static bool IsValidEmailId(string InputEmail)
{
    Regex regex = new Regex(@"^([\w\.\-]+)@([\w\-]+)((\.(\w){2,3
    Match match = regex.Match(InputEmail);
    if (match.Success)
        return true;
    else
        return false;
}

protected void Email_TextChanged(object sender, EventArgs e)
{
    String UserEmail = Email.Text;
    if (IsValidEmailId(UserEmail))
    {
        Label4.Text = "This email is correct formate";
    }
    else
    {
        Label4.Text = "This email isn't correct formate";
    }
}
```

answered Jun 12 '14 at 8:41

RKTUXYN

509 6 17

2

```
For the simple email like goerge@xxx.com, below code is sufficient.

 public static bool ValidateEmail(string email)
        {
            System.Text.RegularExpressions.Regex emailRegex = new
System.Text.RegularExpressions.Regex(@"^([\w\.\-]+)@([\w\-]+)((\.(\w
            System.Text.RegularExpressions.Match emailMatch = emailR
            return emailMatch.Success;
        }
```

answered Aug 26 '17 at 17:39

user2211290

651 7 14

2

The most voted answer from @Cogwheel is best answer however i have tried to implement `trim()` string method so it will trim all user white space from string start to end. Check the code bellow for full example-

```
bool IsValidEmail(string email)
{
    try
    {
        email = email.Trim();
        var addr = new System.Net.Mail.MailAddress(email);
        return addr.Address == email;
    }
    catch
    {
        return false;
    }
}
```

answered Nov 6 '18 at 4:52

Liakat Hossain
482 5 14

1

/Using the Internal Regex used in creating the "new EmailAddressAttribute();" component in .Net4.5 >>> using System.ComponentModel.DataAnnotations; //To Validate an Email Address......Tested and Working.

