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

アンテナハウス株式会社

目次

1.はじめに

本書は Linux 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、Linux 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。
ビルド環境の構築について、C++, Java の各インターフェースのそれぞれにおける説明をします。
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。

注意：

本マニュアルにおける Linux の操作説明は全て「Red Hat Enterprise Linux (RHEL)」における表記を用いています。その他のディストリビューションをお使いの場合は該当部分を読み替えてご使用ください。

2.PDF Tool API の開発環境を整える

ここでは、Linux における PDF Tool API の環境整備について解説します。
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。

必要な手順の概要を、以下の順で説明します。。
1. 各種モジュールファイルを配置する
2. ライセンスファイルを配置する
3. フォントの準備
4. Linux 開発・実行環境における環境変数の設定

Linux 版を含む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.各種モジュールファイルを配置する

2.1.1. インストーラによる配置

(1) 「AHPDFToolLib80-8.0-***.x86_64.rpm」に対して rpm コマンドを実行してインストールをします。
「***」には R1、MR1 などの改訂バージョン名が入ります。

```
rpm -i AHPDFToolLib80-8.0-***.x86_64.rpm [--prefix インストールパス]
```

(2) インストールパスを指定しなかった場合、以下のパスにインストールされます。
/usr/AHPDFToolLib80

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

- 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。

注意：
環境変数「PTL80_LIC_PATH」が自動で指定されることはありません。具体的な操作に関しては「2.2.2 ライセンスファイルの参照先指定」をご参照ください。

2.2.2. ライセンスファイルの参照先指定

(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。

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。
Linux ではフォント情報を扱うための「フォント構築ファイル」が必要となります。

2.3.1. 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）：(Liunx 環境で「/root/TestFont」フォルダを指定したい場合)

```
<font-config>
  <font-folder path="/root/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.Linux 開発・実行環境における環境変数の設定

Linux 環境で PDF Tool API を扱う場合、以下の環境変数の設定が必要です。

環境変数名	設定すべき値
LD_LIBRARY_PATH	API の実行に必要なモジュールファイルが存在するディレクトリのパス
PTL80_LIC_PATH	ライセンスファイル「ptalic.dat」が存在するディレクトリのパス
PTL80_FONT_CONFIGFILE	フォント構築ファイル「font-config.xml」のフルパス
PTL80_ICCPROFILE_PATH	カラープロファイル「sRGB2014.icc」「JapanColor2001Coated.icc」の配置ディレクトリのパス

※注意：

各環境変数に関する詳細はそれぞれ以下のリンクをご参照ください。

[LD_LIBRARY_PATH]…ライブラリ版マニュアル『モジュールファイルについて』

[PTL80_LIC_PATH]…ライブラリ版マニュアル『ライセンスファイルについて』

[PTL80_FONT_CONFIGFILE]…ライブラリ版マニュアル『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』

[PTL80_ICCPROFILE_PATH]…ライブラリ版マニュアル『カラープロファイルの扱い』

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 GCC を用いたビルド方法です。具体的には、サンプルプログラムのビルドと実行を行います。
- PDF Tool API と互換性のあるコンパイラをご使用ください。バージョンに関して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。
- 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。
- C++のサンプルコードのソースファイルは、製品 zip ファイルを解凍した先の以下ディレクトリに配置されています。
  「***」には R1、MR1 などの改訂バージョン名が入ります。
  AHPDFToolAPI8-Linux_X86_64-Lib_***/SampleCode/cpp

3.1.1. C++ビルド・実行環境の設定

(1) C++でビルド・実行をする際に必要な環境変数（※1）を設定します。
環境変数をコンソール上で指定する場合、以下の例のようになります。

```
> export LD_LIBRARY_PATH=/usr/AHPDFToolLib80/lib:${LD_LIBRARY_PATH}
> export PTL80_LIC_PATH=/usr/AHPDFToolLib80/License
> export PTL80_FONT_CONFIGFILE=/usr/AHPDFToolLib80/fontconfig/font-config.xml
> export PTL80_ICCPROFILE_PATH=/usr/AHPDFToolLib80/icc
```

（※1）
Linux 環境で必要な環境変数の詳細は「2.4 Linux 開発・実行環境における環境」をご参照ください。

3.1.2. サンプルコードのビルド

ここでは C++サンプルコードのビルド・実行をコンソール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「SHIFT-JIS」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『3.1 C++個別のサンプルコードのビルド方法』を参照してください。

以下は、サンプルプログラム「GetDocInfo.cpp」をビルド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ディレクトリに切り替えます。
以下は「/root」に SampleCode を配置した例です。

```
> cd /root/SampleCode/cpp
```

(2) ビルドしたい cpp ファイルを指定し、ビルドします。
本例では「GetDocInfo.cpp」をビルド対象に指定し、出力ファイル名は「GetDocInfo」です。

```
> g++ GetDocInfo.cpp -o GetDocInfo -I /usr/AHPDFToolLib80/Include -L /usr/AHPDFToolLib80/lib -lPtkAHCommon -lPtkAHDMC -lPtkAHGraphicService -lPtkAHFontService -lPdfTk -lPdfTkEx -lPtkAHPDFLib -lPtkAHPDFEditLib -lPtkPDFLinearizer -licuuc -licui18n -licudata
```

3.1.3. C++実行ファイルの実行

ビルド後の実行に必要な環境変数等の設定は「3.1.1 C++ビルド・実行環境の設定」をご参照ください。

(1) コンソールを開き、ビルドしたファイルを実行します。

