Antenna House PDF Tool API V8.0 サンプルコードのビルドと実行手順・(.NET8)

アンテナハウス株式会社

目次

1.はじめに

本書では、まず、PDF Tool API の C#サンプルコードを Windows 環境の Microsoft Visual Studio 2022 でビルドあるいはアプリケーション発行する方法と、発行したアプリケーションファイルを Windows と Linux それぞれで実行する方法を説明します。
次に、Visual Studio Code を利用して、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 において、Microsoft VIsual Studio 2022 を利用して PDF Tool API のサンプルコードをビルドするための環境整備について説明します。
Linux 版などの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 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」に配置した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

または、

Lib_Windows¥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	ライセンスファイル「ptalic.dat」が存在するフォルダパス	「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	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	カラープロファイル「sRGB2014.icc」「JapanColor2001Coated.icc」のフルパス	「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

※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。

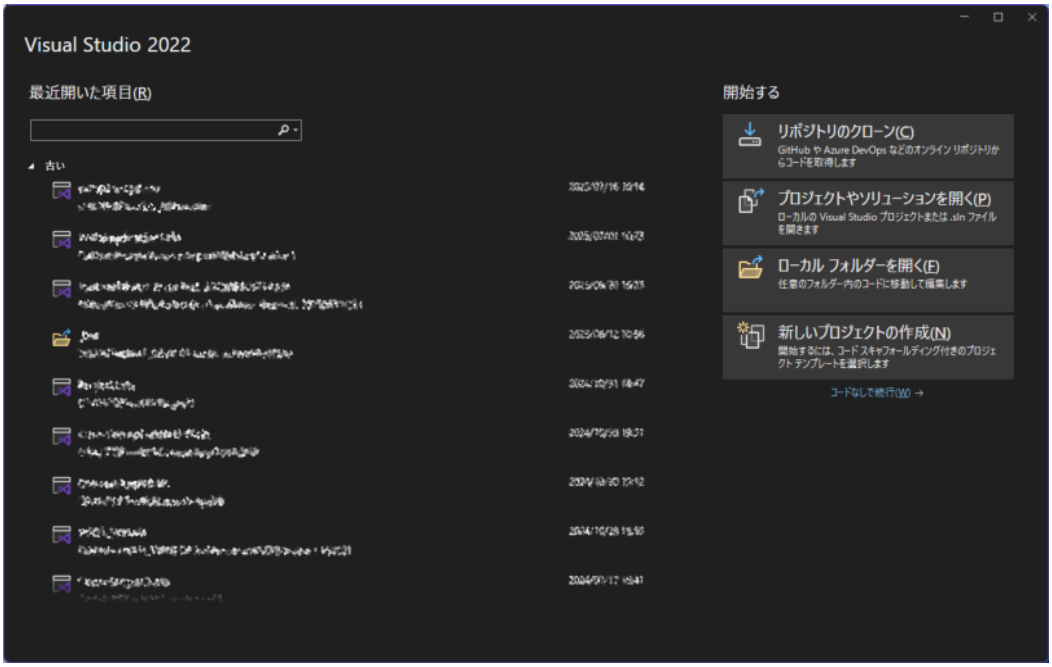
3.サンプルコードのプロジェクト作成とビルド方法

ここでは、サンプルコードをビルドするための Microsoft Visual Studio（Visual Studio）プロジェクトの作成とビルド手順について説明します。
.NET の 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。

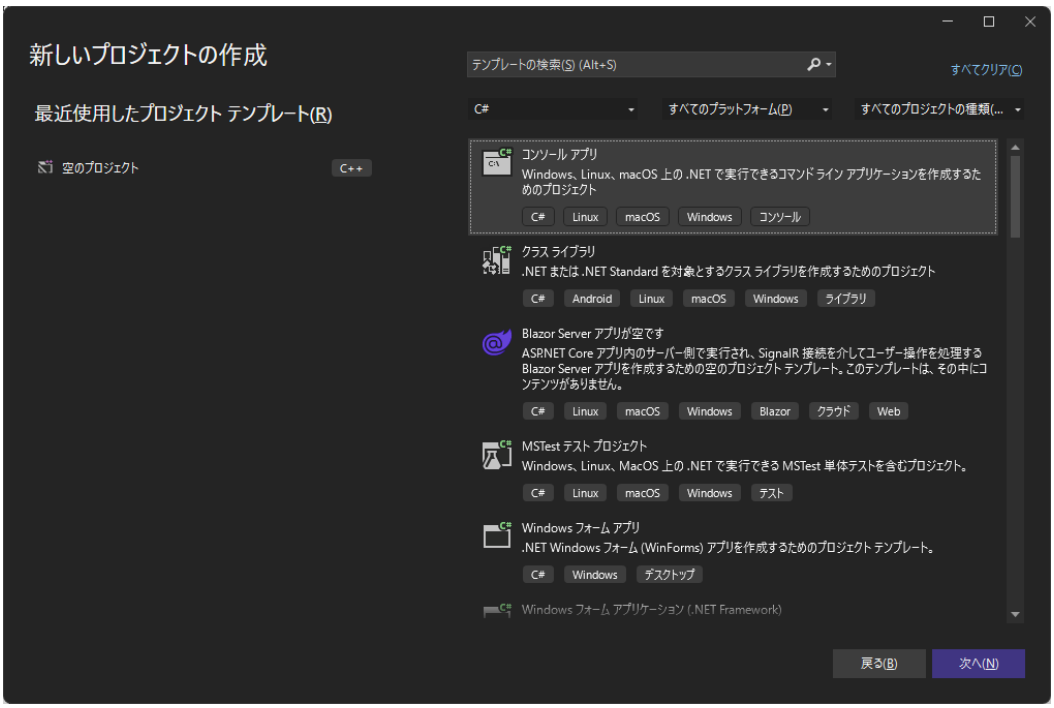
- Windows：{インストールフォルダ}¥samples¥dotnet
- Linux：製品版／評価版パッケージの「SampleCode/dotnet」ディレクトリ

3.1.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

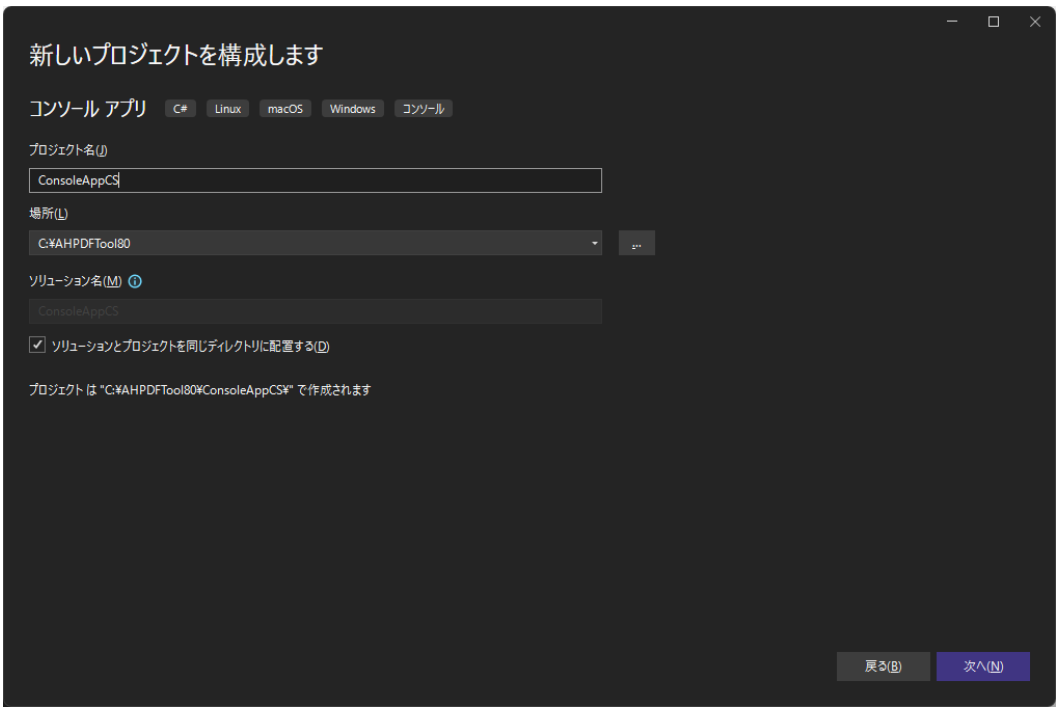
(1) Visual Studio 2022 を起動し、「新しいプロジェクトの作成」を選択します。



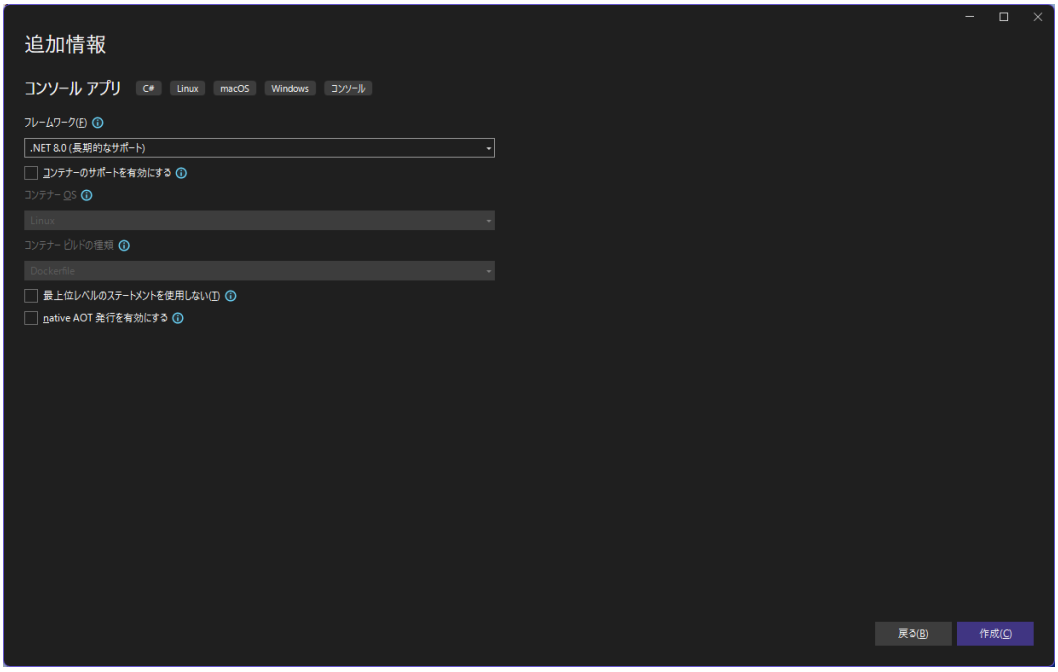
(2) 「新しいプロジェクトを作成」において、C#の「コンソールアプリ」を選択し「次へ」ボタンをクリックします。



