Antenna House PDF Tool API V8.0 サンプルコードのビルドと実行手順・(Windows)

アンテナハウス株式会社

目次

1.はじめに

本書は Windows 環境におけるサンプルコードのビルド及び実行の手順書です。
最初に、Windows における開発環境の構築を解説します。
次に、PDF Tool API のサンプルコードのビルド及び実行に関して解説します。具体的には、各種インターフェースに分け、手順を解説します。解説するインターフェースは C++、C#(.NET 8)、C#(.NET interface)、Java です。ここでは主に Visual Studio 2022 における開発環境構築を説明します。
最後に、Visual Studio Code を利用した.NET の開発に関して、環境構築、プロジェクト作成、ビルドと実行方法を説明します。

2.PDF Tool API の開発環境を整える

ここでは、Windows における PDF Tool API の環境整備について説明します。
言い換えれば、サンプルコードをビルドするための下準備です。

詳細なインストール方法についてはライブラリ版マニュアルの『インストール／アンインストール』をご参照ください。

2.1.ライブラリファイル、ヘッダーファイルを配置する

2.1.1. インストーラによる配置

(1) 「Setup-Windows¥AHPDFToolLib_V80_***_x64.exe」をダブルクリックするなどして起動します。
「***」には R1、MR1 などの改訂バージョン名が入ります。

(2) ダイアログの指示にしたがってインストールを行います。

(3) インストール途中に、Microsoft Visual C++ 2022 ランタイムライブラリのインストールを促すダイアログが表示された場合は、指示にしたがってインストールを行ってください。

(4) デフォルトのインストール先は以下のフォルダパスです。

{システムドライブ}: ¥AHPDFToolLib_80

2.2.ライセンスファイルを配置する

2.2.1. インストーラによる配置

- ライセンスファイル「ptalic.dat」が、インストーラにより以下のフォルダに配置されます。

  ライセンスファイルの配置先：{インストールフォルダ}¥License

- 環境変数「PTL80_LIC_PATH」に上記フォルダパスが設定されます。
- インストールされるライセンスファイルは、インストール後 30 日間有効の評価版ライセンスです。
  出力 PDF に透かしが入るなど、評価版の動作には制限事項があります。
- 弊社より発行するライセンスファイルで上書きすることで正規版ライセンスに置き換えることが可能です。
- 詳細はライブラリ説明書に記載の『ライセンスファイルについて』をご参照ください。

2.2.2. インストーラを利用しない配置

◆ 配置方法 1

(1) 弊社より発行するライセンスファイルを開発環境の任意の場所に配置します。

(2) 環境変数「PTL80_LIC_PATH」を作成し、配置したフォルダパスを設定します。

◆ 配置方法 2

(1) ライセンスファイルを、PDF Tool API のモジュールファイル「PdfTk80.dll」と同じ場所に配置します。
この場合、環境変数「PTL80_LIC_PATH」の作成は必要ありません。

詳細はライブラリ説明書に記載の『ライセンスファイルの参照先指定』をご参照ください。

2.3.フォントの準備

テキスト透かしを挿入したりページ上に文字を描画するには、フォント情報が必要です。
Windows 版では、PDF Tool API はシステムのフォントフォルダに存在するフォントを参照します。

2.3.1. フォントの場所

Windows では、オペレーティングシステムの仕様によりフォントファイルは下記のフォントフォルダに存在しています。

{システムドライブ}:¥WINDOWS¥Fonts

{システムドライブ}:¥Users¥{ユーザー名}¥AppData¥Local¥Microsoft¥Windows¥Fonts

上記のフォントフォルダとは異なる場所にあるフォントを参照する場合には、「フォント構築ファイル」を設定します。

2.3.2. フォント構築ファイルの作成

(1) フォント構築ファイルは、下記の場所にあります。

{インストールフォルダ}¥fontconfig

(2) fontconfig フォルダ内には以下の 2 つのファイルがあります。

font-config.xml ：フォント構築ファイルのひな型

font-config.dtd ：font-config.xml の定義ファイル

(3) 「font-config.xml」の「font-folder path」タグに、フォントファイルが存在するフォルダパスを記述します。

（例）：(Windows 環境で「C:¥TestFont」フォルダを指定したい場合)

```
<font-config>
  <font-folder path="C:¥TestFont"></font-folder>
</font-config>
```

(4) font-config.xml と font-config.dtd を任意の場所に配置します。

(5) 環境変数「PTL80_FONT_CONFIGFILE」を作成し、font-config.xml のフルパスを設定します。

2.4.Windows 開発・実行環境における環境変数のまとめ

環境変数名	設定値	設定が必要な場合
PTL80_LIC_PATH（※1）	ライセンスファイル「ptalic.dat」の配置フォルダパス	「PdfTk80.dll」とは異なる場所にライセンスファイルを配置する場合
PTL80_FONT_CONFIGFILE（※2）	フォント構築ファイル「font-config.xml」のフルパス	システムのフォントフォルダとは異なる場所にあるフォントを参照する場合
PTL80_ICCPROFILE_PATH（※3）	カラープロファイル「sRGB2014.icc」「JapanColor2001Coated.icc」の配置フォルダパス	「PdfTk80.dll」とは異なる場所にカラープロファイルを配置する場合
PATH	「PdfTk80.dll」をはじめとしたモジュールファイルの配置パス	インストーラ実行時にダイアログで「環境変数に追加する」を指定しなかった場合

Windows 開発・実行環境では上記の環境変数が必要です。

- 「PATH」を除く環境変数に関してはインストーラにより値が設定されます。
- 「PATH」はインストーラ実行時にダイアログで指定した場合に値が設定されます。
- 各設定値などの詳細はライブラリ版マニュアルの『インストーラによりシステムに設定される内容』をご参照ください。

※1

[PTL80_LIC_PATH]に関してはライブラリ版マニュアルの『ライセンスファイルについて』をご参照ください。

※2

[PTL80_FONT_CONFIGFILE]に関してはライブラリ版マニュアルの『描画とフォント埋め込みに使用するフォントの参照先について』をご参照ください。

※3

[PTL80_ICCPROFILE_PATH]に関してはライブラリ版マニュアルの『カラープロファイルの扱い』をご参照ください。

3.C++のビルドと実行手順

3.1.C++個別のサンプルコードのビルド方法

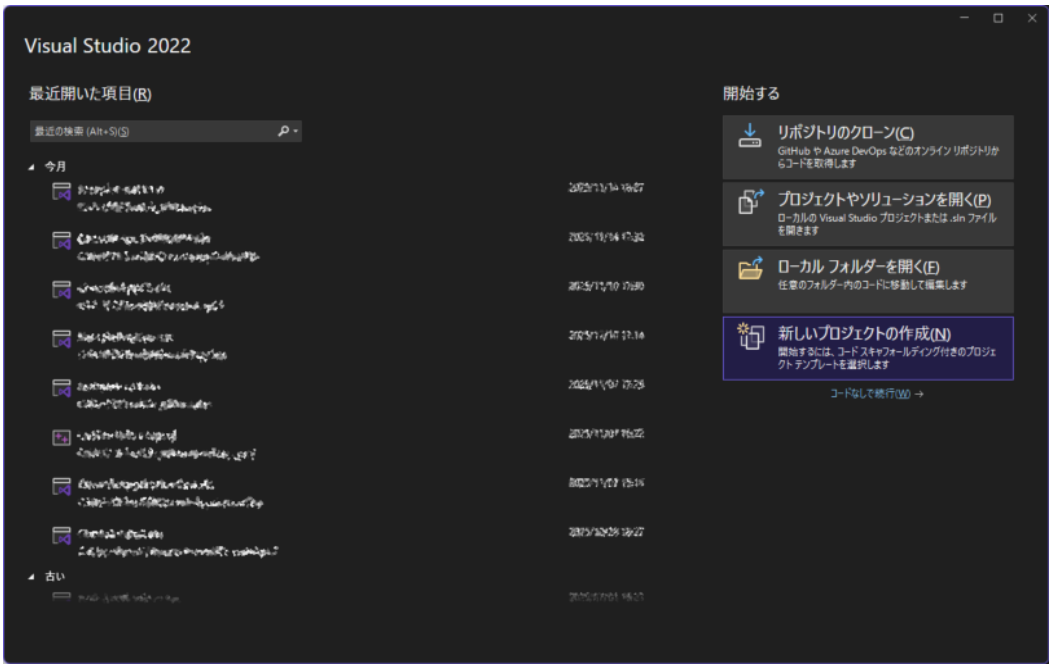
ここでは、C++で個別のサンプルコードをビルドする手順について説明します。

- ここで解説するのは Microsoft Visual Studio（Visual Studio）2022 を用いたビルド方法です。具体的にはプロジェクトの作成、ビルド、デバッグビルド、アプリケーションの発行を解説します。
- 本章の各実行例におけるパスは、PDF Tool API のインストール時にパス指定をしなかった場合の配置パスとなっています。具体的なパスは『PDF Tool API の開発環境を整える』の『2.1.1 インストーラによる配置』をご参照ください。
- 今回扱う C++のサンプルコードのソースファイルは、インストーラによって以下の場所に配置されています。

  {インストールフォルダ}¥samples¥cpp

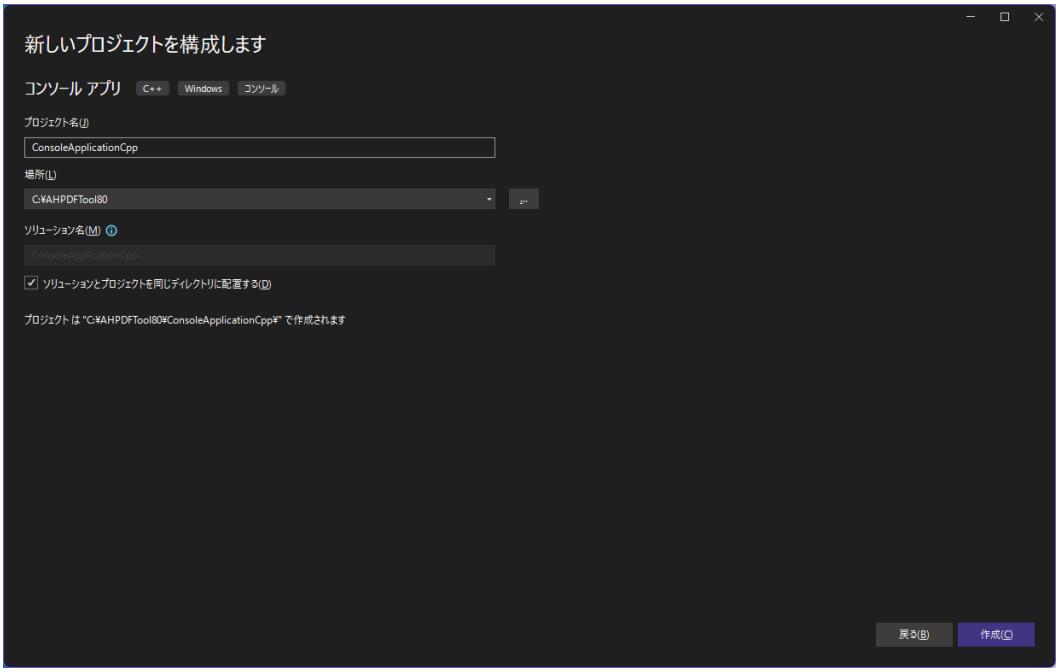
3.1.1. プロジェクトの新規作成と PDF Tool API モジュールファイルの参照追加

(1) Visual Studio 2022 を起動し、「新しいプロジェクトの作成」を選択します。

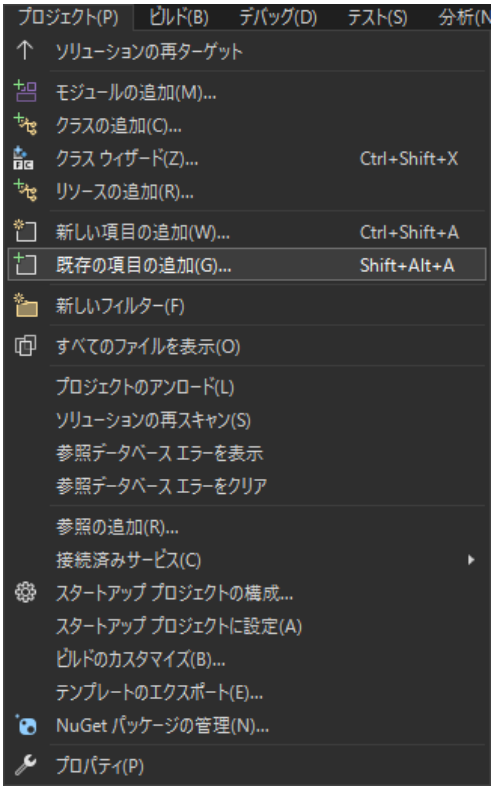


(2) 「新しいプロジェクトを作成」において、C++の「コンソールアプリ」を選択し「次へ」ボタンをクリックします。

(3) 「プロジェクト名」、「場所」を設定し「次へ」ボタンをクリックします。
これでプロジェクトが作成されます。



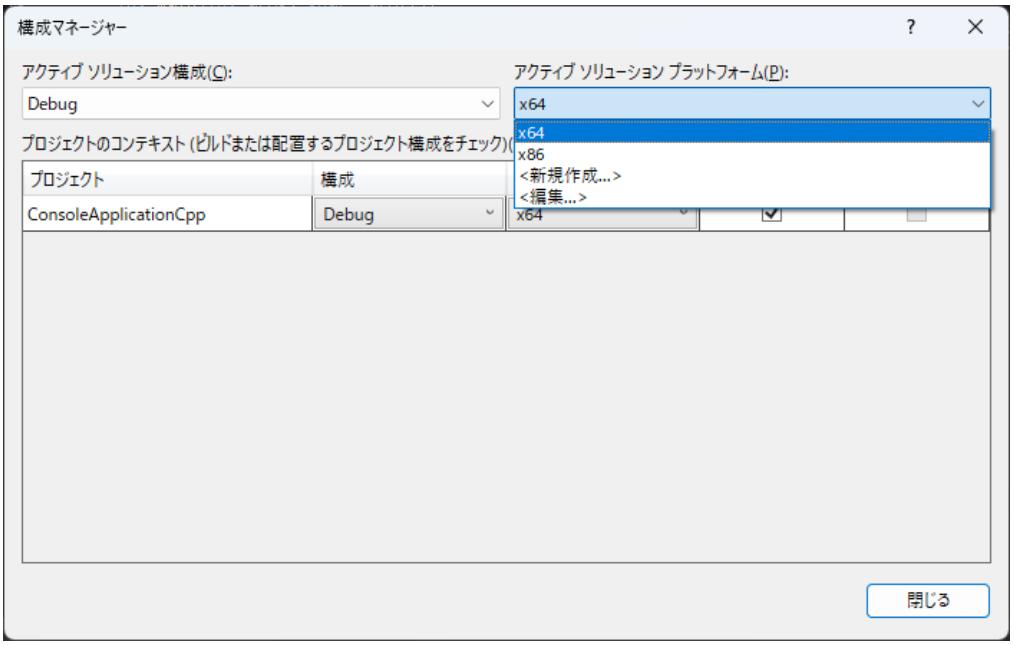
(4) 「プロジェクト」メニューの「既存の項目の追加...」を選択するとファイル選択ダイアログが表示されます。



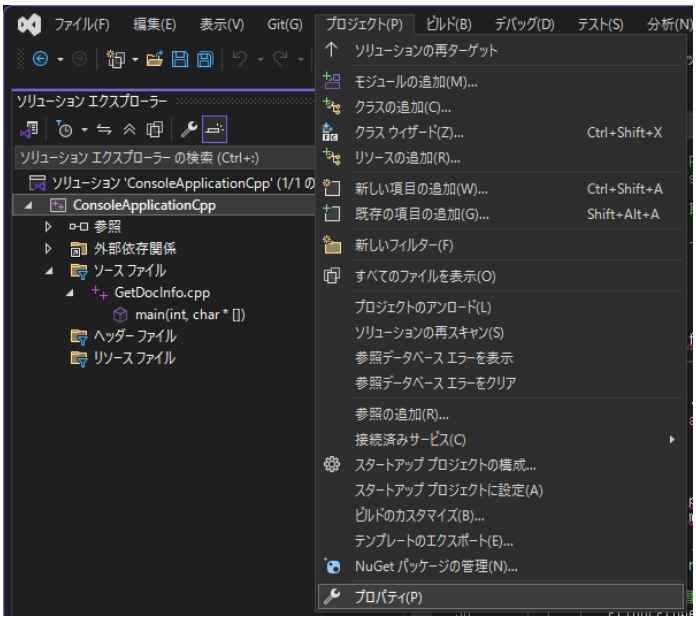
(5) サンプルの cpp ファイルをひとつ選択します。C++用サンプルコードはプロジェクトのソースファイルとして追加されます。
プロジェクト生成時に作成される cpp ファイルは不要です。プロジェクトから削除してください。

(6) 「ビルド」メニューの「構成マネージャー...」を選択します。

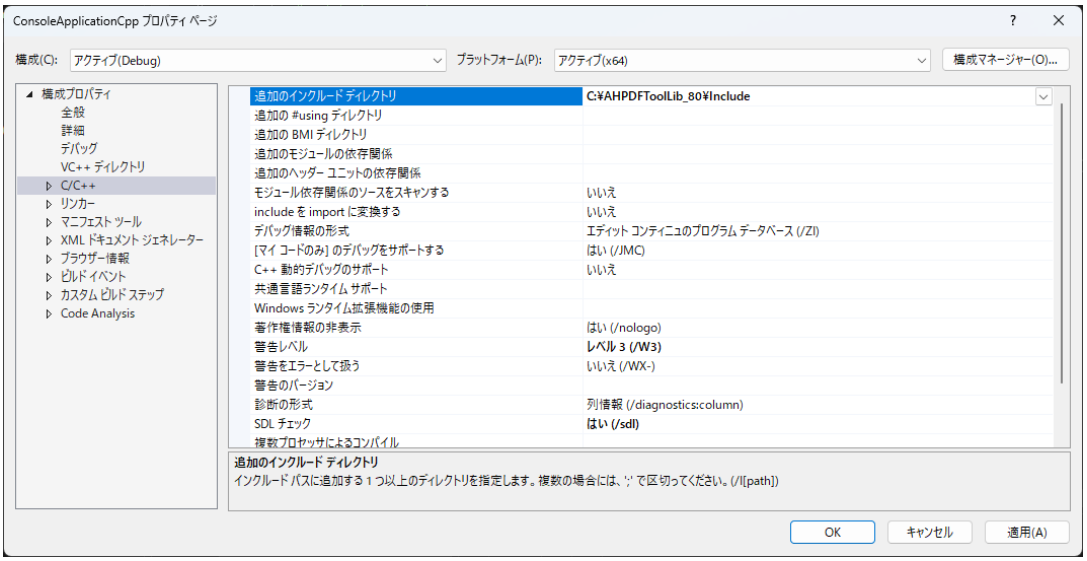
(7) 「アクティブソリューション構成」と「アクティブソリューションプラットフォーム」を設定します。プラットフォームは「x64」を選択します。



(8) ソリューション エクスプローラーでプロジェクトを選択した上で「プロジェクト」メニューを開き、「プロパティ」を選択します。(ソリューション・エクスプローラーの状態によっては、「(プロジェクト名) のプロパティ」の表記になっている場合もあります。)
これでプロジェクトのプロパティウィンドウが開きます。

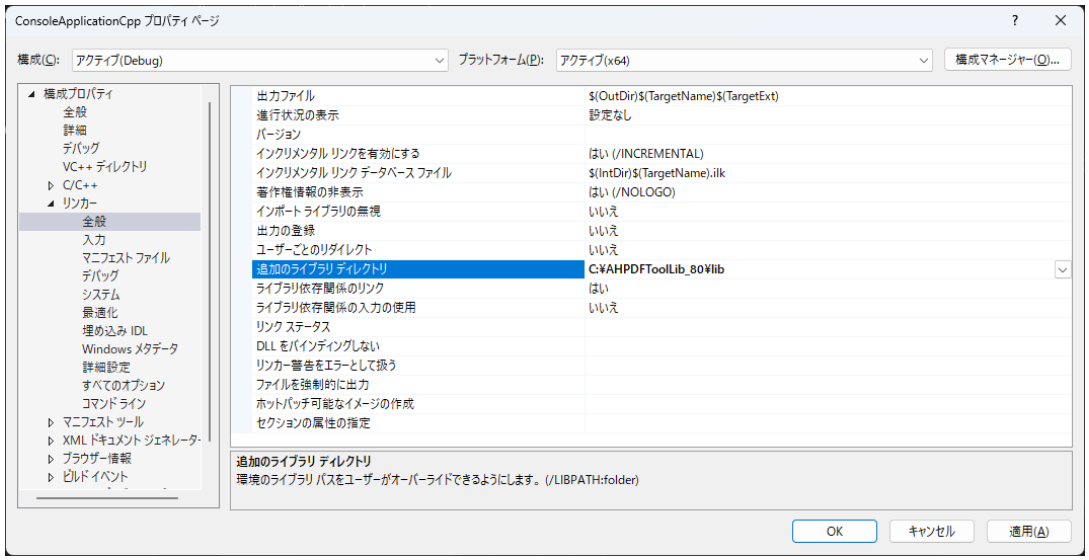


(9) 「構成プロパティ」の「C/C++」－「全般」タブにおいて、「追加のインクルードディレクトリ」を設定します。PDF Tool API の「PdfTk.h」ファイルがあるフォルダパスを指定します。

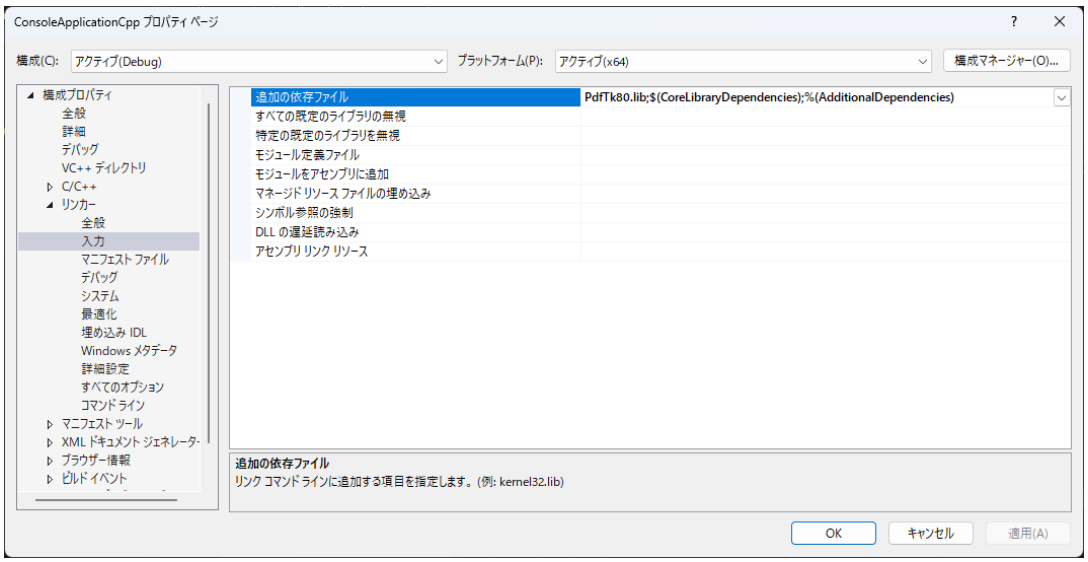


(10)「構成プロパティ」の「リンカー」 - 「全般」タブを開き、「追加のライブラリディレクトリ」を設定します。

PDF Tool API のライブラリファイル「PdfTk80.lib」のフォルダパスを指定します。



(11)「構成プロパティ」の「リンカー」 - 「入力」タブを開き、「追加の依存ファイル」に「PdfTk80.lib」を追加します。

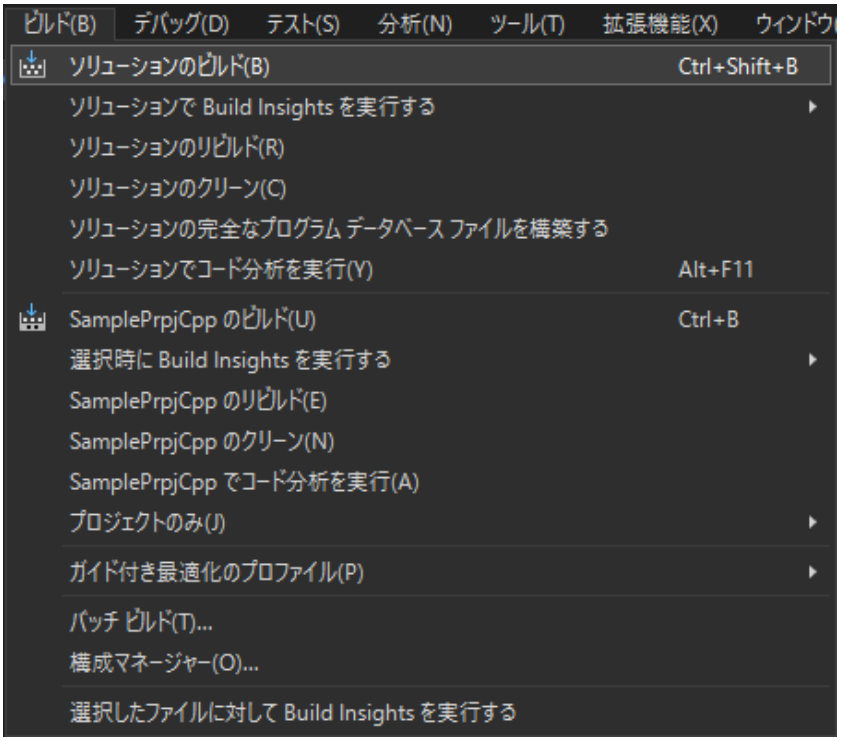


(12)プロパティウィンドウの「OK」ボタンを押して閉じます。

3.1.2. ビルドの設定とビルド

Windows 用のビルドを行います。

(1) 「ビルド」メニューの「(プロジェクト名) のビルド」をクリックすると、ビルドが開始されます。
(ソリューション内のプロジェクトが 1 つだった場合は「ソリューションのビルド」でもビルドを実行できます。)



