

LogisticaTRUCKServer- I 距離計算 WebApi 作成手順

1. VS2022 のインストール

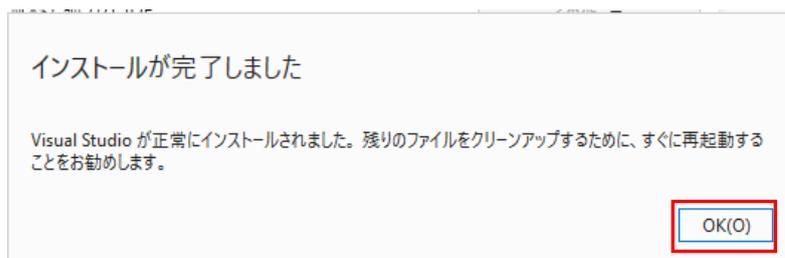
1.1 Visual Studio Community からダウンロード

1.1.1 Visual Studio Community

<https://visualstudio.microsoft.com/ja/vs/community/>



1.1.2 インストールする項目の選択



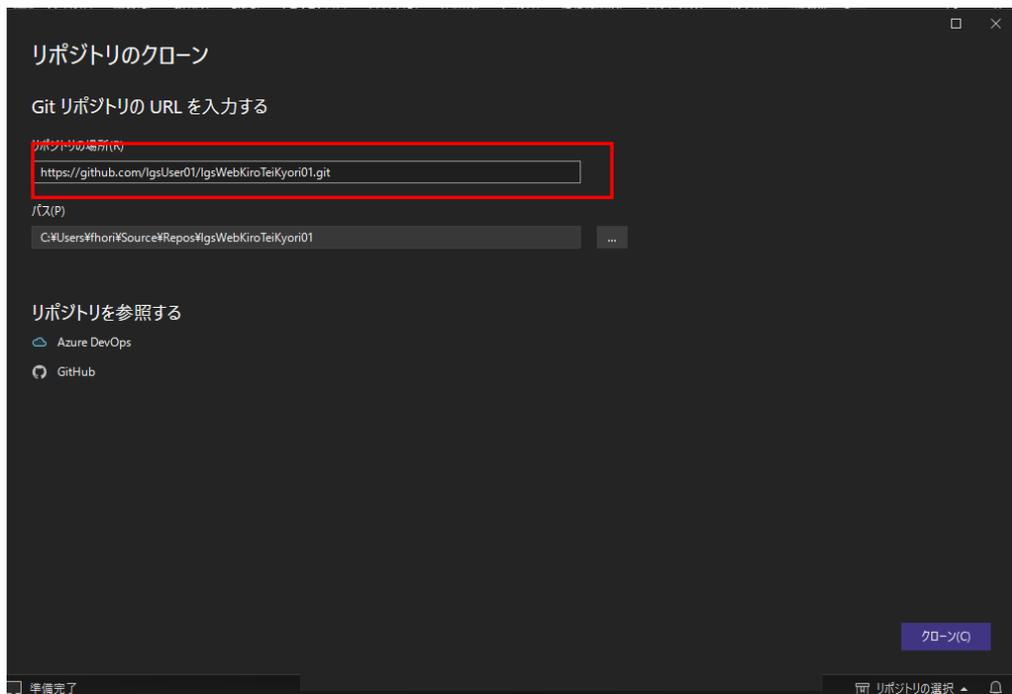
2. GitHub リポジトリのクローンから WebApi 発行

2.1 VS2022 ポジトリのクローン

2.1.1 VS2022 起動 リポジトリのクローン

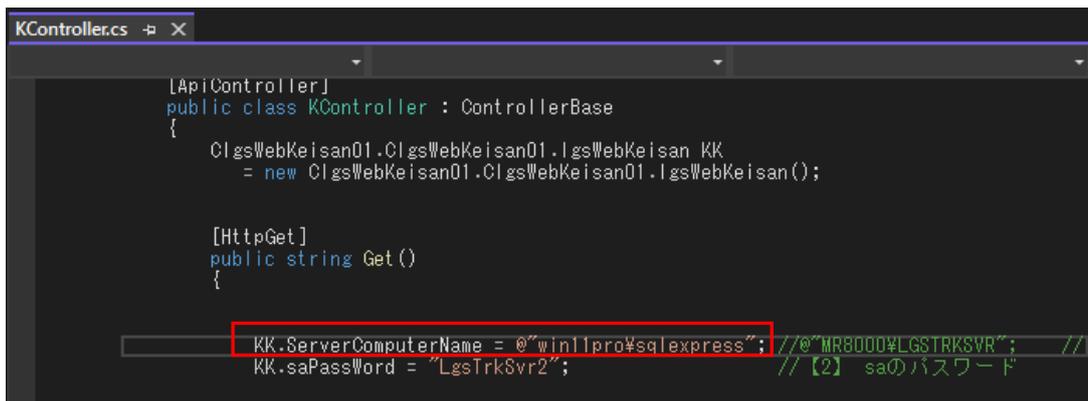


2.1.2 リポジトリの場所 <https://github.com/lgsUser01/lgsWebKiroTeiKiyori01.git>



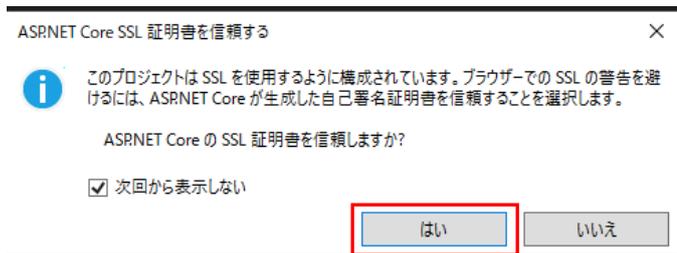
2.1.3 KController.cs

KK.ServerComputerName = 距離計算サーバのインスタンス名を指定する



2.2 デバッグ

2.2.1 デバッグの開始 証明書・セキュリティ関連のメッセージは「はい」



2.2.2 デバッグ IIS Express Debug

https://localhost:5000/api/K/?f=13102&t=27127

```
{"ShuyouKyorikm":576,"SaitanKyorikm":555,"Error":""}
```



