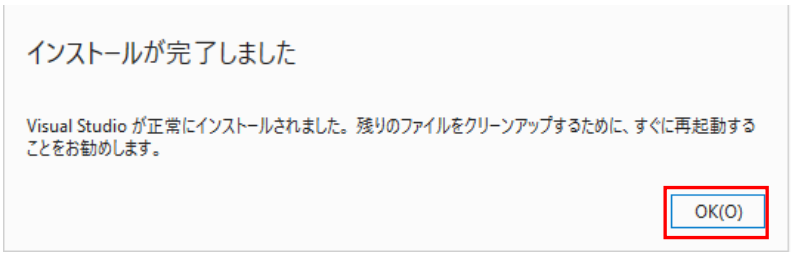
LogisticaTRUCKServer-Ⅰ 距離計算 WebApi 作成手順

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. IIS と ASP.NET Core 8 Windows Hosting Bundle のインストール

2.1 IIS 機能の有効化

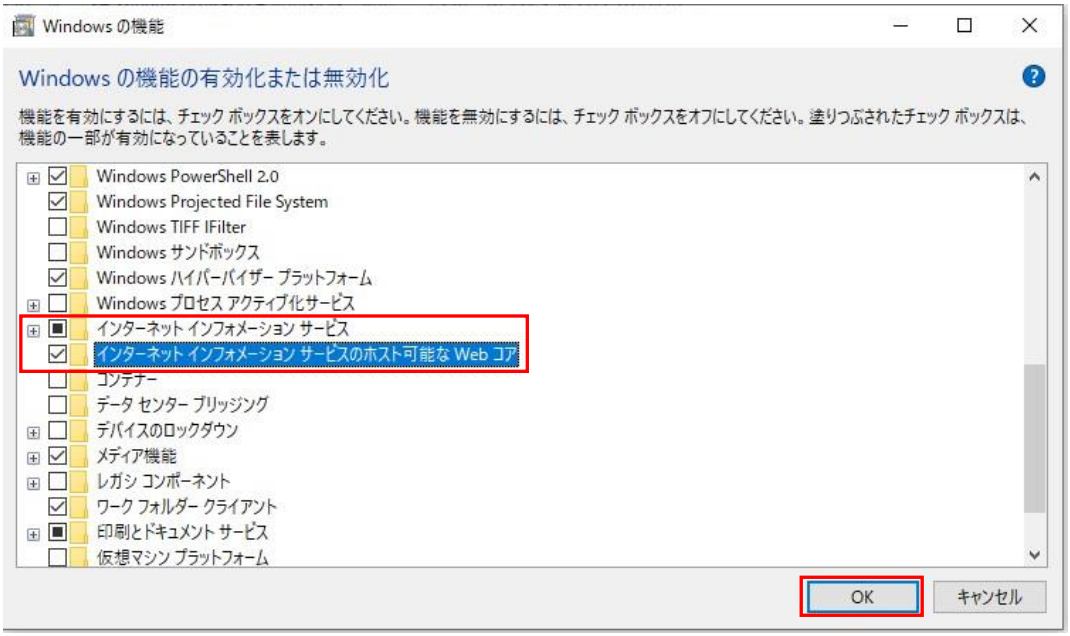
2.1.1. Windows11 の場合

スタートボタンをクリックし、検索窓に「コントロールパネル」と入力して、コントロールパネルを開き、

「プログラム」→「Windows の機能の有効化または無効化」をクリックします。

「**インターネットインフォーメーションサービス**」、「**インターネットインフォメーションサービスのホスト可能な Web コア**」の

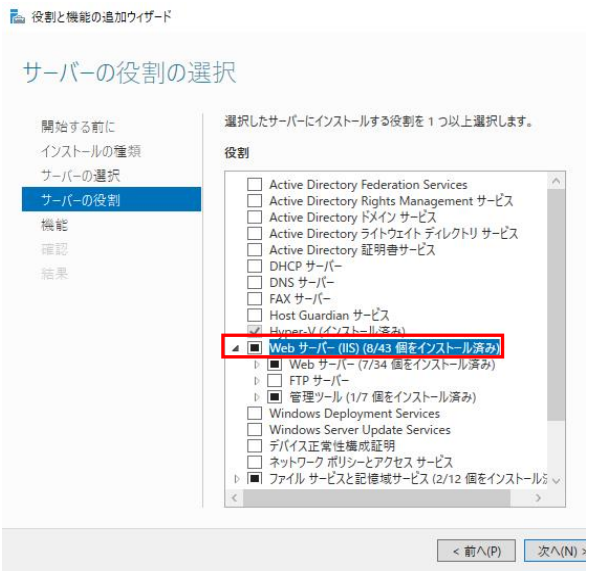
チェックボックスにチェックを入れて、「OK」をクリックします。

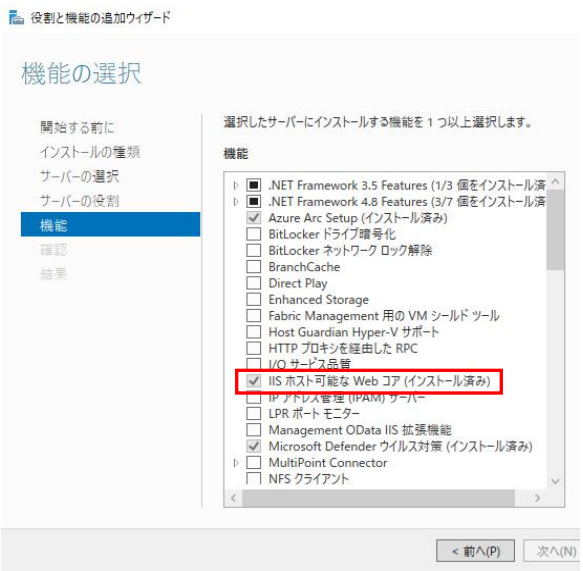


2.1.2 Windows Serrver2022 の場合

スタートボタンをクリックし、検索窓に「サーバマネージャー」と入力します。

「サーバマネージャー」→「役割と機能の追加ウィザード」で下記 2 画面のように選択し、画面指示に従いインストールします。





2.2 ASP.NET Core 8 Windows Hosting Bundle をインストールする

2.2.1 Microsoft .NET8 のダウンロードページを開きます。

https://dotnet.microsoft.com/en-us/download/dotnet/8.0

[ASP.NET Core ランタイム 8.0.22]のセクションの[Windows]の[Hosting Bundle]をダブルクリックします。

へ 8.0.22

リリース ノート　最新リリース日 2025年11月11日

アプリのビルド - SDK

SDK 8.0.416

OS	インストーラー	バイナリ
Linux	パッケージマネージャーの手順	Arm32 | Arm32 Alpine | Arm64 | Arm64 Alpine | x64 | x64 Alpine
macOS	Arm64 | x64	Arm64 | x64
Windows	x64 | x86 | Arm64 | winget の手順	x64 | x86 | Arm64
すべて	dotnet-install scripts	

