

メシウス株式会社

Cloud Functions と DioDocs で Excel や PDF ファイルを出力する

2024 年 4 月 8 日

※本資料は、[弊社ブログ](#)に投稿された記事「[Cloud Functions と DioDocs で Excel や PDF ファイルを出力する \(1\) ~ \(3\)](#)」の連載記事をまとめて資料化した内容となります。下記が記事の原文となります。

[Cloud Functions と DioDocs で Excel や PDF ファイルを出力する \(1\)](#)

[Cloud Functions と DioDocs で Excel や PDF ファイルを出力する \(2\)](#)

[Cloud Functions と DioDocs で Excel や PDF ファイルを出力する \(3\)](#)

目次

Cloud Functions と DioDocs で Excel や PDF ファイルを出力する (1)	3
Cloud Functions とは？	3
実装する内容	3
事前準備	3
Google Cloud でプロジェクトの作成、選択、サービスの有効化	3
Cloud SDK のインストールと初期化	3
.NET の開発環境の設定	3
テンプレートパッケージのインストール	3
Cloud Functions アプリケーションを作成	4
NuGet パッケージを追加	5
DioDocs for Excel を使うコードを追加	6
デバッグ実行で確認	7
Google Cloud へデプロイ	8
デプロイしたアプリケーションを確認	9
PDF を出力するには？	11
さいごに	12
Cloud Functions と DioDocs で Excel や PDF ファイルを出力する (2)	13
実装する内容	13
事前準備	13
Cloud Functions アプリケーションを作成	13
NuGet パッケージの追加	15
DioDocs for Excel	15
Google.Cloud.Storage.V1 (Google Cloud Client Libraries for .NET)	15
Cloud Storage にバケットを作成	16
DioDocs for Excel を使うコードを追加	17
デバッグ実行で確認	17
Google Cloud へデプロイ	19
デプロイしたアプリケーションを確認	20
PDF を出力するには？	21
さいごに	22
Cloud Functions と DioDocs で Excel や PDF ファイルを出力する (3)	23
セルに追加するテキストの日本語フォント (DioDocs for Excel)	23
ワークシートを PDF 出力する際の日本語フォント (DioDocs for Excel)	23

Visual Studio で開発する際の注意点.....	25
PDF ドキュメントを保存する際の日本語フォント (DioDocs for PDF)	26

Cloud Functions と DioDocs で Excel や PDF ファイルを出力する (1)

本記事では、Google Cloud で提供されている FaaS、Cloud Functions で「[DioDocs \(ディオドック\)](#)」を使用した C# (.NET 6、8) の関数アプリケーションを作成し、Excel や PDF ファイルを出力する方法について紹介します。

Cloud Functions とは？

[Cloud Functions](#) は [Google Cloud](#) で提供されている、各種イベントをトリガーに処理を実行するサーバーレスなアプリケーションを作成できるクラウドサービスです。

実装する内容

今回実装する内容は非常にシンプルです。関数アプリケーションで [HTTP 関数](#) を作成します。この関数の実行時に DioDocs を使用して Excel と PDF ファイルを作成し、HTTP リクエストのクエリパラメータで受け取った文字列を追加します。その後、作成した Excel と PDF ファイルを関数から HTTP レスポンスで直接ローカルへ出力する、といった内容です。

事前準備

Google Cloud でプロジェクトの作成、選択、サービスの有効化

「[クイックスタート: .NET を使用して HTTP Cloud Functions の関数を作成してデプロイする](#)」の「[gcloud CLI を使用した GCP プロジェクトの作成](#)」の手順 1~4 のとおりに、Google Cloud プロジェクトを作成、選択して「Cloud Functions API」サービスと「Cloud Build API」サービスを有効にします。

Cloud SDK のインストールと初期化

「[gcloud CLI を使用した GCP プロジェクトの作成](#)」の手順 5「gcloud CLI をインストールして初期化します。」を実施します。このリンクからは Cloud SDK のトップページにリンクされているだけで少々わかりづらいのですが、ここからリンクされている「[Cloud SDK のインストール](#)」の手順に沿って Cloud SDK をインストール、初期化します。

これで gcloud CLI が使えるようになったのでその後、手順 6「`gcloud` コンポーネントを更新してインストールします。」を実行しておきます。

.NET の開発環境の設定

「[gcloud CLI を使用した GCP プロジェクトの作成](#)」の手順 7「開発環境を準備します。」を実施します。

テンプレートパッケージのインストール

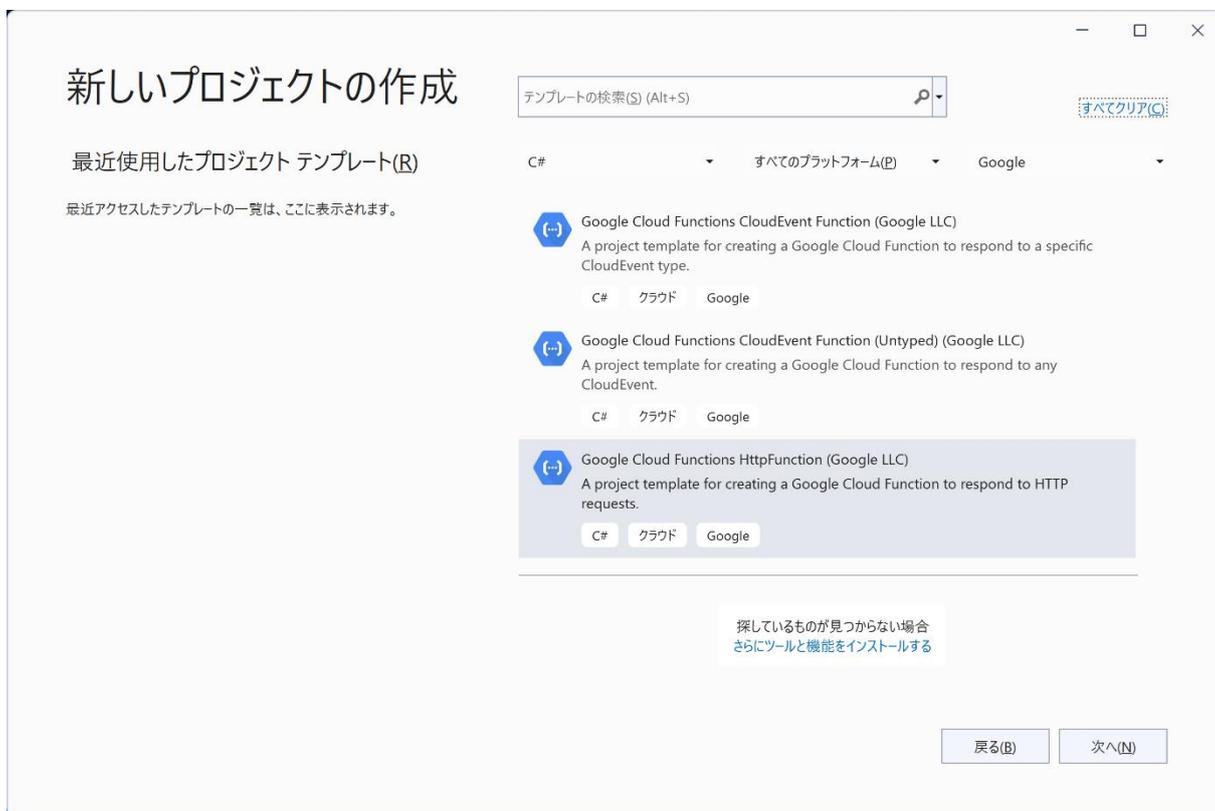
`dotnet new install` コマンドを使用して Cloud Functions のプロジェクトテンプレートをインストールしておきます。「[テンプレート パッケージの使用](#)」に記載の以下のコマンドを実行します。

```
dotnet new -i Google.Cloud.Functions.Templates
```

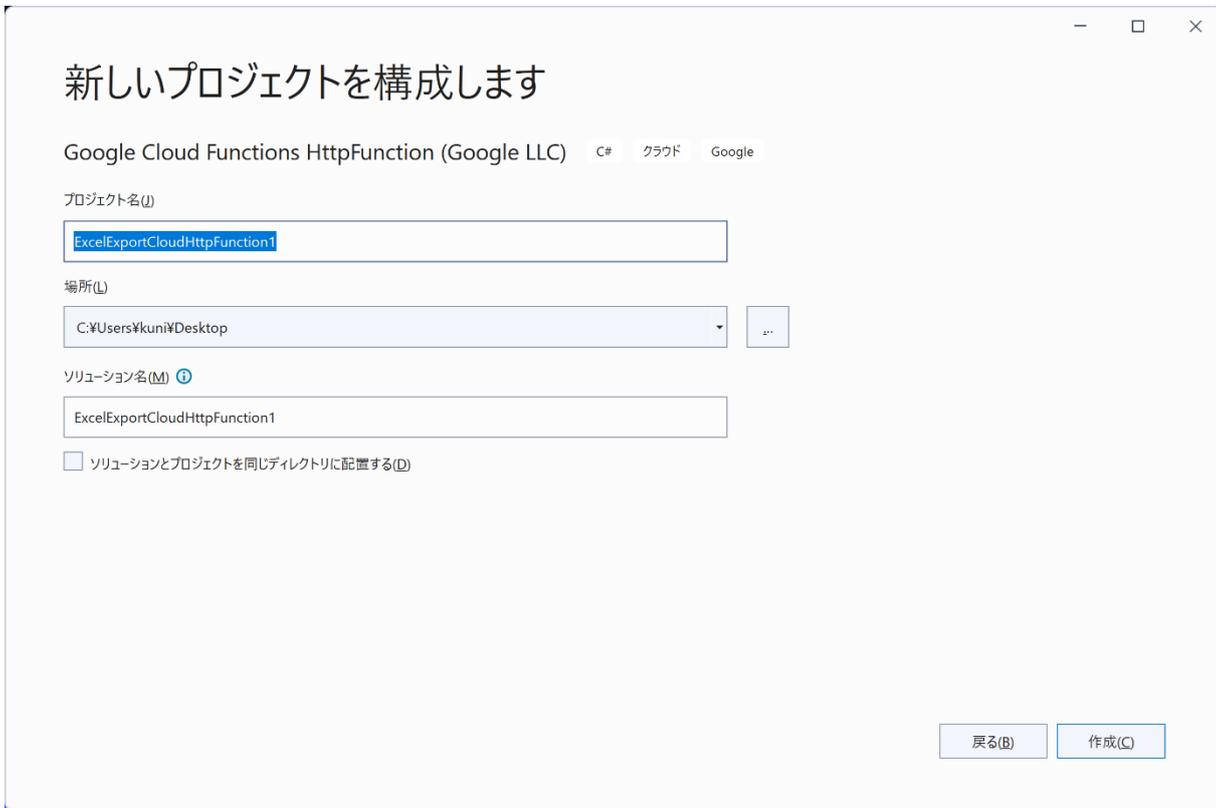
これで Visual Studio から Cloud Functions のプロジェクトを作成できるようになります。

Cloud Functions アプリケーションを作成

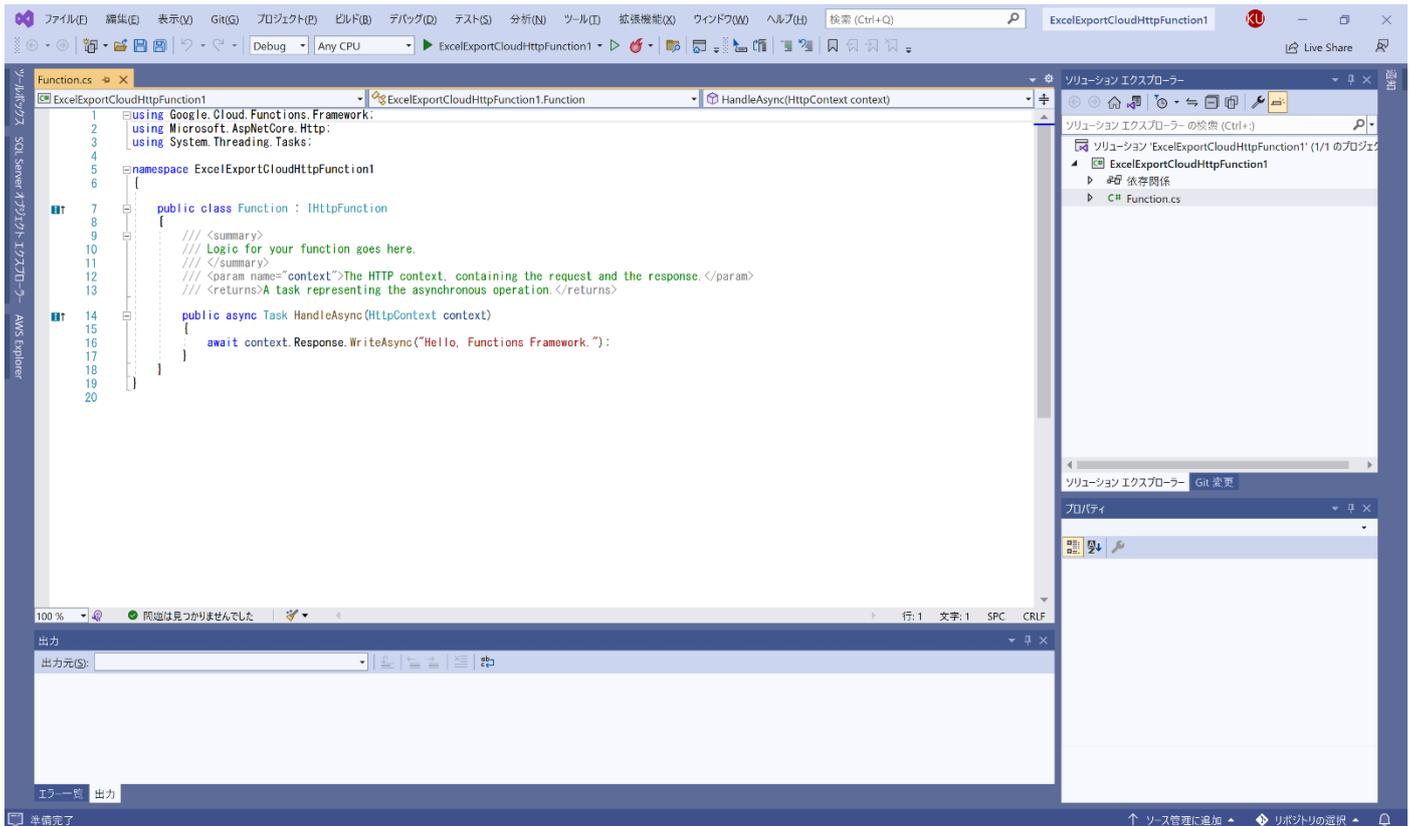
Visual Studio 2022 でプロジェクトテンプレート「Google Cloud Functions HttpFunction (Google LLC)」を選択して [次へ] をクリックします。



