Path.Combine for URLs?

Ask Question

1116

128

Path.Combine is handy, but is there a similar function in the .NET framework for URLs?

I'm looking for syntax like this:

```
Url.Combine("http://MyUrl.com/", "/Images/Image.jpg")
```

which would return:

```
"http://MyUrl.com/Images/Image.jpg"
```

c# .net asp.net url path

edited Feb 3 '15 at 15:01

Peter Mortensen
13.9k 19 87 113

asked Dec 16 '08 at 21:42

Brian MacKay
17.2k 13 70 109

12 Flurl includes a `Url.Combine` method that does just that. – Todd Menier Feb 21 '14 at 6:18

1 Actually, the // is handled by the routing of the website or server and not by the browser. It will send what you put into the address bar. That's why we get problems when we type htp:// instead of http:// So the // can cause major problems on some sites. I am writing a .dll for a crawler which handles a particular website which throws a 404 if you have // in the url. – Dave Gordon Jul 7 '14 at 8:11

33 Answers

1 2 next

21

There is a Todd Menier's comment above that Flurl includes a Url.Combine.

More details:

> Url.Combine is basically a Path.Combine for URLs, ensuring one and only one separator character between parts:

```
var url = Url.Combine(
    "http://foo.com/",
    "/too/", "/many/", "/slashes/",
    "too", "few?",
    "x=1", "y=2"
// result: "http://www.foo.com/too/many/slashes/too/few?x=1&y=2"
```

Get Flurl.Http on NuGet:

PM> Install-Package Flurl.Http

Or get the stand-alone URL builder without the HTTP features:

PM> Install-Package Flurl

edited Jan 9 at 20:54

answered Apr 13 '18 at 6:00

Michael Freidgeim
13.6k 6 91 115

3 Well, this question gets a lot of traffic, and the answer with 1000+ upvotes does not actually work in all cases. Years later, I actually use Flurl for this, so I am accepting this one. It seems to work in all cases I have encountered. If people don't want to take a dependency, I posted an answer that also works fine. – Brian MacKay Nov 28 '18 at 15:33

0

For what it's worth, here a couple of extension methods. The first one will combine paths and the second one adds parameters to the URL.

```
    public static string CombineUrl(this string root, string path, p
paths)
    {
        if (string.IsNullOrWhiteSpace(path))
        {
            return root;
        }

        Uri baseUri = new Uri(root);
        Uri combinedPaths = new Uri(baseUri, path);

        foreach (string extendedPath in paths)
        {
```

```
            combinedPaths = new Uri(combinedPaths, extendedPath);
        }

        return combinedPaths.AbsoluteUri;
    }

    public static string AddUrlParams(this string url, Dictionary<st
parameters)
    {
        if (parameters == null || !parameters.Keys.Any())
        {
            return url;
        }

        var tempUrl = new StringBuilder($"{url}?");
        int count = 0;

        foreach (KeyValuePair<string, string> parameter in parameter
        {
            if (count > 0)
            {
                tempUrl.Append("&");
            }

            tempUrl.Append($"{WebUtility.UrlEncode(parameter.Key)}=
{WebUtility.UrlEncode(parameter.Value)}");
            count++;
        }

        return tempUrl.ToString();
    }
```

answered Mar 25 at 17:29

LawMan
2,529 21 24

1073

Uri has a constructor that should do this for you: `new Uri(Uri baseUri, string relativeUri)`

Here's an example:

```
Uri baseUri = new Uri("http://www.contoso.com");
Uri myUri = new Uri(baseUri, "catalog/shownew.htm");
```

Note from editor: Beware, this method does not work as expected. It can cut part of baseUri in some cases. See comments and other answers.

edited Nov 26 '18 at 9:32

Karel Kral
3,113 4 26 33

answered Oct 6 '09 at 19:37

Joel Beckham
15k 3 27 56

334 I like the use of the Uri class, unfortunately it will not behave like Path.Combine as the OP asked. For example new Uri(new Uri("test.com/mydirectory/"), "/helloworld.aspx").ToString() gives you "test.com/helloworld.aspx"; which would be incorrect if we wanted a Path.Combine style result. – Doctor Jones Oct 28 '10 at 15:20

