Can I convert a C# string value to an escaped string literal

Ask Question

171

47

In C#, can I convert a string value to a string literal, the way I would see it in code? I would like to replace tabs, newlines, etc. with their escape sequences.

If this code:

```
Console.WriteLine(someString);
```

produces:

```
Hello
World!
```

I want this code:

```
Console.WriteLine(ToLiteral(someString));
```

to produce:

```
\tHello\r\n\tWorld!\r\n
```

c# string escaping

edited Jan 15 '10 at 14:34
Groo
35.8k 14 88 159

asked Nov 27 '08 at 12:39
Hallgrim
11.3k 9 38 51

15 Answers

158

I found this:

```
private static string ToLiteral(string input)
{
    using (var writer = new StringWriter())
    {
        using (var provider = CodeDomProvider.CreateProvider("CSharp
        {
            provider.GenerateCodeFromExpression(new CodePrimitiveExp
writer, null);
            return writer.ToString();
        }
    }
}
```

This code:

```
var input = "\tHello\r\n\tWorld!";
Console.WriteLine(input);
Console.WriteLine(ToLiteral(input));
```

Produces:

```
    Hello
    World!
"\tHello\r\n\tWorld!"
```

edited Jul 25 '12 at 5:30
John Gietzen
38.7k 26 128 181

answered Nov 27 '08 at 23:40
Hallgrim
11.3k 9 38 51

1 Just found this from google the subject. This has to be best, no point in reinventing stuff that .net can do for us – Andy Morris Jan 19 '10 at 13:58

12 Nice one, but be aware that for longer strings, this will insert "+" operators, newlines and indentation. I couldn't find a way to turn that off. – Timwi May 4 '10 at 21:49

1 What about the inverse ? If you have a file with text containg escape sequences incluidng especial character escaped with its ascii code ? How to produce a raw version ? – Luciano Nov 29 '12 at 16:57

1 If you run: void Main() { Console.WriteLine(ToLiteral("test \"\'\\\0\a\b\f\n\r\t\v\uaaaa \\\blah")); } you'll notice that this doesn't take care of a few escapes. Ronnie Overby pointed \f, the others are \a and \b – costa Feb 1 '13 at 21:34

3 Is there a way to make it output verbatim (`@"..."`) literals? – rookie1024 Mar 27 '16 at 20:35

26

What about Regex.Escape(String) ?

> Regex.Escape escapes a minimal set of characters (\, *, +, ?, |, {, [, (,), ^, $,., #, and white space) by replacing them with their escape codes.

edited Aug 22 '14 at 15:31
HugoRune
8,238 5 44 111

answered Jan 16 '13 at 11:31
Shqdooow
453 4 3

5 +1 no idea why this is way below. Other answers are just too verbose and look like reinventing wheels – Adrian Carneiro Jul 10 '14 at 22:38

25 This is not what OP is asking for. It doesn't return a string literal, it returns a string with Regex special characters escaped. This would turn `Hello World?` into `Hello World\?` , but that is an invalid string literal. – atheaos May 22 '15 at 20:00

1 I agree with @atheaos, this is a great answer to a very different question. – hypehuman Jul 31 '15 at 20:58

5 +1 even though it doesn't quite answer the OP's question it was what I (and so I suspect maybe others) were looking for when I came across this question. :) – GazB Jun 8 '16 at 15:29