プロジェクト名に「ExcelExportCloudHttpFunction1」を入力して [作成] をクリックします。

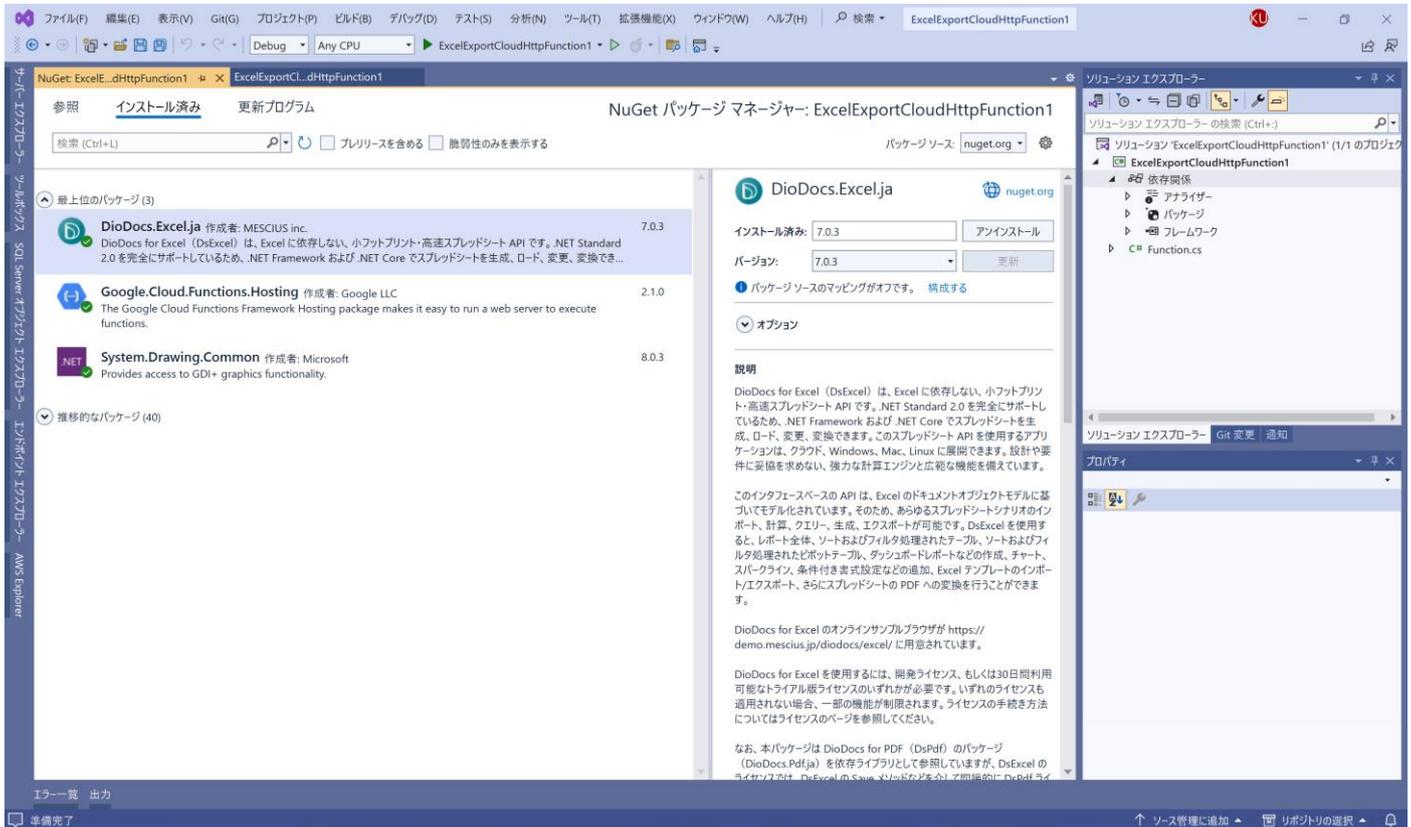


「ExcelExportCloudHttpFunction1」プロジェクトが作成されます。



NuGet パッケージを追加

Visual Studio の「NuGet パッケージ マネージャー」から DioDocs for Excel のパッケージ「[DioDocs.Excel.ja](#)」をインストールします。



DioDocs for Excel を使うコードを追加

DioDocs for Excel で Excel ファイルを作成するコードを追加して `HandleAsync` を以下のように更新します。

```
using GrapeCity.Documents.Excel;
using System.IO;

:

public async Task HandleAsync(HttpContext context)
{
    HttpRequest request = context.Request;
    string name = request.Query["name"].ToString();

    string Message = string.IsNullOrEmpty(name)
        ? "Hello, World!!"
        : $"Hello, {name}!!";

    //Workbook.SetLicenseKey("");

    Workbook workbook = new Workbook();
    workbook.Worksheets[0].Range["A1"].Value = Message;

    byte[] output;
```

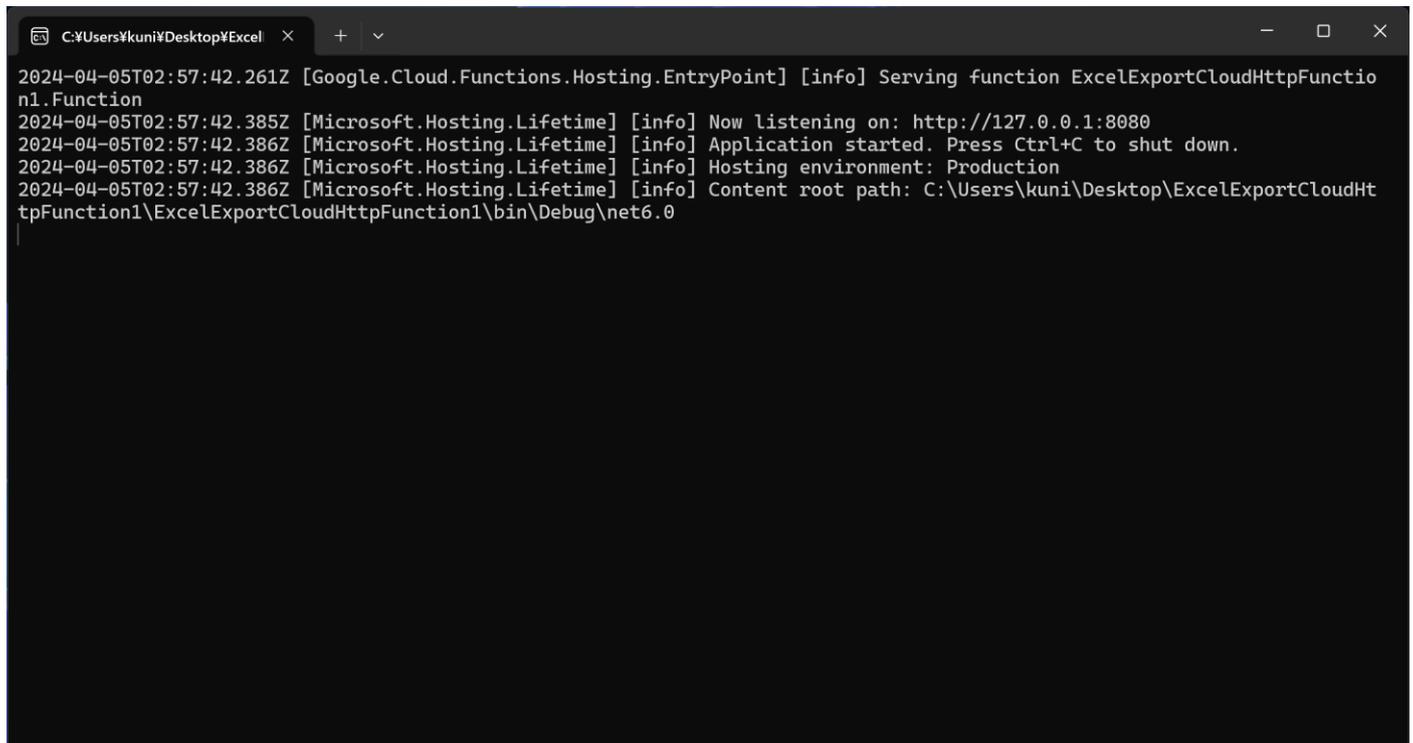
```
using (var ms = new MemoryStream())
{
    workbook.Save(ms, SaveFileFormat.Xlsx);
    output = ms.ToArray();
}

context.Response.ContentType = "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet";
context.Response.Headers.Add("Content-Disposition", "attachment;filename=Result.xlsx");
await context.Response.Body.WriteAsync(output);
}
```

DioDocs for Excel で作成した Excel ファイルを `MemoryStream` に保存し、これを `HttpResponse` の `Body` に設定してダウンロードできるように設定しています。

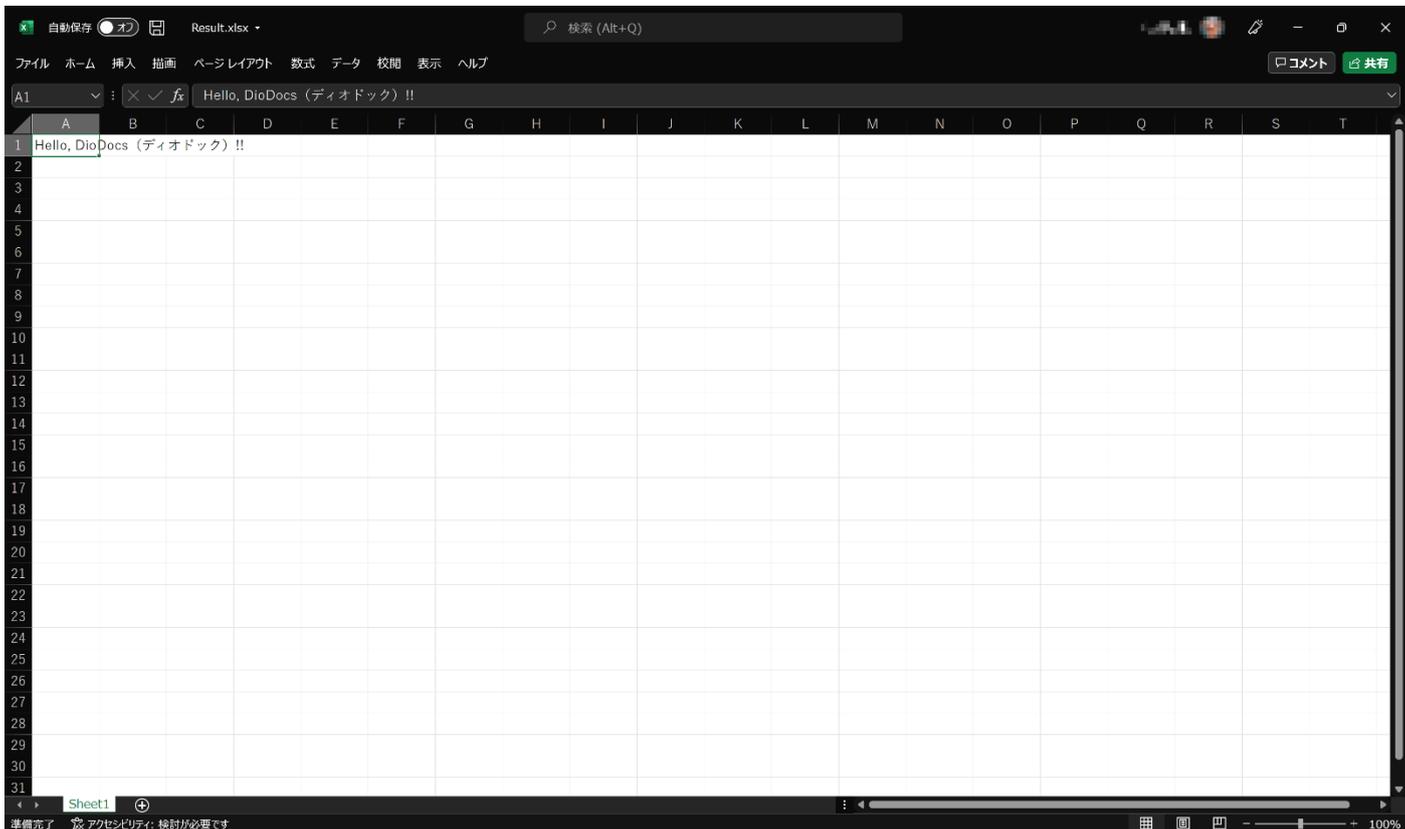
デバッグ実行で確認

作成した Cloud Functions の関数アプリケーションを Visual Studio からデバッグ実行して確認します。ブラウザのアドレスバーにコンソールに表示される URL 「<http://127.0.0.1:8080>」に、クエリパラメータと文字列「`?name=DioDocs (ディオドック)`」を追加して、関数アプリケーションを実行します。



