

Visual COBOL チュートリアル

IIS と .NET COBOL コンポーネントを連携させた RESTful Web サービスの開発

1 目的

Visual COBOL では、他言語・他システムとの連携手法として、「Enterprise Server」を利用したサービス連携だけでなく、COBOL プログラムに一切の変更を行わずに、.NET 技術と連携可能な .NET COBOL 機能も提供しています。本機能を利用することで、C#、VB .NET などの .NET 言語で作成した RESTful Web サービス上で COBOL を利用するといった方法も可能となります。

このドキュメントでは .NET COBOL 機能を利用して C# で実装する RESTful Web サービスと COBOL の連携方法について説明します。

2 前提

本チュートリアルは、下記の環境を前提に作成されています。

- 開発クライアント ソフトウェア

Windows 11

COBOL 開発環境製品 Visual COBOL 10.0 for Visual Studio

RESTful Web サービスの実装やテストクライアントなどにおいて、C#、IIS サーバー、jQuery などの技術を利用したチュートリアルとなっておりますが、これらの技術に関する説明は本チュートリアルでは行っておりません。別途資料やオンライン文献などを参照ください。

- チュートリアル用サンプルプログラム

下記のリンクから事前にチュートリアル用のサンプルファイルをダウンロードして、任意のフォルダーに解凍してください。

このサンプルプログラムは、COBOL で作成された簡単な書籍情報を管理するアプリケーションであり、索引ファイルを利用しています。

[サンプルプログラムのダウンロード](#)

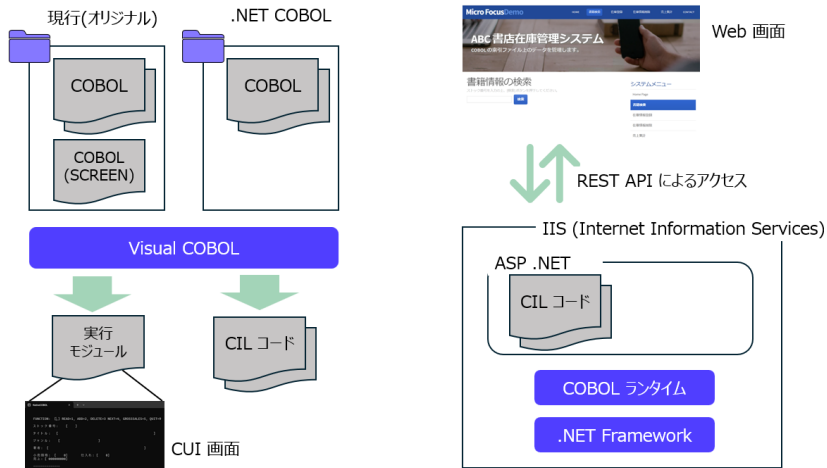
内容

- 1 目的
- 2 前提
- 3 チュートリアル手順
 - 3.1 今回作成するアプリケーション概要
 - 3.2 プロジェクトの作成と COBOL アプリケーションの確認
 - 3.2.1 プロジェクトの作成
 - 3.2.2 アプリケーションの動作確認
 - 3.3 .NET COBOL プロジェクトの作成
 - 3.4 C# による RESTful Web サービスの作成
 - 3.5 IIS サーバーへのサービス配置
 - 3.6 RESTful Web サービスの確認
- 4 補足
 - 4.1 IIS サーバーのインストール方法

3 チュートリアル手順

3.1 今回作成するアプリケーション概要

従来のコンソールアプリケーションである書籍情報を管理するアプリケーションを .NET COBOL の機能を利用して COBOL プログラムを変更することなく CIL コードを生成します。生成された CIL コードを利用して、Windows の IIS サーバーに RESTful Web アプリケーションとして登録した後、登録したアプリケーションが実際に動作することを確認します。



3.2 プロジェクトの作成と COBOL アプリケーションの確認

3.2.1 プロジェクトの作成

- 1) Visual Studio XXXX (XXXX はバージョン番号です。今回は 2022) を起動します。
- 2) [新しいプロジェクトの作成(N)] をクリックします。

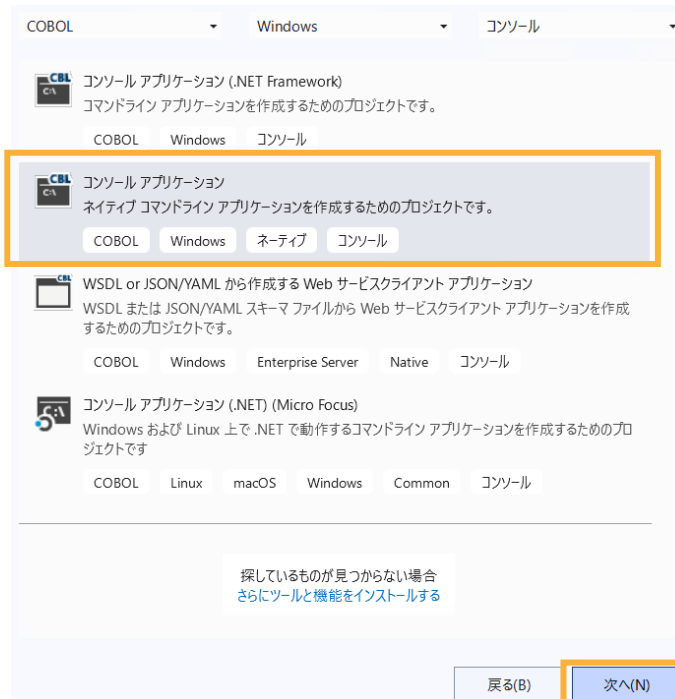


- 3) 以下のフィルタを設定し、[コンソールアプリケーション] を選択し、[次へ(N)] をクリックします。

全ての言語： COBOL

全てのプラットフォーム： Windows

全てのプロジェクトの種類： コンソール



- 4) 以下の入力を行い、[作成(C)] をクリックします。

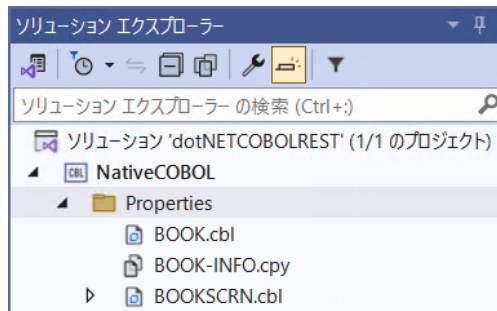
プロジェクト名: "NativeCOBOL"

ソリューション名: "dotNETCOBOLREST"



- 5) 自動作成された Program1.cbl は不要のため、削除してください。

- 6) チュートリアルファイル解凍フォルダー¥COBOL フォルダー配下の「BOOKSCRN.cbl」, 「BOOK.cbl」, 「BOOK-INFO.cpy」をプロジェクト内にドラッグアンドドロップします。



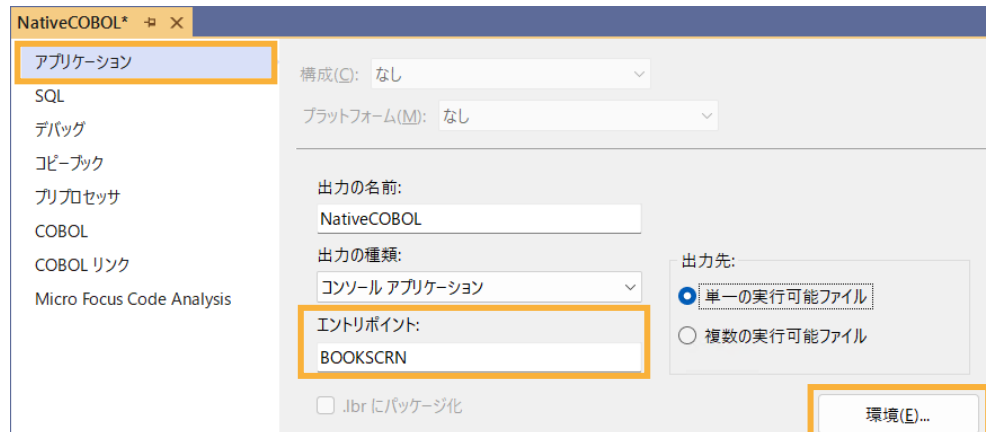
- 7) チュートリアルファイル解凍フォルダー配下の DAT フォルダーを任意のフォルダーにコピーします。フォルダー配下には、書籍情報を管理する索引ファイルが保存されています。
本チュートリアルでは、C:¥dotNETManageTutorial¥DAT としています。

3.2.2 アプリケーションの動作確認

- 1) NativeCOBOL プロジェクト名を選択し、マウスの右クリックにてコンテキストメニューを表示した上で、[プロパティ(R)]を選択します。



- 2) 「アプリケーション」タブを選択し、「エンリポイント」に“BOOKSCRN”を入力した上で、[環境(E)] をクリックします。



NativeCOBOL* 環境(E)...