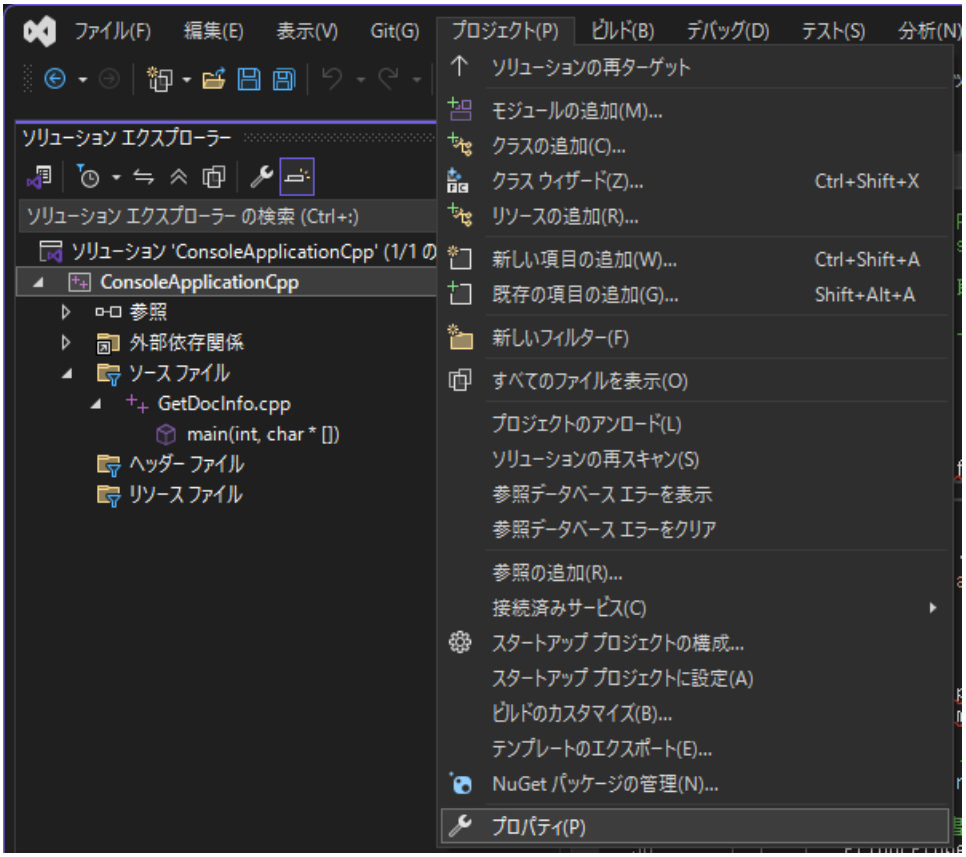
(2) ビルドが完了すると、出力ウィンドウに exe ファイルの出力先が表示されます。exe ファイルはダブルクリックやコンソールなどで実行可能です。
なお、実行時は、実行環境にモジュールファイルやライセンスファイルが必要です。

- 必要なモジュールファイルに関する詳細はライブラリ版マニュアルの『モジュールファイルについて』をご参照ください。
- PDF Tool API のインストーラを実行していた場合、インストーラにより各種ライセンスファイルが設定されています。それ以外の場合は設定が必要です。
  詳細は『2.4 Windows 開発・実行環境における環境変数のまとめ』をご参照ください。
- PDF Tool API のインストーラを実行した際にダイアログで指定していた場合、環境変数「PATH」にモジュールファイルへのパスが指定されています。それ以外の場合は設定が必要です。
  詳細は『2.4 Windows 開発・実行環境における環境変数のまとめ』をご参照ください。

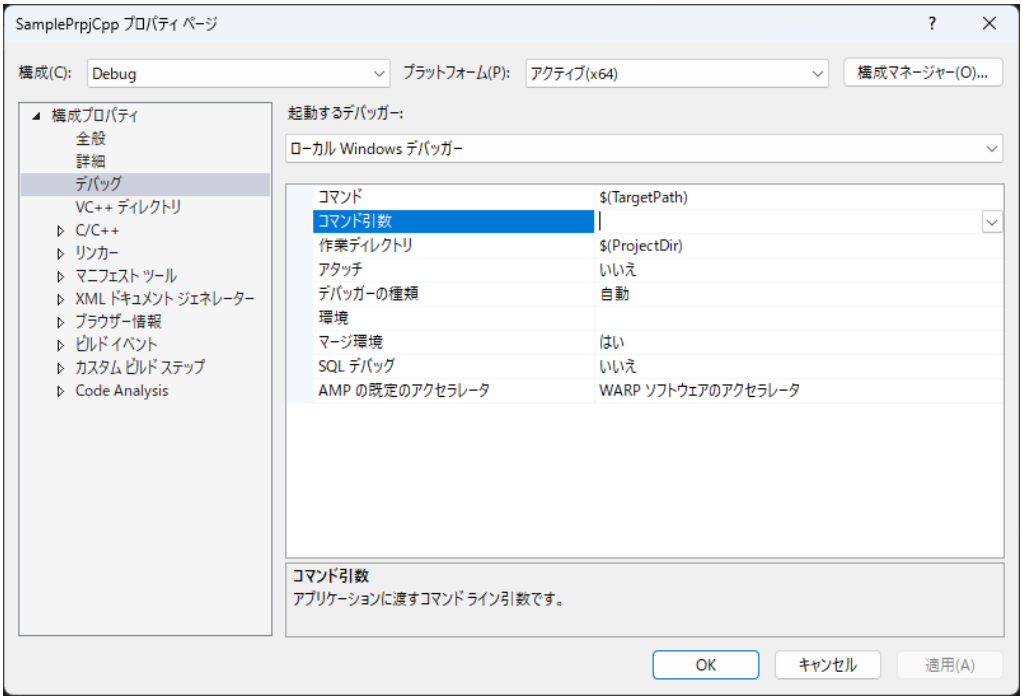
3.1.3. デバッグビルドの設定とデバッグ実行

Windows 用のデバッグビルドを行います。

(1) 「プロジェクト」メニューの「プロパティ」を選択し、プロパティウィンドウを開きます。
「プロパティ」を開く操作の詳細に関してはこちらをご参照ください。

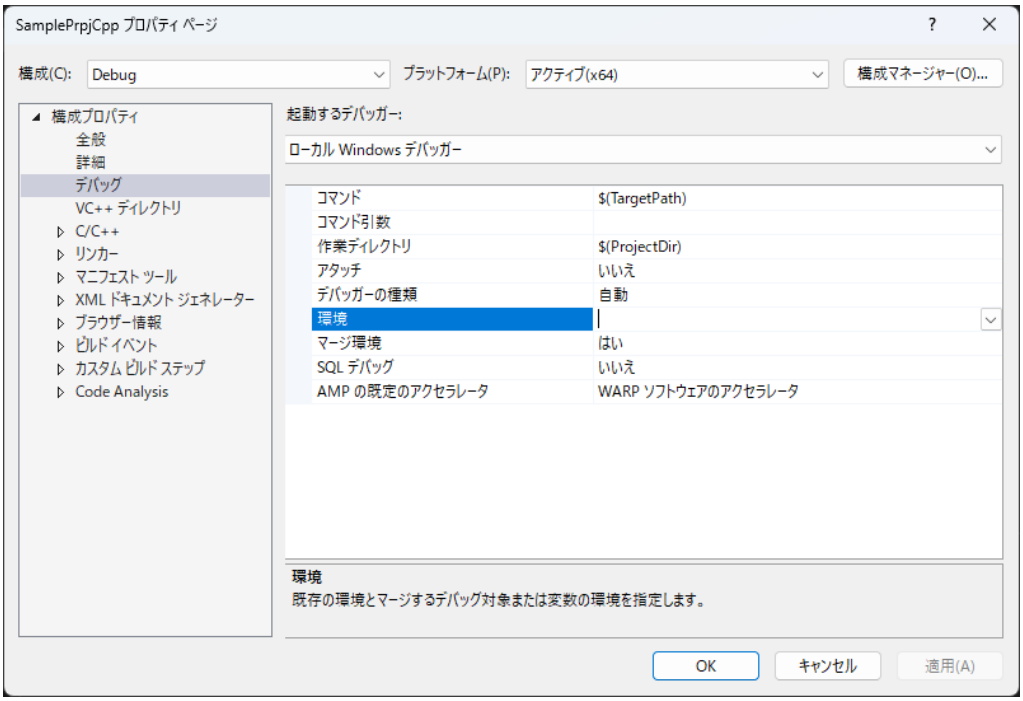


(2) 指定したい引数がある場合は、「構成プロパティ」の「デバッグ」タブにおいて「コマンド引数」に引数を指定します。

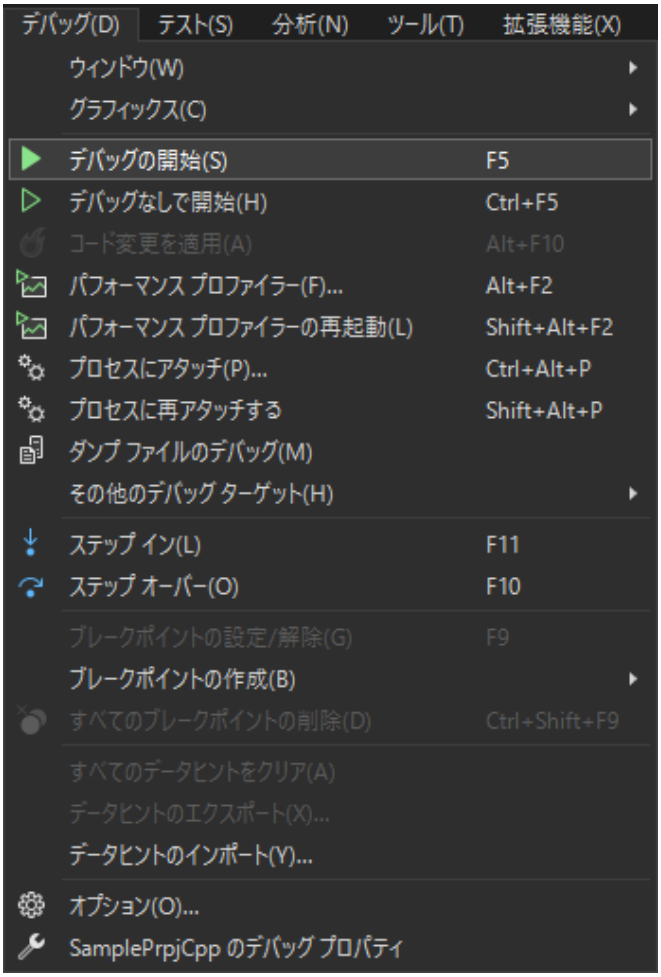


(3) 指定したい環境変数がある場合は「デバッグ」タブにおいて、「環境」に環境変数を記入します。

たとえば、モジュールファイルの配置パスなど、「PATH」に指定する内容を設定できます。



(4) 「デバッグ」メニューの「デバッグの開始」をクリックすることでデバッグが実行されます。



3.2.C++サンプルコード実行ソリューションの利用方法

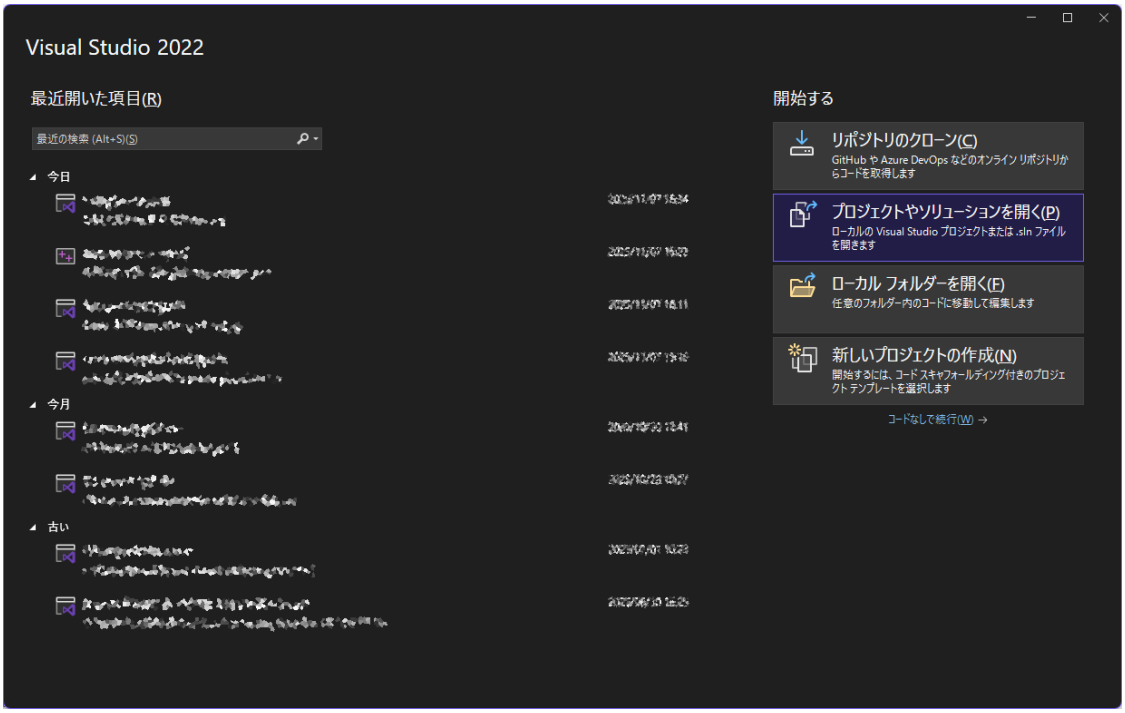
PDF Tool API のサンプルには、C++の各種サンプルコードをビルドするためのソリューションが同梱されています。本項ではその利用法を説明します。

- ソリューションファイルを用いることで、以下の項目が設定済みの状態で用意されたサンプルコードのビルドを行うことができます。
  - ソースコードの読み込み
  - 「追加のインクルードディレクトリ」の設定
  - 「追加のライブラリディレクトリ」の設定
  - 「追加の依存ファイル」の設定
- 本ソリューションを実行するためには Visual Studio 2022 が必要です。
- 本ソリューションファイルは以下の場所に配置されています。

  {インストールフォルダ}¥samples¥samples-cpp.sln

3.2.1. C++サンプルコード実行用ソリューションの読み込み

(1) Visual Studio 2022 を起動し、「プロジェクトやソリューションを開く」を選択します。

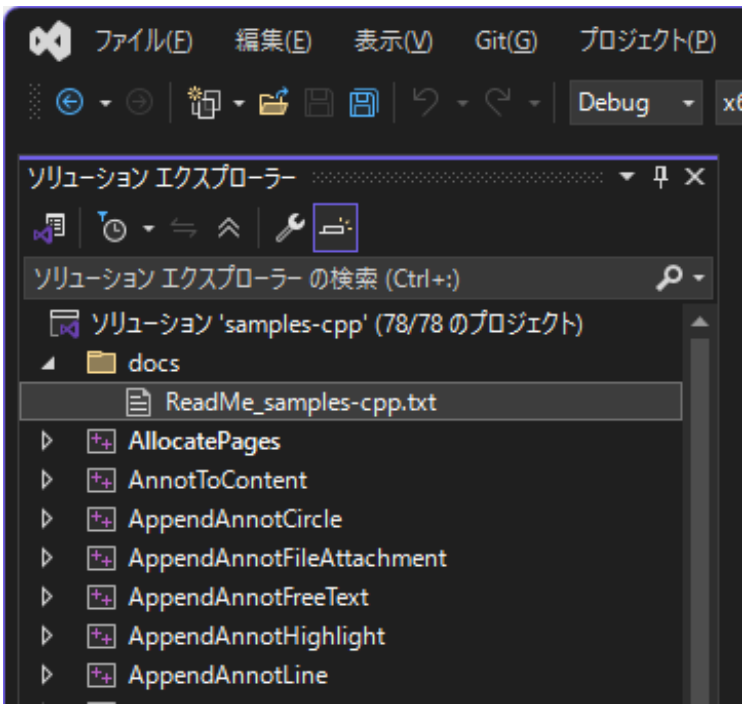


(2) 配置フォルダにある「samples-cpp.sln」を開きます。

(3) 以下の手順でサンプルコード実行の説明書を開くことができます。

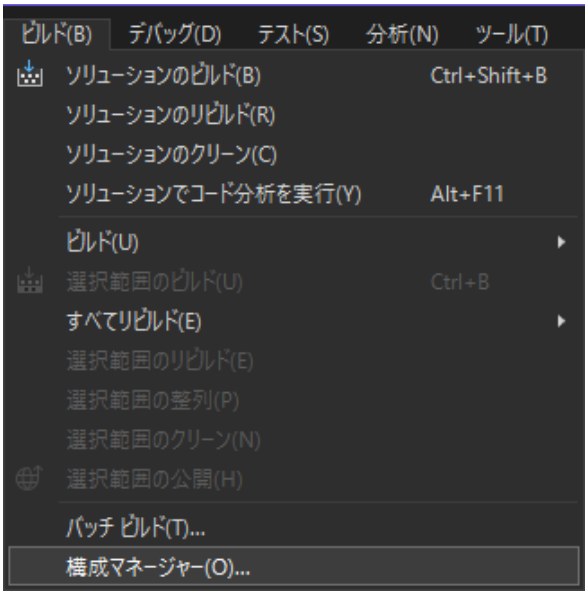
ソリューション エクスプローラーから doc フォルダを選択し、内部にある「ReadMe_samples-cpp.txt」を開きます。

各サンプルの実行内容や実行に必要なモジュールファイルに関する詳細は「ReadMe_samples-cpp.txt」をご参照ください。

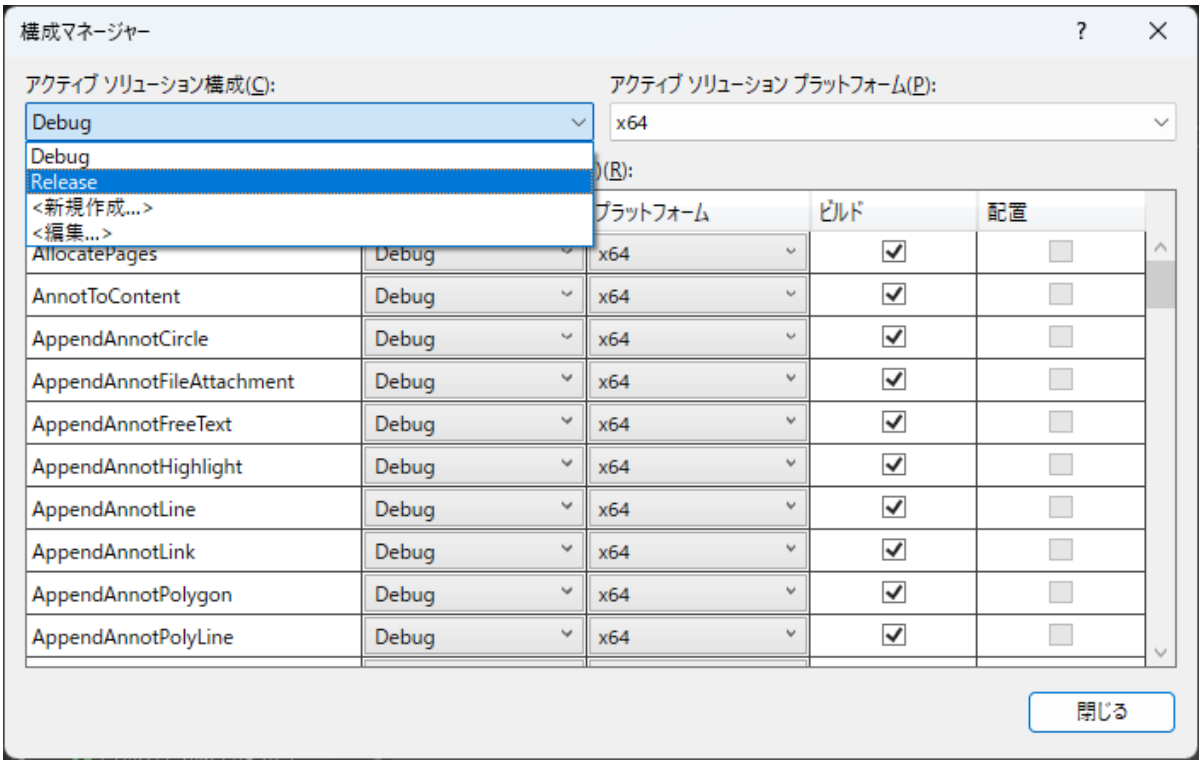


3.2.2. ビルドの設定とビルド

(1) 「ビルド」メニューの「構成マネージャー...」を選択します。



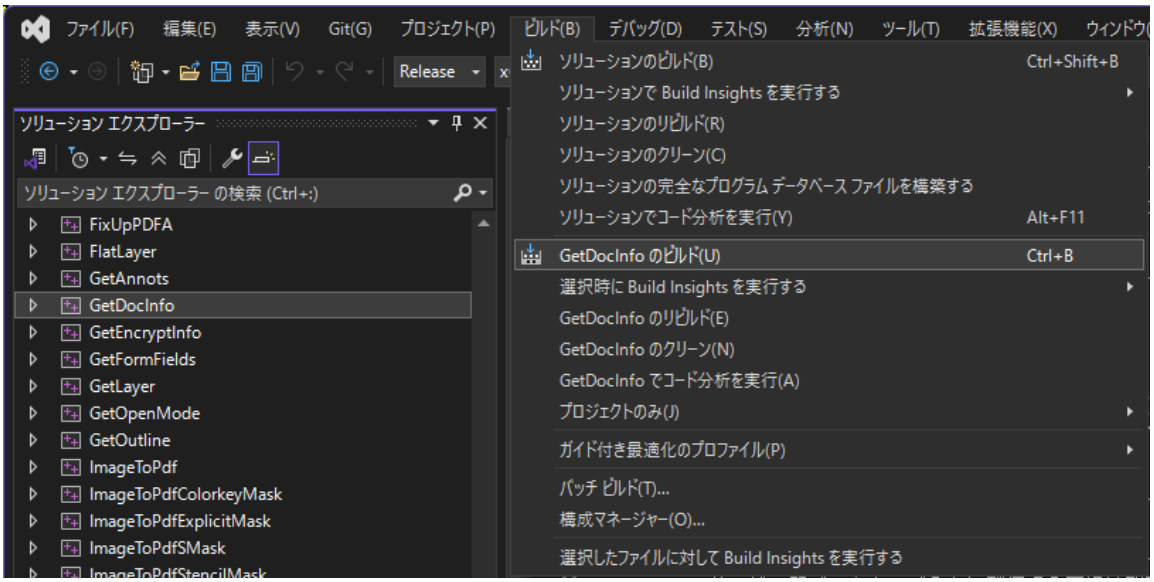
(2) 「アクティブソリューション構成」は「Release」を、プラットフォームは「x64」を選択して「OK」ボタンを押して構成マネジャー画面を閉じます。



(3) 実行したいサンプルのプロジェクトを選択し、ビルドします。

1. ソリューション エクスプローラーから実行したいサンプルのプロジェクトを選択します。
2. 「ビルド」メニューを開きます。
3. 「(プロジェクト名)のビルド」をクリックすると、ビルドが開始されます。

（実行したいプロジェクトを右クリックして「ビルド」を選択することでも実行できます）



(4) コンソールウィンドウの出力メッセージに、ビルドされた exe ファイルの出力パスが記載されています。

4..NET のビルドと実行手順

4.1..NET 8(C#)サンプルコードのビルドと発行方法

ここでは.NET 8 を用いて C#サンプルコードをビルド、実行する手順について説明します。

- ここで解説するのは Microsoft Visual Studio（Visual Studio）2022 を用いたビルド方法です。具体的にはプロジェクトの作成、ビルド、デバッグビルド、アプリケーションの発行を解説します。
- 本章の各実行例におけるパスは、PDF Tool API のインストール時にパス指定をしなかった場合の配置パスとなっています。具体的なパスは『PDF Tool API の開発環境を整える』の『2.1.1 インストーラによる配置』をご参照ください。
- .NET で扱う C#のサンプルコードのソースファイルは、それぞれ、以下の場所に配置されています。