付加済みランタイム
.NET Runtime 8.0.22
ASP.NET Core ランタイム 8.0.22
.NET デスクトップ ランタイム 8.0.22

言語サポート
C# 12.0
F# 8.0
Visual Basic 16.9

アプリの実行 - ランタイム

ASP.NET Core ランタイム 8.0.22

ASP.NET Core ランタイムを使用すると、既存の Web/サーバー アプリケーションを実行できます。 **Windows では、.NET ランタイムと IIS サポートを含むホスティング バンドルをインストールすることをお勧めします。**

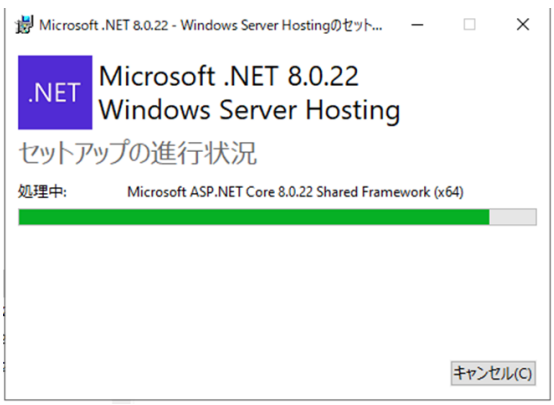
IIS ランタイム サポート (ASP.NET Core モジュール v2)
18.0.25301.22

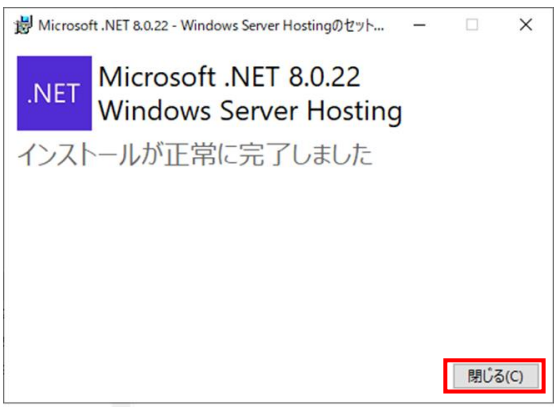
OS	インストーラー	バイナリ
Linux	パッケージマネージャーの手順	Arm32 | Arm32 Alpine | Arm64 | Arm64 Alpine | x64 | x64 Alpine
macOS		Arm64 | x64
Windows	x64 | x86 | Arm64 | Hosting Bundle | winget の手順	x64 | x86 | Arm64

.NET デスクトップ ランタイム 8.0.22

2.2.2 インストーラファイルがダウンロードされます。

2.2.3 ダウンロードしたインストーラファイル dotnet-hosting-dotnet-hosting-8.0.22-win.exe を実行します。
以降は画面指示に従って ASP.NET Core 8 Windows Hosting Bundle をインストールします。





