How to make HTTP POST web request

966

484

Canonical: How can I make an HTTP request and send some data using the `POST` method? I can do `GET` request but have no idea how to make a `POST` .

c# .net post httpwebrequest httprequest

edited Apr 17 at 3:45
Jeremy Thompson
41.5k 13 113 211

asked Oct 25 '10 at 14:05
Hooch
11.1k 26 73 138

10 Answers

1882

There are several ways to perform HTTP `GET` and `POST` requests:

Method A: HttpClient

Available in: .NET Framework 4.5+, .NET Standard 1.1+, .NET Core 1.0+

Currently the preferred approach. Asynchronous. Portable version for other platforms available via NuGet.

```
using System.Net.Http;
```

Setup

It is recommended to instantiate one `HttpClient` for your application's lifetime and share it.

Join Stack Overflow to learn, share knowledge, and build your career.

Email Sign Up OR SIGN IN WITH Google Facebook

```
var values = new Dictionary<string, string>
{
   { "thing1", "hello" },
   { "thing2", "world" }
};

var content = new FormUrlEncodedContent(values);

var response = await client.PostAsync("http://www.example.com/recepticle.aspx",
content);

var responseString = await response.Content.ReadAsStringAsync();
```

GET

```
var responseString = await
client.GetStringAsync("http://www.example.com/recepticle.aspx");
```

Method B: 3rd-Party Libraries

RestSharp

Tried and tested library for interacting with REST APIs. Portable. Available via NuGet.

Flurl.Http

Newer library sporting a fluent API and testing helpers. HttpClient under the hood. Portable. Available via NuGet.

```
using Flurl.Http;
```

POST

```
var responseString = await "http://www.example.com/recepticle.aspx"
    .PostUrlEncodedAsync(new { thing1 = "hello", thing2 = "world" })
```

Join Stack Overflow to learn, share knowledge, and build your career.

Email Sign Up OR SIGN IN WITH Google Facebook

```
var responseString = await "http://www.example.com/recepticle.aspx"
    .GetStringAsync();
```

Method C: Legacy

Available in: .NET Framework 1.1+, .NET Standard 2.0+, .NET Core 1.0+

```
using System.Net;
using System.Text;  // for class Encoding
using System.IO;    // for StreamReader
```

POST

```
var request =
(HttpWebRequest)WebRequest.Create("http://www.example.com/recepticle.aspx");

var postData = "thing1=hello";
    postData += "&thing2=world";
var data = Encoding.ASCII.GetBytes(postData);

request.Method = "POST";
request.ContentType = "application/x-www-form-urlencoded";
request.ContentLength = data.Length;

using (var stream = request.GetRequestStream())
{
    stream.Write(data, 0, data.Length);
}

var response = (HttpWebResponse)request.GetResponse();

var responseString = new StreamReader(response.GetResponseStream()).ReadToEnd();
```

GET

Join Stack Overflow to learn, share knowledge, and build your career.

Email Sign Up OR SIGN IN WITH Google Facebook

Method D: WebClient (Also now legacy)

Available in: .NET Framework 1.1+, .NET Standard 2.0+, .NET Core 2.0+

```
using System.Net;
using System.Collections.Specialized;
```

POST

```
using (var client = new WebClient())
{
    var values = new NameValueCollection();
    values["thing1"] = "hello";
    values["thing2"] = "world";

    var response = client.UploadValues("http://www.example.com/recepticle.aspx",
values);

    var responseString = Encoding.Default.GetString(response);
}
```

GET

```
using (var client = new WebClient())
{
    var responseString =
client.DownloadString("http://www.example.com/recepticle.aspx");
}
```

edited Sep 15 '17 at 21:56

community wiki
25 revs, 17 users 45%
Evan Mulawski

2 @Lloyd: `HttpWebResponse response = (HttpWebResponse)HttpWReq.GetResponse();` – Evan Mulawski Mar 25 '11 at 17:44

Join Stack Overflow to learn, share knowledge, and build your career.

