HttpListener. HTTP-сервер

Последнее обновление: 09.11.2022

Для прослушивания подключений по протоколу HTTP и ответа на HTTP-запросы предназначен класс **HttpListener** из пространства имен `System.Net`. Данный класс построен на базе библиотеки HTTP.sys, которая является слушателем режима ядра и обрабатывает весь трафик HTTP для Windows. (Исходный код класса HttpListener)

Для создания объекта HttpListener применяется его единственный конструктор без параметров:

```
1 HttpListener server = new HttpListener();
```

Свойства и методы HttpListener

Для получения информации или установки различных аспектов HttpListener можно применять свойства класса, основные из них:

- **AuthenticationSchemes**: возвращает или задает схему аутентификации клиентов
- **IsListening**: указывает, был ли запущен объект HttpListener
- **IsSupported**: указывает, можно ли использовать прослушиватель HttpListener в текущей операционной системе.
- **Prefixes**: возвращает префиксы URI, обрабатываемые этим объектом HttpListener.

Для управления HttpListener применяются методы класса. Некоторые из них:

- **Close()**: завершает работу HttpListener.
- **GetContext() / GetContextAsync()**: ожидает входящие запросы
- **Start()**: запускает прослушивание входящих запросов

- **Stop()**: останавливает получение новых входящих запросов и завершает обработку всех текущих запросов.

Запуск HttpListener

Для работы с HttpListener нам надо вначале задать адреса, которые будут использоваться для обращения к приложению. Адреса задаются через свойство **Prefixes** класса HttpListener. При этом адрес должен оканчиваться на слеш, например:

```
using System.Net;

HttpListener server = new HttpListener();
// установка адресов прослушки
server.Prefixes.Add("http://127.0.0.1:8888/connection/");
```

В данном случае сервер будет прослушивать подключения по адресу "http://127.0.0.1:8888/connection/".

Чтобы начать прослушивать входящие подключения, надо вызвать метод **Start()**, для остановки сервера - метод **Stop()**, а для окончательного закрытия HttpListener - метод **Close()**.

```
using System.Net;

HttpListener server = new HttpListener();
// установка адресов прослушки
server.Prefixes.Add("http://127.0.0.1:8888/connection/");
server.Start(); // начинаем прослушивать входящие подключения
// ............
server.Stop(); // останавливаем сервер
server.Close(); // закрываем HttpListener
```

Контекст

С помощью метода **GetContext()/GetContextAsync()** можно получить контекст входящего подключения в виде объекта **HttpListenerContext**. С помощью свойств этого объекта можно получить информацию о входящем запросе и установить ответ:

- **Request**: возвращает информацию о запросе в виде объекта `HttpListenerRequest`
- **Response**: устанавливаливает ответ сервера в виде объекта `HttpListenerResponse`
- **User**: представляет данные об аутентифицированном пользователе в виде объекта `System.Security.Principal.IPrincipal?`

```
using System.Net;

HttpListener server = new HttpListener();
// установка адресов прослушки
server.Prefixes.Add("http://127.0.0.1:8888/connection/");
server.Start(); // начинаем прослушивать входящие подключения

// получаем контекст
var context = await server.GetContextAsync();

var request = context.Request;  // получаем данные запроса
var response = context.Response;    // получаем объект для установки ответа
var user = context.User;        // получаем данные пользователя

server.Stop(); // останавливаем сервер
```

Отправка ответа

Свойство **Response** объекта HttpListenerContext представляет объект **HttpListenerResponse**, который позволяет настроить и отправить клиенту некоторый ответ.

Для настройки ответа можно применять свойства HttpListenerResponse:

- **ContentEncoding**: устанавливает кодировку ответа в виде объекта Encoding из заголовка Content-Type в виде объекта System.Text.Encoding.
- **ContentLength64**: устанавливает заголовок Content-Length (размер ответа).
- **ContentType**: устанавливает значение заголовка Content-Type.
- **Cookies**: устанавливает файлы cookie, которые отправляются клиенту в виде объекта `System.Net.CookieCollection`.
- **Headers**: устанавливает заголовки в виде коллекцию `WebHeaderCollection` (аналоги словаря, где названия заголовков служат ключами).
- **KeepAlive**: устанавливает значение `bool`, которое указывает, требуется ли постоянное подключение.
- **OutputStream**: представляет поток ответа.
- **ProtocolVersion**: устанавливает версию протокола HTTP, которая используется в ответе.