```
public bool IsEmail(string email)
{
    if (String.IsNullOrEmpty(email))
    {   return false;  }
    try
    {
        Regex _regex = new Regex("^((([a-z]|\\d|[!#\\$%&'\\*\\+\\-\\
[\\u00A0-\\uD7FF\\uF900-\\uFDCF\\uFDF0-\\uFFEF])" +
                "+(\\.([a-z]|\\d|[!#\\$%&'\\*\\+\\-\\/=\\?\\^_`{\\|}
\\uD7FF\\uF900-\\uFDCF\\uFDF0-\\uFFEF])+)*)|((\\x22)" +
                "((((\\x20|\\x09)*(\\x0d\\x0a))?(\\x20|\\x09)+)?(([\
\\x08\\x0b\\x0c\\x0e-\\x1f\\x7f]|\\x21|[\\x23-\\x5b]|[\\x5d-\\x7e]|"
                "[\\u00A0-\\uD7FF\\uF900-\\uFDCF\\uFDF0-\\uFFEF])|(\
\\x09\\x0b\\x0c\\x0d-\\x7f]|[\\u00A0-\\uD7FF\\uF900-\\uFDCF\\u" +
                "FDF0-\\uFFEF]))))*(((\\x20|\\x09)*(\\x0d\\x0a))?(\\
(\\x22)))@((([a-z]|\\d|[\\u00A0-\\uD7FF\\uF900-\\uFDCF\\uFDF0-\\uFFE
                "(([a-z]|\\d|[\\u00A0-\\uD7FF\\uF900-\\uFDCF\\uFDF0-
z]|\\d|-|\\.|_|~|[\\u00A0-\\uD7FF\\uF900-\\uFDCF\\uFDF0-\\uFFEF])*([
                "[\\u00A0-\\uD7FF\\uF900-\\uFDCF\\uFDF0-\\uFFEF])))\
[\\u00A0-\\uD7FF\\uF900-\\uFDCF\\uFDF0-\\uFFEF])|(([a-z]|[\\u00A0-\\
                "-\\uFDCF\\uFDF0-\\uFFEF])([a-z]|\\d|-|\\.|_|~|[\\u0
\\uFDCF\\uFDF0-\\uFFEF])*([a-z]|[\\u00A0-\\uD7FF\\uF900-\\uFDCF\\uFD
                "EF])))\\.?$", RegexOptions.IgnoreCase | RegexOption
RegexOptions.Compiled);
        return _regex.IsMatch(email);
    }
    catch (RegexMatchTimeoutException)
    {
        return false;
    }
}
```

Also, You can use this:

http://msdn.microsoft.com/en-us/library/01escwtf(v=vs.110).aspx

edited Oct 22 '14 at 13:57

answered Oct 22 '14 at 13:48

Aina Ademola C
39 4

1 First, the email is valid w.r.t. example@example.co or example@example.com....The email has all valid string and criterial except for the last character " ö " and you can easily add a simple condition to validate such character. Second, am not sure about that mandrill api error, you might want to verify your method of usage bcos I have used this validation on some other environment / api's and its been good to me. – Aina Ademola C May 29 '15 at 13:24

1

Here is an answer to your question for you to check.

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

public class RegexUtilities
{
   public bool IsValidEmail(string strIn)
   {
       if (String.IsNullOrEmpty(strIn))
       {
          return false;
```

```
            }

            // Use IdnMapping class to convert Unicode domain names.

            try
            {
               strIn = Regex.Replace(strIn, @"(@)(.+)$", this.DomainMappe
     RegexOptions.None, TimeSpan.FromMilliseconds(200));

            }
            catch (RegexMatchTimeoutException)
            {
                return false;

            }

            if (invalid)
            {
                return false;

            }

            // Return true if strIn is in valid e-mail format.

            try
            {
               return Regex.IsMatch(strIn, @"^(?("")("".+?(?<!\\)""@)|(([
     [-!#\$%&'\*\+/=\?\^`\{\}\|~\w])*)(?<=[0-9a-z])@))(?(\[)(\[(\d{1,3}\.
     9a-z][-\w]*[0-9a-z]*\.)+[a-z0-9][\-a-z0-9]{0,22}[a-z0-9]))$", RegexO
     TimeSpan.FromMilliseconds(250));

            }
            catch (RegexMatchTimeoutException)
            {
               return false;

            }

        }


        private string DomainMapper(Match match)
        {
           // IdnMapping class with default property values.

           IdnMapping idn = new IdnMapping();
```

```
            string domainName = match.Groups[2].Value;

            try
            {
               domainName = idn.GetAscii(domainName);

            }
            catch (ArgumentException)
            {
               invalid = true;

            }

            return match.Groups[1].Value + domainName;

         }

      }
```

edited Aug 24 '17 at 12:40
Knickerless-Noggins
5,291 3 47 59

answered Jun 22 '16 at 13:12
Parsa Karami
302 1 5 17

1

There are a lot of strong answers here. However, I recommend that we take a step back. @Cogwheel answers the question https://stackoverflow.com/a/1374644/388267. Nevertheless, it could be costly in a bulk validation scenario, if many of the email address being validated are invalid. I suggest that we employ a bit of logic before we enter into his try-catch block. I know that the following code could be written using RegEx but that could be costly for new developers to understand. This is my twopence worth:

```
public static bool IsEmail(this string input)
{
    if (string.IsNullOrWhiteSpace(input)) return false;

    // MUST CONTAIN ONE AND ONLY ONE @
    var atCount = input.Count(c => c == '@');
    if (atCount != 1) return false;

    // MUST CONTAIN PERIOD
    if (!input.Contains(".")) return false;

    // @ MUST OCCUR BEFORE LAST PERIOD
    var indexOfAt = input.IndexOf("@", StringComparison.Ordinal)
    var lastIndexOfPeriod = input.LastIndexOf(".", StringCompari
    var atBeforeLastPeriod = lastIndexOfPeriod > indexOfAt;
    if (!atBeforeLastPeriod) return false;

    // CODE FROM COGWHEEL'S ANSWER: https://stackoverflow.com/a/
    try
    {
        var addr = new System.Net.Mail.MailAddress(input);
        return addr.Address == input;
    }
    catch
    {
        return false;
    }
}
```