  {インストールフォルダ}¥samples¥dotnet

4.1.1. . NET SDK のインストール

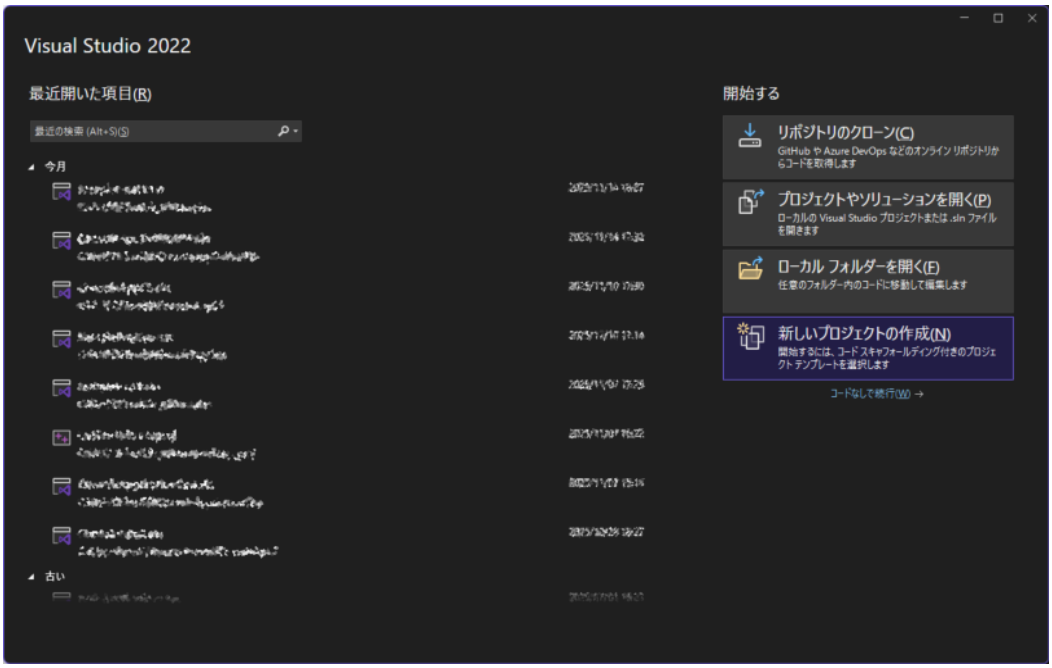
.NET 8 の開発には.NET SDK のインストールが必要です。
以下の web ページにアクセスし、「SDK 8.0.xx」の項にある「Windows」の「x64」インストーラーをダウンロードしてセットアップします。

.NET8.0 のダウンロード
https://dotnet.microsoft.com/ja-jp/download/dotnet/8.0

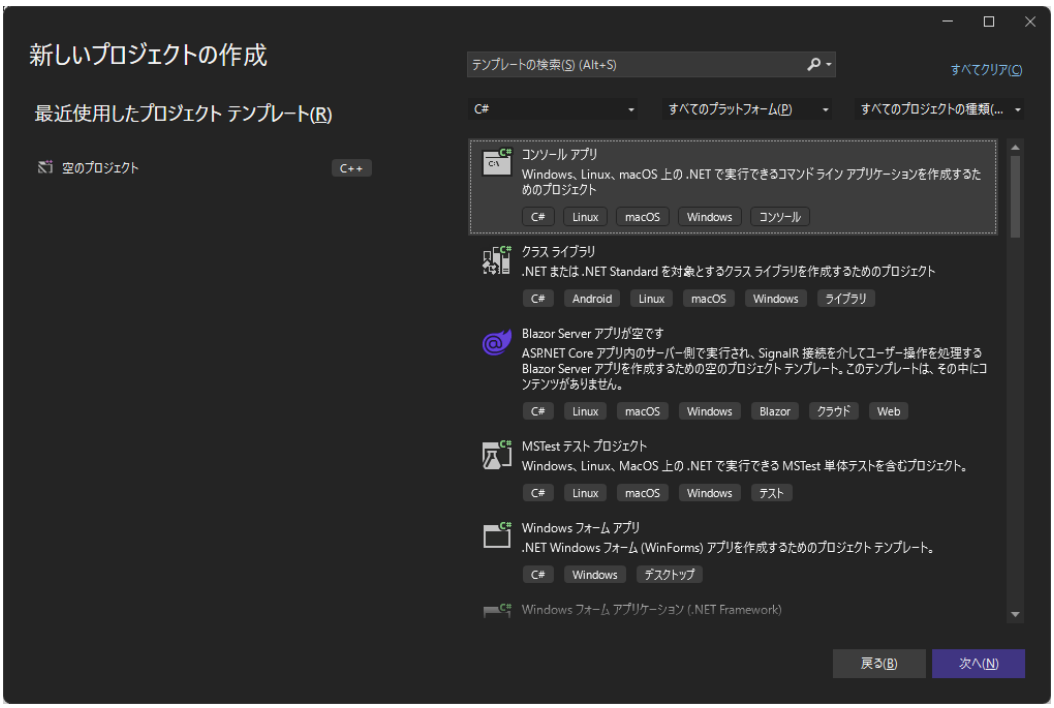
.NET 8.0 のダウンロード

4.1.2. プロジェクトの新規作成と PDF Tool API モジュールファイルの参照追加

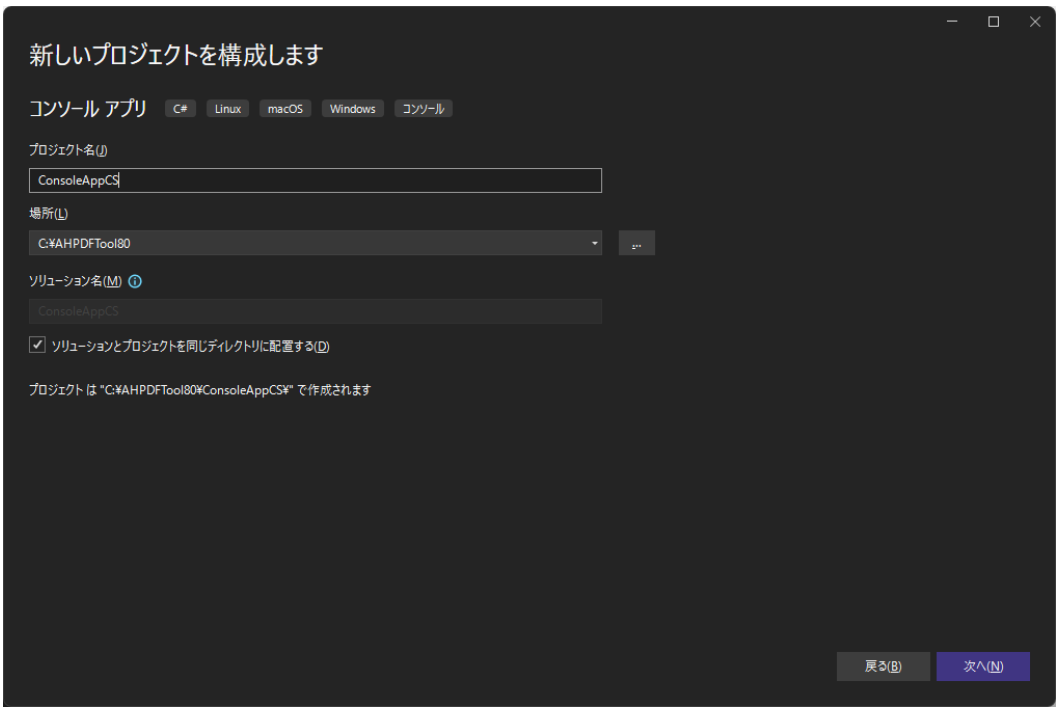
(1) Visual Studio 2022 を起動し、「新しいプロジェクトの作成」を選択します。



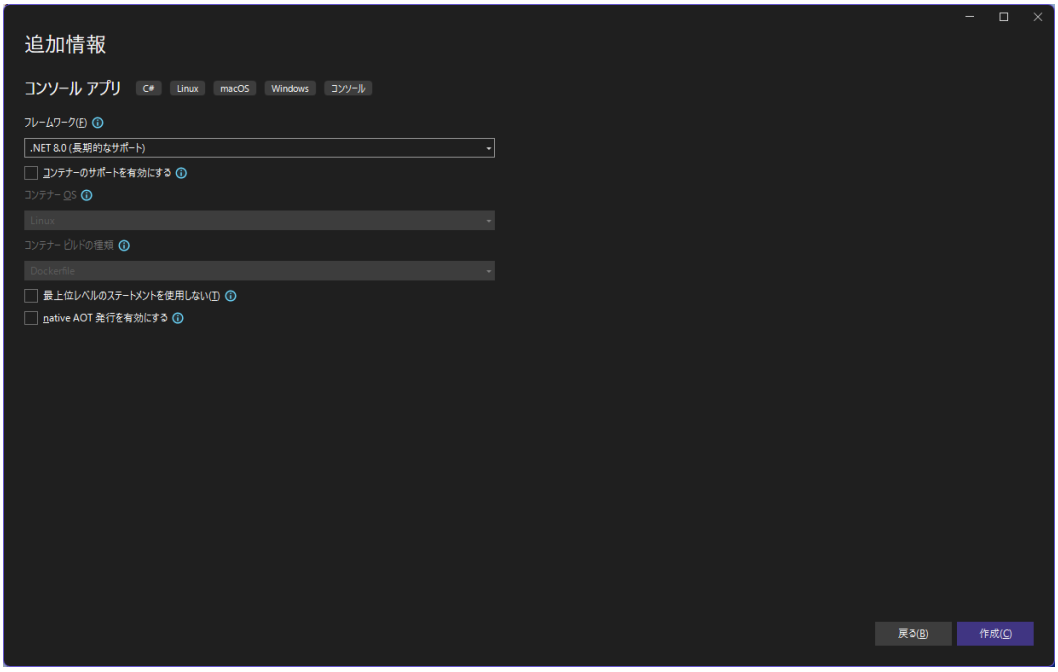
(2) 「新しいプロジェクトを作成」において、C#の「コンソールアプリ」を選択し「次へ」ボタンをクリックします。



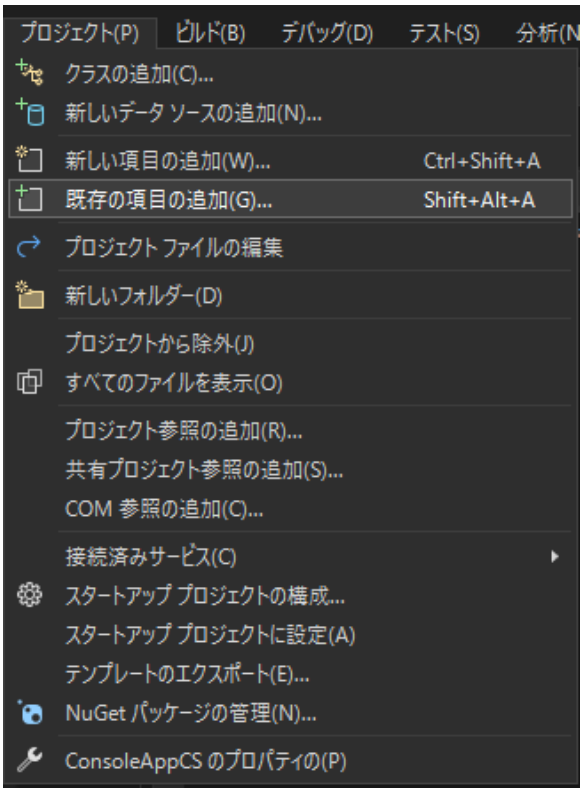
(3) 「プロジェクト名」、「場所」を設定し「次へ」ボタンをクリックします。



(4) 「フレームワーク」に「.NET 8.0（長期的なサポート)」が選択されているのを確認し「作成」ボタンをクリックするとプロジェクトが開きます。

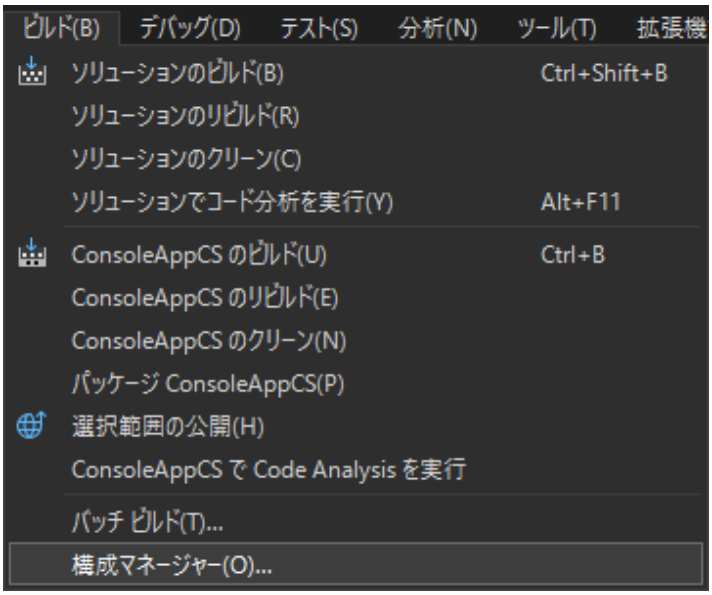


(5) 「プロジェクト」メニューの「既存の項目の追加...」を選択するとファイル選択ダイアログが表示されます。

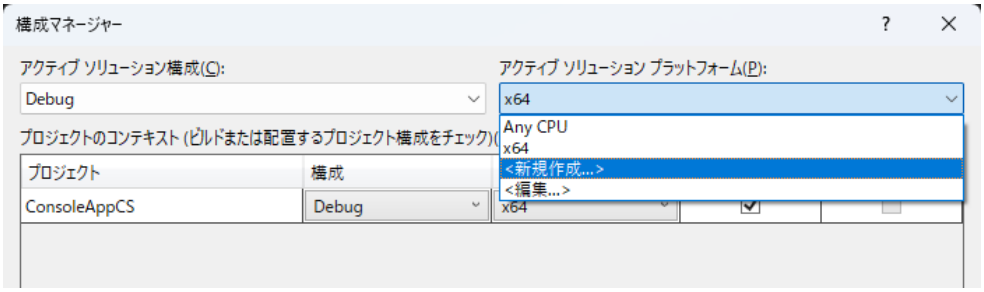


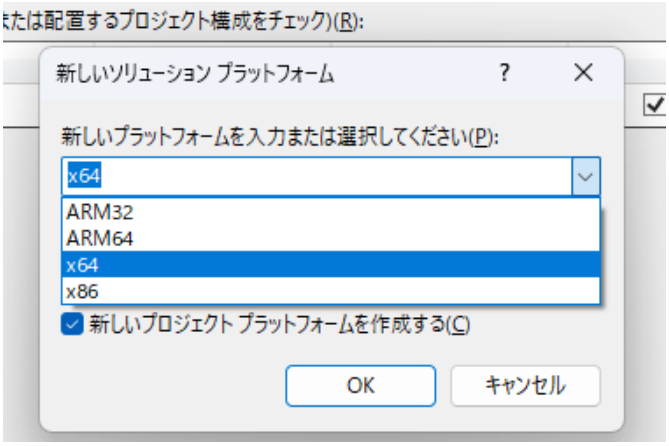
(6) サンプルの cs ファイルをひとつ選択します。C#用サンプルコードはプロジェクトのソースファイルとして追加されます。
プロジェクト生成時に作成される cs ファイルは不要です。プロジェクトから削除してください。

(7) 「ビルド」メニューの「構成マネージャー...」を選択します。

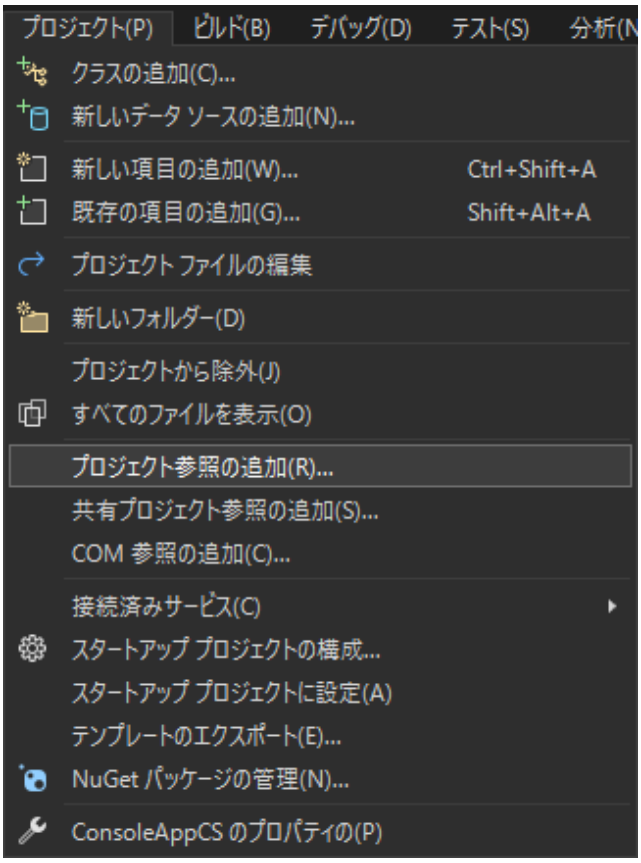


(8) 「アクティブソリューション構成」と「アクティブソリューションプラットフォーム」を設定します。プラットフォームは「x64」を選択して新規作成します。

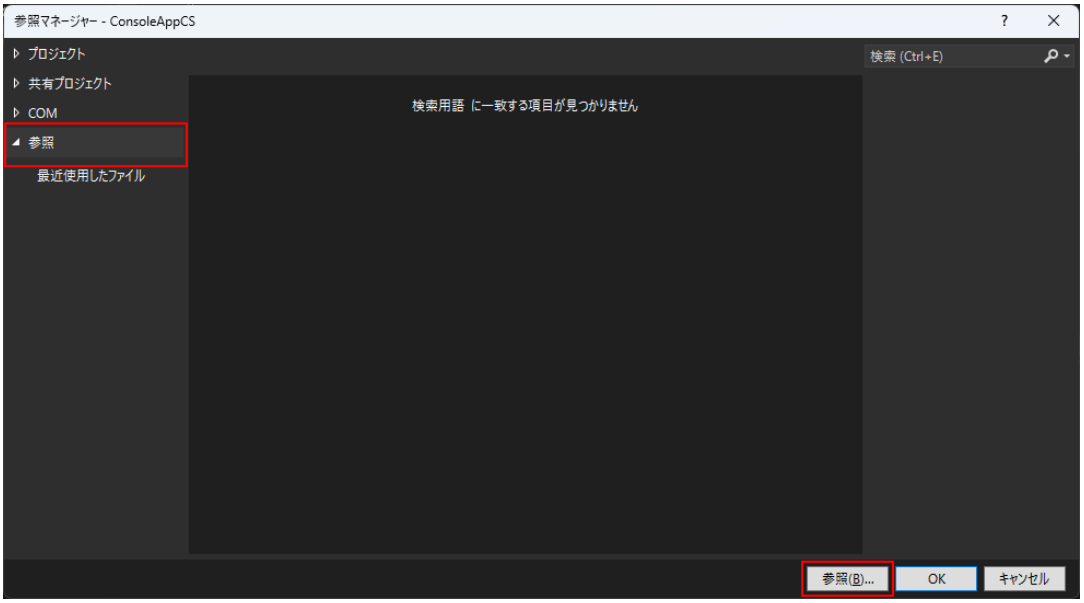




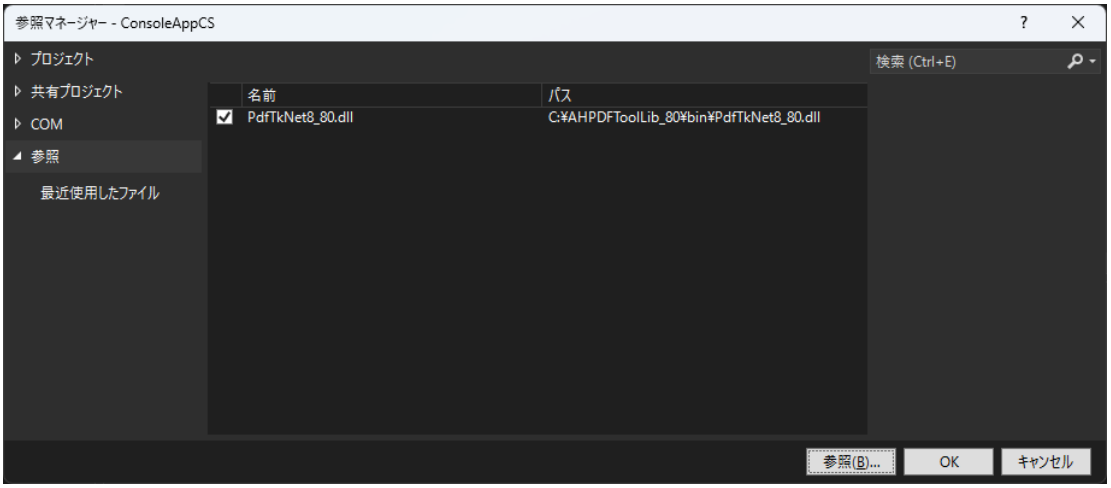
(9) 「プロジェクト」メニューの「プロジェクト参照の追加...」を選択すると、「参照マネージャー」ダイアログが開きます。



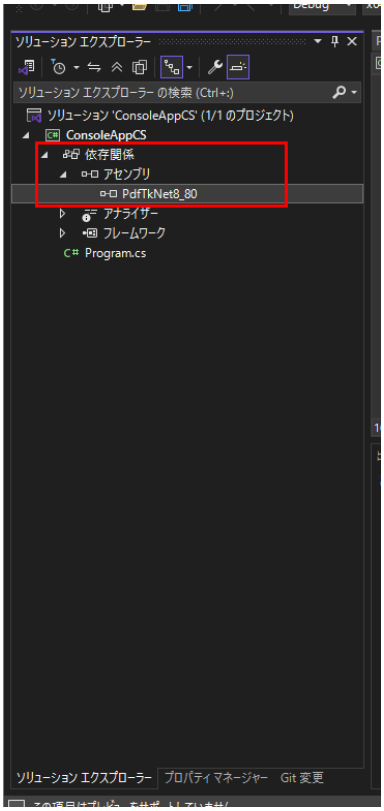
(10)「参照」タブを開き、ダイアログ右下の「参照...」ボタンをクリックします。



(11)ファイル選択ダイアログが開きます。インストールフォルダまたは任意の場所に配置した PDF Tool API モジュールファイルの「PdfTkNet8_80.dll」を選択します。
「OK」ボタンをクリックして「参照マネージャー」ダイアログを閉じます。



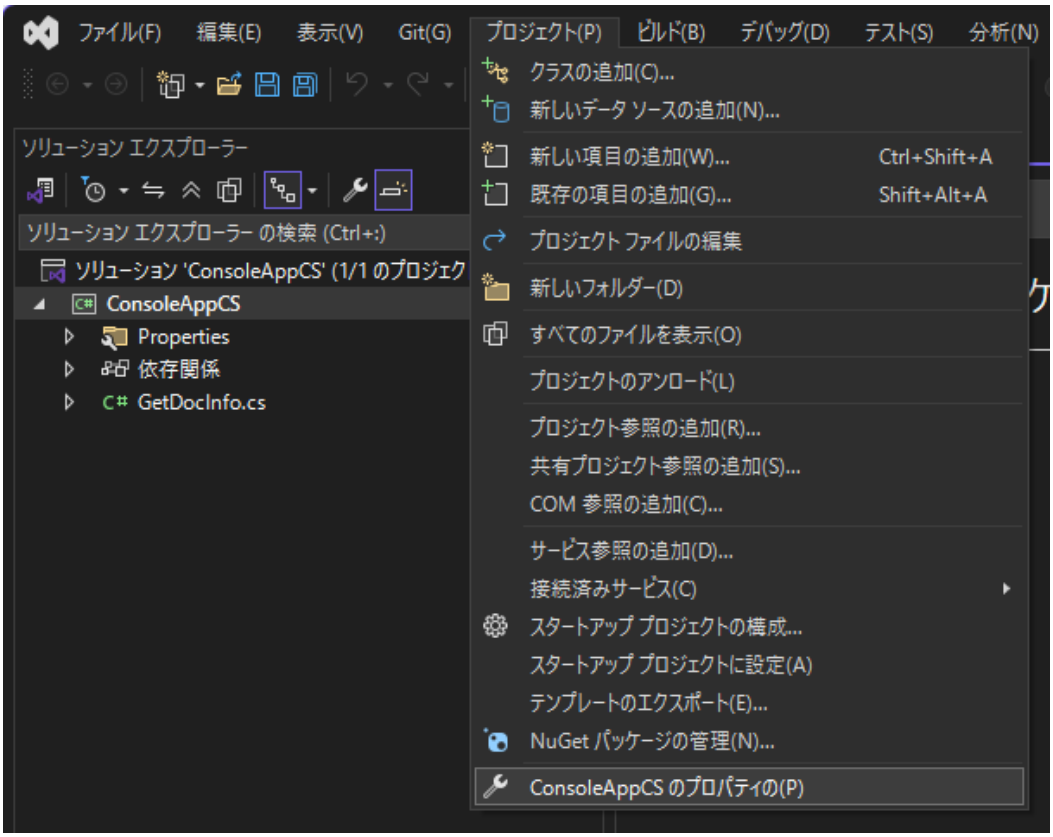
(12)参照した DLL は、ソリューションエクスプローラーの「依存関係」 - 「アセンブリ」で確認できます。



