Antenna House PDF Tool API V7.0 サンプルコードのビルド

と実行手順 (.NET6)

アンテナハウス株式会社

目次

Antenna Hous	se PDF Tool API V7.0 サンプルコードのビルドと実行手順 (.NET6)	1
1. はじめに		3
2. PDF Too	API の開発環境を整える	4
2.1. ライ	イブラリファイル、ヘッダーファイルを配置する	4
2.1.1.	インストーラを利用する場合	4
2.1.2.	インストーラを利用しない場合	4
2.2. ライ	イセンスファイルを配置する	5
2.2.1.	インストーラを利用した場合	5
2.2.2.	インストーラを利用しなかった場合	5
2.3. フォ	- ントの準備	6
2.3.1.	フォントの場所	6
2.3.2.	フォント構築ファイルの作成	6
2.4. Win	dows 開発環境における環境変数のまとめ	7
3. サンプル	マコードのプロジェクト作成とビルド方法	8
3.1. プロ	コ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	8
3.2. ビル	~ドの設定とビルド	15
3.3. ディ	ヾッグビルドの設定とデバッグ実行	17
3.4. アフ	プリケーションの発行	19
4. 発行され	たファイルの実行方法について	25
4.1. Win	dows での実行について	25
4.2. Linu	JX での実行について	26
付録 1N	IET6 ランタイムのインストール方法	27
付録 2.IP	A フォントのインストール方法	29
5. Visual St	udio Code でのプログラム作成と実行方法	
5.1. Linu	」x 環境	
5.1.1.	NET SDK のインストール	
5.1.2.	Visual Studio Code のインストール	
5.1.3.	プロジェクトの作成	35
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	
5.2. Win	dows 環境	43
5.2.1.	NET SDK のインストール	43
5.2.2.	Visual Studio Code のインストール	43
5.2.3.	プロジェクトの作成	
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	

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。

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. インストーラを利用する場合

- (1) 「Setup-Windows¥setup-lib.exe」をダブルクリックするなどして起動します。
- (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++ 2019 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}:¥Program Files¥Antenna House¥PDFToolAPI_V7_lib

2.1.2. インストーラを利用しない場合

「Lib-Windows」フォルダにライブラリファイルやヘッダファイルなどがあります。開発環境の 任意の場所に配置します。 2.2. ライセンスファイルを配置する

2.2.1. インストーラを利用した場合

インストーラにより、ライセンスファイル「ptalic.dat」は「{インストールフォルダ}¥License」フ ォルダに配置され、環境変数「PTL70_LIC_PATH」に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 日間有効の評価用です。出力さ れる PDF ファイルに透かし文字列が挿入されます。

2.2.2. インストーラを利用しなかった場合

- ◆ 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「PTL70_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

ライセンスファイルを、PDF Tool API のモジュールファイル「PdfTk70.dll」と同じ場所に配置 します。

この場合、環境変数「PTL70_LIC_PATH」の作成は必要ありません。

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)

<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 [PTL70_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
PTL70_LIC_PATH	ライセンスファイル	「PdfTk70.dll」とは異なる場所
	「ptalic.dat」が存在する	にライセンスファイルを配置
	フォルダパス	する場合
PTL70_FONT_CONFIGFILE	フォント構築ファイル	システムのフォントフォルダ
	「font-config.xml」のフルパ	とは異なる場所にあるフォン
	ス	トを参照する場合

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、製品版/評価版パッケージの「SampleCode/dotnet」ディレクトリにあります。
- 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 を起動します。

- م	₽ ;	リポジトリのクローン(C) GitHub や Azure DevOps などのオンライン リポジトリ からコードを取得します
	E.	
	Ŀ	プロジェクトやソリューションを開く(P) ローカルの Visual Studio プロジェクトまたは .sin ファイル を聞きます
	F	ローカル フォルダーを開く(F) 任意のフォルダー内のコードに移動して編集します
		新しいプロジェクトの作成(N) 開始するには、コードスキャフォールディング付きのプロ ジェクトテンプレートを選択します
		コードなしで統行(W) →