アプリケーション

SQL

デバッグ

コピーブック

プリプロセッサ

COBOL

COBOL リンク

Micro Focus Code Analysis

構成(C): なし

プラットフォーム(M): なし

出力の名前: NativeCOBOL

出力の種類: コンソール アプリケーション

出力先: ☒ 単一の実行可能ファイル ☐ 複数の実行可能ファイル

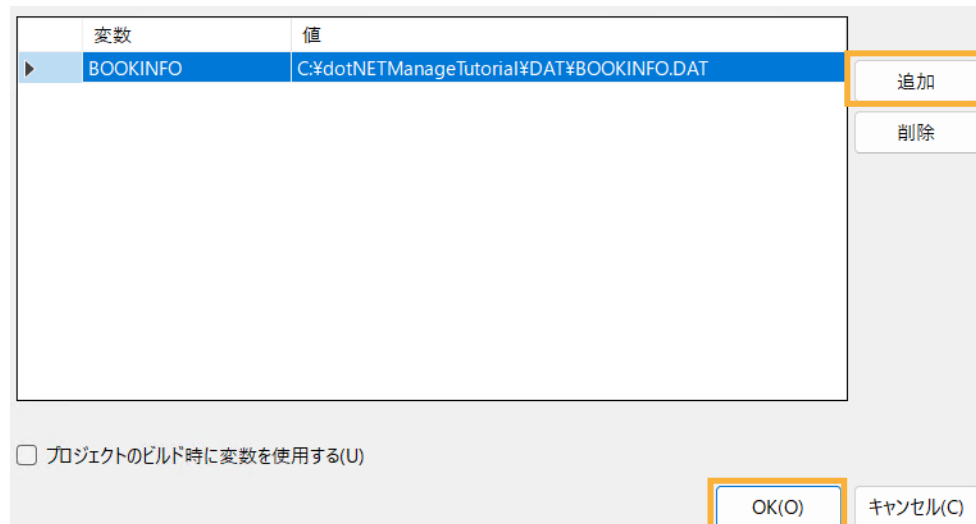
エンリポイント: BOOKSCRN

☐ .lbr にパッケージ化

- 3) [追加] をクリックし、以下の入力を行ったうえで、[OK(O)] をクリックします。

変数: “BOOKINFO”

値: “C:¥dotNETManageTutorial¥DAT¥BOOKINFO.DAT”



変数	値
BOOKINFO	C:¥dotNETManageTutorial¥DAT¥BOOKINFO.DAT

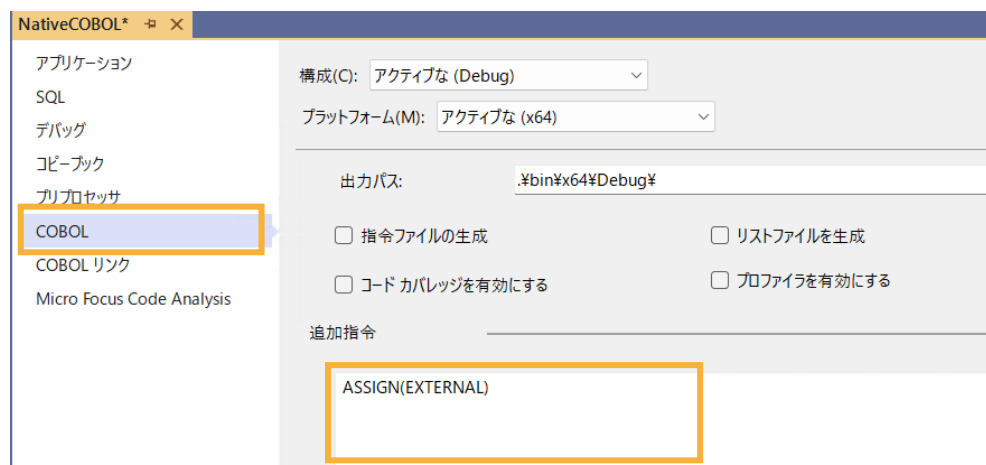
追加

削除

☐ プロジェクトのビルド時に変数を使用する(U)

OK(O) キャンセル(C)

- 4) 「COBOL」タブを選択し、「追加指令」欄に“ASSIGN(EXTERNAL)”を入力します。



NativeCOBOL* 環境(E)...

アプリケーション

SQL

デバッグ

コピーブック

プリプロセッサ

COBOL

COBOL リンク

Micro Focus Code Analysis

構成(C): アクティブな (Debug)

プラットフォーム(M): アクティブな (x64)

出力パス: ¥bin¥x64¥Debug¥

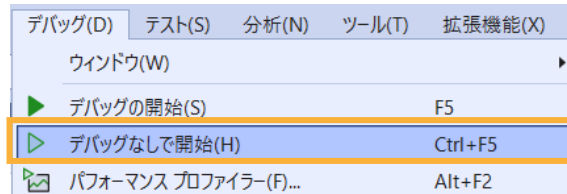
☐ 指令ファイルの生成 ☐ リストファイルを生成

☐ コード カパレレッジを有効にする ☐ プロファイラを有効にする

追加指令

ASSIGN(EXTERNAL)

- 5) 変更を保存します。
- 6) [デバッグ(D)] > [デバッグなしで開始(H)] をクリックします。



上記のような画面が表示されます。

以下の入力を行った後、Enter キーを打鍵すると、書籍情報が取得できます。

FUNCTION: "1"

ストック番号 : "1111"

```
NativeCOBOL
FUNCTION: [1] READ=1, ADD=2, DELETE=3 NEXT=4, GROSSSALES=S, QUIT=9
ストック 番号 : [1111]
タイトル : [LOAD OF THE RING ]
ジャンル : [FANTASY ]
著者 : [TOLKIEN ]
小売価格 : [ 1500] 仕入れ : [ 2000]
売上 : [ 000001000]
-----
STATUS: [00 ]
```

現在、ストック番号 : 4444 の書籍は未登録のため、以下の情報を入力し、Enter キーを打鍵することで、情報を追加します。

FUNCTION: "2"

ストック番号 : "4444"

タイトル : "ブレイブストーリー"

ジャンル : "ファンタジー"

著者 : "宮部みゆき"

小売価格 : "3000"

仕入れ : "1000"

売上 : "900"

```
FUNCTION: [2] READ=1, ADD=2, DELETE=3 NEXT=4, GROSSSALES=S, QUIT=9
ストック 番号 : [4444]
タイトル : [ブレイブストーリー ]
ジャンル : [ファンタジー ]
著者 : [宮部みゆき ]
小売価格 : [ 3000] 仕入れ : [ 1000]
売上 : [ 000000900]
-----
STATUS: [00 ]
```

実行後、READ 機能を用いて、追加した情報が参照できていることを確認してください。

確認した後、以下の入力後、Enter キーを打鍵することで、情報を削除します。

FUNCTION: "3"

ストック番号: "4444"

```
FUNCTION: [3] READ=1, ADD=2, DELETE=3 NEXT=4, GROSSSALES=S, QUIT=9
ストック 番号: [4444]
タイトル: [プレイブストーリー ]
ジャンル: [ファンタジー ]
著者: [宮部みゆき ]
小売価格: [ 3000] 仕入れ: [ 1000]
売上: [ 000000900]
-----
STATUS: [00 ]
```

実行後、READ 機能でストック番号: 4444 が削除されていることを確認してください。

FUNCTION=S の GROSSSALES 機能は登録された全書籍情報の売上額を集計します。

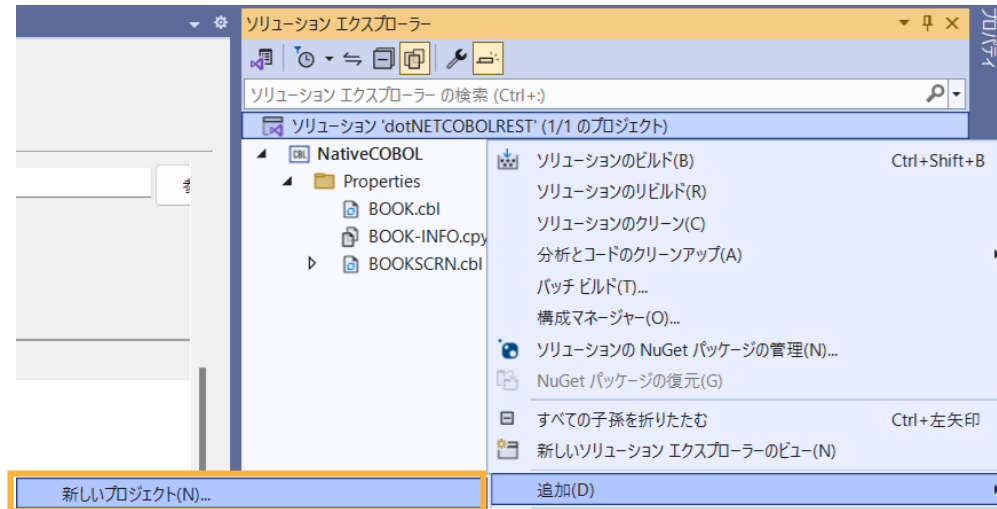
```
FUNCTION: [S] READ=1, ADD=2, DELETE=3 NEXT=4, GROSSSALES=S, QUIT=9
ストック 番号: [ ]
タイトル: [*****]
ジャンル: [*****]
著者: [*****]
小売価格: [ 0] 仕入れ: [ 0]
売上: [ 017000000]
-----
STATUS: [00 ]
```

確認後、FUNCTION=9 でアプリケーションを終了してください。

3.3 .NET COBOL プロジェクトの作成

本節では、さきほど確認した従来の COBOL プログラムを .NET COBOL としてコンパイルを行います。

- 1) Visual Studio IDE のソリューションエクスプローラーより、“dotNETCOBOLREST” ソリューション名を選択し、マウスの右クリックにてコンテキストメニューを表示した上で、[追加(D)] > [新しいプロジェクト(N)] をクリックします。