4.1.3. ビルドの設定とビルド

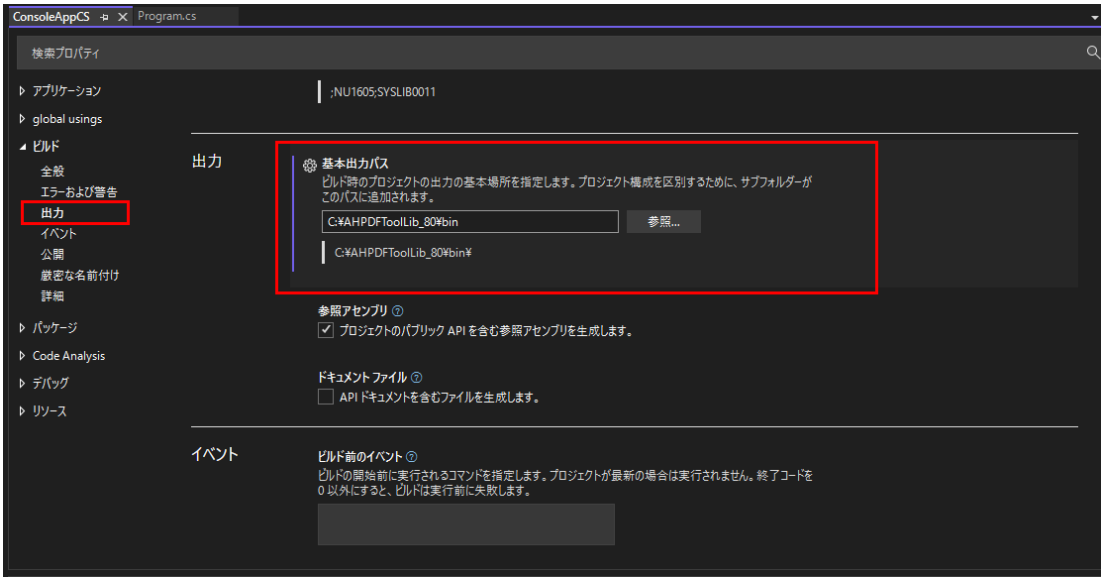
Windows 用のビルドを行います。

(1) ソリューション エクスプローラーでプロジェクトを選択した上で「プロジェクト」メニューを開き、「(プロジェクト名) のプロパティ」を選択します。



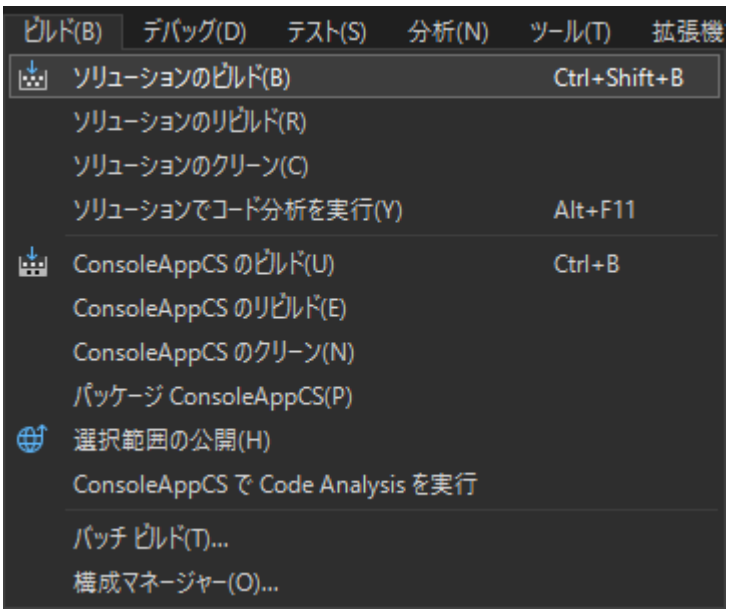
(2) プロパティのウィンドウが開きます。左側の「ビルド」タブを開き「出力」を選択します。

(3) 「基本出力パス」の欄に PdfTkNet8_80.dll、PdfTkNet8.dll を含む PDF Tool API の dll が存在するフォルダを指定します。



(4) 「ビルド」メニューの「(プロジェクト名) のビルド」をクリックすると、ビルドが開始されます。

(ソリューション内のプロジェクトが 1 つだった場合は「ソリューションのビルド」でもビルドを実行できます。)

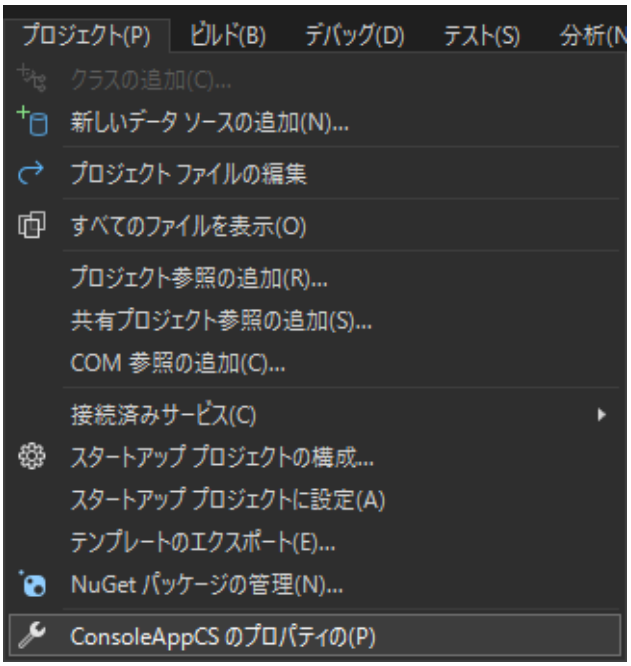


(5) ビルドされた exe を実行するには、exe とともに作成される、「exe と同名の dll ファイル」と「.runtimeconfig.json」が必要です。

4.1.4. デバッグビルドの設定とデバッグ実行

Windows 用のデバッグビルドを行います。

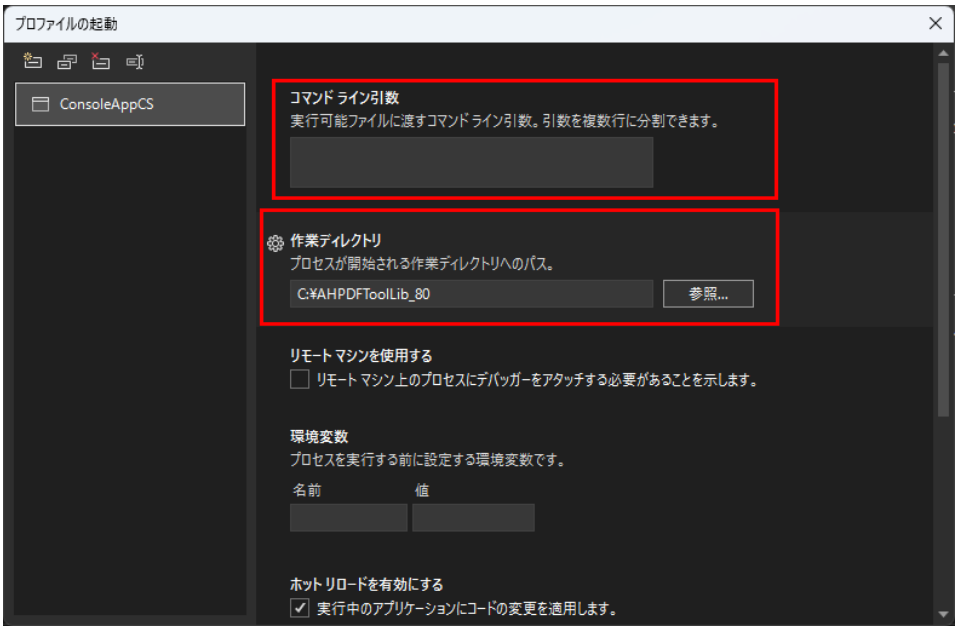
(1) 「プロジェクト」メニューの「(プロジェクト名) のプロパティ」を選択します。
(図のプロジェクト名は ConsoleAppCS です。)



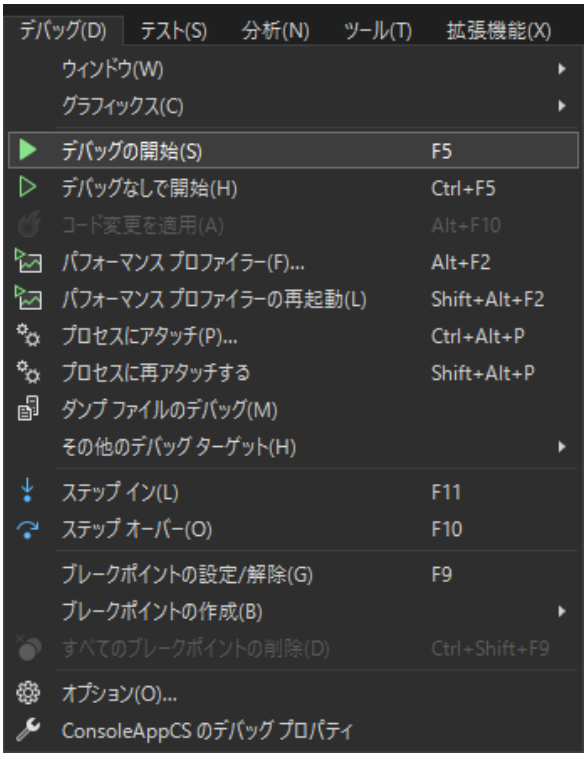
(2) 左側の「デバッグ」タグを開き「デバッグ起動プロファイル UI を開く」をクリックして「プロファイルの起動」ダイアログを開きます。

(3) 「プロファイルの起動」ダイアログの必要事項を指定します。

- 「作業ディレクトリ」の欄に PdfTkNet8_80.dll、PdfTkNet8.dll を含む PDF Tool API の dll のあるフォルダを指定します。
- 必要に応じて、「コマンドライン引数」を指定します。



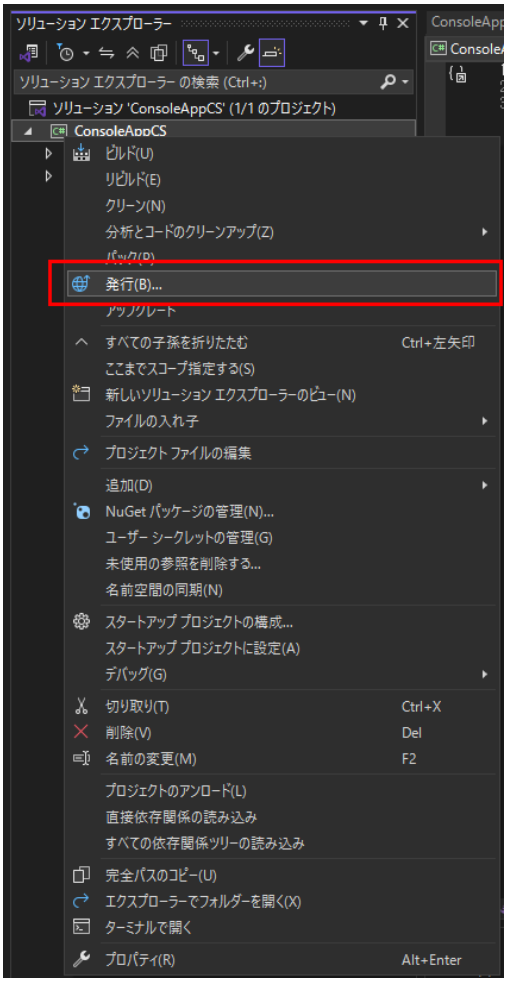
(4) 「デバッグ」メニューの「デバッグの開始」をクリックすると、デバッグ実行できます。


デバッグ(D)
テスト(S)
分析(N)
ツール(T)
拡張機能(X)
ウィンドウ(W)
グラフィックス(C)
デバッグの開始(S) F5
デバッグなしで開始(H) Ctrl+F5
コード変更を適用(A) Alt+F10
パフォーマンス プロファイラー(F)... Alt+F2
パフォーマンス プロファイラーの再起動(L) Shift+Alt+F2
プロセスにアタッチ(P)... Ctrl+Alt+P
プロセスに再アタッチする Shift+Alt+P
ダンプ ファイルのデバッグ(M)
その他のデバッグ ターゲット(H)
ステップ イン(L) F11
ステップ オーバー(O) F10
ブレークポイントの設定/解除(G) F9
ブレークポイントの作成(B)
すべてのブレークポイントの削除(D) Ctrl+Shift+F9
オプション(O)...
ConsoleAppCS のデバッグ プロパティ

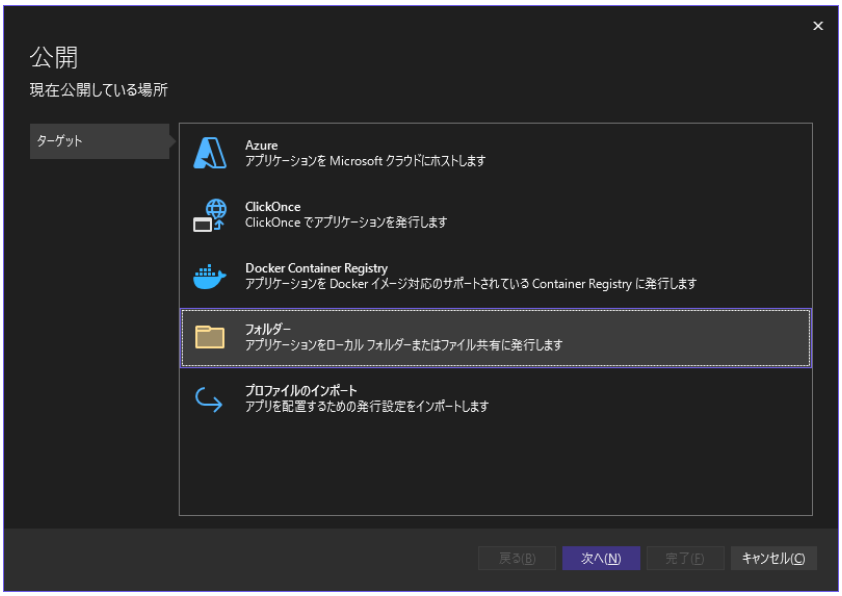
4.1.5. アプリケーションの発行

Windows 用、Linux 用それぞれのアプリケーションファイルを発行します。

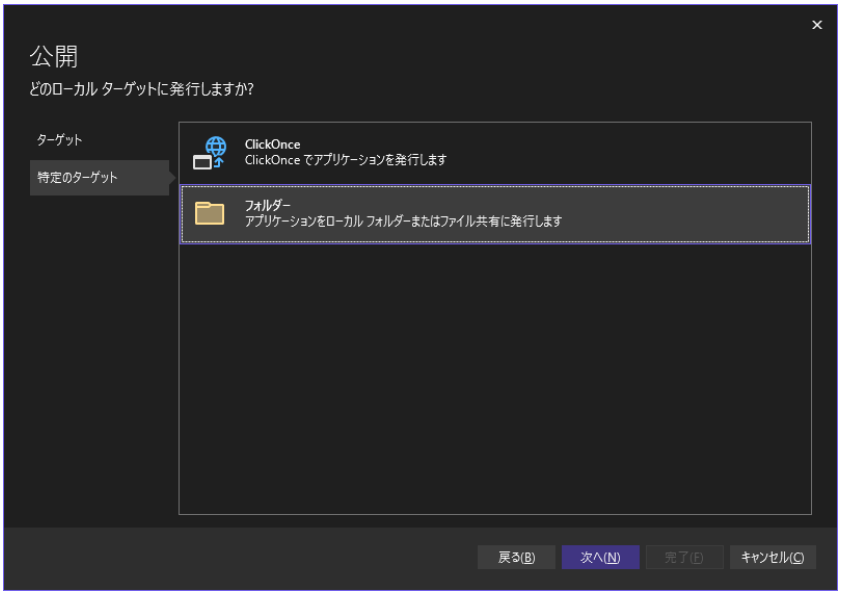
(1) プロジェクトを右クリックし、開いたメニュー上の「発行...」を選択します。



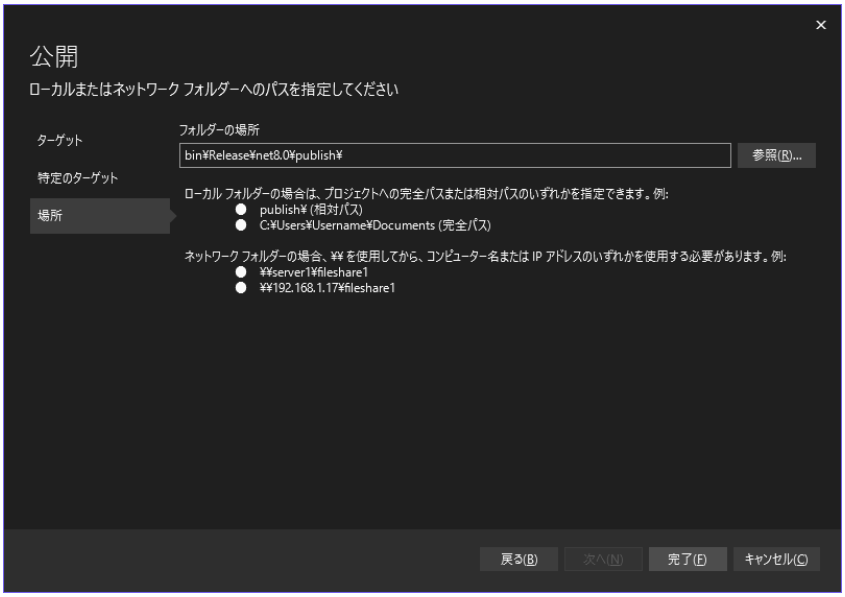
(2) 「公開」ダイアログが開くので公開先を指定します。
ここでは、ローカルフォルダーにアプリケーションを作成しますので、「フォルダー」を選択し、右下の「次へ」ボタンをクリックします。



(3) 次の画面でも「フォルダー」を選択し右下の「次へ」ボタンをクリックします。

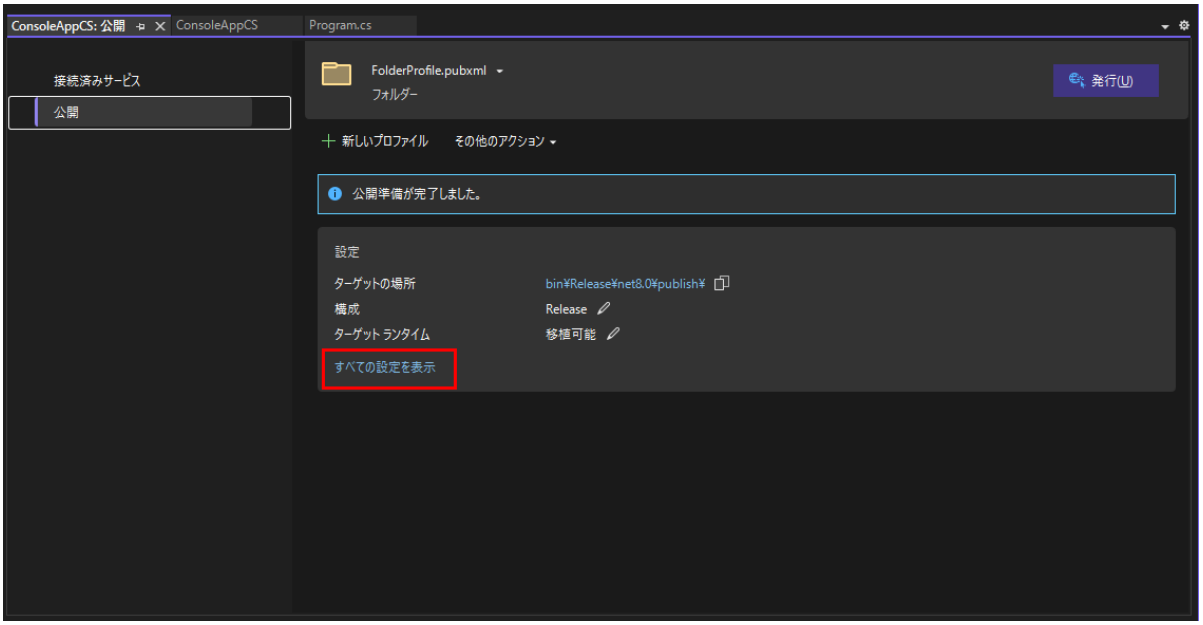


(4) 「フォルダーの場所」を指定し、「完了」を押します。



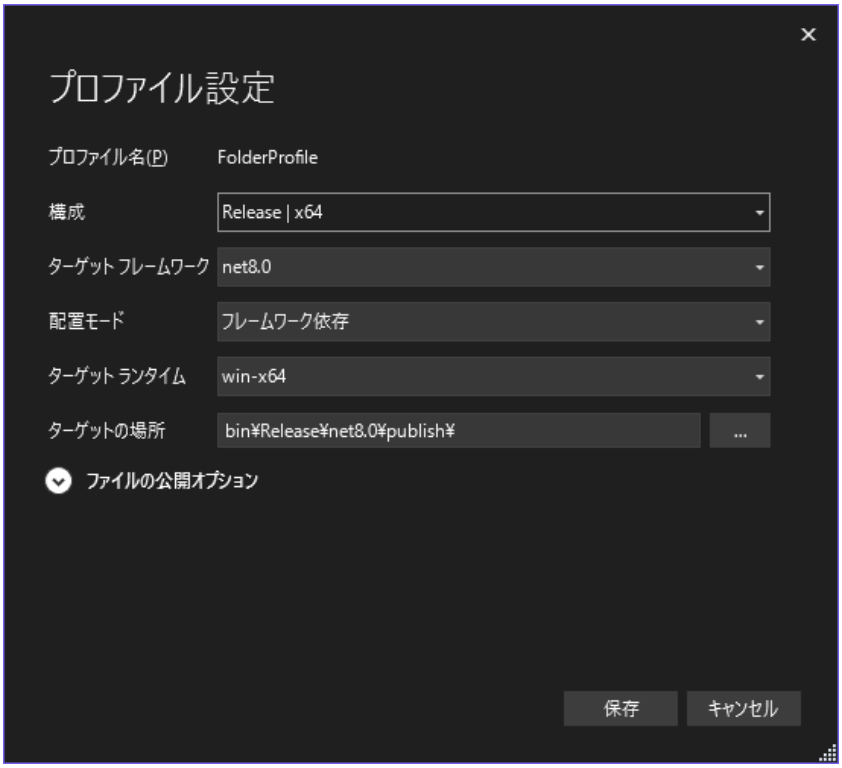
(5) 「実行プロファイル作成の進行状況」が表示された場合は、「閉じる」ボタンをクリックします。