171 It's all in the slashes. If the relative path part starts with a slash, then it behaves as you described. But, if you leave the slash out, then it works the way you'd expect (note the missing slash on the second parameter): new Uri(new Uri("test.com/mydirectory/"), "helloworld.aspx").ToString() results in "test.com/mydirectory/helloworld.aspx". Path.Combine behaves similarly. If the relative path parameter starts with a slash, it only returns the relative path and doesn't combine them. – Joel Beckham Oct 28 '10 at 22:11

61 If your baseUri happened to be "test.com/mydirectory/mysubdirectory" then the result would be "test.com/mydirectory/helloworld.aspx" instead of "test.com/mydirectory/mysubdirectory/helloworld.aspx". The subtle difference is the lack of trailing slash on the first parameter. I'm all for using existing framework methods, if I have to have the trailing slash there already then I think that doing partUrl1 + partUrl2 smells a lot less - I could've potentially been chasing that trailing slash round for quite a while all for the sake of not doing string concat. – Carl Jan 12 '11 at 16:10

58 The only reason I want a URI combine method is so that I don't have to check for the trailing slash. Request.ApplicationPath is '/' if your application is at the root, but '/foo' if it's not. – nickd Mar 25 '11 at 16:44

19 I -1 this answer because this doesn't answer the problem. When you want to combine url, like when you want to use Path.Combine, you don't want to care about the trailing /. and with this, you have to care. I prefer solution of Brian MacKay or mdsharpe above – Baptiste Pernet Apr 21 '11 at 15:56

3

I find the following useful and has the following features :

- Throws on null or white space
- Takes multiple `params` parameter for multiple Url segments
- throws on null or empty

Class

```
public static class UrlPath
{
   private static string InternalCombine(string source, string dest)
   {
      if (string.IsNullOrWhiteSpace(source))
         throw new ArgumentException("Cannot be null or white space"

      if (string.IsNullOrWhiteSpace(dest))
         throw new ArgumentException("Cannot be null or white space"

      return $"{source.TrimEnd('/', '\\')}/{dest.TrimStart('/', '\\'
   }

   public static string Combine(string source, params string[] args)
       => args.Aggregate(source, InternalCombine);
}
```

Tests

```
UrlPath.Combine("test1", "test2");
UrlPath.Combine("test1//", "test2");
UrlPath.Combine("test1", "/test2");

// Result = test1/test2

UrlPath.Combine(@"test1\/\/\", @"\/\/\\\\\//test2", @"\/\/\\\\\//te

// Result = test1/test2/test3

UrlPath.Combine("/test1/", "/test2/", null);
UrlPath.Combine("", "/test2/");
UrlPath.Combine("/test1/", null);

// Throws an ArgumentException
```

edited Oct 27 '18 at 1:18

answered Feb 28 '17 at 2:11

Michael Randall
38k 8 45 73

1

Why not just use the following.

```
System.IO.Path.Combine(rootUrl, subPath).Replace(@"\", "/")
```

edited Oct 22 '18 at 8:42

answered Nov 17 '17 at 13:11

Andreas
2,308 1 27 46

1

I found that the `Uri` constructor flips '\' into '/'. So you can also use `Path.Combine`, with the `Uri` constructor.

```
Uri baseUri = new Uri("http://MyUrl.com");
string path = Path.Combine("Images", "Image.jpg");
Uri myUri = new Uri(baseUri, path);
```

edited Oct 18 '18 at 20:10

Peter Mortensen
13.9k 19 87 113

answered Sep 30 '18 at 10:43

skippy
31 4

0

We use the following simple helper method to join an arbitrary number of URL parts together:

```
public static string JoinUrlParts(params string[] urlParts)
{
    return string.Join("/", urlParts.Where(up =>
!string.IsNullOrEmpty(up)).ToList().Select(up => up.Trim('/')).ToArr
}
```

Note, that it doesn't support '../../something/page.htm'-style relative URLs!

edited Oct 18 '18 at 20:08

Peter Mortensen
13.9k 19 87 113

answered Mar 22 '18 at 14:19

pholpar
1,217 9 17

0

Both of these work:

```
  Uri final = new Uri(Regex.Replace(baseUrl + "/" + relativePath, "(
"/"));
```

Or

```
  Uri final =new Uri(string.Format("{0}/{1}", baseUrl.ToString().Tri
relativePath.ToString().TrimStart('/')));
```