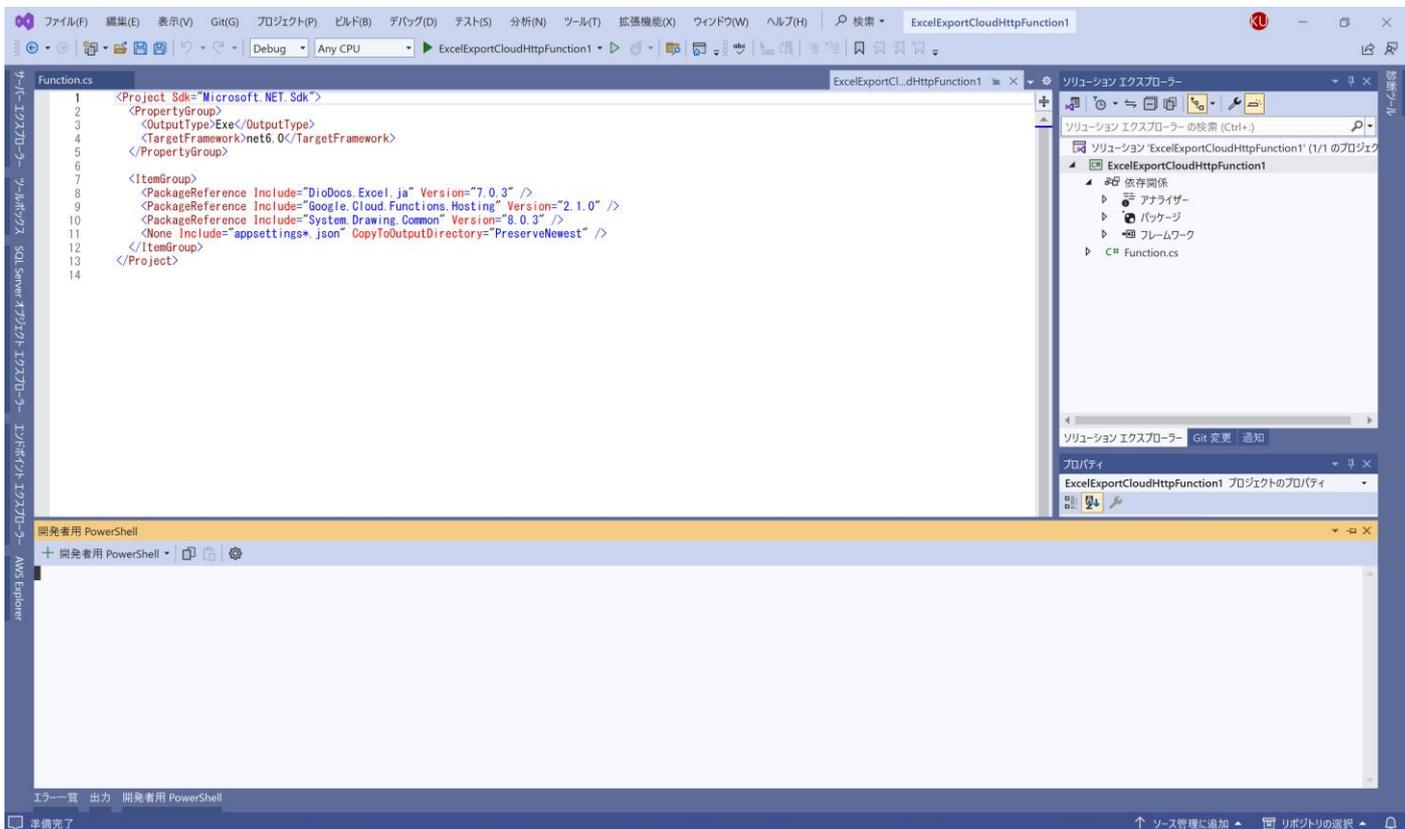
```
C:\Users\kuni\Desktop\Excel >
2024-04-05T02:57:42.261Z [Google.Cloud.Functions.Hosting.EntryPoint] [info] Serving function ExcelExportCloudHttpFunction1.Function
2024-04-05T02:57:42.385Z [Microsoft.Hosting.Lifetime] [info] Now listening on: http://127.0.0.1:8080
2024-04-05T02:57:42.386Z [Microsoft.Hosting.Lifetime] [info] Application started. Press Ctrl+C to shut down.
2024-04-05T02:57:42.386Z [Microsoft.Hosting.Lifetime] [info] Hosting environment: Production
2024-04-05T02:57:42.386Z [Microsoft.Hosting.Lifetime] [info] Content root path: C:\Users\kuni\Desktop\ExcelExportCloudHttpFunction1\ExcelExportCloudHttpFunction1\bin\Debug\net6.0
```

実行するとクエリパラメータで渡した文字列「`DioDocs (ディオドック)`」が追加された Excel ファイル「`Result.xlsx`」がローカルに出力されます。



Google Cloud へデプロイ

Visual Studio の「ソリューション エクスプローラー」からプロジェクトを右クリックして、コンテキストメニューから「ターミナルで開く」をクリックして「開発者用 PowerShell」を起動します。



`gcloud functions deploy` コマンドを実行して Google Cloud へ Cloud Functions の関数アプリケーションをデプロイします。

```
gcloud functions deploy diodocs-excel-export-function1 --gen2 --entry-point
ExcelExportCloudHttpFunction1.Function --runtime dotnet6 --trigger-http --allow-unauthenticated
```

Google Cloud の Cloud Functions にも以下のようにデプロイが完了した関数アプリケーションが表示されます。

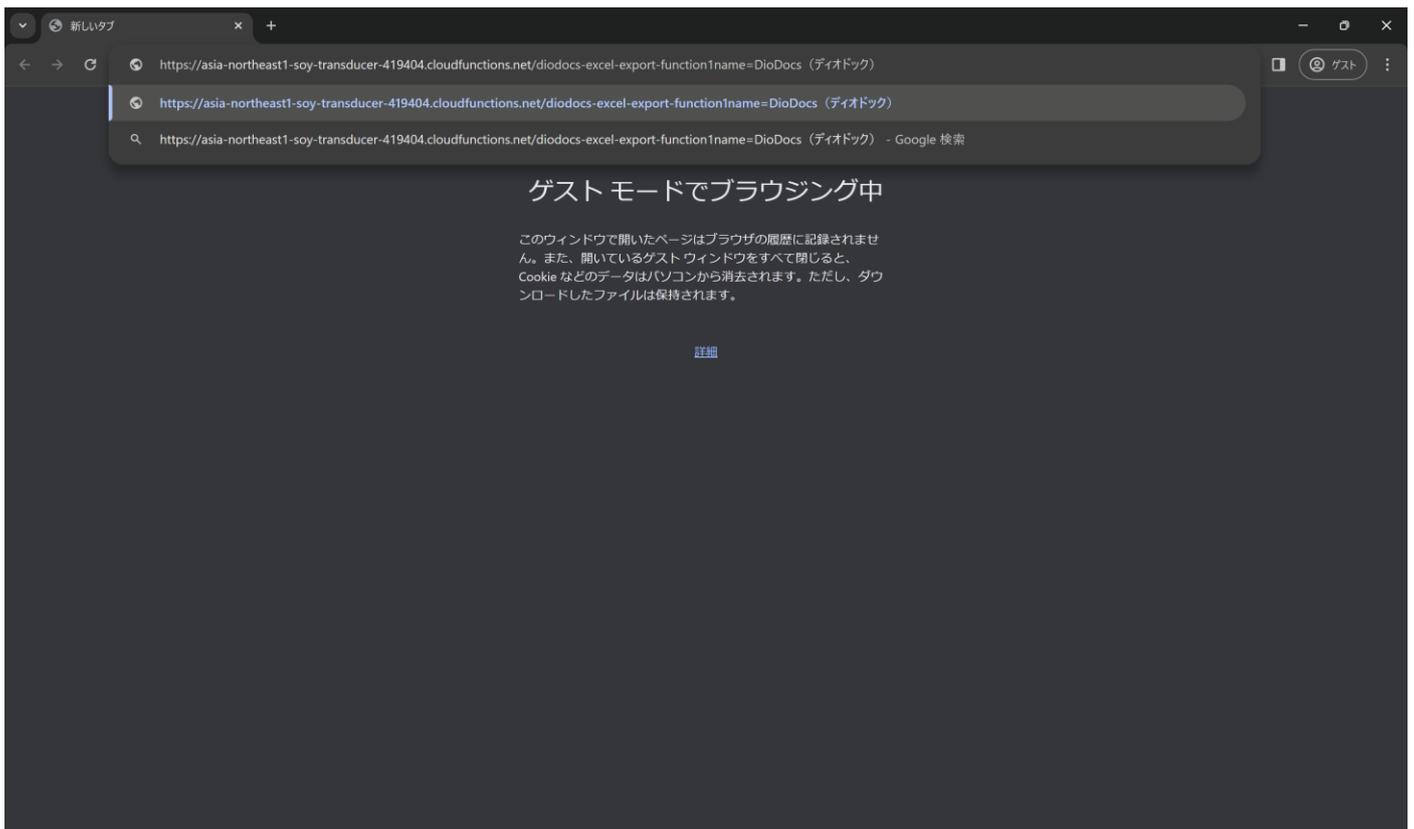
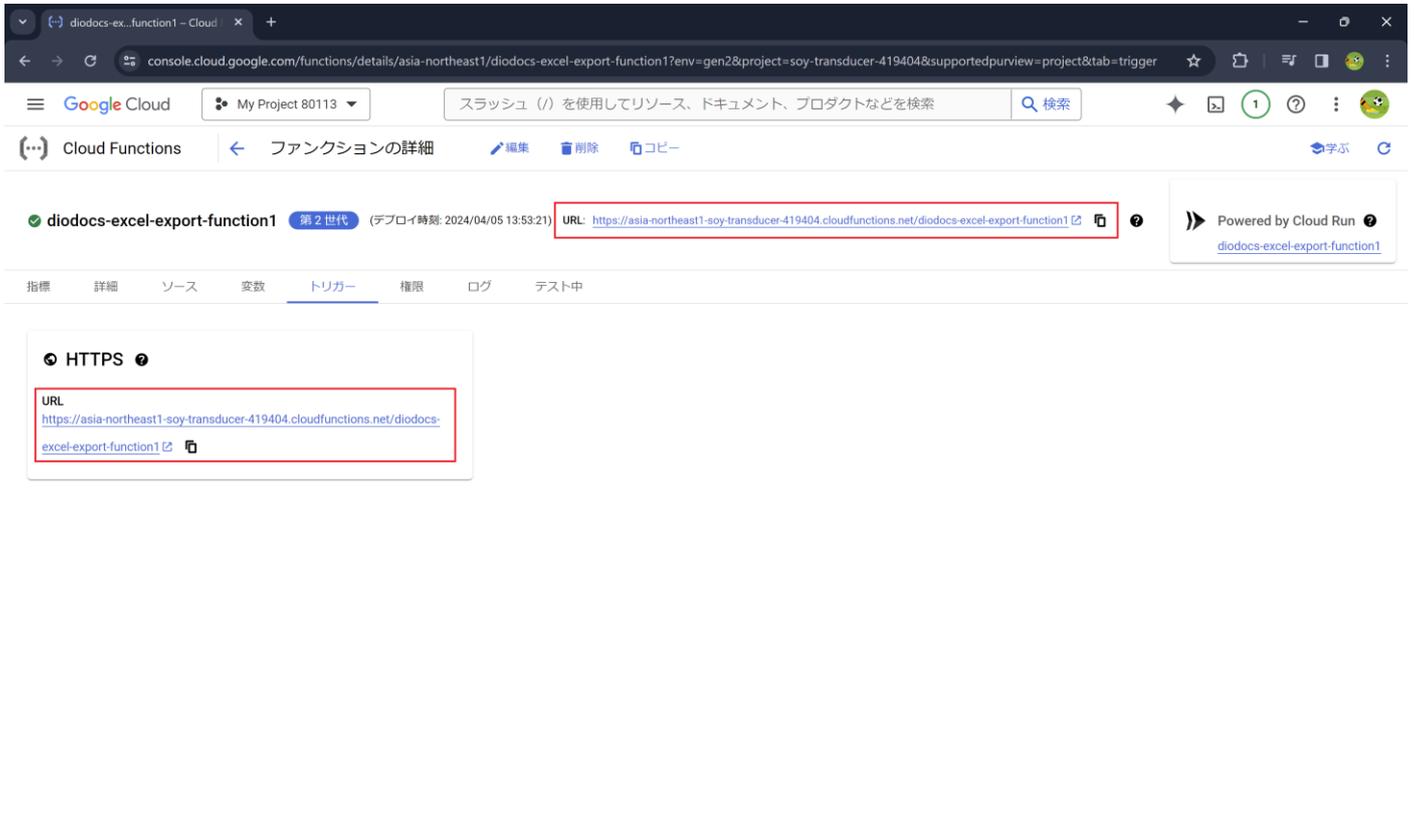


The screenshot shows the Google Cloud Functions console interface. The browser address bar displays the URL: `console.cloud.google.com/functions/list?project=soy-transducer-419404&supportedpurview=project`. The page title is "Functions - Cloud Functions". The main content area shows a table of functions with the following data:

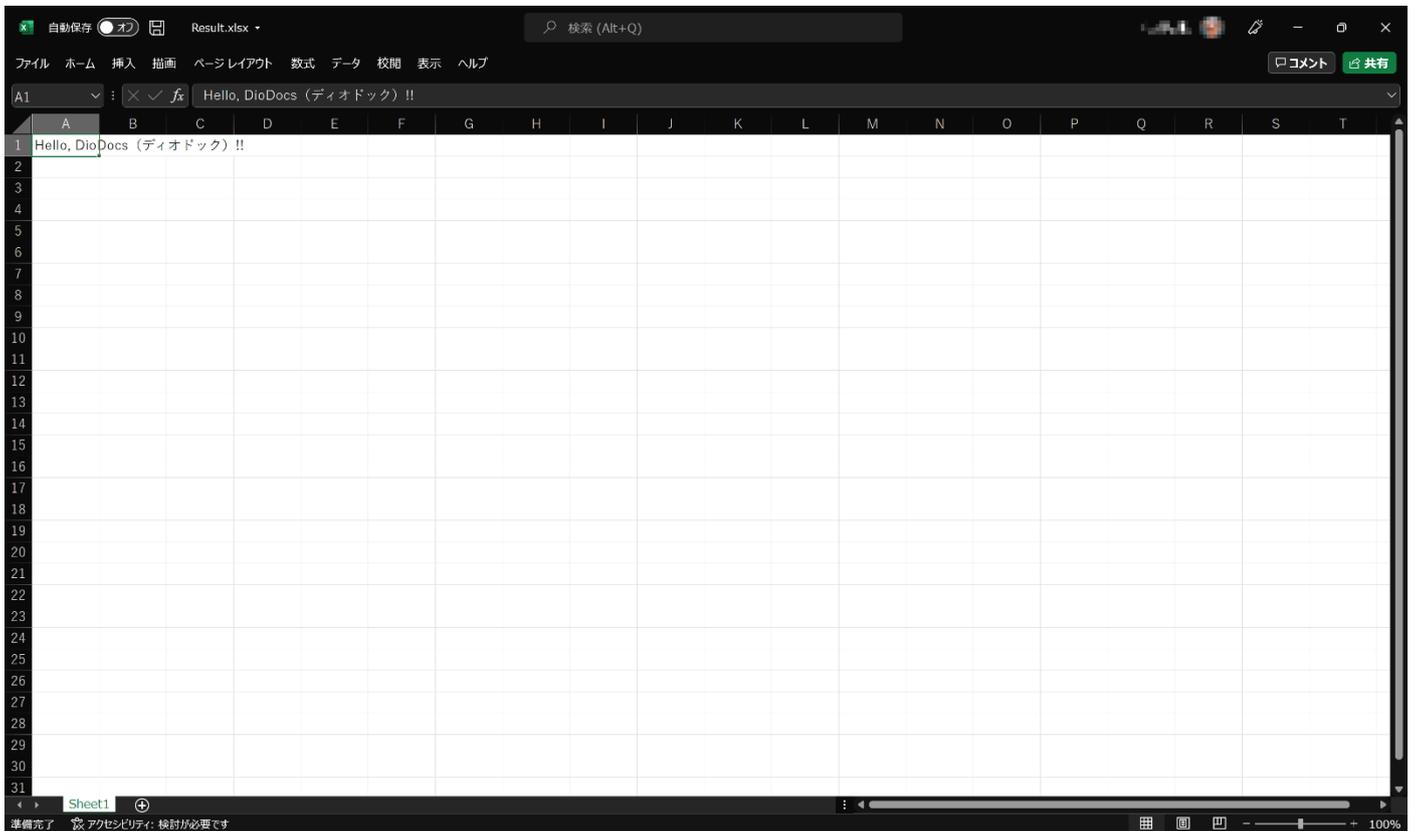
環境	名前	最終デプロイ日	リージョン	推奨事項	トリガー	ランタイム	割り当てられたメモリ	実行済みの関数	操作
2nd gen	diodocs-excel-export-function1	2024/04/05 13:53:21	asia-northeast1		HTTP	.NET 6.0	256 MB	ExcelExportCloudHttpFunction1.Function	