Email Sign Up OR SIGN IN WITH Google Facebook

18 @Hiep: They are not deprecated, there are just newer (and is most cases, better and more flexible) ways of making web requests. In my opinion, for simple, non-critical operations, the old ways are just fine - but it's up to you and whatever you are most comfortable with. – Evan Mulawski Nov 18 '15 at 14:47

350

+50

Simple GET request

```
using System.Net;

...

using (var wb = new WebClient())
{
    var response = wb.DownloadString(url);
}
```

Simple POST request

```
using System.Net;
using System.Collections.Specialized;

...

using (var wb = new WebClient())
{
    var data = new NameValueCollection();
    data["username"] = "myUser";
    data["password"] = "myPassword";

    var response = wb.UploadValues(url, "POST", data);
    string responseInString = Encoding.UTF8.GetString(response);
}
```

edited Jan 6 '18 at 9:55
J. Doe
3 2

answered Nov 11 '11 at 9:28
Pavlo Neyman
5,957 3 22 26

Join Stack Overflow to learn, share knowledge, and build your career.

Email Sign Up OR SIGN IN WITH Google Facebook

I accepted your answer as good because it is much more simpler and clearer. – Hooch Jan 3 '14 at 22:09

12 I would like to add that the response variable for the POST request is a byte array. In order to get the string response you just do Encoding.ASCII.GetString(response); (using System.Text) – Sindre Jan 17 '14 at 19:52

1 Further, you can send a bit complex array $_POST['user'] as: data["user[username]"] = "myUsername"; data["user[password]"] = "myPassword"; – Bimal Poudel Jul 17 '16 at 0:19

58

MSDN has a sample.

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

namespace Examples.System.Net
{
    public class WebRequestPostExample
    {
        public static void Main()
        {
            // Create a request using a URL that can receive a post.
            WebRequest request =
WebRequest.Create("http://www.contoso.com/PostAccepter.aspx");
            // Set the Method property of the request to POST.
            request.Method = "POST";
            // Create POST data and convert it to a byte array.
            string postData = "This is a test that posts this string to a Web server.";
            byte[] byteArray = Encoding.UTF8.GetBytes(postData);
            // Set the ContentType property of the WebRequest.
            request.ContentType = "application/x-www-form-urlencoded";
            // Set the ContentLength property of the WebRequest.
            request.ContentLength = byteArray.Length;
            // Get the request stream.
            Stream dataStream = request.GetRequestStream();
            // Write the data to the request stream.
            dataStream.Write(byteArray, 0, byteArray.Length);
            // Close the Stream object.
```

Join Stack Overflow to learn, share knowledge, and build your career.

Email Sign Up OR SIGN IN WITH Google Facebook

```
                dataStream = response.GetResponseStream();
                // Open the stream using a StreamReader for easy access.
                StreamReader reader = new StreamReader(dataStream);
                // Read the content.
                string responseFromServer = reader.ReadToEnd();
                // Display the content.
                Console.WriteLine(responseFromServer);
                // Clean up the streams.
                reader.Close();
                dataStream.Close();
                response.Close();
            }
        }
    }
```

edited Jul 31 '17 at 16:31
Endorphinex
16 11

answered Oct 25 '10 at 14:07
Otávio Décio
63.4k 13 146 215

For some reason it didnt work when i was sending large amount of data – AnKing Jul 30 '14 at 14:48

20

This is a complete working example of sending/receiving data in JSON format, I used VS2013 Express Edition

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.OleDb;
using System.IO;
using System.Linq;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;
using System.Web.Script.Serialization;

namespace ConsoleApplication1
{
```

Join Stack Overflow to learn, share knowledge, and build your career.