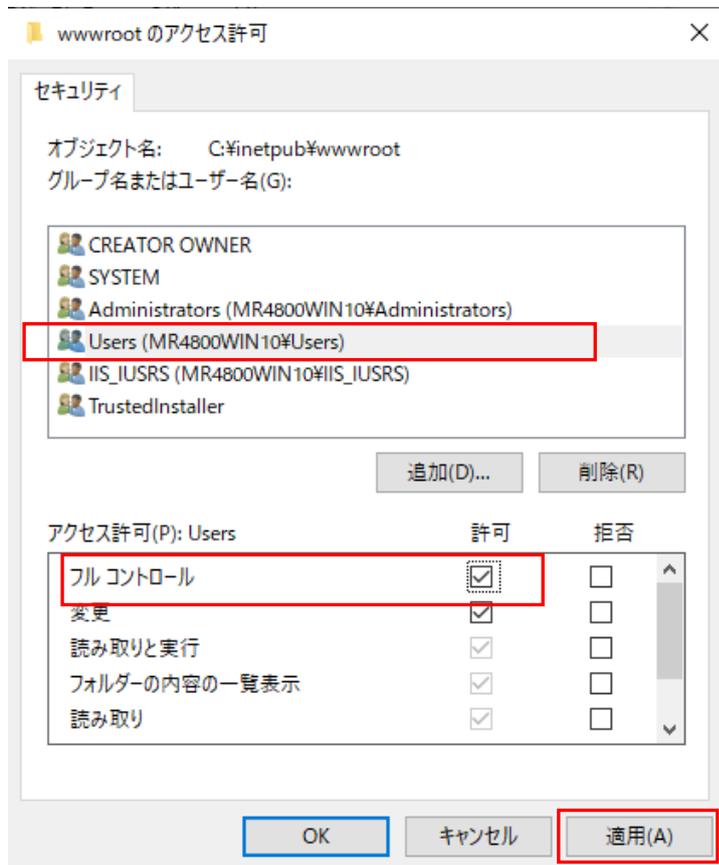
```
{"ShuyouKyorikm":576,"SaitanKyorikm":555,"Error":""}
```

2.3 発行・公開

2.3.1 ローカルフォルダ発行

発行先フォルダ C:\inetpub\wwwroot\lgsWebKiroTeiKyorio1

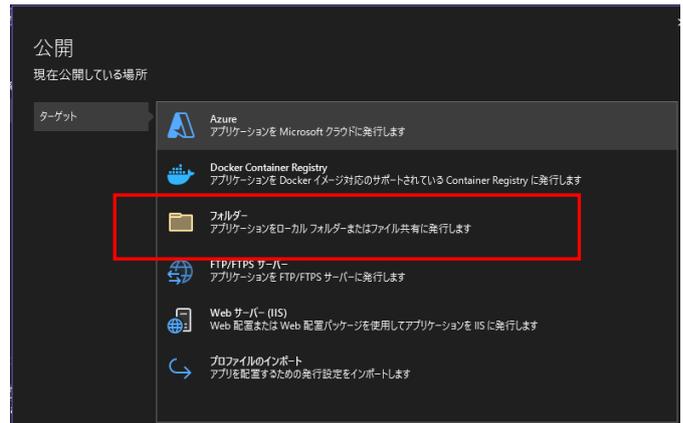
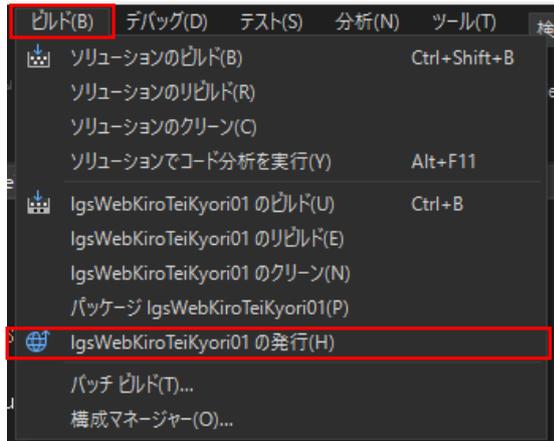
C:\inetpub\wwwroot\のセキュリティ Users で書き込み可能 設定



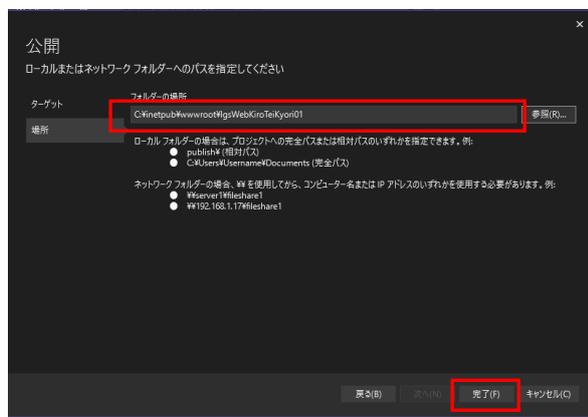
2. GitHub リポジトリのクローンから WebApi 発行

・ 5 - 2.3 発行・公開

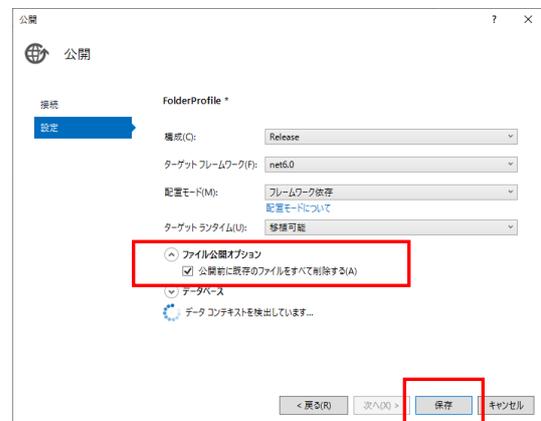
2.3.2 ビルド → 発行



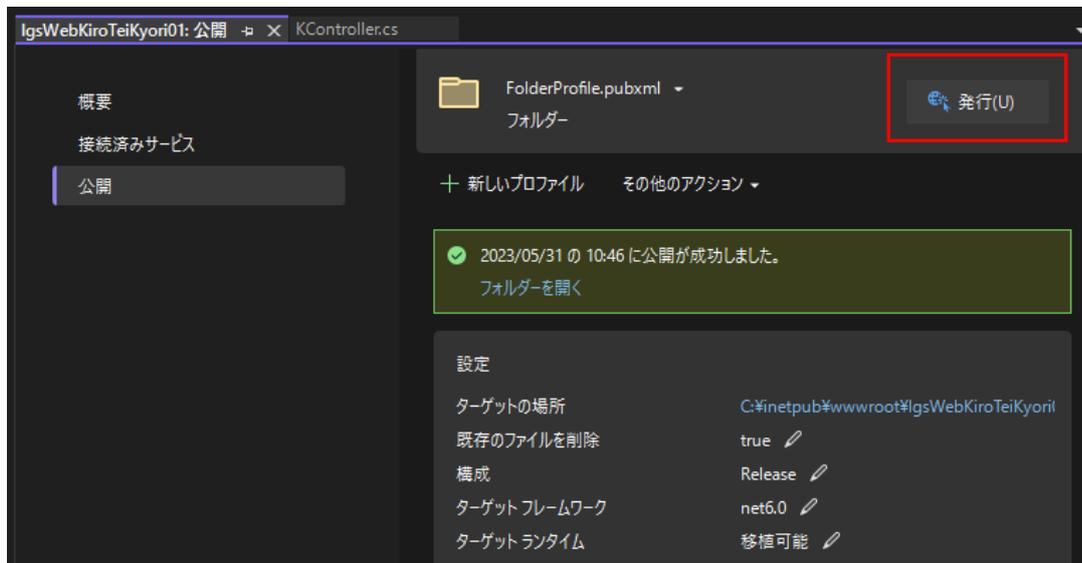
2.3.4 ターゲット C:\inetpub\wwwroot\IgsWebKiroTeiKyor01¥



