HttpClient

.NET

The HttpClient class provides the ability to send HTTP requests and receive HTTP responses from a resource identified by a URI.

https://learn.microsoft.com/en-us/dotnet/fundamentals/networking/http/http-overview

HttpClient Overview

https://learn.microsoft.com/en-us/dotnet/fundamentals/networking/http/httpclient
https://learn.microsoft.com/en-us/dotnet/api/system.net.http.httpclient.defaultrequestversion?view=net-6.0#system-net-http-httpclient-defaultrequestversion

---

HttpClient is .NET's Class for internet communication. An ***HttpClient*** instance is a collection of settings that's applied to all requests executed by that instance. You can make any type of HTTP request using ***HttpClient***.

An ***HttpClient*** instance contains useful fields and methods that help you configure defaults your requests, such as:

- DefaultRequestHeaders – set the default headers for your requests.
- DefaultRequestVersion – Gets or sets the default HTTP Version.
- HttpContent – Used to represent an HTTP request body and headers needed for the request.

HTTP endpoints usually return Serialized JSON data.
For convenience, System.Net.Http.Json (NuGet package) provides several extension methods for ***HttpClient*** that perform automatic serialization and deserialization using ***System.Text.Json***.

```
class Program
{
    // HttpClient is intended to be instantiated once per application.
    // It's strongly recommended to reuse HttpClient instances during the appl
    static readonly HttpClient client = new HttpClient();

    static async Task Main()
    {
        client.BaseAddress = new Uri("https://jsonplaceholder.typicode.com");
```

HttpContent

https://learn.microsoft.com/en-us/dotnet/fundamentals/networking/http/httpclient#http-content

---

The ***HttpContent*** class is used to hold the HTTP request body and the request content headers.

The ***HttpContent*** class is also used as the response body of the ***HttpResponseMessage***., which holds the content return ed from the server.

It is accessible on the property, HttpResponseMessage.Content.

```
StringContent jsonContent = //stringify a json object.
    new(JsonSerializer.Serialize(
        new
        {
            userId = 777,
            id = 1,
            title = "MarkSample",
            completed = false
        }
            ),
    Encoding.UTF8,      // include the encoding (mandatory)
    "application/json"  // tell the server the type of the data
);

// POST the new entity
HttpResponseMessage response = await client.PostAsync("todos", jsonContent);
```

HttpContent

https://learn.microsoft.com/en-us/dotnet/fundamentals/networking/http/httpclient#http-content

Although most returns are JSON formatted, subclasses exist in ***HttpClient*** for different content types. These types are known as Multipurpose Internet Mail Extensions (MIME).

Class/Type Name	Purpose
FormUrlEncodedContent	HTTP content for name/value tuples encoded using "application/x-www-form-urlencoded" MIME type.
ByteArrayContent	HTTP content based on a byte array.
JsonContent	HTTP content ba sed on JSON.
MultipartContent	A collection of ***HttpContent*** objects that get serialized using the "multipart/*" MIME type specification.
MultipartFormDataContent	A container for content encoded using "multipart/form-data" MIME type.
ReadOnlyMemoryContent	HTTP content based on a ReadOnlyMemory<T>.
StreamContent	HTTP content based on a stream.
StringContent	HTTP content based on a string.

HttpClient Post Step-by-step (1/2)

https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/collections

---

1. Create a HttpClient Object with:

```
static readonly HttpClient client = new HttpClient();
```

2. (optional) Set the base URL for your queries with:

```
client.BaseAddress = new Uri("https://jsonplaceholder.typicode.com");
```

3. Create a ***StringContent*** Object leveraging the ***System.Text.Json.JsonSerializer*** Class with:

```
StringContent jsonContent = //stringify a json object.
    new(JsonSerializer.Serialize(
        new
        {
            userId = 777,
            id = 1,
            title = "MarkSample",
            completed = false
        }
            ),
    Encoding.UTF8,       // include the encoding (mandatory)
    "application/json"   // tell the server the type of the data
);
```

HttpClient Post Step-by-step (2/2)

https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/collections

4. POST the new data to the external API with:

```
HttpResponseMessage response = await client.PostAsync("todos", jsonContent);
```

5. Create a model to hold the result of the POST request.

```
public class Todo_Class
{
    public int userId { get; set; }
    public int id { get; set; }
    public string title { get; set; } = "";
    public bool completed { get; set; }
}
```

6. Verify that the Status Code result of the POST was 201 and access the content of the response message with:

```
if (response.IsSuccessStatusCode)
{
    string jsonResponse = await response.Content.ReadAsStringAsync(); // Get the content of the response as a string...
    Todo_Class? resultAsObj = JsonSerializer.Deserialize<Todo_Class>(response.Content.ReadAsStream());// ...OR a C# object
    Console.WriteLine($"The C# object POST return is - {resultAsObj?.title}");
}
```

HttpClient

https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/collections

---

https://jsonplaceholder.typicode.com/guide/

This is where the MS Tutorial sends requests.