3　GitHub リポジトリのクローンから WebApi 発行

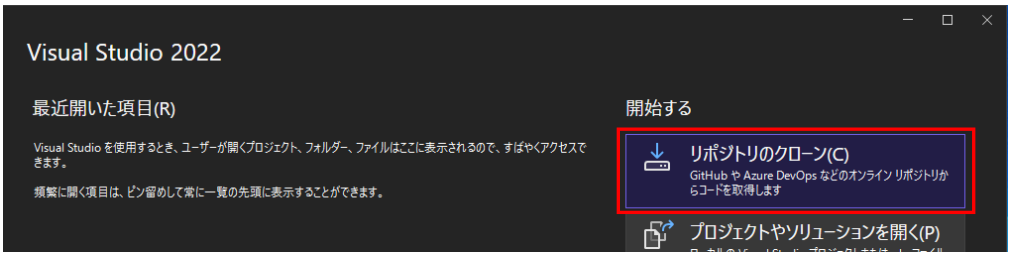
3.1　VS2022 ポジトリのクローン

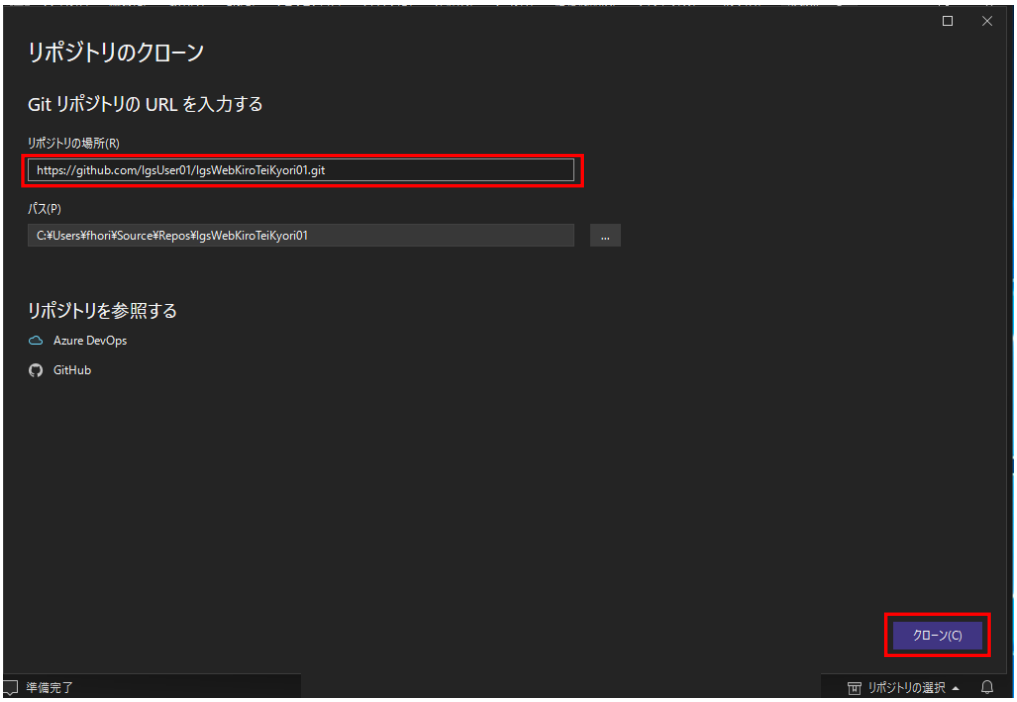
3.1.1 リポジトリのクローン

VS2022 を起動し、「開始する」の「リポジトリのクローン」をクリックします。

「リポジトリの場所」に下記 URL を入力し、「クローン」をクリックします。

https://github.com/lgsUser01/lgsWebKiroTeiKyori01.git



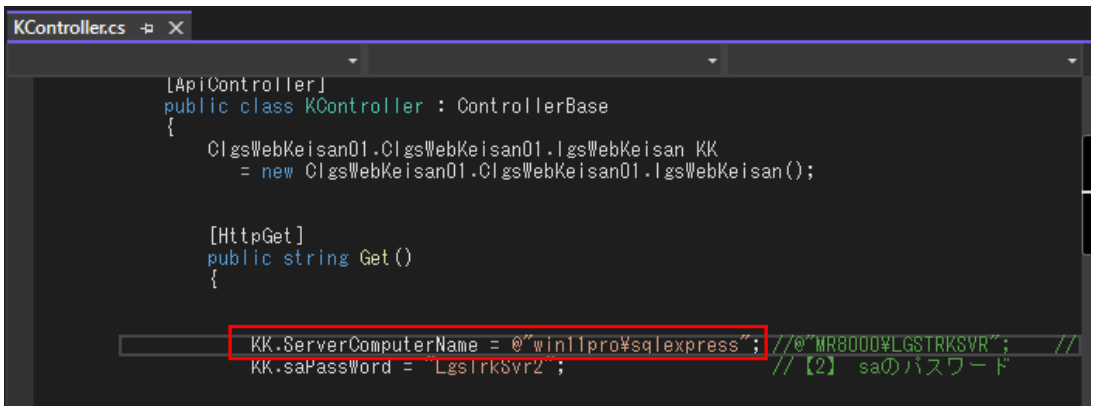


3.1.2 lgsWebKiroTeiKyori01.sln を開きます。

3.1.3 lgsWebKiroTeiKyori01 プロジェクトの ¥Contorolers¥KContorollers.cs で 1 か所変更します。

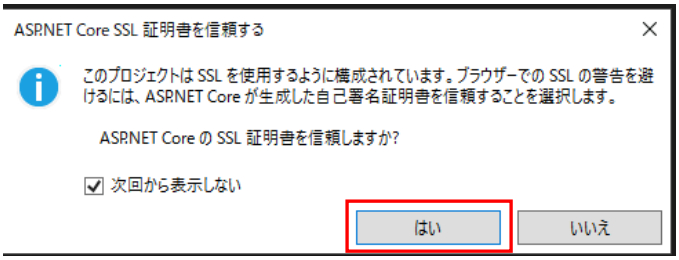
20 行目：距離計算 DB サーバの SQLServer インスタンス名を指定する

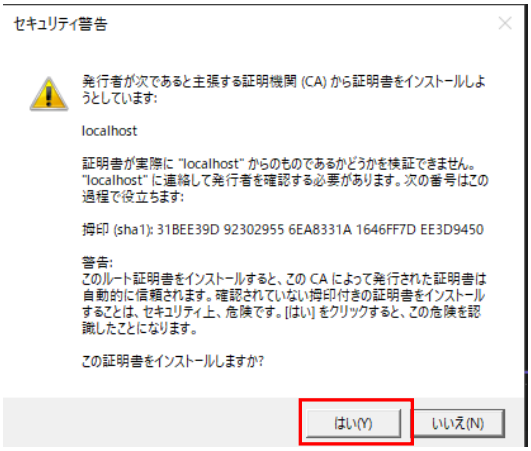
KK.ServerComputerName = @"**コンピュータ名** ¥lgstrksvr";



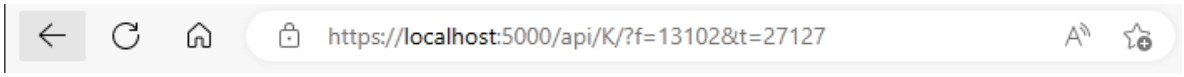
3.2 デバッグ

3.2.1 ソリューションエクスプローラーの「lgsWebKiroTeiKyori01」を右クリックして、「スタートアップ プロジェクトに設定」をクリックします。
デバッグの開始を実行し、証明書・セキュリティ関連のメッセージは「はい」をクリックします。





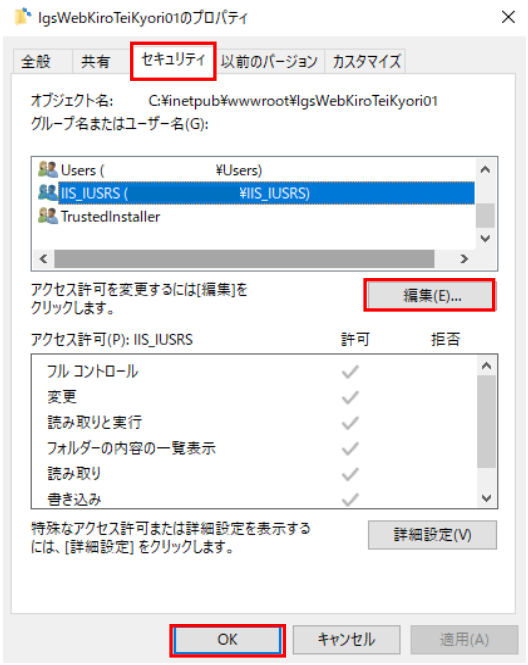
3.2.2 デバッグ　IIS Express Debug
ブラウザから下記 URL を入力します。
（東京都中央区:JIS 市区町村コード 13102　→　大阪市北区：JIS 市区町村コード 27127）
https://localhost:5000/api/K/?f=13102&t=27127
{"ShuyouKyoriKm":576,"SaitanKyoriKm":555,"Error":""}

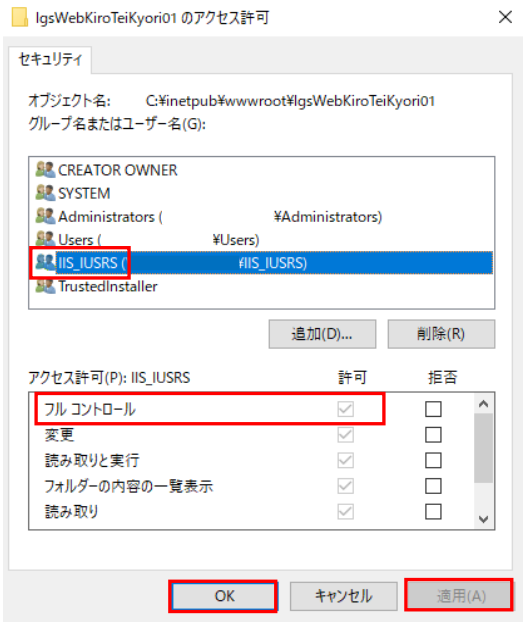


{"ShuyouKyoriKm":576,"SaitanKyoriKm":555,"Error":""}

3.3 発行・公開

3.3.1 発行先ローカルフォルダ 作成
発行先フォルダ C:¥inetpub¥wwwroot¥lgsWebKiroTeiKyori01 を作成します。
作成した C:¥inetpub¥wwwroot¥ lgsWebKiroTeiKyori01 フォルダを右クリックし、
「プロパティ」→「セキュリティ」→「編集」をクリックし、
グループ名またはユーザー名で「IIS_IUSRS」を選択し、アクセス許可をフルコントロールにチェックを入れ、
「適用」→「OK」→「OK」をクリックします。