デプロイしたアプリケーションを確認

ブラウザのアドレスバーに、関数アプリケーションの URL にクエリパラメータと文字列「`?name=DioDocs (ディオドック)`」を追加して、関数アプリケーションを実行します。



この関数アプリケーションを実行するとクエリパラメータで渡した文字列「**DioDocs (ディオドック)**」が追加された Excel ファイル「Result.xlsx」がローカルに出力されます。



PDF を出力するには？

Visual Studio の「NuGet パッケージ マネージャー」から DioDocs for PDF のパッケージ「[DioDocs.Pdf.ja](#)」をインストールします。 DioDocs for PDF で PDF ファイルを作成するコードを追加して `HandleAsync` を以下のように更新します。

```
using GrapeCity.Documents.Pdf;
using GrapeCity.Documents.Text;
using System.Drawing;
using System.IO;

:

public async Task HandleAsync(HttpContext context)
{
    HttpRequest request = context.Request;
    string name = request.Query["name"].ToString();

    string Message = string.IsNullOrEmpty(name)
        ? "Hello, World!!"
        : $"Hello, {name}!!";

    //GcPdfDocument.SetLicenseKey("");
```

```
GcPdfDocument doc = new GcPdfDocument();
GcPdfGraphics g = doc.NewPage().Graphics;

g.DrawString(Message,
    new TextFormat() { Font = StandardFonts.Helvetica, FontSize = 12 },
    new PointF(72, 72));

byte[] output;

using (var ms = new MemoryStream())
{
    doc.Save(ms, false);
    output = ms.ToArray();
}

context.Response.ContentType = "application/pdf";
context.Response.Headers.Add("Content-Disposition", "attachment;filename=Result.pdf");
await context.Response.Body.WriteAsync(output);
}
```

さいごに

動作を確認できる Cloud Functions アプリケーションのサンプルはこちらです。

<https://github.com/MESCIUSJP/ExcelExportCloudHttpFunction1>

<https://github.com/MESCIUSJP/PDFExportCloudHttpFunction1>

Cloud Functions と DioDocs で Excel や PDF ファイルを出力する (2)

[前回](#)に引き続き、本記事でも Cloud Functions で「[DioDocs \(ディオドック\)](#)」を使用した C# (.NET 6、8) の関数アプリケーションを作成し、Excel や PDF ファイルを出力する方法について紹介します。

実装する内容

今回も関数アプリケーションで [HTTP 関数](#) を作成します。この関数の実行時に DioDocs を使用して Excel と PDF ファイルを作成し、HTTP リクエストのクエリパラメータで受け取った文字列を追加します。

前回は作成した Excel と PDF ファイルを関数から HTTP レスポンスで直接ローカルへ出力していましたが、今回は作成した Excel と PDF ファイルを [Cloud Storage](#) へ出力します。

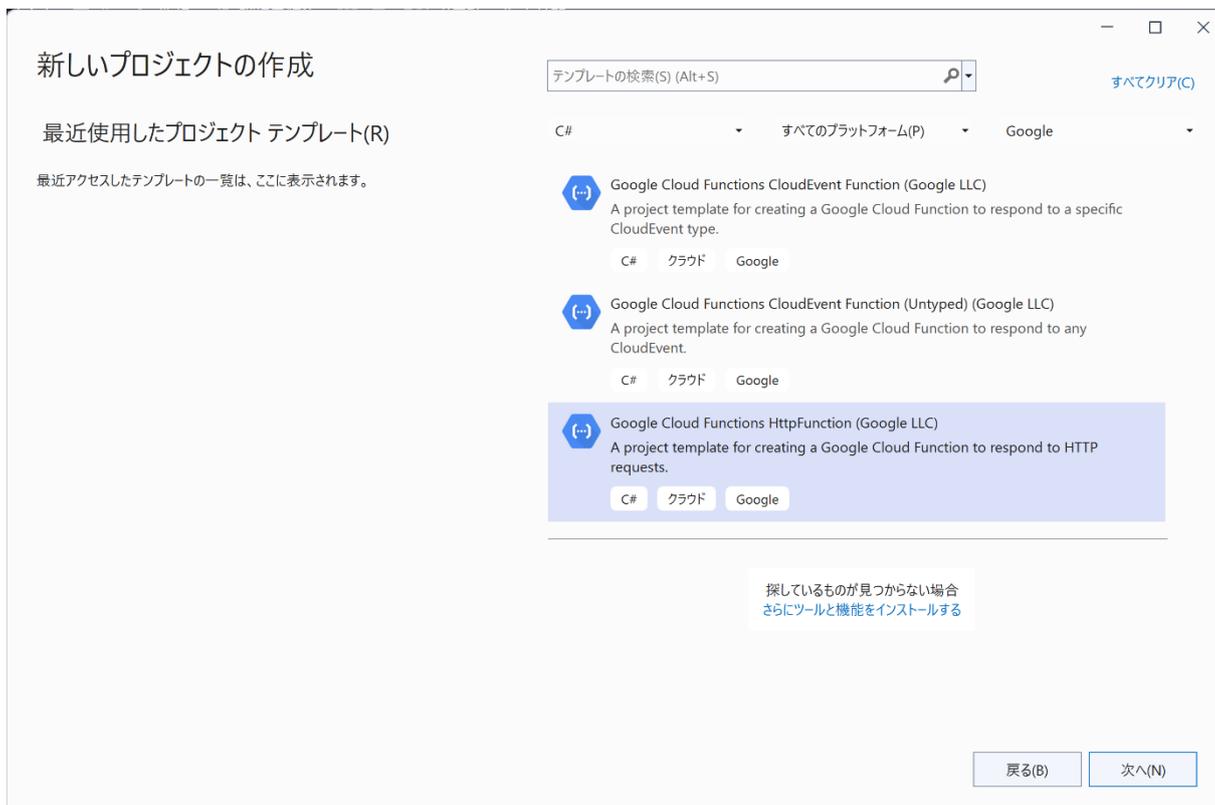
Cloud Storage への保存には Google Cloud API を各言語で使用するための [クライアントライブラリ](#) で、.NET アプリケーション開発向けに提供されている「[Google Cloud Client Libraries for .NET](#)」を使用します。

事前準備

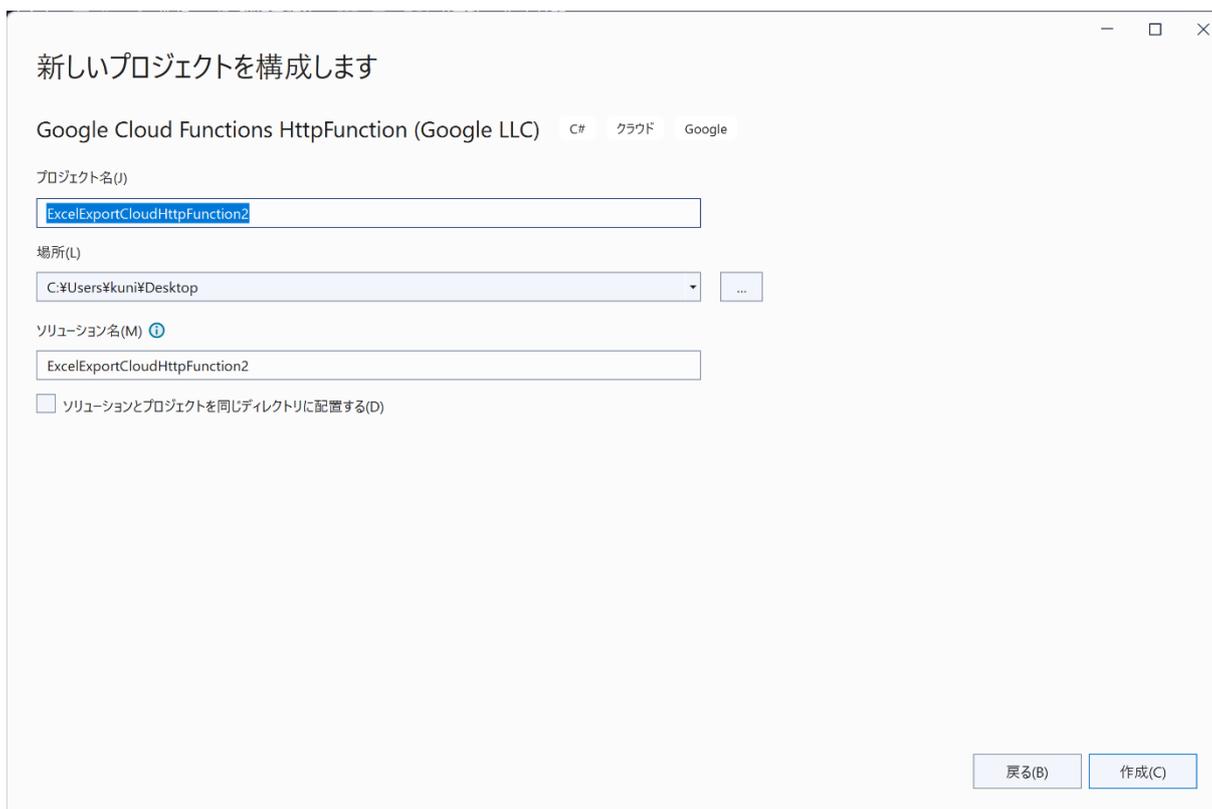
事前準備については[前回](#)の内容を参照してください。

Cloud Functions アプリケーションを作成

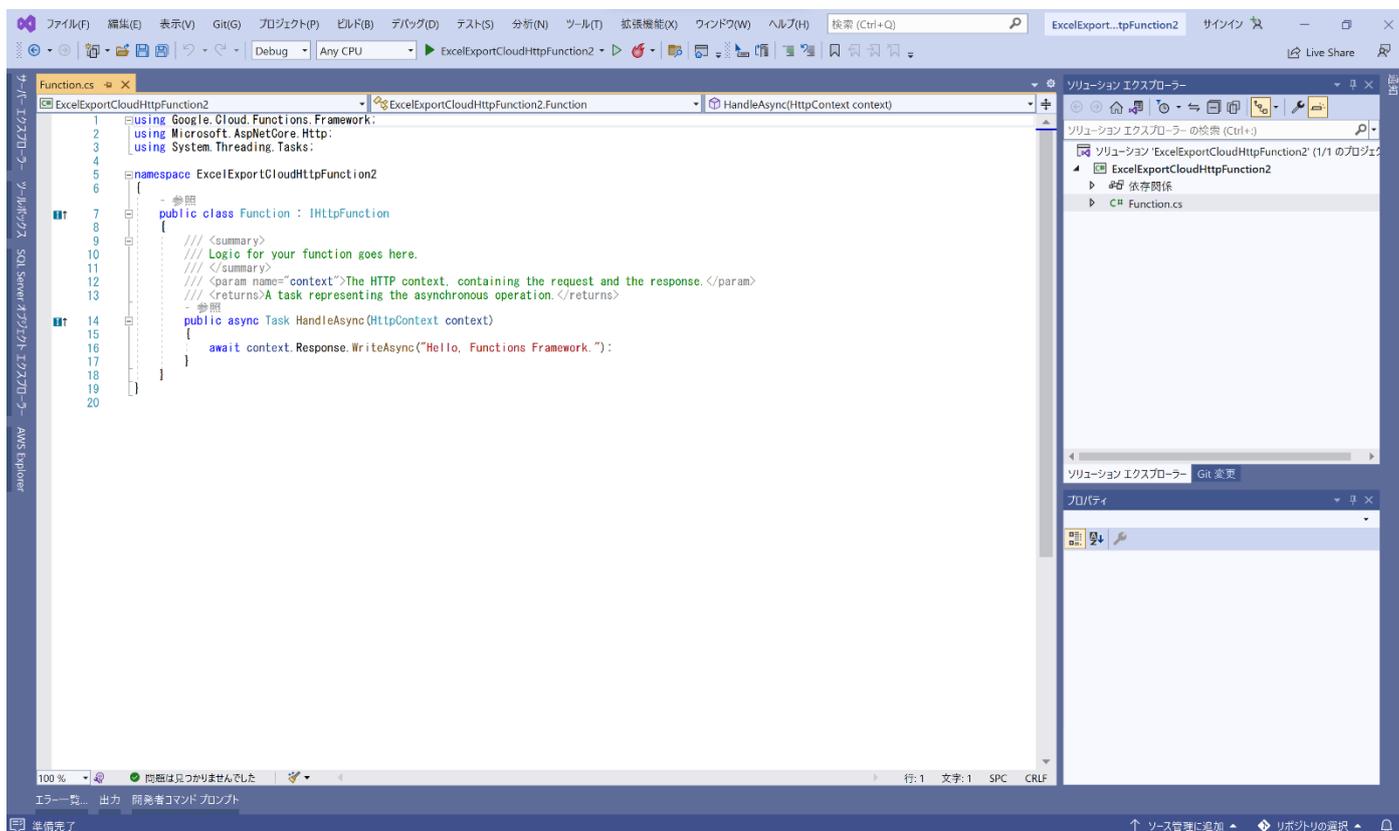
Visual Studio 2022 でプロジェクトテンプレート「Google Cloud Functions HttpFunction (Google LLC)」を選択して [次へ] をクリックします。