I.e. if

```
baseUrl = "http://tesrurl.test.com/Int18"
```

and

```
relativePath = "To_Folder"

output = http://tesrurl.test.com/Int18/To_Folder
```

Some errors will appear for the code below:

```
// If you use the below code, some issues will be there in the fina
Uri final = new Uri(baseUrl, relativePath);
```

edited Oct 18 '18 at 20:07

Peter Mortensen
13.9k 19 87 113

answered Apr 24 '17 at 7:43

DAre G
142 9

3

I created this function that will make your life easier:

```
    /// <summary>
    /// The ultimate Path combiner of all time
    /// </summary>
    /// <param name="IsURL">
    /// true - if the paths are Internet URLs, false - if the paths
is very important as this will be used to decide which separator wil
    /// </param>
    /// <param name="IsRelative">Just adds the separator at the begi
    /// <param name="IsFixInternal">Fix the paths from within (by re
separators and correcting the separators)</param>
    /// <param name="parts">The paths to combine</param>
    /// <returns>the combined path</returns>
    public static string PathCombine(bool IsURL , bool IsRelative ,
params string[] parts)
    {
        if (parts == null || parts.Length == 0) return string.Empty;
        char separator = IsURL ? '/' : '\\';

        if (parts.Length == 1 && IsFixInternal)
        {
            string validsingle;
            if (IsURL)
            {
                validsingle = parts[0].Replace('\\' , '/');
            }
            else
            {
                validsingle = parts[0].Replace('/' , '\\');
            }
            validsingle = validsingle.Trim(separator);
            return (IsRelative ? separator.ToString() : string.Empty
        }

        string final = parts
            .Aggregate
```

```
                (
                (string first , string second) =>
                {
                    string validfirst;
                    string validsecond;
                    if (IsURL)
                    {
                        validfirst = first.Replace('\\' , '/');
                        validsecond = second.Replace('\\' , '/');
                    }
                    else
                    {
                        validfirst = first.Replace('/' , '\\');
                        validsecond = second.Replace('/' , '\\');
                    }
                    var prefix = string.Empty;
                    if (IsFixInternal)
                    {
                        if (IsURL)
                        {
                            if (validfirst.Contains("://"))
                            {
                                var tofix = validfirst.Substring(validfi
3);
                                prefix = validfirst.Replace(tofix ,
string.Empty).TrimStart(separator);

                                var tofixlist = tofix.Split(new[] { sepa
StringSplitOptions.RemoveEmptyEntries);

                                validfirst = separator + string.Join(sep
tofixlist);
                            }
                            else
                            {
                                var firstlist = validfirst.Split(new[] {
StringSplitOptions.RemoveEmptyEntries);
                                validfirst = string.Join(separator.ToStr
                            }

                            var secondlist = validsecond.Split(new[] { s
StringSplitOptions.RemoveEmptyEntries);
                            validsecond = string.Join(separator.ToString
                        }
                        else
                        {
                            var firstlist = validfirst.Split(new[] { sep
```

```
StringSplitOptions.RemoveEmptyEntries);
                        var secondlist = validsecond.Split(new[] { s
StringSplitOptions.RemoveEmptyEntries);

                        validfirst = string.Join(separator.ToString(
                        validsecond = string.Join(separator.ToString
                    }
                }
                return prefix + validfirst.Trim(separator) + separat
validsecond.Trim(separator);
            }
            );
        return (IsRelative ? separator.ToString() : string.Empty) +
    }
```

It works for URLs as well as normal paths.

Usage:

```
    // Fixes internal paths
    Console.WriteLine(PathCombine(true , true , true , @"\/\/folder
1\/\/\/\\/\folder2\///folder3\\/" , @"/\somefile.ext\/\//\"));
    // Result: /folder 1/folder2/folder3/somefile.ext

    // Doesn't fix internal paths
    Console.WriteLine(PathCombine(true , true , false , @"\/\/folder
1\/\/\/\\/\folder2\///folder3\\/" , @"/\somefile.ext\/\//\"));
    //result : /folder 1//////////folder2////folder3/somefile.ext

    // Don't worry about URL prefixes when fixing internal paths
    Console.WriteLine(PathCombine(true , false , true ,
@"/\/\/https:/\/\/\lul.com\/\/\/\\/\folder2\///folder3\\/" , @"/\som
    // Result: https://lul.com/folder2/folder3/somefile.ext

    Console.WriteLine(PathCombine(false , true , true ,
@"../../../\\..\...\./../somepath" , @"anotherpath"));
    // Result: \..\..\..\..\...\.\..\somepath\anotherpath
```