(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。

新しいプロジェクトを構成します	-		×
コンソール アプリ C# Linux macOS Windows コンソール			
プロジェクト名(/)			
ConsoleApp1			
場所(L)			
レーション名(M) ①			
ConsoleApp1			
□ ソリューションとプロジェクトを同じディレクトリに配置する(D)			
プロジェクトは * ¥ConsoleApp1¥ConsoleApp1¥* で作成されま す			
戻る(8)	3	たへ(N)	

(4) 「フレームワーク」に「.NET 6.0 (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。

追加情報		-		×
コンソール アプリ C# Linux macOS Windows コンソール				
フレームワーク(F) 🚯				
.NET 6.0 (長期的なサポート)				
▲ 上位レベルのステートメントを使用しない(1) ①				
	戻る(B)	作	咸(C)	

(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。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。

DO	77	イル(F)	編集(E)	表示(V)	Git(G)	プロジェクト(P)	ビルド(B)
	拡		ソリューションの	ビルド(B)		Ctrl+	Shift+B
× e) -		ソリューションの	リビルド(R)			
			ソリューションの	ウリーン(C)			
₩-/	sam		ソリューションで	コード分析を	宝行(Y)	Alt+	F11
Т Н	C# (4	ConsoleApp	1 のビルド(U)	Ctrl+	В
7 77			ConsoleApp	1 のリビルド	(E)		
Ĵ□			ConsoleApp	1 のクリーン(N)		
-Ū			パッケージ Cor	nsoleApp1((P)		
<u> </u>		¢	選択範囲の公	<開(H)			
ルボッ			ConsoleApp	1でCode A	Analysis を	実行	
カス			バッチ ビルド(T)			
			構成マネージ†	?−(O)			

(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、「x86」または「x64」のどちらかを新規作成します。

構成マネージャー			?	\times
アクティブ ソリューション構成(2):	アクティブ ソリューション プラットフォーム(P):		
Release		Any CPU		~
		Any CPU		
ノロシェクトのコンテキスト (ヒ)	レトまたは配置するノロジェクト構成を	^{ナエ} <新規作成>		
プロジェクト	構成	<編集>		
ConsoleApp1	Release	Any CPU 💙		

₹たは配置するフロシェクト構成をチェック)(R):

?	×
	~
∈成する(C)	
キャン	セル
	? = 成する(C) キャン

(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「PdfTkNet6_70.dll」を選択します。ソリューションプ ラットフォームが「x64」の場合は「bin64」フォルダにある 64bit 用、「x86」の場合は「bin32」 フォルダにある 32bit 用の「PdfTkNet6 70.dll」を選択してください。

参照マネージャー - ConsoleA	pp1			? ×
▶ プロジェクト				検索 (Ctrl+E)
▶ 共有プロジェクト ▶ COM	V	名前 PdfTkNet6_70.dll	КZ	名前: PdfTkNet6_70.dll 作成者:
 参照 最近使用したファイル 				PdfTkNet6_70 ファイル バージョン: 1.0.0.0

(12)「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。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」 - 「アセンプリ」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 ConsoleApp1 です。)