2.3.5 公開前に既存のファイルをすべて削除する



2.3.6 発行



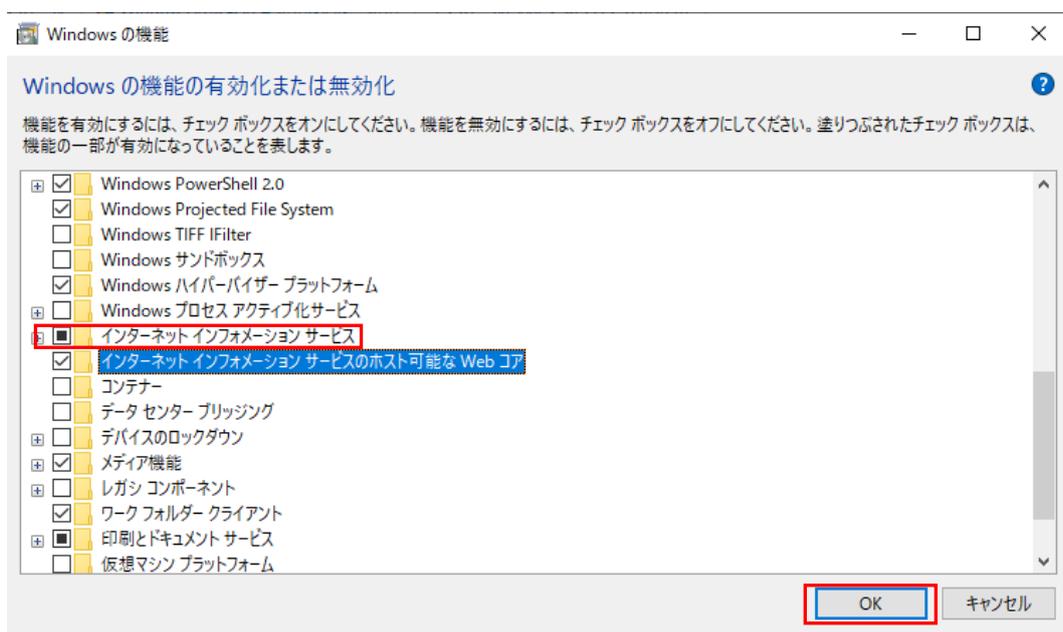
3.IIS で WebApi のアプリケーション設定

3.1 IIS 機能の有効化

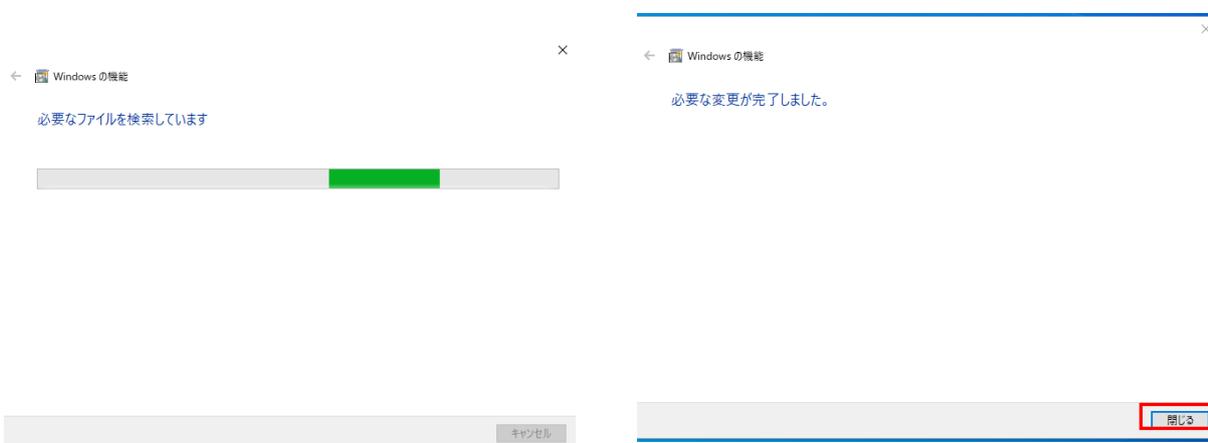
3.1.1. 設定でコントロールパネルを検索 Windows の機能の有効化



3.1.2 インターネットインフォメーションサービスのチェックボックスをオンにして OK する。



3.1.3 処理 完了



3.2 ASP.NET Core 6 Windows Hosting Bundle をインストールする

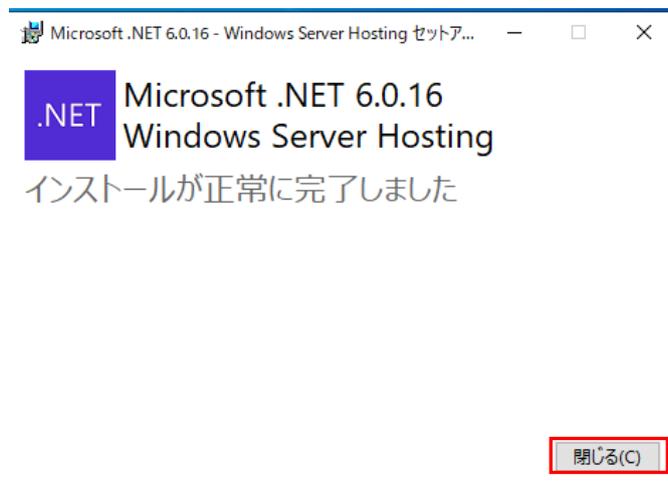
3.2.1 Microsoft .NET6 のダウンロードページ <https://dotnet.microsoft.com/en-us/download/dotnet/6.0> を開きます。
[ASP.NET Core Runtime 6.0.x]のセクションの[Windows]の[Hosting Bundle]をクリックします。

The screenshot shows the Microsoft .NET 6.0 download page. The main heading is "Download .NET 6.0". Under the "Run apps - Runtime" section for "ASP.NET Core Runtime 6.0.16", there is a table of installers. The "Windows" row has a "Hosting Bundle" link highlighted with a red box. Below this, the ".NET Desktop Runtime 6.0.16" section is partially visible.