```
./GetDocInfo in.pdf
```

3.2.同梱の C++実行用シェルスクリプトについて

PDF Tool API のサンプルディレクトリには C++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シェルスクリプト「cppsample-compileandrun.sh」が同梱されています。

- シェルスクリプトを用いることで、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
- シェルスクリプト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。
  詳細は配置ディレクトリにあります「ReadMe-samples_Linux.txt」をご参照ください。
- シェルスクリプト及び解説書が配置されたサンプルディレクトリは製品 zip ファイルを解凍した以下の場所です。
  「***」には R1、MR1 などの改訂バージョン名が入ります。
  AHPDFToolAPI8-Linux_X86_64-Lib_***/SampleCode

3.2.1. シェルスクリプトの操作方法

本項では、シェルスクリプト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、ターミナルから「cppsample-compileandrun.sh」を呼び出します。

```
> ./cppsample-compileandrun.sh ImageToPdf
```

実行が完了すると完了メッセージが出てシェルスクリプトが終了します。

(2) シェルスクリプトと同じ階層に以下のディレクトリが生成され、各実行結果が出力されます。
  - 「cpp-bin」ディレクトリ：ビルドされた実行ファイルの出力先
  - 「cpp-out」ディレクトリ：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。
  - 「cpp-out-ExtractPage」ディレクトリ：サンプルプログラム「ExtractPage」専用の出力先

4..NET のビルドと実行手順

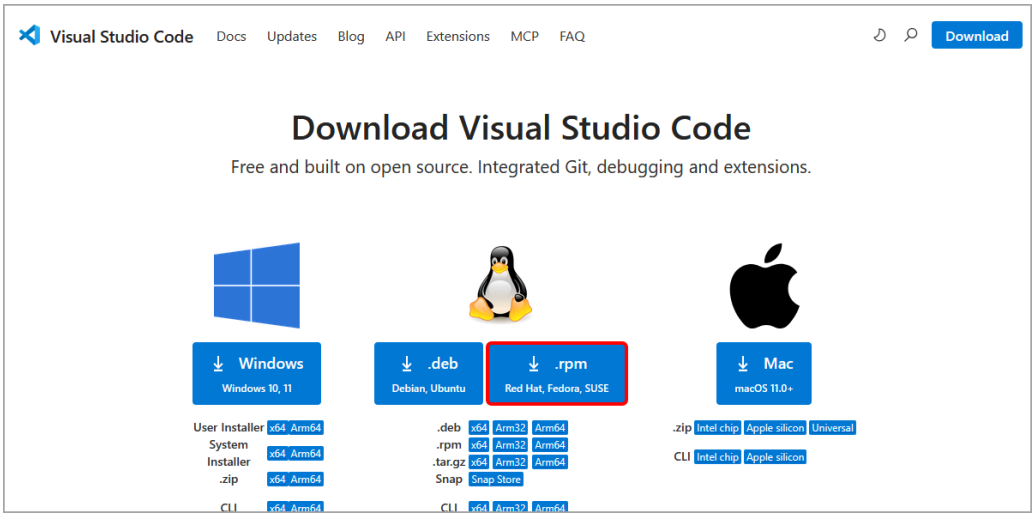