(6) 中央下にある「すべての設定を表示」をクリックします。

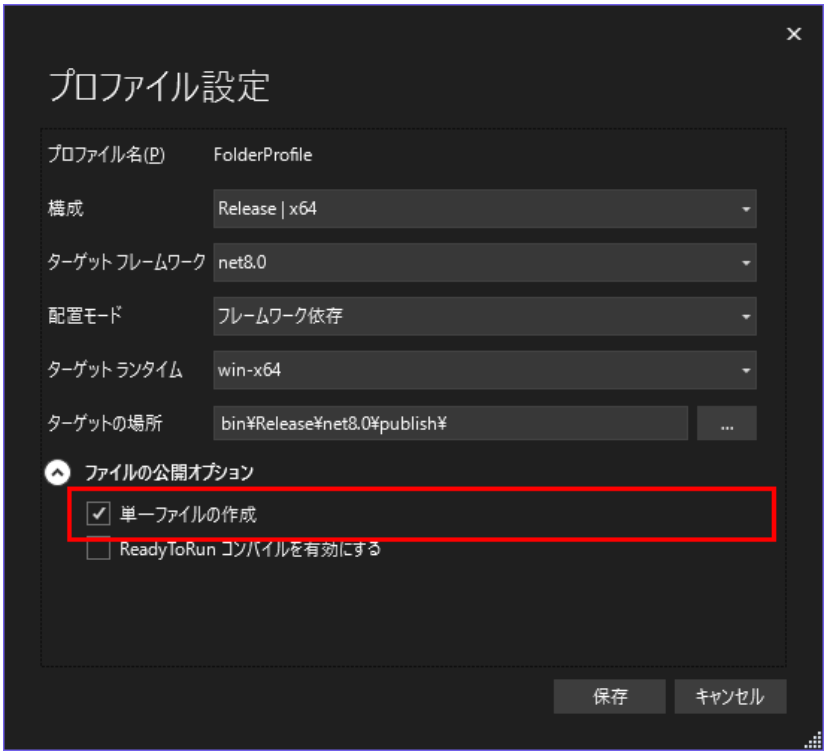


(7) 「プロファイル設定」ダイアログが開くので各項目を設定します。

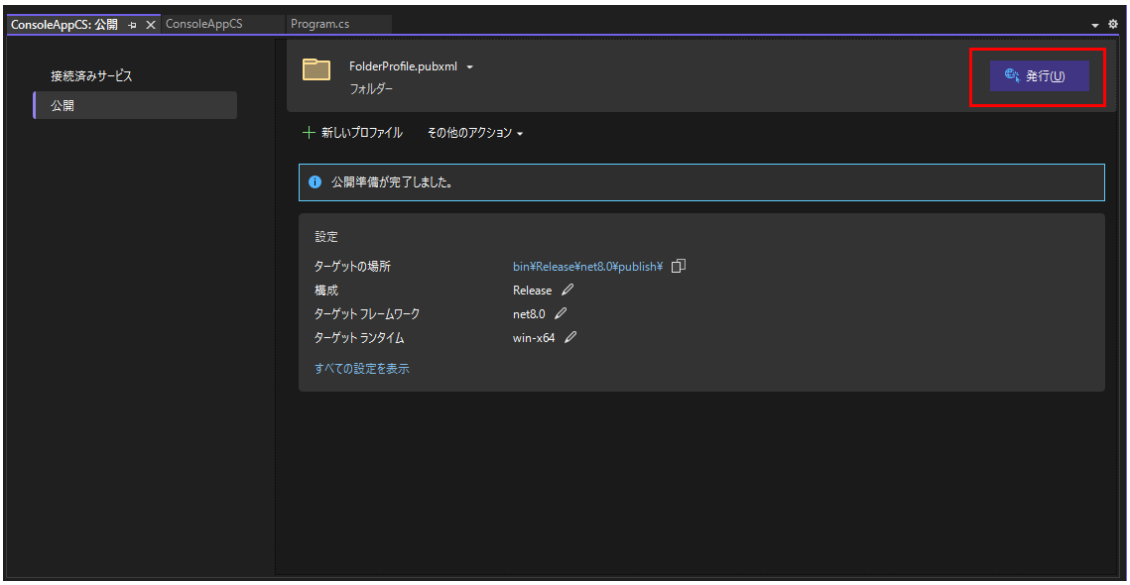
- ターゲットフレームワーク：「net8.0」であることを確認
- 配置モード：以下２つから選択　違いは以下の通りです。
  - 「自己完結」
    発行されるアプリケーションファイルには、ターゲットフレームワークのランタイムライブラリーが含まれます。このため、アプリケーションファイルを実行する環境に、ターゲットフレームワークの.NET ランタイムライブラリーをインストールする必要はありません。ただし、ファイルサイズは大きくなります。
  - 「フレームワーク依存」
    発行されるアプリケーションファイルには、ターゲットフレームワークのランタイムライブラリーは含まれません。アプリケーションファイルを実行する環境には、ターゲットフレームワークの.NET ランタイムライブラリーをインストールしてください。
- ターゲットランタイム：「win-x64　(Windows 64bit)」「linux-x64 (Linux 64bit)」のいずれかを選択
  PDF Tool API は、arm、osx（Mac OS X）には対応していません。



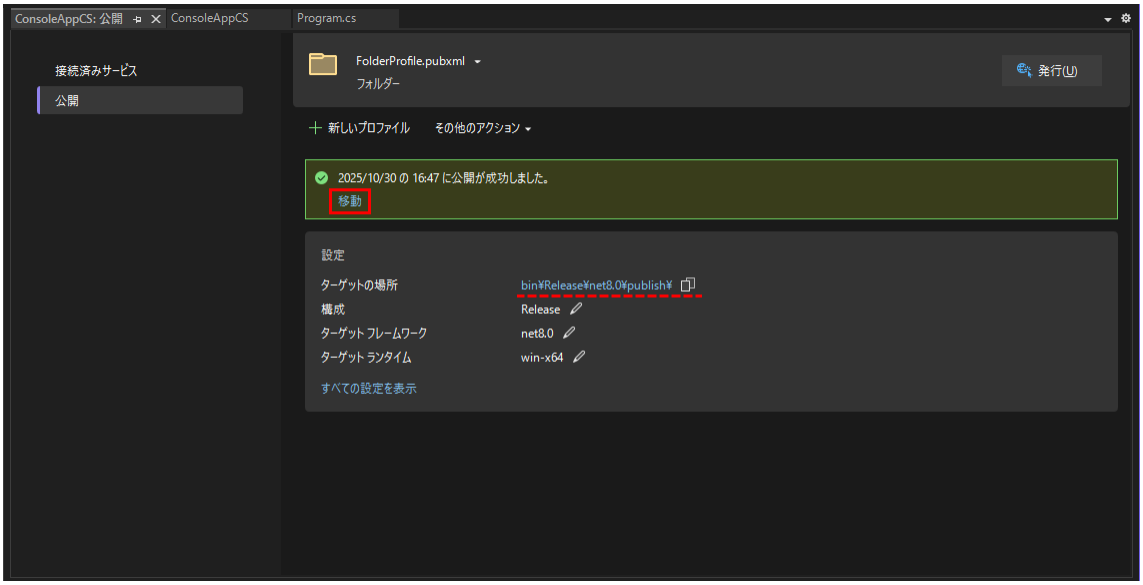
(8) 「ファイルの公開オプション」を開きます。「単一ファイルの作成」にチェックを入れます。



(9) 「発行」ボタンをクリックすると、「(日時) に公開が成功しました。」と表示され発行が完了します。



(10)「フォルダーを開く」または「ターゲットの場所」をクリックすると「フォルダーの場所」に指定したフォルダが開き、発行されたファイルを確認できます。



(11) ターゲットランタイムごとの各実行ファイルは以下の通りです。

- 「ターゲットランタイム」が「win-x64」の場合、発行された拡張子「.exe」ファイルが実行ファイルです。.exe ファイルと PDF Tool API のモジュールファイルを同じフォルダに配置することで実行可能です。
- 「ターゲットランタイム」が「linux-x64」の場合、発行された拡張子のないファイルが実行ファイルです。

4.1.6. .NET 8 で発行された実行ファイルの実行方法について

ここでは、『4.1 .NET 8(C#)サンプルコードのビルドと発行方法』で発行されたアプリケーションファイルを、開発環境とは異なるコンピュータ上で実行する方法を説明します。

Windows での実行について

(1) アプリケーションファイルの発行時に「配置モード」を「フレームワーク依存」で指定した場合、ターゲットフレームワークの.NET ランタイムライブラリーをインストールしてください。(※1)

(2) PDF Tool API のモジュールファイルを配置します。(※2)

(3) 発行されたアプリケーションファイルを、PDF Tool API のモジュールファイルと同じ場所に配置します。あるいは、環境変数「PATH」に、PDF Tool API のモジュールファイルが存在するフォルダパスを設定します。

(4) PDF Tool API のライセンスファイルを配置します。「PdfTk80.dll」と同じ場所に配置する、または、ライセンスファイルを配置したフォルダパスを環境変数「PTL80_LIC_PATH」に設定してください。(※3)

(5) 必要に応じ、フォント構築ファイルの設定を行ってください。(※4)

(6) 必要に応じ、カラープロファイル、フォント構築ファイルを利用するための環境変数を設定します。(※3)

(7) コマンドプロンプトを起動しアプリケーションファイルが存在するフォルダパスをカレントディレクトリにして実行します。必要に応じて、コマンド入力時に引数を指定してください。

（※1）

.NET8 ランタイム　インストーラの入手先

https://dotnet.microsoft.com/ja-jp/download/dotnet/8.0

「.NET Runtime 8.0.xx」の項にある「Windows」の「x64」インストーラーをダウンロードしてセットアップしてください。

なお、ランタイム単体のインストーラーではなく、ランタイムが含まれる SDK のインストーラーで代用することも可能です。

（※2）

実行環境にモジュールファイルを配置する場合、PDF Tool API 付属のインストーラは使用しないでください。

（※3）

『Windows 開発環境における環境変数のまとめ』をご参照くさい。

（※4）

『フォントの準備』をご参照ください。

Linux での実行について

(1) 「配置モード」が「フレームワーク依存」であるアプリケーションファイルの場合、あらかじめ、ターゲットフレームワークの.NET ランタイムライブラリーをインストールしてください。(※1)

(2) PDF Tool API のモジュールファイルをセットアップします。(※2)

(3) アプリケーションファイルを任意の場所に配置します。

(4) PDF Tool API のライセンスファイルを配置します。

(5) テキスト透かしやテキスト追加など文字を扱う処理を行う場合、「font-config.xml」(フォント構築ファイル) にてフォントディレクトリの設定を行います。
Linux では、フォント構築ファイルは必須です。

(6) モジュールファイル、ライセンスファイル、カラープロファイル、フォント構築ファイル、を利用するための環境変数を設定します。(※3)

(7) 「端末」を起動し、アプリケーションファイルがあるディレクトリをカレントにして実行します。必要に応じて、コマンド入力時に引数を指定してください。(※4a) (※4b)

（※1）

Linux 版サンプル実行手順の『.NET ランタイムのセットアップ』をご参照ください。

（※2）

実行環境にモジュールファイルをセットアップする場合、PDF Tool API 付属のインストーラは使用しないでください。

（※3）

環境変数と設定値の例

LD_LIBRARY_PATH={PDF Tool API モジュールファイルのディレクトリ}:${LD_LIBRARY_PATH}

PTL80_LIC_PATH={ライセンスファイルのディレクトリ}

PTL80_FONT_CONFIGFILE={ font-config.xml のフルパス}

PTL80_ICCPROFILE_PATH={「sRGB2014.icc」「JapanColor2001Coated.icc」の配置ディレクトリ}

（※4a）

「配置モード」が「自己完結」設定のアプリケーションファイルを実行する場合

「dotnet」コマンドは不要です。アプリケーションファイルに.NET8 ランタイムライブラリーが含まれています。

必要コマンド：

```
./{アプリケーションファイル名} {オプション}
```

（※4b）

「配置モード」が「フレームワーク」設定のアプリケーションファイルを実行する場合

「dotnet」コマンドを付けて実行します。

必要コマンド：

```
dotnet {アプリケーションファイル名} {オプション}
```

4.2..NET Framework(C#)サンプルコードのビルドと実行方法

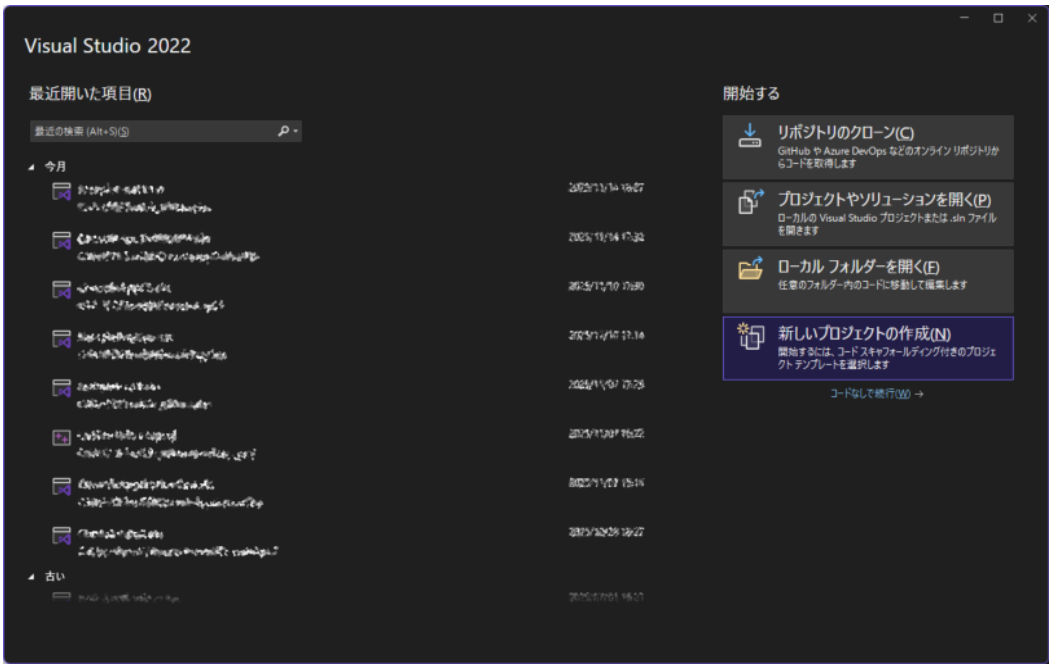
ここでは、.NET Framework を用いて C#サンプルコードをビルド、実行する手順について説明します。

- ここで解説するのは Microsoft Visual Studio（Visual Studio）2022 を用いたビルド方法です。具体的にはプロジェクトの作成、ビルド、デバッグビルド、アプリケーションの発行を解説します。
- 本章の各実行例におけるパスは、PDF Tool API のインストール時にパス指定をしなかった場合の配置パスとなっています。具体的なパスは『PDF Tool API の開発環境を整える』の『2.1.1 インストーラによる配置』をご参照ください。
- .NET Framework で用いる C#のサンプルコードのソースファイルは、以下の場所に配置されています。

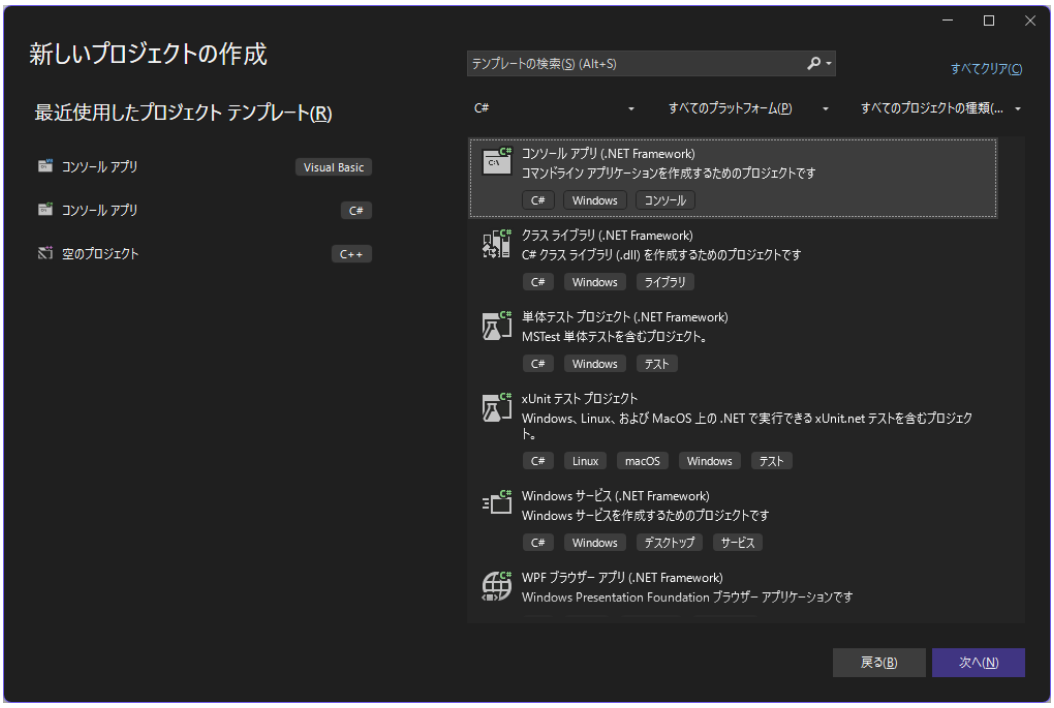
  {インストールフォルダ}¥samples¥dotnet

4.2.1. プロジェクトの新規作成と PDF Tool API モジュールファイルの参照追加

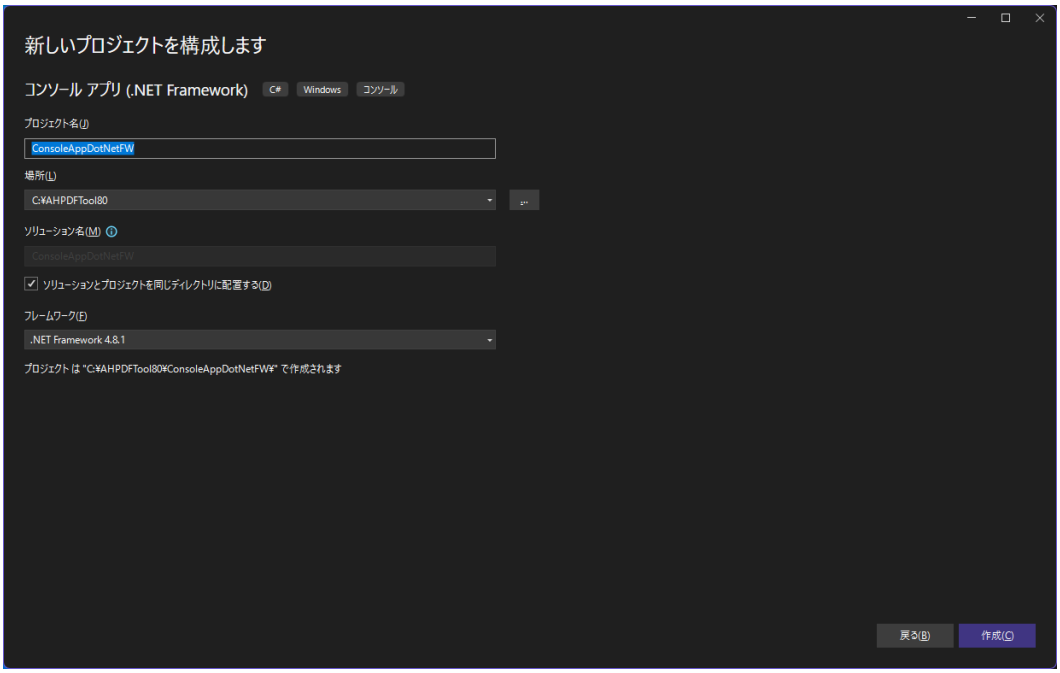
(1) Visual Studio 2022 を起動し、「新しいプロジェクトの作成」を選択します。



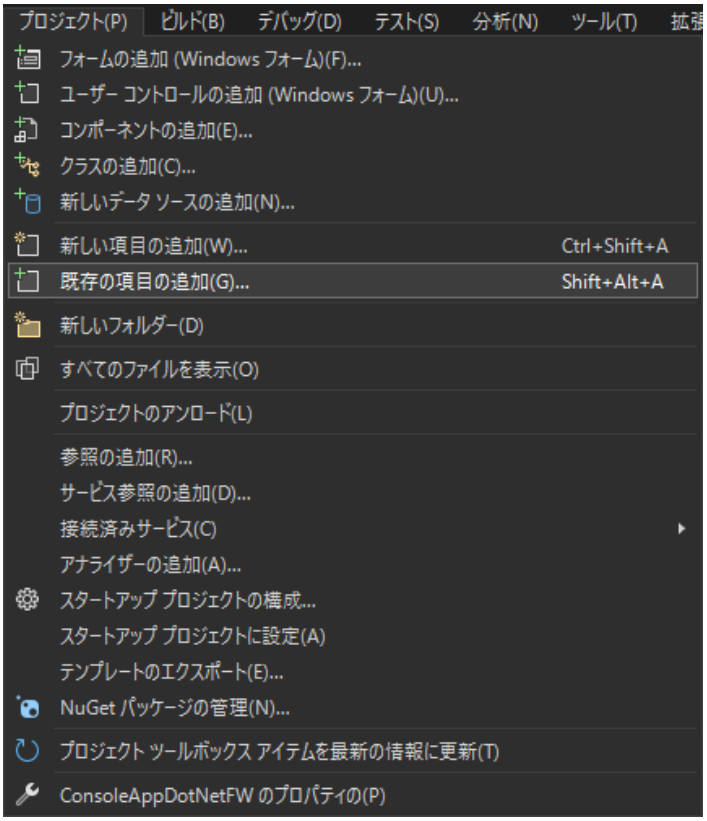
(2) 「新しいプロジェクトを作成」において、C#の「コンソールアプリ(.NET Framework)」を選択し「次へ」ボタンをクリックします。



(3) 「プロジェクト名」、「場所」を設定します。
さらに「フレームワーク」で.NET Framework 4.8 以降が選択されているのを確認します。
「作成」ボタンをクリックするとプロジェクトが開きます。

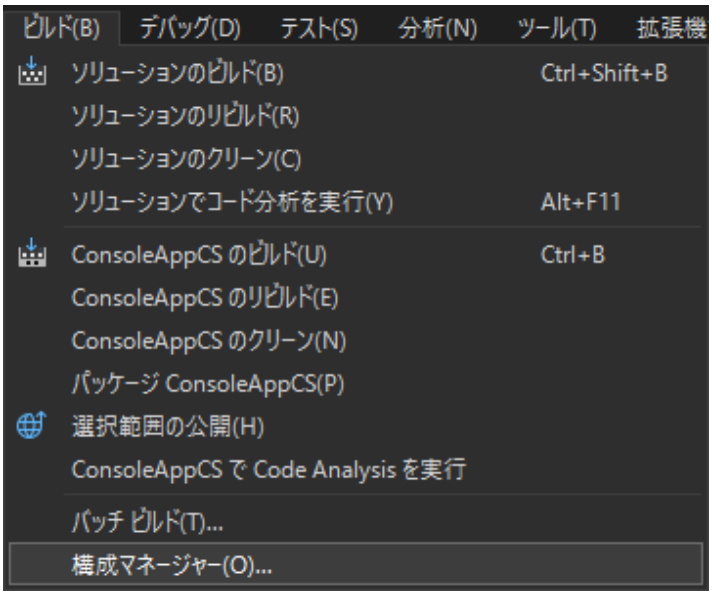


(4) 「プロジェクト」メニューの「既存の項目の追加...」を選択するとファイル選択ダイアログが表示されます。

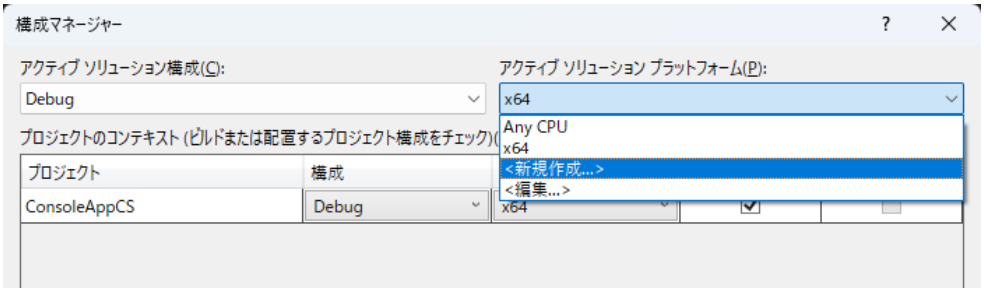


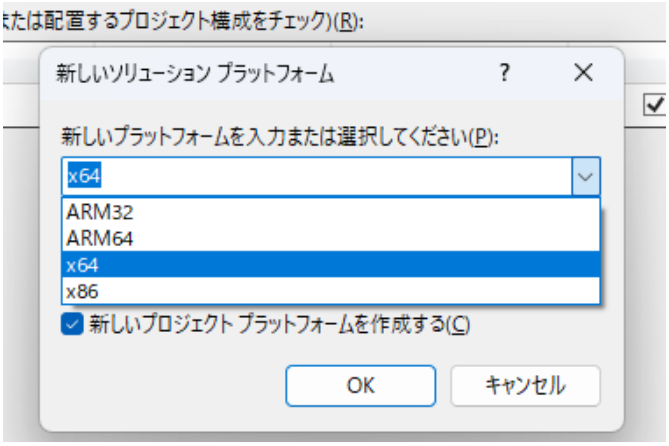
(5) サンプルの cs ファイルをひとつ選択します。C#用サンプルコードはプロジェクトのソースファイルとして追加されます。
プロジェクト生成時に作成される cs ファイルは不要です。プロジェクトから削除してください。