3.2.2 インストーラファイルがダウンロードされます。

The screenshot shows a web browser window displaying the download page for the ASP.NET Core 6.0 Runtime (v6.0.16) - Windows Hosting Bundle Installer. The browser address bar shows the URL: <https://dotnet.microsoft.com/en-us/download/dotnet/thank-you/run...>. The page content includes a large heading: "Thanks for downloading ASP.NET Core 6.0 Runtime (v6.0.16) - Windows Hosting Bundle Installer!". Below this, there is a message: "ASP.NET Core Hosting Bundle contains everything you need to run existing web/ server apps. The bundle includes the .NET runtime, the ASP.NET Core runtime, and if installed on a machine with IIS it will also add the ASP.NET Core IIS Module. If you only want the .NET or ASP.NET Core runtime, you'll find them in .NET 6.0 Downloads." A download notification bubble is visible, showing the file "dotnet-hosting-6.0.16-win.exe".

3.2.3 ダウンロードしたインストーラファイル dotnet-hosting-6.0X-win.exe を実行します。
以降は画面指示に従って ASP.NET Core 6 Windows Hosting Bundle をインストールします。

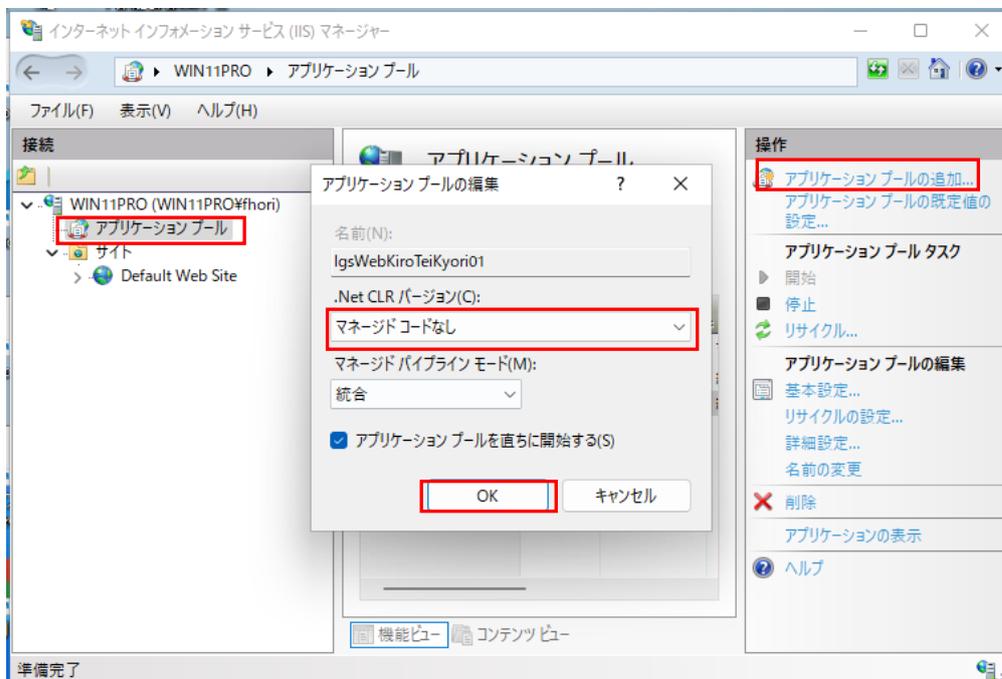


3.3 WebApi のアプリケーション設定

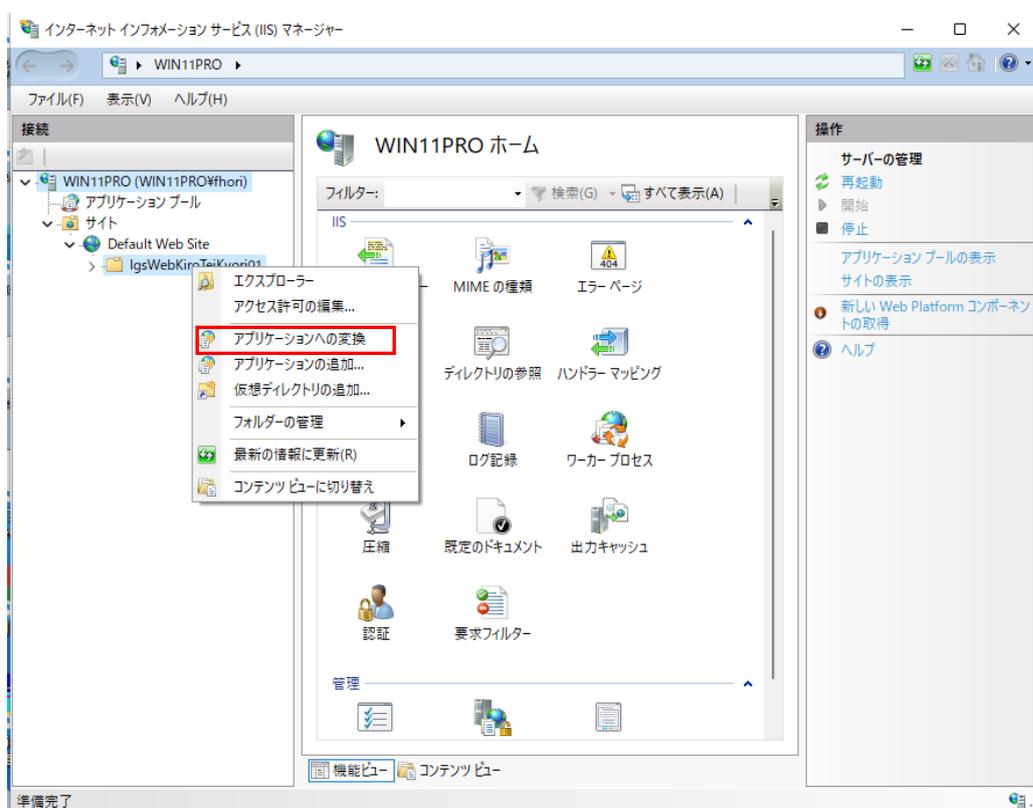
3.3.1 IIS の起動

Windows11 ではスタートボタン→すべてのアプリ→Windows ツール→インターネットインフォメーションサービス (IIS)
 Windows10 では Windows 管理ツール→インターネットインフォメーションサービス (IIS)