- 📢 ファイル(F) 編集(E) 表示(M) Git(G) プロジェクト(P) ビルド(B) デバッグ(D) テスト(S) 分析(M) ツール(T) 検索 (Ctrl+Q) 👂 Con...pp1 × 拡張機能(X) ウィンドウ(W) ヘルプ(H) 🖁 📀 • ⊚ | 🎁 • 🗃 💾 🗐 | ♡ • ♡ • | Release • 🛛 🗙 🕯 🔹 🕨 ConsoleApp1 🔹 🖄 🍯 🗸 🖻 Live Share 🛛 🖗 ConsoleApp1 - 🛛 🗙 sampleapi01_cs ☆ 🖉 🗿 • ≒ 🗐 検索プロパティ ノリューション エクスプローラー の検い 🔎 ;NU1605 ▶ global usings マリューション 'ConsoleApp1' (1/1 ▲ ConsoleApp1 ▶ アプリケーション ▲ 昭 依存関係 ▲ ENF ▲ BE アセンブリ 出力 基本出力パス PdfTkNet6_70 全般 ビルド時のプロジェクトの出力の基本場所を指定します。プロジェクト構成を区別す るために、サプフォルダーがこのバスに追加されます。 ▶ 📑 アナライザー エラーおよび警告 ・ フレームワーク 出力 参照... C# sampleapi01_cs.cs イベント bin¥ 厳密な名前付け 詳細 ▶ パッケージ 基本中間出力パス ビルド時のプロジェクトの中間出力の基本場所を指定します。プロジェクト構成を 区別するために、サプフォルダーがこのパスに追加されます。 Code Analysis ▶ デバッグ 参照... obj¥ ▶ IJIJ-ス 参照アセンブリ ② ✓ プロジェクトのパブリック API を含む参照アセンブリを生成します。 ドキュメント ファイル ② API ドキュメントを含むファイルを生成します。 イベント ビルド前のイベント ③ ソリューション エクスプローラー プロパティ ↑ ソース管理に追加 ヘ 団 リポジトリの選択 ヘ 📮 □ 準備完了
- (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 PdfTkNet6_70.dll、PdfTkNet6.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「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 ConsoleApp1 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 ∪ 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。

マアイル(F) 編集 拡張機能(X) ウ ※ ・ ● 10 ●	(E) 表示(V) Git(G) プロジェク ィンドウ(W) ヘルプ(H) 🇉 🖹 🖹 🏷 👻 🖓 🐂 🤇	トP ビルド(8) デパッグ(D) アスト(S) 分析(N) ツール(T) 検索(Ctrl+Q) P Conpp1 - □ × e • x64 • ▶ ConsoleApp1 • ▷ ♂ • 節 罰 。 Ich Live Share <i>R</i>
y ConsoleApp1 + 検索プロパライ ・ り global usings ・ アプリケーション ・ レルド ・ Code Analysis ・ デバッグ	× sampleapi01_cs.cs デバッグ 全般	ConsoleApp1* × ◆ ジリューション エクパブローク・ ● ● × ● × ● ジリューション エクパブローク・ ● ● × ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●
<u>≄₩</u> ▶ リソ-ス	<u>リソース</u> 全般	アセンブリリソースの管理は、プロジェクト・プロパティ経由では実行されなくなりました。代わりに、ソジューション エクスプローラーから RESX ファイルを直接開きます。低 室上、以下のリンクからアクセスできます。 アセンブリリソースを作成する/驚く ソリューション エクスプローラー プロパティ
□ 準備完了		个 ソース管理に追加 🔺 🔟 リポジトリの選択 🔺 🚨

 (3) 「プロファイルの起動」ダイアログの「作業ディレクトリ」の欄に PdfTkNet6_70.dll、 PdfTkNet6.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。

プロファイルの起動		×
는 문 는 팩	コマンド ライン引数 実行可能ファイルに渡すコマンド ライン引数。引数を複数行に分割できます。	Î
	作業ディレクトリ プロセスが開始される作業ディレクトリへのパス。 参照	
	リモート マシンを使用する リモートマシン上のプロセスにデパッガーをアタッチする必要があることを示します。	
	環境変数 プロセスを実行する前に設定する環境変数です。 名前 値	

(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。

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。

公開 現在公開している場所			×
ターゲット		Azure アブリケーションを Microsoft クラウドに発行します	
		ClickOnce ClickOnce でアプリケーションを発行します	
	.	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	
	Đ	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	
	4	ブロファイルのインボート アブリを配置するための発行設定をインボートします	
		戻る(B) 次へ(N) 完了(F) キャンセ	JV(C)

(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。

公開 Kan-tu a-ťubis	w=1 ± ±	40 J	×
ターゲット		ッ? ClickOnce ClickOnceでアプリケーションを発行します	
特定のターゲット		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	
		戻る(8) 次へ(N) 完了(F) キャンセル(6))

(4) 「フォルダーの場所」を指定します。

		×
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	
to Hart	フォルダーの場所	
ターケット	bublish¥ 参照(R)	
特定のターゲット		
場所	 □-カル フォルターの場合は、フ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¥User¥Usernam¥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(8) 次へ(N) 完了(F) キャンセル(C)	5