ここでは、Visual Studio Code(VSCode)を使用した、.NET の各種プログラム作成と実行方法を説明します。

- 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『各種モジュールファイルを配置する』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#サンプルコードのソースファイルは、製品 zip ファイルを解凍した先の以下ディレクトリに配置されています。
  「***」には R1、MR1 などの改訂バージョン名が入ります。
  AHPDFToolAPI8-Linux_X86_64-Lib_***/SampleCode/dotnet

4.1.Visual Studio Code のインストールと準備

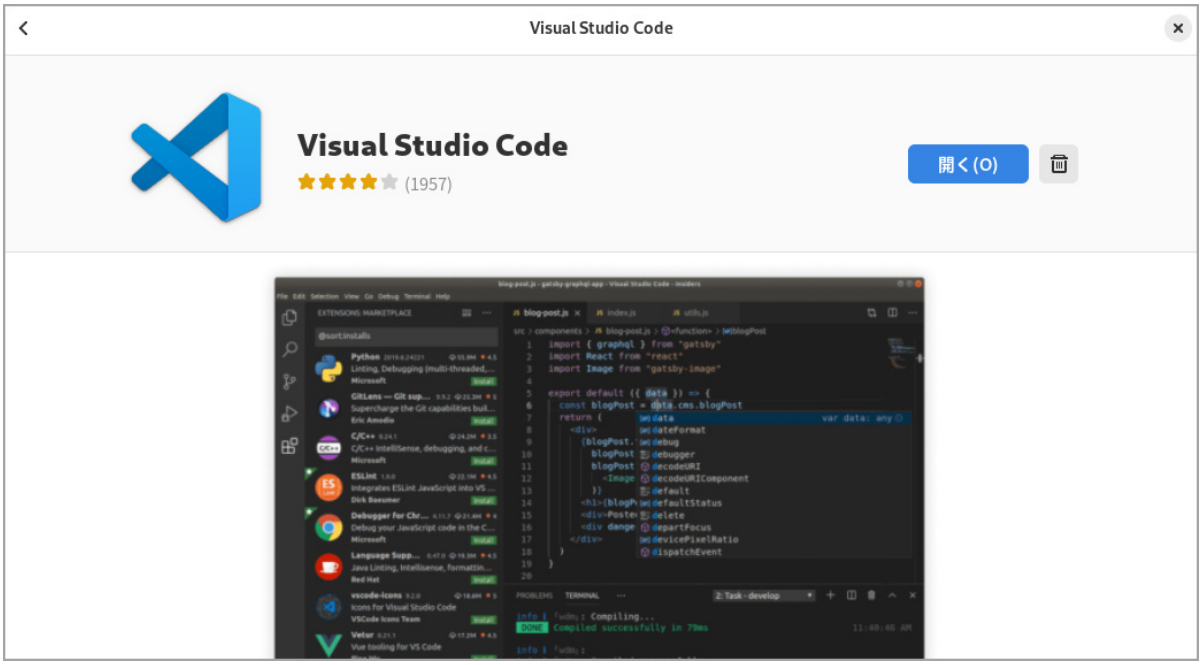
4.1.1. Visual Studio Code のインストール

(1) Visual Studio Code の公式サイトより、.rpm ファイルをダウンロードします。
『Download Visual Studio Code』を開いて「.rpm」を選択してください。
https://code.visualstudio.com/download



(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。

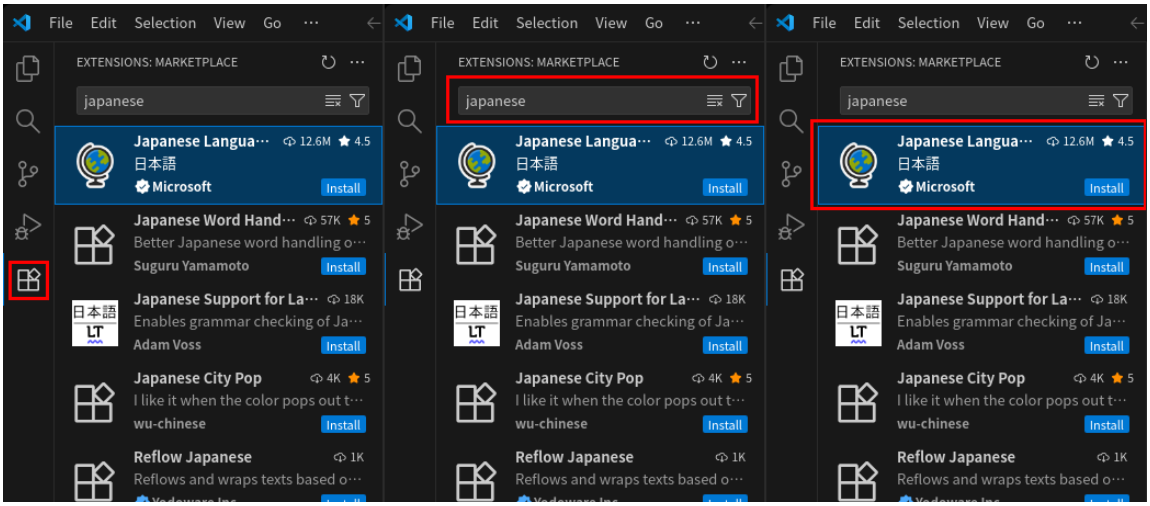
(3) インストールが完了した後、Visual Studio Code を開きます。



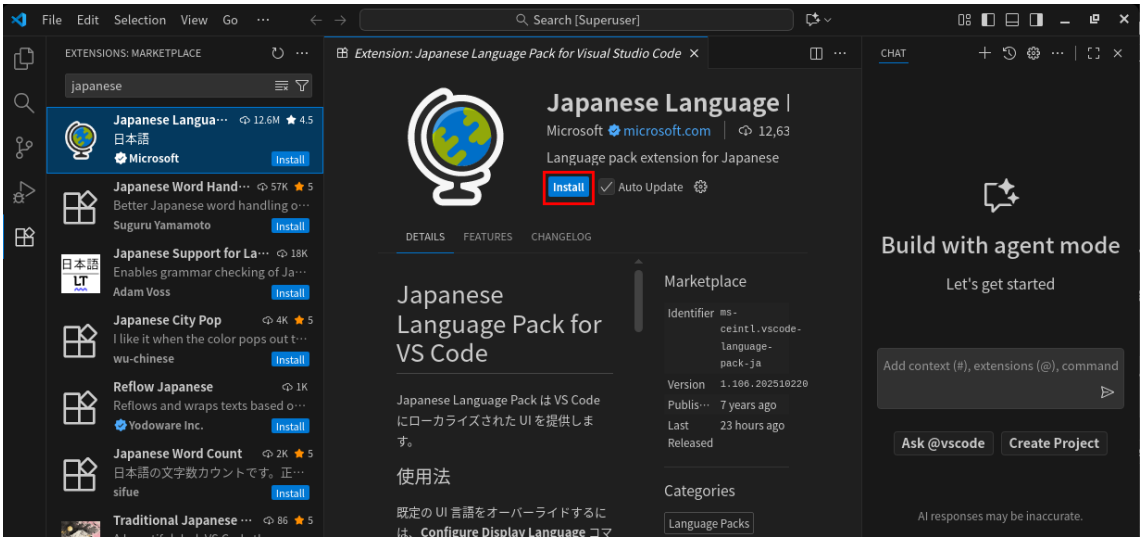
(4) VSCode 拡張機能「Japanese Language Pack for VS Code」のインストール

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。

4.2..NET 8 のサンプルコードビルド・実行

4.2.1. .NET SDK のインストール

Linux 環境への.NET SDK 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-sdk-8.0
```