Email Sign Up OR SIGN IN WITH Google Facebook

```
    public class Program
    {
        private static readonly HttpClient _Client = new HttpClient();
        private static JavaScriptSerializer _Serializer = new JavaScriptSerializer();

        static void Main(string[] args)
        {
            Run().Wait();
        }

        static async Task Run()
        {
            string url = "http://www.example.com/api/Customer";
            Customer cust = new Customer() { Name = "Example Customer", Address = "Some 
example address", Phone = "Some phone number" };
            var json = _Serializer.Serialize(cust);
            var response = await Request(HttpMethod.Post, url, json, new 
Dictionary<string, string>());
            string responseText = await response.Content.ReadAsStringAsync();

            List<YourCustomClassModel> serializedResult = 
_Serializer.Deserialize<List<YourCustomClassModel>>(responseText);

            Console.WriteLine(responseText);
            Console.ReadLine();
        }

        /// <summary>
        /// Makes an async HTTP Request
        /// </summary>
        /// <param name="pMethod">Those methods you know: GET, POST, HEAD, etc...
</param>
        /// <param name="pUrl">Very predictable...</param>
        /// <param name="pJsonContent">String data to POST on the server</param>
        /// <param name="pHeaders">If you use some kind of Authorization you should use 
this</param>
        /// <returns></returns>
        static async Task<HttpResponseMessage> Request(HttpMethod pMethod, string pUrl, 
string pJsonContent, Dictionary<string, string> pHeaders)
        {
            var httpRequestMessage = new HttpRequestMessage();
            httpRequestMessage.Method = pMethod;
```

Join Stack Overflow to learn, share knowledge, and build your career.

Email Sign Up OR SIGN IN WITH Google Facebook

```
            {
                case "POST":
                    HttpContent httpContent = new StringContent(pJsonContent,
Encoding.UTF8, "application/json");
                    httpRequestMessage.Content = httpContent;
                    break;

            }

            return await _Client.SendAsync(httpRequestMessage);
        }
    }
}
```

edited Sep 29 '17 at 20:04

answered Sep 29 '17 at 19:55

Ivanzinho
440 2 6 14

4

Simple (one-liner, no error checking, no wait for response) solution i've found so far

```
(new WebClient()).UploadStringAsync(new Uri(Address), dataString);
```

use with caution!

answered Sep 24 '17 at 14:59

Ohad Cohen
2,787 1 23 24

4 That is quite bad. I don't recommend it as there is no error handling of any kind and debugging it is pain. Additionally there already is great answer to this question. – Hooch Sep 25 '17 at 11:24

1 @Hooch others might be interested in this type of answers, even if it's not the best one. – Mitulát báti Dec 30 '17 at 16:11

Agreed, the only context in which this would be useful is code golfing and who golfs in C# ;) – Extragorey May 9 at 3:58

Join Stack Overflow to learn, share knowledge, and build your career.

Email Sign Up OR SIGN IN WITH Google Facebook

4

```
        var client = new WebClient();
        string credentials = Convert.ToBase64String(Encoding.ASCII.GetBytes(userName +
":" + passWord));
        client.Headers[HttpRequestHeader.Authorization] = $"Basic {credentials}";
        //If you have your data stored in an object serialize it into json to pass to
the webclient with Newtonsoft's JsonConvert
        var encodedJson = JsonConvert.SerializeObject(newAccount);

        client.Headers.Add($"x-api-key:{ApiKey}");
        client.Headers.Add("Content-Type:application/json");
        try
        {
            var response = client.UploadString($"{apiurl}", encodedJson);
            //if you have a model to deserialize the json into Newtonsoft will help bind
the data to the model, this is an extremely useful trick for GET calls when you have a
lot of data, you can strongly type a model and dump it into an instance of that class.
            Response response1 = JsonConvert.DeserializeObject<Response>(response);
```

edited Oct 10 '18 at 20:51

answered Oct 10 '18 at 18:45

Adam
146 1 4

Useful, thanks. BTW It looks like the above technique for setting header-properties also works for the older (deprecated?), HttpWebRequest approach. e.g. myReq.Headers[HttpRequestHeader.Authorization] = $"Basic {credentials}"; – Zeek2 Jun 24 at 15:04

3

When using **Windows.Web.Http** namespace, for POST instead of FormUrlEncodedContent we write HttpFormUrlEncodedContent. Also the response is type of HttpResponseMessage. The rest is as Evan Mulawski wrote down.

answered Feb 20 '18 at 21:28

S4NNY1
174 1 4 13

Join Stack Overflow to learn, share knowledge, and build your career.

