

1 System.Uri Class

```
2 [ILAsm]  
3 .class public serializable Uri extends System.Object  
4 [C#]  
5 public class Uri
```

6 Assembly Info:

- 7 • *Name:* System
- 8 • *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00]
- 9 • *Version:* 2.0.x.x
- 10 • *Attributes:*
 - 11 ○ CLSCompliantAttribute(true)

12 Summary

13 Provides an object representation of a uniform resource identifier (URI) as defined by
14 IETF RFC 2396.

15 Inherits From: System.Object

16

17 **Library:** Networking

18

19 **Thread Safety:** All public static members of this type are safe for multithreaded operations.
20 No instance members are guaranteed to be thread safe.

21

22 Description

23 [*Note:* A Uniform Resource Identifier (URI) is a compact string of characters used to
24 identify a resource located on a computer. A resource can be anything that has identity.
25 Examples of resources that might be accessed using a URI include an electronic
26 document, an image, a web service, and a collection of other resources. A URI is
27 represented as a sequence of characters. While the exact format of a URI is determined
28 by the protocol used to access the resource, many URI consist of four major
29 components:

30

31 *<scheme>://<authority><path>?<query>*

32

33 *Scheme* - Indicates a protocol used to access the resource.

34

35 *Authority* - Indicates the naming authority (server or registry) that governs the
36 namespace defined by the remainder of the URI. The authority component is composed
37 of *userinfo*, *host* and *port* subcomponents in the form *<userinfo>@<host>:<port>*.
38 Only the *host* subcomponent is required to be present in the *Authority* component.
39 Authority information is stored in the *System.Uri.Authority* property.

40

41 *Path* - Identifies the resource within the scope of the scheme and, if present, the
42 authority. This information is stored in the *System.Uri.AbsolutePath*,

1 `System.Uri.PathAndQuery`, and `System.Uri.LocalPath` properties.

2
3 *Query* - Parameter information that is passed to the executable script identified by the
4 URI. The query, if present, is the last element in a URI and begins with a "?". This
5 information is stored in the `System.Uri.Query` property.

6
7 *Userinfo* - [Subcomponent of *Authority*] Consists of a user name and, optionally,
8 scheme-specific authorization information used to access *Host*. The *userinfo*, if present,
9 is separated from the *Host* component by the "@" character. Note that for some URI
10 schemes, the format of the *userinfo* subcomponent is "username:password". Passing
11 authorization information in this manner is strongly discouraged due to security issues.
12 The *userinfo* information is stored in the `System.Uri.UserInfo` property.

13
14 *Host* - [Subcomponent of *Authority*] The Domain Name system (DNS) name or IP4
15 address of a machine that provides access to the resource. This information is stored in
16 the `System.Uri.Host` property.

17
18 *Port* - [Subcomponent of *Authority*] The network port number used to connect to the
19 host. If no port number is specified in the URI, most schemes designate protocols that
20 have a default port number. This information is stored in the `System.Uri.Port`
21 property.

22
23 *Fragment* - The fragment is not part of the URI, but is used in conjunction with the URI
24 and is included here for completeness. This component contains resource-specific
25 information that is used after a resource is retrieved. The *fragment*, if present, is
26 separated from the URI by the "#" character. This information is stored in the
27 `System.Uri.Fragment` property.

28
29 URIs include components consisting of or delimited by certain special (reserved)
30 characters that have a special meaning in a URI component. If the reserved meaning is
31 not intended, then the character is required to be escaped in the URI. An escaped
32 character is encoded as a character triplet consisting of the percent character "%"
33 followed by the US-ASCII character code specified as two hexadecimal digits. For
34 example, "%20" is the escaped encoding for the US-ASCII space character. The URI
35 represented by a `System.Uri` instance is always in "escaped" form. The following
36 characters are reserved:

- 37 • Semi-colon (";")
- 38 • Forward slash ("/")
- 39 • Question mark ("?")
- 40 • Colon (":")
- 41 • At-sign ("@")
- 42 • Ampersand ("&")
- 43 • Equal sign ("=")
- 44 • Plus sign ("+")

1 • US Dollar sign (" \$")

2 • Comma (",")

3 To transform the URI contained in a `System.Uri` instance from an escape encoded URI to a
4 human-readable URI, use the `System.Uri.ToString` method.

5

6]

7

8

9

10 URIs are stored as canonical URIs in escaped encoding, with all characters with ASCII
11 values greater than 127 replaced with their hexadecimal equivalents. The `System.Uri`
12 constructors do not escape URI strings if the string is a well-formed URI, including a scheme
13 identifier, that contains escape sequences. To put the URI in canonical form, the
14 `System.Uri` constructors perform the following steps.

15 • Converts the URI scheme to lowercase.

16 • Converts the host name to lowercase.

17 • Removes default and empty port numbers.

18 • Simplifies the URI by removing superfluous segments such as "/" and "/test"
19 segments.

20 The `System.Uri` class stores only absolute URIs (for example,
21 "`http://www.contoso.com/index.htm`"). Relative URIs (for example, "`/new/index.htm`") are
22 expanded to absolute form using a specified base URI. The `System.Uri.MakeRelative`
23 method converts absolute URIs to relative URIs.

24

25 The `System.Uri` class properties are read-only; to modify a `System.Uri` instance use the
26 `System.UriBuilder` class.

27

1 Uri(System.String) Constructor

```
2 [ILAsm]  
3 public rtspecialname specialname instance void .ctor(string uriString)  
4 [C#]  
5 public Uri(string uriString)
```

6 Summary

7 Constructs and initializes a new instance of the `System.Uri` class by parsing the
8 specified URI.

9 Parameters

Parameter	Description
<i>uriString</i>	A <code>System.String</code> containing a URI.

10

11 Description

12 This constructor is equivalent to calling the `System.Uri (System.String,`
13 `System.Boolean)` constructor, and specifying *uriString* and `false` as the arguments.

14 Exceptions

Exception	Condition
System.ArgumentNullException	<i>uriString</i> is null.
System.UriFormatException	<i>uriString</i> is a zero length string or contains only spaces.
	-OR- <i>uriString</i> is in an invalid form and cannot be parsed.

