

# HttpRequest Class

.NET Framework (current version)

## Note

The .NET API Reference documentation has a new home. Visit the [.NET API Browser](https://docs.microsoft.com/dotnet/api/) on docs.microsoft.com to see the new experience.

Enables ASP.NET to read the HTTP values sent by a client during a Web request.

**Namespace:** [System.Web](#)

**Assembly:** System.Web (in System.Web.dll)

## Inheritance Hierarchy

[System.Object](#)


System.Web.HttpRequest

## Syntax

C#























```
public sealed class HttpRequest
```























## Constructors



	Name	Description
	<a href="#">HttpRequest(String, String, String)</a>	This API supports the product infrastructure and is not intended to be used directly from your code. Initializes an HttpRequest object.

## Properties















	Name	Description




	<a href="#">AcceptTypes</a>	Gets a string array of client-supported MIME accept types.
	<a href="#">AnonymousID</a>	Gets the anonymous identifier for the user, if present.
	<a href="#">ApplicationPath</a>	Gets the ASP.NET application's virtual application root path on the server.
	<a href="#">AppRelativeCurrentExecutionFilePath</a>	Gets the virtual path of the application root and makes it relative by using the tilde (~) notation for the application root (as in "~/page.aspx").
	<a href="#">Browser</a>	Gets or sets information about the requesting client's browser capabilities.
	<a href="#">ClientCertificate</a>	Gets the current request's client security certificate.
	<a href="#">ContentEncoding</a>	Gets or sets the character set of the entity-body.
	<a href="#">ContentLength</a>	Specifies the length, in bytes, of content sent by the client.
	<a href="#">ContentType</a>	Gets or sets the MIME content type of the incoming request.
	<a href="#">Cookies</a>	Gets a collection of cookies sent by the client.
	<a href="#">CurrentExecutionFilePath</a>	Gets the virtual path of the current request.
	<a href="#">CurrentExecutionFilePathExtension</a>	Gets the extension of the file name that is specified in the <a href="#">CurrentExecutionFilePath</a> property.
	<a href="#">FilePath</a>	Gets the virtual path of the current request.
	<a href="#">Files</a>	Gets the collection of files uploaded by the client, in multipart MIME format.
	<a href="#">Filter</a>	Gets or sets the filter to use when reading the current input stream.
	<a href="#">Form</a>	Gets a collection of form variables.
	<a href="#">Headers</a>	Gets a collection of HTTP headers.
	<a href="#">HttpChannelBinding</a>	Gets the <a href="#">ChannelBinding</a> object of the current <a href="#">HttpWorkerRequest</a> instance.
	<a href="#">HttpMethod</a>	Gets the HTTP data transfer method (such as <b>GET</b> , <b>POST</b> , or <b>HEAD</b> ) used by the client.
	<a href="#">InputStream</a>	Gets the contents of the incoming HTTP entity body.
	<a href="#">IsAuthenticated</a>	Gets a value indicating whether the request has been authenticated.
	<a href="#">IsLocal</a>	Gets a value indicating whether the request is from the local computer.

	<a href="#">IsSecureConnection</a>	Gets a value indicating whether the HTTP connection uses secure sockets (that is, HTTPS).
	<a href="#">Item[String]</a>	Gets the specified object from the <a href="#">QueryString</a> , <a href="#">Form</a> , <a href="#">Cookies</a> , or <a href="#">ServerVariables</a> collections.
	<a href="#">LogonUserIdentity</a>	Gets the <a href="#">WindowsIdentity</a> type for the current user.
	<a href="#">Params</a>	Gets a combined collection of <a href="#">QueryString</a> , <a href="#">Form</a> , <a href="#">Cookies</a> , and <a href="#">ServerVariables</a> items.
	<a href="#">Path</a>	Gets the virtual path of the current request.
	<a href="#">PathInfo</a>	Gets the additional path information for a resource with a URL extension.
	<a href="#">PhysicalApplicationPath</a>	Gets the physical file system path of the currently executing server application's root directory.
	<a href="#">PhysicalPath</a>	Gets the physical file system path corresponding to the requested URL.
	<a href="#">QueryString</a>	Gets the collection of HTTP query string variables.
	<a href="#">RawUrl</a>	Gets the raw URL of the current request.
	<a href="#">ReadEntityBodyMode</a>	Gets a value that indicates whether the request entity body has been read, and if so, how it was read.
	<a href="#">RequestContext</a>	Gets the <a href="#">RequestContext</a> instance of the current request.
	<a href="#">RequestType</a>	Gets or sets the HTTP data transfer method ( <b>GET</b> or <b>POST</b> ) used by the client.
	<a href="#">ServerVariables</a>	Gets a collection of Web server variables.
	<a href="#">TimedOutToken</a>	Gets a <a href="#">CancellationToken</a> object that is tripped when a request times out.
	<a href="#">TlsTokenBindingInfo</a>	Gets the TLS token binding information. The property enables applications to retrieve token information from incoming HTTP requests for enhanced authentication.
	<a href="#">TotalBytes</a>	Gets the number of bytes in the current input stream.
	<a href="#">Unvalidated</a>	Gets the HTTP request values without triggering request validation.
	<a href="#">Url</a>	Gets information about the URL of the current request.
	<a href="#">UrlReferrer</a>	Gets information about the URL of the client's previous request that linked to the current URL.
	<a href="#">UserAgent</a>	Gets the raw user agent string of the client browser.
	<a href="#">UserHostAddress</a>	Gets the IP host address of the remote client.