Email Sign Up OR SIGN IN WITH Google Facebook

- Compatible with .NET 4.5+.
- Tested with no parameters (requires a "GET" behind the scenes).
- Tested with parameters (requires a "POST" behind the scenes).
- Tested with a standard web page such as Google.
- Tested with an internal Java-based webservice.

Reference:

```
// Add a Reference to the assembly System.Web
```

Code:

```
using System;
using System.Collections.Generic;
using System.Collections.Specialized;
using System.Net;
using System.Net.Http;
using System.Threading.Tasks;
using System.Web;

private async Task<WebResponse> CallUri(string url, TimeSpan timeout)
{
    var uri = new Uri(url);
    NameValueCollection rawParameters = HttpUtility.ParseQueryString(uri.Query);
    var parameters = new Dictionary<string, string>();
    foreach (string p in rawParameters.Keys)
    {
        parameters[p] = rawParameters[p];
    }

    var client = new HttpClient { Timeout = timeout };
    HttpResponseMessage response;
    if (parameters.Count == 0)
    {
        response = await client.GetAsync(url);
    }
    else
```

Join Stack Overflow to learn, share knowledge, and build your career.

Email Sign Up OR SIGN IN WITH Google Facebook

```
        var responseString = await response.Content.ReadAsStringAsync();

        return new WebResponse(response.StatusCode, responseString);
    }

    private class WebResponse
    {
        public WebResponse(HttpStatusCode httpStatusCode, string response)
        {
            this.HttpStatusCode = httpStatusCode;
            this.Response = response;
        }
        public HttpStatusCode HttpStatusCode { get; }
        public string Response { get; }
    }
```

To call with no parameters (uses a "GET" behind the scenes):

```
 var timeout = TimeSpan.FromSeconds(300);
 WebResponse response = await this.CallUri("http://www.google.com/", timeout);
 if (response.HttpStatusCode == HttpStatusCode.OK)
 {
     Console.Write(response.Response); // Print HTML.
 }
```

To call with parameters (uses a "POST" behind the scenes):

```
 var timeout = TimeSpan.FromSeconds(300);
 WebResponse response = await this.CallUri("http://example.com/path/to/page?
name=ferret&color=purple", timeout);
 if (response.HttpStatusCode == HttpStatusCode.OK)
 {
     Console.Write(response.Response); // Print HTML.
 }
```

edited Apr 3 at 15:55

answered Apr 3 at 15:21

Contango
43.2k 49 193 242

Join Stack Overflow to learn, share knowledge, and build your career.

Email Sign Up OR SIGN IN WITH Google Facebook

2

```
await new RequestBuilder<ExampleObject>()
.SetHost("https://httpbin.org")
.SetContentType(ContentType.Application_Json)
.SetType(RequestType.Post)
.SetModelToSerialize(dto)
.Build()
.Execute();
```

Run code snippet   Expand snippet

I'm the author of the library so feel free to ask questions or check the code in github

answered Apr 16 '18 at 11:33
Nikolay Hristov
81 2

1 That is nice. But one more additional dependency for something that works really well in .NET implementation is not something that would be allowed in professional projects and is also unnecessary for even small hobby projects. – Hooch Apr 17 '18 at 9:19

Hi Hooch, I do agree with your argument, but for me the efficiency in the nugget is the model parsing, since depending on your request you can easily parse models to form-data or to json. – Nikolay Hristov Apr 18 '18 at 10:17

0

If you like fluent API you can use Tiny.RestClient it's available at Nuget

```
var client = new TinyRestClient(new HttpClient(), "http://MyAPI.com/api");
// POST
 var city = new City() { Name = "Paris" , Country = "France"};
// With content
var response = await client.PostRequest("City", city).
                ExecuteAsync<bool>();
```

Hope that helps!

Join Stack Overflow to learn, share knowledge, and build your career.

Email Sign Up OR SIGN IN WITH Google Facebook

protected by Evan Mulawski Apr 5 '15 at 2:09

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?

Join Stack Overflow to learn, share knowledge, and build your career.

Email Sign Up OR SIGN IN WITH Google Facebook