これで.NET 8 の.NET SDK 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内の以下の位置にインストールされます。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。

($HOME)/.dotnet

(3) インストールした.NET SDK の「dotnet」コマンドをコンソールで使用するには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、システムを再起動することで、以後の環境変数の設定が不要になります。

```
export DOTNET_ROOT=${HOME}/.dotnet
export PATH=${PATH}:${DOTNET_ROOT}:${DOTNET_ROOT}/tools
```

(4) 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
```

実行した場合、以下のように表示されます。

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

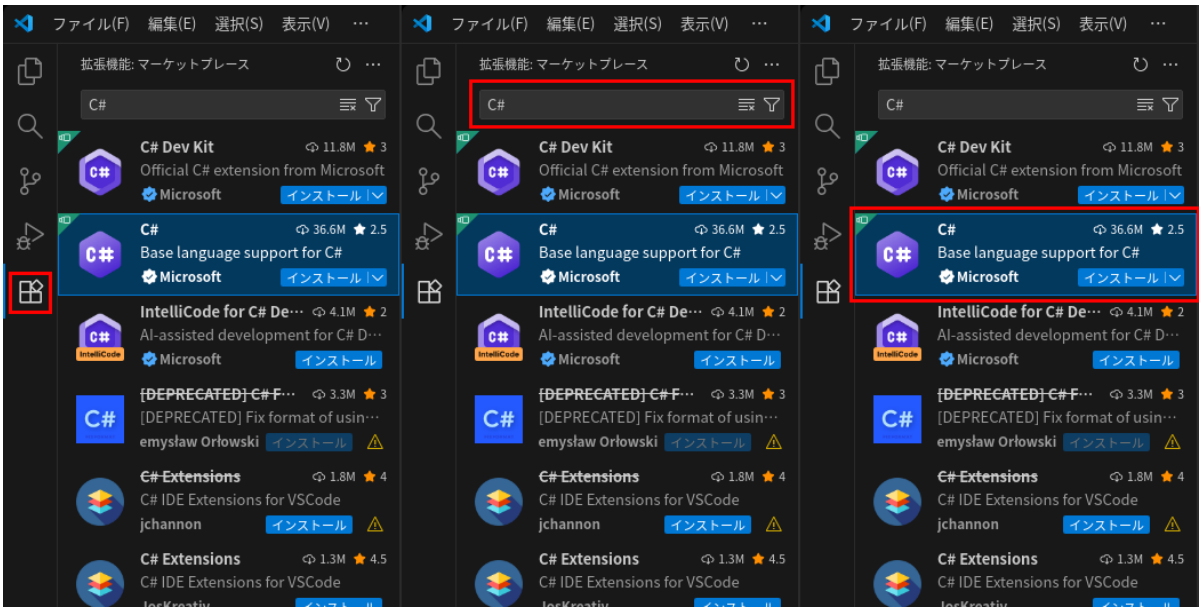