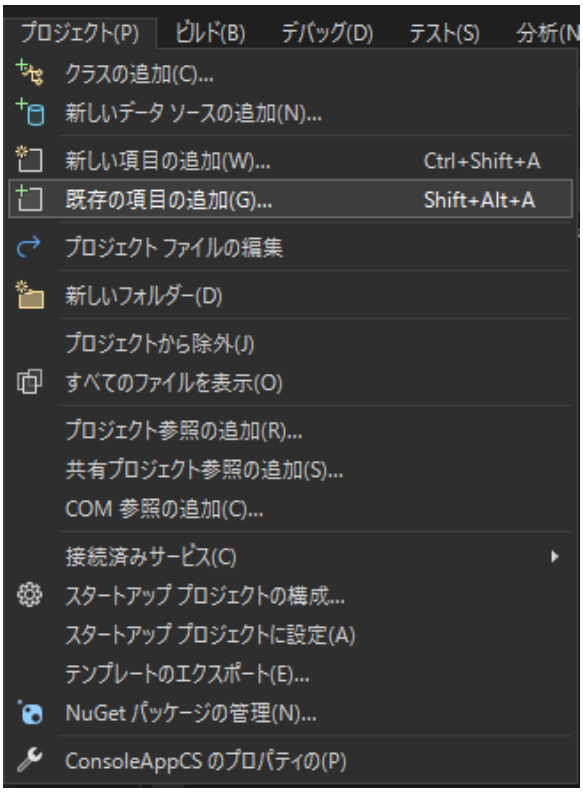
(3) 「プロジェクト名」、「場所」を設定し「次へ」ボタンをクリックします。



(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。

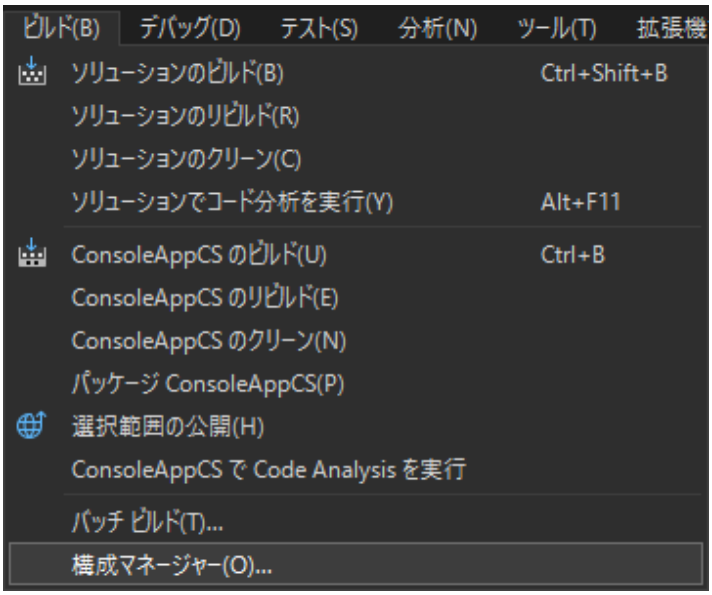


(5) 「プロジェクト」メニューの「既存の項目の追加...」を選択するとファイル選択ダイアログが表示されます。

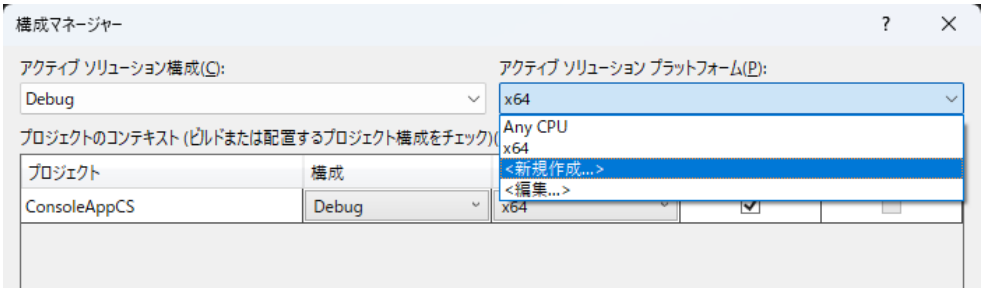


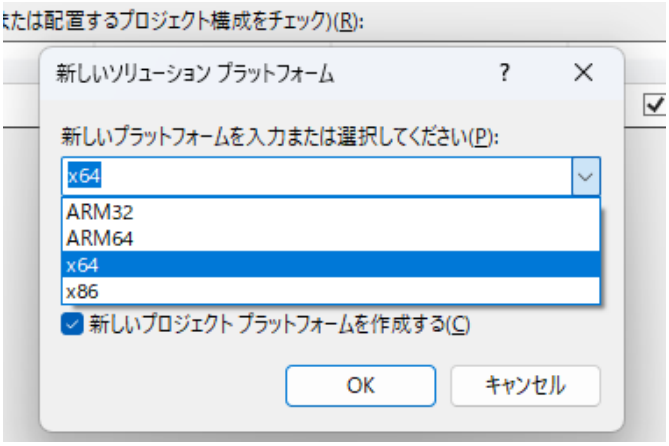
(6) サンプルの cs ファイルをひとつ選択します。C#用サンプルコードはプロジェクトのソースファイルとして追加されます。
プロジェクト生成時に作成される cs ファイルはプロジェクトから削除してください。

(7) 「ビルド」メニューの「構成マネージャー...」を選択します。

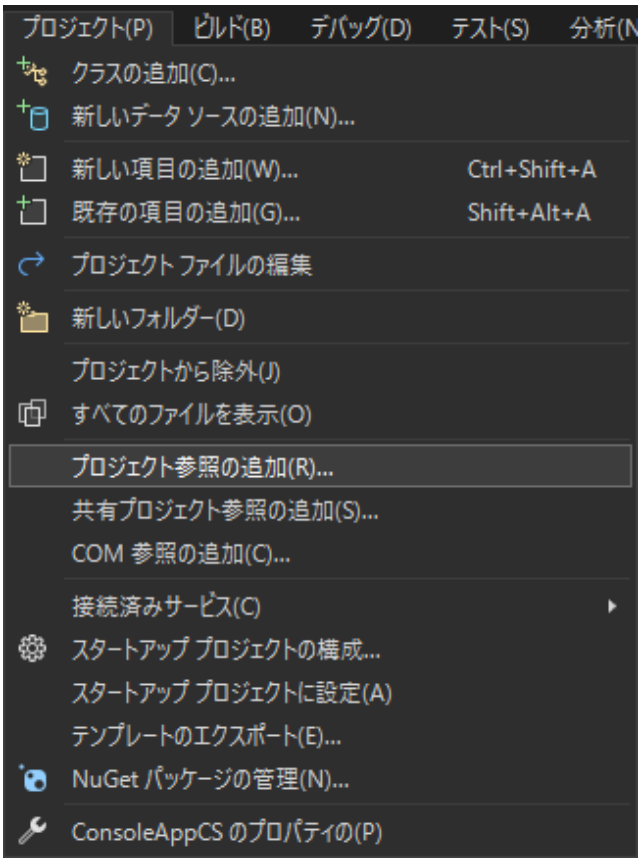


(8) 「アクティブソリューション構成」と「アクティブソリューションプラットフォーム」を設定します。プラットフォームは「x64」を選択して新規作成します。

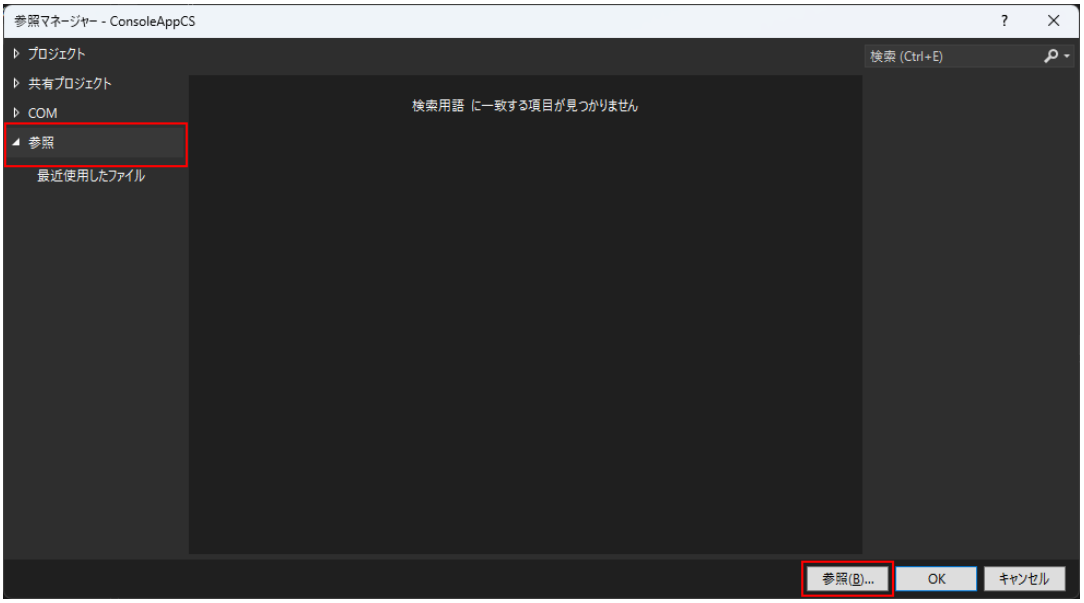




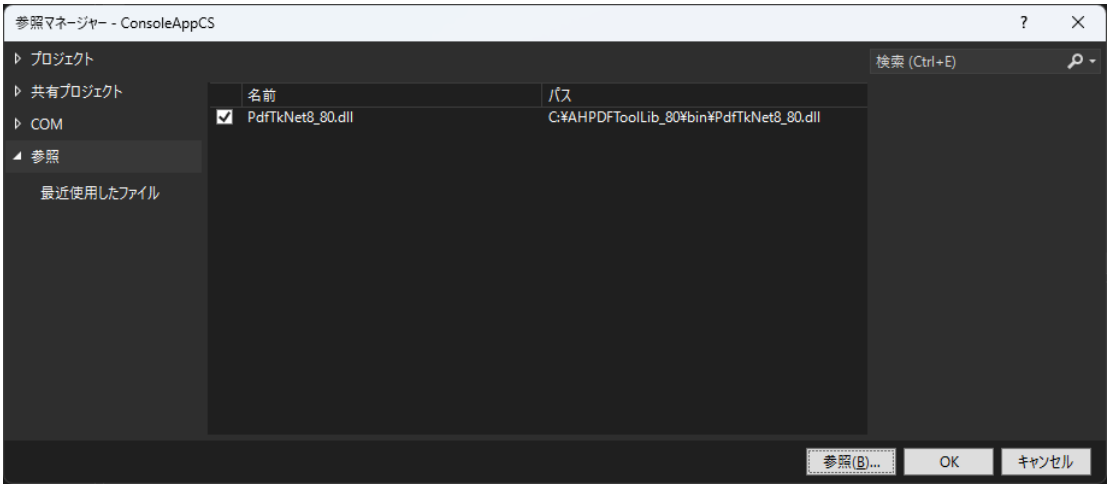
(9) 「プロジェクト」メニューの「プロジェクト参照の追加...」を選択すると、「参照マネージャー」ダイアログが開きます。



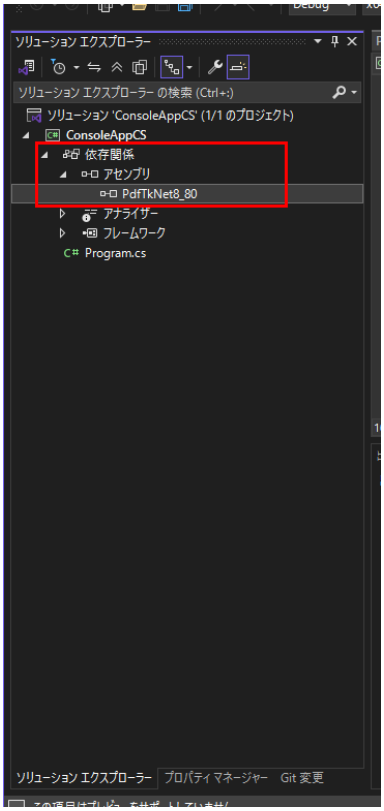
(10)「参照」タブを開き、ダイアログ右下の「参照...」ボタンをクリックします。