- **RedirectLocation**: устанавливает адрес для переадресации.
- **SendChunked**: представляет значение bool, которое указывает, будет ли ответ разбиваться на чанки/части.
- **StatusCode**: устанавливает статусный код ответа.
- **StatusDescription**: устанавливает описание к статусному коду (строку статуса).

Кроме того, класс предоставляет ряд методов для настройки поведения при отправке ответа:

- `void AddHeader (string name, string value)`: добавляет в ответ заголовок с именем name и значением value
- `void AppendCookie (System.Net.Cookie cookie);`: добавляет в ответ куки
- `void AppendHeader (string name, string value)`: добавляет к заголовоку с именем name значение value
- `void Redirect (string url)`: выполняет редирект на адрес url
- `void SetCookie (System.Net.Cookie cookie)`: устанавливает куки

Например, отправим ответ клиенту:

```
using System.Net;
using System.Text;

HttpListener server = new HttpListener();
// установка адресов прослушки
server.Prefixes.Add("http://127.0.0.1:8888/connection/");
server.Start(); // начинаем прослушивать входящие подключения

// получаем контекст
var context = await server.GetContextAsync();

var response = context.Response;
// отправляемый в ответ код htmlвозвращает
string responseText =
    @"<!DOCTYPE html>
    <html>
        <head>
            <meta charset='utf8'>
            <title>METANIT.COM</title>
```

```
        </head>
        <body>
            <h2>Hello METANIT.COM</h2>
        </body>
    </html>";
byte[] buffer = Encoding.UTF8.GetBytes(responseText);
// получаем поток ответа и пишем в него ответ
response.ContentLength64 = buffer.Length;
using Stream output = response.OutputStream;
// отправляем данные
await output.WriteAsync(buffer);
await output.FlushAsync();

Console.WriteLine("Запрос обработан");

server.Stop();
```

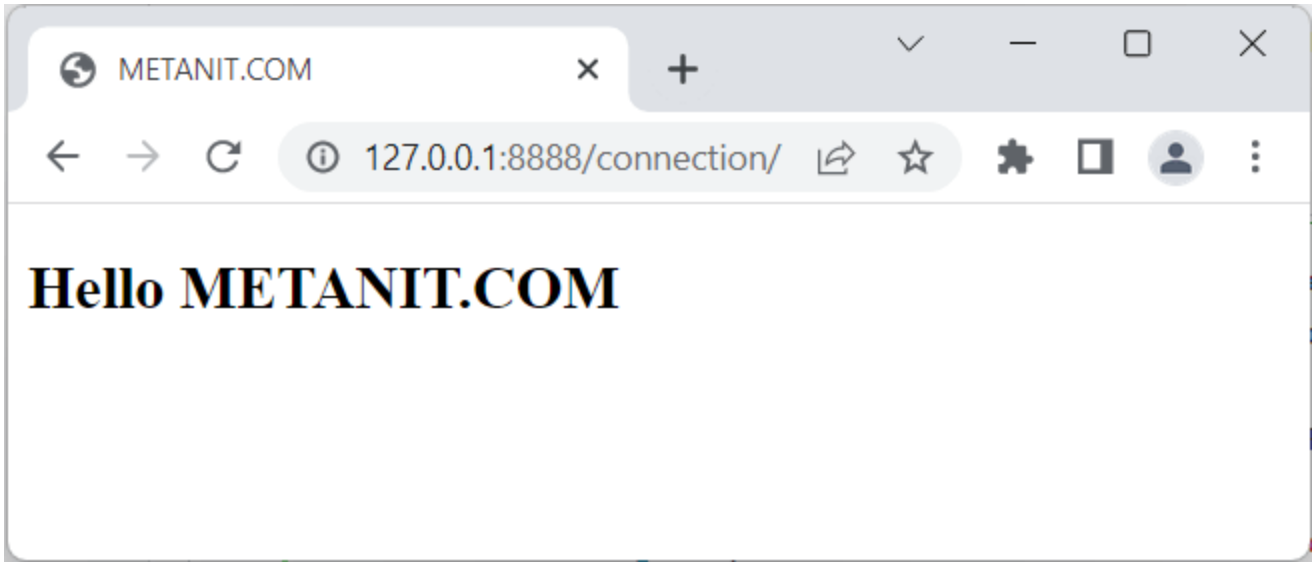
В данном случае наша программа будет прослушивать все обращения по локальному адресу `http://localhost:8888/connection/`