15

16

1 Uri(System.String, System.Boolean)

2 Constructor

```
3 [ILAsm]  
4 public rtspecialname specialname instance void .ctor(string uriString,  
5 bool dontEscape)  
  
6 [C#]  
7 public Uri(string uriString, bool dontEscape)
```

8 Summary

9 Constructs and initializes a new instance of the `System.Uri` class by parsing the
10 specified URI.

11 Parameters

Parameter	Description
<i>uriString</i>	A <code>System.String</code> containing a URI.
<i>dontEscape</i>	true if the URI in <i>uriString</i> is already escaped; otherwise, false.

12 13 Description

14 This constructor parses the URI, places its components into the appropriate properties,
15 and puts the URI in canonical form. If the specified URI does not contain a scheme
16 component, the URI is parsed using "file" as the scheme.

17 Exceptions

Exception	Condition
System.ArgumentNullException	<i>uriString</i> is null.
System.UriFormatException	<i>uriString</i> is a zero length string or contains only spaces.
	-or-
	The parsing routine detected a scheme in an invalid form.
	-or-
	The parser detected more than two consecutive slashes

in a URI that does not use the "file" scheme.

-or-

uriString is in an invalid form and cannot be parsed.

1

2 **Example**

3 The following example creates a `System.Uri` instance for the URI
4 "http://www.contoso.com/Hello%20World.htm". Because the URI contains escaped
5 characters, the third parameter, *dontEscape*, is set to `true`.

6

7 [C#]

8 `using System;`

9

10 `public class UriTest {`

11 `public static void Main() {`

12

13 `Uri myUri = new Uri("http://www.contoso.com/Hello%20World.htm", true);`

14

15 `Console.WriteLine(myUri.ToString());`

16 `}`

17 `}`

18 The output is

19

20 `http://www.contoso.com/Hello World.htm`

21

22

1 Uri(System.Uri, System.String) Constructor

```
2 [ILAsm]  
3 public rtspecialname specialname instance void .ctor(class System.Uri  
4 baseUri, string relativeUri)  
  
5 [C#]  
6 public Uri(Uri baseUri, string relativeUri)
```

7 Summary

8 Constructs and initializes a new instance of the `System.Uri` class by combining the
9 specified base and relative URIs.

10 Parameters

Parameter	Description
<i>baseUri</i>	A <code>System.Uri</code> containing a base URI.
<i>relativeUri</i>	A <code>System.String</code> containing a relative URI.

11

12 Description

13 This constructor is equivalent to calling the `System.Uri (System.Uri, System.String,
14 System.Boolean)` constructor, and specifying *baseUri*, *relativeUri*, and `false` as the
15 arguments.

16 Exceptions

Exception	Condition
System.UriFormatException	<i>relativeUri</i> is in an invalid form.
System.NullReferenceException	<i>baseUri</i> is null.

17

18

Uri(System.Uri, System.String, System.Boolean) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(class System.Uri
baseUri, string relativeUri, bool dontEscape)

[C#]
public Uri(Uri baseUri, string relativeUri, bool dontEscape)
```

Summary

Constructs and initializes a new instance of the `System.Uri` class by combining the specified base and relative URIs.

Parameters

Parameter	Description
<i>baseUri</i>	A <code>System.Uri</code> containing the base URI. This parameter can, but is not required to contain a terminating slash ("/") character.
<i>relativeUri</i>	A <code>System.String</code> containing the relative URI to add to the base URI. This parameter can, but is not required to contain a leading slash ("/") character.
<i>dontEscape</i>	true if <i>baseUri</i> and <i>relativeUri</i> are already escaped; otherwise, false.

Description

This constructor compensates for the presence or absence of a terminating slash in *baseUri* and/or a leading slash in *relativeUri* to produce a well-formed URI.

If the relative URI contains a `System.Uri.Scheme` that is the same as the scheme of the base URI and the `System.Uri.SchemeDelimiter` is not present, or the relative URI does not contain a scheme, the new instance is composed of the relative URI (without its scheme component, if any) qualified by the scheme and authority information from the base URI.

If the relative URI contains a `System.Uri.Scheme` followed by the `System.Uri.SchemeDelimiter`, it is treated as an absolute URI and the base URI is ignored. If the relative URI contains a scheme that differs from the scheme of the base URI, the base URI is ignored. If the `System.Uri.SchemeDelimiter` is not present in the relative URI, it is assumed, and the new instance is constructed as though the relative URI were an absolute URI.

[*Note:* When the base URI is ignored, only the components of the relative URI are used to construct the new instance.]

1
2

3 Exceptions

Exception	Condition
System.UriFormatException	<i>relativeUri</i> is in an invalid form.
System.NullReferenceException	<i>baseUri</i> is null.

4
5

Example

6 The following example creates new instances of the `System.Uri` class by combining a
7 `System.Uri` instance representing the base URI and a string containing a relative URI.

8
9

[C#]

```
10 using System;
11
12 public class UriTest {
13     public static void Main() {
14
15         // Typical base and relative URI constructor usage.
16
17         Uri baseUri = new Uri("http://www.contoso.com", true);
18         Uri myUri = new Uri(baseUri, "index.htm",true);
19         Console.WriteLine("Typical usage: {0}",myUri.ToString());
20
21         // Base and relative URI contain slashes.
22         Uri baseUri2 = new Uri("http://www.contoso.com/", true);
23         Uri myUri2 = new Uri(baseUri2, "/index.htm",true);
24         Console.WriteLine("Slash example: {0}",myUri2.ToString());
25
26         // Relative URI contains a different scheme than the base URI.
27         Uri baseUri3 = new Uri("http://www.contoso.com/", true);
28         Uri myUri3 = new Uri(baseUri3, "ftp://www.contoso2.com/index.htm",true);
29         Console.WriteLine("Different schemes: {0}", myUri3.ToString());
30
31
32         // Relative URI contains the same scheme as the base URI.
33         // The scheme delimiter is not present in the relative URI.
34         Uri baseUri4 = new Uri("http://www.contoso.com/", true);
35         Uri myUri4 = new Uri(baseUri4, "http:www.contoso2.com/index.htm",true);
36         Console.WriteLine("Same schemes - relative treated as relative:
37 {0}",myUri4.ToString());
38
39         // Relative URI contains the same scheme as the base URI.
40         // The scheme delimiter is present in the relative URI.
41         Uri baseUri5 = new Uri("http://www.contoso.com/", true);
42         Uri myUri5 = new Uri(baseUri5, "http://www.contoso2/index.htm",true);
```