(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。



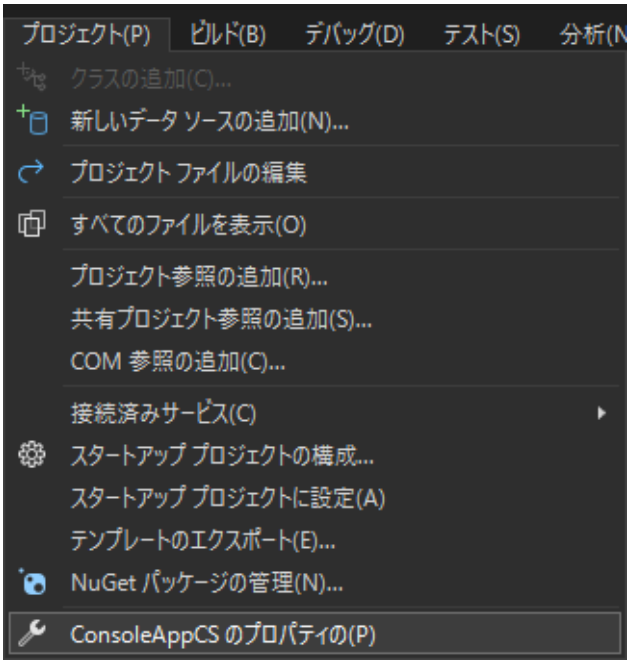
(12)参照した DLL は、ソリューションエクスプローラーの「依存関係」 - 「アセンブリ」で確認できます。



3.2. ビルドの設定とビルド

Windows 用のビルドを行います。

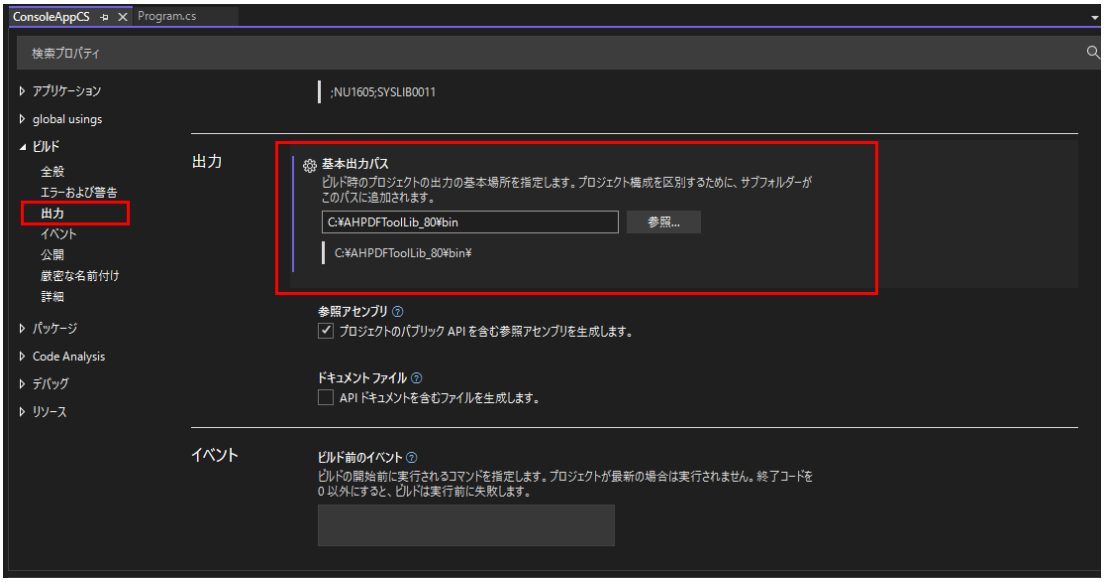
(1) 「プロジェクト」メニューの「(プロジェクト名) のプロパティ」を選択します。
(図のプロジェクト名は ConsoleAppCS です。)



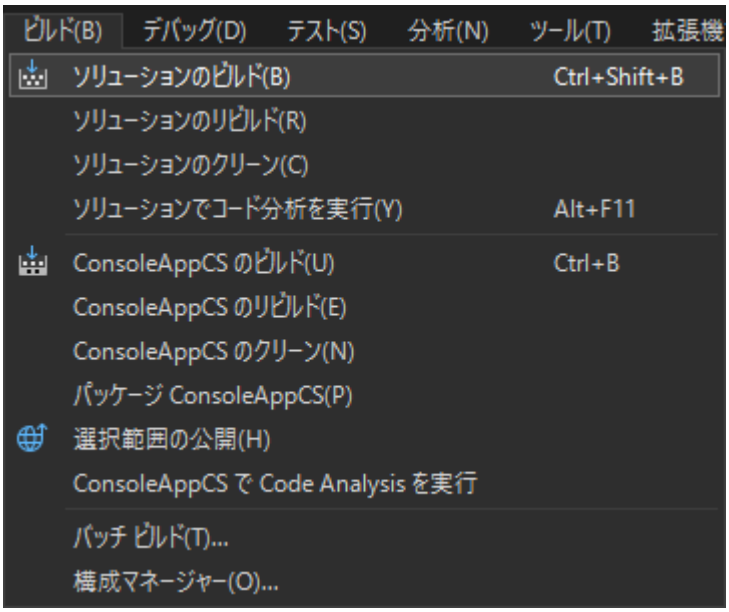
(2) 左側の「ビルド」タブを開き「出力」を選択します。

(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。

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。



(4) 「ビルド」メニューの「ソリューションのビルド」をクリックすると、ビルドが開始されます。

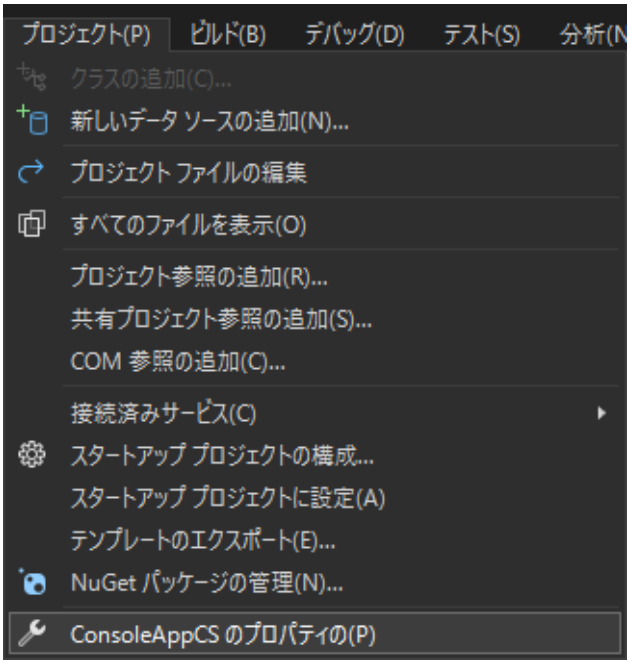


(5) ビルドされた exe を実行するには、exe とともに作成される、exe と同名の「.dll」と「.runtimeconfig.json」が必要です。

3.3. 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。

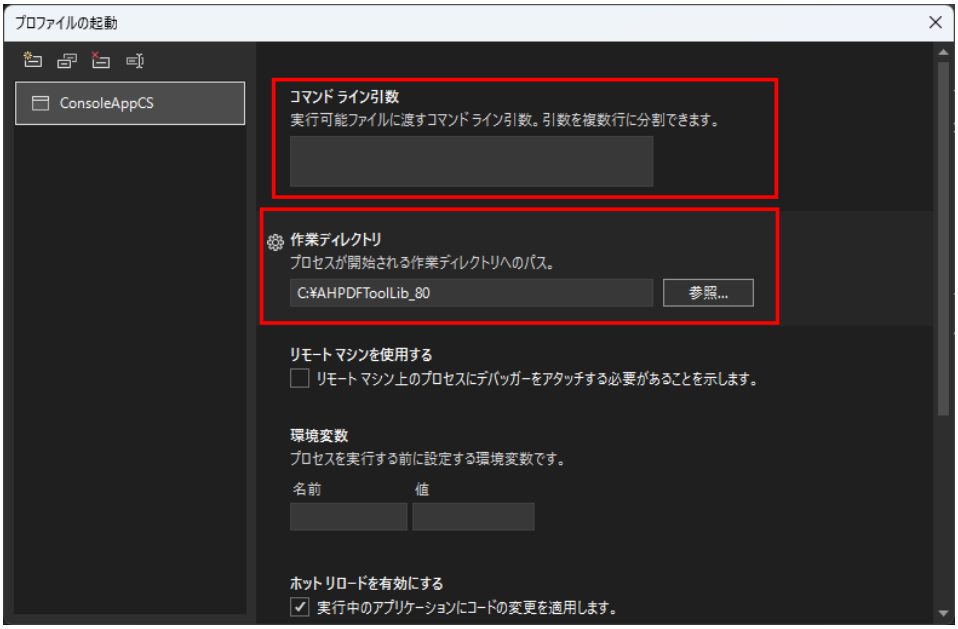
(1) 「プロジェクト」メニューの「(プロジェクト名) のプロパティ」を選択します。
(図のプロジェクト名は ConsoleAppCS です。)



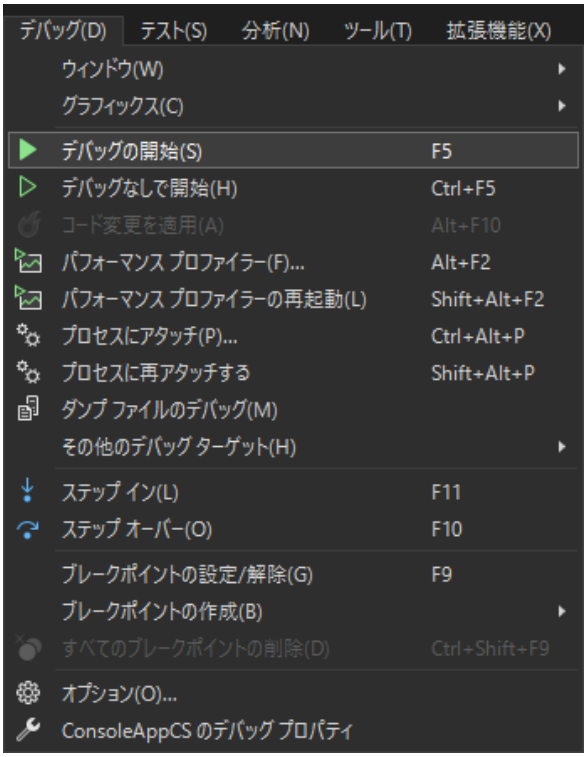
(2) 左側の「デバッグ」タグを開き「デバッグ起動プロファイル UI を開く」をクリックして「プロファイルの起動」ダイアログを開きます。

(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。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。
- 必要に応じて、「コマンドライン引数」を指定します。