3.3.2 アプリケーションプールの追加で lgsWebKiroTeiKyori01 マネージコードなし で OK



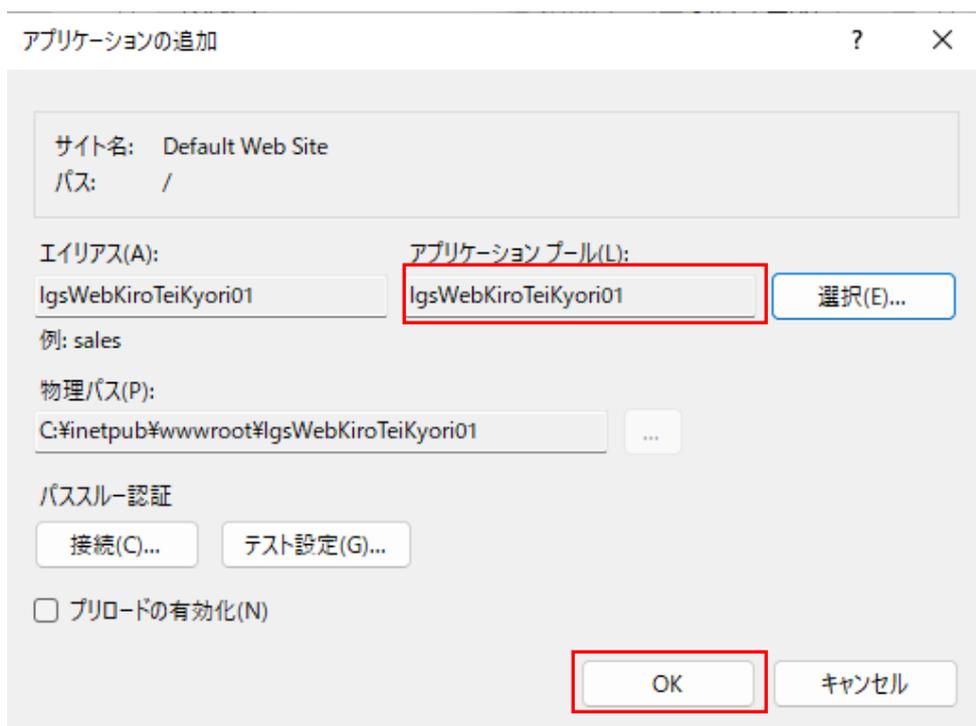
3.3.3 DefaultWebSite 下のフォルダ lgsWebKiroTeiKyori01 を右クリックしてメニューからアプリケーションへの変換を選択



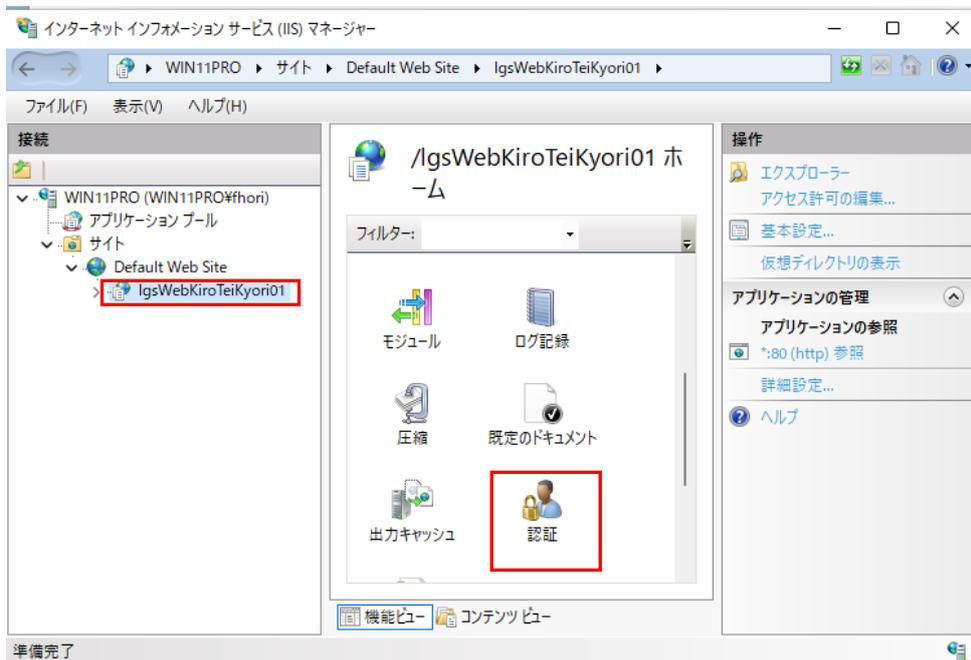
3.IIS の WebApi のアプリケーション設定

10 - 3.3 WebApi のアプリケーション設定

3.3.4 アプリケーションの追加でアプリケーションプールを lgsWebKiroTeiKyori01 として下図のように設定して OK



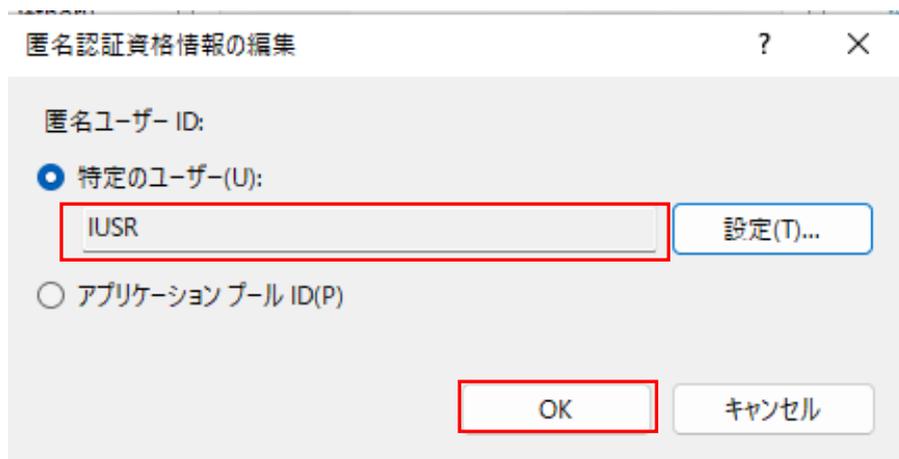
3.3.5 アプリケーション lgsWebKiroTeiKyori01 の認証を選択する