edited Oct 18 '18 at 20:04

Peter Mortensen

13.9k 19 87 113

answered Jul 21 '16 at 18:19

bigworld12
593 6 26

82

Ryan Cook's answer is close to what I'm after and may be more appropriate for other developers. However, it adds http:// to the beginning of the string and in general it does a bit more formatting than I'm after.

Also, for my use cases, resolving relative paths is not important.

mdsharp's answer also contains the seed of a good idea, although that actual implementation needed a few more details to be complete. This is an attempt to flesh it out (and I'm using this in production):

C#

```
public string UrlCombine(string url1, string url2)
{
    if (url1.Length == 0) {
        return url2;
    }

    if (url2.Length == 0) {
        return url1;
    }

    url1 = url1.TrimEnd('/', '\\');
    url2 = url2.TrimStart('/', '\\');

    return string.Format("{0}/{1}", url1, url2);
}
```

VB.NET

```
Public Function UrlCombine(ByVal url1 As String, ByVal url2 As Strin
    If url1.Length = 0 Then
```

```
            Return url2
        End If

        If url2.Length = 0 Then
            Return url1
        End If

        url1 = url1.TrimEnd("/"c, "\"c)
        url2 = url2.TrimStart("/"c, "\"c)

        Return String.Format("{0}/{1}", url1, url2)
    End Function
```

This code passes the following test, which happens to be in VB:

```
    <TestMethod()> Public Sub UrlCombineTest()
        Dim target As StringHelpers = New StringHelpers()

        Assert.IsTrue(target.UrlCombine("test1", "test2") = "test1/test2
        Assert.IsTrue(target.UrlCombine("test1/", "test2") = "test1/test
        Assert.IsTrue(target.UrlCombine("test1", "/test2") = "test1/test
        Assert.IsTrue(target.UrlCombine("test1/", "/test2") = "test1/tes
        Assert.IsTrue(target.UrlCombine("/test1/", "/test2/") = "/test1/
        Assert.IsTrue(target.UrlCombine("", "/test2/") = "/test2/")
        Assert.IsTrue(target.UrlCombine("/test1/", "") = "/test1/")
    End Sub
```

edited Oct 18 '18 at 20:01

Peter Mortensen
13.9k 19 87 113

answered May 10 '10 at 21:53

Brian MacKay
17.2k 13 70 109

4 Talking of details: what about the mandatory `ArgumentNullException("url1")` if the argument is `Nothing` ? Sorry, just being picky ;-). Note that a backslash has nothing to do in a URI (and if it is there, it should not be trimmed), so you can remove that from your TrimXXX. – Abel Aug 26 '10 at 22:21

4 you can use params string[] and recursively join them to allow more than 2

combinations – Jaider Jun 12 '12 at 22:47

4 I sure wish this was in the Base Class Library like Path.Combine. – Uriah Blatherwick Aug 18 '14 at 18:30

1 @MarkHurd I edited the code again, so that it's behaviorally the same as the C#, and syntactically equivalent as well. – JJS Jul 7 '16 at 19:23

1 @BrianMacKay i broke it, markhurd pointed out my mistake and rolled back, i updated again... cheers – JJS Jul 11 '16 at 13:28

7

An easy way to combine them and ensure it's always correct is:

```
string.Format("{0}/{1}", Url1.Trim('/'), Url2);
```

edited Oct 18 '18 at 20:00

Peter Mortensen
13.9k 19 87 113

answered May 12 '10 at 21:42

Alex
86 1 3

-1

I have to point out that `Path.Combine` appears to work for this also directly, at least on .NET 4.

edited Oct 18 '18 at 19:59

Peter Mortensen
13.9k 19 87 113

answered Jul 1 '10 at 16:02

Chris Marisic

22.1k 18 134 235

10 If you use Path.Combine u will end up with something like this: www.site.com/foo\wrong\icon.png – Lehto Oct 25 '10 at 13:56