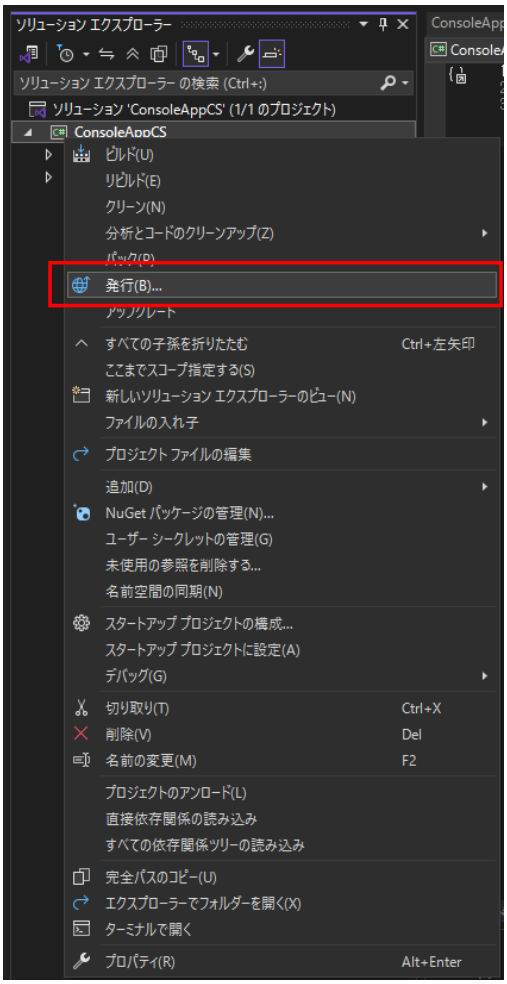
(4) 「デバッグ」メニューの「デバッグの開始」をクリックすると、デバッグ実行できます。


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

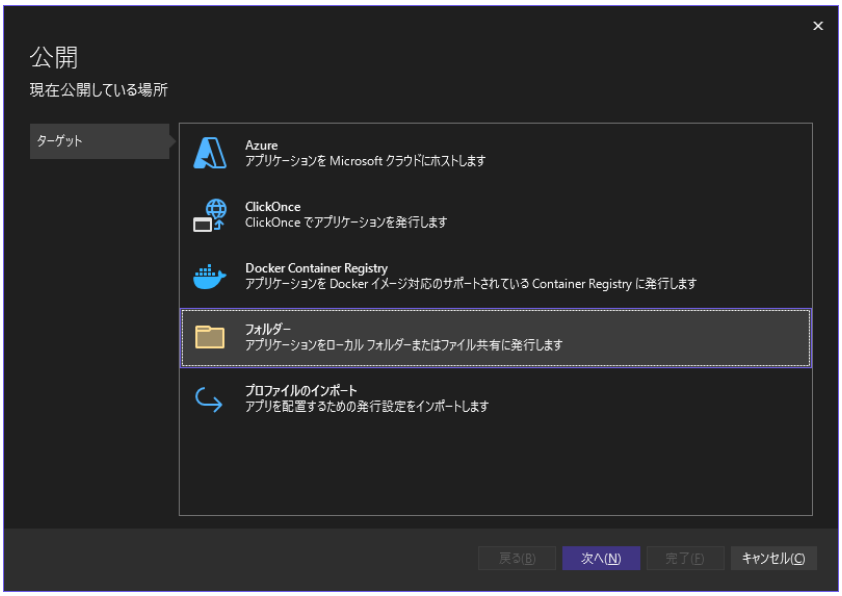
3.4.アプリケーションの発行

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。

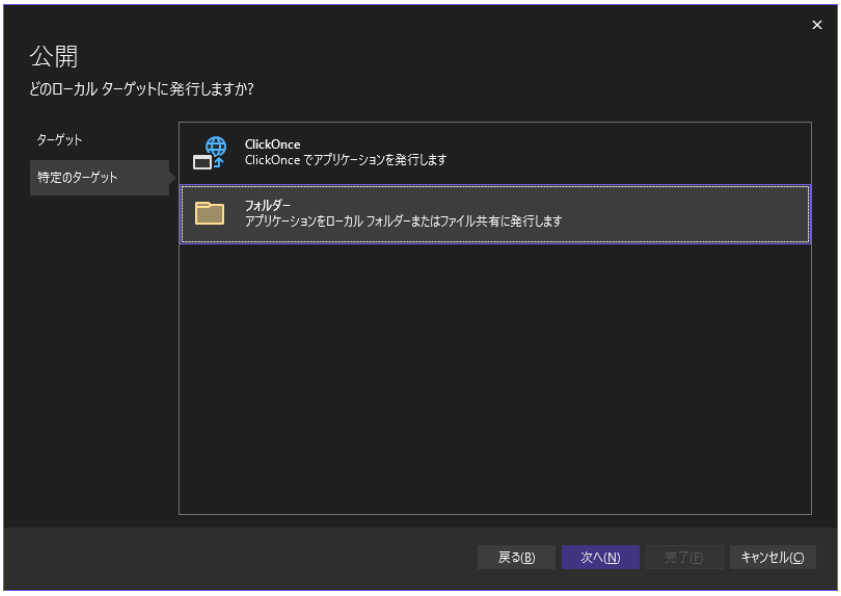
(1) プロジェクトで右クリックし開いたメニューで「発行...」を選択します。



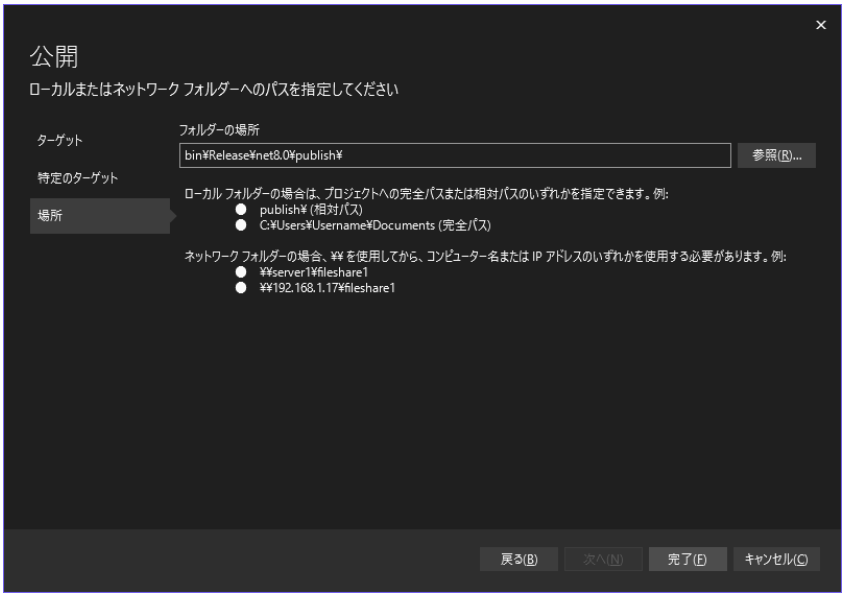
(2) 「公開」ダイアログが開くので公開先を指定します。ここでは、ローカルフォルダーにアプリケーションを作成しますので、「フォルダー」を選択し、右下の「次へ」ボタンをクリックします。



(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。

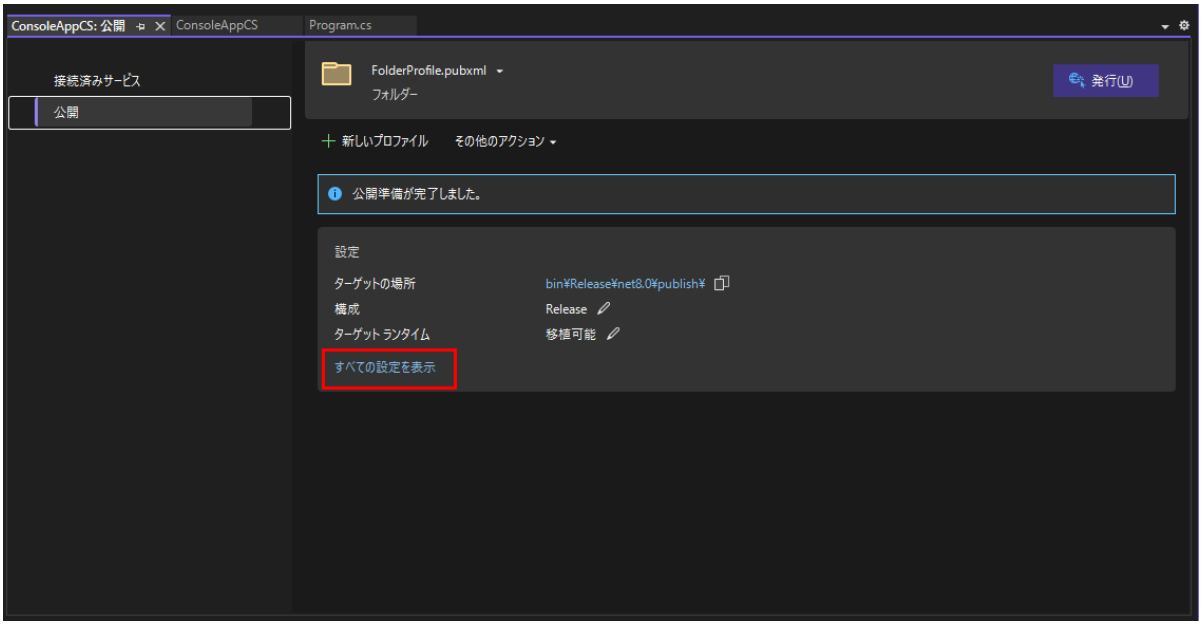


(4) 「フォルダーの場所」を指定し、「完了」を押します。



(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

配置モード：「自己完結」

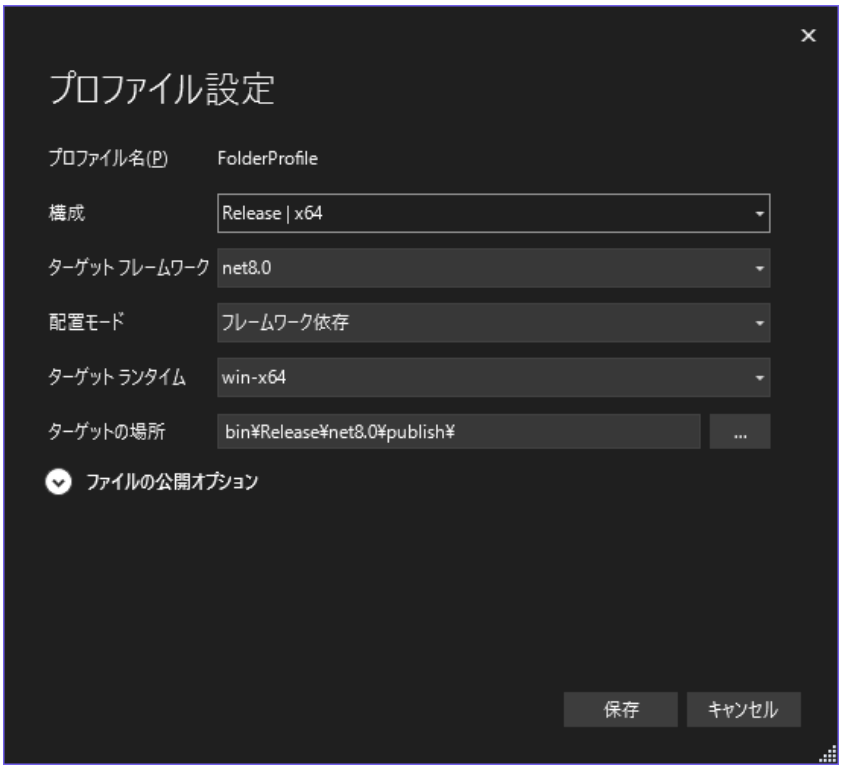
発行されるアプリケーションファイルには、ターゲットフレームワークのランタイムライブラリーが含まれます。このため、アプリケーションファイルを実行する環境に、ターゲットフレームワークの.NET ランタイムライブラリーをインストールする必要はありません。ただし、ファイルサイズは大きくなります。