- 2) 以下のフィルタを設定し、[クラスライブラリ (.NET Framework)] を選択し、[次へ(N)] をクリックします。

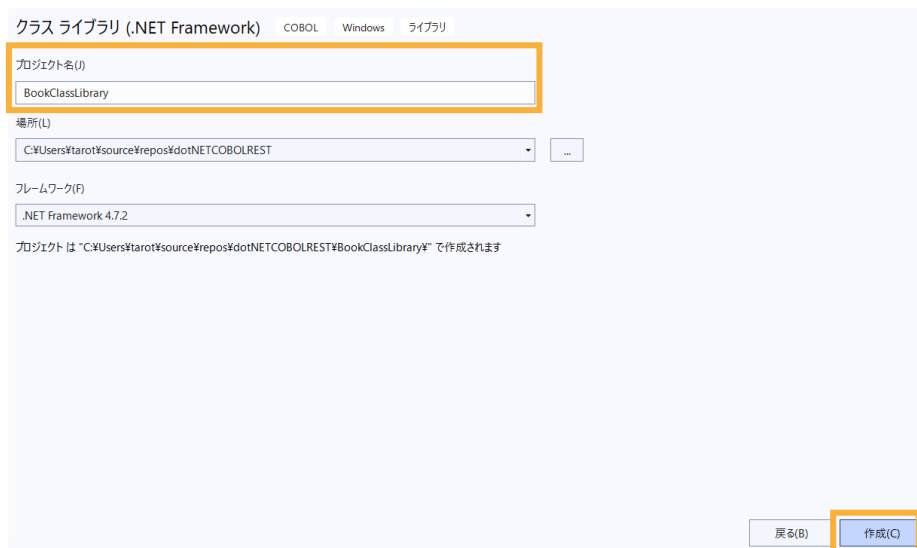
全ての言語： COBOL

全てのプラットフォーム： Windows

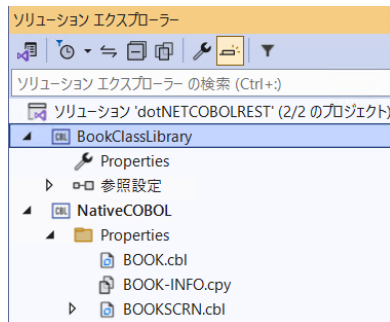
全てのプロジェクトの種類： ライブラリ



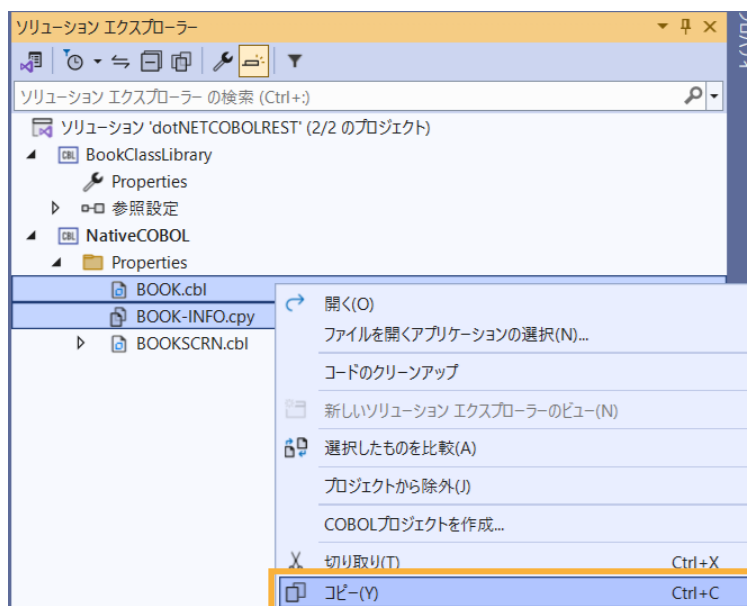
- 3) プロジェクト名に “BookClassLibrary” を入力し、[作成(C)] をクリックします。



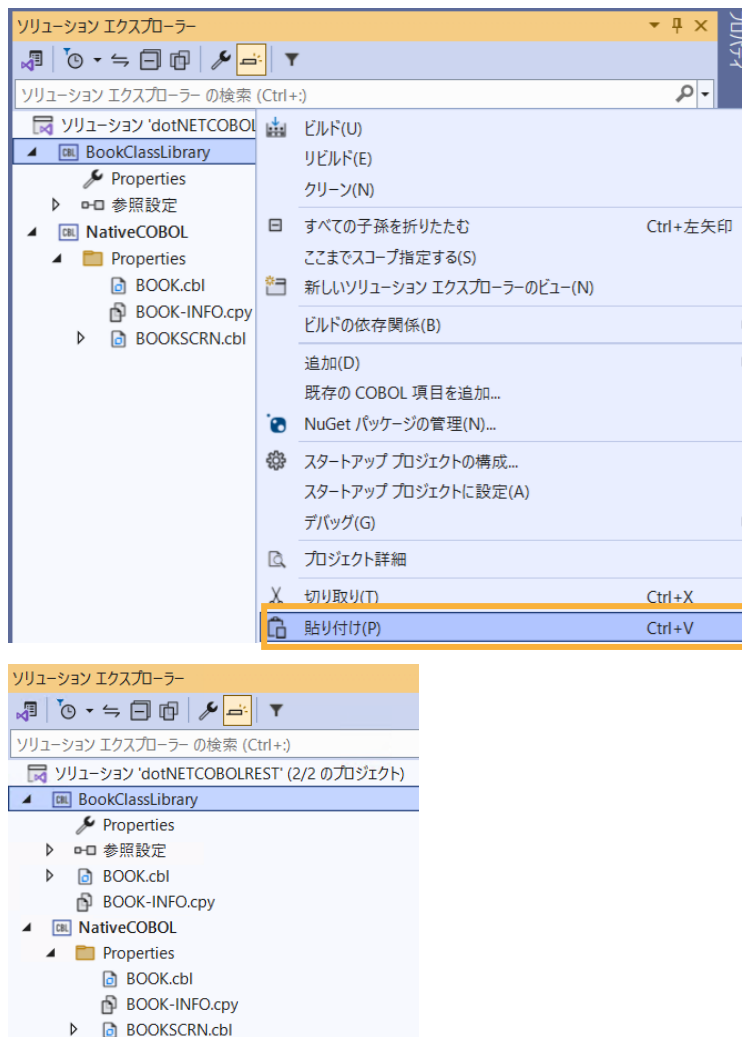
- 4) BookClassLibrary プロジェクト作成時に自動作成される Class1.cbl は不要なため、削除してください。



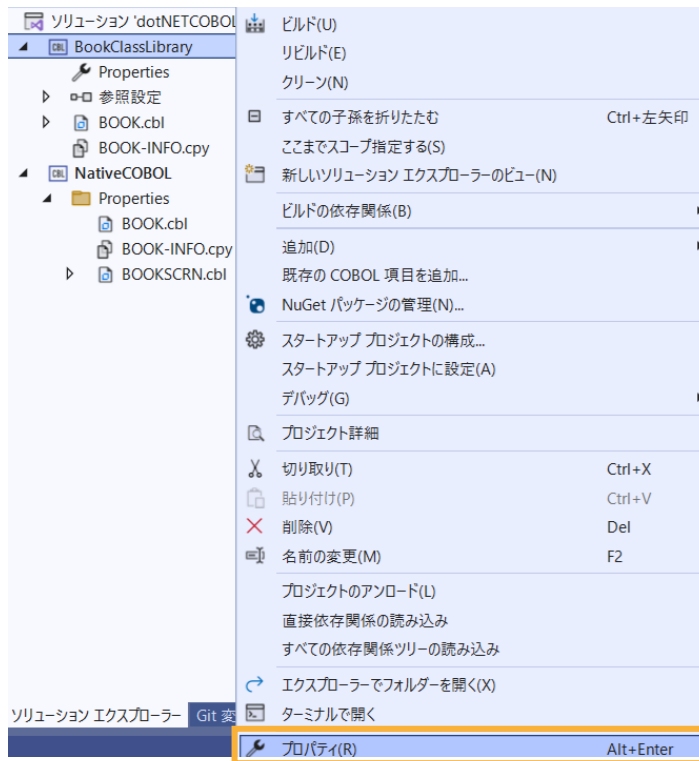
- 5) NativeCOBOL プロジェクト配下の「BOOK.cbl」,「BOOK-INFO.cpy」を選択し、マウスの右クリックにてコンテキストメニューを表示した上で、[コピー(Y)] をクリックします。



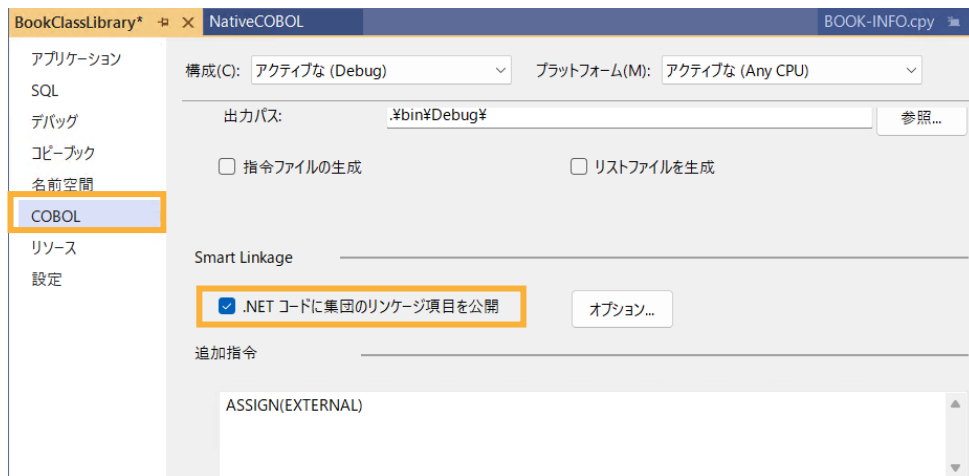
- 6) BookClassLibrary プロジェクト名を選択し、マウスの右クリックにてコンテキストメニューを表示した上で、[貼り付け(P)] をクリックします。