```
1 Console.WriteLine("Same schemes - relative treated as absolute:
2 {0}",myUri5.ToString());
3
4 }
5 }
6
7 The output is
8
9 Typical usage: http://www.contoso.com/index.htm
10
11
12 Slash example: http://www.contoso.com/index.htm
13
14
15 Different schemes: ftp://www.contoso2.com/index.htm
16
17
18 Same schemes - relative treated as relative:
19 http://www.contoso.com/www.contoso2.com/index.htm
20
21
22 Same schemes - relative treated as absolute: http://www.contoso2/index.htm
23
24
```

1 Uri.SchemeDelimiter Field

```
2 [ILAsm]  
3 .field public static initOnly string SchemeDelimiter  
4 [C#]  
5 public static readonly string SchemeDelimiter
```

6 Summary

7 A `System.String` containing the characters that separate the scheme component from
8 the remainder of a URI.

9 Description

10 This field is read-only. The value of this field is "://".

11

1 Uri.UriSchemeFile Field

```
2 [ILAsm]  
3 .field public static initOnly string UriSchemeFile  
4 [C#]  
5 public static readonly string UriSchemeFile
```

6 Summary

7 A `System.String` containing the characters that indicate that a URI identifies a file.

8 Description

9 This field is read-only. The value of this field is "file".

10

1 Uri.UriSchemeFtp Field

```
2 [ILAsm]  
3 .field public static initOnly string UriSchemeFtp  
4 [C#]  
5 public static readonly string UriSchemeFtp
```

6 Summary

7 A `System.String` containing the characters that indicate that a URI is accessed through
8 the File Transfer Protocol (FTP).

9 Description

10 This field is read-only. The value of this field is "ftp".

11

1 Uri.UriSchemeGopher Field

```
2 [ILAsm]  
3 .field public static initOnly string UriSchemeGopher  
4 [C#]  
5 public static readonly string UriSchemeGopher
```

6 Summary

7 A `System.String` containing the characters that indicate that a URI is accessed through
8 the Gopher protocol.

9 Description

10 This field is read-only. The value of this field is "gopher".

11

1 Uri.UriSchemeHttp Field

```
2 [ILAsm]  
3 .field public static initOnly string UriSchemeHttp  
4 [C#]  
5 public static readonly string UriSchemeHttp
```

6 Summary

7 A `System.String` containing the characters that indicate that a URI is accessed through
8 the Hypertext Transfer Protocol (HTTP).

9 Description

10 This field is read-only. The value of this field is "http".

11

1 Uri.UriSchemeHttps Field

```
2 [ILAsm]  
3 .field public static initOnly string UriSchemeHttps  
4 [C#]  
5 public static readonly string UriSchemeHttps
```

6 Summary

7 A `System.String` containing the characters that indicate that a URI is accessed through
8 the Secure Hypertext Transfer Protocol (HTTPS).

9 Description

10 This field is read-only. The value of this field is "https".

11

1 Uri.UriSchemeMailto Field

```
2 [ILAsm]  
3 .field public static initOnly string UriSchemeMailto  
4 [C#]  
5 public static readonly string UriSchemeMailto
```

6 Summary

7 A `System.String` containing the characters that indicate that a URI is an email address
8 and is accessed through the Simple Network Mail Protocol (SNMP).

9 Description

10 This field is read-only. The value of this field is "mailto".

11

1 Uri.UriSchemeNews Field

```
2 [ILAsm]  
3 .field public static initOnly string UriSchemeNews  
4 [C#]  
5 public static readonly string UriSchemeNews
```

6 Summary

7 A `System.String` containing the characters that indicate that a URI is an Internet news
8 group and is accessed through the Network News Transport Protocol (NNTP).

9 Description

10 This field is read-only. The value of this field is "news".

11

1 Uri.UriSchemeNntp Field

```
2 [ILAsm]  
3 .field public static initOnly string UriSchemeNntp  
4 [C#]  
5 public static readonly string UriSchemeNntp
```

6 Summary

7 A `System.String` containing the characters that indicate that a URI is an Internet news
8 group and is accessed through the Network News Transport Protocol (NNTP).

9 Description

10 This field is read-only. The value of this field is "nntp".

11

1 Uri.Canonicalize() Method

```
2 [ILAsm]  
3 .method family hidebysig virtual void Canonicalize()  
4 [C#]  
5 protected virtual void Canonicalize()
```

6 Summary

7 Converts the components of the URI represented by the current instance to canonical
8 form.

9 Behaviors

10 This method converts the URI to a format suitable for machine interpretation according
11 to the scheme of the current instance. The conversions are required to preserve all
12 information that could, if removed or altered, change the URI represented by the current
13 instance.

15 Default

16 This method performs the following conversions:

- 17 • Converts file references to the format of the current platform, for example on a
18 Windows system, file://c:/AFile.txt is converted to "file:///c:/AFile.txt".
- 19 • Converts any backslash characters ('\') to forward slashes ('/').
- 20 • Compresses multiple consecutive forward slashes ('/') in the path component to a
21 single forward slash.
- 22 • Compresses any path meta sequences ("/.\" and "/..").

23 How and When to Override

24 Override this method to canonicalize the type derived from `System.Uri`.

26 Usage

27 Applications do not call this method; it is called by constructors after parsing the URI
28 and escaping the components.

Uri.CheckHostName(System.String) Method

```
[ILAsm]
.method public hidebysig static valuetype System.UriHostNameType
CheckHostName(string name)

[C#]
public static UriHostNameType CheckHostName(string name)
```

Summary

Returns a value that describes the format of a host name string.

Parameters

Parameter	Description
<i>name</i>	A System.String containing the host name to validate.

Return Value

A System.UriHostNameType that indicates the type of the host name. If the type of the host name cannot be determined, or the host name is null or a zero-length string, returns System.UriHostNameType.Unknown.

Example

The following example demonstrates using the System.Uri.CheckHostName method.