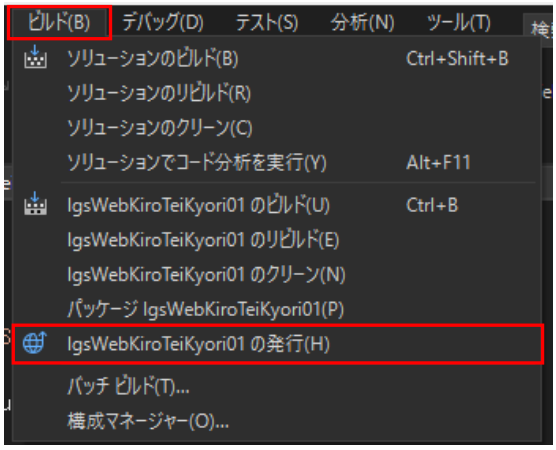


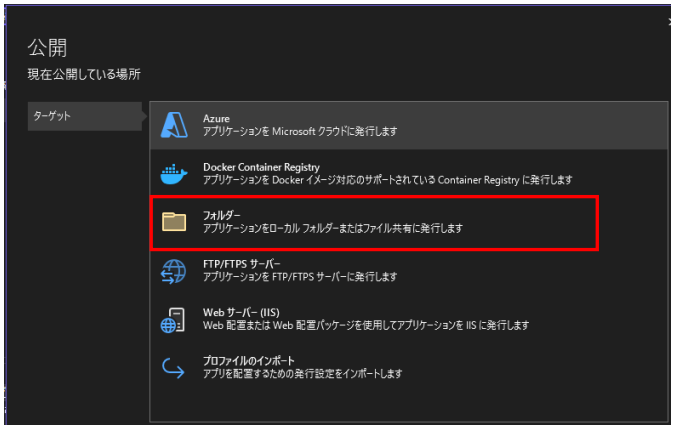
3.3.2 ビルド　→　発行

ターゲット C:¥inetpub¥wwwroot¥ lgsWebKiroTeiKyori01¥　で発行・公開します。

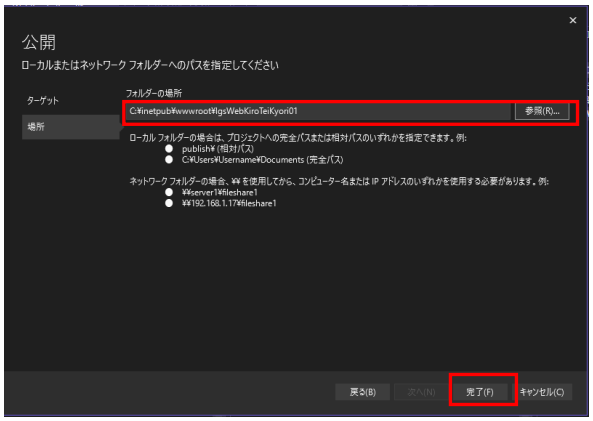
VS2022 で lgsWebKiroTeiKyori01.sln を開き、「ビルド」　→　「lgsWebKiroTeiKyori01 の発行」をクリックします。

ターゲット:フォルダー、場所:C:¥inetpub¥wwwroot¥lgsWebJitsuiKyori01 で完了をクリックします。





「参照」から C:¥inetpub¥wwwroot¥lgsWebKiroTeiKyori01¥を選択し、「完了」をクリックします。



「すべての設定を表示」をクリックし、下記画面のように選択します。

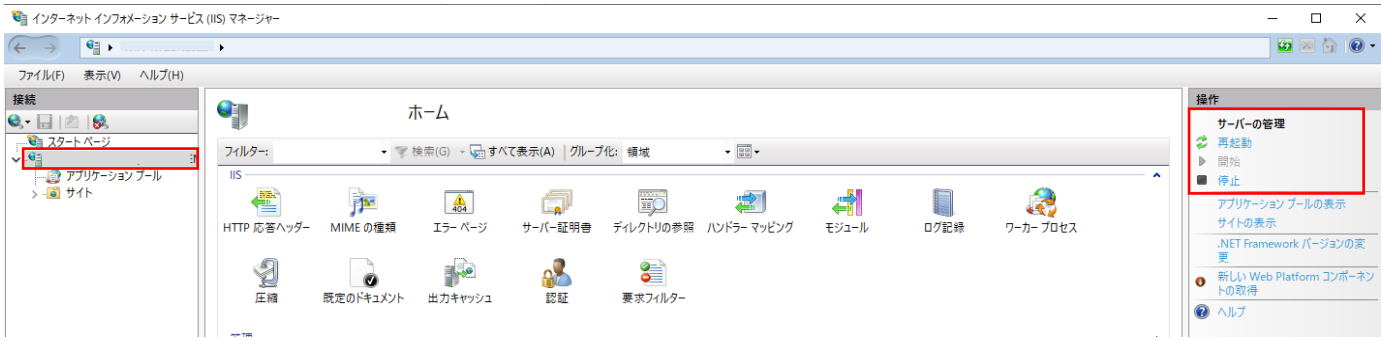
ファイル公開オプションでは「公開前に既存のファイルをすべて削除」にのみチェックをし、「保存」→「発行」をクリックします。