При вызове метода `listener.GetContextAsync()` текущий поток блокируется, и слушатель начинает ожидать входящие подключения. Из этого метода получаем контекста HttpListenetContext, а у него объект ответа - `context.Response`.

После получения ответа получаем выходной поток через свойство response.OutputStream и пишем в него массив байтов:

```
await output.WriteAsync(buffer);
```

В итоге, если мы запустим приложение и обратимся в каком-нибудь браузере по адресу `http://localhost:8888/connection/`, то нам отобразится сформированный в приложении код html:



Получение информации о запросе

Свойство Request возвращает информацию о запросе в виде объекта `HttpListenerRequest`. Вся эта информация хранится в свойствах класса, которые в основном возвращают значения тех или иных заголовоков запроса:

- **AcceptTypes**: возвращает значение заголовка Accept в виде массива string[].
- **ClientCertificateError**: возвращает код ошибки, связанной с сертификатом X509Certificate, предоставленным клиентом.
- **ContentEncoding**: возвращает значение кодировки из заголовка Content-Type в виде объекта System.Text.Encoding.
- **ContentLength64**: возвращает заголовка Content-Length (если он не установлен, то возвращается -1).
- **ContentType**: возвращает значение заголовка Content-Type.
- **Cookies**: возвращает файлы cookie, отправленные вместе с запросом в виде объекта `System.Net.CookieCollection`.
- **HasEntityBody**: возвращает `true`, если в запросе кроме заголовоков переданы какие-нибудь данные (тело запроса).
- **Headers**: возвращает все заголовки в виде коллекцию `System.Collections.Specialized.NameValueCollection` (аналоги словаря, где названия заголовков служат ключами).

- **HttpMethod**: возвращает метод HTTP, использованный для отправки запроса клиентом.
- **InputStream**: возвращает поток, которые содержит данные тела запроса.
- **IsAuthenticated**: возвращает значение `bool`, которое указывает, аутентифицирован ли клиент, отправивший этот запрос.
- **IsLocal**: возвращает значение `bool`, которое указывает, был ли запрос отправлен с локального компьютера.
- **IsSecureConnection**: возвращает значение `bool`, которое указывает, применяется ли для запроса протокол SSL.
- **IsWebSocketRequest**: возвращает значение `bool`, которое указывает, использовался ли при запросе WebSocket.
- **KeepAlive**: возвращает значение `bool`, которое указывает, требует ли клиент постоянного подключения.
- **LocalEndPoint**: возвращает IP-адрес сервера и номер порта, на который будет перенаправлен запрос.
- **ProtocolVersion**: возвращает версию протокола HTTP, которая используется в запросе.
- **QueryString**: возвращает строку запроса.
- **RawUrl**: возвращает URL-адрес запроса (без узла и порта).
- **RemoteEndPoint**: возвращает IP-адрес клиента, который отправил запрос.
- **RequestTraceIdentifier**: возвращает идентификатор HTTP-запроса.
- **Url**: возвращает адрес запроса в виде объекта `Uri`.
- **UrlReferrer**: возвращает URI ресурса, который перенаправляет клиент на сервер.
- **UserAgent**: возвращает заголовок User-Agent.
- **UserHostAddress**: возвращает IP-адрес, на который будет перенаправлен запрос.
- **UserHostName**: возвращает DNS-хост, указанный клиентом.