(6) 「ビルド」メニューの「構成マネージャー...」を選択します。

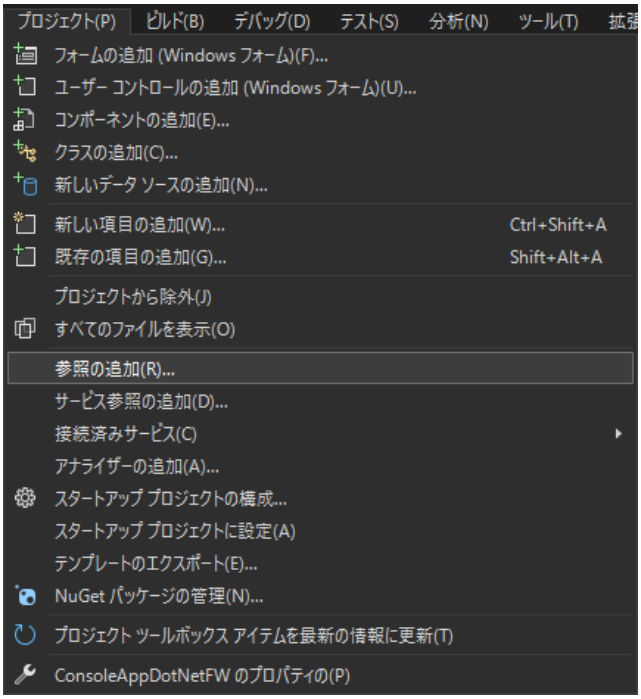


(7) 「アクティブソリューション構成」と「アクティブソリューションプラットフォーム」を設定します。プラットフォームは「x64」を選択して新規作成します。

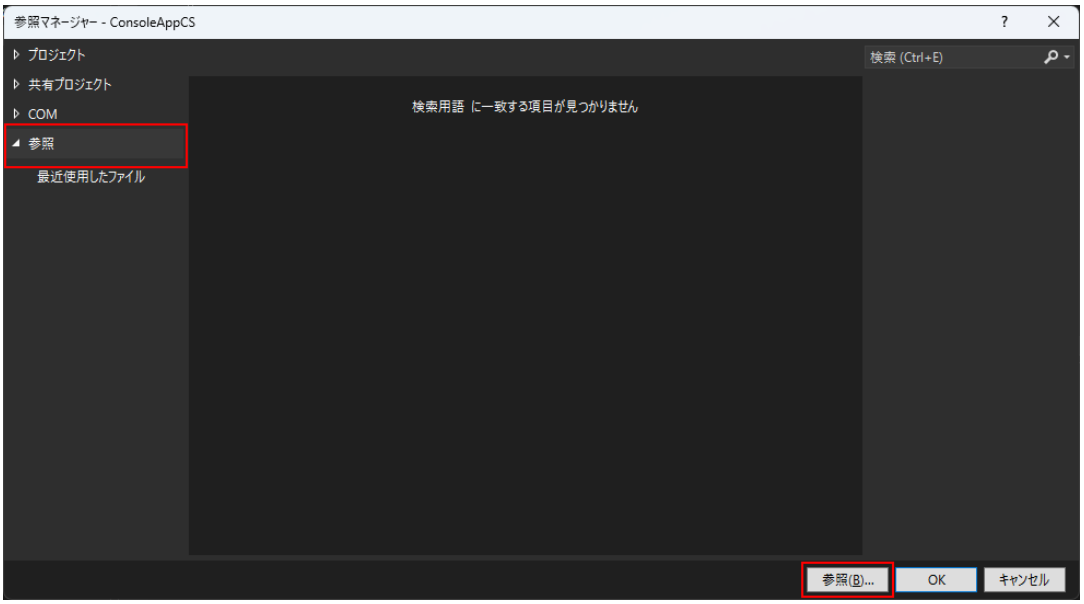




(8) 「プロジェクト」メニューの「参照の追加...」を選択すると、「参照マネージャー」ダイアログが開きます。

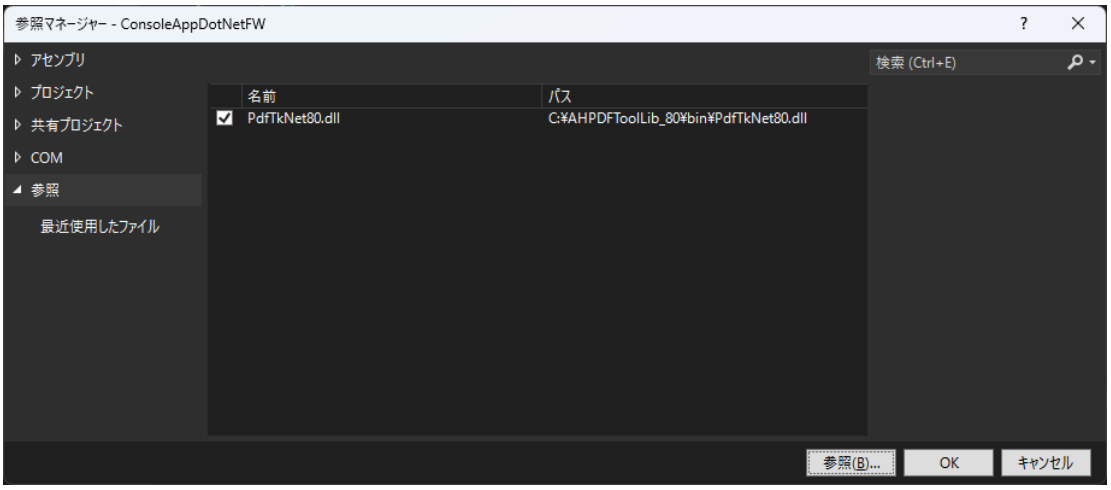


(9) 「参照」タブを開き、ダイアログ右下の「参照...」ボタンをクリックします。

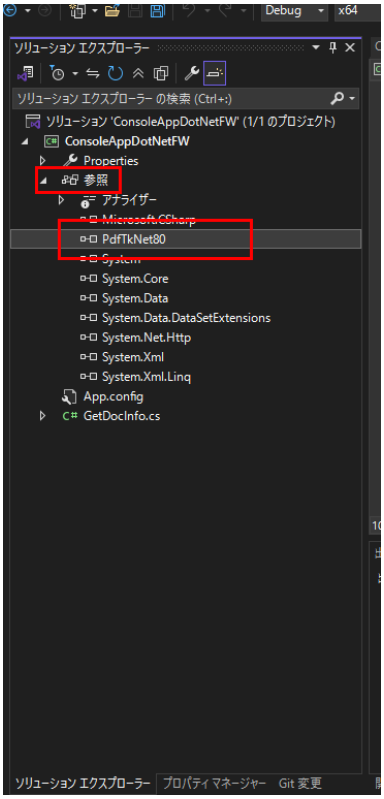


(10)ファイル選択ダイアログが開きます。インストールフォルダなどに配置された PDF Tool API モジュールファイルの「PdfTkNet80.dll」を選択します。

「OK」ボタンをクリックして「参照マネージャー」ダイアログを閉じます。

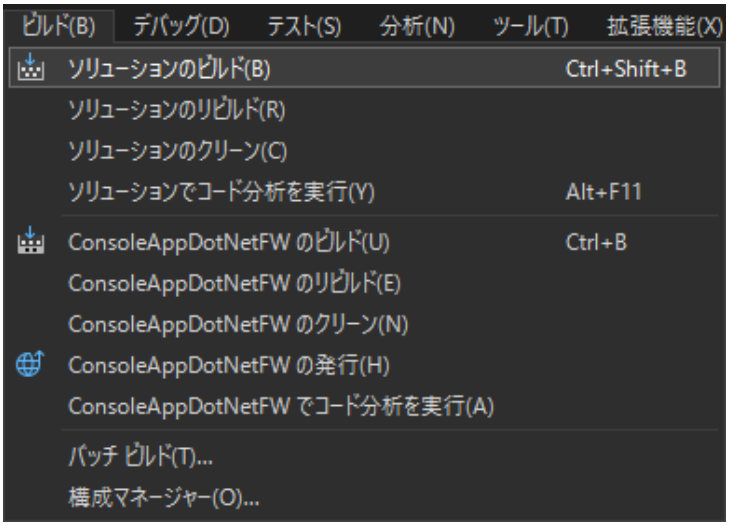


(11)参照した DLL は、ソリューションエクスプローラーの「参照」タブを開くと確認できます。



4.2.2. ビルドの設定とビルド

(1) 「ビルド」メニューの「(プロジェクト名) のビルド」をクリックすると、ビルドが開始されます。
（ソリューション内のプロジェクトが 1 つだった場合は「ソリューションのビルド」でもビルドを実行できます。）

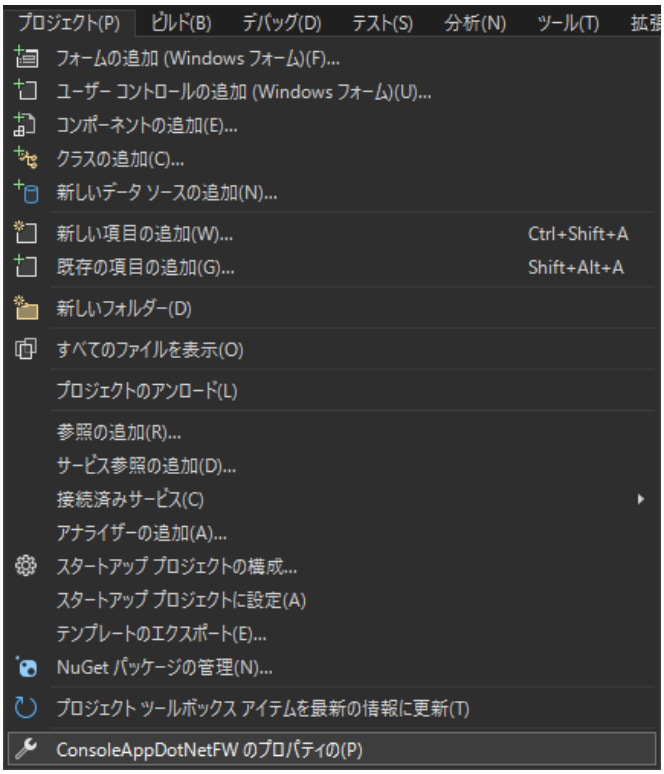


(2) ビルドが完了すると、出力ウィンドウに exe ファイルの出力先が表示されます。

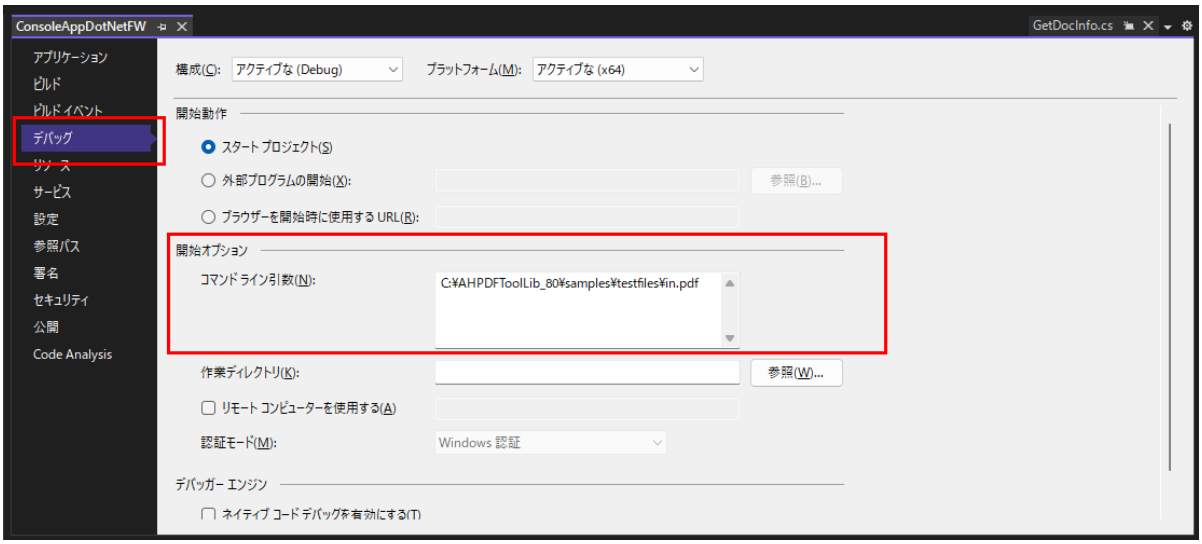
4.2.3. デバッグビルドの設定とデバッグ実行

Windows 用のデバッグビルドを行います。

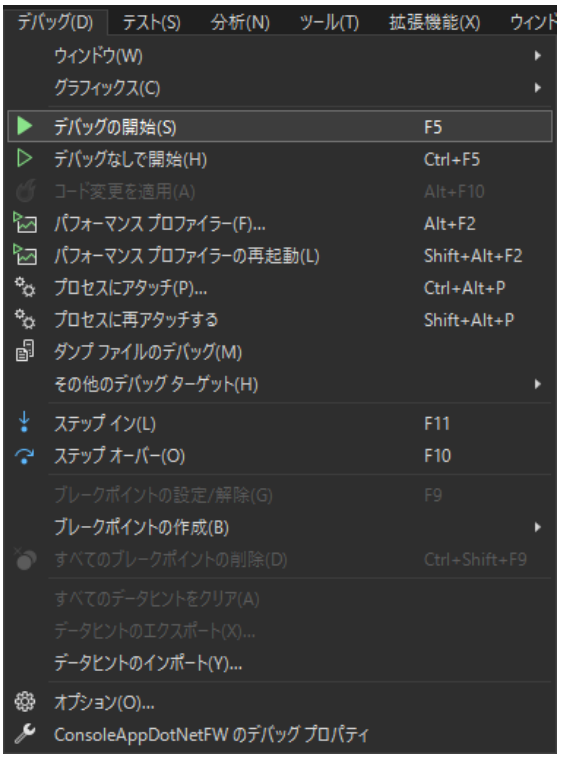
(1) 「プロジェクト」メニューの「(プロジェクト名) のプロパティ」を選択します。
(図のプロジェクト名は ConsoleAppDotNetFW です。)



(2) 指定したい引数がある場合は、左側の「デバッグ」タグを開き「開始オプション」―「コマンド引数」に引数を指定します。



(3) 「デバッグ」メニューの「デバッグの開始」をクリックします。
コンソールが新規に開き、デバッグが実行されます。



4.2.4. .NET Framework でビルドされた exe ファイルの実行

ビルドされた exe ファイルはダブルクリックやコンソールなどで実行可能です。
.NET Framework でビルドされた exe ファイルは Windows でのみ実行可能です。実行時には、以下の条件が必要です。

- 実行環境にモジュールファイルがあること（※1）（※2）
- 同じ階層に[PdfTkNet80.dll]が配置されていること

※1：
必要なモジュールファイルに関する詳細はライブラリ版マニュアルの『モジュールファイルについて』をご参照ください。

※2：
PDF Tool API のインストーラを実行した際にダイアログで指定していた場合、環境変数「PATH」にモジュールファイルへのパスが指定されています。

4.3..NET 8 サンプルコード実行ソリューションの利用方法

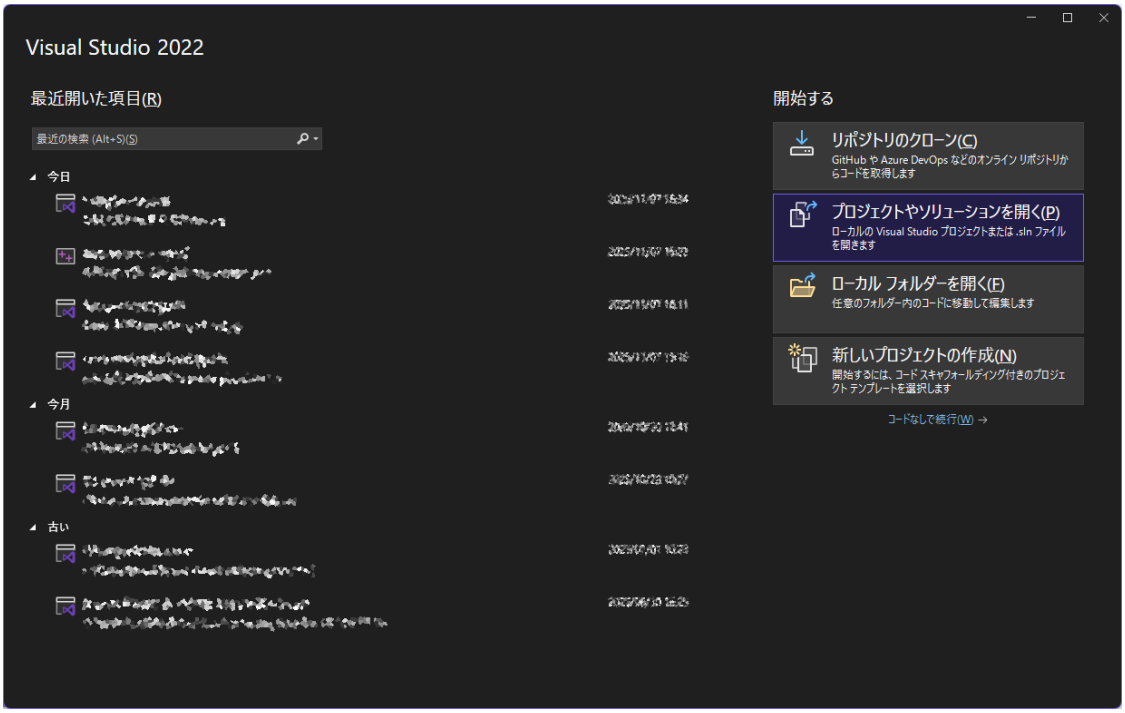
PDF Tool API のサンプルには、.NET 8 の各種サンプルコードをビルドするためのソリューションが同梱されています。本項ではその利用法を説明します。

- ソリューションファイルを用いることで、以下の項目が設定済みの状態で用意されたサンプルコードのビルドを行うことができます。
  - ソースコードの読み込み
  - 「フレームワーク」の指定
  - 「プロジェクト参照の追加...」の設定
- 本ソリューションを実行するためには Visual Studio 2022 が必要です。
- 本ソリューションファイルは以下の場所に配置されています。

  {インストールフォルダ}¥samples¥samples-net8.sln

4.3.1. .NET 8 サンプルコード実行用ソリューションの読み込み

(1) Visual Studio 2022 を起動し、「プロジェクトやソリューションを開く」を選択します。

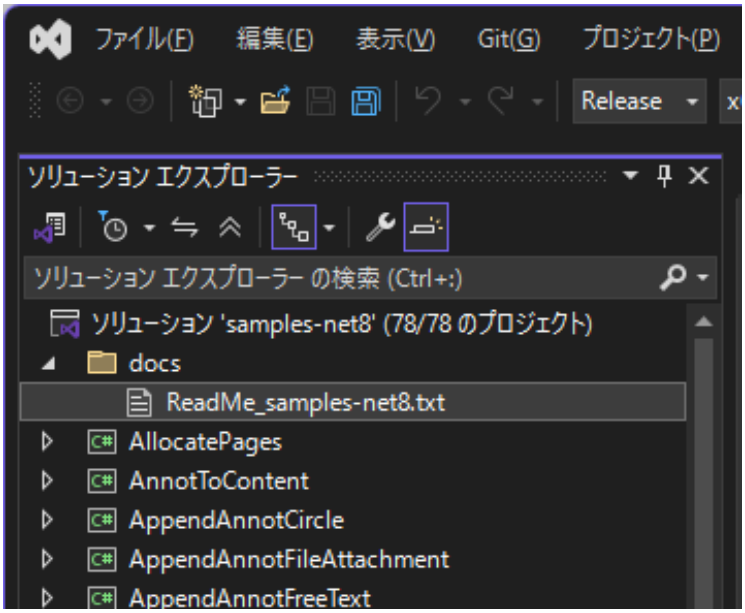


(2) 配置フォルダにある「samples-net8.sln」を開きます。

(3) 以下の手順でサンプルコード実行の説明書を開きます。

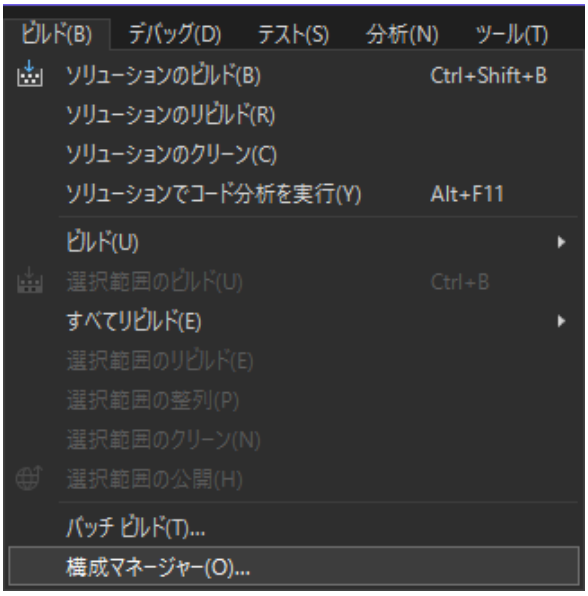
ソリューション エクスプローラーから doc フォルダを選択し、内部にある「ReadMe_samples-net8.txt」を開きます。

各サンプルの実行内容や実行に必要なモジュールファイルに関しては「ReadMe_samples-net8.txt」をご参照ください。

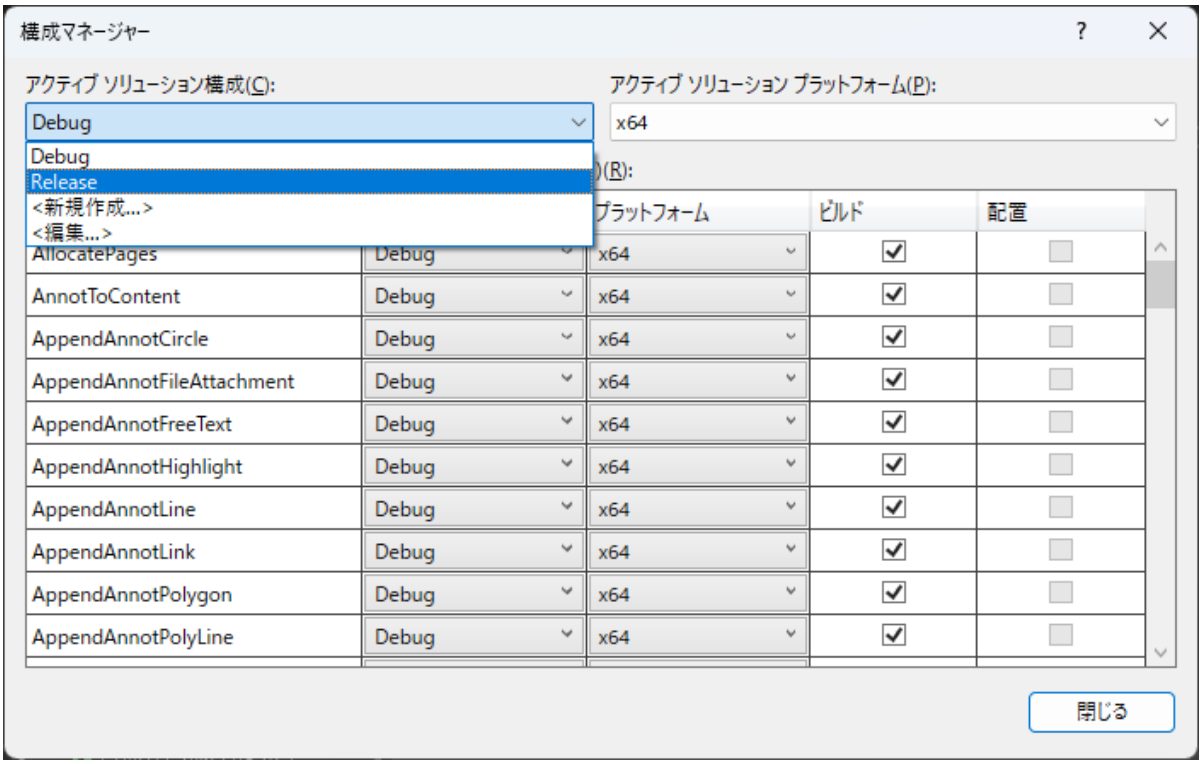


4.3.2. ビルドの設定とビルド

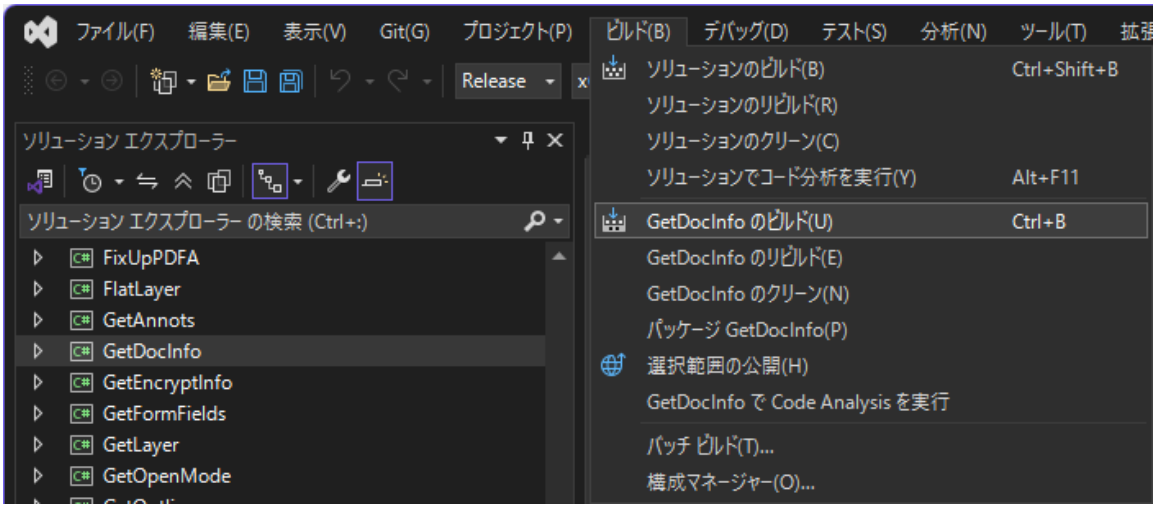
(1) 「ビルド」メニューの「構成マネージャー...」を選択します。



(2) 「アクティブソリューション構成」は「Release」を、プラットフォームは「x64」を選択して「OK」ボタンを押して構成マネジャー画面を閉じます。



(3) 実行したいサンプルのプロジェクトを選択し、ビルドします。
ソリューション エクスプローラーから実行したいサンプルのプロジェクトを選択します。
「ビルド」メニューを開きます。
「(プロジェクト名)のビルド」をクリックすると、ビルドが開始されます。
(実行したいプロジェクトを右クリックして「ビルド」を選択することでも実行できます)