answered Oct 7 '18 at 5:31

CarneyCode

2,800 2 21 40

0

```
private static bool IsValidEmail(string emailAddress)
{
    const string validEmailPattern = @"^(?!\.)(""([^""\r\\]|\\[""\r\
                                     + @"([-a-z0-9!#$%&'*+/=?^_`{|}~
<!\.)"
                                     + @"@[a-z0-9][\w\.-]*[a-z0-9]\.
z]$";
```

```
    return new Regex(validEmailPattern, RegexOptions.IgnoreCase).IsMa
}
```

answered Mar 17 '14 at 21:13

ErwanLent
448 6 12

0

I succinctified Poyson 1's answer like so:

```
public static bool IsValidEmailAddress(string candidateEmailAddr)
{
    string regexExpresion = "\\w+([-+.']\\w+)*@\\w+([-.]\\w+)*\\.\\w
    return (Regex.IsMatch(candidateEmailAddr, regexExpresion)) &&
           (Regex.Replace(candidateEmailAddr, regexExpresion, string
0);
}
```

answered Feb 12 '16 at 17:23

B. Clay Shannon
1,146 93 319 620

0

I wrote an function to check if an email is valid or not. It seems working well for me in most cases.

Results:

```
dasddas-@.com => FALSE
-asd@das.com => FALSE
as3d@dac.coas- => FALSE
dsq!a?@das.com => FALSE
_dasd@sd.com => FALSE
dad@sds => FALSE
```

```
asd-@asd.com => FALSE
dasd_-@jdas.com => FALSE
asd@dasd@asd.cm => FALSE
da23@das..com => FALSE
_dasd_das_@9.com => FALSE

d23d@da9.co9 => TRUE
dasd.dadas@dasd.com => TRUE
dda_das@das-dasd.com => TRUE
dasd-dasd@das.com.das => TRUE
```

Code:

```
    private bool IsValidEmail(string email)
    {
        bool valid = false;
        try
        {
            var addr = new System.Net.Mail.MailAddress(email);
            valid = true;
        }
        catch
        {
            valid = false;
            goto End_Func;
        }

        valid = false;
        int pos_at = email.IndexOf('@');
        char checker = Convert.ToChar(email.Substring(pos_at + 1, 1)
        var chars = "qwertyuiopasdfghjklzxcvbnm0123456789";
        foreach (char chr in chars)
        {
            if (checker == chr)
            {
                valid = true;
                break;
            }
        }
        if (valid == false)
        {
            goto End_Func;
        }

        int pos_dot = email.IndexOf('.', pos_at + 1);
        if(pos_dot == -1)
```

```
{
    valid = false;
    goto End_Func;
}

valid = false;
try
{
    checker = Convert.ToChar(email.Substring(pos_dot + 1, 1)
    foreach (char chr in chars)
    {
        if (checker == chr)
        {
            valid = true;
            break;
        }
    }
}
catch
{
    valid = false;
    goto End_Func;
}

Regex valid_checker = new Regex(@"^[a-zA-Z0-9_@.-]*$");
valid = valid_checker.IsMatch(email);
if (valid == false)
{
    goto End_Func;
}

List<int> pos_list = new List<int> { };
int pos = 0;
while (email.IndexOf('_', pos) != -1)
{
    pos_list.Add(email.IndexOf('_', pos));
    pos = email.IndexOf('_', pos) + 1;
}

pos = 0;
while (email.IndexOf('.', pos) != -1)
{
    pos_list.Add(email.IndexOf('.', pos));
    pos = email.IndexOf('.', pos) + 1;
}

pos = 0;
```

```
    while (email.IndexOf('-', pos) != -1)
    {
        pos_list.Add(email.IndexOf('-', pos));
        pos = email.IndexOf('-', pos) + 1;
    }

    int sp_cnt = pos_list.Count();
    pos_list.Sort();
    for (int i = 0; i < sp_cnt - 1; i++)
    {
        if (pos_list[i] + 1 == pos_list[i + 1])
        {
            valid = false;
            break;
        }

        if (pos_list[i]+1 == pos_at || pos_list[i]+1 == pos_dot)
        {
            valid = false;
            break;
        }
    }

    if(valid == false)
    {
        goto End_Func;
    }

    if (pos_list[sp_cnt - 1] == email.Length - 1 || pos_list[0]
    {
        valid = false;
    }

End_Func:;
    return valid;
}
```

