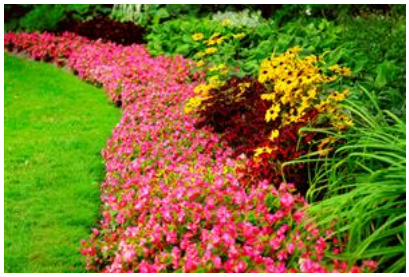
Simple is hard. --Steve Lautenschlager

# How Do I Get Paths and URL fragments from the HttpRequest object?

By **Steve (https://www.cambiaresearch.com/users/a375d2e1-f59c-4a09-9157-c65dbe019745/steve)** on **Tuesday, January 09, 2007**
Updated **Friday, April 22, 2016**
Viewed **258,243** times. (**50** times today.)

Developer (https://www.cambiaresearch.com/articles/53/how-do-i-get-paths-and-url-fragments-from-the-httprequest-object) Tips and Tutorials (https://www.cambiaresearch.com/articles/53/how-do-i-get-paths-and-url-fragments-from-the-httprequest-object)



## Summary

If you have done much ASP.NET programming you have probably spent a lot of time mapping urls to physical disk locations and vice versa. This need arises whenever you store files on the server or do some kind of URL parsing and re-routing of incoming requests.

This article will examine the properties of the Request object that will provide path and url information related to the application and the current request.

First, here are a couple of tables of useful properties on the Request object and an example of the text they return for a given input URL.

For some reason unclear to me, the Url.Fragment property is usually empty instead of showing "#fragment".

# Input: http://localhost:96/Cambia3/Temp/Test.aspx?q=item#fragment

Some HttpRequest path and URL properties:

| | |
|---|---|
| Request.ApplicationPath: | **/Cambia3** |
| Request.CurrentExecutionFilePath: | **/Cambia3/Temp/Test.aspx** |
| Request.FilePath: | **/Cambia3/Temp/Test.aspx** |
| Request.Path: | **/Cambia3/Temp/Test.aspx** |
| Request.PathInfo: | |
| Request.PhysicalApplicationPath: | **D:\Inetpub\wwwroot\CambiaWeb\Cambia3** |
| Request.RawUrl: | **/Cambia3/Temp/Test.aspx?query=arg** |
| Request.Url.AbsolutePath: | **/Cambia3/Temp/Test.aspx** |
| Request.Url.AbsoluteUri: | **http://localhost:96/Cambia3/Temp/Test.aspx?query=arg** |
| Request.Url.Fragment: | |
| Request.Url.Host: | **localhost** |
| Request.Url.Authority: | **localhost:96** |
| Request.Url.LocalPath: | **/Cambia3/Temp/Test.aspx** |
| Request.Url.PathAndQuery: | **/Cambia3/Temp/Test.aspx?query=arg** |
| Request.Url.Port: | **96** |
| Request.Url.Query: | **?query=arg** |
| Request.Url.Scheme: | **http** |
| Request.Url.Segments: | **/**<br>**Cambia3/**<br>**Temp/**<br>**Test.aspx** |

# Input: http://localhost:96/Cambia3/Temp/Test.aspx/path/info?q=item#fragment

Some HttpRequest path and URL properties:

| | |
|---|---|
| Request.ApplicationPath: | **/Cambia3** |
| Request.CurrentExecutionFilePath: | **/Cambia3/Temp/Test.aspx** |
| Request.FilePath: | **/Cambia3/Temp/Test.aspx** |
| Request.Path: | **/Cambia3/Temp/Test.aspx/path/info** |
| Request.PathInfo: | **/path/info** |
| Request.PhysicalApplicationPath: | **D:\Inetpub\wwwroot\CambiaWeb\Cambia3** |
| Request.RawUrl: | **/Cambia3/Temp/Test.aspx/path/info?query=arg** |
| Request.Url.AbsolutePath: | **/Cambia3/Temp/Test.aspx/path/info** |
| Request.Url.AbsoluteUri: | **http://localhost:96/Cambia3/Temp/Test.aspx/path/info?query=arg** |
| Request.Url.Fragment: | |
| Request.Url.Host: | **localhost** |
| Request.Url.LocalPath: | **/Cambia3/Temp/Test.aspx/path/info** |
| Request.Url.PathAndQuery: | **/Cambia3/Temp/Test.aspx/path/info?query=arg** |
| Request.Url.Port: | **96** |
| Request.Url.Query: | **?query=arg** |
| Request.Url.Scheme: | **http** |
| Request.Url.Segments: | **/**<br>**Cambia3/**<br>**Temp/**<br>**Test.aspx/**<br>**path/**<br>**info** |