4.2.2. Visual Studio Code 拡張のインストール

VSCode 拡張機能「C#」をインストールします。

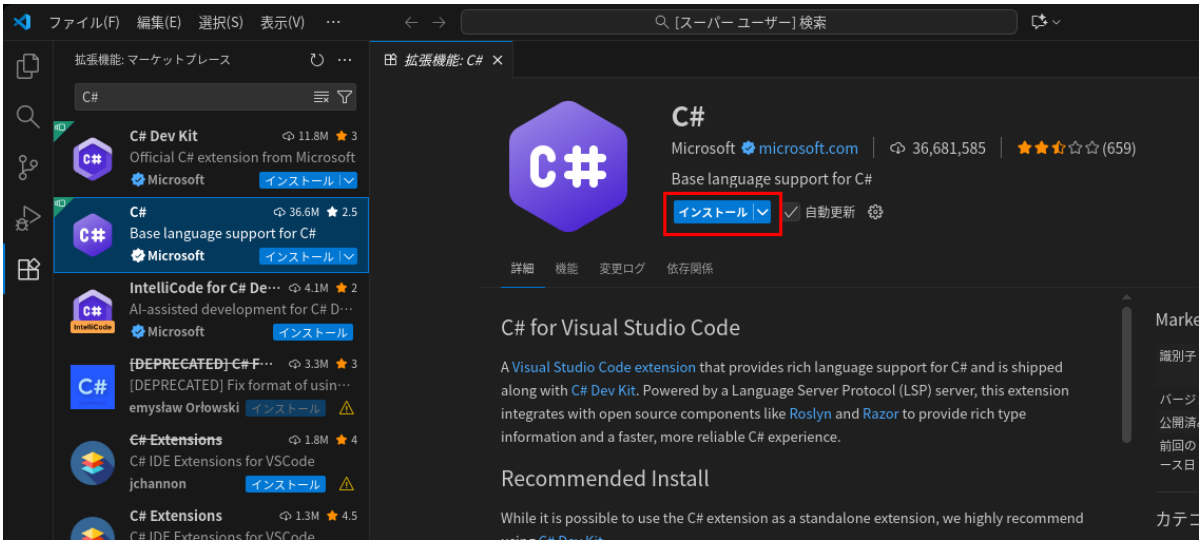
(1) 左側アクティビティバーにある拡張機能をクリック

(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。

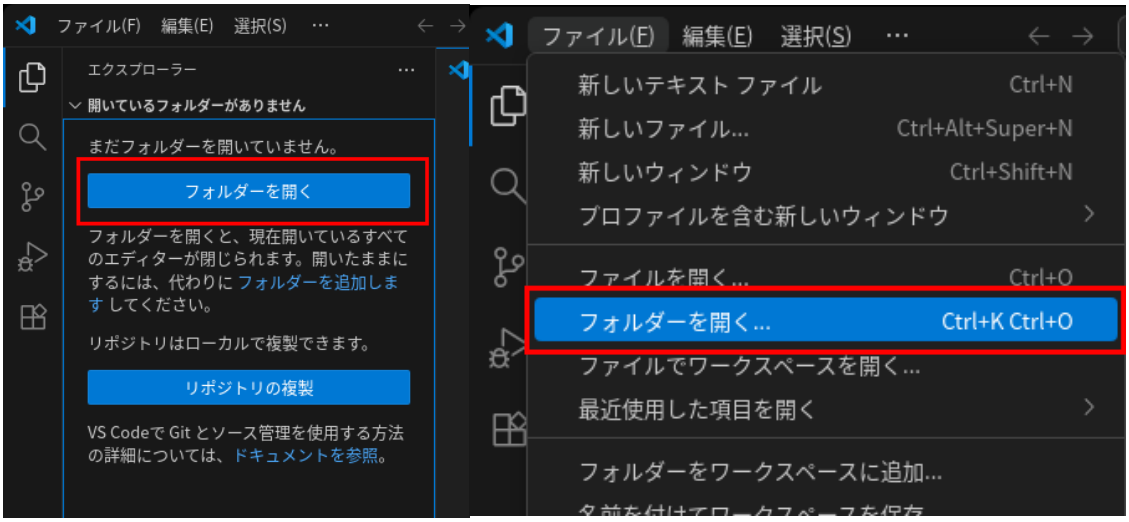
4.2.3. プロジェクトの作成

「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。

4.2.3.1. フォルダを開く

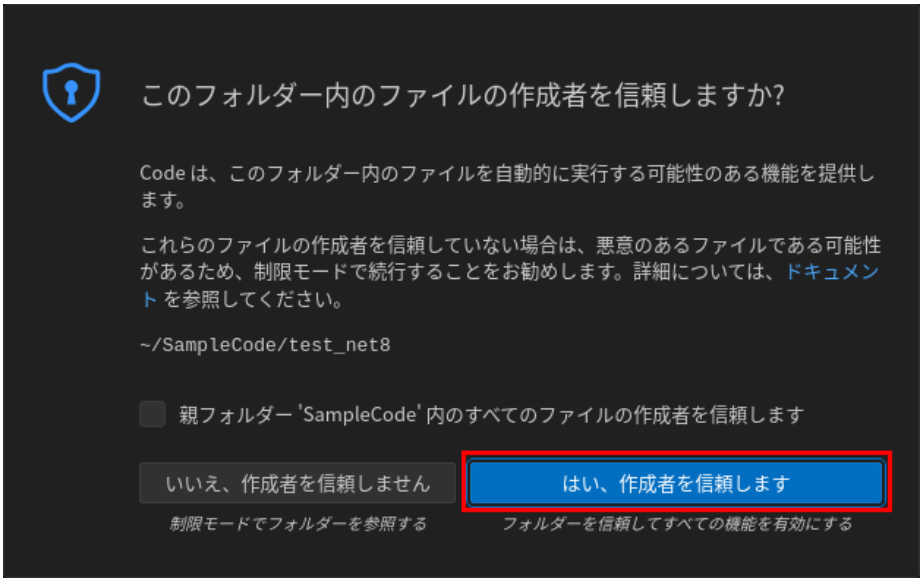
(1) 左側アクティビティバーにあるエクスプローラーを選択します。

(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。

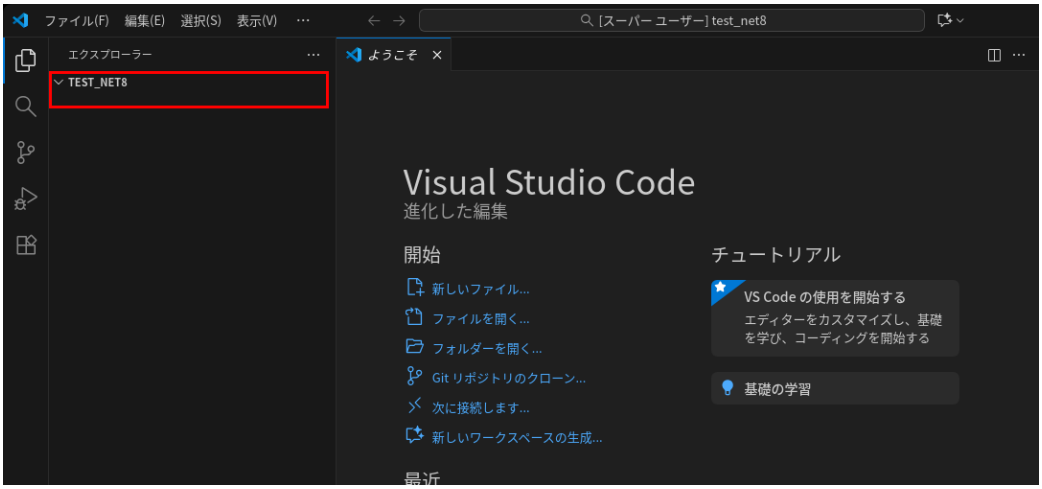


(3) プロジェクトのファイルを生成するフォルダを指定して右上の「開く」をクリックします。
（画像では test_net8 というフォルダを指定しています）