16

I just put together a small extension method:

```
public static string UriCombine (this string val, string append)
        {
            if (String.IsNullOrEmpty(val)) return append;
            if (String.IsNullOrEmpty(append)) return val;
            return val.TrimEnd('/') + "/" + append.TrimStart('/');
        }
```

It can be used like this:

```
"www.example.com/".UriCombine("/images").UriCombine("first.jpeg");
```

edited Oct 18 '18 at 19:59

Peter Mortensen
13.9k 19 87 113

answered Nov 25 '10 at 8:43

urza
177 1 2

29

Path.Combine does not work for me because there can be characters like "|" in QueryString arguments and therefore the URL, which will result in an ArgumentException.

I first tried the new `Uri(Uri baseUri, string relativeUri)` approach, which failed for me because of URIs like

`http://www.mediawiki.org/wiki/Special:SpecialPages` :

```
new Uri(new Uri("http://www.mediawiki.org/wiki/"), "Special:SpecialP
```

will result in Special:SpecialPages, because of the colon after `Special` that denotes a scheme.

So I finally had to take mdsharpe/Brian MacKays route and developed it a bit further to work with multiple URI parts:

```
public static string CombineUri(params string[] uriParts)
{
    string uri = string.Empty;
    if (uriParts != null && uriParts.Count() > 0)
    {
        char[] trims = new char[] { '\\', '/' };
        uri = (uriParts[0] ?? string.Empty).TrimEnd(trims);
        for (int i = 1; i < uriParts.Count(); i++)
        {
            uri = string.Format("{0}/{1}", uri.TrimEnd(trims), (uriP
string.Empty).TrimStart(trims));
        }
    }
    return uri;
}
```

Usage: `CombineUri("http://www.mediawiki.org/", "wiki", "Special:SpecialPages")`

edited Oct 18 '18 at 19:58

Peter Mortensen
13.9k 19 87 113

answered Jul 15 '11 at 8:17

Mike Fuchs
9,880 2 42 61

1 +1: Now we're talking... I'm going to try this out. This might even end up being the new accepted answer. After trying to new Uri() method I really

don't like it. Too finnicky. – Brian MacKay Jul 18 '11 at 14:23

Use this:

1

```
public static class WebPath
{
    public static string Combine(params string[] args)
    {
        var prefixAdjusted = args.Select(x => x.StartsWith("/") && !x
? x.Substring(1) : x);
        return string.Join("/", prefixAdjusted);
    }
}
```

edited Oct 18 '18 at 19:57

Peter Mortensen
13.9k 19 87 113

answered Dec 10 '12 at 15:31

Martin Murphy
1,507 2 14 23

1 x.StartsWith("/") && !x.StartsWith("http") - why the http check? what do you gain? – penguat Dec 13 '12 at 10:03

I have combined all the previous answers:

0

```
public static string UrlPathCombine(string path1, string path2)
{
    path1 = path1.TrimEnd('/') + "/";
    path2 = path2.TrimStart('/');
```

```
        return Path.Combine(path1, path2)
            .Replace(Path.DirectorySeparatorChar, Path.AltDirectoryS
    }

    [TestMethod]
    public void TestUrl()
    {
        const string P1 = "http://msdn.microsoft.com/slash/library//
        Assert.AreEqual("http://msdn.microsoft.com/slash/library/sit
UrlPathCombine(P1, "//site.aspx"));

        var path = UrlPathCombine("Http://MyUrl.com/", "Images/Image

        Assert.AreEqual(
            "Http://MyUrl.com/Images/Image.jpg",
            path);
    }
```

edited Oct 18 '18 at 19:56

Peter Mortensen
13.9k 19 87 113

answered Apr 25 '13 at 10:51

Per G
322 3 16

1 You could have used VirtualPathUtiliy class to append and remove trailing slashes safely. Check out my answer: stackoverflow.com/a/23399048/3481183 – Believe2014 May 1 '14 at 15:38

7

I found `UriBuilder` worked really well for this sort of thing:

```
UriBuilder urlb = new UriBuilder("http", _serverAddress, _webPort, _
Uri url = urlb.Uri;
return url.AbsoluteUri;
```

See UriBuilder Class - MSDN for more constructors and documentation.