- (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。

	*12 PHP) ビルF(8) デバッグ(D) テスト(5) 分析(N) ブール(T) 検索(Ch1+Q) Lesse ・ 1 x64 ・ ト ConsoleApp1 ・ ト グ ・ 節 蒙 。	P Con…pp1 - □ × E Live Share
910-00000000000000000000000000000000000	sumplexploit_c.cs ConsoleApp1' * X * Folderfrollie.pubml * フルガー ・ ・	
□ 準備完了	↑ ソース音楽	単に追加 🔺 🔟 リボジトリの選択 🔺 🚨 🍃

(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:net6.0

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)」、「win-x86 (Windows 32bit)」、「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。

		x
プロファイル	設定	
プロファイル名(P)	FolderProfile	
構成	Release x64 🔹	
ターゲット フレームワーク	net6.0 🔹	
配置モード	自己完結	
ターゲット ランタイム	win-x64 💌	
ターゲットの場所	publish¥	
→ ファイルの公開オ :	プション	
	「休仔 キャノゼル	

(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。

畫成	Release I x64	-
ターゲット フレームワーク	net6.0	
記置モード	自己完結	•
ターゲット ランタイム	win-x64	-
7ーゲットの場所	publish¥	
 ファイルの公開オ: 単一ファイル ReadyToRu 	プション の作成 n コンパイルを有効にする	

(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。

CP+1ル(F) 編集(E) 表示(V) Git(G) プロジ 拡張機能(X) ウォンドウ(W) ヘルプ(H) W(G) ・ ○ 御・番 回 図 ウ ・ ○ ・ Re	ジェクト(P) ビルド(B) デパッグ(D) テスト(S) 分析(N) ツール(I) 検索(Ctrl+Q) kease • 1x64 • ▶ ConsoleApp1 • ▷ ♂ • ■ 扇 。	Conpp1 - 🗆 X
Y=Cfc_17770-7- 接続済みサービス 公園 10載 山力	Implepend c.cs ConsoleApp1**** FolderProfile.pubml - フスルゲー マスルゲー + 新祝 その他のフクジョン・ ・ ・ 新祝 その他のフクジョン・ ・ ・ 新祝 その他のアクジョン・ ・ ・ 「 小 小 小 小 小 小 小 小 小 小 小 小 小 小 小 小 小 小	
↓ 準備完了	个 ソース管	理に追加 🔺 🔟 リポジトリの選択 🍝 🗍

(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。

ConsoleApp1: 公開 💠 🗙 sampleapi01_cs.o	s	 * ♥ ソリューション エクスフローラ- * ₽ ○ ○ 命 ๗ ๗ '○ * ≒ □
接続済みサービス	100eFr0mepb04m - フォルダー	ソリューション エクスプローラー の検索 の マリリューション 'ConsoleApp1' (1/
	+ 新規 その他のアクション ★	▲ ConsoleApp1 ▶ 🎝 Properties ▶ 8日 依存關係
	<u> </u>	 publish C # sampleapi01_cs.cs
		_
	システ ターゲットの場所 publish¥ ロ	
	構成 Release ターゲットフレームワーク net6.0	
	ターゲットランタイム win-x64 タ すべての設定を表示	

(11)「ターゲットランタイム」が「win-x86」、「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、『<u>アプリケーションの発行</u>』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
- 「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。「「PdfTk70.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「PTL70_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.NET6 ランタイム インストーラの入手先

https://dotnet.microsoft.com/ja-jp/download/dotnet/6.0

「.NET Runtime 6.0.xx」の項にある「Windows」の「x64」または「x86」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

- 「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 『付録 1.,NET6 ランタイムのインストール方法』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} PTL70_LIC_PATH={ライセンスファイルのディレクトリ} PTL70_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.NET6 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。