	UserHostName	Gets the DNS name of the remote client.
	UserLanguages	Gets a sorted string array of client language preferences.

## Methods

	Name	Description
	Abort()	Forcibly terminates the underlying TCP connection, causing any outstanding I/O to fail. You might use this method in response to an attack by a malicious HTTP client.
	BinaryRead(Int32 )	Performs a binary read of a specified number of bytes from the current input stream.
	Equals(Object)	Determines whether the specified object is equal to the current object.(Inherited from <a href="#">Object</a> .)
	GetBufferedInputStream()	
	GetBufferlessInputStream()	Gets a <a href="#">Stream</a> object that can be used to read the incoming HTTP entity body.
	GetBufferlessInputStream(Boolean)	Gets a <a href="#">Stream</a> object that can be used to read the incoming HTTP entity body, optionally disabling the request-length limit that is set in the <a href="#">MaxRequestLength</a> property.
	GetHashCode()	Serves as the default hash function. (Inherited from <a href="#">Object</a> .)
	GetType()	Gets the <a href="#">Type</a> of the current instance.(Inherited from <a href="#">Object</a> .)
	InsertEntityBody()	Provides IIS with a copy of the HTTP request entity body.
	InsertEntityBody(Byte[], Int32, Int32)	Provides IIS with a copy of the HTTP request entity body and with information about the request entity object.
	MapImageCoordinates(String)	Maps an incoming image-field form parameter to appropriate x-coordinate and y-coordinate values.
	MapPath(String)	Maps the specified virtual path to a physical path.
	MapPath(String, String, Boolean)	Maps the specified virtual path to a physical path.
	MapRawImageC	Maps an incoming image field form parameter into appropriate x and y coordinate

	<code>ordinates(String )</code>	values.
	<code>SaveAs(String, Boolean)</code>	Saves an HTTP request to disk.
	<code>ToString()</code>	Returns a string that represents the current object.(Inherited from <a href="#">Object</a> .)
	<code>ValidateInput()</code>	Causes validation to occur for the collections accessed through the <a href="#">Cookies</a> , <a href="#">Form</a> , and <a href="#">QueryString</a> properties.

## Remarks

The methods and properties of the HttpRequest class are exposed through the **Request** properties of the [HttpApplication](#), [HttpContext](#), [Page](#), and [UserControl](#) classes.

To access data from the [QueryString](#), [Form](#), [Cookies](#), or [ServerVariables](#) collections, you can write `Request["key"]`, as shown in the example for the [QueryString](#) property.

### Note

Unicode support for HttpRequest class members requires IIS version 6.0 or later.

## Examples

The following examples access the HttpRequest instance for the current request by using the [Request](#) property of the [Page](#) class.

You can use simplified syntax for accessing data from the [QueryString](#), [Form](#), [Cookies](#), or [ServerVariables](#) collections. You can write `Request["key"]`.

The first example shows how to retrieve a query string value when loading a page.

**C#**

```
public partial class AddToCart : Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        string rawId = Request["ProductID"];
        int productId;
        if (!String.IsNullOrEmpty(rawId) && int.TryParse(rawId, out productId))
        {
            using (ShoppingCartActions usersShoppingCart = new ShoppingCartActions())
            {
                usersShoppingCart.AddToCart(productId);
            }
        }
    }
}
```

```

    }
    else
    {
        throw new Exception("Tried to call AddToCart.aspx without setting a ProductId.");
    }
    Response.Redirect("ShoppingCart.aspx");
}
}

```

The next example shows how to check if the request is authenticated and retrieve the raw URL.

**C#**

```

public partial class RestrictedPage : Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!Request.IsAuthenticated)
        {
            var rawUrl = Request.RawUrl;
            Response.Redirect("/Account/Login?ru=" + Server.HtmlEncode(rawUrl));
        }
    }
}

```

A Visual Studio Web site project with source code is available to accompany this topic: [Download](#).