edited Oct 18 '18 at 19:55

Peter Mortensen
13.9k 19 87 113

answered May 22 '13 at 12:19

javajavajavajavajava
730 2 9 19

1

Here is my approach and I will use it for myself too:

```
public static string UrlCombine(string part1, string part2)
{
    string newPart1 = string.Empty;
    string newPart2 = string.Empty;
    string seperator = "/";

    // If either part1 or part 2 is empty,
    // we don't need to combine with seperator
    if (string.IsNullOrEmpty(part1) || string.IsNullOrEmpty(part2))
    {
        seperator = string.Empty;
    }

    // If part1 is not empty,
    // remove '/' at last
    if (!string.IsNullOrEmpty(part1))
    {
        newPart1 = part1.TrimEnd('/');
    }

    // If part2 is not empty,
    // remove '/' at first
    if (!string.IsNullOrEmpty(part2))
    {
        newPart2 = part2.TrimStart('/');
    }
```

```
    // Now finally combine
    return string.Format("{0}{1}{2}", newPart1, seperator, newPart2)
}
```

edited Oct 18 '18 at 19:54

Peter Mortensen
13.9k 19 87 113

answered Aug 3 '13 at 3:39

Amit Bhagat
2,740 2 16 20

1

Use:

```
    private Uri UriCombine(string path1, string path2, string path3 =
"")
    {
        string path = System.IO.Path.Combine(path1, path2.TrimStart(
path3.TrimStart('\\', '/'), path4.TrimStart('\\', '/'));
        string url = path.Replace('\\','/');
        return new Uri(url);
    }
```

It has the benefit of behaving exactly like `Path.Combine` .

edited Oct 18 '18 at 19:53

Peter Mortensen
13.9k 19 87 113

answered Feb 18 '14 at 15:31

TruthOf42
879 3 10 32

8

Combining multiple parts of a URL could be a little bit tricky. You can use the two-parameter constructor `Uri(baseUri, relativeUri)`, or you can use the `Uri.TryCreate()` utility function.

In either case, you might end up returning an incorrect result because these methods keep on truncating the relative parts off of the first parameter `baseUri`, i.e. from something like `http://google.com/some/thing` to `http://google.com`.

To be able to combine multiple parts into a final URL, you can copy the two functions below:

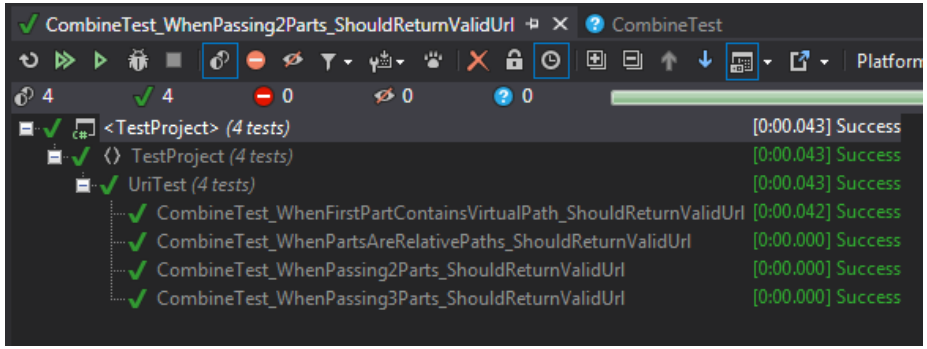
```
public static string Combine(params string[] parts)
{
    if (parts == null || parts.Length == 0) return string.Empty;

    var urlBuilder = new StringBuilder();
    foreach (var part in parts)
    {
        var tempUrl = tryCreateRelativeOrAbsolute(part);
        urlBuilder.Append(tempUrl);
    }
    return VirtualPathUtility.RemoveTrailingSlash(urlBuilder.ToS
}

private static string tryCreateRelativeOrAbsolute(string s)
{
    System.Uri uri;
    System.Uri.TryCreate(s, UriKind.RelativeOrAbsolute, out uri)
    string tempUrl = VirtualPathUtility.AppendTrailingSlash(uri.
    return tempUrl;
}
```

Full code with unit tests to demonstrate usage can be found at https://uricombine.codeplex.com/SourceControl/latest#UriCombine/Uri.cs