The following is the C# method I used to process the URLs above and generate the output tables. You may use this method in a code-behind file for an aspx page with a Label control names lblOutput.

```
private void DisplayRequestObjectProperties()
{
   lblOutput.Text = "<table cellpadding=2 border=1>";

   lblOutput.Text += "<tr><td colspan=2 align=center>";
   lblOutput.Text +=  "Some HttpRequest path and ULR properties:";
   lblOutput.Text += "</td></tr>";

   // application path
   lblOutput.Text += "<tr><td>";
   lblOutput.Text += "Request.ApplicationPath:";
   lblOutput.Text += "</td><td>";
   lblOutput.Text += "<b>" + Request.ApplicationPath + "</b>";
   lblOutput.Text += "</td></tr>";

   // current execution file path
   lblOutput.Text += "<tr><td>";
   lblOutput.Text += "Request.CurrentExecutionFilePath:";
   lblOutput.Text += "</td><td>";
   lblOutput.Text += "<b>" + Request.CurrentExecutionFilePath + "</b>";
   lblOutput.Text += "</td></tr>";

   // file path
   lblOutput.Text += "<tr><td>";
   lblOutput.Text += "Request.FilePath:";
   lblOutput.Text += "</td><td>";
   lblOutput.Text += "<b>" + Request.FilePath + "</b>";
   lblOutput.Text += "</td></tr>";

   // path
   lblOutput.Text += "<tr><td>";
   lblOutput.Text += "Request.Path:";
   lblOutput.Text += "</td><td>";
   lblOutput.Text += "<b>" + Request.Path + "</b>";
   lblOutput.Text += "</td></tr>";

   // path info
   lblOutput.Text += "<tr><td>";
   lblOutput.Text += "Request.PathInfo:";
   lblOutput.Text += "</td><td>";
   lblOutput.Text += "<b>" + Request.PathInfo + "</b>";
   lblOutput.Text += "</td></tr>";

   // physical application path
   lblOutput.Text += "<tr><td>";
   lblOutput.Text += "Request.PhysicalApplicationPath:";
   lblOutput.Text += "</td><td>";
   lblOutput.Text += "<b>" + Request.PhysicalApplicationPath + "</b>";
   lblOutput.Text += "</td></tr>";

   // raw url
   lblOutput.Text += "<tr><td>";
   lblOutput.Text += "Request.RawUrl:";
   lblOutput.Text += "</td><td>";
   lblOutput.Text += "<b>" + Request.RawUrl + "</b>";
   lblOutput.Text += "</td></tr>";

   // absolute path
   lblOutput.Text += "<tr><td>";
   lblOutput.Text += "Request.Url.AbsolutePath:";
   lblOutput.Text += "</td><td>";
   lblOutput.Text += "<b>" + Request.Url.AbsolutePath + "</b>";
   lblOutput.Text += "</td></tr>";

   // absolute uri
   lblOutput.Text += "<tr><td>";
   lblOutput.Text += "Request.Url.AbsoluteUri:";
   lblOutput.Text += "</td><td>";
   lblOutput.Text += "<b>" + Request.Url.AbsoluteUri + "</b>";
   lblOutput.Text += "</td></tr>";

   // fragment
   lblOutput.Text += "<tr><td>";
   lblOutput.Text += "Request.Url.Fragment:";
   lblOutput.Text += "</td><td>";
   lblOutput.Text += "<b>" + Request.Url.Fragment + "</b>";
   lblOutput.Text += "</td></tr>";

   // host
   lblOutput.Text += "<tr><td>";
   lblOutput.Text += "Request.Url.Host:";
```

```
   lblOutput.Text += "</td><td>";
   lblOutput.Text += "<b>" + Request.Url.Host + "</b>";
   lblOutput.Text += "</td></tr>";

   // authority
   lblOutput.Text += "<tr><td>";
   lblOutput.Text += "Request.Url.Authority:";
   lblOutput.Text += "</td><td>";
   lblOutput.Text += "<b>" + Request.Url.Authority + "</b>";
   lblOutput.Text += "</td></tr>";

   // local path
   lblOutput.Text += "<tr><td>";
   lblOutput.Text += "Request.Url.LocalPath:";
   lblOutput.Text += "</td><td>";
   lblOutput.Text += "<b>" + Request.Url.LocalPath + "</b>";
   lblOutput.Text += "</td></tr>";

   // path and query
   lblOutput.Text += "<tr><td>";
   lblOutput.Text += "Request.Url.PathAndQuery:";
   lblOutput.Text += "</td><td>";
   lblOutput.Text += "<b>" + Request.Url.PathAndQuery + "</b>";
   lblOutput.Text += "</td></tr>";

   // port
   lblOutput.Text += "<tr><td>";
   lblOutput.Text += "Request.Url.Port:";
   lblOutput.Text += "</td><td>";
   lblOutput.Text += "<b>" + Request.Url.Port + "</b>";
   lblOutput.Text += "</td></tr>";

   // query
   lblOutput.Text += "<tr><td>";
   lblOutput.Text += "Request.Url.Query:";
   lblOutput.Text += "</td><td>";
   lblOutput.Text += "<b>" + Request.Url.Query + "</b>";
   lblOutput.Text += "</td></tr>";

   // scheme
   lblOutput.Text += "<tr><td>";
   lblOutput.Text += "Request.Url.Scheme:";
   lblOutput.Text += "</td><td>";
   lblOutput.Text += "<b>" + Request.Url.Scheme + "</b>";
   lblOutput.Text += "</td></tr>";

   // segments
   lblOutput.Text += "<tr><td>";
   lblOutput.Text += "Request.Url.Segments:";
   lblOutput.Text += "</td><td>";
   string[] segments = Request.Url.Segments;
   foreach (string s in segments)
      lblOutput.Text += "<b>" + s + "</b><br>";
   lblOutput.Text += "</td></tr>";

   lblOutput.Text += "</table>";
}
```