- 7) BookClassLibrary プロジェクト名を選択し、マウスの右クリックにてコンテキストメニューを表示した上で、[プロパティ(R)]をクリックします。



- 8) 「COBOL」タブを選択し、[.NET コードに集団のリンケージ項目を公開] にチェックを行い、追加指令に“ASSIGN(EXTERNAL)”を入力します。



補足)

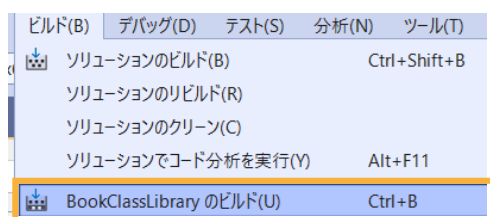
[.NET コードに集団のリンケージ項目を公開] にチェックを行うことで、ILSMARTLINKAGE というコンパイラ指令を設定したことになります。

本指令の詳細については、製品マニュアルトップより、[リファレンス] > [コンパイラ指令] > [.NET COBOL コマンド ライン コンパイラ指令] > [コード生成指令] > [ILSMARTLINKAGE] を参照してください。

- 9) 構成を“Release”に変更し、前手順と同様の設定を行い、保存します。



10) [ビルド(B)] > [BookClassLibrary のビルド(U)] を選択して、ライブラリのビルドを行います。

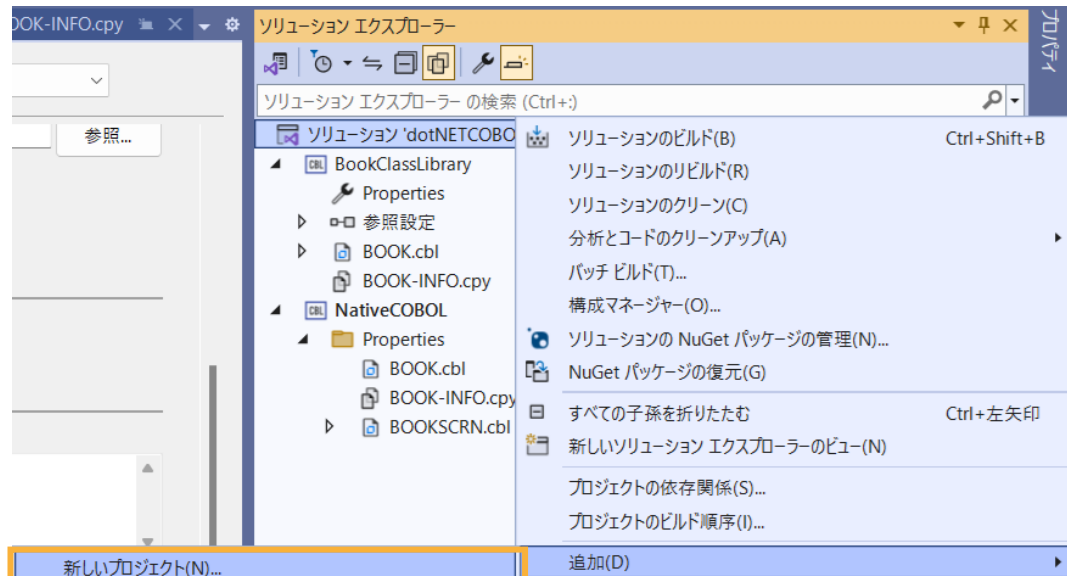


生成された BookClassLibrary.dll は Visual COBOL のコンパイラにより、コードの変更なく .NET Framework 上で動作するよう、CIL 形式で DLL を作成しています。このため、C#, VB .NET のような .NET 言語と連携することが可能になります。

3.4 C# による RESTful Web サービスの作成

さきほど作成した .NET COBOL のライブラリを RESTful Web サービスから呼び出します。RESTful Web サービスは、C# で作成します。

- 1) Visual Studio IDE のソリューションエクスプローラーより、“dotNETCOBOLREST” ソリューション名を選択し、マウスの右クリックにてコンテキストメニューを表示した上で、[追加(D)] > [新しいプロジェクト(N)] をクリックします。



- 2) 以下のフィルタを設定し、[ASP.NET Web アプリケーション (.NET Framework)] を選択し、[次へ(N)] をクリックします。

全ての言語： C#

全てのプラットフォーム： Windows

全てのプロジェクトの種類： Web



- 3) プロジェクト名に “RESTfulWS” を入力し、[作成(C)] をクリックします。

ASP.NET Web アプリケーション (.NET Framework) C# Windows クラウド Web

プロジェクト名(I)
RESTfulWS

場所(L)
C:\Users*tarot*\source\repos\dotNETCOBOLREST

フレームワーク(F)
.NET Framework 4.7.2

プロジェクトは "C:\Users*tarot*\source\repos\dotNETCOBOLREST\RESTfulWS" で作成されます

戻る(B) 作成(C)

- 4) 表示されたフォームにて「Web API」を選択し、[作成] をクリックします。

新しい ASP.NET Web アプリケーションを作成する

 空
ASP.NET アプリケーションを作成するための空のプロジェクト テンプレート。このテンプレートには内容はありません。

 Web Forms
ASP.NET Web Forms アプリケーションを作成するためのプロジェクト テンプレート。ASP.NET Web Formsを使用すると、一般的なドラッグアンドドロップのイベント駆動モデルを使用して、動的な Web サイトを構築できます。デザイン画面および数百のコントロールとコンポーネントを利用して、データ アクセスを備えた高度で強力な UI 主導のサイトを、短時間で作成できます。

 MVC
ASP.NET MVC アプリケーションを作成するためのプロジェクト テンプレート。ASP.NET MVC では、Model-View-Controller アーキテクチャを使用してアプリケーションを構築できます。ASP.NET MVC には、高速なテスト駆動開発をはじめとした最新の標準技術を使用するアプリケーションを作成するための多くの機能が備わっています。

 Web API
ブラウザやモバイル デバイスを含む幅広いクライアントに到達できる RESTful HTTP サービスを作成するためのプロジェクト テンプレート。

 シングル ページ アプリケーション
ASP.NET Web API を使用してリッチ クライアント側の JavaScript 駆動型 HTML5 アプリケーションを作成するためのプロジェクト テンプレート。Single Page Application では HTML5、CSS3、および JavaScript を使用したクライアント側インタラクションが含まれるリッチ ユーザー エクスperiencesを提供します。

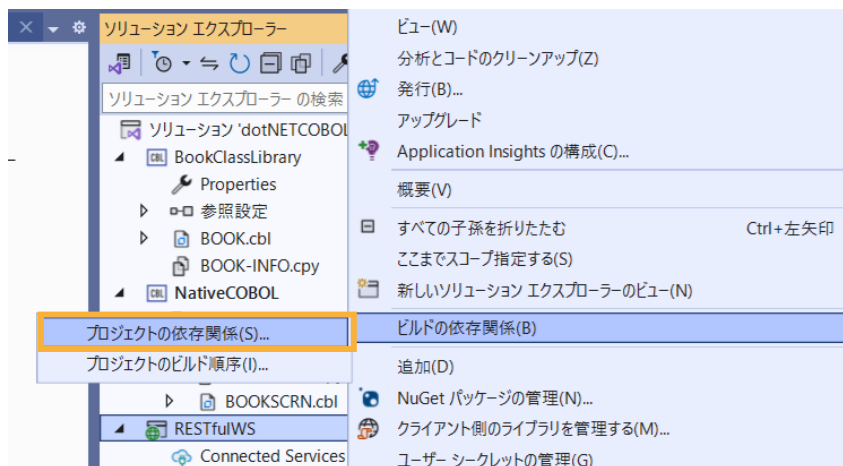
認証
なし

フォルダーおよびコア参照を追加する
☐ Web フォーム(F)
☒ MVC(M)
☒ Web API(W)

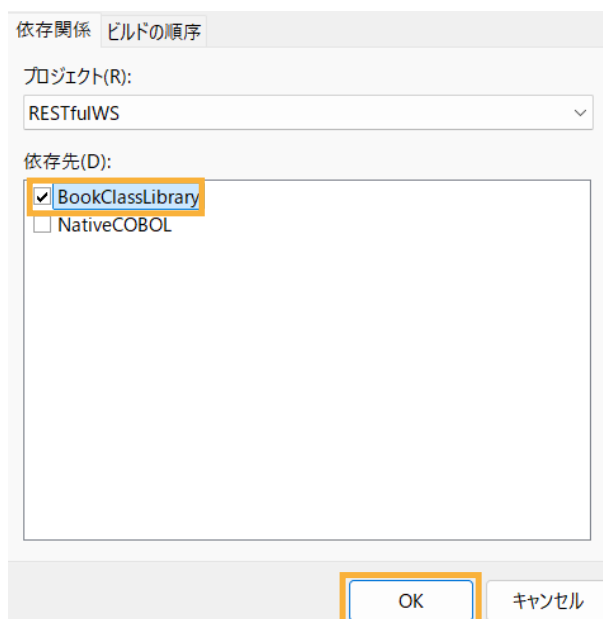
詳細設定
☒ HTTPS 用の構成(C)
☐ コンテナー サポート
(Docker Desktop が必要)
☐ 単体テストのためのプロジェクトも作成する(U)
RESTfulWS.Tests