I have unit tests to cover the three most common cases:



edited Oct 18 '18 at 19:52

Peter Mortensen
13.9k 19 87 113

answered Apr 30 '14 at 22:21

Believe2014
2,875 1 19 33

1 Looks pretty good to me. Although you could replace the I loop with a foreach loop for better clarity. – Chris Marisic May 1 '14 at 18:55

1 +1 for all the extra effort. I need to maintain this question a bit for some of the higher voted answers, you have thrown down the gauntlet. ;) – Brian MacKay May 4 '14 at 11:42

0

Well, I just concatenate two strings and use regular expressions to do the cleaning part.

```
public class UriTool
{
    public static Uri Join(string path1, string path2)
    {
        string url = path1 + "/" + path2;
        url = Regex.Replace(url, "(?<!http:)/{2,}", "/");
```

```
            return new Uri(url);
        }
    }
```

So, you can use it like this:

```
    string path1 = "http://someaddress.com/something/";
    string path2 = "/another/address.html";
    Uri joinedUri = UriTool.Join(path1, path2);

    // joinedUri.ToString() returns
"http://someaddress.com/something/another/address.html"
```

edited Oct 18 '18 at 19:50

Peter Mortensen
13.9k 19 87 113

answered May 15 '14 at 23:49

Marcio Martins
111 3 9

Rules while combining URLs with a URI

2

To avoid strange behaviour there's one rule to follow:

- The path (directory) must end with '/'. If the path ends without '/', the last part is treated like a file-name, and it'll be concatenated when trying to combine with the next URL part.
- There's one exception: the base URL address (without directory info) needs not to end with '/'
- the path part must not start with '/'. If it start with '/', every existing relative information from URL is dropped...adding a `string.Empty` part path will remove the relative directory from the URL too!

If you follow rules above, you can combine URLs with the code below. Depending on your situation, you can add multiple 'directory' parts to the URL...

```
        var pathParts = new string[] { destinationBaseUrl, destinati
fileName };

        var destination = pathParts.Aggregate((left, right) =>
        {
            if (string.IsNullOrWhiteSpace(right))
                return left;

            return new Uri(new Uri(left), right).ToString();
        });
```

edited Oct 18 '18 at 19:49

Peter Mortensen
13.9k 19 87 113

answered Apr 5 '16 at 5:04

baHI
444 6 11

32

Based on the sample URL you provided, I'm going to assume you want to combine URLs that are relative to your site.

Based on this assumption I'll propose this solution as the most appropriate response to your question which was: "Path.Combine is handy, is there a ***similar function*** in the framework for URLs?"

Since there the is a ***similar function*** in the framework for URLs I propose the correct is: "VirtualPathUtility.Combine" method. Here's the MSDN reference link: VirtualPathUtility.Combine Method

There is one caveat: I believe this only works for URLs relative to your site (that is, you cannot use it to generate links to another web

site. For example, `var url = VirtualPathUtility.Combine("www.google.com", "accounts/widgets");`).

edited Sep 20 '18 at 9:05

Kolappan Nathan
606 1 15 20

answered Mar 28 '10 at 0:21

Jeronimo Colon III
521 5 6

2 The caveat is correct, it cannot work with absolute uris and the result is always relative from the root. But it has an added benefit, it processes the tilde, as with "~/". This makes it a shortcut for `Server.MapPath` and combining. – Abel Aug 26 '10 at 22:18

0

A simple one liner:

```
public static string Combine(this string uri1, string uri2) => $"{uri1.TrimEnd('/')}/{uri2.TrimStart('/')}";
```

Inspired by @Matt Sharpe's answer.

answered Nov 6 '17 at 12:41

Nick N.
8,028 2 31 62

4

Here's Microsoft's (OfficeDev PnP) method UrlUtility.Combine:

```
const char PATH_DELIMITER = '/';

/// <summary>
```

```
        /// Combines a path and a relative path.
        /// </summary>
        /// <param name="path"></param>
        /// <param name="relative"></param>
        /// <returns></returns>
        public static string Combine(string path, string relative)
        {
            if(relative == null)
                relative = String.Empty;

            if(path == null)
                path = String.Empty;

            if(relative.Length == 0 && path.Length == 0)
                return String.Empty;

            if(relative.Length == 0)
                return path;

            if(path.Length == 0)
                return relative;

            path = path.Replace('\\', PATH_DELIMITER);
            relative = relative.Replace('\\', PATH_DELIMITER);

            return path.TrimEnd(PATH_DELIMITER) + PATH_DELIMITER +
relative.TrimStart(PATH_DELIMITER);
        }
```