プロジェクト名に「ExcelExportCloudHttpFunction2」を入力して [作成] をクリックします。



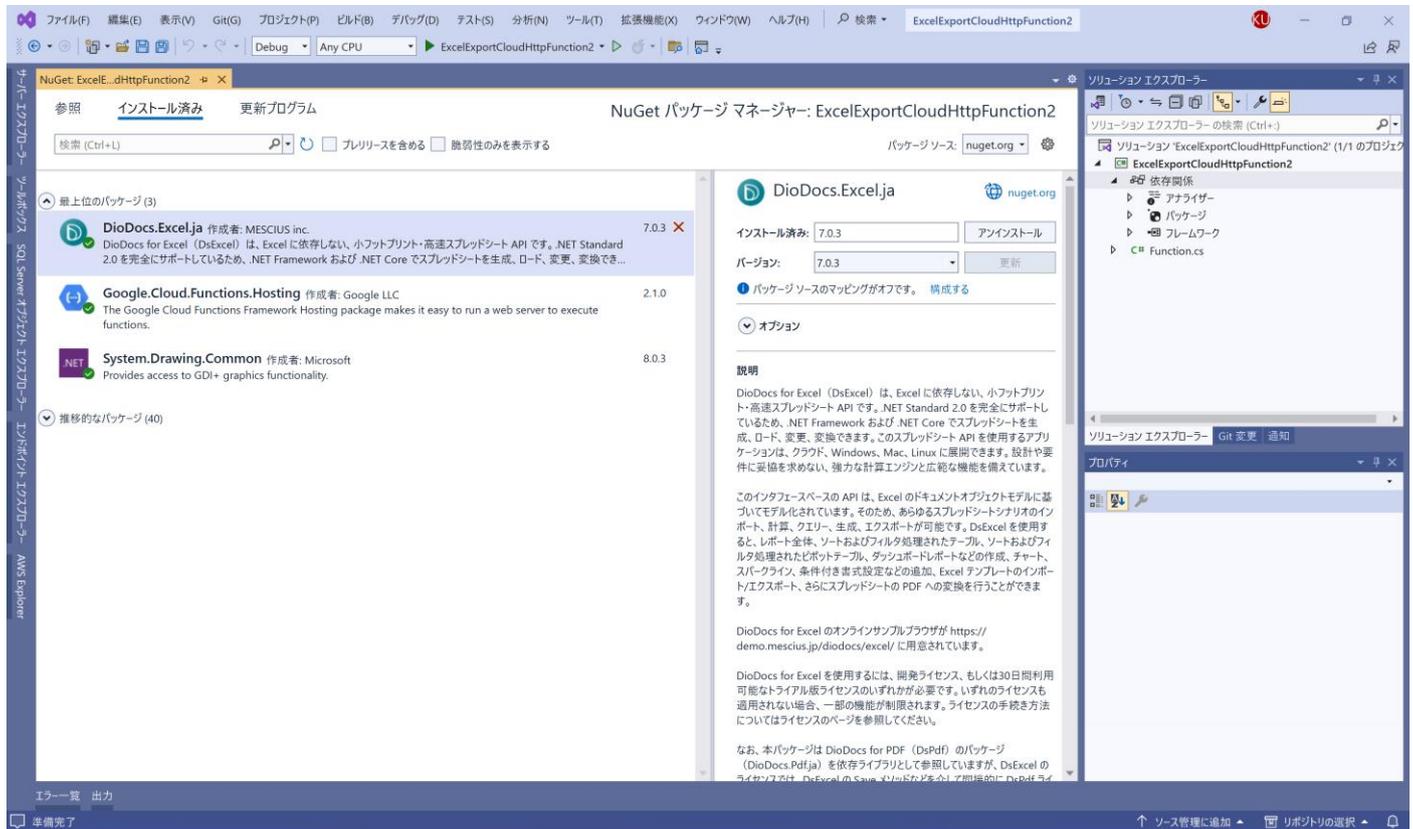
「ExcelExportCloudHttpFunction2」プロジェクトが作成されます。



NuGet パッケージの追加

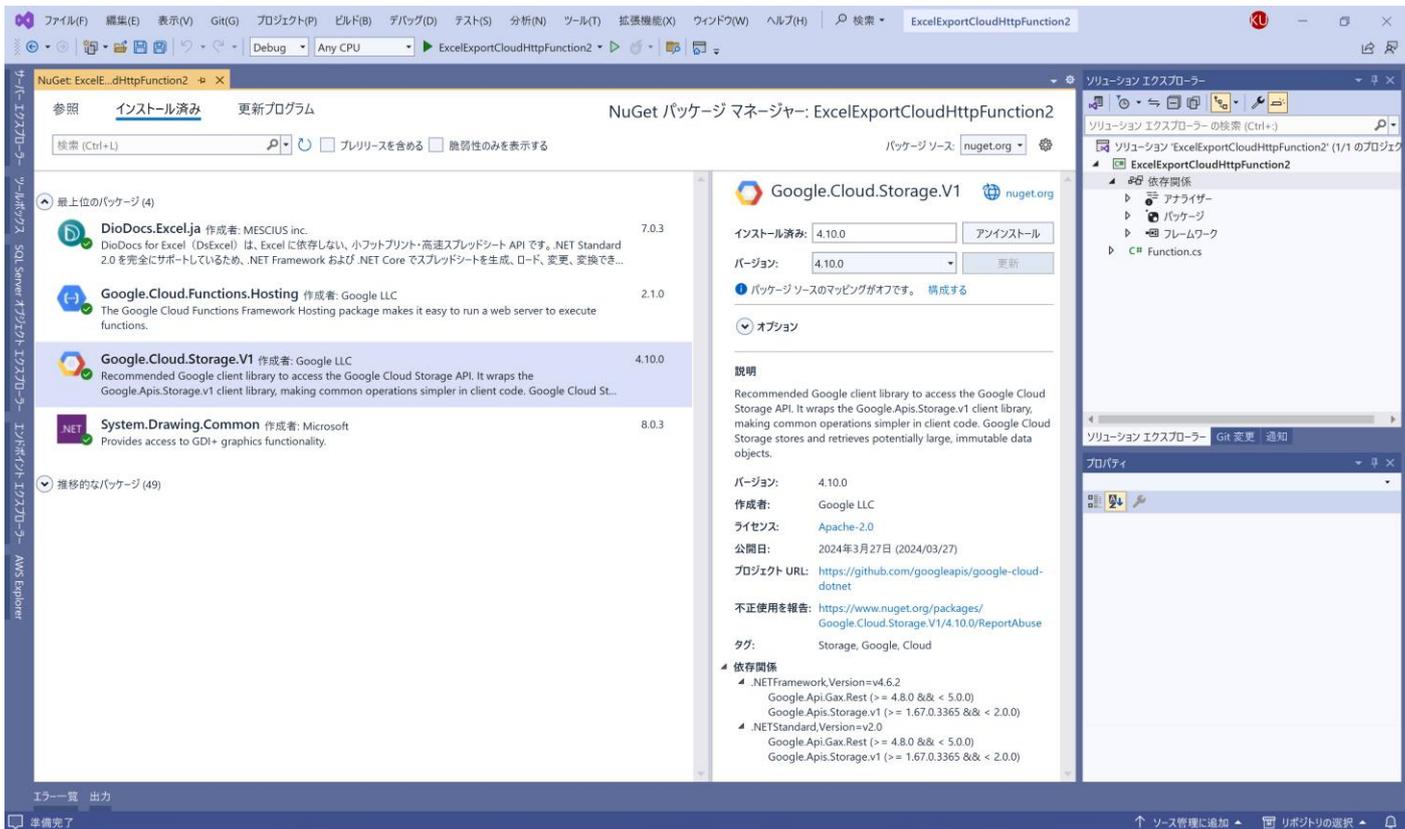
DioDocs for Excel

Visual Studio の「NuGet パッケージ マネージャー」から DioDocs for Excel のパッケージ「[DioDocs.Excel.ja](#)」をインストールします。



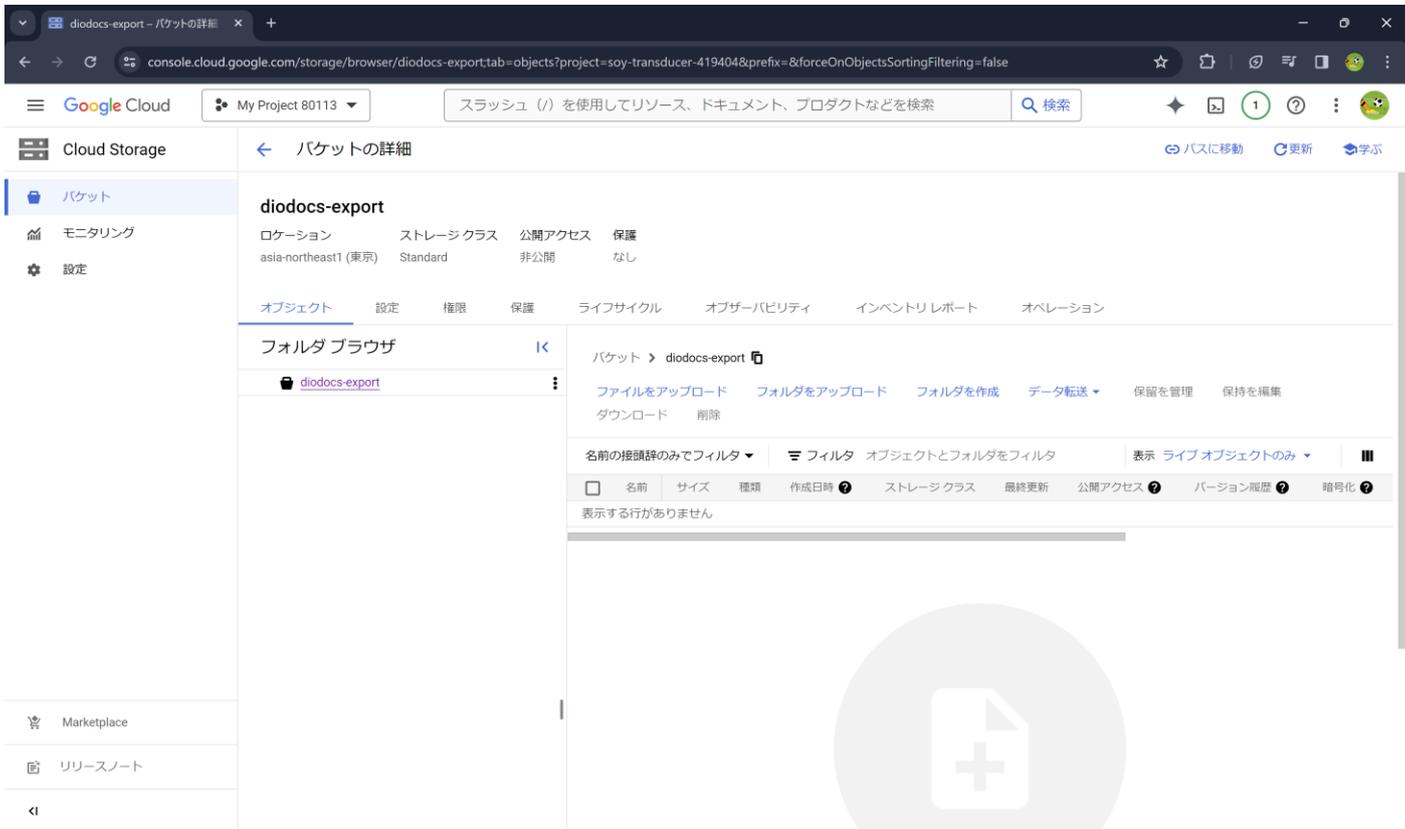
Google.Cloud.Storage.V1 (Google Cloud Client Libraries for .NET)

同じ手順で Cloud Storage 用のクライアントアプリのパッケージ「[Google.Cloud.Storage.V1](#)」をインストールします。



Cloud Storage にバケットを作成

Cloud Storage に DioDocs for Excel で作成した Excel ファイルの保存先になるバケット「**diodocs-export**」を、Google Cloud のコンソールから作成します。



DioDocs for Excel を使うコードを追加

DioDocs for Excel で Excel ファイルを作成するコードを追加して `HandleAsync` を以下のように更新します。

```
using Google.Cloud.Functions.Framework;
using Google.Cloud.Storage.V1;
using GrapeCity.Documents.Excel;
using Microsoft.AspNetCore.Http;
using System.IO;
using System.Threading.Tasks;

:

public async Task HandleAsync(HttpContext context)
{
    HttpRequest request = context.Request;
    string name = request.Query["name"].ToString();

    string Message = string.IsNullOrEmpty(name)
        ? "Hello, World!!"
        : $"Hello, {name}!!";

    Workbook workbook = new Workbook();
    workbook.Worksheets[0].Range["A1"].Value = Message;

    using MemoryStream outputstream = new MemoryStream();
    StorageClient sc = StorageClient.Create();
    workbook.Save(outputstream, SaveFileFormat.Xlsx);
    await sc.UploadObjectAsync("diodocs-export", "Result.xlsx", "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet", outputstream);
}
```