4 IIS WebApi のアプリケーション設定

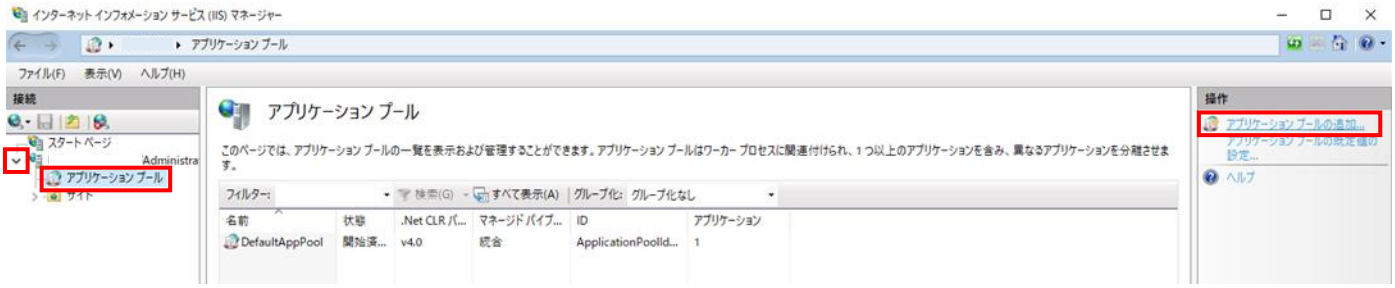
4.1.1 IIS の起動

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ーメーションサービス(IIS) を選択します。
Windows Serrver2022 ではスタートボタンをクリックし、検索窓に「サーバマネージャー」と入力します。

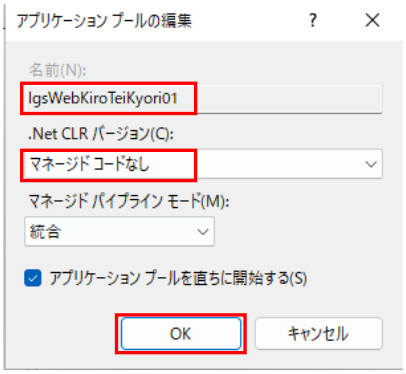
「サーバマネージャー」→「ツール」→「インターネットインフォーメーションサービス(IIS)」→「開始」



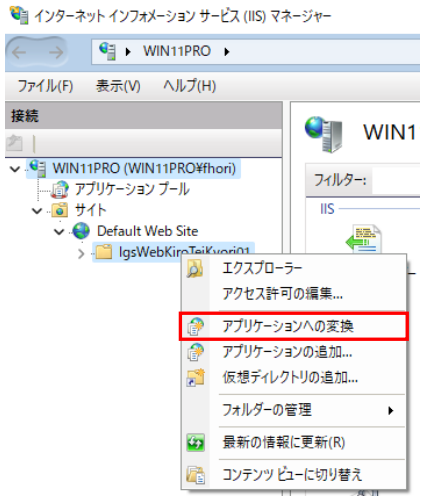
4.1.2 アプリケーションプールの追加



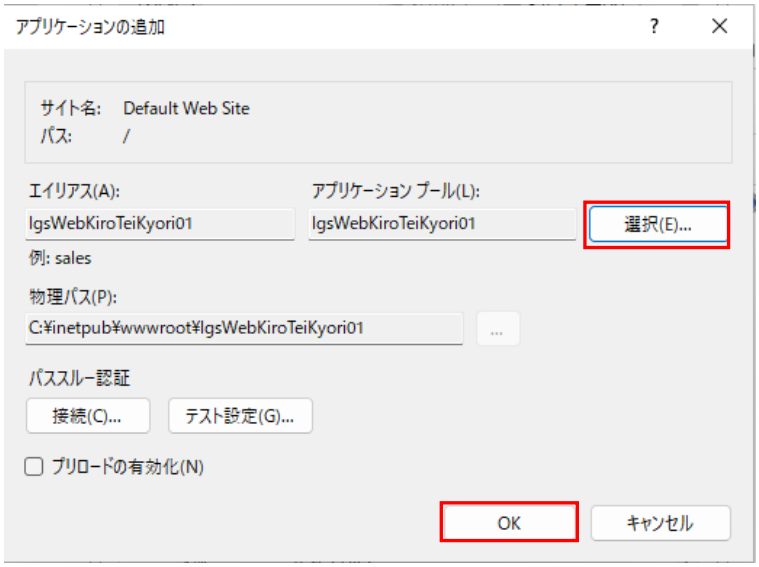
「アプリケーションプールの追加」画面で
名前：lgsWebKiroTeiKyori01 、.Net CLR バージョン：マネージコードなし　を選択し、「OK」をクリックします。



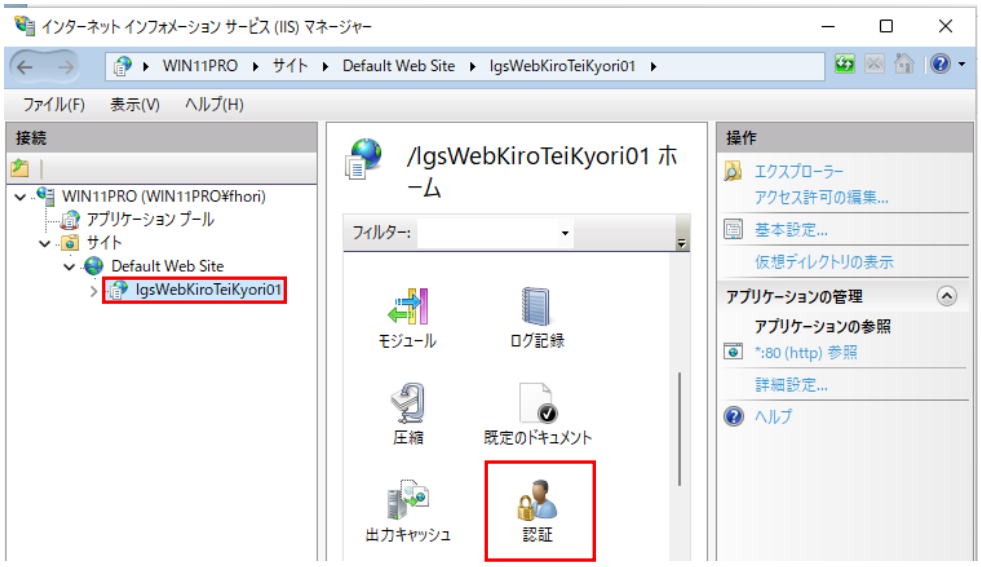
4.1.3 DefaultWebSite 下のフォルダ lgsWebKiroTeiKyori01 を右クリックし、「アプリケーションへの変換」をクリックします。



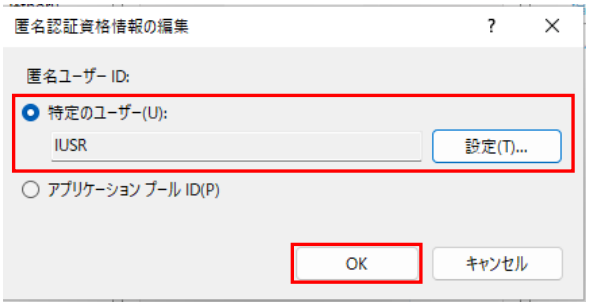
4.1.4「選択」からアプリケーションプールを lgsWebKiroTeiKyori01 に設定して「OK」をクリックします。