```
[C#]
using System;

public class UriTest {
    public static void Main() {
        Console.WriteLine(Uri.CheckHostName("www.contoso.com"));
    }
}
```

The output is

Dns

1 Uri.CheckSchemeName(System.String)

2 Method

```
3 [ILAsm]  
4 .method public hidebysig static bool CheckSchemeName(string schemeName)  
5 [C#]  
6 public static bool CheckSchemeName(string schemeName)
```

7 Summary

8 Returns a `System.Boolean` value indicating whether the specified scheme name is valid.

9 Parameters

Parameter	Description
<i>schemeName</i>	A <code>System.String</code> containing the scheme name to validate.

10

11 Return Value

12 `true` if the scheme name is valid; otherwise, `false`. If *schemeName* is null or is a
13 zero-length string, returns `false`.

14 Description

15 [*Note:* The scheme name is required to begin with a letter, and contain only letters,
16 digits, and the characters '.', '+' or '-'.]
17
18

19

1 Uri.CheckSecurity() Method

```
2 [ILAsm]  
3 .method family hidebysig virtual void CheckSecurity()  
4 [C#]  
5 protected virtual void CheckSecurity()
```

6 Summary

7 Checks the current instance for character sequences that can result in unauthorized
8 access to resources, and removes them.

9 Behaviors

10 This method checks for invalid or dangerous character sequences in the components of
11 the current instance, and removes them. The semantics that determine whether a
12 character sequence presents a security risk are determined by the scheme of the
13 current instance.

14

15 Default

16 The default implementation does nothing.

17

18 How and When to Override

19 Override this method to provide security checks for types derived from `System.Uri`.

20

21 Usage

22 Invoke this method on instances of types derived from `System.Uri` to remove any URI
23 content that allows unauthorized access to resources.

24

25

Uri.Equals(System.Object) Method

```
[ILAsm]  
.method public hidebysig virtual bool Equals(object comparand)  
  
[C#]  
public override bool Equals(object comparand)
```

Summary

Compares the current instance and the specified object for equality.

Parameters

Parameter	Description
<i>comparand</i>	The <code>System.Uri</code> instance to compare with the current instance. This argument can be a <code>System.String</code> or a <code>System.Uri</code> .

Return Value

`true` if *comparand* represents the same URI (ignoring any fragment or query information) as the current instance; otherwise, `false`. If *comparand* is null, a zero-length string, or is not an instance of `System.String` or `System.Uri`, returns `false`.

Description

If *comparand* is a `System.String`, it is converted to a `System.Uri` by calling `System.Uri(comparand)`.

The `System.Uri.Scheme`, `System.Uri.Host` and unescaped version of the `System.Uri.AbsolutePath` of the current instance and *comparand* are compared for equality.

If the scheme of the current instance is the `System.Uri.UriSchemeFile` scheme, the absolute paths are compared in accordance with the case sensitivity of the current platform.

[*Note:* This method overrides `System.Object.Equals`.]

1 Uri.Escape() Method

```
2 [ILAsm]  
3 .method family hidebysig virtual void Escape()  
4 [C#]  
5 protected virtual void Escape()
```

6 Summary

7 Converts any unsafe or reserved characters in the `System.Uri.AbsolutePath`
8 component to equivalent escaped hexadecimal sequences.

9 Description

10 Behaviors

11 Converts any unsafe or reserved characters in the `System.Uri.AbsolutePath`
12 component to a character sequence consisting of a "%" followed by the hexadecimal
13 value of the character as described by IETF 2396.
14

15 If the path component of the current instance is `null`, the escaped path is
16 `System.String.Empty`.

17 Default

18 As described above.
19

20 How and When to Override

21 Override this method to customize the escaping behavior provided by the `System.Uri`
22 type.
23

24 Usage

25 Applications typically do not call this method; it is intended for use by the constructors.
26
27

28 [*Note:* For additional information on escaping URI, see section 2 of RFC 2396.]
29
30
31

1 Uri.EscapeString(System.String) Method

```
2 [ILAsm]  
3 .method family hidebysig static string EscapeString(string str)  
4 [C#]  
5 protected static string EscapeString(string str)
```

6 Summary

7 Converts a string to its escaped representation.

8 Parameters

Parameter	Description
<i>str</i>	A <code>System.String</code> to convert to its escaped representation.

9 10 Return Value

11 A `System.String` containing the escaped representation of *str*.

12 Description

13 The string is escaped in accordance with RFC 2396.

14

1 Uri.FromHex(System.Char) Method

```
2 [ILAsm]  
3 .method public hidebysig static int32 FromHex(valuetype System.Char digit)  
4 [C#]  
5 public static int FromHex(char digit)
```

6 Summary

7 Returns the decimal value of a hexadecimal digit.

8 Parameters

Parameter	Description
<i>digit</i>	The hexadecimal digit (0-9, a-f, A-F) to convert.

9 Return Value

11 A `System.Int32` containing an integer from 0 - 15 that corresponds to the specified
12 hexadecimal digit.

13 Exceptions

Exception	Condition
System.ArgumentException	<i>digit</i> is not a valid hexadecimal digit (0-9, a-f, A-F).

14
15

1 Uri.GetHashCode() Method

```
2 [ILAsm]  
3 .method public hidebysig virtual int32 GetHashCode()  
4 [C#]  
5 public override int GetHashCode()
```

6 Summary

7 Generates a hash code for the current instance.

8 Return Value

9 A `System.Int32` containing the hash code for this instance.

10 Description

11 The hash code is generated without the fragment component. For example, the URIs
12 "http://www.contoso.com/index.htm#search" and "http://www.contoso.com/index.htm"
13 produce the same hash code.

14 The algorithm used to generate the hash code is unspecified.

15 [Note: This method overrides `System.Object.GetHashCode`.]
16
17
18
19

20

Uri.GetLeftPart(System.UriPartial) Method

```
[ILAsm]
.method public hidebysig instance string GetLeftPart(valuetype
System.UriPartial part)

[C#]
public string GetLeftPart(UriPartial part)
```

Summary

Returns the specified portion of the URI represented by the current instance.

Parameters

Parameter	Description
<i>part</i>	A <code>System.UriPartial</code> value that specifies the component to return.

Return Value

A `System.String` containing all components up to the specified portion of the URI, or `System.String.Empty` if the current instance does not contain the component identified by *part*.

Description

The `System.Uri.GetLeftPart` method returns a string containing the URI components starting with the left-most component of the URI and ending with the component specified by *part*. The returned string does not include fragment or query information.

`System.Uri.GetLeftPart` includes delimiters as follows:

- `System.UriPartial.Scheme` has the scheme delimiter added.
- `System.UriPartial.Authority` does not have the path delimiter added.
- `System.UriPartial.Path` includes any delimiters in the original URI up to the query or fragment delimiter.

Exceptions

Exception	Condition
System.ArgumentException	The <i>part</i> parameter is not a valid <code>System.UriPartial</code> value.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

Example

The following example demonstrates the `System.Uri.GetLeftPart` method.

[C#]