Source: GitHub

edited Jun 6 '17 at 21:25

Chris Marisic

22.1k 18 134 235

answered Nov 1 '15 at 18:34

user3638471

2 Edited to clarify what class it belongs to – user3638471 Nov 3 '15 at 19:57

I used this code to solve the problem:

0

```
string[] brokenBaseUrl = Context.Url.TrimEnd('/').Split('/');
string[] brokenRootFolderPath = RootFolderPath.Split('/');

for (int x = 0; x < brokenRootFolderPath.Length; x++)
{
    //if url doesn't already contain member, append it to the end of
in front
    if (!brokenBaseUrl.Contains(brokenRootFolderPath[x]))
    {
        if (x == 0)
        {
            RootLocationUrl = Context.Url.TrimEnd('/');
        }
        else
        {
            RootLocationUrl += String.Format("/{0}", brokenRootFolde
        }
    }
}
```

edited Mar 17 '16 at 15:50

gunr2171

7,709 10 47 67

answered Dec 3 '15 at 23:17

Joshua Smith

111 10

My generic solution:

3

```
public static string Combine(params string[] uriParts)
{
    string uri = string.Empty;
    if (uriParts != null && uriParts.Any())
```

```
    {
        char[] trims = new char[] { '\\', '/' };
        uri = (uriParts[0] ?? string.Empty).TrimEnd(trims);

        for (int i = 1; i < uriParts.Length; i++)
        {
            uri = string.Format("{0}/{1}", uri.TrimEnd(trims), (uriP
string.Empty).TrimStart(trims));
        }
    }

    return uri;
}
```

answered May 17 '15 at 7:52

Alex Titarenko

456 3 7

12

Witty example, Ryan, to end with a link to the function. Well done.

One recommendation Brian: if you wrap this code in a function, you may want to use a UriBuilder to wrap the base URL prior to the TryCreate call.

Otherwise, the base URL MUST include the scheme (where the UriBuilder will assume http://). Just a thought:

```
public string CombineUrl(string baseUrl, string relativeUrl) {
    UriBuilder baseUri = new UriBuilder(baseUrl);
    Uri newUri;

    if (Uri.TryCreate(baseUri.Uri, relativeUrl, out newUri))
        return newUri.ToString();
    else
        throw new ArgumentException("Unable to combine specified url
}
```

edited Feb 3 '15 at 15:04

Peter Mortensen
13.9k 19 87 113

answered Jul 30 '09 at 15:08

mtazva
917 7 13

117

There's already some great answers here. Based on mdsharpe suggestion, here's an extension method that can easily be used when you want to deal with Uri instances:

```
using System;
using System.Linq;

public static class UriExtensions
{
    public static Uri Append(this Uri uri, params string[] paths)
    {
        return new Uri(paths.Aggregate(uri.AbsoluteUri, (current, pa
string.Format("{0}/{1}", current.TrimEnd('/'), path.TrimStart('/')))
    }
}
```

And usage example:

```
var url = new Uri("http://example.com/subpath/").Append("/part1/", "
```

This will produce http://example.com/subpath/part1/part2

edited Mar 20 '14 at 5:17

jdphenix
10.7k 3 32 60

answered Nov 3 '11 at 10:20

Ales Potocnik Hahonina

2,082 1 22 30

2 This solution makes it trivial to write a UriUtils.Combine("base url", "part1", "part2", ...) static method that is very similar to Path.Combine(). Nice! – angularsen Nov 19 '11 at 15:23

-2

I haven't used the following code yet, but found it during my internet travels to solve a URL combine problem - hoping it's a succinct (and successful!) answer:

```
VirtualPathUtility.Combine
```

answered Feb 28 '13 at 3:40

missbassethorn
1

2 Not too useful really. There's a number of Google hits explaining some of its issues, but, as well as not liking "http://..." at the start, it actually removes the last sub path of the first argument if it doesn't end in a "/"! The MSDN description sounds fine though! – Mark Hurd Mar 2 '13 at 7:21

1 2 next

protected by Sheridan Oct 22 '14 at 14:51

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?