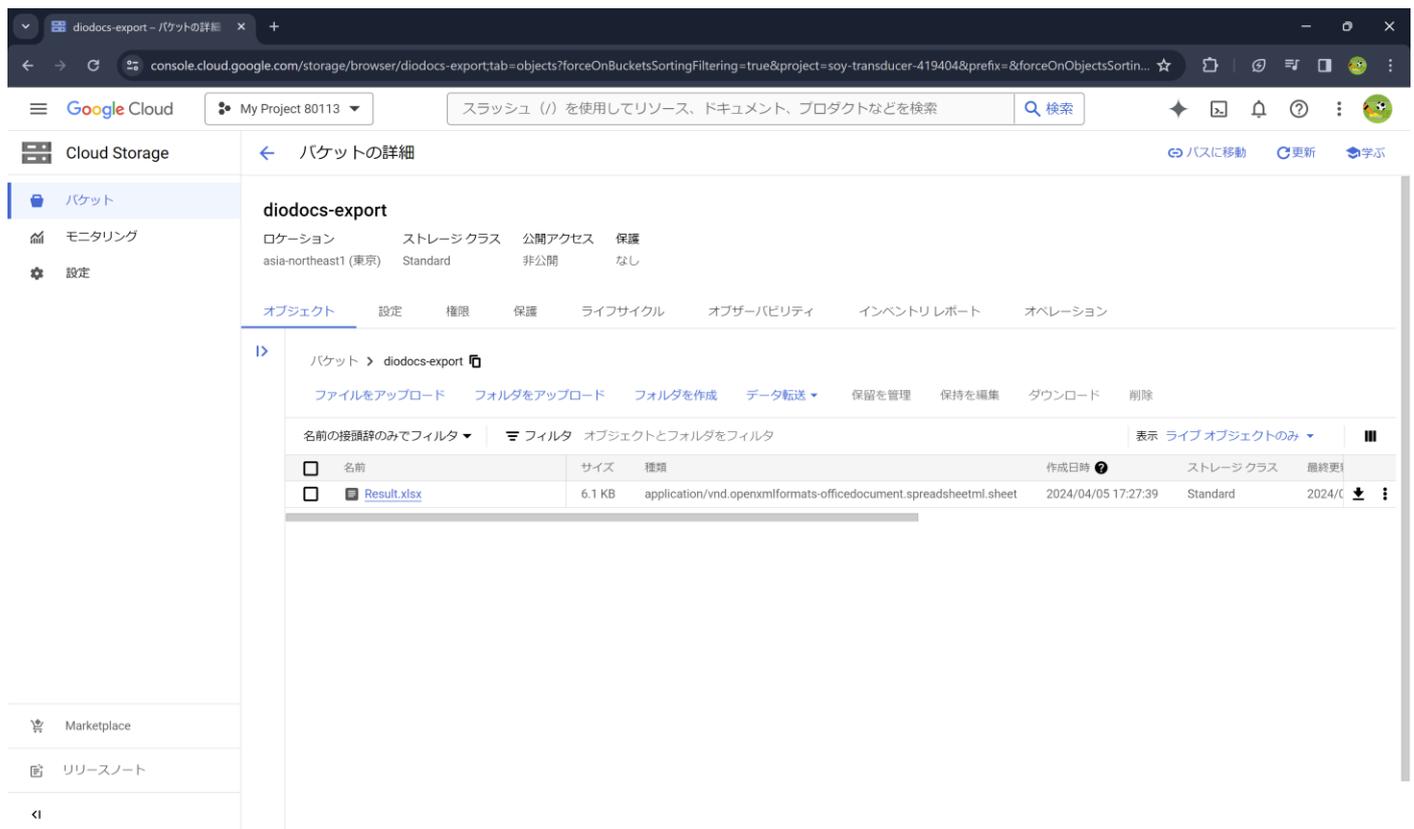
DioDocs for Excel で作成した Excel ファイル「Result.xlsx」を `MemoryStream` に保存し、これを `StorageClient` クラスの `UploadObjectAsync` メソッドを使用して Cloud Storage へアップロードしています。

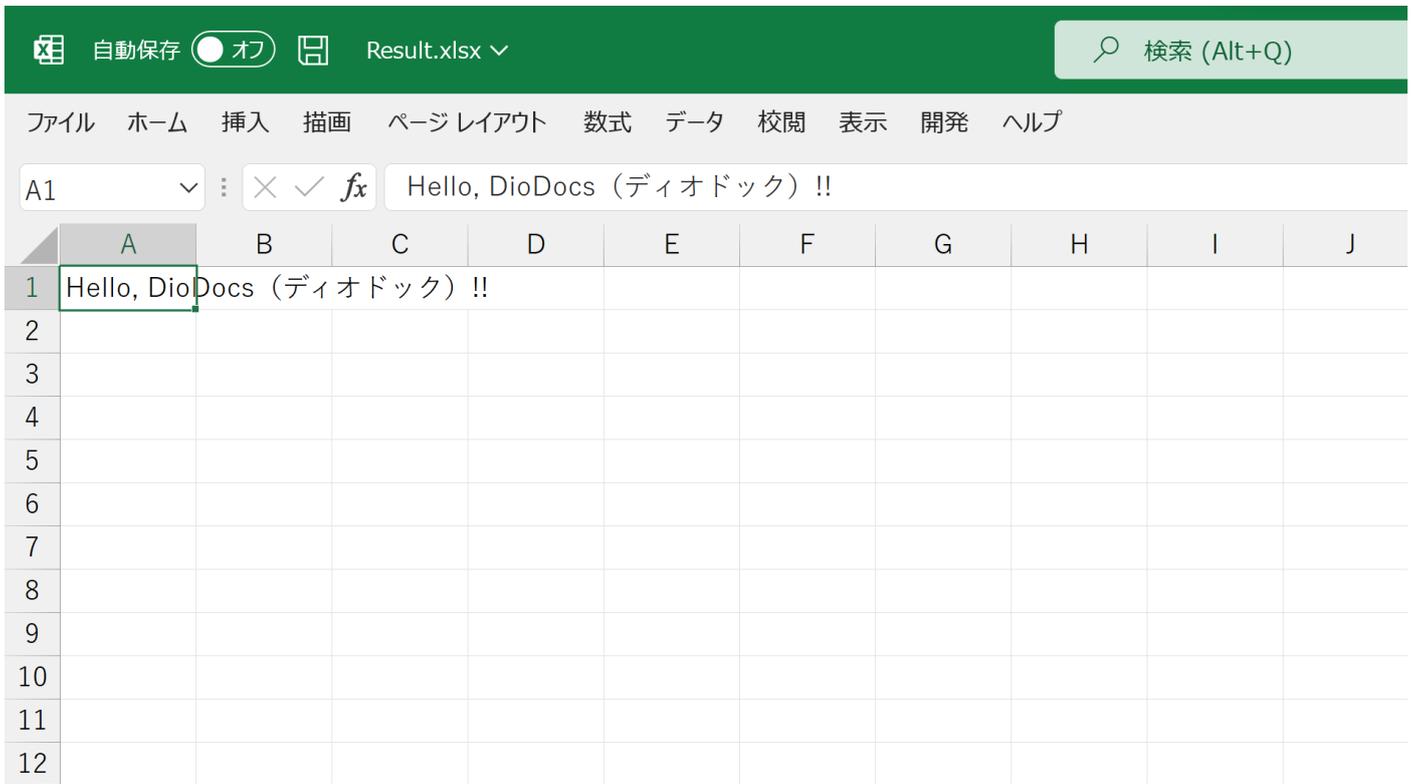
デバッグ実行で確認

作成した Cloud Functions の関数アプリケーションを Visual Studio からデバッグ実行して確認します。ブラウザのアドレスバーにコンソールに表示される URL 「<http://127.0.0.1:8080>」に、クエリパラメータと文字列「`?name=DioDocs (ディオドック)`」を追加して、関数アプリケーションを実行します。

```
C:\Users\kuni\Desktop\Excel
2024-04-05T08:26:29.281Z [Google.Cloud.Functions.Hosting.EntryPoint] [info] Serving function ExcelExportCloudHttpFunction2.Function
2024-04-05T08:26:29.544Z [Microsoft.Hosting.Lifetime] [info] Now listening on: http://127.0.0.1:8080
2024-04-05T08:26:29.548Z [Microsoft.Hosting.Lifetime] [info] Application started. Press Ctrl+C to shut down.
2024-04-05T08:26:29.549Z [Microsoft.Hosting.Lifetime] [info] Hosting environment: Production
2024-04-05T08:26:29.550Z [Microsoft.Hosting.Lifetime] [info] Content root path: C:\Users\kuni\Desktop\ExcelExportCloudHttpFunction2\ExcelExportCloudHttpFunction2\bin\Debug\net6.0
```

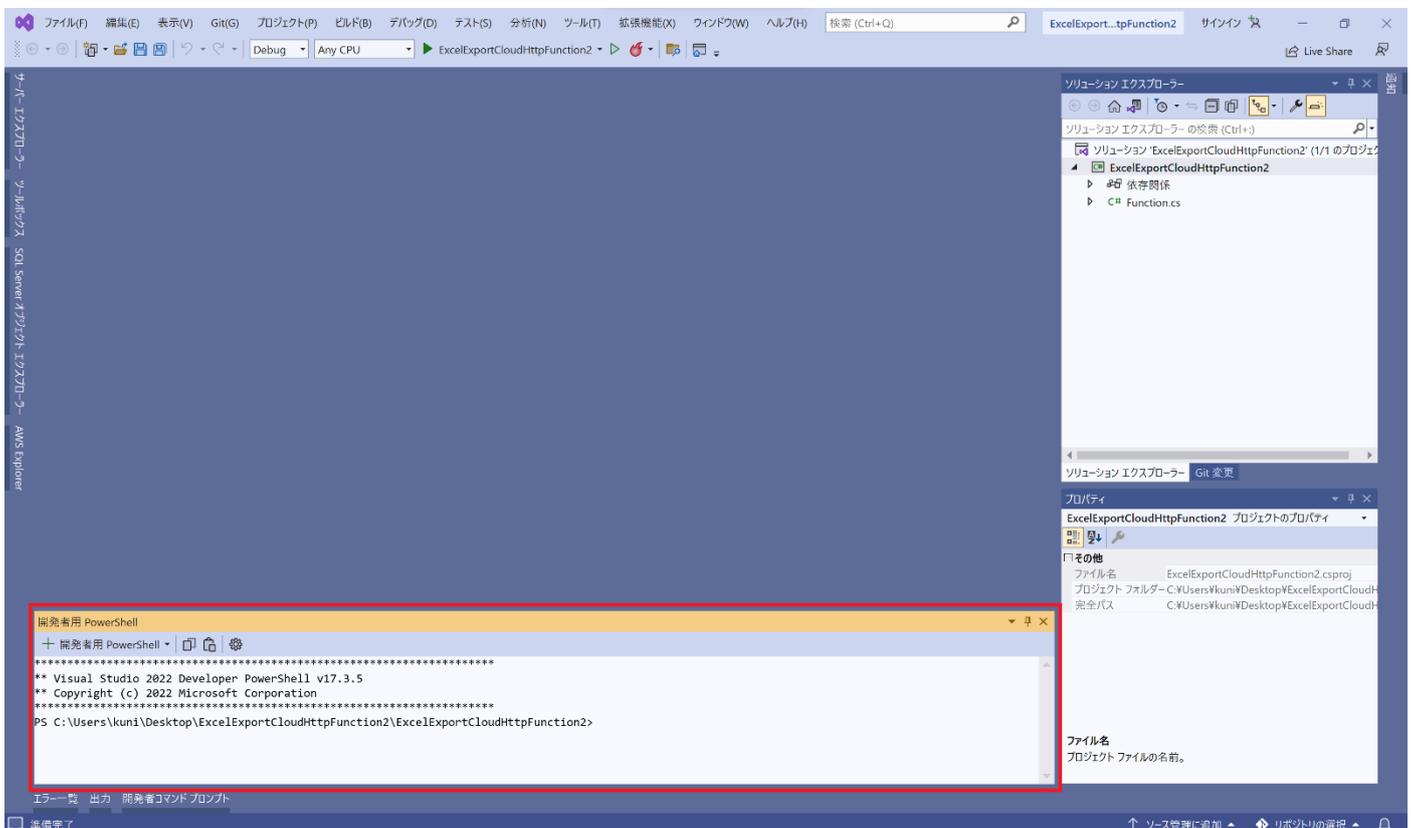
実行するとクエリパラメータで渡した文字列「DioDocs (ディオドック)」が追加された Excel ファイル「Result.xlsx」が Cloud Storage に出力されます。





Google Cloud へデプロイ

Visual Studio の「ソリューション エクスプローラー」からプロジェクトを右クリックして、コンテキストメニューから [ターミナルで開く] をクリックして「開発者用 PowerShell」を起動します。



`gcloud functions deploy` コマンドを実行して Google Cloud へ Cloud Functions の関数アプリケーションを

デプロイします。

```
gcloud functions deploy diodocs-excel-export-function2 --gen2 --entry-point  
ExcelExportCloudHttpFunction2.Function --runtime dotnet6 --trigger-http --allow-unauthenticated
```

Google Cloud の Cloud Functions にも以下のようにデプロイが完了した関数アプリケーションが表示されます。



環境	名前	最終デプロイ日	リージョン	推奨事項	トリガー	ランタイム	割り当てられたメモリ	実行済みの開放	操作
2nd gen	diodocs-excel-export-function2	2024/04/05 17:37:40	asia-northeast1		HTTP	.NET 6.0	256 MB	ExcelExportCloudHttpFunction2.Function	

デプロイしたアプリケーションを確認

ブラウザのアドレスバーに、関数アプリケーションの URL にクエリパラメータと文字列「?name=DioDocs (ディオドック)」を追加して、関数アプリケーションを実行します。



URL: <https://asia-northeast1-soy-transducer-419404.cloudfunctions.net/diodocs-excel-export-function2>

URL: <https://asia-northeast1-soy-transducer-419404.cloudfunctions.net/diodocs-excel-export-function2?name=DioDocs>

