

Forum:[Hack Forums](#)

Thread: [Splitting a list of email and hashes with no spaces](#)

webwarlock72 - January 22, 2018, 8:12 pm

Hi everyone

Im returning to a project after nearly 6 months away from it and what Im trying to do is split a list of emails and hashes with no spaces into a workable list with the email and hash on seperate lines. The hashes are SHA1 which contain 40 characters. What I have been attempting is to split the list using the delimiter then counting the first 40 characters then split it.

for example the list is

name@email.com:3fdbedbb509c3173cfe5b00c9c5edec23b1d9ac4name@email.com:e58aa6ec8db9543f3edb08cbab97299062f3f517

and i want to split the list into a listbox as

name@email.com:3fdbedbb509c3173cfe5b00c9c5edec23b1d9ac4
name@email.com:e58aa6ec8db9543f3edb08cbab97299062f3f517

I didnt think Id have a problem with this but Ive been struggling with it. Any help pleas

Thanks

Ninja - January 23, 2018, 12:20 am

If the hash is of a fixed length then you can you split at the index of the first ':' plus N characters.

webwarlock72 - January 23, 2018, 9:22 am

(Jan 23, 2018 01:20 AM)Ninja Wrote: [\[CITE\]](#) If the hash is of a fixed length then you can you split at the index of the first ':' plus N characters.

Thanks Ninja for your reply. This is what ive tried but still showing errors

```
Code Dim itemCount As Integer = RawHasLB.Items.Count
      If itemCount = 0 Then
          ErrorListFRM.Label1.Text = "You must first load a list."
          ErrorListFRM.Show()
      Else
          Dim taskA = Task.Factory.StartNew(Sub() DoSomeWork(10000000))
          taskA.Wait()
      End If
      HashLB.Items.Clear()
      HashemailLB.Items.Clear()

      Dim i As Integer
      For i = 0 To RawHasLB.Items.Count - 1
          Dim arrWords() As String = RawHasLB.Items(i).Split(":", 40)
          HashLB.Items.Add(arrWords(0))
          HashemailLB.Items.Add(arrWords(1))
      Next i
```

Next

Ninja - January 23, 2018, 10:02 am

(Jan 23, 2018 10:22 AM)webwarlock72 Wrote: [\[CITE\]](#) Thanks Ninja for your reply. This is what ive tried but still showing errors

```
Code Dim itemCount As Integer = RawHasLB.Items.Count
    If itemCount = 0 Then
        ErrorListFRM.Label1.Text = "You must first load a list."
        ErrorListFRM.Show()
    Else
        Dim taskA = Task.Factory.StartNew(Sub() DoSomeWork(10000000))
        taskA.Wait()
    End If
    HashLB.Items.Clear()
    HashemailLB.Items.Clear()

    Dim i As Integer
    For i = 0 To RawHasLB.Items.Count - 1
        Dim arrWords() As String = RawHasLB.Items(i).Split(":", 40)
        HashLB.Items.Add(arrWords(0))
        HashemailLB.Items.Add(arrWords(1))
    Next
```

That is not how the string.Split method works. Also, wtf is that task for?

Are you sure that there isn't a problem with the newline? I know Windows uses a carrier return followed by a line feed, while Linux uses only a line feed. Maybe that causes your text editor to display the text file in a weird way.

zonne - January 23, 2018, 10:19 am

Use Regex for this and as Ninja said, hash has same length so regex if perfect for this.

Ninja - January 23, 2018, 10:21 am

```
Code const string input =
"name@email.com:3fdbedbb509c3173cfe5b00c9c5edec23b1d9ac4name2@email.com:e58aa6ec8
db9543f3edb08cbab97299062f3f517";
int colonIndex = 0, comboBegin = 0;
IDictionary<string, string> combos = new Dictionary<string, string>();
while ((colonIndex = input.IndexOf(':', colonIndex + 1)) != -1)
{
    string email = input.Substring(comboBegin, colonIndex - comboBegin);
    string hash = input.Substring(colonIndex + 1, 40);
    combos.Add(email, hash);

    comboBegin += colonIndex + 41;
}
```

Here is some quick code I threw together. Seemed to work with your example. The match fucks my brain so I am pretty sure this can be improved.

(Jan 23, 2018 11:19 AM)zonne Wrote: [\[CITE\]](#) Use Regex for this and as Ninja said, hash has same length so regex if perfect for this.

Isn't regex a bit overkill? You can perfectly fine parse each combo as the hash is always 40 characters, prefixed by a colon.