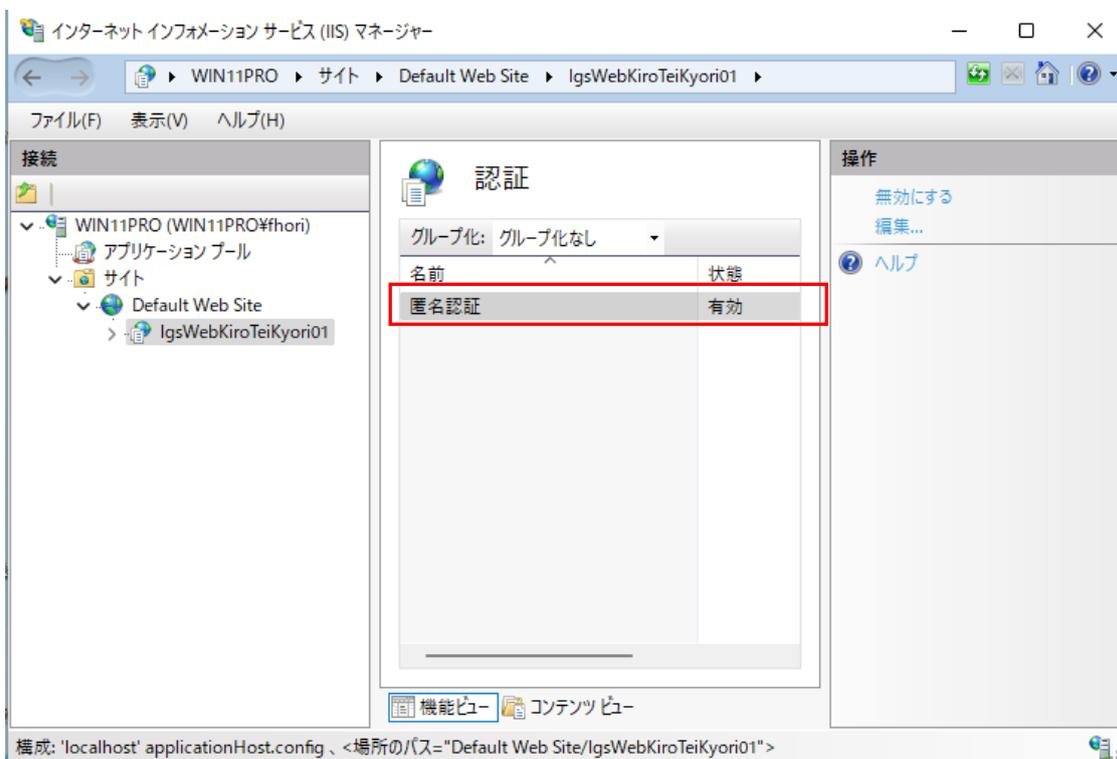
3.IIS で WebApi のアプリケーション設定

11 - 3.3 WebApi のアプリケーション設定

3.3.6 匿名認証資格情報の編集で特定のユーザーを IUSR と設定して OK



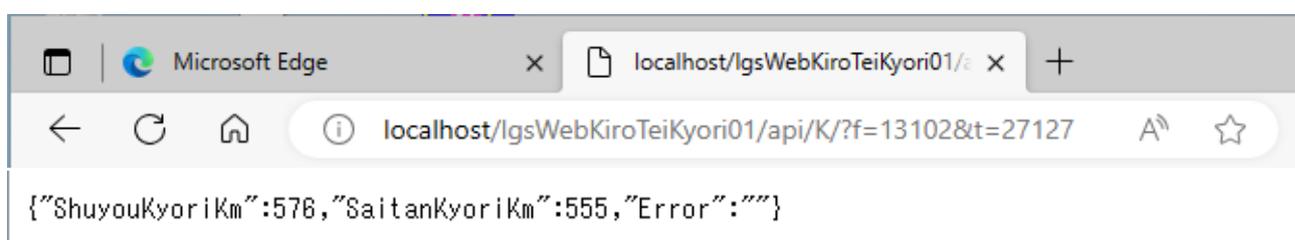
3.3.7 認証で匿名認証が有効を確認



3.3.8 WebApi lgsWebKiroTeiKyorio1 の動作確認

<http://localhost/lgsWebKiroTeiKyorio1/api/K/?f=13102&t=27127>

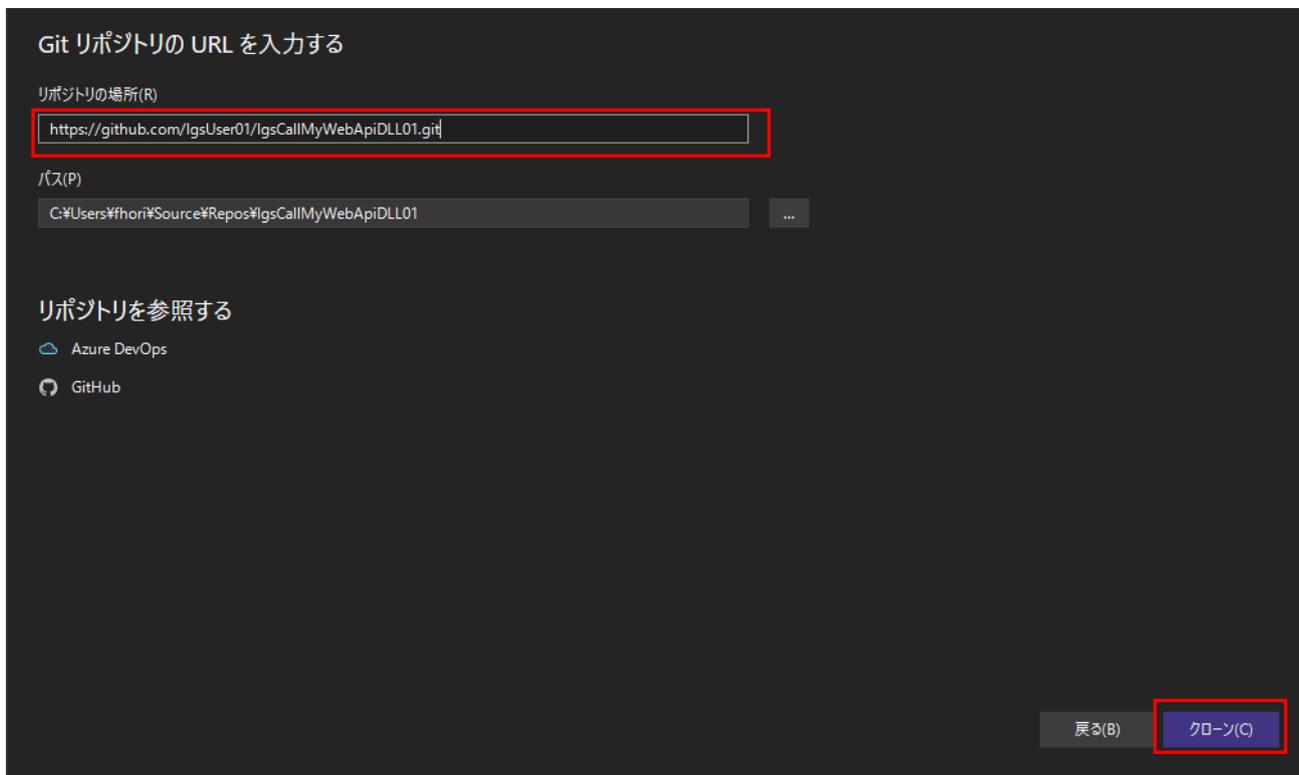
```
{"ShuyouKyorioKm":576,"SaitanKyorioKm":555,"Error":""}
```