```
using System;

public class UriTest {
    public static void Main() {
        string[] myUri = {
            "http://www.contoso.com/index.htm",
            "http://www.contoso.com/index.htm#mark",
            "mailto:user@contoso.com?subject=uri",
            "nntp://news.contoso.com/123456@contoso.com"
        };
        foreach (string s in myUri) {
            Uri aUri = new Uri(s);
            Console.WriteLine("URI: {0}", aUri.ToString());
            Console.WriteLine("Scheme: {0}", aUri.GetLeftPart(UriPartial.Scheme));
            Console.WriteLine("Authority:
{0}", aUri.GetLeftPart(UriPartial.Authority));
            Console.WriteLine("Path: {0}", aUri.GetLeftPart(UriPartial.Path));
        }
    }
}
```

The output is

```
URI: http://www.contoso.com/index.htm
Scheme: http://
Authority: http://www.contoso.com
Path: http://www.contoso.com/index.htm

URI: http://www.contoso.com/index.htm#mark
Scheme: http://
Authority: http://www.contoso.com
Path: http://www.contoso.com/index.htm
```

1
2 URI: mailto:user@contoso.com?subject=uri
3
4
5 Scheme: mailto:
6
7
8 Authority:
9
10
11 Path: mailto:user@contoso.com
12
13
14 URI: nntp://news.contoso.com/123456@contoso.com
15
16
17 Scheme: nntp://
18
19
20 Authority: nntp://news.contoso.com
21
22
23 Path: nntp://news.contoso.com/123456@contoso.com
24
25

1 Uri.HexEscape(System.Char) Method

```
2 [ILAsm]  
3 .method public hidebysig static string HexEscape(valuetype System.Char  
4 character)  
  
5 [C#]  
6 public static string HexEscape(char character)
```

7 Summary

8 Converts a specified ASCII character into its escaped hexadecimal equivalent.

9 Parameters

Parameter	Description
<i>character</i>	A <code>System.Char</code> containing the character to convert to escaped hexadecimal representation.

10

11 Return Value

12 A `System.String` containing the escaped hexadecimal representation of the specified
13 character.

14 Description

15 The returned string is in the form "%XX", where X represents a hexadecimal digit (0-9,
16 A-F).

17 Exceptions

Exception	Condition
<code>System.ArgumentOutOfRangeException</code>	The numerical value of <i>character</i> is greater than 255.

18

19

1 Uri.HexUnescape(System.String, 2 System.Int32&) Method

```
3 [ILAsm]  
4 .method public hidebysig static valuetype System.Char HexUnescape(string  
5 pattern, int32& index)  
  
6 [C#]  
7 public static char HexUnescape(string pattern, ref int index)
```

8 Summary

9 Converts a specified escaped hexadecimal representation of a character to the
10 character.

11 Parameters

Parameter	Description
<i>pattern</i>	A System.String containing the hexadecimal representation of a character.
<i>index</i>	A System.Int32 containing the location in <i>pattern</i> where the hexadecimal representation of a character begins.

12

13 Return Value

14 A System.Char containing a character. If the character pointed to by *index* is a "%" and
15 there are at least two characters following the "%", and the two characters are valid
16 hexadecimal digits, the hexadecimal digits are converted to System.Char. Otherwise,
17 the character at *index* is returned. Valid hexadecimal digits are: 0-9, a-f, A-F.

18

19 On return, the value of *index* contains the index of the character following the one
20 returned.

21 Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	<i>index</i> < 0, or <i>index</i> >= the number of characters in <i>pattern</i> .

22

23

Uri.IsBadFileSystemCharacter(System.Char) Method

```
[ILAsm]  
.method family hidebysig virtual bool IsBadFileSystemCharacter(valuetype  
System.Char character)  
  
[C#]  
protected virtual bool IsBadFileSystemCharacter(char character)
```

Summary

Returns a `System.Boolean` value that indicates whether the specified character would be an invalid character if used in a file system name.

Parameters

Parameter	Description
<i>character</i>	A <code>System.Char</code> containing the character to check.

Return Value

`true` if the specified character is not acceptable for use in a file system name; otherwise, `false`.

The value returned by this method is implementation-specific.

Behaviors

This method returns `false` if the specified character cannot be used in a URI that identifies a file, as defined by the current file system on the current platform.

Default

As described above.

How and When to Override

Override this method to provide a check for invalid characters as defined by the current file system on the current platform.

1 **Usage**

2 Use this method to determine if a character can be used in a file name.

3

4

1 Uri.IsExcludedCharacter(System.Char)

2 Method

```
3 [ILAsm]  
4 .method family hidebysig static bool IsExcludedCharacter(valuetype  
5 System.Char character)  
  
6 [C#]  
7 protected static bool IsExcludedCharacter(char character)
```

8 Summary

9 Returns a `System.Boolean` value that indicates whether the specified character is
10 excluded from use or is unwise in URIs, as defined by IETF RFC 2396.

11 Parameters

Parameter	Description
<i>character</i>	A <code>System.Char</code> containing the character to check.

12 Return Value

14 `true` if the specified character is required to be escaped; otherwise, `false`.

15 Description

16 This method returns `true` for the following characters:

Character(s)	Description
<i>character</i> < 0x0020	Any character with the ASCII value less than hexadecimal 0x20 (32).
<i>character</i> < 0x007f	Any character with the ASCII value greater than hexadecimal 0x7f (127).
<	Less than sign.
>	Greater than sign.
#	Number sign (crosshatch, pound sign).
%	Percent.

"	Quotation mark.
{	Left curly brace.
}	Right curly brace.
	Pipe sign (vertical bar).
\	Backward slash.
^	Circumflex (caret).
[Left square bracket.
]	Right square bracket.
`	Grave accent.

1

2

1 Uri.IsHexDigit(System.Char) Method

```
2 [ILAsm]  
3 .method public hidebysig static bool IsHexDigit(valuetype System.Char  
4 character)  
  
5 [C#]  
6 public static bool IsHexDigit(char character)
```

7 Summary

8 Returns a `System.Boolean` value that indicates whether the specified character is a valid
9 hexadecimal digit.

10 Parameters

Parameter	Description
<i>character</i>	A <code>System.Char</code> containing the character to validate.

11 12 Return Value

13 `true` if the character is a valid hexadecimal digit (0-9, A-F, a-f); otherwise `false`.

14

1 Uri.IsHexEncoding(System.String, 2 System.Int32) Method