戻る(B) 作成

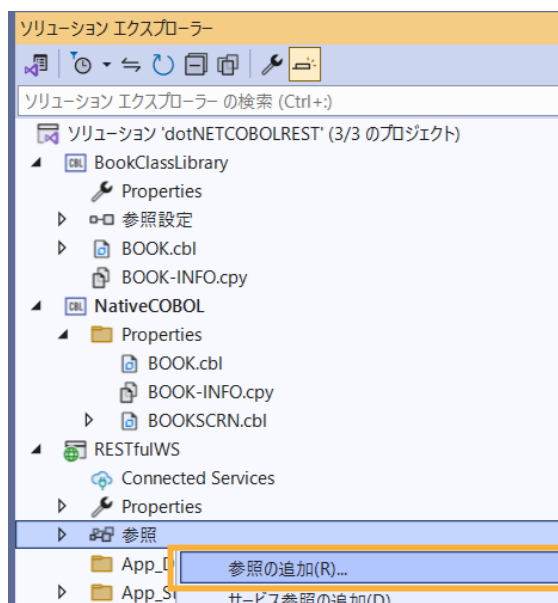
- 5) RESTfulWS プロジェクト名を選択し、マウスの右クリックにてコンテキストメニューを表示した上で、[ビルドの依存関係(B)] > [プロジェクトの依存関係(S)] を選択します。



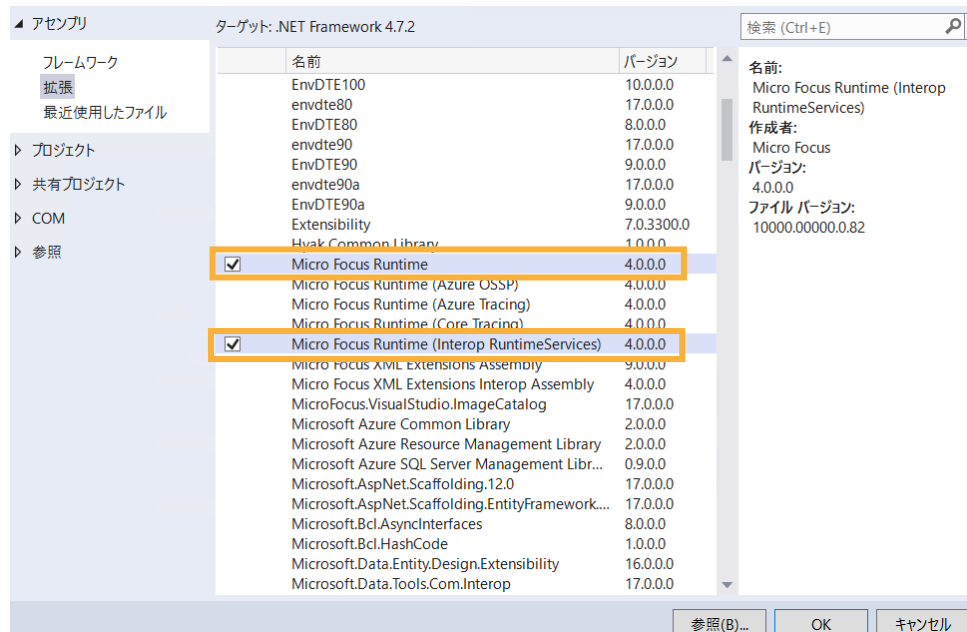
- 6) [BookClassLibrary] にチェックを行った後、[OK] をクリックします。



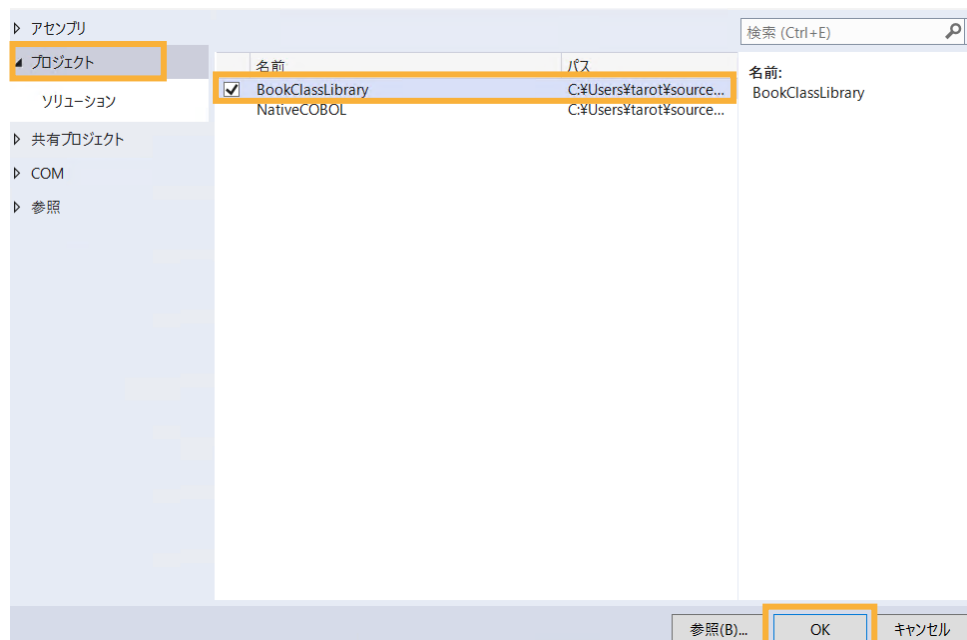
- 7) RESTfulWS プロジェクト配下の参照を選択し、マウスの右クリックにてコンテキストメニューを表示した上で、[参照の追加 (R)] を選択します。



- 8) 「アセンブリ」タブ配下の「拡張」タブを選択し、「Micro Focus Runtime」と「Micro Focus Runtime(Interop RuntimeServices)」のチェックを行います。

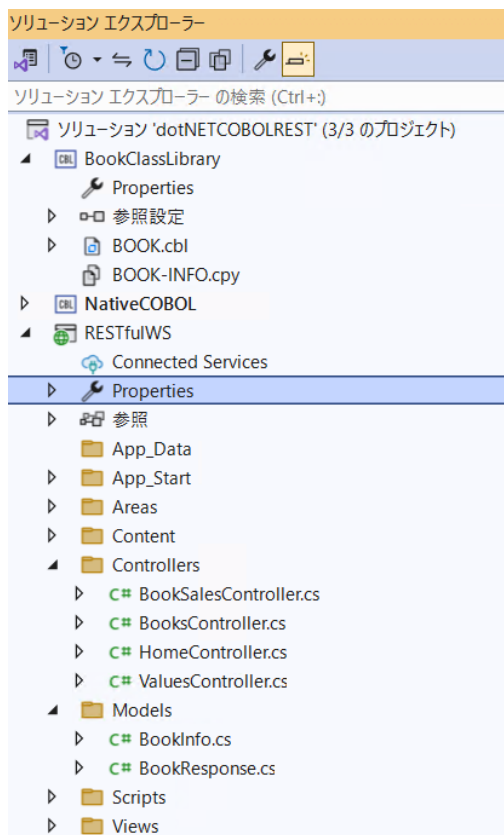
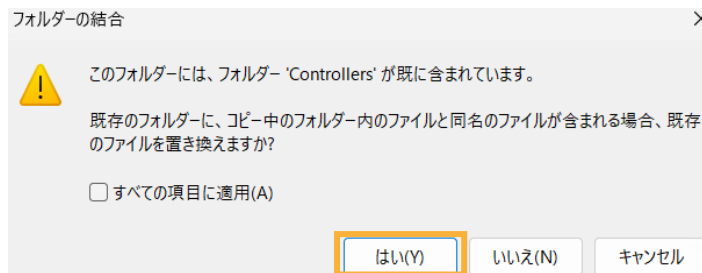


- 9) [プロジェクト] を選択し、「BookClassLibrary」にチェックを行った後、[OK] をクリックします。



- 10) チュートリアルファイル解凍フォルダー¥WebService 配下の Models と Controllers フォルダを RESTfulWS プロジェクトにドラッグアンドドロップします。

フォルダーの結合に関するダイアログは [はい(Y)] をクリックします。



- 11) BooksController.cs をダブルクリックして、COBOL 呼出し部の確認を行います。

.NET COBOL 機能により、従来の COBOL プログラムから PROGRAM-ID 名でクラス、および、メソッドが自動作成されます。今回の BOOK.cbl は PROGRAM-ID が “BOOK” であるため、BOOK クラスとなります。また、前手順で設定した ILSMARTLINKAGE コンパイラ指令により、LINKAGE SECTION に指定されていた LNK-FUNCTION, LNK-FILE-STATUS, LNK-B-DETAILS に対応するラッパークラスも合わせて作成されています。このラッパークラスを利用することで、.NET 言語と COBOL のデータ型の差異を意識することなく、一般的な C# のコーディングで記述できていることが確認できます。

```
BOOK book = new BOOK();
LnkFunction lnkFunction = new LnkFunction();
LnkFileStatus lnkFileStatus = new LnkFileStatus();
LnkBDetails lnkBDetails = new LnkBDetails();

RunUnit runUnit = new RunUnit();
runUnit.Add(book);
runUnit.Add(lnkFunction);
runUnit.Add(lnkFileStatus);
runUnit.Add(lnkBDetails);
lnkFunction.LnkFunction = "1";
lnkBDetails.LnkBStockno = id;
book.BOOK(lnkFunction, lnkBDetails, lnkFileStatus);
```