(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。

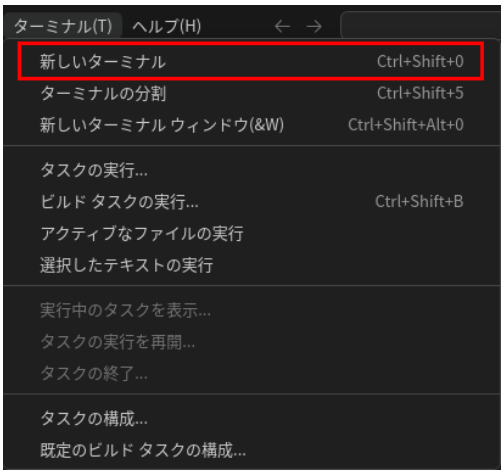


(5) 左側のサイドバーのエクスプローラーに指定したファイル名が表示されます。

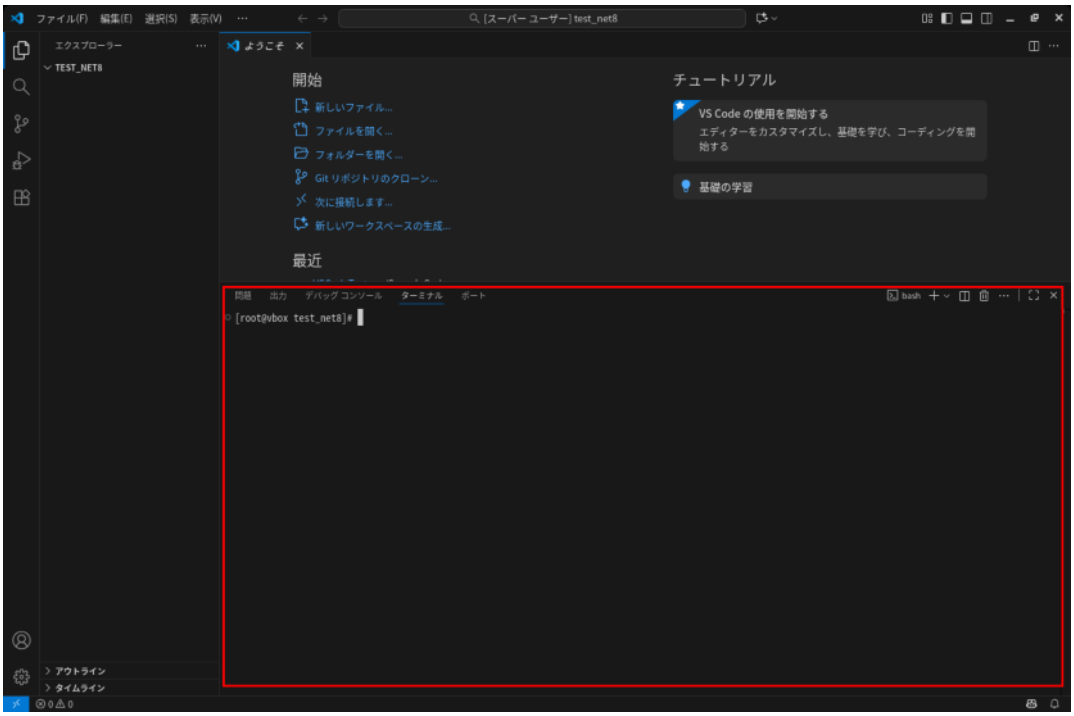


4.2.3.2. ターミナルによるプロジェクトの作成

(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 ）

4.2.4. 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。

「4.2.3 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。

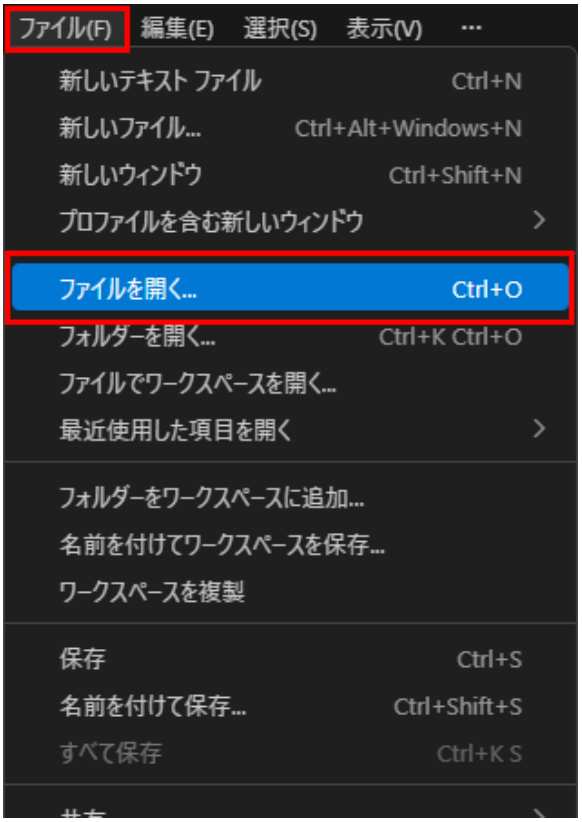
csproj ファイルはプロジェクト作成時に生成されます。

<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。

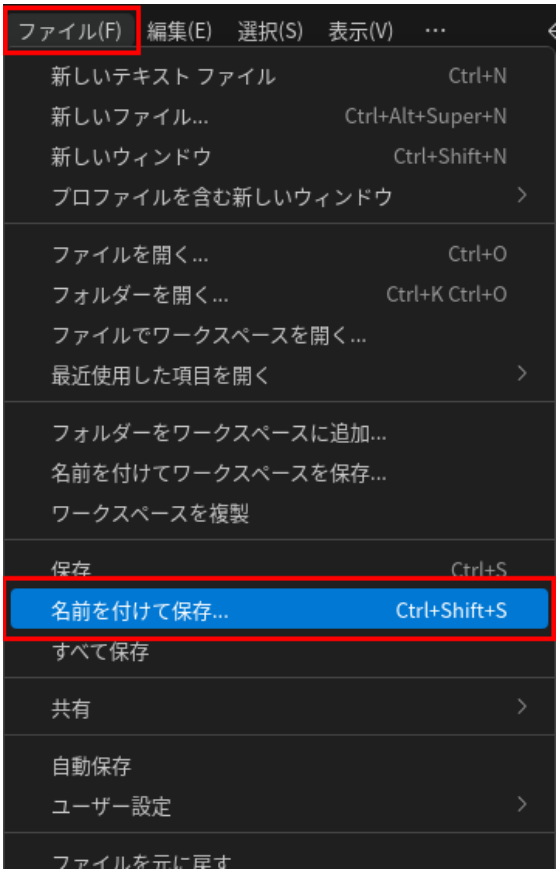
```
<Project Sdk="Microsoft.NET.Sdk">