```
3 [ILAsm]  
4 .method public hidebysig static bool IsHexEncoding(string pattern, int32  
5 index)  
  
6 [C#]  
7 public static bool IsHexEncoding(string pattern, int index)
```

8 Summary

9 Returns a `System.Boolean` value that indicates whether a substring of the specified
10 string is in escaped hexadecimal encoding format ("% followed by two hexadecimal
11 characters).

12 Parameters

Parameter	Description
<i>pattern</i>	The <code>System.String</code> to check.
<i>index</i>	A <code>System.Int32</code> containing the location in <i>pattern</i> to check for hex encoding.

13 14 Return Value

15 `true` if the specified location in *pattern* contains a substring in escaped hexadecimal
16 encoding format; otherwise, `false`.

17 Description

18 The `System.Uri.IsHexEncoding` method checks for hexadecimal digits case-
19 insensitively.

20

1 Uri.IsReservedCharacter(System.Char)

2 Method

```
3 [ILAsm]  
4 .method family hidebysig virtual bool IsReservedCharacter(valuetype  
5 System.Char character)  
  
6 [C#]  
7 protected virtual bool IsReservedCharacter(char character)
```

8 Summary

9 Returns a `System.Boolean` value that indicates whether a character is part of the URI
10 reserved set.

11 Return Value

12 `true` if *character* is a URI reserved character as defined by IETF RFC 2396; otherwise,
13 `false`.

14 Description

15 The following characters are reserved for the use in URI:

Character	Description
;	Semi-colon.
/	Forward slash.
:	Colon.
@	At sign (commercial at).
&	Ampersand.
=	Equals sign.
+	Plus sign.
\$	US Dollar sign.
,	Comma.

1 **Behaviors**

2 As described above.

3

4 **How and When to Override**

5 Override this method to customize the escaping behavior provided by the `System.Uri`
6 type.

7

8 **Usage**

9 Use this method to determine if a character is reserved.

10

11

Uri.MakeRelative(System.Uri) Method

```
[ILAsm]  
.method public hidebysig instance string MakeRelative(class System.Uri  
toUri)  
  
[C#]  
public string MakeRelative(Uri toUri)
```

Summary

Returns the specified `System.Uri` as a relative URI.

Parameters

Parameter	Description
<i>toUri</i>	The URI to compare to the current URI.

Return Value

A `System.String` with the difference between the current instance and *toUri* if the two URIs are the same except for the path information. If the two URIs differ in more than the `System.Uri.AbsolutePath`, this method returns the `System.String` representation of *toUri*.

Example

The following example demonstrates the `System.Uri.MakeRelative` method.

```
[C#]  
  
using System;  
public class UriTest {  
    public static void Main() {  
        Uri myUri = new Uri("http://www.contoso.com/Hello%20World.htm", true);  
        Console.WriteLine(myUri.ToString());  
        Console.WriteLine(myUri.MakeRelative(new Uri  
("http://www.contoso.com/index.htm")));  
    }  
}
```

The output is

```
http://www.contoso.com/Hello World.htm
```

```
index.htm
```

1 Uri.Parse() Method

```
2 [ILAsm]  
3 .method family hidebysig virtual void Parse()  
4 [C#]  
5 protected virtual void Parse()
```

6 Summary

7 Parses the URI into its constituent components.

8 Behaviors

9 This method parses the `System.Uri.AbsolutePath` property, separates it into various
10 URI components, and stores the components in the appropriate `System.Uri` properties.

11

12 Default

13 This method parses path components as defined in IETF RFC 2396.

14

15 How and When to Override

16 Override this method to provide parsing for URIs in formats that are not defined in IETF
17 RFC 2396.

18

19 Usage

20 Applications typically do not call this method; it is intended for use by the constructors.

21

22 Exceptions

Exception	Condition
System.UriFormatException	The scheme of the URI is in an invalid format. -or-

	The URI is in an invalid form and cannot be parsed.
--	---

1

2

1 Uri.ToString() Method

```
2 [ILAsm]  
3 .method public hidebysig virtual string ToString()  
4 [C#]  
5 public override string ToString()
```

6 Summary

7 Returns the unescaped, canonical form of the URI information used to construct the
8 current instance.

9 Return Value

10 A `System.String` containing the unescaped, canonical form of the URI represented by
11 the current instance.

12 Description

13 The string returned by this method includes the `System.Uri.Query` and
14 `System.Uri.Fragment` components.

15
16 [*Note:* This method overrides `System.Object.ToString.`]
17
18

19

1 Uri.Unescape(System.String) Method

```
2 [ILAsm]  
3 .method family hidebysig virtual string Unescape(string path)  
4 [C#]  
5 protected virtual string Unescape(string path)
```

6 Summary

7 Converts escape sequences in the specified `System.String` into their unescaped
8 equivalents.

9 Parameters

Parameter	Description
<i>path</i>	The <code>System.String</code> to unescape.

10

11 Return Value

12 A `System.String` containing *path* with its escaped characters converted to their
13 unescaped equivalents. If *path* is `null` or a zero-length string, returns
14 `System.String.Empty`.

15 Description

16 [*Note:* Escape sequences can be hex-encoded reserved characters (for example "%40")
17 or hex-encoded UTF-8 sequences (for example "%C4%D2").

18]
19]

20

1 Uri.AbsolutePath Property

```
2 [ILAsm]
3 .property string AbsolutePath { public hidebysig specialname instance
4 string get_AbsolutePath() }
5
6 [C#]
7 public string AbsolutePath { get; }
```

7 Summary

8 Gets the absolute path of the resource identified by the current instance.

9 Property Value

10 A System.String containing the absolute path to the resource.

11 Description

12 This property is read-only.

13
14 The System.Uri.AbsolutePath property contains the path to the resource identified by
15 the current instance. The System.Uri.AbsolutePath property always returns at least a
16 slash ('/').

17
18 If, when the current instance was constructed, the URI was already escaped or the
19 constructor's *dontEscape* parameter was set to *false*, the value returned by this
20 property is escaped.

21
22 [*Note:* The path information does not include the scheme, host name, query, or
23 fragment components of the URI.]

24
25

26 Example

27 The following example outputs the absolute path of a URI.