(4) コンソールウィンドウの出力メッセージに、ビルドされた exe ファイルの出力パスが記載されます。

4.3.3. アプリケーションの発行

『4.1 .NET 8(C#)サンプルコードのビルドと発行方法』にある『4.1.5 アプリケーションの発行』をご参照ください。

5.Java のコンパイルと実行手順

5.1. Java 個別のサンプルコードのコンパイル・実行

ここでは、Java のサンプルコードを指定してコンパイル・実行する方法を解説します。

- 本章の各実行例におけるパスは、PDF Tool API のインストール時にパス指定をしなかった場合の配置パスとなっています。具体的なパスは『PDF Tool API の開発環境を整える』の『2.1.1 インストーラによる配置』をご参照ください。
- 対応する Java バージョンなどに関する詳細はライブラリ版マニュアルの『対応プログラム言語』をご参照ください。
- Java のサンプルコードのソースファイルは、以下の場所に配置されています。

  {インストールフォルダ}¥samples¥java

5.1.1. Java コンパイル・実行環境の設定

(1) Java でコンパイル・実行をする際に必要な環境変数を設定します。
Windows 環境に必要なもの（※1）に加えて、Java では以下の環境変数も設定される必要があります。

環境変数名	設定値
CLASSPATH	「PdfTkJava80.jar」のフルパス

環境変数をコマンドプロンプト上で指定する場合、以下の例のようになります。
他の環境変数と異なり、CLASSPATH はインストーラによって自動で設定されることはありません。従って、手動での設定が必要です。

```
> set PATH=C:¥AHPDFToolLib_80¥bin;%PATH%
> set PTL80_LIC_PATH=C:¥AHPDFToolLib_80¥License
> set PTL80_FONT_CONFIGFILE=C:¥AHPDFToolLib_80¥fontconfig¥font-config.xml
> set PTL80_ICCPROFILE_PATH=C:¥AHPDFToolLib_80¥icc
> set CLASSPATH=C:¥AHPDFToolLib_80¥bin¥PdfTkJava80.jar;%CLASSPATH%
```

（※1）

Windows 環境で必要な環境変数の詳細は『2.4 Windows 開発・実行環境における環境変数のまとめ』をご参照ください。

注意：アプリケーションサーバにおける Java 使用上の注意

Tomcat などのアプリケーションサーバにおいて Java インターフェイスを使用する場合、「PdfTkJava80.jar」を WEB アプリケーションの「WEB-INF/lib」に置かないようにしてください。

代わりに、システムクラスローダなどロードが一度だけ行われるクラスローダで読み込ませるように設定してください。

JavaVM では、仕様により JNI のネイティブライブラリは複数のクラスローダから読み込めません。そのため、各 WEB アプリケーションディレクトリに PdfTkJava80.jar を置いた場合、複数の WEB アプリケーションから使用することができなくなります。

システムクラスローダの利用はこれを防ぐための措置となります。

5.1.2. サンプルコードのコンパイル

ここでは Java サンプルコードのコンパイル・実行をコマンドプロンプト上で実行する方法を解説します。

- サンプルコードはエンコーディング「UTF-8」を使用しています。
- サンプルコードの配置フォルダについては『5.1 Java 個別のサンプルコードのコンパイル・実行』をご参照ください。

以下は、サンプルプログラム「GetDocInfo.java」をコンパイルして実行する例です。

(1) カレントディレクトリをサンプルコードの配置フォルダに切り替えます。

```
> cd C:¥AHPDFToolLib_80¥samples¥java
```

(2) コンパイルしたい java ファイルを指定し、コンパイルします。

```
> javac -encoding UTF-8 GetDocInfo.java
```

5.1.3. Java サンプルの実行

Java 実行に必要な環境変数等の設定は『5.1.1 Java コンパイル・実行環境の設定』をご参照ください。

(1) サンプルを実行するためにはコンパイルされた class ファイルをパッケージフォルダに移動する必要があります。
具体的には、サンプルコードでは「package Sample;」が定義されています。そのため、本例

では「GetDocInfo.class」を「Sample」フォルダに移動します。

上記操作は例えば、以下コマンドで実現可能です。

```
> md Sample
> move GetDocInfo.class Sample¥GetDocInfo.class
```

(2) コンパイルした class ファイルを実行します。

```
java Sample.GetDocInfo in.pdf
```

注意：

java コマンド実行時、カレントディレクトリはパッケージフォルダの上の階層である必要があります。

例えば、class ファイルを移動した[Sample]フォルダが[C:¥test]に配置されている場合、

実行時のカレントディレクトリは[C:¥test]である必要があります。[C:¥test¥Sample]ではありません。

5.2.同梱の Java 実行用バッチファイルについて

PDF Tool API のサンプルフォルダには Java インターフェースでサンプルプログラムを簡易に実行するためのバッチファイル「samples-java.bat」が同梱されています。

- バッチファイルを用いることで、以下の項目が設定済みの状態で用意されたサンプルコードのコンパイルと実行を行うことができます。
  - ソースコードの読み込み
  - 環境変数の設定
  - 各サンプルに適した引数及びサンプル PDF の指定
- バッチファイルの使用方法やサンプルプログラムの内容に関する解説書も同梱されています。詳細は配置フォルダにあります『ReadMe_samples-java.txt』をご参照ください。
- バッチファイル及び解説書の配置フォルダは以下のパスにあります。

  {インストールフォルダ}¥samples

5.2.1. バッチファイルの操作方法

本項では、バッチファイルの簡易な実行方法を解説します。

(1) 実行したいサンプルプログラムを引数に指定し、コマンドラインから「samples-java.bat」を呼び出します。

```
> samples-java.bat ImageToPdf
```

実行が完了すると完了メッセージが出てバッチファイルが終了します。

(2) バッチファイルと同じ階層に以下のフォルダが生成され、各実行結果が出力されます。
  - 「Sample」フォルダ：コンパイルされた class ファイルの格納先
  - 「java-out」フォルダ：実行結果の出力先
    サンプルプログラムで処理された PDF などが「(プログラム名)_out」の名前で出力されます。
  - 「java-out-ExtractPage」フォルダ：サンプルプログラム「ExtractPage」専用の出力先

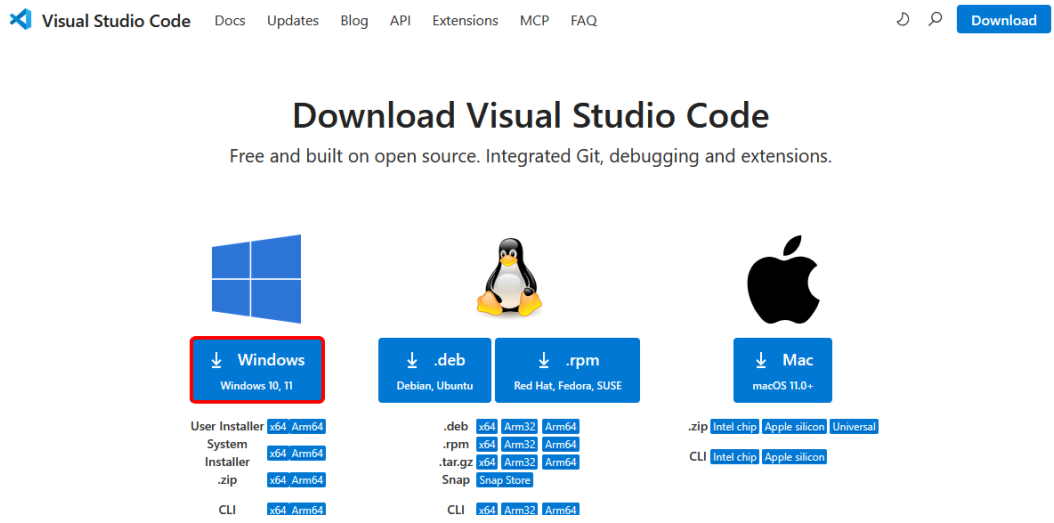
6.Visual Studio Code でのプログラム作成と実行方法

ここでは、Visual Studio Code(VSCode)を使用した、各種プログラム作成と実行方法を説明します。

6.1.Visual Studio Code のインストールと準備

6.1.1. Visual Studio Code のインストール

(1) Visual Studio Code の公式サイトよりインストーラをダウンロードします。
『Download Visual Studio Code』を開いて「windows」を選択してください。
https://code.visualstudio.com/download



(2) ダウンロードした exe ファイルをクリックして開くとインストールが始まります。

6.1.2. Visual Studio Code の日本語化

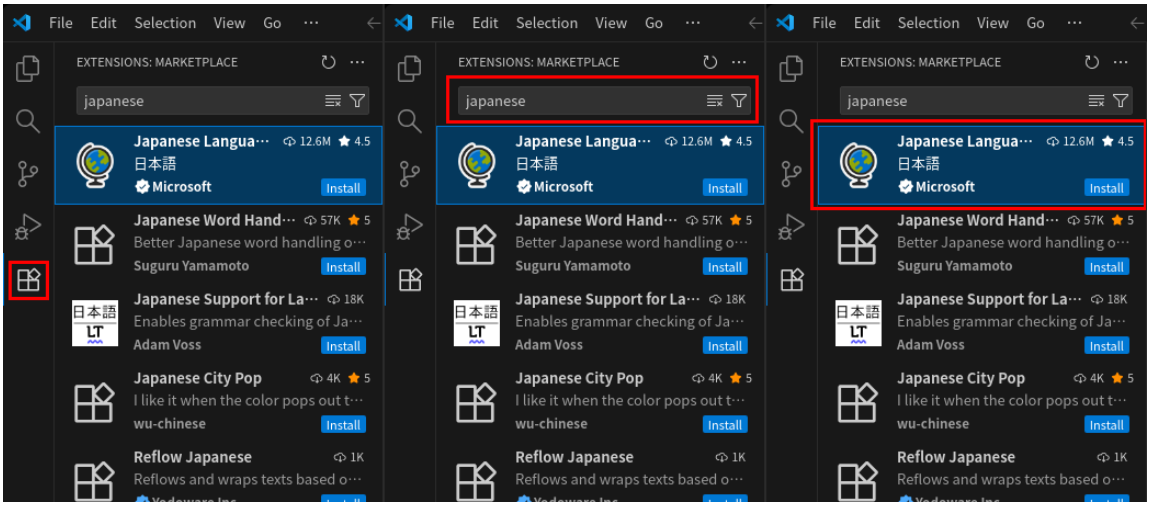
VSCode の日本語化拡張「Japanese Language Pack for VS Code」をインストールします。

(1) Visual Studio Code を起動し、立ち上げる

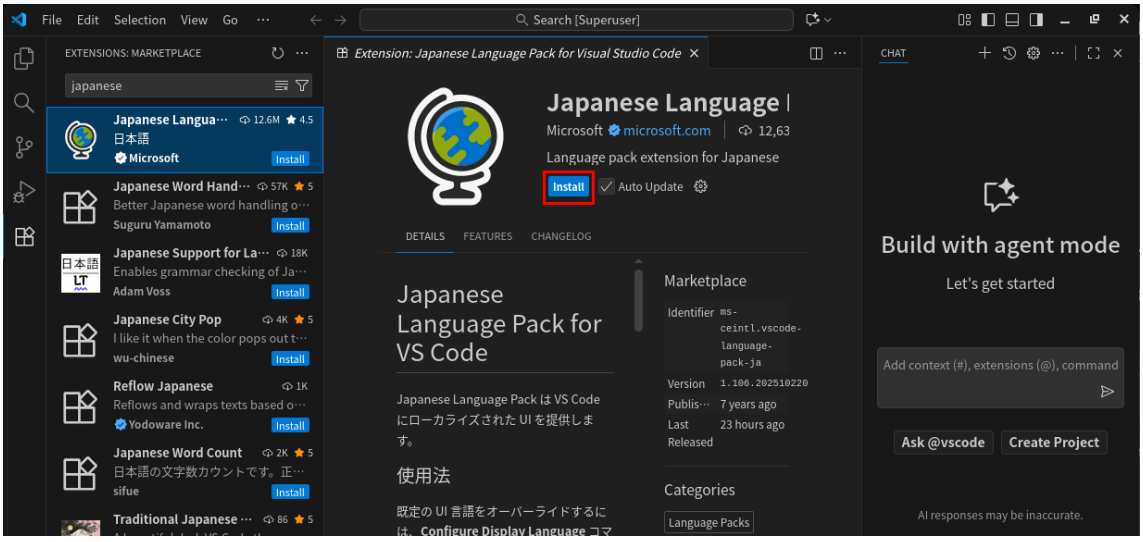
(2) 左側アクティビティバーにある拡張機能をクリック

(3) 開いたサイドバー上部にある検索欄に「Japanese」と記入

(4) 検索結果に表れる「Japanese Language Pack for VS Code」を選択



(5) 青色の「install」ボタンを押す



(6) インストール終了後に VSCode の再起動をする

再起動後は VSCode の表示が日本語になっています。

6.2..NET 8 のサンプルコードビルド・実行

6.2.1. .NET SDK のインストール

.NET 8.0 用ダウンロードサイトへアクセスし、「SDK 8.0.xx」の項にある「Windows」の「x64」インストーラーをダウンロードして実行します。

.NET8.0 のダウンロード

https://dotnet.microsoft.com/ja-jp/download/dotnet/8.0

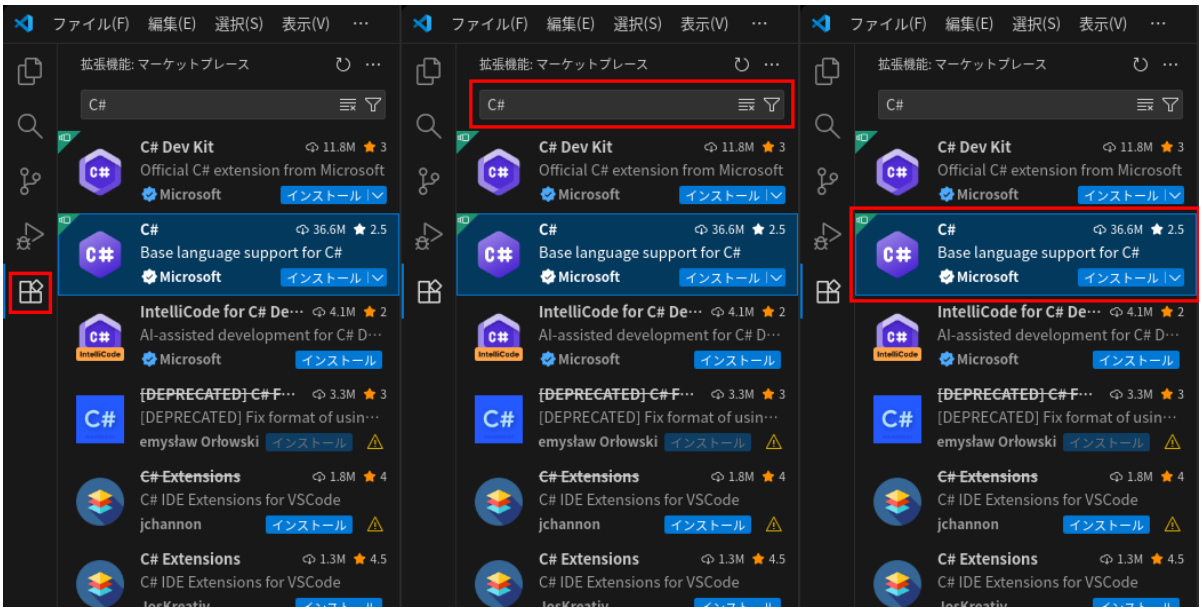
6.2.2. Visual Studio Code 拡張のインストール

VSCode 拡張機能「C#」をインストールします。

(1) 左側アクティビティバーにある拡張機能をクリック

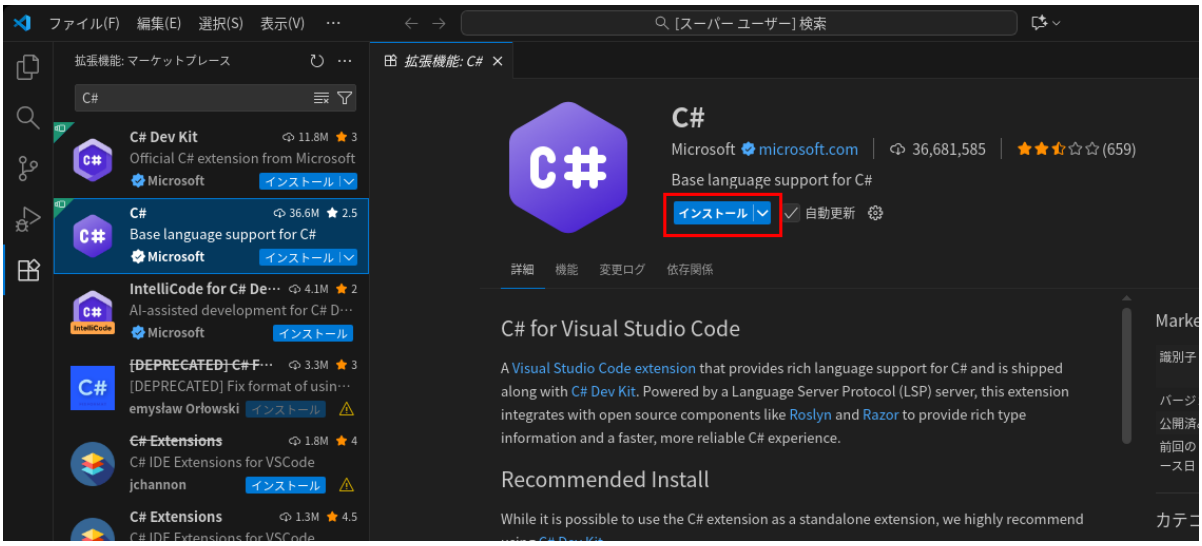
(2) 開いたサイドバー上部にある検索欄に「C#」と記入

(3) 結果に表れる「C#」を選択



(4) 青色の「install」ボタンを押す

これで C#の拡張機能のインストールが完了します。



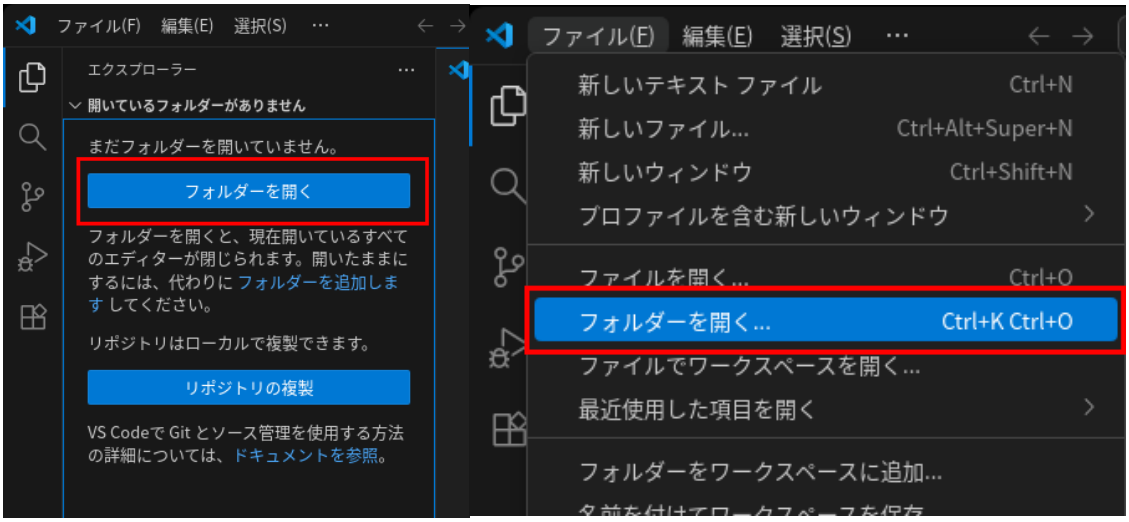
6.2.3. プロジェクトの作成

「フォルダを開く」と「ターミナルによるプロジェクトの作成」の２つの手順を踏む必要があります。

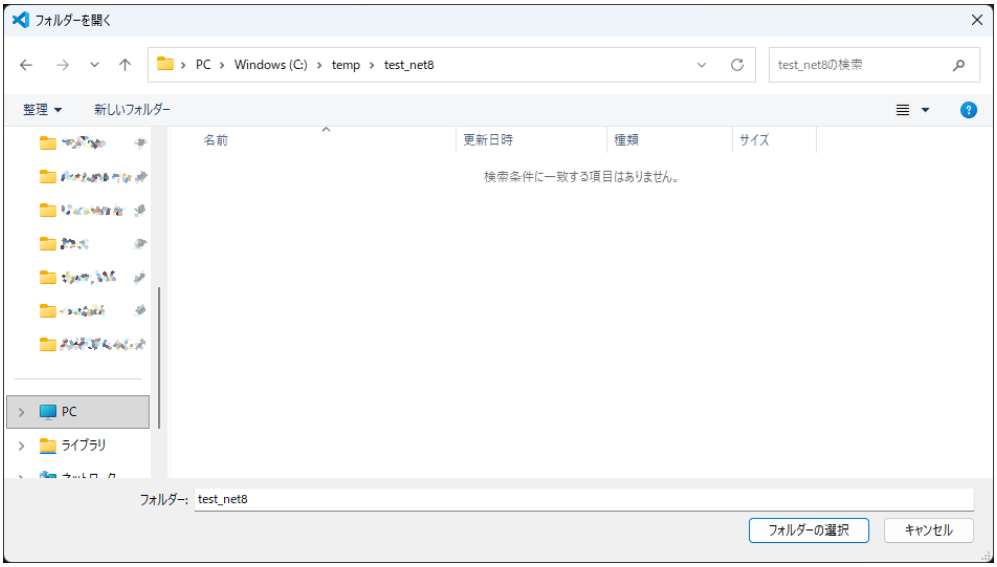
6.2.3.1. フォルダを開く

(1) 左側アクティビティバーにあるエクスプローラーを選択します。

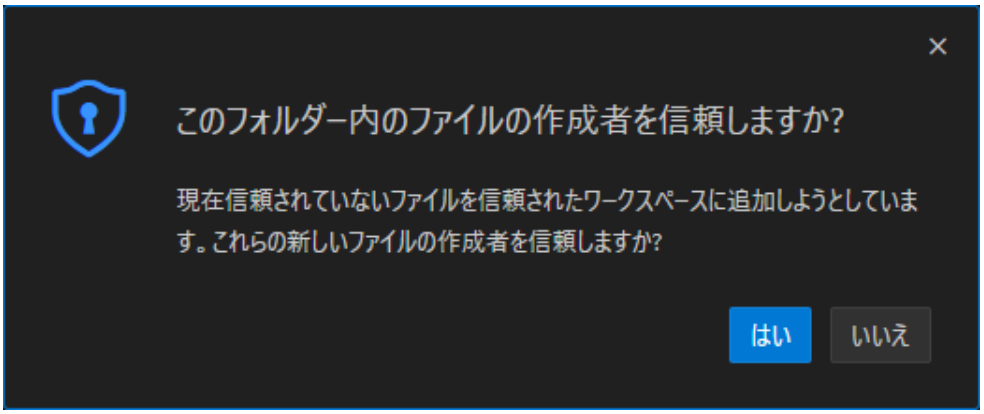
(2) サイドバーに表示された「フォルダーを開く」をクリックします。
または、メニューバー左上の「ファイル」を選択し、開いたメニューの「フォルダーを開く」をクリックします。



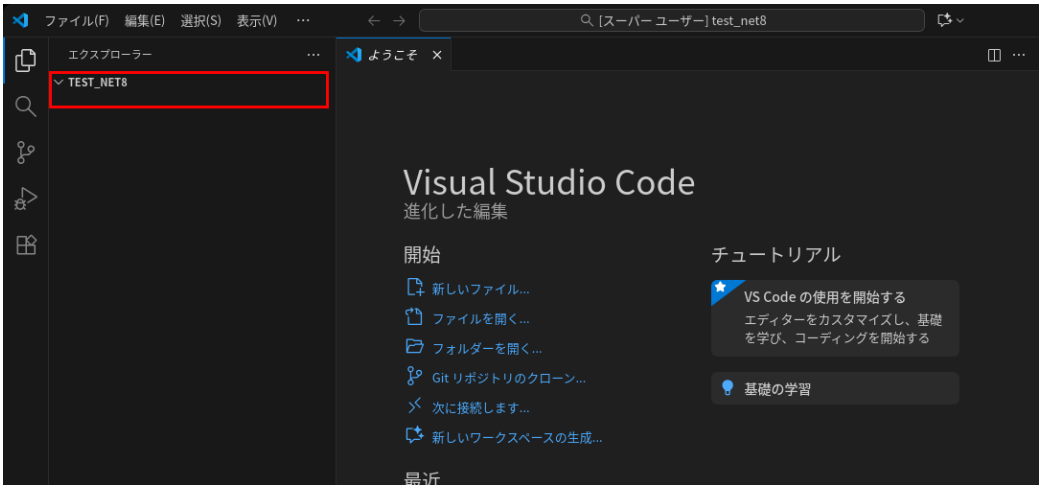
(3) プロジェクトのファイルを生成するフォルダを指定して右上の「開く」をクリックします。
（画像では test_net8 というフォルダを指定しています）