23

EDIT: A more structured approach, including all escape sequences for `string` s and `char` s.
Doesn't replace unicode characters with their literal equivalent.
Doesn't cook eggs, either.

```
public class ReplaceString
{
    static readonly IDictionary<string, string> m_replaceDict
        = new Dictionary<string, string>();

    const string ms_regexEscapes = @"[\a\b\f\n\r\t\v\\""]";

    public static string StringLiteral(string i_string)
    {
        return Regex.Replace(i_string, ms_regexEscapes, match);
    }

    public static string CharLiteral(char c)
    {
        return c == '\'' ? @"'\''" : string.Format("'{0}'", c);
    }

    private static string match(Match m)
    {
        string match = m.ToString();
        if (m_replaceDict.ContainsKey(match))
        {
            return m_replaceDict[match];
        }

        throw new NotSupportedException();
    }

    static ReplaceString()
    {
        m_replaceDict.Add("\a", @"\a");
        m_replaceDict.Add("\b", @"\b");
        m_replaceDict.Add("\f", @"\f");
        m_replaceDict.Add("\n", @"\n");
        m_replaceDict.Add("\r", @"\r");
        m_replaceDict.Add("\t", @"\t");
        m_replaceDict.Add("\v", @"\v");

        m_replaceDict.Add("\\", @"\\");
        m_replaceDict.Add("\0", @"\0");

        //The SO parser gets fooled by the verbatim version
        //of the string to replace - @"\"""
        //so use the 'regular' version
        m_replaceDict.Add("\"", "\\\"");
    }

    static void Main(string[] args){

        string s = "here's a \"\n\tstring\" to test";
        Console.WriteLine(ReplaceString.StringLiteral(s));
        Console.WriteLine(ReplaceString.CharLiteral('c'));
        Console.WriteLine(ReplaceString.CharLiteral('\''));

    }
}
```

edited Nov 27 '08 at 15:10

answered Nov 27 '08 at 12:49

Cristi Diaconescu
16.4k 22 114 198

This is not all escape sequences ;) – TcKs Nov 27 '08 at 12:51

2 It's a good starting point, though. – Dave Van den Eynde Nov 27 '08 at 13:03

1 Works better than the solution above - and other escape sequences can easily be added. – Volkirith Aug 10 '14 at 11:15

Verbatim in the accepted answer was driving me bonkers. This works

100% for my purpose. Replaced regex with `@"[\a\b\f\n\r\t\v\\""/]"` and added `m_replaceDict.Add("/", @"\/");` for `JSON`. – GibralterTop Jun 29 '17 at 17:12

Also, you have to add the enclosing quotations to this if you want those. – GibralterTop Jun 29 '17 at 20:38

17

```
public static class StringHelpers
{
    private static Dictionary<string, string> escapeMapping = new Dic
string>()
    {
        {"\"", @"\\\""},
        {"\\\\", @"\\"},
        {"\a", @"\a"},
        {"\b", @"\b"},
        {"\f", @"\f"},
        {"\n", @"\n"},
        {"\r", @"\r"},
        {"\t", @"\t"},
        {"\v", @"\v"},
        {"\0", @"\0"},
    };

    private static Regex escapeRegex = new Regex(string.Join("|",
escapeMapping.Keys.ToArray()));

    public static string Escape(this string s)
    {
        return escapeRegex.Replace(s, EscapeMatchEval);
    }

    private static string EscapeMatchEval(Match m)
    {
        if (escapeMapping.ContainsKey(m.Value))
        {
            return escapeMapping[m.Value];
        }
        return escapeMapping[Regex.Escape(m.Value)];
    }
}
```

edited May 17 '16 at 9:19

William Jockusch
9,018 42 155 268

answered Nov 27 '08 at 16:00

ICR
10.7k 3 40 71

1 Why is there 3 backslashes and two speech marks in the first value of the dictionary? – James Yeoman Mar 30 '17 at 9:19

Nice answer, @JamesYeoman that's because regex pattern needs to be escaped. – Ali Mousavi Kherad Aug 13 '18 at 22:53

14

try:

```
var t = HttpUtility.JavaScriptStringEncode(s);
```

answered Mar 2 '12 at 11:04

Arsen Zahray
9,820 35 104 180

Does not work. If I have "abc\n123" (without quotes, 8 chars), I want "abc" + \n + "123" (7 chars). Instead it produces "abc" + "\\" + "\n123" (9 chars). Notice the slash was doubled and it still contains a string literal of "\n" as two characters, not the escaped character. – Paul Mar 7 '12 at 20:13