  <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>
```

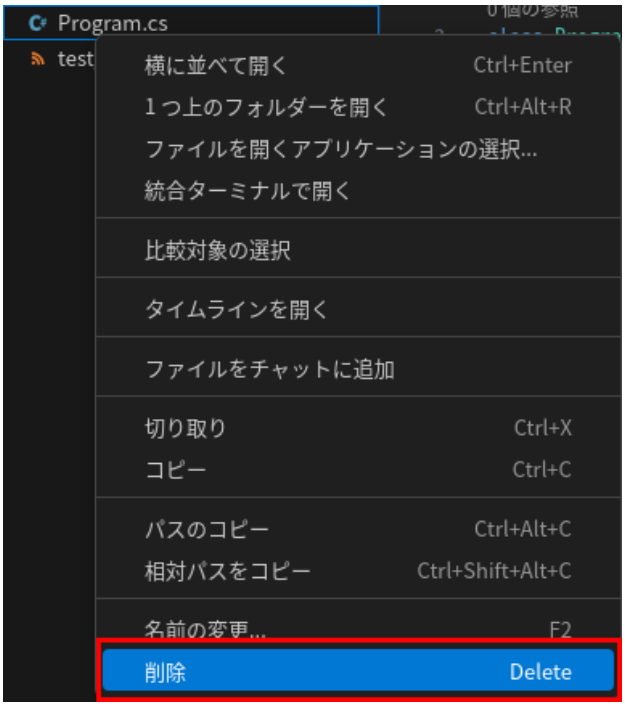
(4) 「ファイル」メニューの「ファイルを開く」を選択して使用したいサンプルコードの cs ファイルを開きます。



「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。

4.3.同梱の.NET 8 実行用シェルスクリプトについて

PDF Tool API のサンプルフォルダには.NET 8 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シェルスクリプト「dotnet8sample-run.sh」が同梱されています。

- シェルスクリプトを用いることで、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
  - 実行に必要な csproj ファイルの読み込み
  - 各サンプルに適した引数及びサンプル PDF の指定
- シェルスクリプト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_Linux.txt」をご参照ください。
- シェルスクリプト及び解説書が配置されたサンプルフォルダは製品 zip ファイルを解凍した以下の場所です。
  「***」には R1、MR1 などの改訂バージョン名が入ります。
  AHPDFToolAPI8-Linux_X86_64-Lib_***/SampleCode

4.3.1. シェルスクリプトの操作方法

本項では、シェルスクリプト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、ターミナルから「dotnet8sample-run.sh」を呼び出します。