answered Jun 29 '16 at 15:44

Louis Tran
503 4 22

0

Simple way to identify the emailid is valid or not.

```
public static bool EmailIsValid(string email)
{
        return Regex.IsMatch(email, @"^([\w-\.]+)@((\[[0-9]{1,3}\.[0
{1,3}\.)|(([\w-]+\.)+))([a-zA-Z]{2,4}|[0-9]{1,3})(\]?)$");
}
```

answered Aug 22 '16 at 13:25

Amit Gorvadiya
11 3

0

There is culture problem in regex in C# rather then js. So we need to use regex in US mode for email check. If you don't use ECMAScript mode, your language special characters are imply in A-Z with regex.

```
Regex.IsMatch(email, @"^([a-zA-Z0-9_\-\.]+)@((\[[0-9]{1,3}\.[0-9]{1,
(([a-zA-Z0-9_\-]+\.)+))([a-zA-Z]{2,4}|[0-9]{1,3})(\]?)$", RegexOptio
```

answered Sep 28 '16 at 7:50

mkysoft
2,218 1 12 20

0

I ended up using this regex, as it successfully validates commas, comments, Unicode characters and IP(v4) domain addresses.

Valid addresses will be:

> " "@example.org
>
> (comment)test@example.org

> тест@example.org
>
> ტესტი@example.org
>
> test@[192.168.1.1]

```
 public const string REGEX_EMAIL = @"^(((\([\w!#$%&'*+\/=?^_`{|}~-]*
[\]\\.,;:\s@\""]+(\.[^<>()[\]\\.,;:\s@\""]+)*)|(\"".+\""))(\([\w!#$%
^_`{|}~-]*\))?@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])
9]+\.)+[a-zA-Z]{2,}))$";
```

answered Nov 22 '16 at 9:32

d.popov
3,070 1 25 39

0

In case you are using FluentValidation you could write something as simple as this:

```
public cass User
{
    public string Email { get; set; }
}

public class UserValidator : AbstractValidator<User>
{
    public UserValidator()
    {
        RuleFor(x => x.Email).EmailAddress().WithMessage("The text e
valid email address.");
    }
}

// Validates an user.
var validationResult = new UserValidator().Validate(new User { Email

// This will return false, since the user email is not valid.
bool userIsValid = validationResult.IsValid;
```

answered Apr 25 '18 at 13:00

Ulysses Alves
1,322 1 11 25

0

Based on the answer of @Cogwheel i want to share a modified solution that works for SSIS and the "Script Component":

1. Place the "Script Component" into your Data Flow connect and then open it.
2. In the section "Input Columns" set the field that contains the E-Mail Adresses to "ReadWrite" (in the example 'fieldName').
3. Switch back to the section "Script" and click on "Edit Script". Then you need to wait after the code opens.
4. Place this code in the right method:

```
public override void Input0_ProcessInputRow(Input0Buffer Row)
{
    string email = Row.fieldName;

    try
    {
        System.Net.Mail.MailAddress addr = new System.Net.Mail.M
        Row.fieldName= addr.Address.ToString();
    }
    catch
    {
        Row.fieldName = "WRONGADDRESS";
    }
}
```

Then you can use a Conditional Split to filter out all invalid records or whatever you want to do.

edited Jun 20 '18 at 6:33

answered Jun 20 '18 at 6:21

user3772108

422 1 6 19

0

a little modification to @Cogwheel answer

```
public static bool IsValidEmail(this string email)
{
  // skip the exception & return early if possible
  if (email.IndexOf("@") <= 0) return false;

  try
  {
    var address = new MailAddress(email);
    return address.Address == email;
  }
  catch
  {
    return false;
  }
}
```

edited Jul 21 '18 at 14:44

answered Jul 20 '18 at 20:36

waitforit

537 2 8 19

1 2 next

protected by Community ♦ Sep 28 '16 at 7:45

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?