- **UserLanguages**: возвращает значение заголовка AcceptLanguage (при его отсутствии возвращается null).

Так, обработаем входящее подключение и получим эту информацию

```
using System.Net;
using System.Text;

HttpListener server = new HttpListener();
// установка адресов прослушки
server.Prefixes.Add("http://127.0.0.1:8888/connection/");
server.Start(); // начинаем прослушивать входящие подключения
Console.WriteLine("Сервер запущен. Ожидание подключений...");

// получаем контекст
var context = await server.GetContextAsync();

var request = context.Request;  // получаем данные запроса

Console.WriteLine($"адрес приложения: {request.LocalEndPoint}");
Console.WriteLine($"адрес клиента: {request.RemoteEndPoint}");
Console.WriteLine(request.RawUrl);
Console.WriteLine($"Запрошен адрес: {request.Url}");
Console.WriteLine("Заголовки запроса:");
foreach(string item in request.Headers.Keys)
{
    Console.WriteLine($"{item}:{request.Headers[item]}");
}

var response = context.Response;    // получаем объект для установки ответа
byte[] buffer = Encoding.UTF8.GetBytes("Hello METANIT");
// получаем поток ответа и пишем в него ответ
response.ContentLength64 = buffer.Length;
using Stream output = response.OutputStream;
// отправляем данные
await output.WriteAsync(buffer);
await output.FlushAsync();

server.Stop(); // останавливаем сервер
```

Также обратимся в браузере к приложению по адресу "http://127.0.0.1:8888/connection/", и консоль приложения выведет данные запроса. Консольный вывод в моем случае:

```
Сервер запущен. Ожидание подключений...
```

```
адрес приложения: 127.0.0.1:8888
адрес клиента: 127.0.0.1:52913
/connection/
Запрошен адрес: http://127.0.0.1:8888/connection/
Заголовки запроса:
sec-ch-ua:"Google Chrome";v="107", "Chromium";v="107", "Not=A?Brand";v="24"
sec-ch-ua-mobile:?0
sec-ch-ua-platform:"Windows"
Upgrade-Insecure-Requests:1
Sec-Fetch-Site:none
Sec-Fetch-Mode:navigate
Sec-Fetch-User:?1
Sec-Fetch-Dest:document
Connection:keep-alive
Accept:text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image
/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding:gzip, deflate, br
Accept-Language:ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7,fr;q=0.6
Host:127.0.0.1:8888
User-Agent:Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/107.0.0.0 Safari/537.36
```

Назад **Содержание** **Назад**

M **max**

5 лет назад edited

Можно свой ай-пи адрес в префиксе указывать , тогда можно будет снаружи подключиться. Типа вот , такого.
HttpListener listener = new HttpListener();
listener.Prefixes.Add("http://"+server_IP +":" + server_PORT + "/");

И ещё ,коллеги , может пригодится кому-нибудь.Обратите внимание на комментарий.
// метод GetContext блокирует текущий поток, ожидая получение запроса
Просто долго возился , пока не понял.
Вот такой код:

```
....
DateTime begin = DateTime.UtcNow;
HttpListenerContext context = listener.GetContext();
....
кокой-то код.
....
DateTime end = DateTime.UtcNow;
Console.WriteLine((end-begin).Milliseconds);
listener.Stop();
```

- Покажет время между последним подключением к серверу и временем выполнения какого-то кода.

А вот такой код:

```
....
HttpListenerContext context = listener.GetContext();
DateTime begin = DateTime.UtcNow;
....
кокой-то код.
```

```
....
DateTime end = DateTime.UtcNow;
Console.WriteLine((end-begin).Milliseconds);
listener.Stop();
```

- Покажет время выполнения самого кода.(т.е. время обработки запроса).

3 0 Ответить •

Альберт Хорошун

7 лет назад edited

Помогите. Как теперь открыть эту страницу с другого устройства? После запуска программы ничего не получается. Пытался в префиксы добавить http://+:8888/, но тоже не работает. Брандмауэр полностью отключал, пытался на телефоне зайти по адресу http://<ipмоегоКомпаВИнтернете>:8888/

1 0 Ответить •

A **Al K** → Альберт Хорошун

7 лет назад

Нужно от имени администратора запускать это приложение.