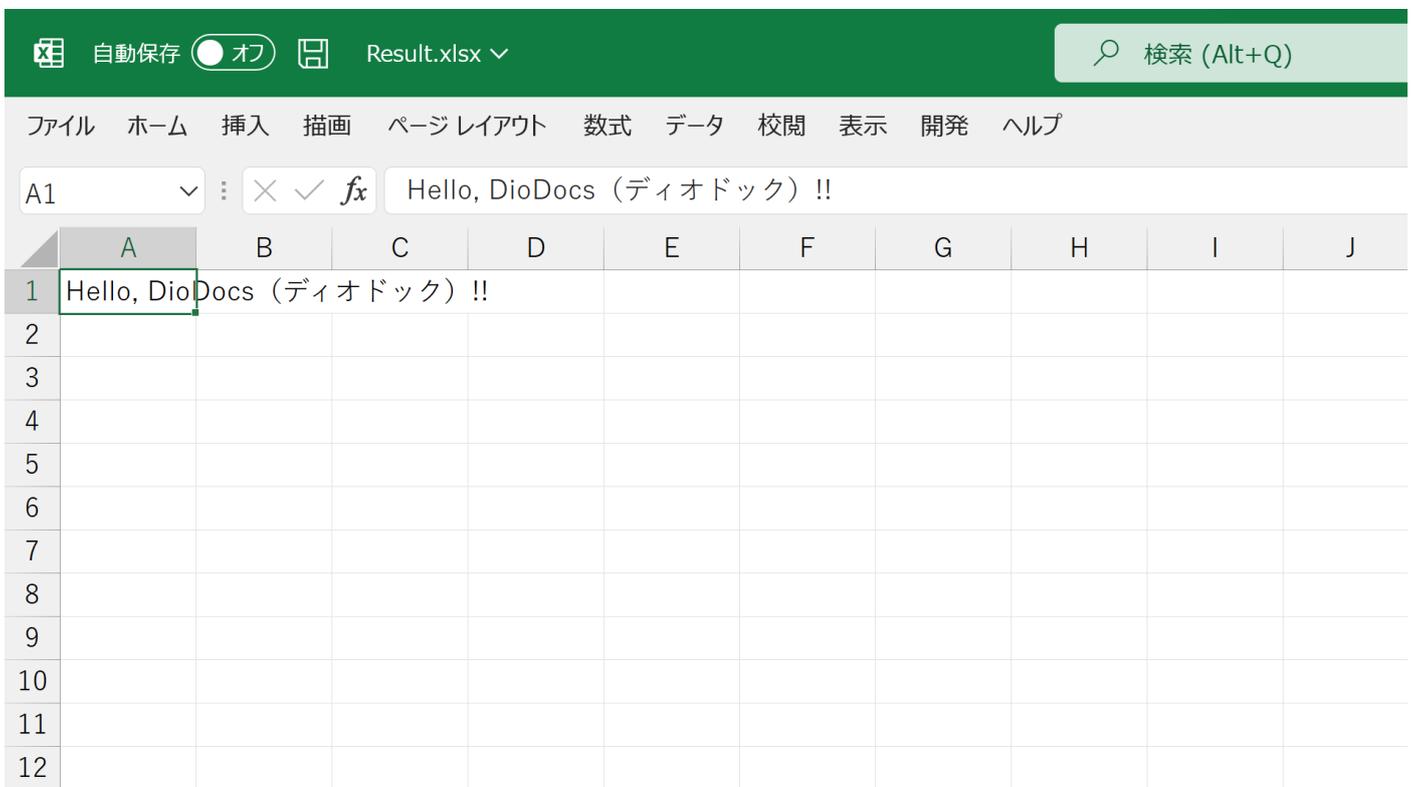


https://us-central1-diodocs-test.cloudfunctions.net/diodocs-excel-export-function2?name=DioDocs (ディオドック)

ゲストモードでブラウジング中

このウィンドウで開いたページはブラウザの履歴に記録されません。また、開いているゲストウィンドウをすべて閉じると、Cookie などのデータはパソコンから消去されます。ただし、ダウンロードしたファイルは保持されます。

この関数アプリケーションを実行するとクエリパラメータで渡した文字列「DioDocs (ディオドック)」が追加された Excel ファイル「Result.xlsx」が Cloud Storage に出力されます。



PDF を出力するには？

Visual Studio の「NuGet パッケージ マネージャー」から DioDocs for PDF のパッケージ「[DioDocs.Pdf.ja](#)」をインストールします。DioDocs for PDF で PDF ファイルを作成するコードを追加して HandleAsync を以下のように更新します。

```
using Google.Cloud.Functions.Framework;  
using Google.Cloud.Storage.V1;  
using GrapeCity.Documents.Pdf;  
using GrapeCity.Documents.Text;  
using Microsoft.AspNetCore.Http;
```

```

using System.Drawing;
using System.IO;
using System.Threading.Tasks;

:

public async Task HandleAsync(HttpContext context)
{
    HttpRequest request = context.Request;
    string name = request.Query["name"].ToString();

    string Message = string.IsNullOrEmpty(name)
        ? "Hello, World!!"
        : $"Hello, {name}!!";

    GcPdfDocument doc = new GcPdfDocument();
    GcPdfGraphics g = doc.NewPage().Graphics;

    g.DrawString(Message,
        new TextFormat() { Font = StandardFonts.Helvetica, FontSize = 12 },
        new PointF(72, 72));

    using MemoryStream outputstream = new MemoryStream();
    StorageClient sc = StorageClient.Create();
    doc.Save(outputstream, false);
    await sc.UploadObjectAsync("diodocs-export", "Result.pdf", "application/pdf", outputstream);
}

```

さいごに

動作を確認できる Cloud Functions アプリケーションのサンプルはこちらです。

<https://github.com/MESCIUSJP/ExcelExportCloudHttpFunction2>

<https://github.com/MESCIUSJP/PDFExportCloudHttpFunction2>

Cloud Functions と DioDocs で Excel や PDF ファイルを出力する (3)

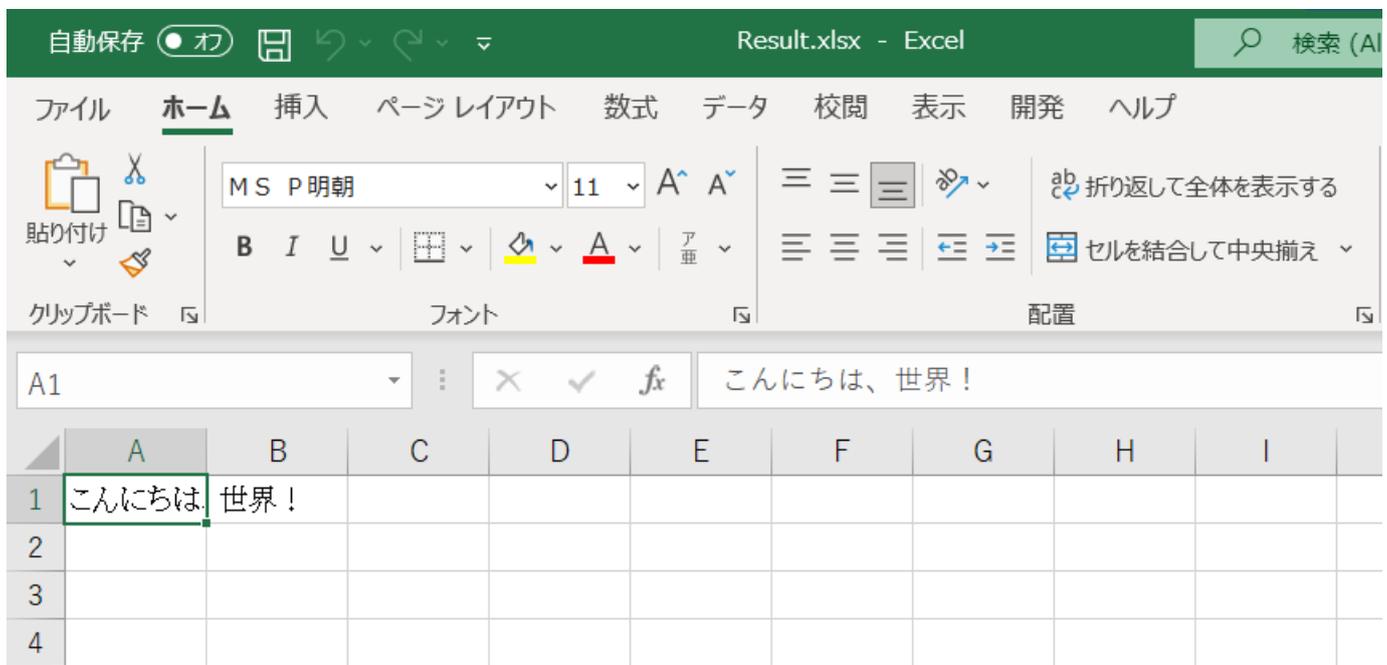
[前回](#)と[前々回](#)の記事では [Cloud Functions](#) で「[DioDocs \(ディオドック\)](#)」を使用した C# (.NET 6、8) の関数アプリケーションを作成し、Excel や PDF ファイルを出力する方法について紹介しました。

今回は Cloud Functions で DioDocs を利用する際に、日本語フォントを使用する Tips を紹介します。

セルに追加するテキストの日本語フォント (DioDocs for Excel)

セルに追加するテキストの日本語フォントを設定したい場合は、Font プロパティを使用します。

```
Workbook workbook = new Workbook();
workbook.Worksheets[0].Range["A1"].Font.Name = "MS P明朝";
```



セルではなくシート全体のフォントを設定したい場合はこちらのナレッジベースを参考にしてください。

[「シート全体のフォントを設定する方法」を見る](#)

ワークシートを PDF 出力する際の日本語フォント (DioDocs for Excel)

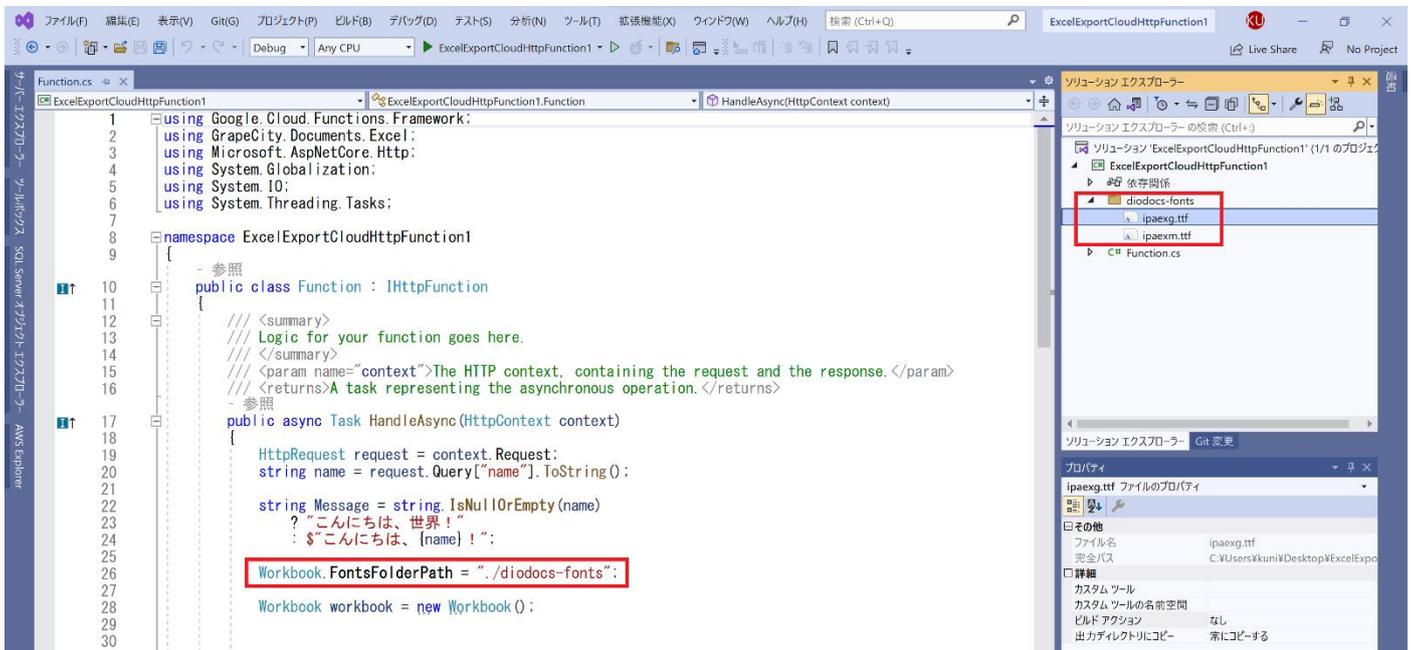
Cloud Functions で .NET 6、8 のランタイムが含まれる関数アプリケーションの実行環境 OS は、「[Ubuntu 22.04](#)」になっています。こちらには日本語を表示可能なフォント「[Droid Sans Fallback](#)」が含まれてはいますが、その他の日本語フォントは含まれていません。そのため、デフォルトの状態では PDF 出力を実行すると指定したフォントで表示されなかったり、文字化けが発生したり、文字列そのものが表示されない現象が発生します。

[「Linux 環境で PDF エクスポートすると文字化けが発生する」を見る](#)

そこで DioDocs for Excel を使用して PDF 出力を実行する場合は、あらかじめ使用する日本語フォントを追加し

ておく必要があります。本記事では「[IPAex フォント](#)」を使用します。

Cloud Functions では、関数アプリケーションのプロジェクト内のフォルダをそのまま参照することができます。プロジェクト内に任意のフォルダ（「diodocs-fonts」フォルダなど）を作成して使用するフォントファイルを格納します。