配置モード：「フレームワーク依存」

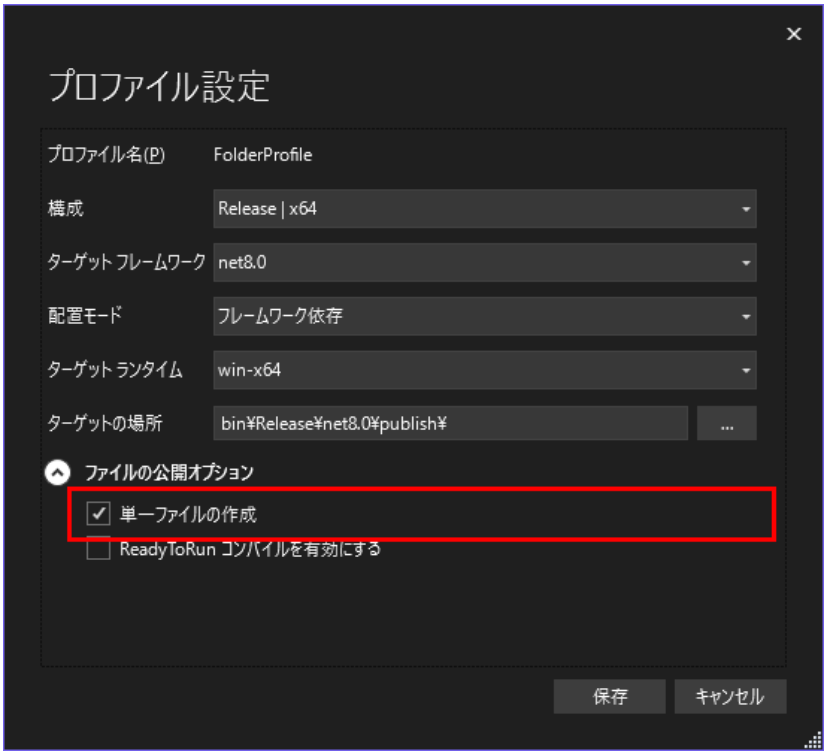
発行されるアプリケーションファイルには、ターゲットフレームワークのランタイムライブラリーは含まれません。アプリケーションファイルを実行する環境には、ターゲットフレームワークの.NET ランタイムライブラリーをインストールしてください。

ターゲットランタイム

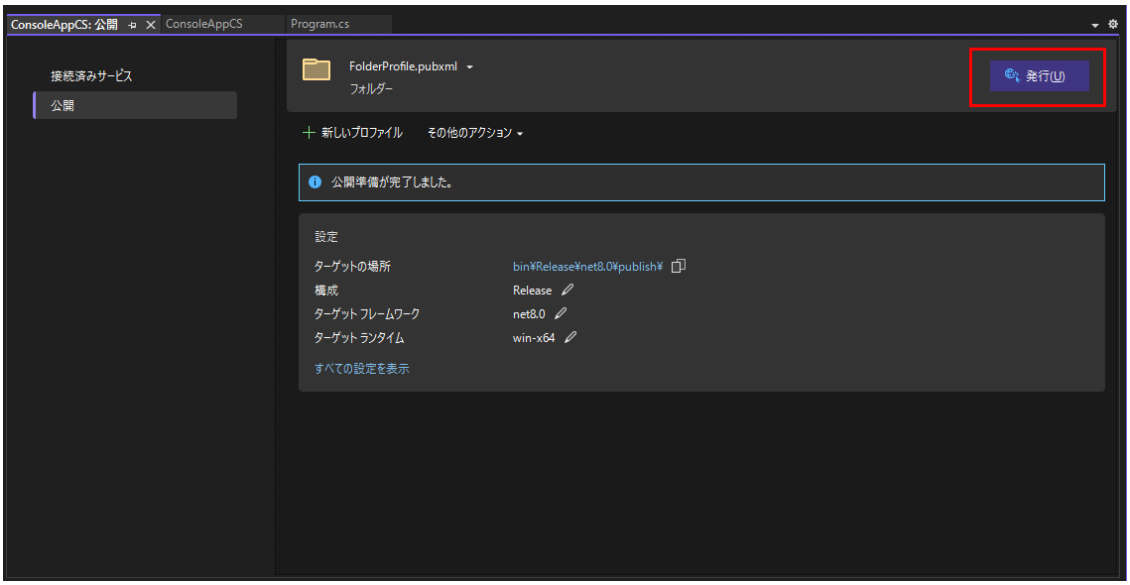
「win-x64　(Windows 64bit)」、「linux-x64 (Linux 64bit)」のいずれかを選択します。PDF Tool API は、arm、osx（Mac OS X）には対応していません。



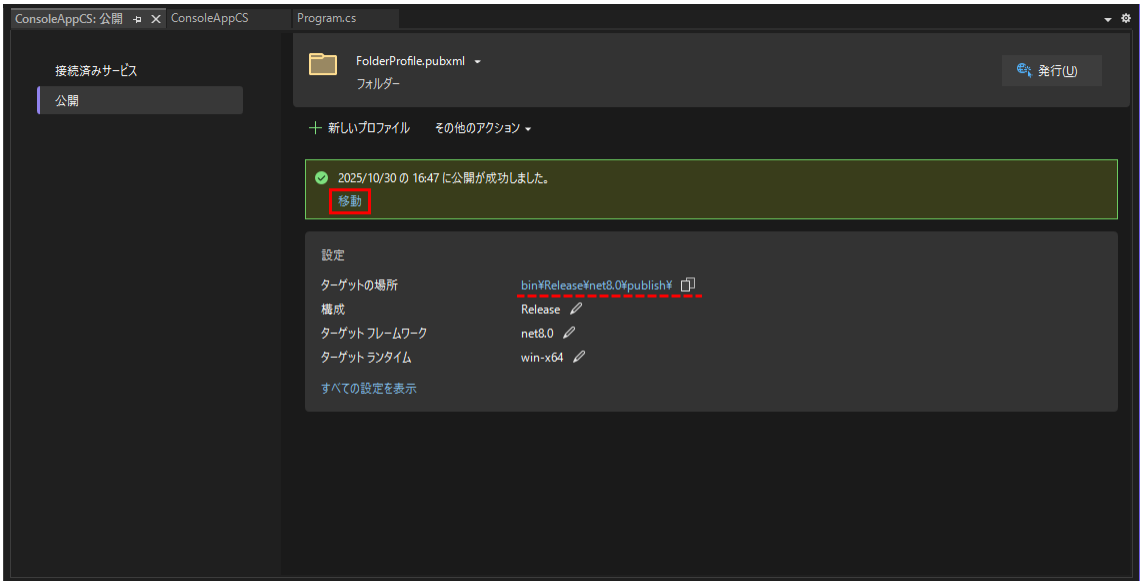
(8) 「ファイルの公開オプション」を開きます。「単一ファイルの作成」にチェックを入れます。



(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」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.発行されたファイルの実行方法について

ここでは、『3.4 アプリケーションの発行』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。

4.1.Windows 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) 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) 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
『Windows 開発環境における環境変数のまとめ』をご参照ください。

※4
『フォントの準備』をご参照ください。

4.2.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、フォント構築ファイル、ライセンス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
『**エラー! 参照元が見つかりません。**』参照

※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

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 のパス}

※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

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るため、「dotnet」コマンドは不要です。

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」コマンド付けて実行ます。

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}
```

4.2.1. 付録 1. Linux 環境への.NET8 ランタイムのインストール方法

Linux 環境への.NET のインストール方法は Linux ディストリビューションにより異なります。
本マニュアルでは例として、「Red Hat Enterprise Linux (RHEL)」におけるインストール方法を解説します。
それ以外のディストリビューションにおけるインストールは以下の公式リファレンスをご参照ください。
https://learn.microsoft.com/ja-jp/dotnet/core/install/linux

(1) コンソールを開き、以下のコマンドを実行します。

```
sudo dnf install dotnet-runtime-8.0
```

これで.NET 8 の ASP.NET Core ランタイムがインストールされます。

(2) インストール先を指定しなかった場合、ホームディレクトリ「($HOME)/.dotnet」にインストールされます。「.dotnet」ディレクトリが表示されない場合、「隠しファイルを表示する」にチェックを入れてください。

(3) インストールした.NET8 ランタイムの「dotnet」コマンドを使用するには、次の環境変数の設定が必要です。
「ホーム」の.bashrc（または.bash_profile）に以下の環境変数を追記し、システムを再起動します。

```
export DOTNET_ROOT=$HOME/.dotnet

export PATH=$PATH:$DOTNET_ROOT:$DOTNET_ROOT/tools
```

(4) インストールしたランタイムのバージョンは以下のコマンドとオプションで確認できます。

```
dotnet --list-runtimes
```

実行した場合、以下のように表示されます。

```
Microsoft.AspNetCore.App 8.0.21 [/usr/lib64/dotnet/shared/Microsoft.AspNetCore.App]

Microsoft.NETCore.App 8.0.21 [/usr/lib64/dotnet/shared/Microsoft.NETCore.App]
```

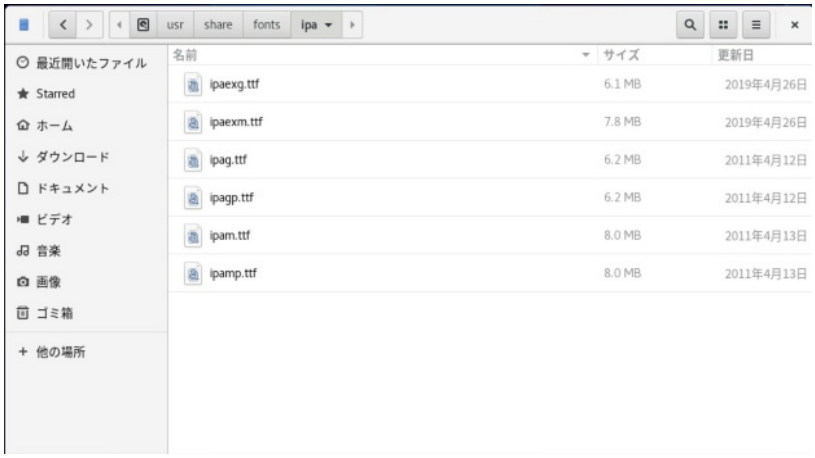
4.2.2. 付録 2. IPA フォントのインストール方法

Linux では、動作環境によって日本語用のフォントが存在しない場合があります。ここでは、「IPA フォント」のインストール方法を説明します。

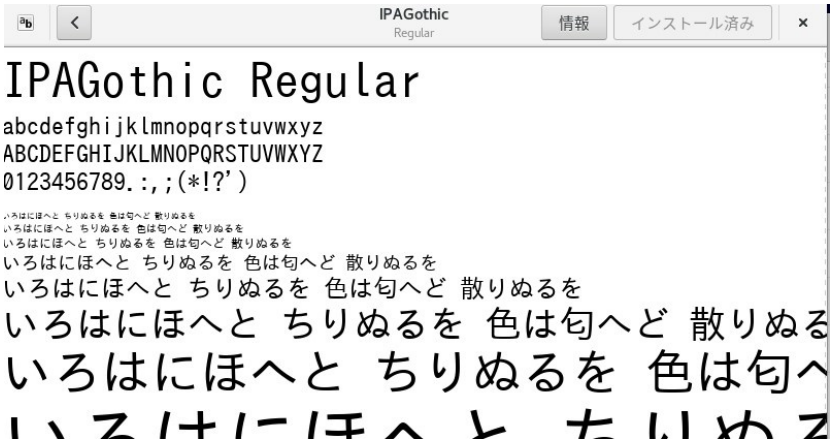
(1) 下記のサイトからフォントファイルをダウンロードします。

文字情報技術促進協議会：https://moji.or.jp/ipafont/

(2) ダウンロードした ttf ファイルを任意のディレクトリに配置します。ここでは、「/usr/share/fonts/ ipa」に配置します。



(3) ttf ファイルを開き、右上の「インストール」ボタンをクリックしてインストールします。



5.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)を使用して、.NET8 のプログラム作成と実行方法を説明します。

5.1.Linux 環境

5.1.1. .NET SDK のインストール

(1) 『4.2.1 付録 1. Linux 環境への.NET8 ランタイムのインストール方法』の(1)のインストールコマンドを