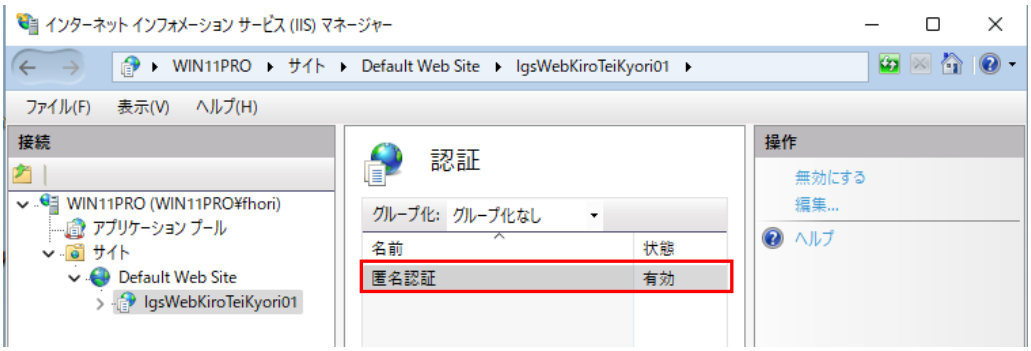
4.1.5 アプリケーション lgsWebKiroTeiKyori01 の認証を選択する