そのフォルダを `Workbook.FontsFolderPath` プロパティで設定します。

```
public async Task HandleAsync(HttpContext context)
{
    HttpRequest request = context.Request;
    string name = request.Query["name"].ToString();

    string Message = string.IsNullOrEmpty(name)
        ? "こんにちは、世界！"
        : $"こんにちは、{name}！";

    Workbook.FontsFolderPath = "./diodocs-fonts";

    Workbook workbook = new Workbook();
    workbook.Worksheets[0].Range["A1"].Font.Name = "IPAex ゴシック";
    workbook.Worksheets[0].Range["A1"].Value = Message;

    workbook.Worksheets[0].Range["A2"].Font.Name = "IPAex 明朝";
    workbook.Worksheets[0].Range["A2"].Value = Message;

    byte[] output;
```

```

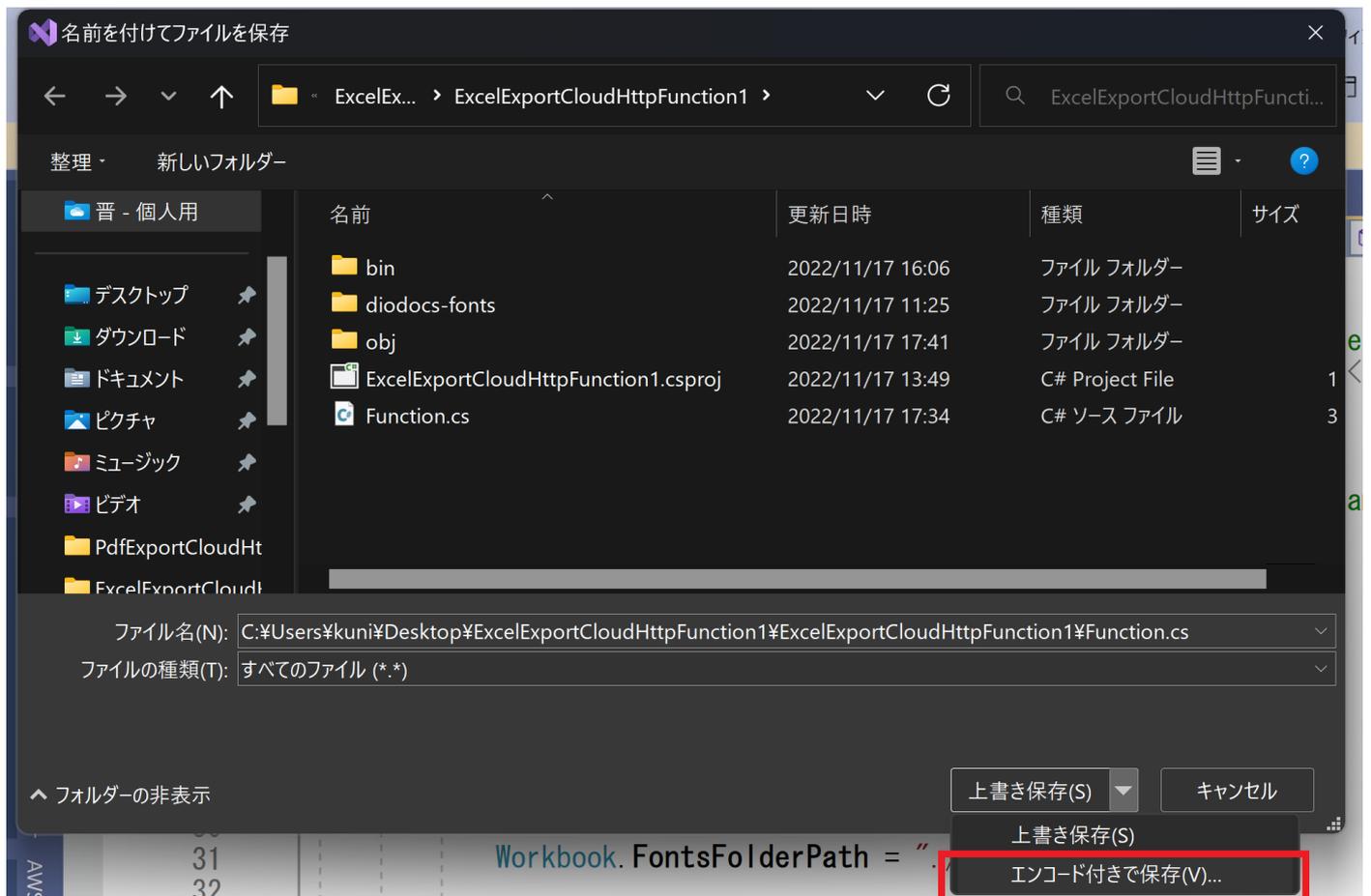
using (var ms = new MemoryStream())
{
    workbook.Save(ms, SaveFileFormat.Pdf);
    output = ms.ToArray();
}

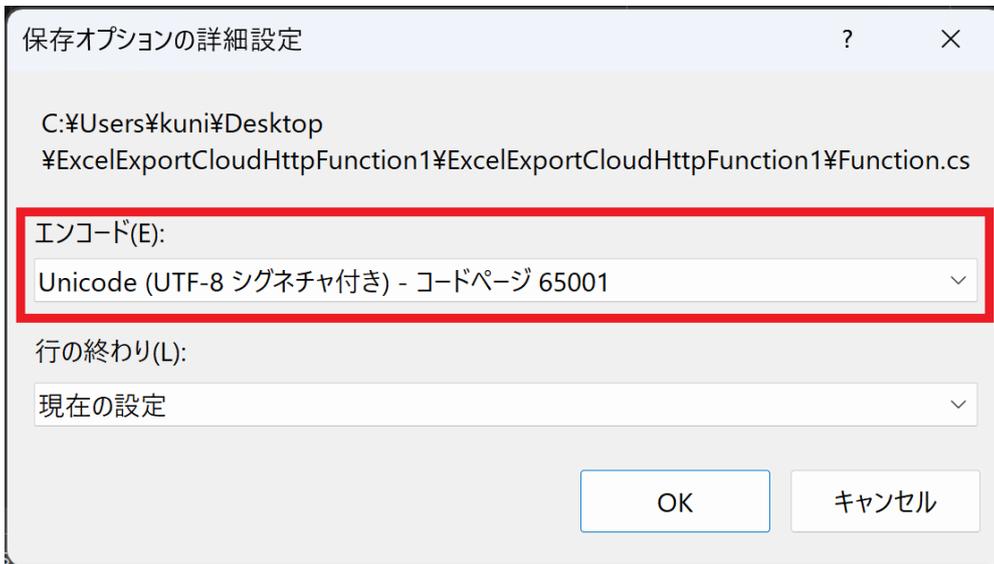
context.Response.ContentType = "application/pdf";
context.Response.Headers.Add("Content-Disposition", "attachment;filename=Result.pdf");
await context.Response.Body.WriteAsync(output);
}

```

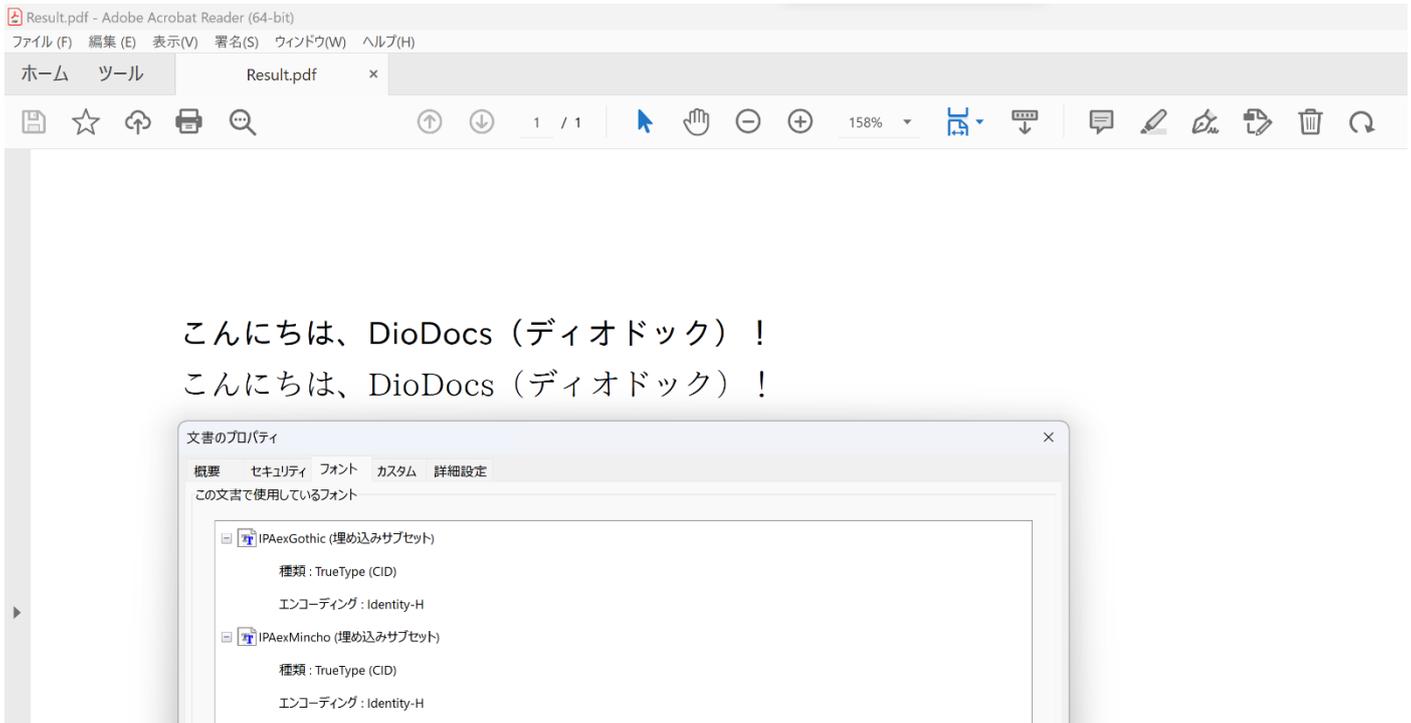
Visual Studio で開発する際の注意点

Visual Studio では Function.cs などのソースファイルを保存する際の文字コードはデフォルトが「Shift-JIS」となっています。そのままデプロイして実行すると Message にコード上で追加した日本語文字列が文字化けしてしまいます。この文字化けを解消するために Visual Studio でソースファイルを保存する際は「UTF-8」を設定して保存するようにしてください。





「IPAex フォント」を設定したセルに日本語の文字列を持つ Excel ワークブックを PDF ファイルへ出力すると、日本語が文字化けしたりすることなく「IPAex フォント」が設定されて正しく文字列が表示されていることが確認できます。



PDF ドキュメントを保存する際の日本語フォント（DioDocs for PDF）

DioDocs for PDF で作成した PDF ドキュメントで日本語フォントを利用する場合も、「ワークシートを PDF 出力する際の日本語フォント（DioDocs for Excel）」と同じ手順で日本語フォントを追加します。

以下のコードのように `Font.FromFile` メソッドで日本語フォントのファイルを設定して、「IPAex ゴシック」を設定した日本語の文字列を持つ PDF ドキュメントを出力すると、日本語が文字化けしたりすることなく「IPAex ゴシック」が設定されて正しく文字列が表示されていることが確認できます。

```

public async Task HandleAsync(HttpContext context)
{
    HttpRequest request = context.Request;
    string name = request.Query["name"].ToString();

    string Message = string.IsNullOrEmpty(name)
        ? "こんにちは、世界！"
        : $"こんにちは、{name}！";

    Font font1 = Font.FromFile(Path.Combine("diodocs-fonts", "ipaexg.ttf"));

    GcPdfDocument doc = new GcPdfDocument();
    GcPdfGraphics g = doc.NewPage().Graphics;

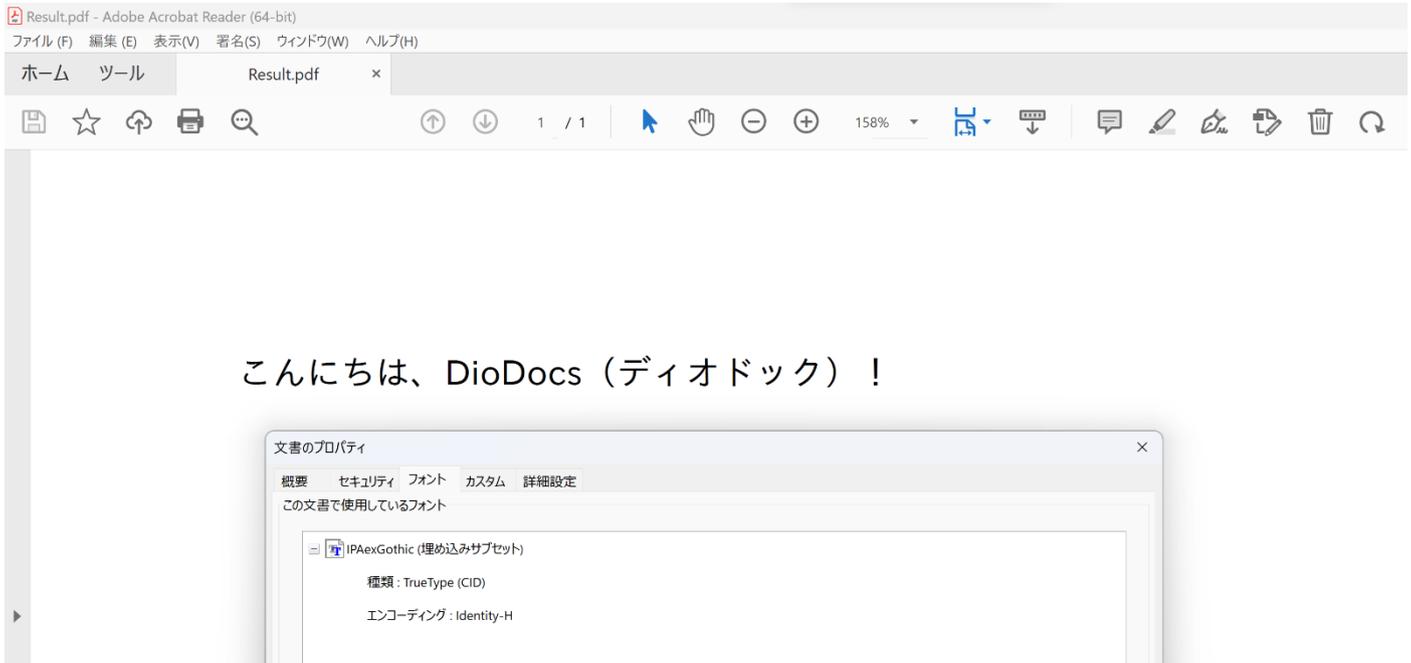
    g.DrawString(Message,
        new TextFormat() { Font = font1, FontSize = 12 },
        new PointF(72, 72));

    byte[] output;

    using (var ms = new MemoryStream())
    {
        doc.Save(ms, false);
        output = ms.ToArray();
    }

    context.Response.ContentType = "application/pdf";
    context.Response.Headers.Add("Content-Disposition", "attachment;filename=Result.pdf");
    await context.Response.Body.WriteAsync(output);
}

```



なお、こちら Visual Studio で開発する場合は、日本語の文字化けを回避するためにソースファイルの文字コードを「Shift-JIS」から「UTF-8」に変更しておく必要があります。

弊社 Web サイトでは、製品の機能を気軽に試せるデモアプリケーションやトライアル版も公開していますので、こちらをご確認いただければと思います。

- [デモアプリケーション（DioDocs for Excel）を試す](#)
- [デモアプリケーション（DioDocs for PDF）を試す](#)
- [トライアル版をダウンロードして試す](#)

また、ご導入前の製品に関するご相談やご導入後の各種サービスに関するご質問など、お気軽にお問合せください。

- [問合せ先を確認する](#)
- [個別相談会（Web 会議）について確認する](#)