Back to top

17 Comments Cambia Research Login

Recommend 1 Share Sort by Best

Join the discussion…

LOG IN WITH OR SIGN UP WITH DISQUS ?

Name

**Richard Hubert** • a year ago
This one is missing, I think. Not the same as .Authority: Request.Url.GetLeftPart(UriPartial.Authority)
Reply • Share ›

**NYC** • 3 years ago
what if I am using http://www.mysite.com/defau... and I just want "price" from the URL
Reply • Share ›

**Allan Gaunt** • 3 years ago
Many thanks for taking the time to produce this
Reply • Share ›

**Keval** • 3 years ago
Nice Information :) (Y)
Reply • Share ›

**Peter Do** • 4 years ago
thank you so much!
Reply • Share ›

**Ekco Husky** • 4 years ago
Thanks a lot. This is very helpful!
Reply • Share ›

**crowel** • 5 years ago
You can parse a url by using regex , find the following example

http://net-informations.com...

tomin
Reply • Share ›

**Steve @ Cambia Research** Mod ➚ crowel • 5 years ago
Or have a look at my article on parsing URLs here: http://www.cambiaresearch.c...
Reply • Share ›

**vinay jatod** • 6 years ago
can i get detail code for accepting the request on my side in java plz send code or link where i can get this code
Reply • Share ›

**dissapoint** • 6 years ago
Request.QueryString != Request.RawUrl

Your post was misleading me... and losing 40 minuts of my time figuring out why I don't get the same result with Request.QueryString as you... Then I looked at your code and to my disapointment I found out you are really using Request.RawUrl.
Shame on you
Reply • Share ›

**Steve @ Cambia Research** Mod ➚ dissapoint • 6 years ago
Took me a while, but I finally got this article updated. I basically had a "Query" label where I should have had "RawUrl".

Hope that helps.
Reply • Share ›

**Anonymous** ➚ dissapoint • 6 years ago
You are correct. That appears to be a mistake which could clearly lead to confusion. I will get it corrected as soon as I can.
Reply • Share ›

**Rick** • 8 years ago
Just when I was getting confused and dispirited, up pops your lovely page. Thank you for the code, for the output, for putting this out here for those of us in need of help. Just wonderful.
Reply • Share ›

**Steve** • 9 years ago
In response to:
For some reason unclear to me, the Url.Fragment property is usually empty instead of showing "#fragment".

This is because most (if not all) web browsers don't send this information to the server. The "fragment" is for scrolling to a section of a web page (client-side).
Reply • Share ›

**Benj Arriola** • 9 years ago
I am a PHP person and I do a lot of algorithmic SEO type of things where I am used to $_SERVER[HTTP_HOST], $_SERVER[REQUEST_URI], $_SERVER[PATH_INFO], $_SERVER[PHP_SELF] etc... But I occasionally get some ASP.net clients every now and then and does not know these by heart. Thanks!

- Benj
http://www.benjarriola.com/
Reply • Share ›

**Oli** • 10 years ago
Thanks, really useful reference.

Btw #fragments are intended for the client and are not sent on the querystring to the server, so why Microsoft has included it I do not know.
Reply • Share ›

**Anonymous** ➚ Oli • 7 years ago
Its because Request.Url inherits from System.Uri which by definition needs to include fragments
Reply • Share ›