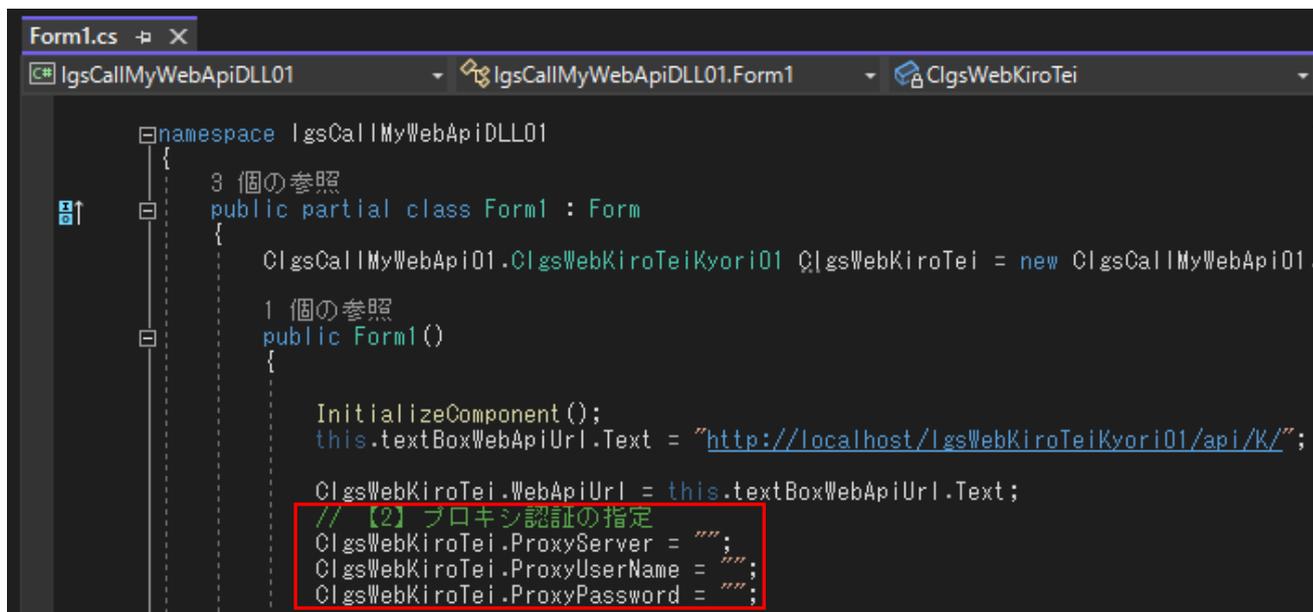
4 WebApi 呼び出し C#サンプル

4.1 WebApi DLL から呼び出し C#サンプル

4.1.1 リポジトリの場所に `https://github.com/lgsUser01/lgsCallMyWebApiDLL01.git` を入力してクローン



4.1.2 Form1.cs で必要ならプロキシ認証の指定をする



- 13 - 4.1 WebApi DLL から呼び出し C#サンプル
- 4.1.3 デバッグの開始

The screenshot shows a Windows application window titled "Form1" with a standard Windows title bar (minimize, maximize, close buttons). The main content area is titled "DLL呼び出し" (DLL Call), which is highlighted with a red rectangular box. Below the title, there are several input fields:

- WebApiURL**: A text box containing the URL `http://localhost/lgsWebKiroTeiKyor01/api/K/`.
- 起点** (Start Point): A text box containing the value `13102`.
- 終点** (End Point): A text box containing the value `27127`.
- 主要距離Km** (Main Distance Km): A text box containing the value `576`.
- 最短距離Km** (Shortest Distance Km): A text box containing the value `555`.
- Error**: A text box that is currently empty.