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。

コマンド:dotnet {アプリケーションファイル名} {オプション

付録 1..NET6 ランタイムのインストール方法

(1) 下記のサイトから NET6 ランタイムのインストーラを入手します。

https://dotnet.microsoft.com/ja-jp/download/dotnet/6.0

「.NET Runtime 6.0.xx」の項にある「dotnet-install scripts」をクリックします。

すべて	dotnet-install scripts			ヤーの手順	Arm64 Arm64 x64 x64 Alpine	Alpine
は加済みう	1911		macOS		Arm64 x64	
19回転のフンタイム NET Runtime 60.15 ASP.NET Core ランタイム 6.0.15 .NET デスクトップ ランタイム 6.0.15		Windows Hosting Bundle Arm64 x64 x86 x64 x86 winget の手順				
言語サポー C# 10.0 F# 6.0 Visual Basi SDK 6.	• F ic 16.9 0.310		.NET フ .NET Deskt クトップフ .NET ラン? 要はありま	「スクトップ top ランタイムを使 アプリケーションを タイムが含まれてい させん。	ランタイム 6 用すると、既存の Wi 実行できます。このリ ます。個別にインスト	.0.15 ndows デス リリースには 〜ールする必
os	インストーラー	バイナリ	05	1-7-5-		152+11
Linux	<u>パッケージマネー</u> <u>ジャーの手順</u>	Arm32 Arm32 Alpine Arm64 Arm64 Alpine x64 x64 Alpine	Windows	<u>Arm64 x64 x8</u>	<u>86 winget の手順</u>	
macOS Arm64 x64 Arm64 x64		.NET Runtime 6.0.15				
Windows	Arm64 x64 x86 wingetの手順	Arm64 x64 x86	.NET Runtime には、コンソール アプリの実行に必要なコンパ ーネントのみが含まれています。 通常、ASP.NET Core Runtime または .NET Desktop Runtime のいずれかもインス		要なコンポ ore もインスト	
すべて	dotnet-install scripts		OS インストーラー バイナリ			
Visual Stu Visual Stuc	dio サポート dio 2022 (v17.2)		Linux	<u>パッケージマネー</u> ジャーの手順	Arm32 Arm32 Al Arm64 Arm64 Al x64 Alpine	oine oine x64
に含まれる Visual Stur	tio 17 2 14		macOS	Arm64 x64	Arm64 x64	
Visual Studie 17.2.14 付加済みランタイム .NET Runtime 6.0.15		Windows	<u>Arm64 x64 </u> <u>x86 </u> <u>wingetの手順</u>	<u>Arm64 x64 x86</u>		
NET デスクトップ ランタイム 6.0.15		すべて	dotnet-install			
言語サポー C# 10.0 F# 6.0	۲. ۲. (۵)			scripts		

- (2) 「Bash」の項のアドレスからスクリプトをダウンロードします。
- (3) インストールを実行します。

コマンド:sh ./dotnet-install.sh --version latest --runtime dotnet (.NET6 ランタイムの最新バージョンをインストールする場合)

- (4) インストール先を指定しなかった場合、「ホーム(\$HOME)/.dotnet」ディレクトリにインスト 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。
- (5) インストールした.NET6 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 (6) 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。 コマンド:dotnet --list-runtimes

「Microsoft.NETCore.App 6.0.16 [/home/test/.dotnet/shared/Microsoft.NETCore.App]」のように表示されます。

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。

- 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。

	名前	▼ サイズ	更新日
♡ 最近開いたファイル ★ Starred	ipaexg.ttf	6.1 MB	2019年4月26日
ひ ホーム	jpaexm.ttf	7.8 MB	2019年4月26日
↓ ダウンロード	ipag.ttf	6.2 MB	2011年4月12日
□ ドキュメント	ipagp.ttf	6.2 MB	2011年4月12日
■ ビデオ ロ 奈楽	a ipam.ttf	8.0 MB	2011年4月13日
5) 画像	a ipamp.ttf	8.0 MB	2011年4月13日
団 ゴミ箱			
+ 他の場所			

(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、.NET6 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

『<u>付録 1..NET6 ランタイムのインストール方法</u>』の(3)でのインストールコマンドを ./dotnet-install.sh --version latest に変更して実行します。このコマンドにより、.NET6 をランタイムとしてではなく、SDK と してインストールします。

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。 コマンド:dotnet --list-sdks 「6.0.408 [/home/test/.dotnet/sdk]」

のように表示されます。

- 5.1.2. Visual Studio Code のインストール
- (1) Download Visual Studio Code

<u>https://code.visualstudio.com/download</u> を開いて「.rpm」をダウンロードします。



Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.