(4) 下記のダイアログが表示されるので「はい、作成者を信頼します」を選択します。

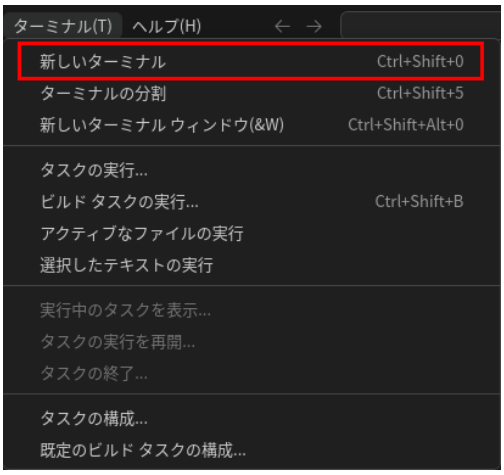


(5) 左側のサイドバーのエクスプローラーに指定したフォルダ名が表示されます。

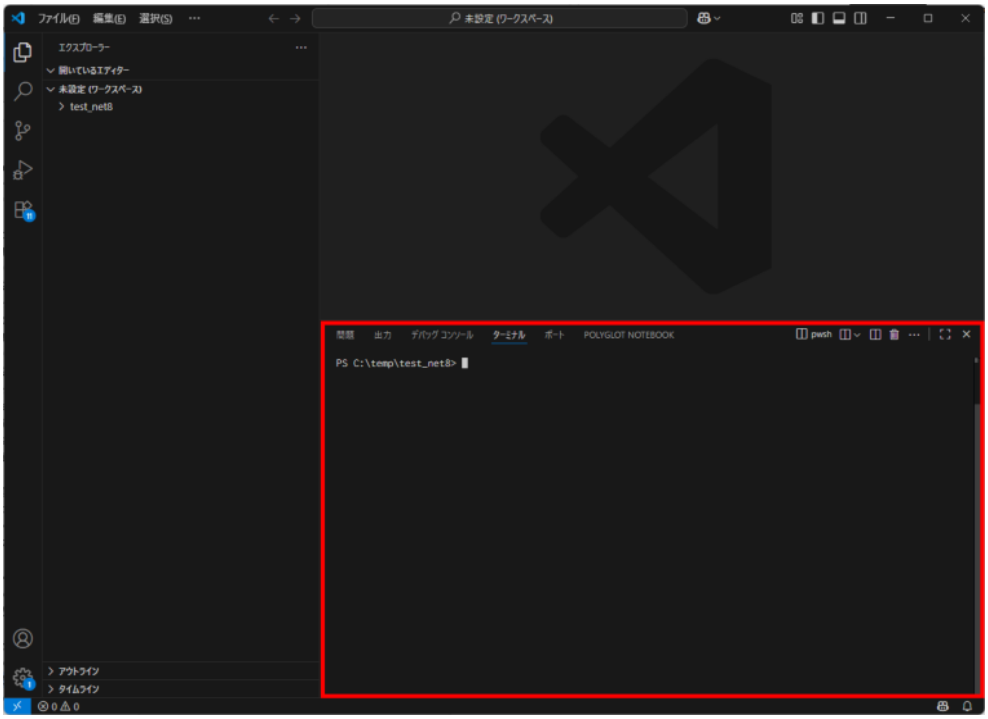


6.2.3.2. ターミナルによるプロジェクトの作成

(1) 「ターミナル」メニューの「新しいターミナル」を選択します。



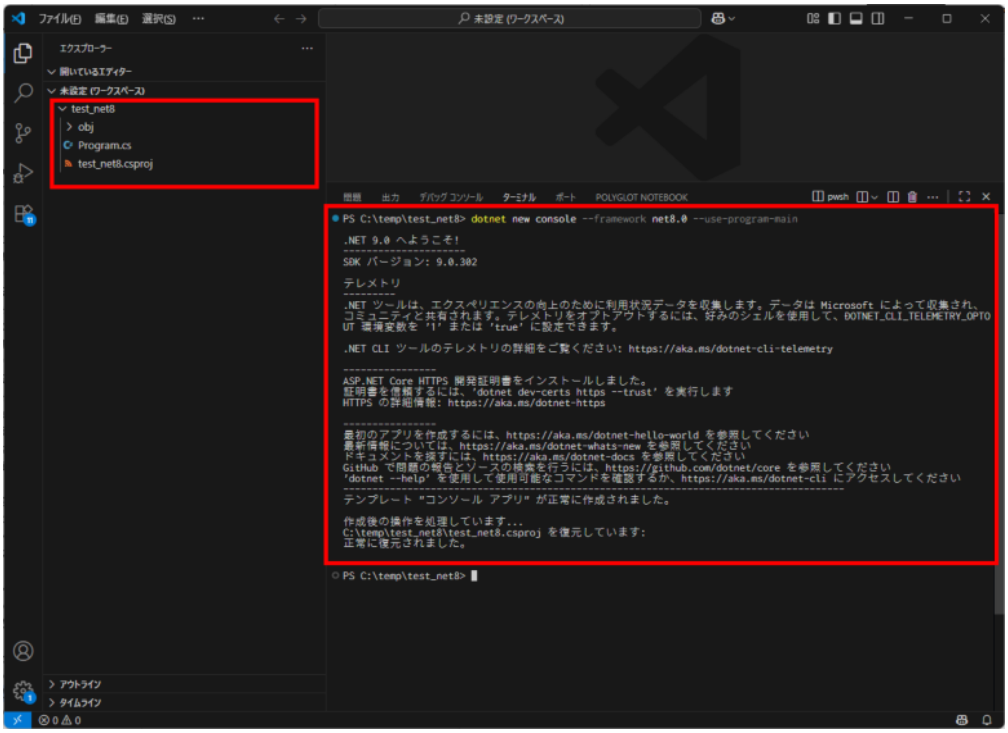
新しいターミナル画面が開きます。



(2) 開かれたターミナル画面に以下の dotnet new コマンドを入力します。

```
dotnet new console --framework net8.0 --use-program-main
```

これにより新しいコンソールアプリケーションのプロジェクトが作成されます。



（ 参考：dotnet new <TEMPLATE> - .NET CLI | Microsoft Learn ）

6.2.4. PDF Tool API のサンプルコードを使用したプログラムの作成

(1) PDF Tool API のインストール
『2 PDF Tool API の開発環境を整える』をご参照ください。

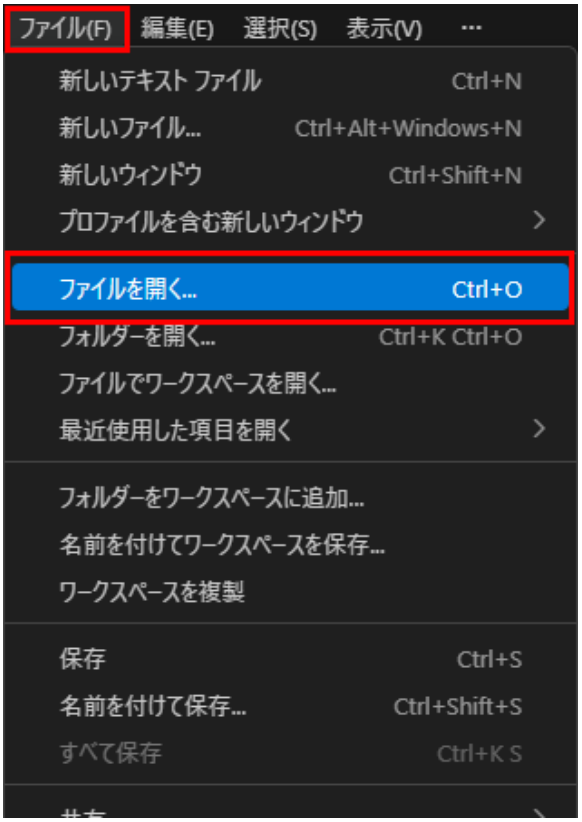
(2) VSCode を起動してプロジェクトを作成します。
『6.2.3 プロジェクトの作成』をご参照ください。

(3) 「(フォルダ名) .csproj」 を開いて以下の赤字箇所を追記します。

<HintPath> に記入するのは PdfTkNet8_80.dll の配置パスです。
以下の例は PDF Tool API のインストール時にパス指定をしなかった場合です。

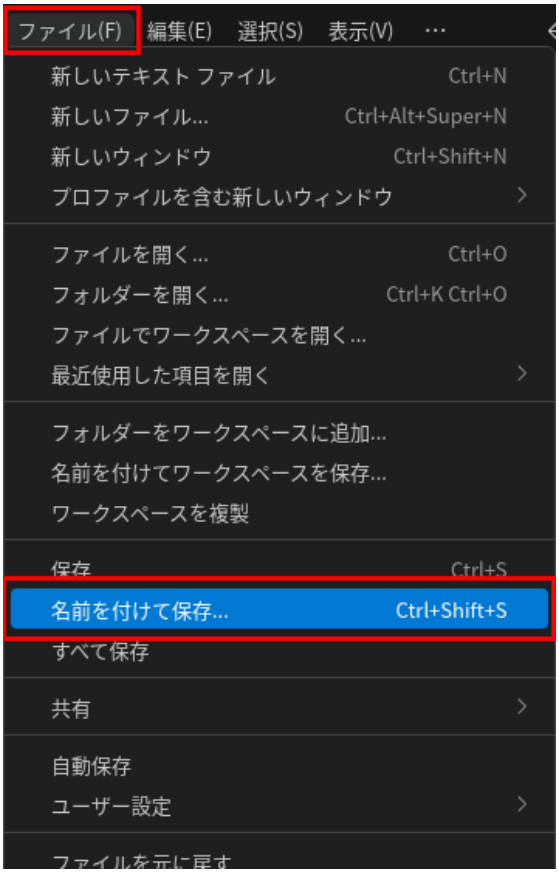
```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net8.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Platform>x64</Platform>
    <Nullable>enable</Nullable>
  </PropertyGroup>

  <ItemGroup>
    <Reference Include="PdfTkNet8_80">
      <HintPath> C:¥AHPDFToolLib_80¥bin¥PdfTkNet8_80.dll</HintPath>
    </Reference>
  </ItemGroup>

</Project>
```

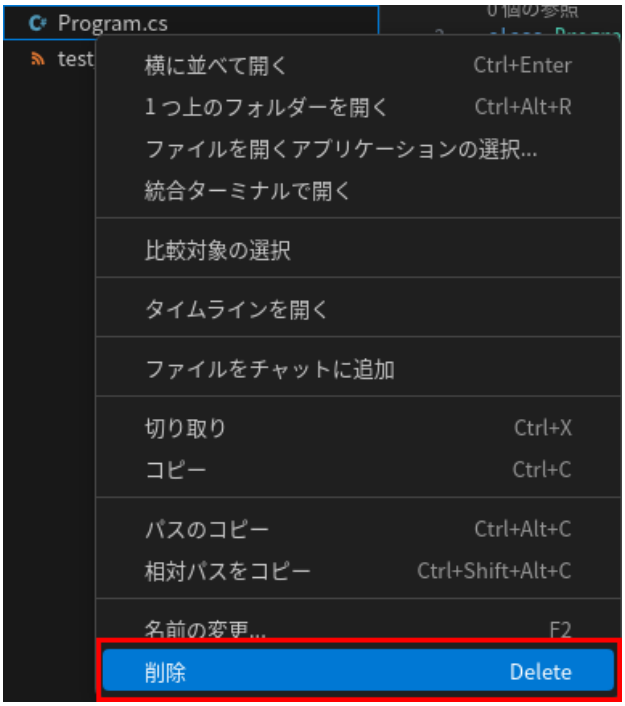
(4) 「ファイル」メニューの「ファイルを開く」を選択して使用したいサンプルコードの cs ファイルを開きます。



「このワークスペースで信頼されていないファイルを許可しますか？」のダイアログが出た場合は「開く」を選択します。

(5) 「ファイル」メニューの「名前を付けて保存」を選択してプロジェクト生成時に作成された「Program.cs」に上書き保存します。
または、「ファイル」メニューの「名前を付けて保存」を選択して別名保存を行い、その後「Program.cs」を選択し右クリックメニューで「削除」します。





(6) ターミナルに下記の dotnet build コマンドを入力しビルドを行います。

```
dotnet build -c Release
```

(7) ビルドが完了したら、出力フォルダに exe ファイルが出力されています。
具体的な実行方法は『4.1.6 .NET 8 で発行された実行ファイルの実行方法について』をご参照ください。
または、ターミナルに下記の dotnet run コマンドを入力することでも実行可能です。

```
dotnet run
```

プログラムに引数が必要な場合は「run」のあとに引数を記入してください。

6.3..NET Framework のサンプルコードビルド・実行

6.3.1. .NET Framework SDK のインストール

ここでは.NET Framework 4.8.1 のインストールについて説明します。

(1) .NET Framework 用ダウンロードサイトへアクセスし、「アプリのビルド-開発パック」の項にある「開発者パック」インストーラーをダウンロードして実行します。

.NET Framework 4.8.1 のダウンロード

https://dotnet.microsoft.com/ja-jp/download/dotnet-framework/net481

なお、必要に応じて日本語の言語パックもダウンロード・インストールをしてください。

6.3.2. Visual Studio Code 拡張のインストール

『6.2 .NET 8 のサンプルコードビルド・実行』内の『6.2.2 Visual Studio Code 拡張のインストール』をご参照ください

6.3.3. プロジェクトの作成

「フォルダを開く」と「ターミナルによるプロジェクトの作成」の２つの手順を踏む必要があります。

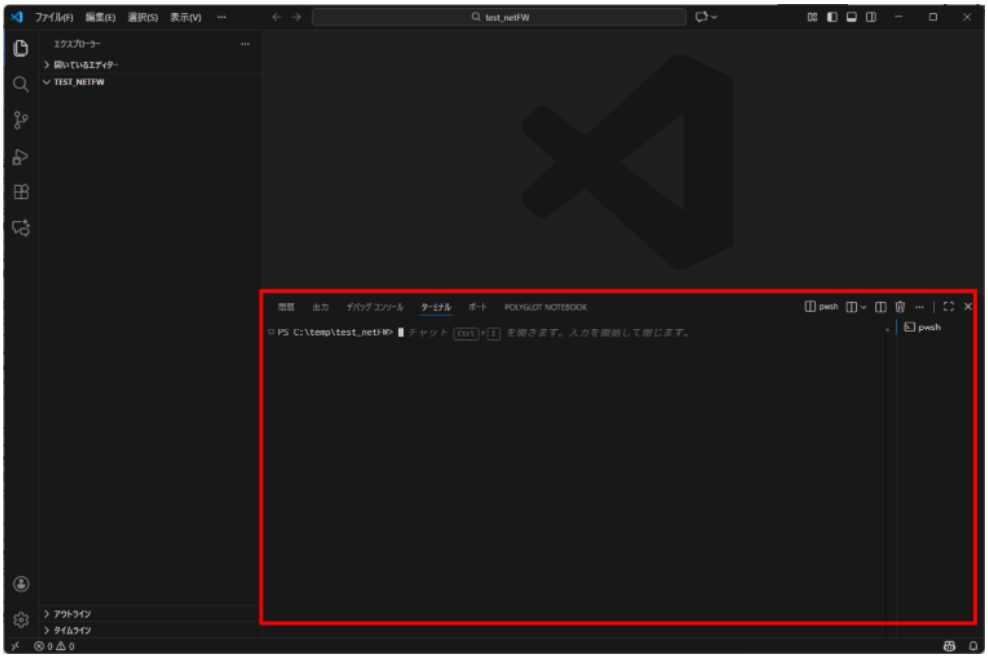
6.3.3.1. フォルダを開く

『6.2 .NET 8 のサンプルコードビルド・実行』内の『6.2.3.1 フォルダを開く』をご参照ください

6.3.3.2. ターミナルによるプロジェクトの作成

(1) 「ターミナル」メニューの「新しいターミナル」を選択します。

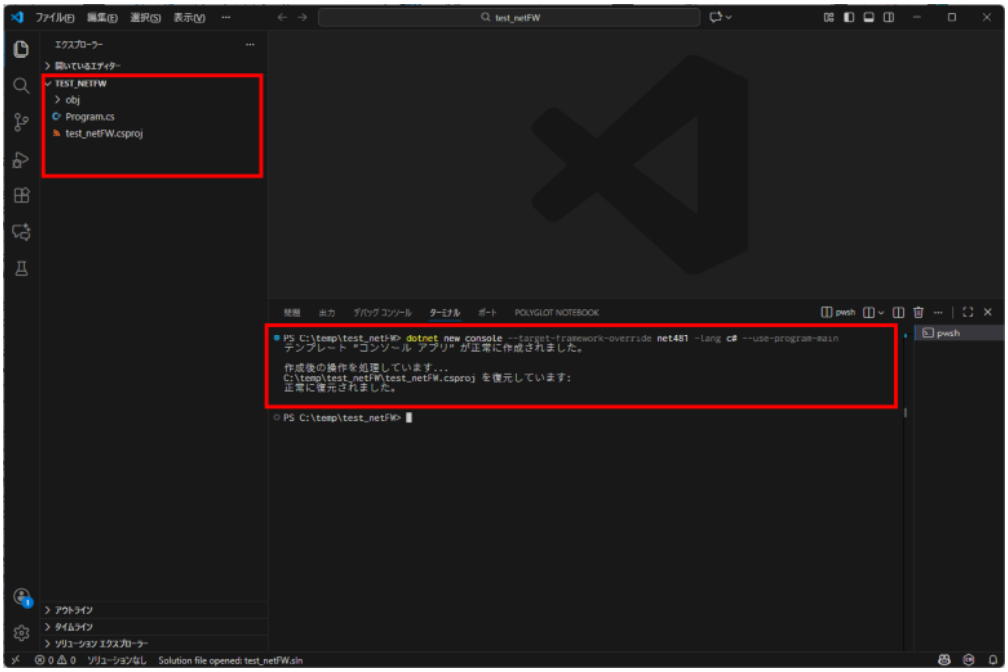
新しいターミナル画面が開きます。



(2) 開かれたターミナル画面に以下の dotnet new コマンドを入力します。(※)

```
dotnet new console --target-framework-override net481 -lang c# --use-program-main
```

これにより新しいコンソールアプリケーションのプロジェクトが作成されます。



※注意：

dotnet new コマンドで.NET Framework を指定することができる[--target-framework-override]オプションは公式リファレンスには掲載されていない、非公式なものです。

当該コマンドが非公式にされている理由としては.NET フォーラム上で、dotnet コマンドは Windows や Linux などのマルチプラットフォームで動作させることを想定している反面、.NET Framework は Windows のみでしか動作しないことが示唆されています。

参考：

dotnet new <TEMPLATE> - .NET CLI | Microsoft Learn

https://learn.microsoft.com/ja-jp/dotnet/core/tools/dotnet-new

Target frameworks in SDK-style projects | Microsoft Learn

https://learn.microsoft.com/ja-jp/dotnet/standard/frameworks

Using dotnet new for targeting .NET Framework | .NET GitHub

https://github.com/dotnet/templating/issues/1406

6.3.4. PDF Tool API のサンプルコードを使用したプログラムの作成

(1) PDF Tool API のインストール

『2 PDF Tool API の開発環境を整える』をご参照ください。

(2) VSCode を起動してプロジェクトを作成します。

『6.3.3 プロジェクトの作成』をご参照ください。

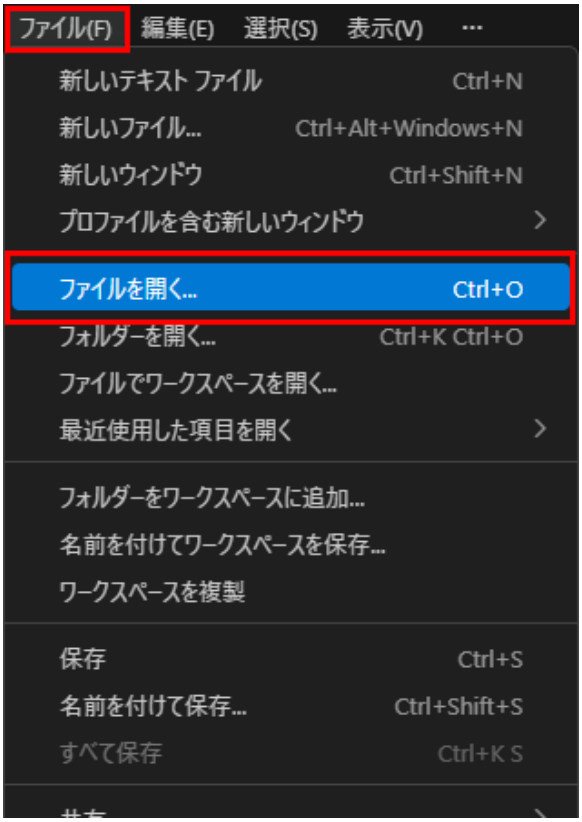
(3) 「(フォルダ名) .csproj」 を開いて以下の赤字箇所を追記します。

<HintPath> に記入するのは PdfTkNet80.dll の配置パスです。

以下の例は PDF Tool API のインストール時にパス指定をしなかった場合です。

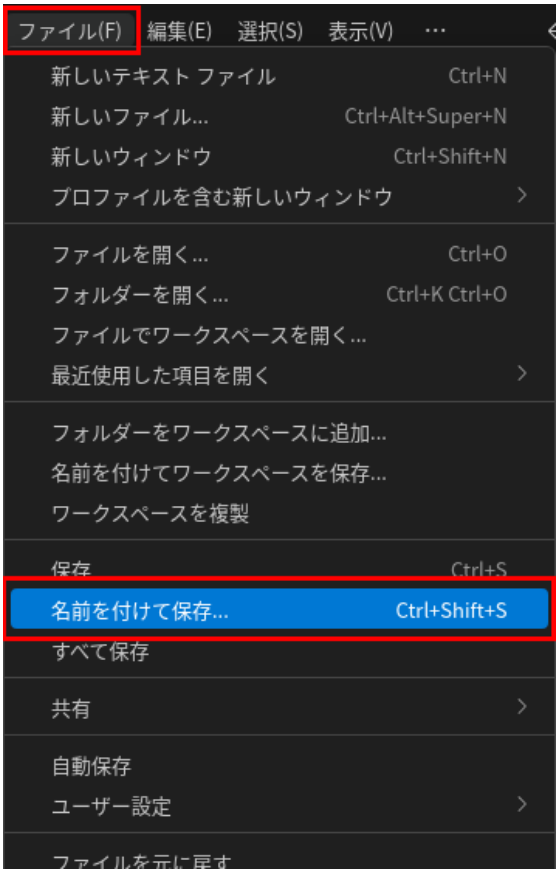
```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net481</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Platform>x64</Platform>
    <LangVersion>10.0</LangVersion>
    <Nullable>enable</Nullable>
  </PropertyGroup>

  <ItemGroup>
    <Reference Include="PdfTkNet80">
      <HintPath>C:¥AHPDFToolLib_80¥bin¥PdfTkNet80.dll</HintPath>
    </Reference>
  </ItemGroup>

</Project>
```

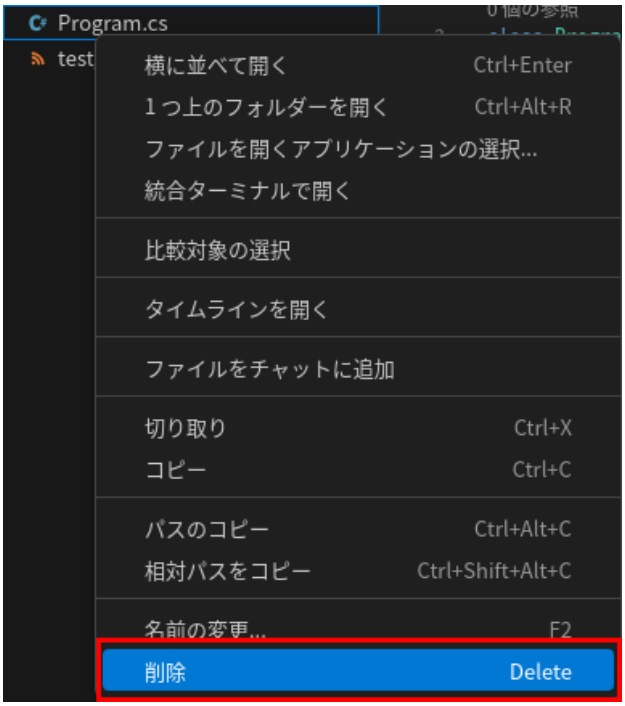
(4) 「ファイル」メニューの「ファイルを開く」を選択して使用したいサンプルコードの cs ファイルを開きます。



「このワークスペースで信頼されていないファイルを許可しますか？」のダイアログが出た場合は「開く」を選択します。

(5) 「ファイル」メニューの「名前を付けて保存」を選択してプロジェクト生成時に作成された「Program.cs」に上書き保存します。
または、「ファイル」メニューの「名前を付けて保存」を選択して別名保存を行い、その後「Program.cs」を選択し右クリックメニューで「削除」します。





(6) ターミナルに下記の dotnet build コマンドを入力しビルドを行います。

```
dotnet build -c Release
```

(7) ビルドが完了すると、出力フォルダに exe ファイルが出力されます。
具体的な実行方法に関しては『4.2.4 .NET Framework でビルドされた exe ファイルの実行』をご参照ください。
または、ターミナルに下記の dotnet run コマンドを入力することで、実行可能です。

```
dotnet run
```

プログラムに引数が必要な場合は「run」のあとに引数を記入してください。

履歴

日付	更新内容
2025.11.25	・初版

Antenna House PDF Tool API V8.0

サンプルコードのビルドと実行手順・(Windows)

2025.11.25