```
28 [C#]
29
30 using System;
31
32 public class UriTest {
33     public static void Main() {
34         Uri myUri = new Uri
35         ("http://www.contoso.com/URI/Hello%20World.htm?date=today", true);
36         Console.WriteLine(myUri.AbsolutePath);
37     }
38 }
39
```

40 The output is

41

1 /URI/Hello%20World.htm

2

3

1 Uri.AbsoluteUri Property

```
2 [ILAsm]  
3 .property string AbsoluteUri { public hidebysig specialname instance  
4 string get_AbsoluteUri() }  
  
5 [C#]  
6 public string AbsoluteUri { get; }
```

7 Summary

8 Gets the absolute URI of the resource identified by the current instance in canonical
9 form.

10 Property Value

11 A `System.String` containing the URI used to construct the current instance, in canonical
12 format.

13 Description

14 This property is read-only.

15
16 The `System.Uri.AbsoluteUri` property includes the entire URI stored in the current
17 `System.Uri` instance, including any fragment or query information. If, when the current
18 instance was constructed, the URI was already escaped or the constructor's *dontEscape*
19 parameter was set to *false*, the value returned by this property is escaped.

20

1 Uri.Authority Property

```
2 [ILAsm]  
3 .property string Authority { public hidebysig specialname instance string  
4 get_Authority() }  
5 [C#]  
6 public string Authority { get; }
```

7 Summary

8 Gets the authority component of the URI used to construct the current instance.

9 Property Value

10 A `System.String` containing the authority component of the current instance. The value
11 returned by this property is composed of the values returned by the `System.Uri.Host`
12 and `System.Uri.Port` properties.

13 Description

14 This property is read-only.

15
16 The `System.Uri.Authority` property returns the `System.Uri.Host` and
17 `System.Uri.Port` information specified in the URI used to construct the current
18 instance. The value of this property includes the port information only if the URI
19 specified a port that is not the default for the current scheme. When port information is
20 included in the value returned by this property, the host and port are separated by a
21 colon (":").

22

1 Uri.Fragment Property

```
2 [ILAsm]
3 .property string Fragment { public hidebysig specialname instance string
4 get_Fragment() }
5
6 [C#]
7 public string Fragment { get; }
```

7 Summary

8 Gets the fragment component of the URI used to construct the current instance.

9 Property Value

10 A `System.String` containing any fragment information contained in the URI used to
11 construct the current instance.

12 Description

13 This property is read-only.

14
15 The `System.Uri.Fragment` property gets any text following a fragment marker ('#') in
16 the URI, including the fragment marker itself. If, when the current instance was
17 constructed, the URI was already escaped or the constructor's *dontEscape* parameter
18 was set to *false*, the value returned by this property is escaped.

19
20 [*Note:* The `System.Uri.Fragment` property is not considered in a `System.Uri.Equals`
21 comparison.

22
23]

24 Example

25 The following example demonstrates the use of the `System.Uri.Fragment` property.

```
26 [C#]
27
28 using System;
29
30 public class UriTest {
31     public static void Main() {
32
33         Uri baseUri = new Uri("http://www.contoso.com/");
34         Uri myUri = new Uri(baseUri, "index.htm#main");
35
36         Console.WriteLine(myUri.Fragment);
37     }
38 }
```

39 The output is

40

```
1 #main
2
3
```

1 Uri.Host Property

```
2 [ILAsm]
3 .property string Host { public hidebysig specialname instance string
4 get_Host() }

5 [C#]
6 public string Host { get; }
```

7 Summary

8 Gets the host component of the URI used to construct the current instance.

9 Property Value

10 A `System.String` containing the DNS host name or IP address of the host server. If the
11 host information was not specified to the constructor, the value of this property is
12 `System.String.Empty`.

13 Description

14 This property is read-only.

15
16 If the host information is an IP6 address, the information is enclosed in square brackets
17 ("[" and "]").

18 Example

19 The following example demonstrates using the `System.Uri.Host` property.

```
20 [C#]
21
22 using System;
23
24 public class UriTest {
25     public static void Main() {
26
27         Uri baseUri = new Uri("http://www.contoso.com:8080/");
28         Uri myUri = new Uri(baseUri, "shownew.htm?date=today");
29
30         Console.WriteLine(myUri.Host);
31     }
32 }
```

33 The output is

```
34
35 www.contoso.com
36
```

37

1 Uri.HostNameType Property

```
2 [ILAsm]  
3 .property valuetype System.UriHostNameType HostNameType { public hidebysig  
4 specialname instance valuetype System.UriHostNameType get_HostNameType() }  
  
5 [C#]  
6 public UriHostNameType HostNameType { get; }
```

7 Summary

8 Gets the format of the host address in the URI used to construct the current instance.

9 Property Value

10 A System.UriHostNameType that indicates the format of the host address information in
11 the current instance.

12 Description

13 This property is read-only.

14

15 If System.Uri.Host is null, the value of this property is
16 System.UriHostNameType.Unknown.

17

1 Uri.IsDefaultPort Property

```
2 [ILAsm]  
3 .property bool IsDefaultPort { public hidebysig specialname instance bool  
4 get_IsDefaultPort() }  
  
5 [C#]  
6 public bool IsDefaultPort { get; }
```

7 Summary

8 Gets a `System.Boolean` value indicating whether the `System.Uri.Port` value of the
9 current instance is the default port for the scheme of the current instance.

10 Property Value

11 `true` if the value in the `System.Uri.Port` property is the default port for the
12 `System.Uri.Scheme`; otherwise, `false`.

13 Description

14 This property is read-only.

15
16 [*Note:* For a list of default port values, see the `System.Uri.Port` property.]
17
18

19

1 Uri.IsFile Property

```
2 [ILAsm]  
3 .property bool IsFile { public hidebysig specialname instance bool  
4 get_IsFile() }  
  
5 [C#]  
6 public bool IsFile { get; }
```

7 Summary

8 Gets a `System.Boolean` value indicating whether the current instance identifies a file.

9 Property Value

10 `true` if the resource identified by the current `System.Uri` is a file; otherwise, `false`.

11 Description

12 This property is read-only.

13

14 The `System.Uri.IsFile` property is true when the `System.Uri.Scheme` property equals
15 `System.Uri.UriSchemeFile`.

16

1 Uri.IsLoopback Property

```
2 [ILAsm]
3 .property bool IsLoopback { public hidebysig specialname instance bool
4 get_IsLoopback() }
5
6 [C#]
7 public bool IsLoopback { get; }
```

7 Summary

8 Gets a `System.Boolean` value indicating whether the host information of the current
9 instance is the current computer.

10 Property Value

11 `true` if the host of the current instance is the reserved hostname "localhost" or the
12 loop-back IP address (127.0.0.1); otherwise, `false`.

13 Description

14 This property is read-only.

15 Example

16 The following example demonstrates the `System.Uri.IsLoopback` property.