4.2 WebApi HttpClient から呼び出し C#サンプル

4.2.1 ChatGPT で「C#で WebAPI を呼び出すサンプルコードを作成してください」を問い合わせた回答です。

以下は、C#で WebAPI を呼び出すサンプルコードです。この例では、HTTP GET メソッドを使用して API エンドポイントにアクセスし、JSON 応答をシリアル化して.NET オブジェクトにデシリアル化します。

```
using System;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Threading.Tasks;
using Newtonsoft.Json;
public class Program
{
    static HttpClient client = new HttpClient();
    static async Task Main(string[] args)
    {
        await CallWebApi();
    }
    static async Task CallWebApi()
    {
        client.BaseAddress = new Uri("https://api.example.com/");
        client.DefaultRequestHeaders.Accept.Clear();
        client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));

        HttpResponseMessage response = await client.GetAsync("users");
        if (response.IsSuccessStatusCode)
        {
            string jsonString = await response.Content.ReadAsStringAsync();
            User[] users = JsonConvert.DeserializeObject<User[]>(jsonString);

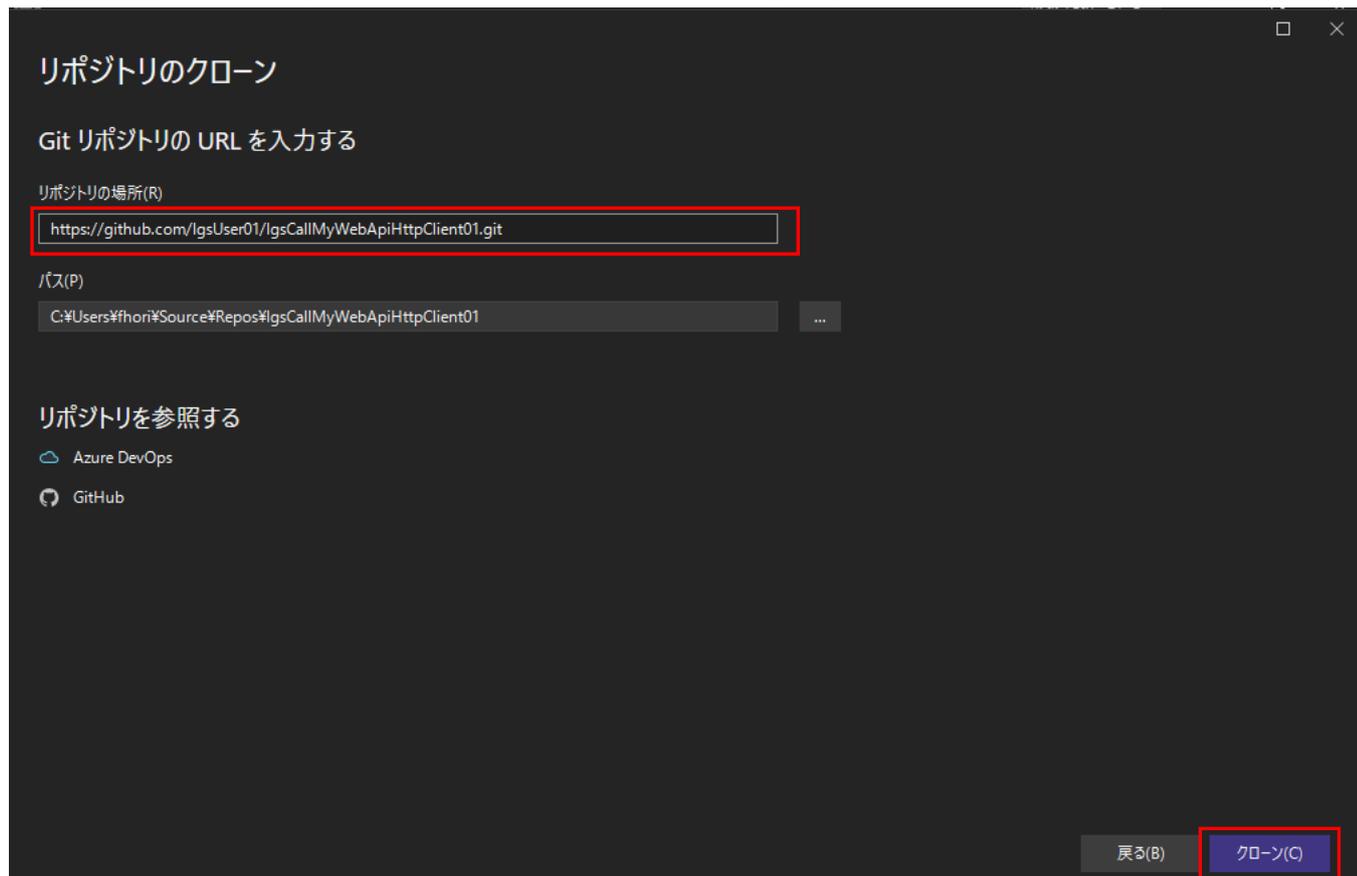
            foreach (User user in users)
            {
                Console.WriteLine($"ID: {user.Id}, Name: {user.Name}");
            }
        }
        else
        {
            Console.WriteLine($"Web API error. Status code: {response.StatusCode}");
        }
    }
}
public class User
{
    public int Id { get; set; }
    public string Name { get; set; }
}
```

この例では、HttpClient クラスを使用して API エンドポイントに HTTP GET リクエストを送信し、レスポンスを取得します。レスポンスが成功した場合は、レスポンス本文を string として取得し、JsonConvert クラスを使用してデシリアライズします。デシリアライズされたデータは、User オブジェクトの配列として取得されます。

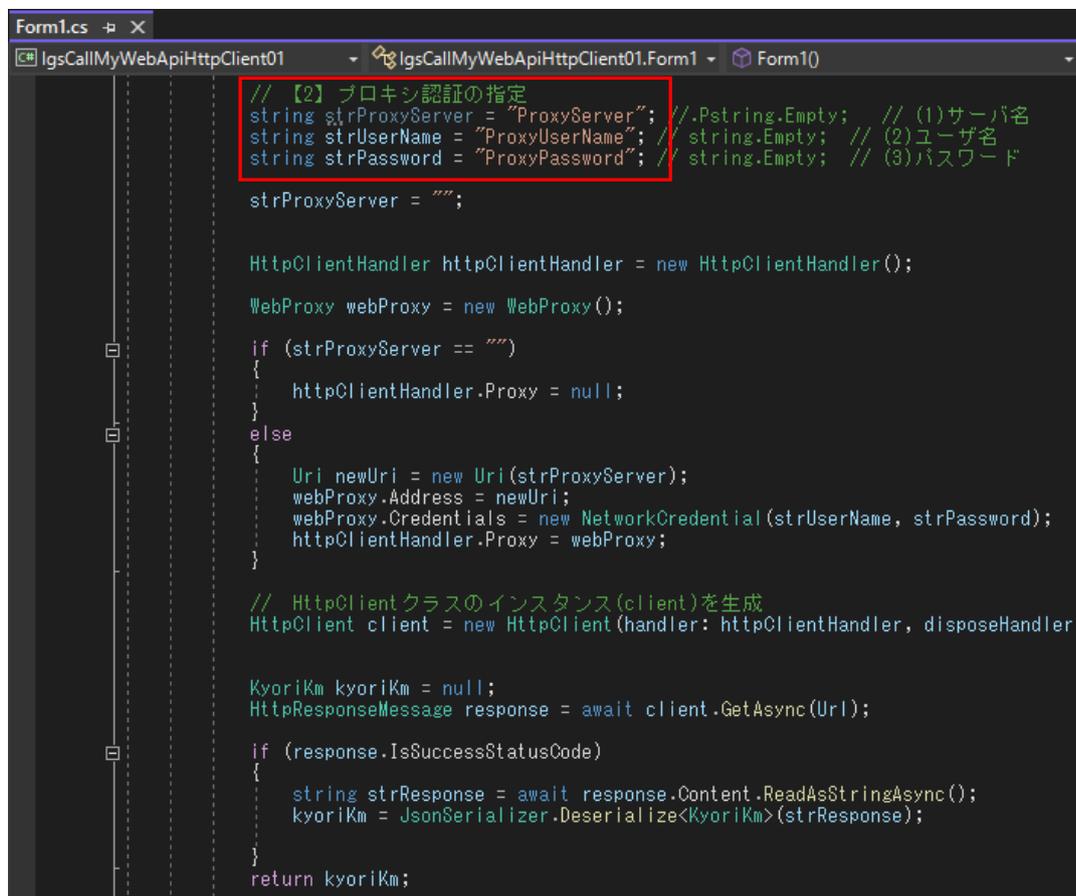
注意: リモート API を呼び出す前に、HttpClient を初期化するために必要なセキュリティ/認証/証明書情報を設定する必要があります。また、例外処理も含めて API 呼び出しを実装してください。

15 - 4.2 WebApi HttpClient から呼び出し C#サンプル

4.2.2 リポジトリの場所 <https://github.com/lgsUser01/lgsCallMyWebApiHttpClient01.git> を入力してクローン



4.2.3 Form1.cs で必要ならプロキシ認証の指定をする



4.2.4 デバッグの開始