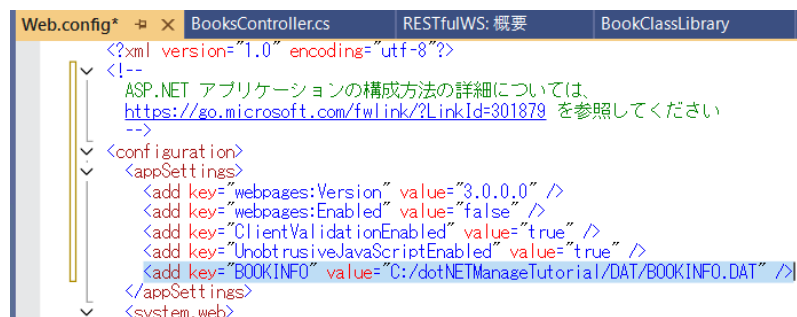
補足)

多くの COBOL プログラムは、一般的にシングルスレッドでの動作を前提としています。しかし、Web アプリケーションは一般的にマルチスレッドのため、WORKING-STORAGE SECTION のようなプロセス共有資源のスレッド間衝突による問題が発生する可能性があります。MicroFocus.COBOL.RuntimeServices.RunUnit は、これらの共有資源を各スレッドで独立して利用できるようにします。

- 12) アプリケーションの設定ファイルを修正します。RestfulWS プロジェクト配下の Web.config をダブルクリックします。
- 13) 以下の変更を行った後、保存します。

/configuration/appSettings 配下に、以下を追加

```
<add key="BOOKINFO" value="C:/dotNETManageTutorial/DAT/BOOKINFO.DAT" />
```



/configuration/system.webServer 配下の以下の行をコメントアウト。

```
<remove name="OPTIONSVerbHandler"/>
```

```

Web.config*  BooksController.cs  RESTfulWS: 概要  BookClassLibrary
<?xml version="1.0" encoding="utf-8"?>
<!--
  ASP.NET アプリケーションの構成方法の詳細については、
  https://go.microsoft.com/fwlink/?LinkId=301879 を参照してください
-->
<configuration>
  <appSettings>
    <add key="webpages:Version" value="3.0.0.0" />
    <add key="webpages:Enabled" value="false" />
    <add key="ClientValidationEnabled" value="true" />
    <add key="UnobtrusiveJavaScriptEnabled" value="true" />
    <add key="BOOK.INFO" value="C:/dotNETManageTutorial/DAT/BOOK.INFO.DAT" />
  </appSettings>
  <system.web>
    <compilation debug="true" targetFramework="4.7.2" />
    <httpRuntime targetFramework="4.7.2" />
  </system.web>
  <system.webServer>
    <handlers>
      <remove name="ExtensionlessUrlHandler-Integrated-4.0" />
      <!--
      <remove name="OPTIONSVerbHandler" />
      -->
      <remove name="TRACEVerbHandler" />
      <add name="ExtensionlessUrlHandler-Integrated-4.0" path="*.?" verb="*" />
    </handlers>
  </system.webServer>
</configuration>

```

/configuration/system.webServer 配下に、以下を追加。

```

<httpProtocol>
  <customHeaders>
    <add name="Access-Control-Allow-Origin" value="*" />
    <add name="Access-Control-Allow-Methods" value="*" />
    <add name="Access-Control-Allow-Headers" value="*" />
  </customHeaders>
</httpProtocol>

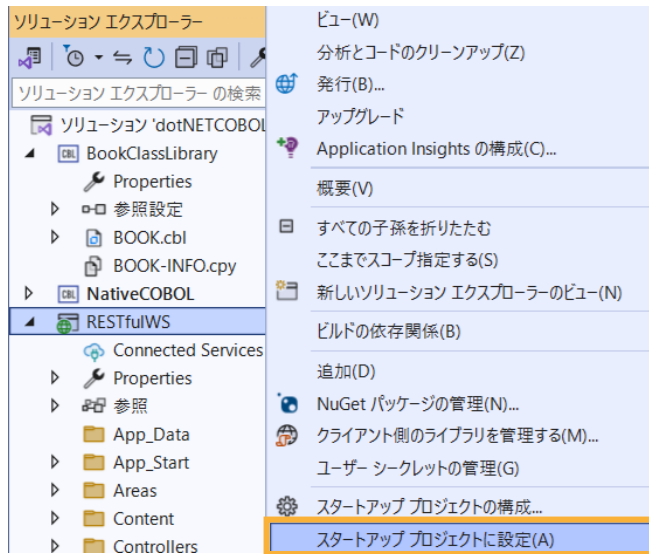
```

```

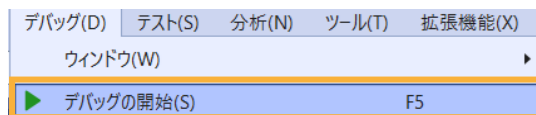
Web.config*  BooksController.cs  RESTfulWS: 概要  BookClassLibrary
<appSettings>
  <add key="webpages:Version" value="3.0.0.0" />
  <add key="webpages:Enabled" value="false" />
  <add key="ClientValidationEnabled" value="true" />
  <add key="UnobtrusiveJavaScriptEnabled" value="true" />
  <add key="BOOK.INFO" value="C:/dotNETManageTutorial/DAT/BOOK.INFO.DAT" />
</appSettings>
<system.web>
  <compilation debug="true" targetFramework="4.7.2" />
  <httpRuntime targetFramework="4.7.2" />
</system.web>
<system.webServer>
  <handlers>
    <remove name="ExtensionlessUrlHandler-Integrated-4.0" />
    <!--
    <remove name="OPTIONSVerbHandler" />
    -->
    <remove name="TRACEVerbHandler" />
    <add name="ExtensionlessUrlHandler-Integrated-4.0" path="*.?" verb="*" />
  </handlers>
  <httpProtocol>
    <customHeaders>
      <add name="Access-Control-Allow-Origin" value="*" />
      <add name="Access-Control-Allow-Methods" value="*" />
      <add name="Access-Control-Allow-Headers" value="*" />
    </customHeaders>
  </httpProtocol>
</system.webServer>

```

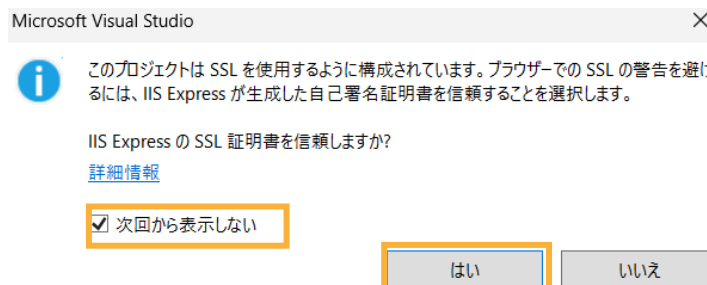
- ソリューションエクスプローラーの RESTfulWS プロジェクト名を選択し、マウスの右クリックにてコンテキストメニューを表示した上で、[スタートアッププロジェクトに設定(A)] を選択します。



- 15) [デバッグ(D)] > [デバッグの開始(S)] を選択し、デバッグを行います。



以下のようなダイアログが表示された場合は、「次回から表示しない」にチェックを行い、[はい] をクリックします。



さらに、[はい(Y)] をクリックします。



16) ブラウザーが起動します。

https://localhost:XXXXXX/api/Books/1111 にアクセスします。

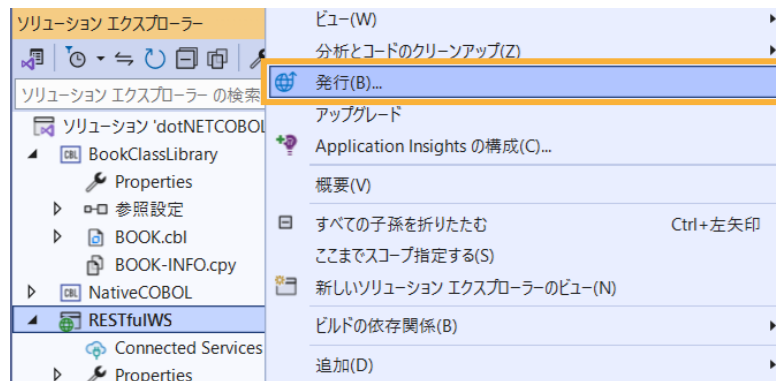
XXXXXX にはポート番号を入力してください。ポート番号は、起動時の URL から確認できます。

上記のように、ストック番号： 1111 の情報が戻されることを確認して、ブラウザーをクローズしてください。



3.5 IIS サーバーへのサービス配置

1) RESTfulWS プロジェクト名を選択し、マウスの右クリックにてコンテキストメニューを表示した上で、[発行(B)] を選択します。



2) 「フォルダー」を選択し、[次へ(N)]をクリックします。

公開

現在公開している場所

ターゲット



Azure
 アプリケーションを Microsoft クラウドにホストします



Docker Container Registry
 アプリケーションを Docker イメージに対応したサポートされている Container Registry に発行します



フォルダー
 アプリケーションをローカル フォルダーまたはファイル共有に発行します



FTP/FTPS サーバー
 アプリケーションを FTP/FTPS サーバーに発行します



Web サーバー (IIS)
 Web 配置または Web 配置パッケージを使用してアプリケーションを IIS に発行します



プロファイルのインポート
 アプリを配置するための発行設定をインポートします

戻る(B)

次へ(N)

完了(F)

キャンセル(C)

- 3) そのまま [完了(F)] をクリックします。

公開

ローカルまたはネットワーク フォルダーへのパスを指定してください

ターゲット

フォルダーの場所

bin¥app.publish¥

参照(R)...

