HttpListener Class

Definition

Namespace:
System.Net
Assemblies:
System.dll, netstandard.dll, System.Net.HttpListener.dll
Provides a simple, programmatically controlled HTTP protocol listener. This class cannot be inherited.
C#

Copy
```
public sealed class HttpListener : IDisposable
```
Inheritance
Object
HttpListener
Implements
IDisposable

Examples

The following code example demonstrates using a HttpListener.
C#

Copy
```
// This example requires the System and System.Net namespaces.
public static void SimpleListenerExample(string[] prefixes)
{
    if (!HttpListener.IsSupported)
    {
        Console.WriteLine ("Windows XP SP2 or Server 2003 is required to use the HttpListener class.");
        return;
    }
    // URI prefixes are required,
    // for example "http://contoso.com:8080/index/".
    if (prefixes == null || prefixes.Length == 0)
      throw new ArgumentException("prefixes");

    // Create a listener.
    HttpListener listener = new HttpListener();
    // Add the prefixes.
    foreach (string s in prefixes)
    {
        listener.Prefixes.Add(s);
    }
    listener.Start();
    Console.WriteLine("Listening...");
    // Note: The GetContext method blocks while waiting
for a request.
    HttpListenerContext context =
listener.GetContext();
    HttpListenerRequest request = context.Request;
    // Obtain a response object.
    HttpListenerResponse response = context.Response;
    // Construct a response.
    string responseString = "<HTML><BODY> Hello world!
</BODY></HTML>";
    byte[] buffer =
System.Text.Encoding.UTF8.GetBytes(responseString);
    // Get a response stream and write the response to
it.
    response.ContentLength64 = buffer.Length;
    System.IO.Stream output = response.OutputStream;
    output.Write(buffer,0,buffer.Length);
    // You must close the output stream.
    output.Close();
    listener.Stop();
}
```

Remarks

Using the HttpListener class, you can create a simple HTTP protocol listener that responds to HTTP requests. The listener is active for the lifetime of the HttpListener object and runs within your application with its permissions.
To use HttpListener, create a new instance of the class using the HttpListener constructor and use the Prefixes property to gain access to the collection that holds the strings that specify which Uniform Resource Identifier (URI) prefixes the HttpListener should process.
A URI prefix string is composed of a scheme (http or https), a host, an optional port, and an optional path. An example of a complete prefix string is http://www.contoso.com:8080/customerData/. Prefixes must end in a forward slash ("/"). The HttpListener object with the prefix that most closely matches a requested URI responds to the request. Multiple HttpListener objects cannot add the same prefix; a Win32Exception exception is thrown if a HttpListener adds a prefix that is already in use.
When a port is specified, the host element can be replaced with "*" to indicate that the HttpListener accepts requests sent to the port if the requested URI does not match any other prefix. For example, to receive all requests sent to port 8080 when the requested URI is not handled by any HttpListener, the prefix is http://*:8080/. Similarly, to specify that the HttpListener accepts all requests sent to a port, replace the host element with the "+" character. For example, https://+:8080. The "*" and "+" characters can be present in prefixes that include paths.
Starting with .NET Core 2.0 or .NET Framework 4.6 on Windows 10, wildcard subdomains are supported in URI prefixes that are managed by an HttpListener object. To specify a wildcard subdomain, use the "*" character as part of the hostname in a URI prefix. For example, http://*.foo.com/. Pass this as the argument to the Add method. This works as of .NET Core 2.0 or .NET

Framework 4.6 on Windows 10; in earlier versions, this generates an HttpListenerException.

Warning

Top-level wildcard bindings (http://*:8080/ and http://+:8080) should **not** be used. Top-level wildcard bindings can open up your app to security vulnerabilities. This applies to both strong and weak wildcards. Use explicit host names rather than wildcards. Subdomain wildcard binding (for example, `*.mysub.com`) doesn't have this security risk if you control the entire parent domain (as opposed to `*.com`, which is vulnerable). See **rfc7230 section-5.4** for more information.

To begin listening for requests from clients, add the URI prefixes to the collection and call the Start method. HttpListener offers both synchronous and asynchronous models for processing client requests. Requests and their associated responses are accessed using the HttpListenerContext object returned by the GetContext method or its asynchronous counterparts, the BeginGetContext and EndGetContext methods.

The synchronous model is appropriate if your application should block while waiting for a client request and if you want to process only one request at a time. Using the synchronous model, call the GetContext method, which waits for a client to send a request. The method returns an HttpListenerContext object to you for processing when one occurs.