This example uses the [StreamWriter](#) class to write the values of several HttpRequest class properties to a file. For properties that are of type string, the values are HTML encoded as they are written to the file. Properties that represent a collection are looped through, and each key/value pair that they contain is written to the file.



### Security Note

This example has a text box that accepts user input, which is a potential security threat. By default, ASP.NET Web pages validate that user input does not include script or HTML elements. For more information, see [Script Exploits Overview](#).

**C#**

```

<%@ Page Language="C#" %>
<%@ import Namespace="System.Threading" %>
<%@ import Namespace="System.IO" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script runat="server">

    /* NOTE: To use this sample, create a c:\temp\CS folder,
    * add the ASP.NET account (in IIS 5.x <machinename>\ASPNET,
    * in IIS 6.x NETWORK SERVICE), and give it write permissions
    * to the folder.*/

    private const string INFO_DIR = @"c:\temp\CS\RequestDetails";
    public static int requestCount;

```

```
private void Page_Load(object sender, System.EventArgs e)
{
    // Create a variable to use when iterating
    // through the UserLanguages property.
    int langCount;

    int requestNumber = Interlocked.Increment(ref requestCount);

    // Create the file to contain information about the request.
    string strFilePath = INFO_DIR + requestNumber.ToString() + @".txt";

    StreamWriter sw = File.CreateText(strFilePath);

    try
    {
        // Write request information to the file with HTML encoding.
        sw.WriteLine(Server.HtmlEncode(DateTime.Now.ToString()));
        sw.WriteLine(Server.HtmlEncode(Request.CurrentExecutionFilePath));
        sw.WriteLine(Server.HtmlEncode(Request.ApplicationPath));
        sw.WriteLine(Server.HtmlEncode(Request.FilePath));
        sw.WriteLine(Server.HtmlEncode(Request.Path));

        // Iterate through the Form collection and write
        // the values to the file with HTML encoding.
        // String[] formArray = Request.Form.AllKeys;
        foreach (string s in Request.Form)
        {
            sw.WriteLine("Form: " + Server.HtmlEncode(s));
        }

        // Write the PathInfo property value
        // or a string if it is empty.
        if (Request.PathInfo == String.Empty)
        {
            sw.WriteLine("The PathInfo property contains no information.");
        }
        else
        {
            sw.WriteLine(Server.HtmlEncode(Request.PathInfo));
        }

        // Write request information to the file with HTML encoding.
        sw.WriteLine(Server.HtmlEncode(Request.PhysicalApplicationPath));
        sw.WriteLine(Server.HtmlEncode(Request.PhysicalPath));
        sw.WriteLine(Server.HtmlEncode(Request.RawUrl));

        // Write a message to the file dependent upon
        // the value of the TotalBytes property.
        if (Request.TotalBytes > 1000)
        {
            sw.WriteLine("The request is 1KB or greater");
        }
        else
        {

```

```

        sw.WriteLine("The request is less than 1KB");
    }

    // Write request information to the file with HTML encoding.
    sw.WriteLine(Server.HtmlEncode(Request.RequestType));
    sw.WriteLine(Server.HtmlEncode(Request.UserHostAddress));
    sw.WriteLine(Server.HtmlEncode(Request.UserHostName));
    sw.WriteLine(Server.HtmlEncode(Request.HttpMethod));

    // Iterate through the UserLanguages collection and
    // write its HTML encoded values to the file.
    for (langCount=0; langCount < Request.UserLanguages.Length; langCount++)
    {
        sw.WriteLine(@"User Language " + langCount + ": " +
Server.HtmlEncode(Request.UserLanguages[langCount]));
    }
}

finally
{
    // Close the stream to the file.
    sw.Close();
}

lblInfoSent.Text = "Information about this request has been sent to a file.";
}

private void btnSendInfo_Click(object sender, System.EventArgs e)
{
    lblInfoSent.Text = "Hello, " + Server.HtmlEncode(txtBoxName.Text) +
        ". You have created a new request info file.";
}

</script>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>ASP.NET Example</title>
</head>
<body>
    <form id="form1" runat="server">
        <p>
            </p>
        <p>
            Enter your name here:
            <asp:TextBox id="txtBoxName" runat="server"></asp:TextBox>
        </p>
        <p>
            <asp:Button id="btnSendInfo" onclick="btnSendInfo_Click" runat="server"
Text="Click Here"></asp:Button>
        </p>
        <p>
            <asp:Label id="lblInfoSent" runat="server"></asp:Label>
        </p>
    </form>
</body>
</html>

```



## Version Information

### .NET Framework

Available since 1.1

## Thread Safety

Any public static ( **Shared** in Visual Basic) members of this type are thread safe. Any instance members are not guaranteed to be thread safe.

## See Also

[System.Web Namespace](#)

[Return to top](#)

© 2018 Microsoft