場所

ローカル フォルダーの場合は、プロジェクトへの完全パスまたは相対パスのいずれかを指定できます。例:

- publish¥ (相対パス)
- C:¥Users¥Username¥Documents (完全パス)

ネットワーク フォルダーの場合、¥¥ を使用してから、コンピューター名または IP アドレスのいずれかを使用する必要があります。例:

- ¥¥server1¥fileshare1
- ¥¥192.168.1.17¥fileshare1

戻る(B)

次へ(N)

完了(F)

キャンセル(C)

今回は出力先のフォルダーを変更していないため、プロジェクトが保存されたフォルダー配下が出力先となります。

- 4) そのまま [閉じる] をクリックします。

実行プロファイル作成の進行状況

ターゲット
場所
完了

 発行プロファイル 'C:\Users\%tarot%\source\%repos%\dotNETCOBOLREST\%RESTfulWS%\Properties\PublishProfiles\%FolderProfile.pubxml' が作成されました。

☐ 成功時に自動的に閉じる

戻る(B) 次へ(N) **閉じる** キャンセル(C)

- 5) [発行(U)] をクリックします。

RESTfulWS: 公開
Web.config BooksController.cs BookClassLibrary NativeCOBOL

概要
接続済みサービス
公開

FolderProfile.pubxml
フォルダー

 発行(U)

+ 新しいプロファイル その他のアクション

 公開準備が完了しました。

設定

ターゲットの場所 bin\%app.publish%

既存のファイルを削除 false

構成 Release

すべての設定を表示

完了するとメッセージが表示されます。

RESTfulWS: 公開
Web.config BooksController.cs BookClassLibrary NativeCOBOL

概要
接続済みサービス
公開

FolderProfile.pubxml
フォルダー

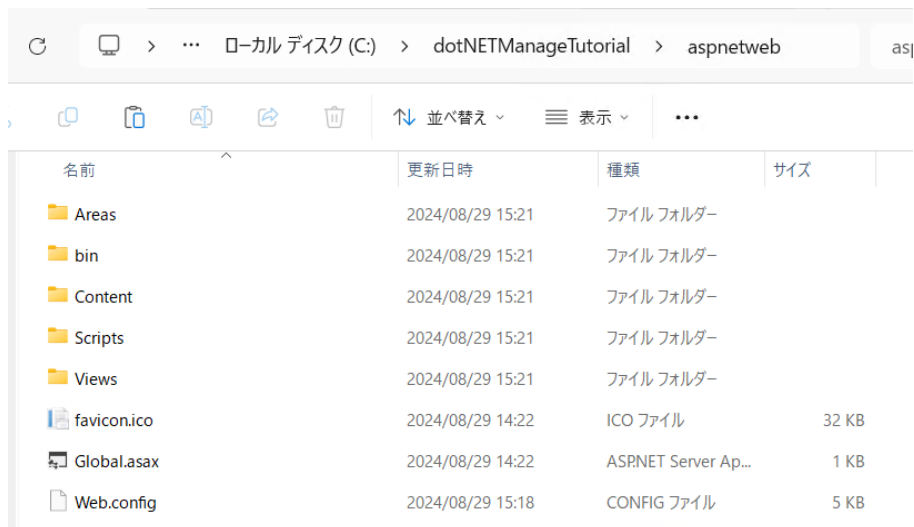
 発行(U)

+ 新しいプロファイル その他のアクション

 2024/08/29 の 15:18 に公開が成功しました。
移動

- 6) IIS サーバーに公開するため、以下のフォルダーを作成したうえで、前手順で出力されたリソース (bin\%app.publish 配下のフォルダー・ファイル) をコピーしてください。

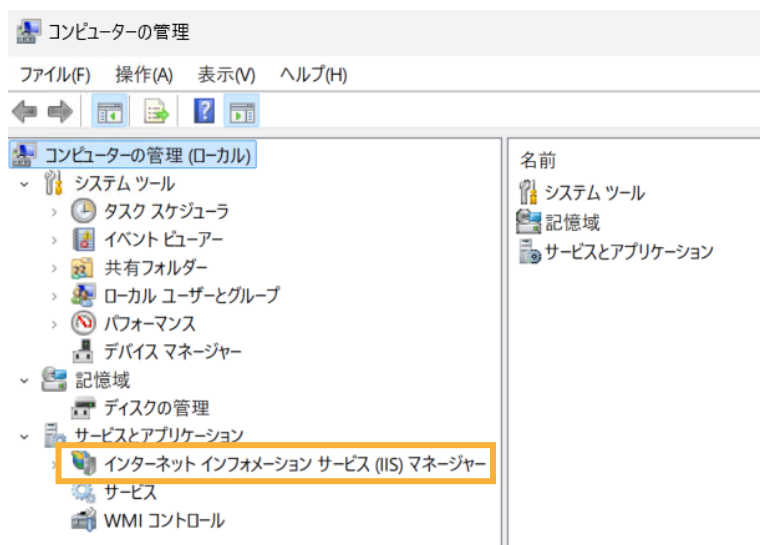
C:\%dotNETManageTutorial%\aspnetweb



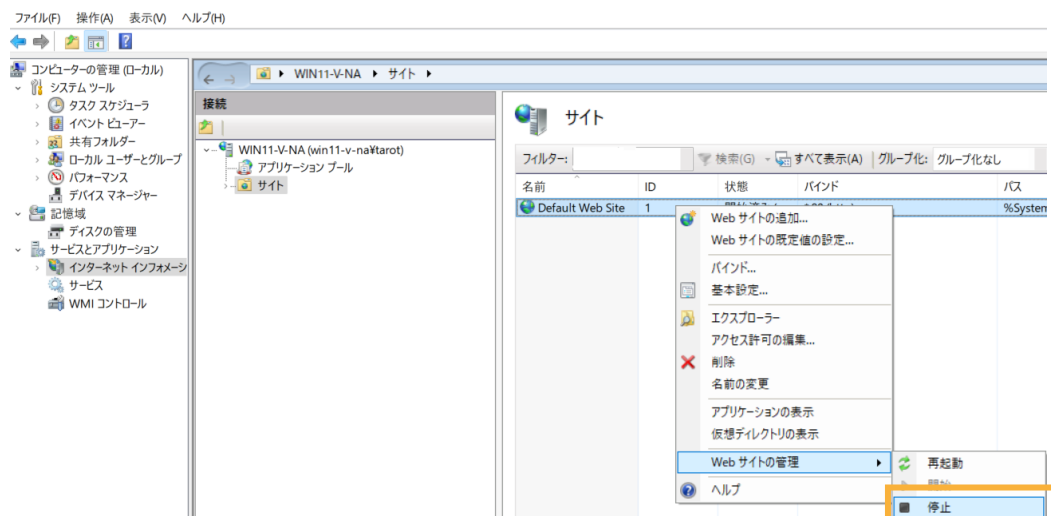
- 7) Windows のスタートメニューの上で、右クリックにてコンテキストメニューを表示し、[コンピューターの管理] を選択します。



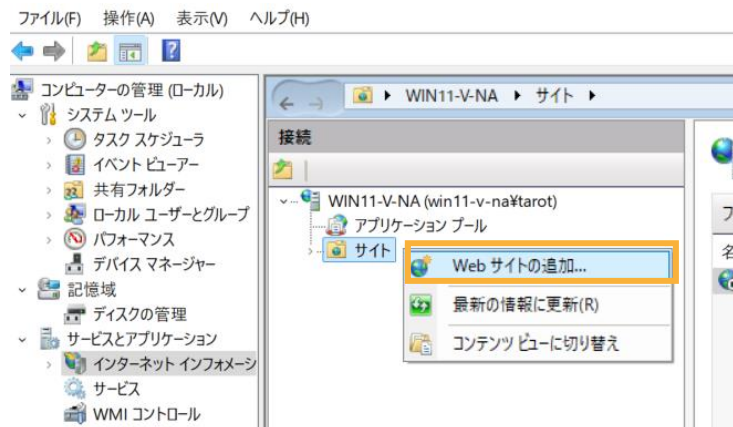
- 8) [サービスとアプリケーション] 配下の [インターネットインフォメーションサービス (IIS) マネージャー] をクリックします。
本項目が表示されていない方は、IIS がインストールされていません。4.1 の手順にてインストールを行ってください。



- 9) 本チュートリアル用に新規サイトを作成するため、[サイト] > [Default Web Site] を選択し、マウスの右クリックでコンテキストメニューを開き、[停止] をクリックします。



- 10) [サイト] を選択し、マウスの右クリックにてコンテキストメニューを表示した上で、[Web サイトの追加] をクリックします。



- 11) 以下の入力を行い、[OK] をクリックします。

サイト名 : “tut”

物理パス : “C:\dotNETManageTutorial\aspnetweb”

Web サイトを直ちに開始する : チェックをはずす

Web サイトの追加

サイト名(S):
tut

アプリケーション プール(L):
tut

選択(E)...

コンテンツ ディレクトリ

物理パス(P):
C:\dotNETManageTutorial\aspnetweb

バススレー認証

接続(C)... テスト設定(G)...