1 0 Ответить •

M **Михаил** → Альберт Хорошун

5 лет назад

нужно открыть порт, в настройках роутера и все заробатает

0 0 Ответить •

D **Disgusting** → Альберт Хорошун

5 лет назад

Используя TcpListener такой фигни не возникает но приходится самому обрабатывать протокол http.

0 0 Ответить •

D **Disgusting** → Альберт Хорошун

5 лет назад

Нужен локальный IP

0 0 Ответить •

D **Disgusting** → Disgusting

5 лет назад

Даже так не работает.

0 0 Ответить •

Ivan Zhukov

10 месяцев назад

Здравствуйте, у Вас в разделе "Контекст" посторяется три раза строка:
Request: возвращает информацию о запросе в виде объекта HttpListenerRequest

Request: возвращает информацию о запросе в виде объекта HttpListenerRequest

Request: возвращает информацию о запросе в виде объекта HttpListenerRequest

0 0 Ответить •

Metanit **Модератор** → Ivan Zhukov

10 месяцев назад

поправил

0 0 Ответить •

Стас Добряков

год назад

То есть HttpListener представляет собой сервер, который слушает подключение по HTTP и отвечает на HTTP-запросы?

0 0 Ответить •

Hafiz Mamedov

2 года назад edited

Я вам бесконечно благодарен за ваш ресурс. Вы помогли мне стать программистом. Сейчас я хочу отплатить. Раздел сетевое программирование достаточно мал. Я буду рад расширять его, сохраняя прежнюю стилистику. Могу даже указывать ресурсы и литературу при написании, что ваши редакторы или специалисты могли проверить достоверность.
Буду очень рад заниматься этим!

0 0 Ответить •

Hafiz Mamedov → Hafiz Mamedov

2 года назад

@Metanit

0 0 Ответить •

Metanit **Модератор** → Hafiz Mamedov

2 года назад

у меня нет никаких редакторов или специалистов, я один работал и работаю над сайтом. Если у вас есть какие-то предложения по статьям, вы можете присылать мне их почту

1 0 Ответить •

Hafiz Mamedov → Metanit

2 года назад

Можно адрес почты?) Я отправлял на один, но не ответили. Отправлял на тот который указан для PayPal-а

0 0 Ответить •

Metanit Модератор → Hafiz Mamedov

2 года назад

отправьте еще раз, возможно, из-за высокой занятости я просмотрел письма

0 0 Ответить •

A **Alexandr Petrov**

3 года назад

Как с https использовать загрузить сертификат ssl

0 0 Ответить •

A **Alexandr Petrov**

3 года назад

Здравствуйте!
Я пытаюсь запустить прослушиватель HttpListener использую код в примера на моем VPS.
Добавляю префикс
listener.Prefixes.Add("https://mysyte.ru/:4000/")
Но соединения по https нет.

Если подключаюсь по http , listener.Prefixes.Add("http://mysyte.ru/:4000/")
Все отлично работает. Соответсвенно дело не в закрытом порте.

как запустить прослушивание HttpListener по https ?
Подскажите пожалуйста!

0 0 Ответить •

Mikhail Balanov → Alexandr Petrov

24 дня назад

У вас ошибка в URL: перед двоеточием не нужен слэш. Должно быть так: "http://mysyte.ru:4000/".

0 0 Ответить •

Дмитрий Гумиров

4 года назад

Почему у меня не подключается к localhost?

0 0 Ответить •

Дмитрий Гумиров → Дмитрий Гумиров

4 года назад

все разобрался я оказывается запрещенный порт на прослушку поставил

0 0 Ответить •

Kovalevich Vladimir

4 года назад

как зайти на свой сервер на пк к которому привязан через роутр конкретный ИП , через браузер телефона, подключенный к сети через тот же роутер??