EdwardSn0wdent - January 27, 2018, 5:12 pm

I have worked with regular expressions daily for 12 years. I would say at least Two hours a day, out of them 12 years I have to write patterns disregarding anything to do with the data given to me.

I witness Once a week on various forums terms like "Isn't regex a bit overkill?", "Regex cannot parse HTML." the latter example is not understood by most.

I get paid to write patterns, without sounding bigheaded because I guess you have your own fields you work with that you would look at and say "I'm an expert". I cannot say I have seen any of your work but with a good rating, I guess you do something right here.

1)How can regex be overkill for pattern matching? it's not like 1980's (for me). No One can comment on current machine speed. If a user is happy using regex, then why not? I use regex for everything. Even down to a simple word match. That's my right being a programmer. I use the tools I am happy with regardless of simplicity.

2) Parsing HTML. That famous post everyone sees constantly. Even more so people repeating it while not fully understanding.

Point One: Regex cannot parse HTML.(did I just not say that post was misunderstood?)

Point Two: HTML is not regular.

Point Three: The key factor others do not get hence why they make remarks. But we can match against HTML

"A pattern may match today, but what about tomorrow?"

That above is what users do not get. This is why HTML can't be parsed because it's not regular. But it can be parsed. A regular pattern should be "as is" Is this making sense?

If One of you wanted to ask me what to choose. As a professional, I would have to suggest a built already parser. It's my choice to choose regex.

To finalize, Regex is an insufficiently sophisticated tool.

First post, hope to get to know some of you in the future.

Ninja - January 27, 2018, 5:19 pm

(Jan 27, 2018 06:12 PM)EdwardSn0wdent Wrote: [\[CITE\]](#) I have worked with regular expressions daily for 12 years. I would say at least Two hours a day, out of them 12 years I have to write patterns disregarding anything to do with the data given to me.

I witness Once a week on various forums terms like "Isn't regex a bit overkill?", "Regex cannot parse HTML." the latter example is not understood by most.

I get paid to write patterns, without sounding bigheaded because I guess you have your own fields you work with that you would look at and say "I'm an expert". I cannot say I have seen any of your work but with a good rating, I guess you do something right here.

1)How can regex be overkill for pattern matching? it's not like 1980's (for me). No One can comment on current machine speed. If a user is happy using regex, then why not? I use regex for everything. Even down to a simple word match. That's my right being a programmer. I use the tools I am happy with regardless of simplicity.

2) Parsing HTML. That famous post everyone sees constantly. Even more so people repeating it while not fully understanding.

Point One: Regex cannot parse HTML.(did I just not say that post was misunderstood?)

Point Two: HTML is not regular.

Point Three: The key factor others do not get hence why they make remarks. But we can match against HTML

"A pattern may match today, but what about tomorrow?"

That above is what users do not get. This is why HTML can't be parsed because it's not regular. But it can be parsed. A regular pattern should be "as is" Is this making sense?

If One of you wanted to ask me what to choose. As a professional, i would have to suggest a built already parser. It's my choice to choose regex.

To finalize, Regex is an insufficiently sophisticated tool.

First post, hope to get to know some of you in the future.

I found your reply to be incoherent and I don't see a clear point made. Are you saying Regex is fine for this and for HTML parsing? Please give us an example where Regex outperforms a HTML parser like HtmlAgilityPack and Anglesharp. HTML parsing already wins in the ease of use part, most of the time.

EdwardSn0wdent - January 27, 2018, 6:55 pm

Please, can you point out where i referenced any speed/performance issues? Are you actually a programmer? I see 4 years, so you are anew programmer correct? You don't understand my post because you do not understand regex. That's fine, nothing be ashamed of.

I have parsed data daily for 8 years longer than you have even coded. I don't get it, are you too much above members here you must always want to be correct?

Within the hour, I have joined it's clear you have quite a bad attitude, not that really educated.

Take care.

Ninja - January 27, 2018, 7:28 pm

(Jan 27, 2018 07:55 PM)EdwardSn0wdent Wrote: [\[CITE\]](#) Please, can you point out where i referenced any speed/performance issues?

There are 2 things that pop into my mind when deciding which approach to use for parsing: speed and ease of use. Ease of use is already there with HTML parsing libraries like Anglesharp, so speed is left, hence I asked you to prove the performance is better with Regex. And your initial reply was so incoherent and badly worded I had no idea what you actually meant. If regex isn't faster, and not easier to use than HTML parsing, then why post here defending Regex at all?