1 @Paul What you want is the opposite of what the question is asking, though. This, according to your description, answers the question, and therefore *does* work. – Nic Hartley Jan 4 '17 at 20:19

I found this useful to escape active directory names in the frontend – chakeda Oct 17 '17 at 17:37

12

Hallgrim's answer is excellent, but the "+", newline and indent additions were breaking functionality for me. An easy way around it is:

```
private static string ToLiteral(string input)
{
    using (var writer = new StringWriter())
    {
        using (var provider = CodeDomProvider.CreateProvider("CSharp"
        {
            provider.GenerateCodeFromExpression(new CodePrimitiveExpr
writer, new CodeGeneratorOptions {IndentString = "\t"});
            var literal = writer.ToString();
            literal = literal.Replace(string.Format("\" +{0}\t\"", En
"");
            return literal;
        }
    }
}
```

answered Feb 6 '13 at 0:41

lesur
161 1 5

Works great. I also added one line before the `return literal` to make it more readable: `literal = literal.Replace("\\r\\n", "\\r\\n\"+\r\n\"");` – Bob May 8 '13 at 17:54

Added this `literal = literal.Replace("/", @"\/");` for `JSON` functionality. – GibralterTop Jun 29 '17 at 15:32

This is 100% straight forward and the only correct answer! All other answers either didn't understand the question or re-invented the wheel. – bytecode77 Dec 27 '17 at 13:47

Sad, cannot get this to work under DOTNET CORE. Anyone has a better answer? – s k Feb 6 '18 at 8:33

11

Fully working implementation, including escaping of Unicode and ASCII non printable characters. Does not insert "+" signs like Hallgrim's answer.

```
static string ToLiteral(string input) {
    StringBuilder literal = new StringBuilder(input.Length + 2);
    literal.Append("\"");
    foreach (var c in input) {
        switch (c) {
            case '\'': literal.Append(@"\'"); break;
            case '\"': literal.Append("\\\""); break;
            case '\\': literal.Append(@"\\"); break;
            case '\0': literal.Append(@"\0"); break;
            case '\a': literal.Append(@"\a"); break;
            case '\b': literal.Append(@"\b"); break;
            case '\f': literal.Append(@"\f"); break;
            case '\n': literal.Append(@"\n"); break;
            case '\r': literal.Append(@"\r"); break;
            case '\t': literal.Append(@"\t"); break;
            case '\v': literal.Append(@"\v"); break;
            default:
                // ASCII printable character
                if (c >= 0x20 && c <= 0x7e) {
                    literal.Append(c);
                // As UTF16 escaped character
                } else {
                    literal.Append(@"\u");
                    literal.Append(((int)c).ToString("x4"));
                }
                break;
        }
    }
    literal.Append("\"");
    return literal.ToString();
}
```

edited May 23 '17 at 12:18
Community ♦
1 1

answered Dec 30 '12 at 2:18
Smilediver
1,245 15 23

2 You should use `Char.GetUnicodeCategory(c) == UnicodeCategory.Control` to decide whether to escape it, or people who don't speak ASCII won't be very happy. – deerchao Jan 24 '13 at 13:15

This depends on situation if your resulting string will be used in the environment supporting unicode or not. – Smilediver Jan 29 '13 at 13:59

I added `input = input ?? string.Empty;` as the first line of the method so I could pass `null` and get back `""` instead of a null reference exception. – Andy Jan 8 '17 at 19:23

8

Interesting question.

If you can't find a better method, you can always replace.
In case you're opting for it, you could use this **C# Escape Sequence List**:

- \' - single quote, needed for character literals
- \" - double quote, needed for string literals
- \ - backslash
- \0 - Unicode character 0
- \a - Alert (character 7)
- \b - Backspace (character 8)
- \f - Form feed (character 12)
- \n - New line (character 10)
- \r - Carriage return (character 13)
- \t - Horizontal tab (character 9)
- \v - Vertical quote (character 11)
- \uxxxx - Unicode escape sequence for character with hex value xxxx