By downloading and using Visual Studio Code, you agree to the license terms and privacy

(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。

<	code	×
	code Code editing. Redefined.	
	インストール()	

<u><</u>	code	×
	code Code editing. Redefined. インストール中 0%	ヤンセル(C)
	ぼ証が要求されました 信頼されていないソフトウェアをインストールする には認証が必要です	
	Visual Studio Code is a new ch キャンセル 認証 nee core edit-build-debug cycle. See maps://codevisuals.tudio.com/ocs/setup/inito/tomistanauon/insuucuons/and/i	rd for the FAQ.

(3) VSCode 拡張機能「Japanese Language Pack for VS Code」のインストール
 左側アクティビティバーにある拡張機能をクリックします。
 開いたサイドバー上部にある検索欄に「Japanese」と記入し結果に表れる「Japanese Language Pack for VS Code」を選択します。



青色の「install」ボタンを押すとインストールが始まります。

¢	EXTENSIONS: MARK- 🖓 🖔 📰 …	Extension: Japanese Language Pack for Visual Studio Code ×	Ω
Q %	japanese Japanese L··· の5.7M ★45 日本語 Microsoft Install	Japanese Language Pack for Visual Studio Co Microsoft © microsoft.com Ф 5,763,292 ★★★★★ (7) Language pack extension for Japanese	ode v1.79.2023060709
° ₽	Japanese Wor ··· O 40% *5 Better Japanese word han	EETALS FEATURE CONTINUENTIONS CHANNELLOG	
	Enables grammar checkin Adam Voss	Japanese Language Pack for VS Code	Categories
	I like it when the color po	Japanese Language Pack は VS Code にローカライズされた UI を提供します。 46 田 注	Extension Resources
	Reflow Japanese 🗠 842 Reflows and wraps texts b Yodoware Inc.	ICC TDA2 現金の UI 書店をオーパーライドするには、 Configure Display Language コマンドを使用して、VS Code の表示言語 を明示的に設定します。Ctr Lisbut ft・P を押して [*] コマンド パレット [®] を表示し、d1sp1ay と入力して Configure	Marketplace Repository Microsoft

インストール終了後に再起動をすることによって日本語が適応されます。

(4) VSCode 拡張機能「C#」のインストール

左側アクティビティバーにある拡張機能をクリックします。 開いたサイドバー上部にある検索欄に「C#」と記入し結果に表れる「C#」を選択します。



青色の「install」ボタンを押すとインストールが始まります。



- 5.1.3. プロジェクトの作成 「フォルダを開く」と「ターミナルを開く」の2つの方法があります。
- 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。
- (4) 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。(画像では TEST_NET6 というフォルダを指定しています)

キャンセル(C)	フォルダーを開く	Q 開<(0)
◎ 最近開いたファイル		C;
命 ホーム	名前	- サイズ 更新日時
↓ ダウンロード		
□ ドキュメント		
■ ビデオ		

(5) 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。



5.1.3.2. 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。

	ようこそ・
ターミナル ヘルプ	
新しいターミナル	Ctrl+Shift+0
ターミナルの分割	Ctrl+Shift+5
タスクの実行…	
ビルド タスクの実行	Ctrl+Shift+B
アクティブなファイルの実行	
選択したテキストの実行	
実行中のタスクを表示…	
タスクの実行を再開…	
タスクの終了…	
タスクの構成…	
既定のビルド タスクの構成…	