```
sudo dnf install dotnet-sdk-8.0
```

に変更して実行します。
これは.NET8 をランタイムとしてではなく、SDK としてインストールするコマンドです。

(2) インストールした SDK のバージョンは以下のコマンドとオプションで確認できます。

```
dotnet --list-sdks
```

これを実行すると、

```
「8.0.21 [/home/test/.dotnet/sdk]」
```

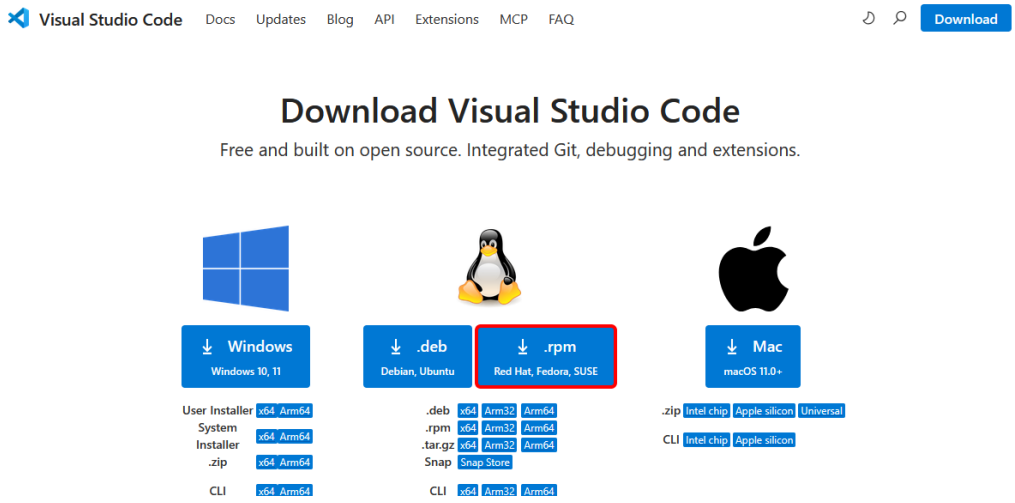
のように SDK バージョンが表示されます。

5.1.2. Visual Studio Code のインストール

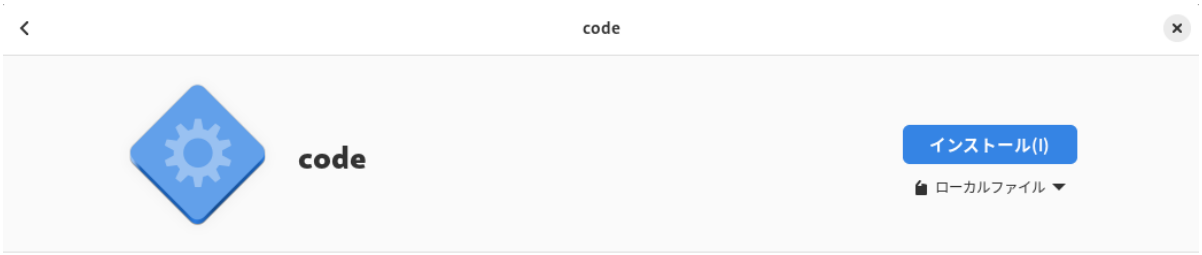
(1) Download Visual Studio Code

https://code.visualstudio.com/download

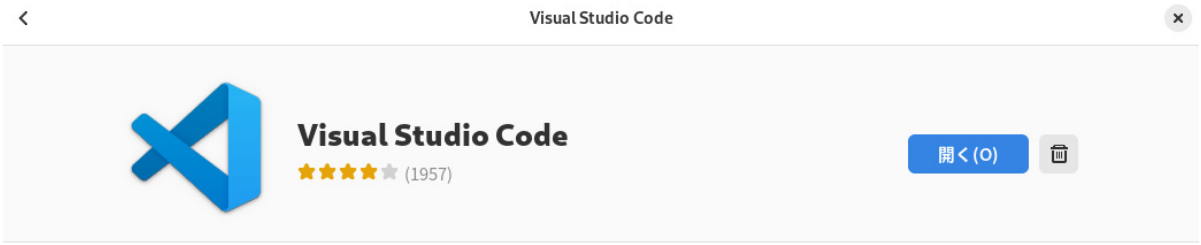
を開いて「.rpm」をダウンロードします。

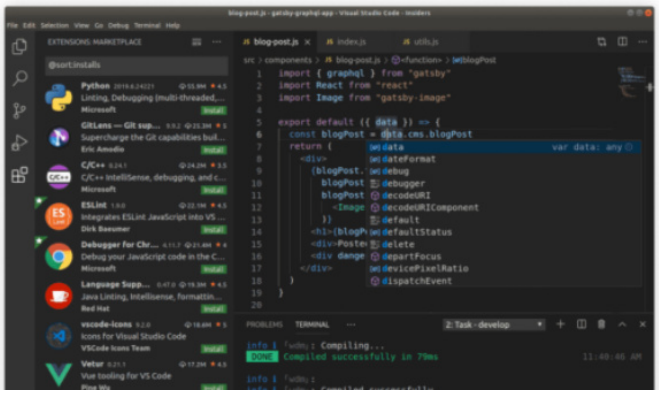


(2) ダウンロードした rpm ファイルをクリックして開きます。
左上の「インストール」ボタンを押すとインストールが始まります。



インストールが完了した後、Visual Studio Code を開きます。

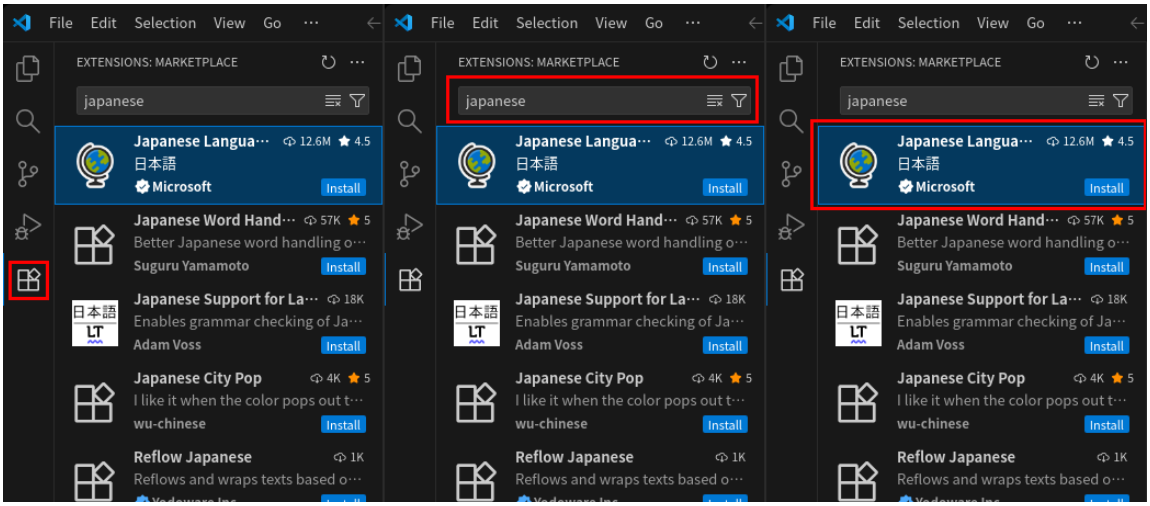




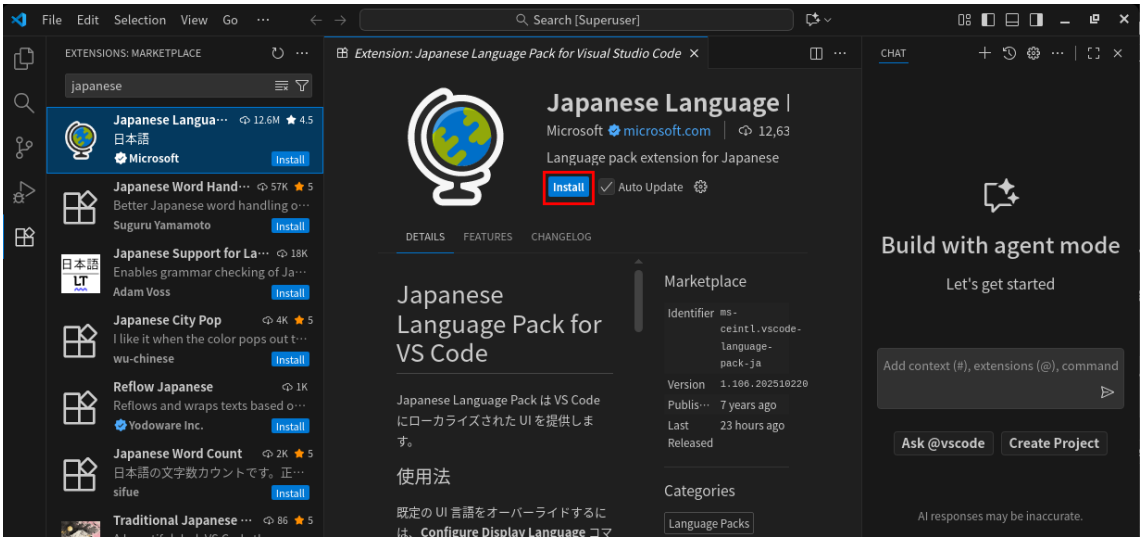
(3) VSCode 拡張機能「Japanese Language Pack for VS Code」のインストール

以下の手順で VSCode の表示を日本語化します。

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「Japanese」と記入
3. 結果に表れる「Japanese Language Pack for VS Code」を選択



4. 青色の「install」ボタンを押す

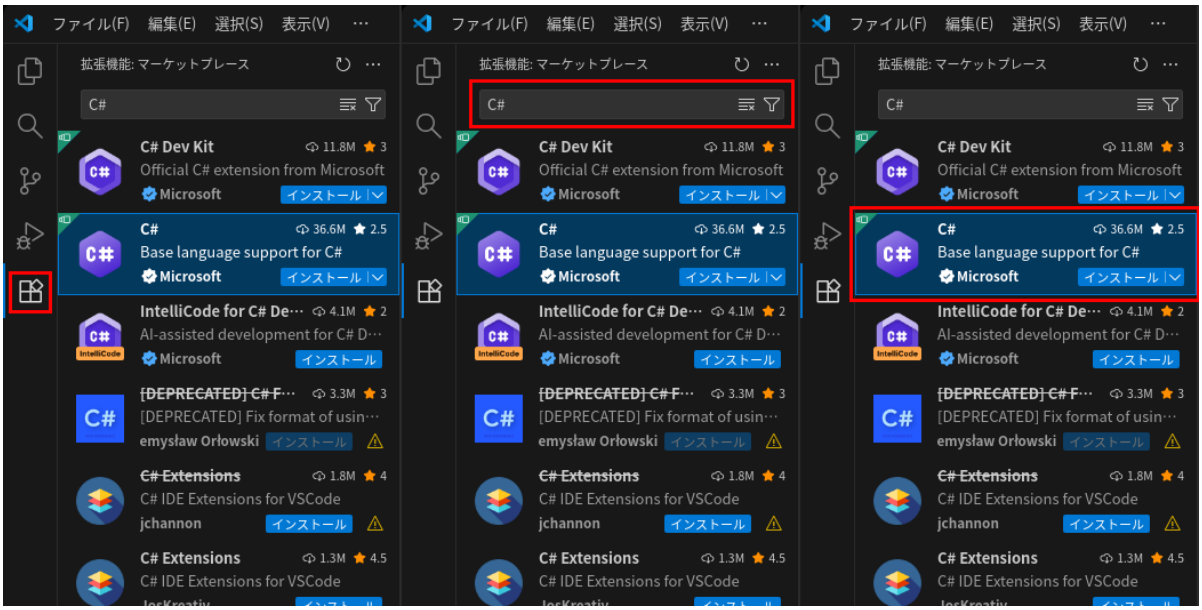


5. インストール終了後に VSCode の再起動をする

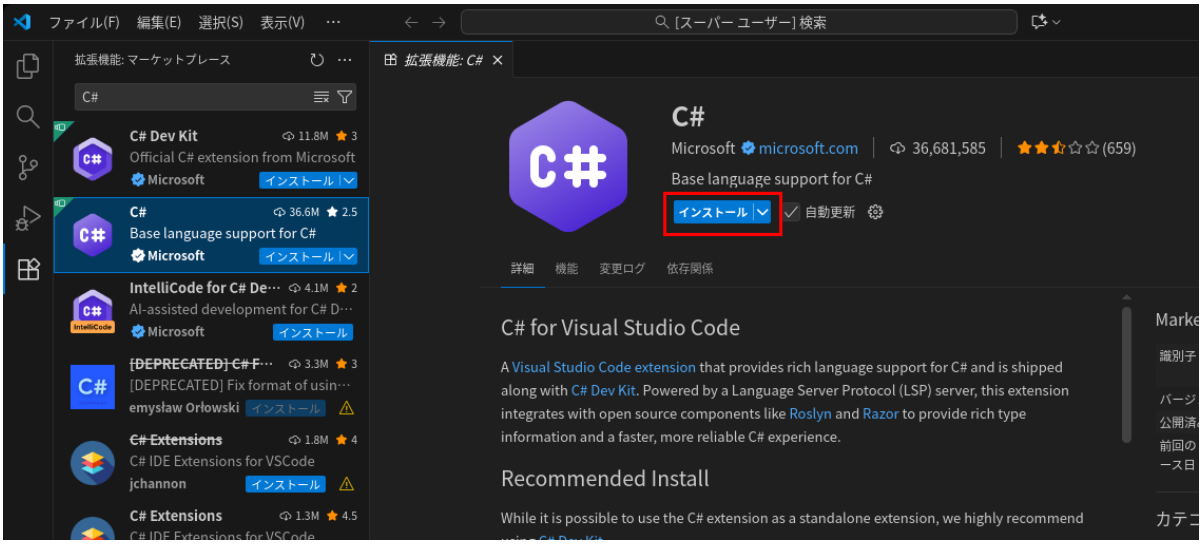
再起動後は VSCode の表示が日本語になっています。

(4) VSCode 拡張機能「C#」のインストール