```
> ./dotnet8sample-run.sh ImageToPdf
```

実行が完了すると完了メッセージが出てシェルスクリプトが終了します。

(2) シェルスクリプトと同じ階層に以下のディレクトリが生成され、各実行結果が出力されます。
  - 「dotnet8-out」ディレクトリ：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。
  - 「dotnet8-out-ExtractPage」ディレクトリ：サンプルプログラム「ExtractPage」専用の出力先

4.4..NET 8 のアプリケーションファイル実行について

ビルド済みアプリケーションファイルの実行のみを行う場合、環境にインストール.NET を SDK の代わりに.NET ランタイムにすることが可能です。

4.4.1. .NET ランタイムのセットアップ

(1) 『4.2.1 .NET SDK のインストール』の(1)のインストールコマンドを以下に変更して実行します。
これは.NET8 を.NET SDK としてではなく、ASP.NET Core ランタイムとしてインストールするコマンドです。

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

※：
インストール先ディレクトリに関しては『4.2.1 .NET SDK のインストール』をご参照ください。

(2) .NET ランタイムの「dotnet」コマンドを使用するために環境変数を設定します。
具体的な設定項目は『4.2.1 .NET SDK のインストール』をご参照ください。

(3) 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]
```

4.4.2. .NET アプリケーションの実行

.NET アプリケーションを実行する手順をまとめると以下のようになります。

(1) ターゲットフレームワークの.NET SDK またはランタイムをインストールしします。(※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) PDF Tool API のライセンスファイルを配置します。

(4) 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) でフォントディレクトリの設定を行います。(※3)

(5) 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。(※4)
以下はコンソールで設定する場合の例です。

```
> export LD_LIBRARY_PATH=/usr/AHPDFToolLib80/lib:${LD_LIBRARY_PATH}
> export PTL80_LIC_PATH=/usr/AHPDFToolLib80/License
> export PTL80_FONT_CONFIGFILE=/usr/AHPDFToolLib80/fontconfig/font-config.xml
> export PTL80_ICCPROFILE_PATH=/usr/AHPDFToolLib80/icc
```

(6) アプリケーションファイルを任意の場所に配置します。

(7) コンソールを起動し、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。
以下は「dotnet」コマンドを用いて実行する場合の例です。

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

(※1)

『4.2.1 .NET SDK のインストール』や『4.4.1 .NET ランタイムのセットアップ』をご参照ください。

なお、アプリケーションファイルにランタイムライブラリーを組み込んだ場合は不要です。

(※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)

テキスト透かしやテキスト追加といった処理に必要です。

詳細は『PDF Tool API の開発環境を整える』の『2.3 フォントの準備』をご参照ください。

(※4)

Linux 環境で必要な環境変数の詳細は「2.4 Linux 開発・実行環境における環境」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。

- 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。
- 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。
- Java のサンプルコードのソースファイルは、製品 zip ファイルを解凍した以下の場所に配置されています。
  「***」には R1、MR1 などの改訂バージョン名が入ります。
  AHPDFToolAPI8-Linux_X86_64-Lib_***/SampleCode/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。
Linux 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

環境変数をコンソール上で指定する場合、以下の例のようになります。

```
> export LD_LIBRARY_PATH=/usr/AHPDFToolLib80/lib:${LD_LIBRARY_PATH}
> export PTL80_LIC_PATH=/usr/AHPDFToolLib80/License
> export PTL80_FONT_CONFIGFILE=/usr/AHPDFToolLib80/fontconfig/font-config.xml
> export PTL80_ICCPROFILE_PATH=/usr/AHPDFToolLib80/icc
> export CLASSPATH=/usr/AHPDFToolLib80/lib/PdfTkJava80.jar:${CLASSPATH}
```

（※1）

Linux 環境で必要な環境変数の詳細は「2.4 Linux 開発・実行環境における環境」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・実行をコンソール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」を参照してください。

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。
以下は「/root」に SampleCode を配置した例です。

```
> cd /root/SampleCode/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。

以下はコンパイルされた「GetDocInfo.class」を実行する例です。

(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。

```
> mkdir Sample
> mv GetDocInfo.class Sample
```

(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[/home/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[/home/test]である必要があります。[/home/test/Sample]ではありません。

5.2.同梱の Java 実行用シェルスクリプトについて

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シェルスクリプト「javasample-compileandrun.sh」が同梱されています。

- シェルスクリプトを用いることで、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
- シェルスクリプト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_Linux.txt」をご参照ください。
- シェルスクリプト及び解説書が配置されたサンプルフォルダは製品 zip ファイルを解凍した以下の場所です。
  「***」には R1、MR1 などの改訂バージョン名が入ります。
  AHPDFToolAPI8-Linux_X86_64-Lib_***/SampleCode

5.2.1. シェルスクリプトの操作方法

本項では、シェルスクリプト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、ターミナルから「javasample-compileandrun.sh」を呼び出します。

```
> ./javasample-compileandrun.sh ImageToPdf
```

実行が完了すると完了メッセージが出てシェルスクリプトが終了します。

(2) シェルスクリプトと同じ階層に以下のディレクトリが生成され、各実行結果が出力されます。
  - 「Sample」ディレクトリ：コンパイルされた class ファイルの格納先
  - 「java-out」ディレクトリ：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」ディレクトリ：サンプルプログラム「ExtractPage」専用の出力先

履歴

日付	更新内容
2025.11.20	・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・(Linux)

2025.11.20