In the more complex asynchronous model, your application does not block while waiting for requests and each request is processed in its own execution thread. Use the BeginGetContext method to specify an application-defined method to be called for each incoming request. Within that method, call the EndGetContext method to obtain the request, process it, and respond.

In either model, incoming requests are accessed using the HttpListenerContext.Request property and are represented by HttpListenerRequest objects. Similarly, responses are accessed using the HttpListenerContext.Response property and are represented by HttpListenerResponse objects. These objects share some functionality with the HttpWebRequest and HttpWebResponse objects, but the latter objects cannot be used in conjunction with HttpListener because they implement client, not server, behaviors.

An HttpListener can require client authentication. You can either specify a particular scheme to use for authentication, or you can specify a delegate that determines the scheme to use. You must require some form of authentication to obtain information about the client's identity. For additional information, see the User, AuthenticationSchemes, and AuthenticationSchemeSelectorDelegate properties.

Note

If you create an **HttpListener** using https, you must select a Server Certificate for that listener. Otherwise, an **HttpWebRequest** query of this **HttpListener** will fail with an unexpected close of the connection.

Note

You can configure Server Certificates and other listener options by using Network Shell (netsh.exe). See **Network Shell (Netsh)** for more details. The executable began shipping with Windows Server 2008 and Windows Vista.

Note

If you specify multiple authentication schemes for the **HttpListener**, the listener will challenge clients in the following order: `Negotiate`, `NTLM`, `Digest`, and then `Basic`.

HTTP.sys

The HttpListener class is built on top of `HTTP.sys`, which is the kernel mode listener that handles all HTTP traffic for Windows. `HTTP.sys` provides connection management, bandwidth throttling, and web server logging. Use the HttpCfg.exe tool to add SSL certificates.

Constructors

HttpListener()	Initializes a new instance of the HttpListener class.

Properties

AuthenticationSchemes	Gets or sets the scheme used to authenticate clients.
AuthenticationSchemeSelectorDelegate	Gets or sets the delegate called to determine the protocol used to authenticate clients.
DefaultServiceNames	Gets a default list of Service Provider Names (SPNs) as determined by registered prefixes.
ExtendedProtectionPolicy	Gets or sets the ExtendedProtectionPolicy to use for extended protection for a session.
ExtendedProtectionSelectorDelegate	Gets or sets the delegate called to determine the ExtendedProtectionPolicy to use for each request.
IgnoreWriteExceptions	Gets or sets a Boolean value that specifies whether your application receives exceptions that occur when an HttpListener sends the response to the client.

IsListening	Gets a value that indicates whether HttpListener has been started.
IsSupported	Gets a value that indicates whether HttpListener can be used with the current operating system.
Prefixes	Gets the Uniform Resource Identifier (URI) prefixes handled by this HttpListener object.
Realm	Gets or sets the realm, or resource partition, associated with this HttpListener object.
TimeoutManager	The timeout manager for this HttpListener instance.
UnsafeConnectionNtlmAuthentication	Gets or sets a Boolean value that controls whether, when NTLM is used, additional requests using the same Transmission Control Protocol (TCP) connection are required to authenticate.

Methods

Abort()	Shuts down the HttpListener object immediately, discarding all currently queued requests.
BeginGetContext(AsyncCallback, Object)	Begins asynchronously retrieving an incoming request.
Close()	Shuts down the HttpListener.

EndGetContext(IAsyncResult)	Completes an asynchronous operation to retrieve an incoming client request.
Equals(Object)	Determines whether the specified object is equal to the current object. (Inherited from Object)
GetContext()	Waits for an incoming request and returns when one is received.
GetContextAsync()	Waits for an incoming request as an asynchronous operation.
GetHashCode()	Serves as the default hash function. (Inherited from Object)
GetType()	Gets the Type of the current instance. (Inherited from Object)
MemberwiseClone()	Creates a shallow copy of the current Object. (Inherited from Object)
Start()	Allows this instance to receive incoming requests.
Stop()	Causes this instance to stop receiving incoming requests.
ToString()	Returns a string that represents the current object. (Inherited from Object)

Explicit Interface Implementations

IDisposable.Dispose()	Releases the resources held by this HttpListener object.

Applies to

.NET Core
3.0 2.2 2.1 2.0
.NET Framework
4.8 4.7.2 4.7.1 4.7 4.6.2 4.6.1 4.6 4.5.2 4.5.1 4.5 4.0 3.5 3.0 2.0
.NET Standard
2.1 2.0
Xamarin.Android
7.1
Xamarin.iOS
10.8
Xamarin.Mac
3.0

See also

- Changes to NTLM authentication for HTTPWebRequest in Version 3.5 SP1
- HttpCfg.exe