- \xn[n][n][n] - Unicode escape sequence for character with hex value nnnn (variable length version of \uxxxx)
- \Uxxxxxxxx - Unicode escape sequence for character with hex value xxxxxxxx (for generating surrogates)

This list can be found in the C# Frequently Asked Questions What character escape sequences are available?

edited Jan 23 at 10:35
Jamie Twells
676 2 11 32

answered Nov 27 '08 at 12:48
Nelson Reis
3,452 9 39 58

2 This link no longer works, a textbook example of why link-only answers are discouraged. – James Apr 4 '17 at 12:44

Very true, @James, but thanks to Jamie Twells the information is available again :+1: – Nelson Reis Jan 23 at 10:50

4

Here is a little improvement for Smilediver's answer, it will not escape all no-ASCII chars but only these are really needed.

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

public static class CodeHelper
{
    public static string ToLiteral(this string input)
    {
        var literal = new StringBuilder(input.Length + 2);
        literal.Append("\"");
        foreach (var c in input)
        {
            switch (c)
```

```
                {
                    case '\'': literal.Append(@"\'"); break;
                    case '\"': literal.Append("\\\""); break;
                    case '\\': literal.Append(@"\\"); break;
                    case '\0': literal.Append(@"\0"); break;
                    case '\a': literal.Append(@"\a"); break;
                    case '\b': literal.Append(@"\b"); break;
                    case '\f': literal.Append(@"\f"); break;
                    case '\n': literal.Append(@"\n"); break;
                    case '\r': literal.Append(@"\r"); break;
                    case '\t': literal.Append(@"\t"); break;
                    case '\v': literal.Append(@"\v"); break;
                    default:
                        if (Char.GetUnicodeCategory(c) != UnicodeCategory
                        {
                            literal.Append(c);
                        }
                        else
                        {
                            literal.Append(@"\u");
                            literal.Append(((ushort)c).ToString("x4"));
                        }
                        break;
                }
            }
            literal.Append("\"");
            return literal.ToString();
        }
    }
```

edited Oct 13 '13 at 16:42

answered Jan 24 '13 at 13:17

deerchao

8,435 5 44 58

```
public static class StringEscape
{
  static char[] toEscape =
"\0\x1\x2\x3\x4\x5\x6\a\b\t\n\v\f\r\xe\xf\x10\x11\x12\x13\x14\x15\x16

  static string[] literals =
@"\0,\x0001,\x0002,\x0003,\x0004,\x0005,\x0006,\a,\b,\t,\n,\v,\f,\r,\
 char[] { ',' });

  public static string Escape(this string input)
  {
    int i = input.IndexOfAny(toEscape);
    if (i < 0) return input;

    var sb = new System.Text.StringBuilder(input.Length + 5);
    int j = 0;
    do
    {
      sb.Append(input, j, i - j);
      var c = input[i];
      if (c < 0x20) sb.Append(literals[c]); else sb.Append(@"\").Appe
    } while ((i = input.IndexOfAny(toEscape, j = ++i)) > 0);

    return sb.Append(input, j, input.Length - j).ToString();
  }
}
```

1

edited Feb 3 '17 at 2:17

answered Jan 30 '17 at 11:10

Serge N
56 6

My attempt at adding ToVerbatim to **Hallgrim's** accepted answer above:

1

```
private static string ToLiteral(string input)
{
    using (var writer = new StringWriter())
```

```
        {
            using (var provider = CodeDomProvider.CreateProvider("CSharp"
            {
                provider.GenerateCodeFromExpression(new CodePrimitiveExpr
writer, new CodeGeneratorOptions { IndentString = "\t" });
                var literal = writer.ToString();
                literal = literal.Replace(string.Format("\" +{0}\t\"", En
"");
                return literal;
            }
        }
    }

private static string ToVerbatim( string input )
{
    string literal = ToLiteral( input );
    string verbatim = "@" + literal.Replace( @"\r\n", Environment.New
    return verbatim;
}
```

answered Nov 2 '17 at 11:15

Derek
5,250 1 23 40

0

If JSON conventions are enough for the unescaped strings you want to get escaped and you already use `Newtonsoft.Json` in your project (it has a pretty large overhead) you may use this package like the following:

```
using System;
using Newtonsoft.Json;

public class Program
{
    public static void Main()
    {
    Console.WriteLine(ToLiteral( @"abc\n123") );
    }

    private static string ToLiteral(string input){
        return JsonConvert.DeserializeObject<string>("\"" + input + "
    }
}
```

answered Feb 4 at 7:57

fantasticoder
1,449 2 16 40

0

Hallgrim's answer was excellent. Here's a small tweak in case you need to parse out additional whitespace characters and linebreaks with a c# regular expression. I needed this in the case of a serialized Json value for insertion into google sheets and ran into trouble as the code was inserting tabs, +, spaces, etc.

```
provider.GenerateCodeFromExpression(new CodePrimitiveExpression(inp
var literal = writer.ToString();
var r2 = new Regex(@"\"" \+.\n[\s]+\""", RegexOptions.ECMAScript);
literal = r2.Replace(literal, "");
return literal;
```

answered Feb 18 at 19:19

Alexander Yoshi
40 7

-1

I submit my own implementation, which handles `null` values and should be more performant on account of using array lookup tables, manual hex conversion, and avoiding `switch` statements.

```
using System;
using System.Text;
using System.Linq;

public static class StringLiteralEncoding {
  private static readonly char[] HEX_DIGIT_LOWER = "0123456789abcdef"
  private static readonly char[] LITERALENCODE_ESCAPE_CHARS;

  static StringLiteralEncoding() {
    // Per http://msdn.microsoft.com/en-us/library/h21280bw.aspx
    var escapes = new string[] { "\aa", "\bb", "\ff", "\nn", "\rr", "
"\\\\", "??", "\00" };
    LITERALENCODE_ESCAPE_CHARS = new char[escapes.Max(e => e[0]) + 1]
    foreach(var escape in escapes)
      LITERALENCODE_ESCAPE_CHARS[escape[0]] = escape[1];
  }

  /// <summary>
  /// Convert the string to the equivalent C# string literal, enclosi
double quotes and inserting
  /// escape sequences as necessary.
  /// </summary>
  /// <param name="s">The string to be converted to a C# string liter
  /// <returns><paramref name="s"/> represented as a C# string litera
  public static string Encode(string s) {
    if(null == s) return "null";

    var sb = new StringBuilder(s.Length + 2).Append('"');
    for(var rp = 0; rp < s.Length; rp++) {
      var c = s[rp];
      if(c < LITERALENCODE_ESCAPE_CHARS.Length && '\0' != LITERALENCC
        sb.Append('\\').Append(LITERALENCODE_ESCAPE_CHARS[c]);
      else if('~' >= c && c >= ' ')
        sb.Append(c);
      else
        sb.Append(@"\x")
          .Append(HEX_DIGIT_LOWER[c >> 12 & 0x0F])
          .Append(HEX_DIGIT_LOWER[c >>  8 & 0x0F])
          .Append(HEX_DIGIT_LOWER[c >>  4 & 0x0F])
          .Append(HEX_DIGIT_LOWER[c       & 0x0F]);
    }

    return sb.Append('"').ToString();
  }
}
```

edited Jun 9 '13 at 16:37

answered Jun 3 '13 at 22:15

J Cracknell
2,909 1 14 13

-7

Code:

```
string someString1 = "\tHello\r\n\tWorld!\r\n";
string someString2 = @"\tHello\r\n\tWorld!\r\n";

Console.WriteLine(someString1);
Console.WriteLine(someString2);
```

Output:

```
    Hello
    World!

\tHello\r\n\tWorld!\r\n
```

Is this what you want?

answered Nov 27 '08 at 14:36
Ricardo Amaral
7,476 39 144 246

I have someString1, but it is read from a file. I want it to appear as someString2 after calling some method. – Hallgrim Nov 27 '08 at 21:51