4.1.6 匿名認証の「編集」で特定のユーザー：「IUSR」と設定して「OK」 をクリックします。



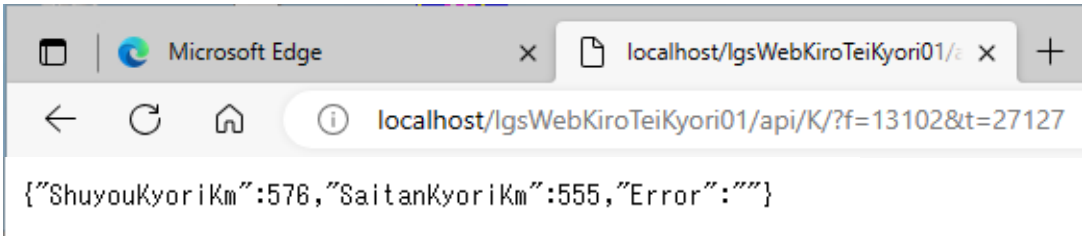
4.1.7 認証で匿名認証が有効を確認



4.1.8 WebApi　lgsWebKiroTeiKyori01　の動作確認

http://localhost/lgsWebKiroTeiKyori01/api/K/?f=13102&t=27127

{"ShuyouKyoriKm":576,"SaitanKyoriKm":555,"Error":""}

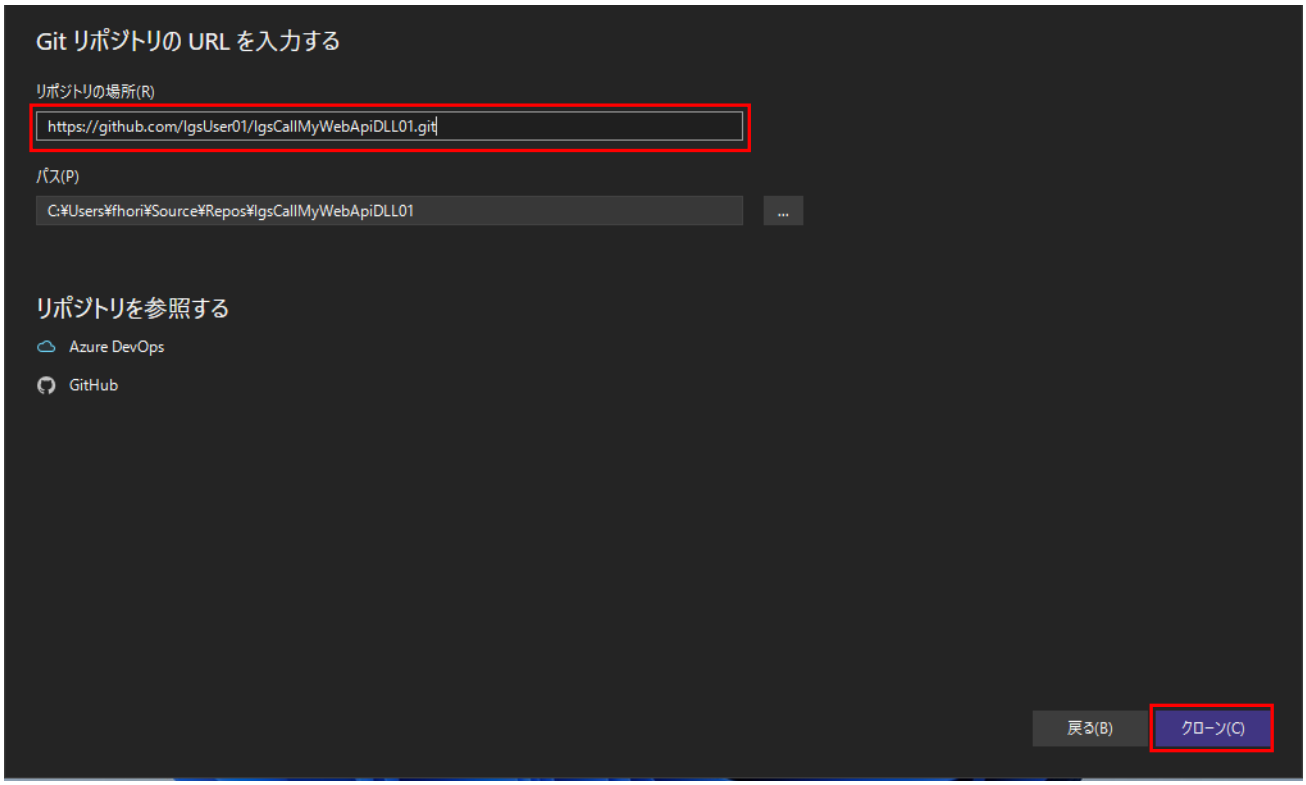


5 WebApi 呼び出し C#サンプル

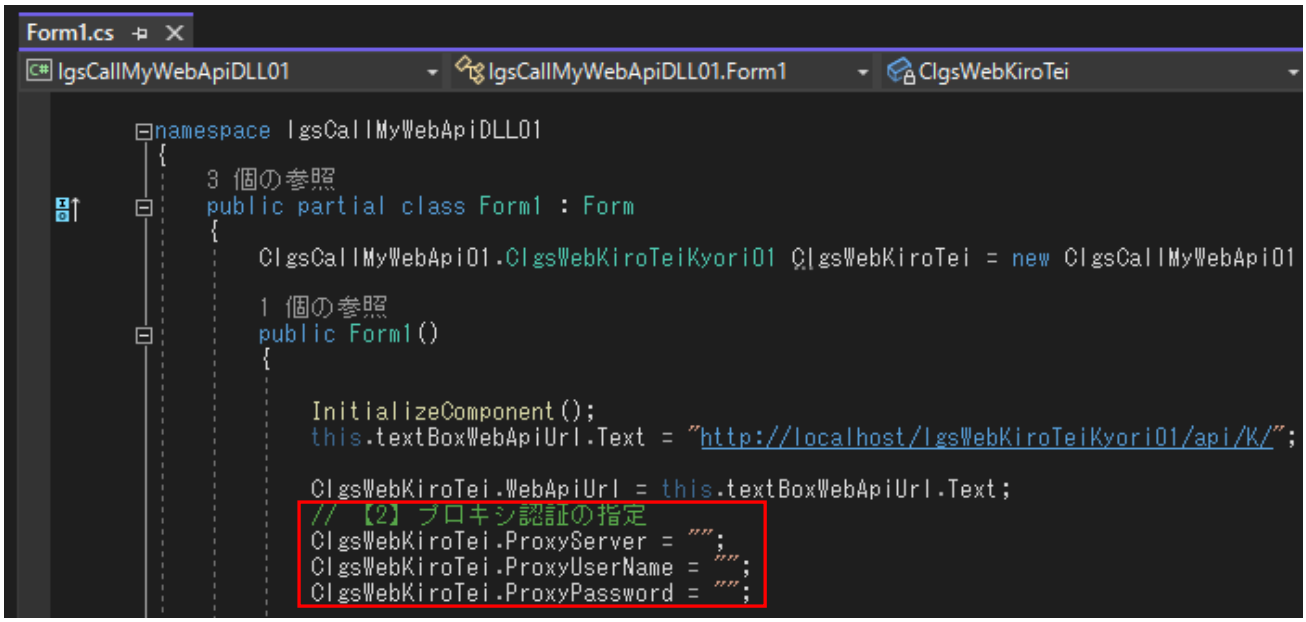
5.1 WebApi DLL から呼び出し C#サンプル

5.1.1 「リポジトリの場所」に下記 URL を入力し、「クローン」をクリックします。

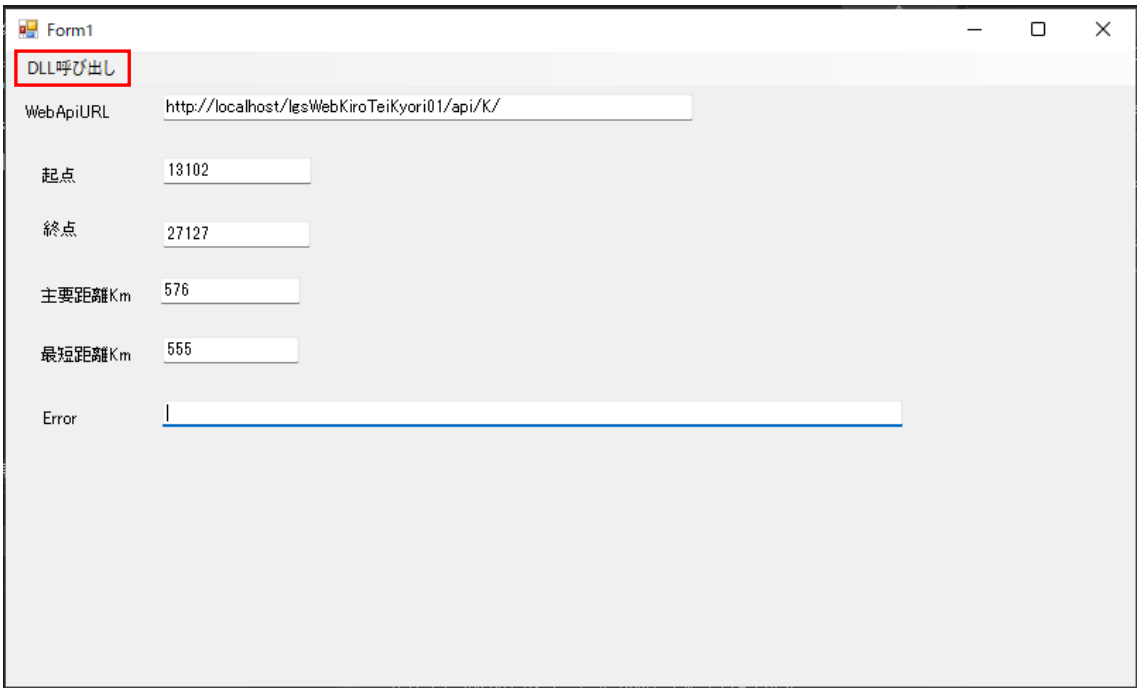
https://github.com/lgsUser01/lgsCallMyWebApiDLL01.git



5.1.2 プロキシ認証が必要な場合、Form1.cs の 25-27 行目に指定します。



5.1.3 デバッグの開始を実行します。Form1 画面の「DLL 呼び出し」をクリックし、
「主要距離 Km」，「最短距離 Km」それぞれに計算結果が表示されることを確認します。