(2) 開かれたターミナル画面に dotnet new のコマンドを入力して新しいコンソールアプリ ケーションのプロジェクトを作成します。

コマンド: dotnet new console --framework net6.0 --use-program-main

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

		ようこそ - test_net6 - V	sual Studio Code	×
771	'ル 編集 選択 表示 移動 実行 タ	マーミナル ヘルプ		
Сh		×1 ようこそ ×		
_	V TEST_NET6			
Q				
ŀ		Visual Studio Code		
¢		進化した編集		
₿		開始	チュートリアル	
		C₄ 新しいファイル	➢VS Code を開始する	
		1 ファイルを聞く 2 フォルダーを聞く	最適なカスタマイズを見つけて、好みの VS Code にしてくださ い。	
			● 基礎の学習	
		最近		
		最近使用したフォルダーはありません。 フォルダーを聞く	開始へ。	
			☑ 起動時にウェルカム ページを表示	
		問題 出力 デバッグコンソール ターミナル	(≢bash +~ 🏾 🛱	
		<pre>[test@localhost test net6]\$</pre>		
8				
633	> 70トライン			
00				8 0



- 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
- PDF Tool API のインストール
 [2.PDF Tool API の開発環境を整える」を参照してください。
- (2) VSCode を起動してプロジェクトを作成します。
- (3)「(フォルダ名).csproj」を開いて赤字の箇所を追記します。<HintPath> には PdfTkNet6_70.dll のパスを記入します。

<project sdk="Microsoft.NET.Sdk"></project>	
<propertygroup></propertygroup>	
<outputtype>Exe</outputtype>	
<targetframework>net6.0</targetframework>	
<implicitusings>enable</implicitusings>	
<platform>x64</platform>	
<nullable>enable</nullable>	
<itemgroup></itemgroup>	
<reference include="PdfTkNet6_70"></reference>	
<hintpath>/usr/AHPDFToolV7-lib/bin/PdfTkNet6_70.dll</hintpath>	

(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。

ファイル 編集 選択 表示 移動 実行 🤅	ターミナル ヘルス	ſ
新しいテキスト ファイル	Ctrl+N	
新しいファイル	Alt+Ctrl+N	
新しいウィンドウ	Ctrl+Shift+N	
ファイルを開く	Ctrl+0	
フォルダーを開く… [Ctrl+K Ctrl+O]		
ファイルでワークスペースを開く		
最近使用した項目を開く		3
フォルダーをワークスペースに追加		
名前を付けてワークスペースを保存…		
ワークスペースを複製		
保存	Ctrl+S	
名前を付けて保存	Ctrl+Shift+S	
すべて保存		
共有		33
自動保存		

(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。

ファイル 編集 選択 表示 移動 実行 ター	ーミナル ヘルプ
新しいテキスト ファイル	Ctrl+N
新しいファイル	Alt+Ctrl+N
新しいウィンドウ	Ctrl+Shift+N
ファイルを開く	Ctrl+O
フォルダーを開く… [Ctrl+K Ctrl+O]	
ファイルでワークスペースを開く	
最近使用した項目を開く	•
フォルダーをワークスペースに追加	
名前を付けてワークスペースを保存	
ワークスペースを複製	
保存	Ctrl+S
名前を付けて保存…	Ctrl+Shift+S
すべて保存	
共有	•
自動保存	
横に並べて開く	Ctrl+Enter
このアイテムのフォルダーを開く	Alt+Ctrl+R
ファイルを開くアプリケーションの選択…	
統合ターミナルで開く	
比較対象の選択	
タイムラインを開く	
切り取り	Ctrl+X
コピー	Ctrl+C
パスのコピー	Alt+Ctrl+C
相対パスをコピー	Alt+Ctrl+Shift+C
名前の変更	F2
削除	Delete

- (6) ターミナルに下記の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、下記のコマンドで実行します。 dotnet run

- 5.2. Windows 環境
- 5.2.1. NET SDK のインストール

.NET6.0 のウンロード <u>https://dotnet.microsoft.com/ja-jp/download/dotnet/6.0</u> 「SDK 6.0.xx」の項にある「Windows」の「x64」または「x86」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
- 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

「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」をご参照ください。

履歴

日付	更新内容
2023.5.29	・初版
2023.7.4	・「5.Visual Studio Code でのプログラム作成と実行方法」を追加しました。
2024.10.31	・「3.サンプルコードのプロジェクト作成とビルド方法」の説明文を修正しま
	した。

Antenna House PDF Tool API V7.0 サンプルコードのビルドと実行手順(.NET6) 2024.10.31 Antenna House Inc. 2023-2024 All Rights Reserved.