The reason we talk Regex down is because people that are new to parsing data are immediately introduced to Regex, no matter if it is JSON or HTML or XML, people always suggest Regex. People don't know how or why, they just use it. Just because it is easier doesn't always mean it is better, and as a programmer you should share that opinion. For HTML you should use a DOM parser. For XML you should use an XML deserializer. And for JSON you should use a JSON deserializer. For raw text you can use Regex or other algorithms like Boyer-Moore.

And sure, if a user needs just 1 string parsed from a document then Regex can be better. But in OP's case it can be solved with simple substringing, as you can always determine the start of the next line by counting the amount of characters from a certain index. OP obviously has a leaked database of some sorts, and it is safe to assume it is big in size as most leaked databases worth something are, so the performance difference with Regex vs substring can definitely be a factor here.

SL x TnT - January 28, 2018, 3:58 am

```
(Jan 23, 2018 11:21 AM)Ninja Wrote: \[CITE\] Code const string input =  
"name@email.com:3fdbedbb509c3173cfe5b00c9c5edec23b1d9ac4name2@email.com:e58aa6ec8  
db9543f3edb08cbab97299062f3f517";  
int colonIndex = 0, comboBegin = 0;  
IDictionary<string, string> combos = new Dictionary<string, string>();  
while ((colonIndex = input.IndexOf(':', colonIndex + 1)) != -1)  
{  
    string email = input.Substring(comboBegin, colonIndex - comboBegin);  
    string hash = input.Substring(colonIndex + 1, 40);  
    combos.Add(email, hash);  
  
    comboBegin += colonIndex + 41;  
}
```

Here is some quick code I threw together. Seemed to work with your example. The match fucks my brain so I am pretty sure this can be improved.

Isn't regex a bit overkill? You can perfectly fine parse each combo as the hash is always 40 characters, prefixed by a colon.

Code `comboBegin = colonIndex + 41;`

Yours will fail after the 2nd one.

Substring vs Regex. To me, Regex was faster for me to write. If I needed speed, I'd go with Substring. It should be a personal preference which is used, but a programmer should be able to do both.

Ninja - January 28, 2018, 7:55 am

(Jan 28, 2018 04:58 AM)SL x TnT Wrote: [\[CITE\]](#) Yours will fail after the 2nd one.

Why is that? I just re-installed my PC so I don't have Visual Studio so I can't test.

EdwardSn0wdent - January 28, 2018, 9:09 am

That's not strictly true. A stream for example only cares about LF. Notepad, for example, will add the CR making it seem like Classic VB CRLFs. The only thing that should be used in Visual Basic is `ControlChars.NewLine` or `Environment.NewLine`.

Compare `ControlChars.NewLine` and `Environment.NewLine`, for example. The difference is that one is a constant and the other is not. `Environment.NewLine` returns the line break sequence for the current environment. That means that under Mono on Linux(etc)it will return just a line feed, while on Windows it will return a carriage return and line feed. `ControlChars.NewLine` is defined as a carriage return and line feed so it will not change regardless of the platform.

Ninja could not follow what I was saying so see's fit when he/she does not understand to -rep a person.

I know i have only been a member for a few hours but why are you posting c# ina vb section? It has

it's own section yes.

Code const string input =

```
"name@email.com:3fdbedbb509c3173cfe5b00c9c5edec23b1d9ac4name2@email.com:e58aa6ec8db9543f3edb08cbab97299062f3f517";
```

A constant is something that does not change. Arguably a string is never a good constant. The whole just reinstalled machine seems like a get out if a user makes a mistake. VS installed or not. That has no business being a constant.

SL x TnT - January 28, 2018, 9:26 am

(Jan 28, 2018 08:55 AM)Ninja Wrote: [\[CITE\]](#) Why is that? I just re-installed my PC so I don't have Visual Studio so I can't test.

```
comboBegin += colonIndex + 41;
```

colonIndex is your current colon position. You always want 41 after that position.

If you're adding to comboBegin, you're adding all previous colonIndex (+41) on top of it. After the 2nd hash, it'll throw an exception

Ninja - January 28, 2018, 11:30 am

(Jan 28, 2018 10:26 AM)SL x TnT Wrote: [\[CITE\]](#) comboBegin += colonIndex + 41;

colonIndex is your current colon position. You always want 41 after that position.

If you're adding to comboBegin, you're adding all previous colonIndex (+41) on top of it. After the 2nd hash, it'll throw an exception

Hmm I don't get what you mean. The hash is always 40 characters long, and so the next combo starts 41 characters after the last colonIndex + 41. Then again, I almost broke my head over this third grade level math, so you are probably correct. I will re-install Visual Studio and check.

EDIT: Gotcha, it should be "comboBegin = colonIndex + 41;" instead. Thanks for the heads up.