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。

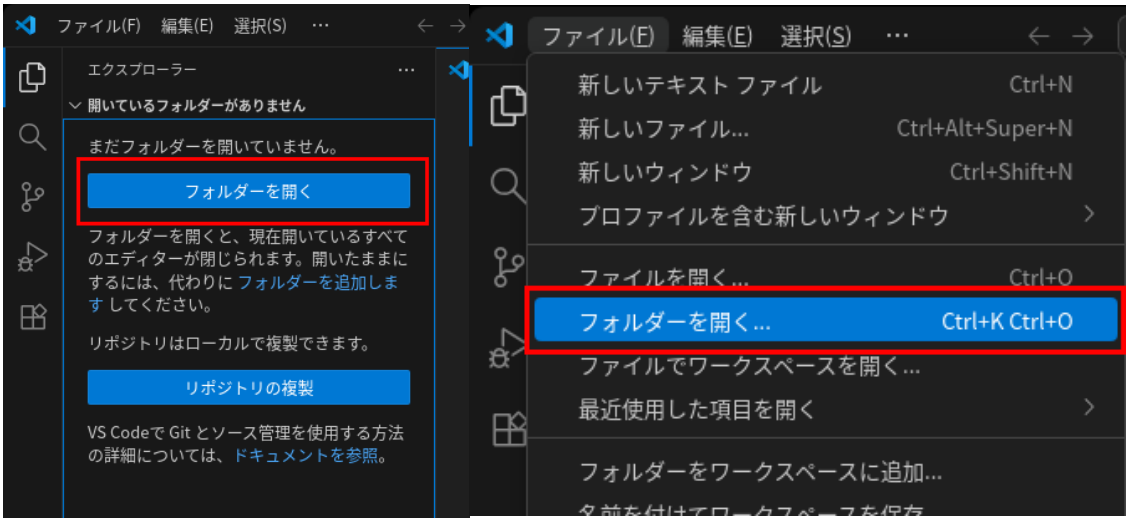
5.1.3. プロジェクトの作成

「フォルダを開く」と「ターミナル経由で作成する」の２つの手順を踏む必要があります。

5.1.3.1. フォルダを開く

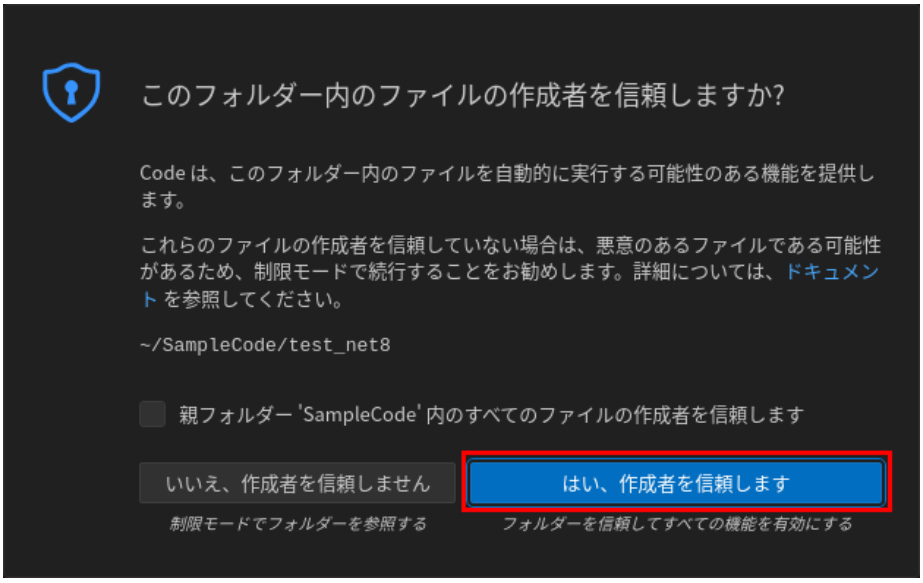
(1) 左側アクティビティバーにあるエクスプローラーを選択します。

(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。

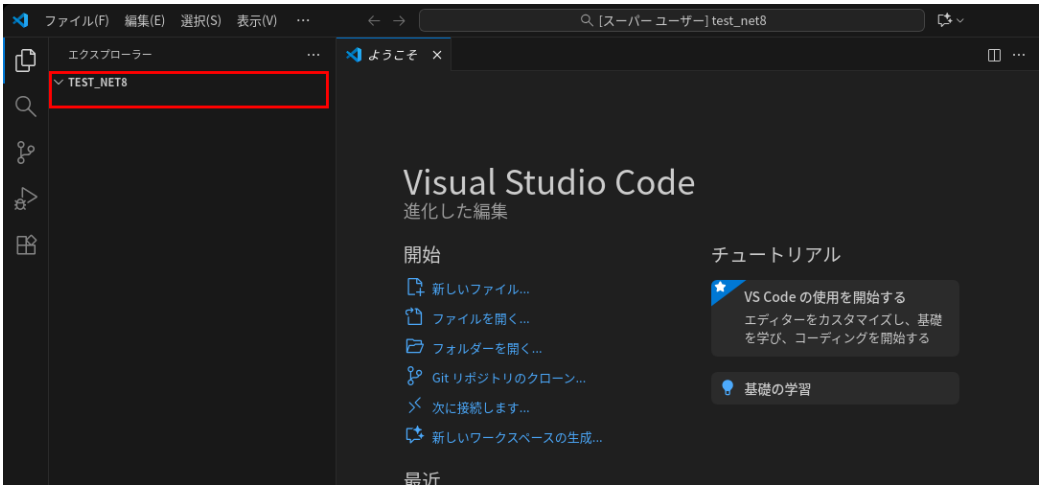


(3) プロジェクトのファイルを生成するフォルダを指定して右上の「開く」をクリックします。
（画像では test_net8 というフォルダを指定しています）

(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。

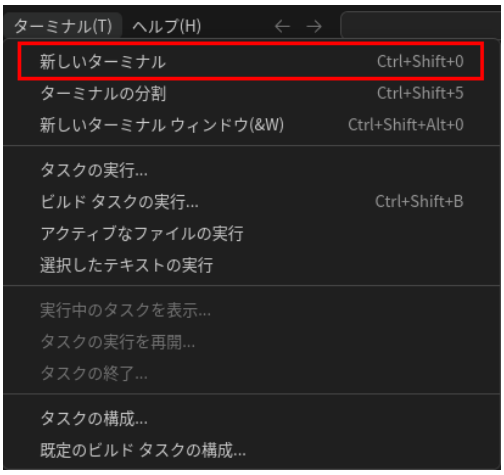


(5) 左側のサイドバーのエクスプローラーに指定したファイル名が表示されます。

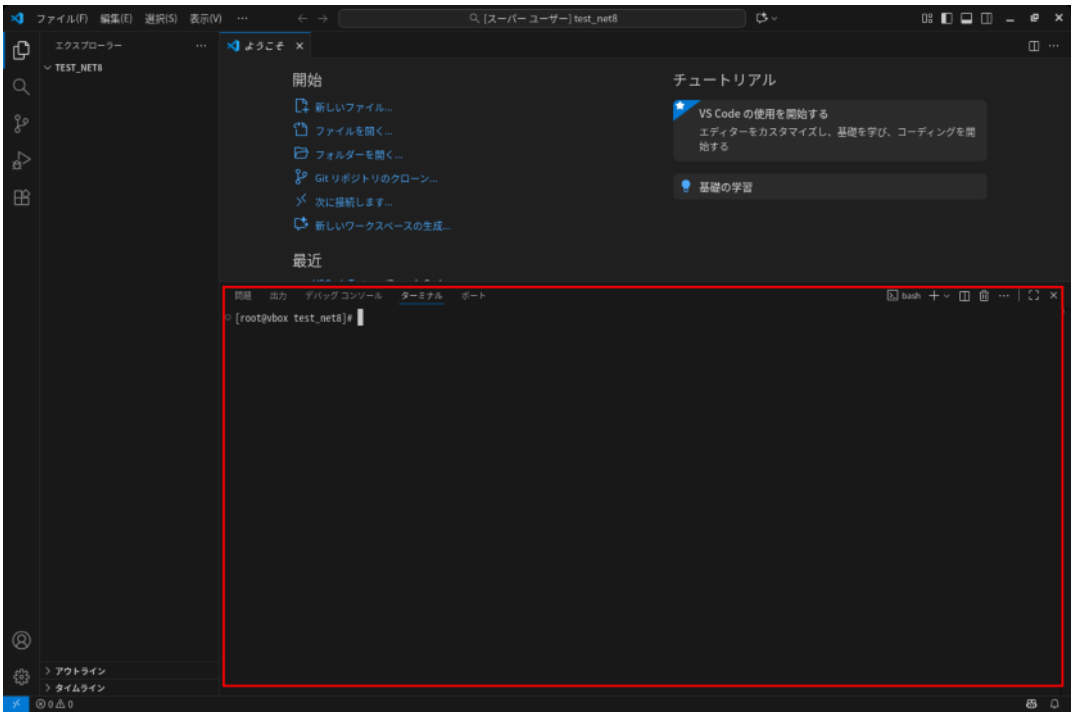


5.1.3.2.　ターミナルを開く

(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 ）

5.1.4. 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。

「5.1.3 プロジェクトの作成」をご参照ください。

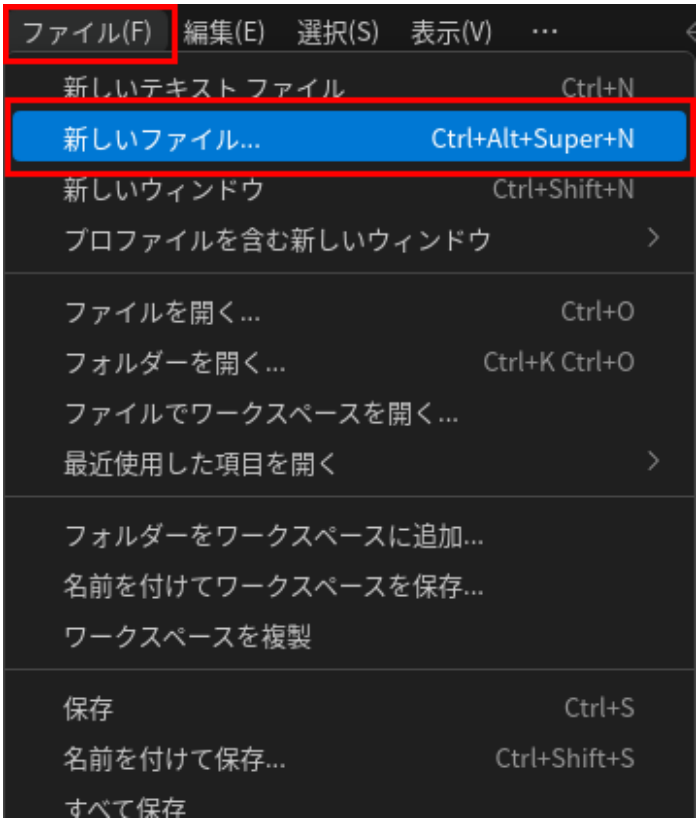
(3) 「(フォルダ名) .csproj」 を開いて以下の赤字箇所を追記します。

<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。

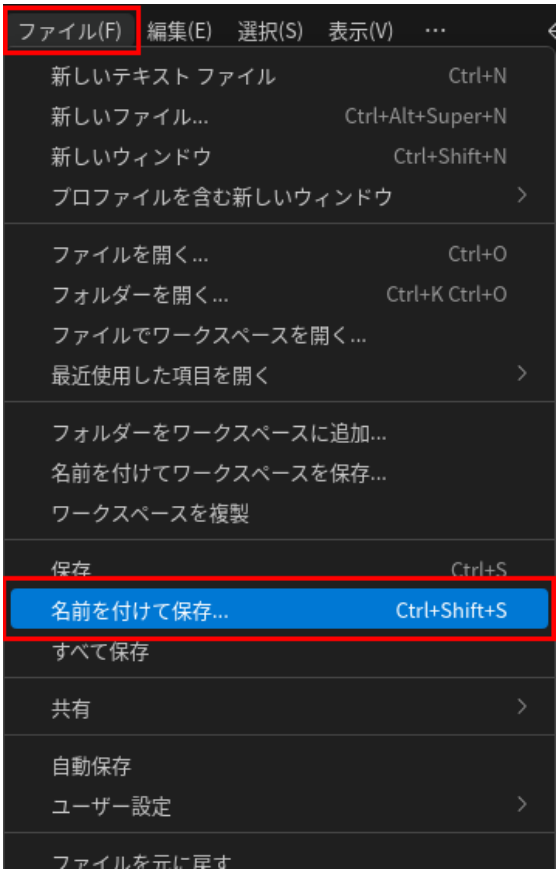
```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net6.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Platform>x64</Platform>
    <Nullable>enable</Nullable>
  </PropertyGroup>