```
17 [C#]
18
19 using System;
20
21 public class UriTest {
22     public static void Main() {
23         Uri myUri = new Uri("http://127.0.0.1/index.htm", true);
24         Console.WriteLine("{0} is loopback? {1}", myUri.ToString(),
25 myUri.IsLoopback);
26
27         myUri = new Uri("http://localhost/index.htm", true);
28         Console.WriteLine("{0} is loopback? {1}", myUri.ToString(),
29 myUri.IsLoopback);
30     }
31 }
32 }
```

33 The output is

34
35 http://127.0.0.1/index.htm is loopback? True

36

37 http://localhost/index.htm is loopback? True

38

1 Uri.LocalPath Property

```
2 [ILAsm]  
3 .property string LocalPath { public hidebysig specialname instance string  
4 get_LocalPath() }  
  
5 [C#]  
6 public string LocalPath { get; }
```

7 Summary

8 Gets the local operating-system representation of the resource identified by the current
9 instance.

10 Property Value

11 A `System.String` containing the local representation of the resource identified by the
12 current instance.

13 Description

14 This property is read-only.

15
16 If the `System.Uri.Scheme` of the current instance is not equal to
17 `System.Uri.UriSchemeFile`, this property returns the same value as
18 `System.Uri.AbsolutePath`.

19
20 If the scheme is equal to `System.Uri.UriSchemeFile`, this property returns an
21 unescaped platform-dependent local representation of the file name.

22

1 Uri.PathAndQuery Property

```
2 [ILAsm]
3 .property string PathAndQuery { public hidebysig specialname instance
4 string get_PathAndQuery() }
5
6 [C#]
7 public string PathAndQuery { get; }
```

7 Summary

8 Gets the System.Uri.AbsolutePath and System.Uri.Query components of the URI
9 used to construct the current instance.

10 Property Value

11 A System.String that contains the values of the System.Uri.AbsolutePath and
12 System.Uri.Query properties.

13 Description

14 This property is read-only.

15 Example

16 The following example uses the System.Uri.PathAndQuery property to extract the path
17 and query information from a System.Uri instance.

```
18
19 [C#]
20 using System;
21
22 public class UriTest {
23     public static void Main() {
24
25         Uri baseUri = new Uri("http://www.contoso.com/");
26         Uri myUri = new Uri(baseUri, "catalog/shownew.htm?date=today");
27
28         Console.WriteLine(myUri.PathAndQuery);
29     }
30 }
```

31 The output is

```
32
33 /catalog/shownew.htm?date=today
34
```

35

1 Uri.Port Property

```
2 [ILAsm]  
3 .property int32 Port { public hidebysig specialname instance int32  
4 get_Port() }  
5 [C#]  
6 public int Port { get; }
```

7 Summary

8 Gets the port number used to connect to the `System.Uri.Host` referenced by the
9 current instance.

10 Property Value

11 A `System.Int32` containing the port number, or -1 if no port is used by the URI
12 `System.Uri.Scheme`. If no port was specified as part of the URI used to construct the
13 current instance, the `System.Uri.Port` property returns the default port for the URI
14 scheme.

15 Description

16 This property is read-only.

17
18 [*Note:* The following table lists the default port number for each supported scheme.

Scheme	Port
file	-1
ftp	21
gopher	70
http	80
https	43
mailto	25
news	119
nntp	119

19]
20]

1 Uri.Query Property

```
2 [ILAsm]  
3 .property string Query { public hidebysig specialname instance string  
4 get_Query() }  
  
5 [C#]  
6 public string Query { get; }
```

7 Summary

8 Gets the query component of the URI used to construct the current instance.

9 Property Value

10 A System.String containing the query information included in the specified URI, or
11 System.String.Empty.

12 Description

13 This property is read-only.

14
15 If, when the current instance was constructed, the URI was already escaped or the
16 constructor's *dontEscape* parameter was set to *false*, the value returned by this
17 property is escaped.

18
19 [Note: Query information is separated from the path information by a question mark
20 ('?') and is located at the end of a URI. The query information includes the leading
21 question mark.]
22
23

24 Example

25 The following example uses the System.Uri.Query property to extract the query from a
26 URI.

```
27 [C#]  
28  
29 using System;  
30  
31 public class UriTest {  
32     public static void Main() {  
33  
34         Uri baseUri = new Uri("http://www.contoso.com/");  
35         Uri myUri = new Uri(baseUri, "catalog/shownew.htm?date=today");  
36  
37         Console.WriteLine(myUri.Query);  
38     }  
39 }
```

40 The output is

41

1 ?date=today
2
3

1 Uri.Scheme Property

```
2 [ILAsm]  
3 .property string Scheme { public hidebysig specialname instance string  
4 get_Scheme() }  
  
5 [C#]  
6 public string Scheme { get; }
```

7 Summary

8 Gets the scheme component of the URI used to construct the current instance.

9 Property Value

10 A System.String containing the URI scheme.

11 Description

12 This property is read-only.

13

1 Uri.UserEscaped Property

```
2 [ILAsm]  
3 .property bool UserEscaped { public hidebysig specialname instance bool  
4 get_UserEscaped() }  
5 [C#]  
6 public bool UserEscaped { get; }
```

7 Summary

8 Gets a `System.Boolean` value that indicates whether the URI information used to
9 construct the current instance was escaped before the current instance was created.

10 Property Value

11 `true` if the *dontEscape* parameter of the constructor for the current instance was set to
12 `true`; otherwise, `false`.

13 Description

14 This property is read-only.

15

1 Uri.UserInfo Property

```
2 [ILAsm]  
3 .property string UserInfo { public hidebysig specialname instance string  
4 get_UserInfo() }  
5 [C#]  
6 public string UserInfo { get; }
```

7 Summary

8 Gets the userinfo component of the URI used to construct the current instance.

9 Property Value

10 A `System.String` containing any user information included in the URI used to construct
11 the current instance, or `System.String.Empty` if no user information was included.

12 Description

13 This property is read-only.

14

15 [*Note:* For details on the userinfo component of a URI, see IETF RFC 2396, 3.2.2.]

16

17

18