5.2 WebApi HttpClient から呼び出し C#サンプル

(参考) 5.2.1 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。

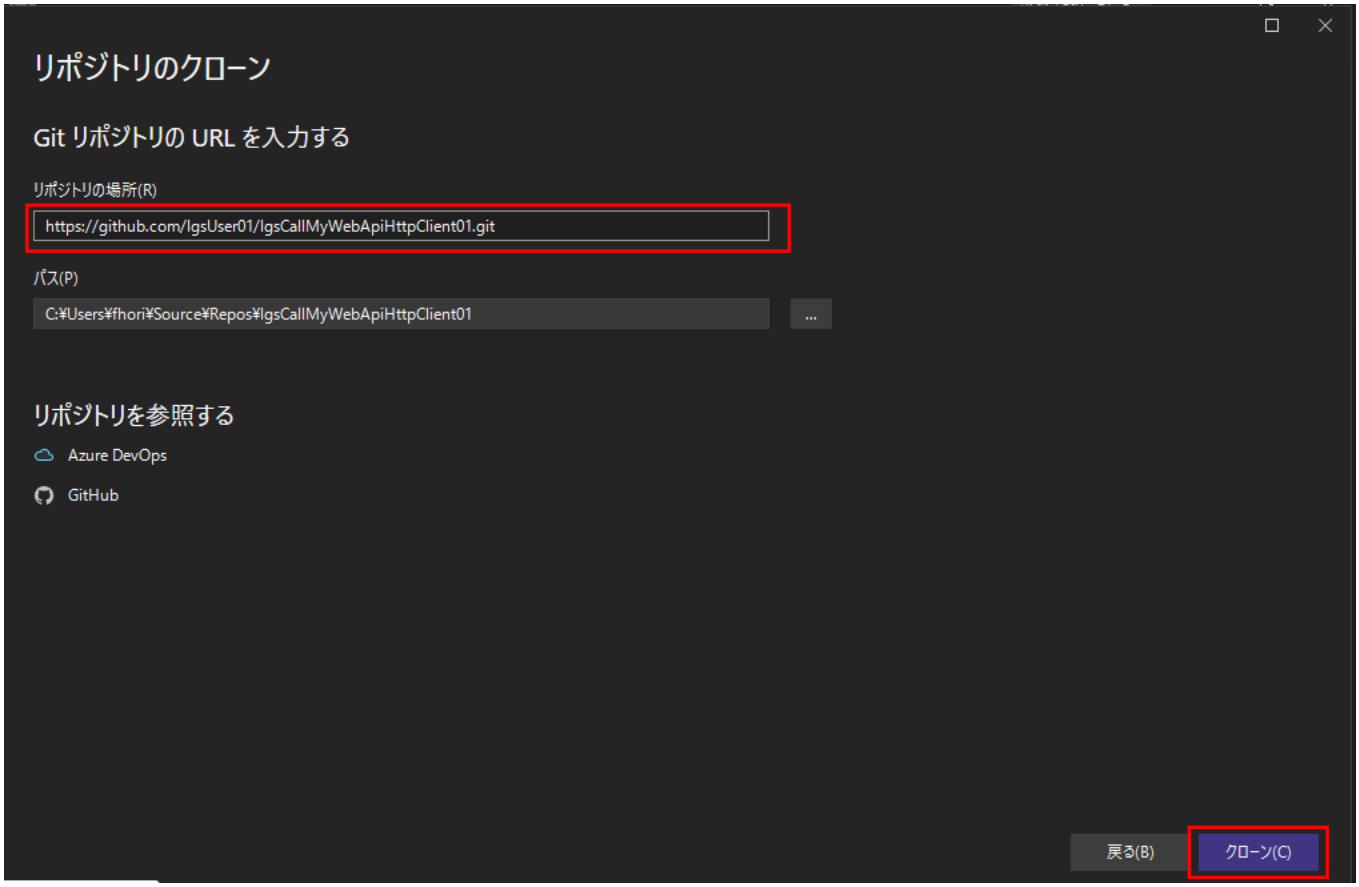
```
using System;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Threading.Tasks;
using Newtonsoft.Json;
public class Program
{
    static HttpClient client = new HttpClient();
    static async Task Main(string[] args)
    {
        await CallWebApi();
    }
    static async Task 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。

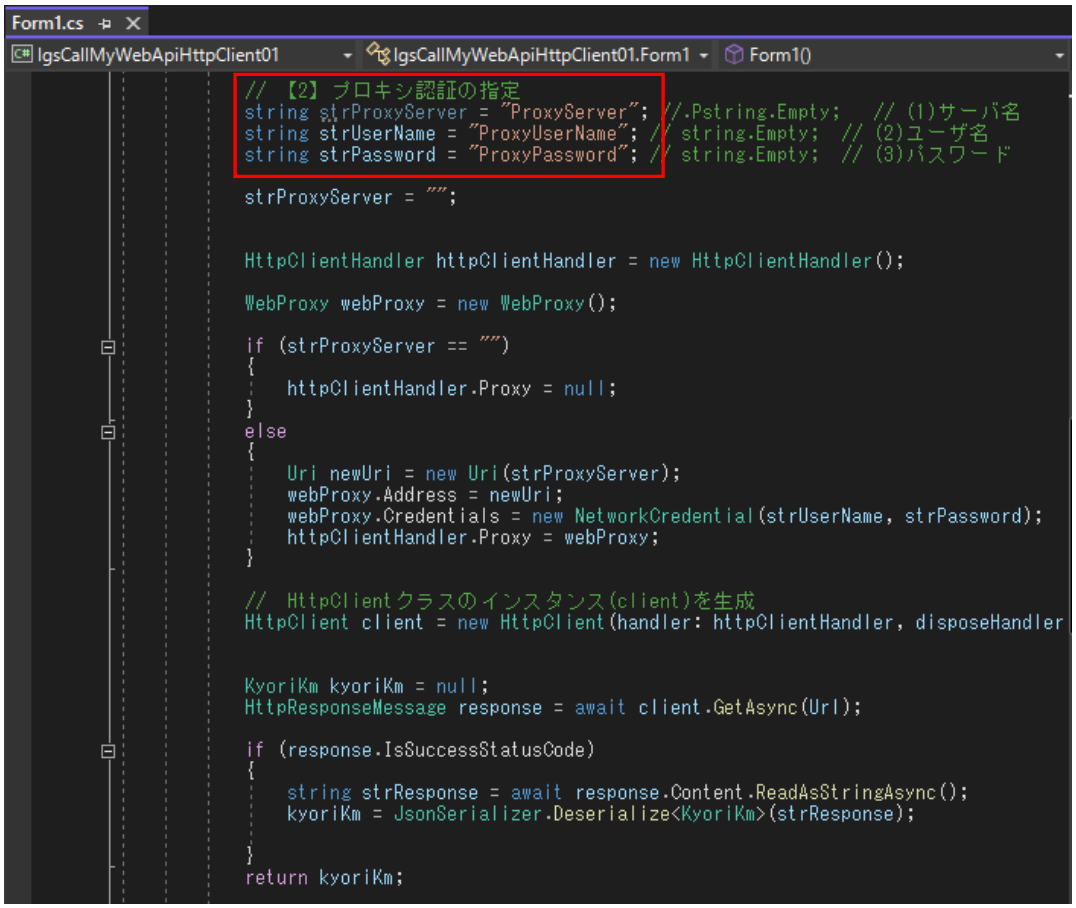
注意: リモート API を呼び出す前に、HttpClient を初期化するために必要なセキュリティー/認証/証明書情報を設定する必要があります。また、例外処理も含めて API 呼び出しを実装してください。

5.2.2「リポジトリの場所」に下記 URL を入力し、「クローン」をクリックします。

https://github.com/lgsUser01/lgsCallMyWebApiHttpClient01.git



5.2.3 プロキシ認証が必要な場合、Form1.cs の 61-63 行目に指定します。



5.2.4 デバッグの開始を実行します。Form1 画面の「HttpClient 呼び出し」をクリックし、
「主要距離 Km」,「最短距離 Km」それぞれに計算結果が表示されることを確認します。