  <ItemGroup>
    <Reference Include="PdfTkNet8_80">
      <HintPath>/usr/AHPDFToolLib80/lib/PdfTkNet8_80.dll</HintPath>
    </Reference>
  </ItemGroup>

</Project>
```

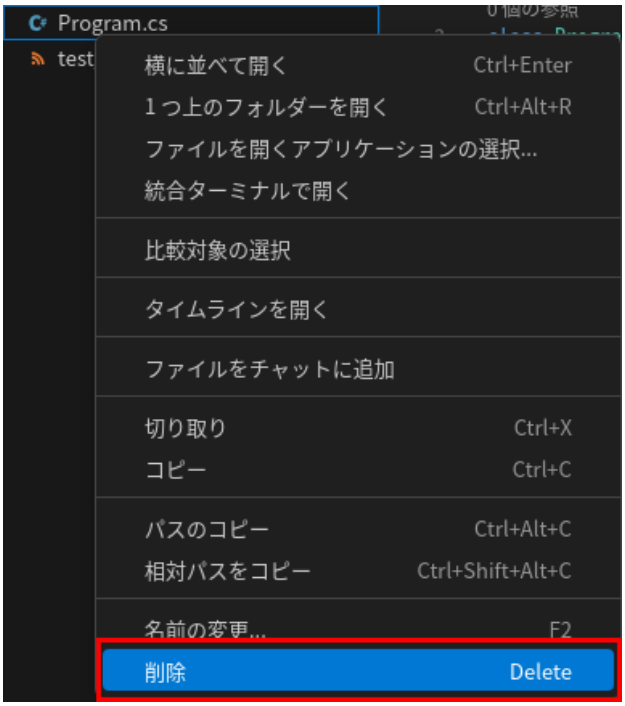
(4) 「ファイル」メニューの「ファイルを開く」を選択して使用したいサンプルコードの cs ファイルを開きます。



「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、ターミナルに下記の dotnet run コマンドを入力し、実行します。

```
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。

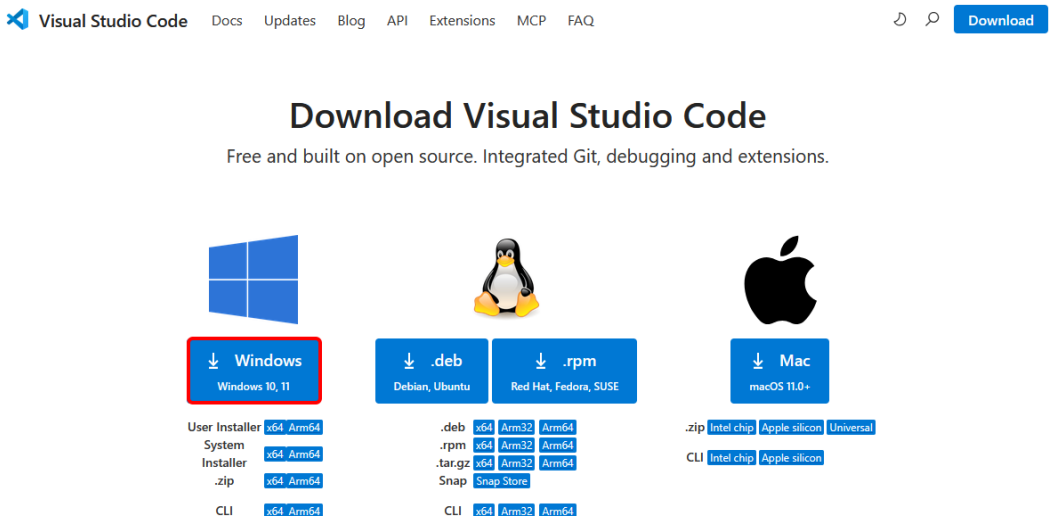
5.2.Windows 環境

5.2.1. NET SDK のインストール

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

5.2.2. Visual Studio Code のインストール

(1) Download Visual Studio Code
https://code.visualstudio.com/download
を開いて「windows」をダウンロードします。



(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。

(3) VSCode の拡張機能「Japanese Language Pack for VS Code」と「C#」をインストールします。「5.1.2 Visual Studio Code のインストール」の(3)、(4)をご参照ください。

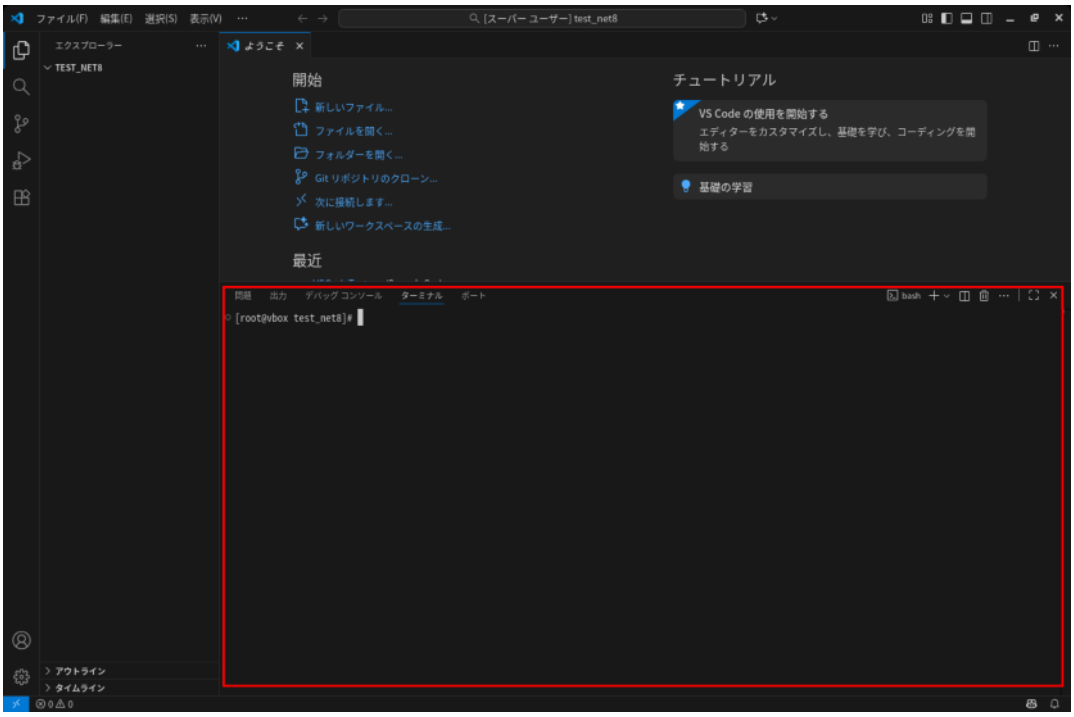
5.2.3. プロジェクトの作成

「5.1.3 プロジェクトの作成」をご参照ください。

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

(3)　「(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。



(4) 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 ）

PDF Tool API のサンプルコードを使用したプログラムの作成」をご参照ください。

履歴

日付	更新内容
2025.10.30	・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(.NET8)

2025.10.30