0 0 Ответить •

Д

Денис Медведев

5 лет назад

Спасибо Вам за сайт, все очень информативно и доступно!
Однако хотелось бы больше инфы по протоколу http, например не понятно, как передать, допустим png картинку или получить ее назад и сохранить.
В общем передачу файлов по http приходится нагугливать в других ресурсах, но за основы работы с http и вообще с System.Net спасибо Вам большое =)

0 0 Ответить •

D

Disgusting → Денис Медведев

5 лет назад

Считываешь .png файл через FileStream в buffer а потом передаёшь через output.Write(buffer, 0, buffer.Length);
и заголовок http Content-Type меняешь на image/png

1 0 Ответить •

Семён Новиков

5 лет назад

А можно вместо http://localhost:8888/connection/ подставлять любые http сайты? А то у меня ошибка с этим : Прослушивание префикса 'http://****', невозможно. Так как он вступает в конфликт с существующей регистрацией на этом компьютере .

0 0 Ответить •

Metanit **Модератор** → Семён Новиков

5 лет назад

нет, нельзя, это фактически локальный http-сервер

0 0 Ответить •

Yurii Nosach
6 лет назад

А можно забиндить listener к адресу устройства в локальной сети (192.168.*.*) и обращаться по этому адресу с другого устройства, подключенного к роутеру?

0 0 Ответить •

Rotozey → Yurii Nosach
3 года назад

Можно.
Но для регистрации такого префикса процесс должен иметь привилегии администратора.

0 0 Ответить •

Metanit Модератор → Yurii Nosach
6 лет назад

нет, лучше к локальному адресу 127.0.0.1

0 0 Ответить •

Disgusting → Metanit
5 лет назад

а чё так?

0 0 Ответить •

bas-tion.ru
7 лет назад edited

Не совсем понятно зачем асинхронно ожидать получения контекста:

```
HttpListenerContext context = await listener.GetContextAsync();
```

если всё равно далее выполнение программы невозможно, из-за `await`?

0 0 Ответить •

Metanit Модератор → bas-tion.ru
7 лет назад

ну так после получения контекста программа дальше продолжит свое выполнение

0 0 Ответить •

bas-tion.ru → Metanit
7 лет назад

Я имел ввиду:"Зачем асинхронно?" ведь всё равно нужно ждать запроса от клиента как и при синхронном получении контекста.

0 0 Ответить •

romanelliua bas-tion.ru

Некропост, конечно, но вдруг кому понадобится
Дело в том, что при синхронном выполнении поток будет занимать процессорное время(хоть и небольшое), вместо того чтобы "уснуть" и позволить занять это время другим потокам системы
Также, класс Task - абстракция над ThreadPool. Т.е. при работе с "тасками" TaskSheduler определяет, стоит ли кинуть awaitable код на свободный поток пулла. Таким образом, таски упрощают разработку асинхронного/многопоточного кода
Правда, все зависит от задачи - асинхронность полезна не всегда, так как подразумевает дополнительные накладные расходы, которые в нек ситуациях(таких не много, но они есть:)) не окупаются

1 0 Ответить •

Kovalevich Vladimir bas-tion.ru

4 года назад

у меня тот же вопрос

0 0 Ответить •

Metanit Модератор bas-tion.ru

7 лет назад

приведено просто как вариант асинхронного получения контекста, вы же можете выбрать тот способ, который вам больше нравится - синхронный или асинхронный

0 0 Ответить •

B **bas-tion.ru** Metanit

7 лет назад

Thanks,

0 0 Ответить •

B **bas-tion.ru**

7 лет назад edited

В первом примере, перед `listener.Stop()`, необходимо поставить задержку `Thread.Sleep(20);`. Иначе ответ не успевает обработаться.

0 0 Ответить •

B **bas-tion.ru**

7 лет назад edited

Пытаюсь запустить HTTP прослушивание.
Если ввожу адрес вместо localhost вываливается исключение "Отказано в доступе". В чём причина?

```
try
{
const string ADDRESS = "http://127.0.0.1";
const string PORT = "9999";
string prefix = $"{ADDRESS}:{PORT}/";

HttpListener listener = new HttpListener();
listener.Prefixes.Add(prefix);

listener.Start();
```