バインド

種類(T): http IP アドレス(I): 未使用の IP アドレスすべて ポート(O): 80

ホスト名(H):

例: www.contoso.com または marketing.contoso.com

☐ Web サイトを直ちに開始する(M)

OK キャンセル

以下のダイアログには、[はい(Y)] をクリックします。

Web サイトの追加

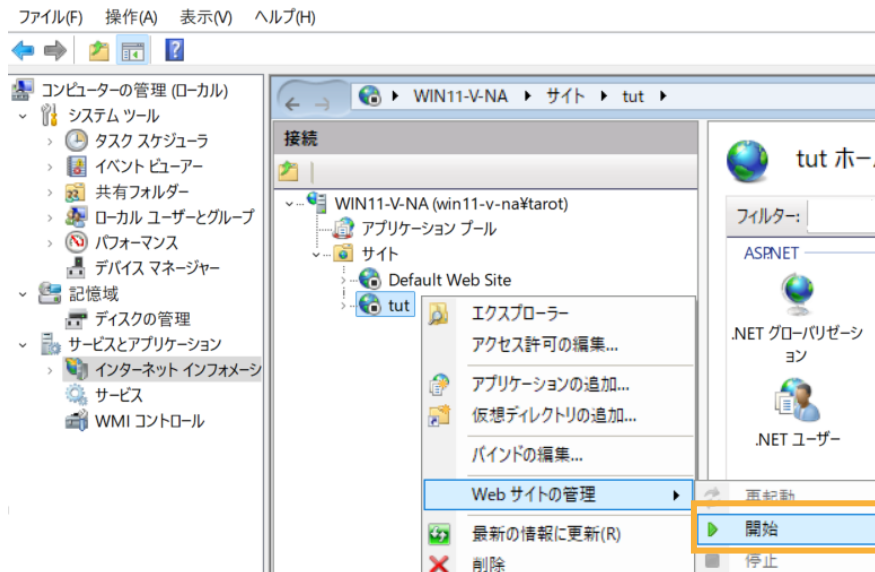
?

バインド "80" は別のサイトに割り当てられています。このサイトに同じバインドを割り当てると、いずれか一方のサイトしか開始できなくなります。重複するバインドを追加しますか?

はい(Y) いいえ(N)

3.6 RESTful Web サービスの確認

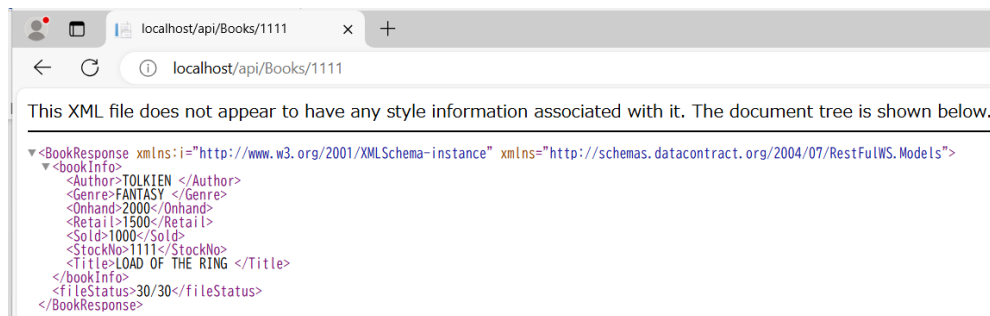
- 1) 前手順で使用した “インターネット インフォメーションサービス (IIS) マネージャー” に戻り、“tut” サイトを選択し、画面右側より [開始] をクリックします。



- 2) Web ブラウザーを開き、以下の URL にアクセスします。

<http://localhost/api/Books/1111>

さきほどのデバッグ時と同様、書籍情報が JSON 形式で戻されているため、C# で作成された RESTful Web サービスが COBOL プログラムを正しく呼び出していることが分かります。



- 3) チュートリアルファイル解凍フォルダー配下の WebClient フォルダー配下の Search.html を Web ブラウザーで開いてください。



こちらは、今回作成した RESTful Web サービスを呼び出す Web 画面となります。さきほど確認したように REST API を介して JSON 形式でデータを送受信を行うモダンなシステムインターフェースとなっています。

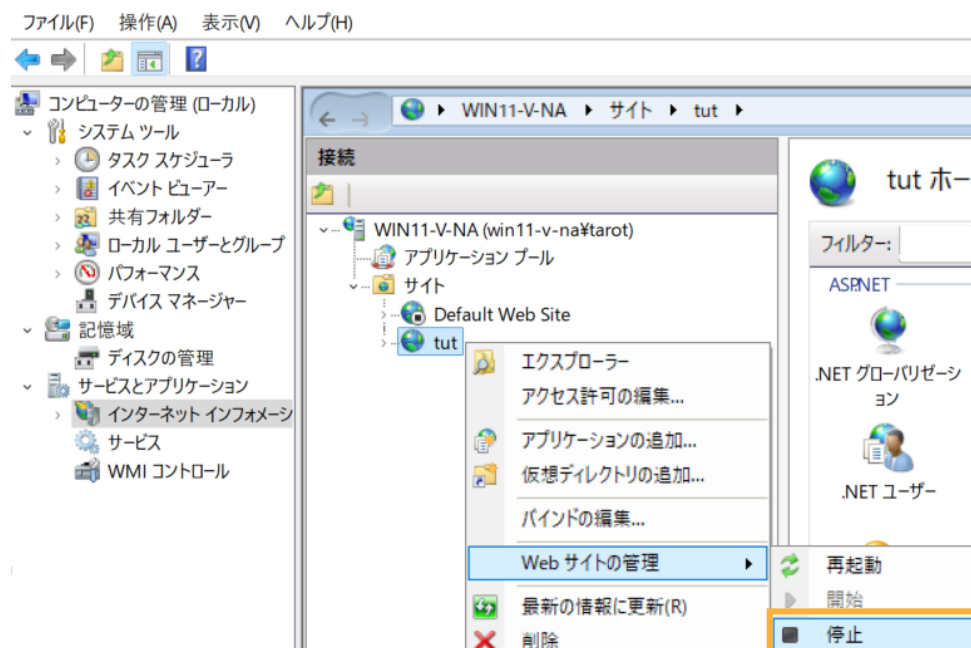
例えば、“1111”を入力して、[検索] をクリックすると、以下のように REST API から戻されたデータが表示されます。

書籍情報の検索

ストック番号を入力の上、[検索] ボタンを押下してください。

1111	検索
ストック番号	1111
タイトル	LOAD OF THE RING
ジャンル	FANTASY
著者	TOLKIEN
小売価格	1500
仕入れ数	2000
売り上げ数	1000

- 4) “インターネット インフォメーションサービス (IIS) マネージャー” に戻り、“tut” サイトを選択し、画面右側より [停止] をクリックします。



4 補足

4.1 IIS サーバーのインストール方法

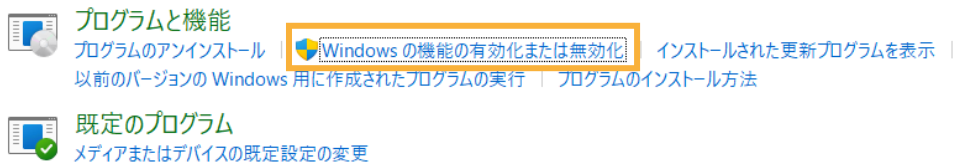
- 1) Windows でコントロールパネルを開きます。
- 2) [プログラム] をクリックします。

コンピューターの設定を調整します

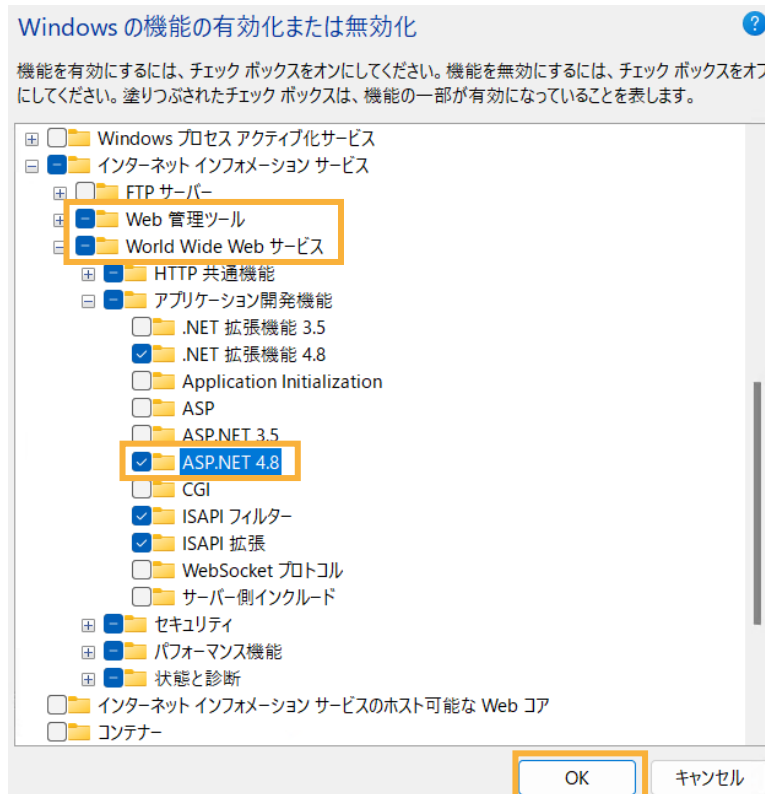
表



- 3) [Windows の機能の有効化または無効化] をクリックします。



- 4) 以下のチェックを行い、[OK] をクリックします。
 - 「インターネット インフォメーションサービス」配下の「Web 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」配下の「World Wide Web サービス」
 - 「インターネット インフォメーションサービス」 > 「World Wide Web サービス」 > 「アプリケーション開発機能」配下の「ASP.NET 4.8」



自動的にチェックされているものはそのまま構いません。

インストール完了後、[閉じる] をクリックします。

免責事項

ここで紹介したソースコードは、機能説明のためのサンプルであり、製品の一部ではございません。ソースコードが実際に動作するか、御社業務に適合するかなどに関しまして、一切の保証はございません。ソースコード、説明、その他すべてについて、無謬性は保障されません。

ここで紹介するソースコードの一部、もしくは全部について、弊社に断りなく、御社の内部に組み込み、そのままご利用頂いても構いません。

本ソースコードの一部もしくは全部を二次的著作物に対して引用する場合、著作権法に基づき、適切な扱いを行ってください。