Visual COBOL チュートリアル

COBOL 開発: Visual Studio -.NET COBOL の単体テスト

1. 目的

本チュートリアルでは、.NET COBOL プログラムに対するテスト作成、実行方法、および、テスト結果を表示させる方法の習得を目的として います。

MFUnit は、Visual COBOL に搭載された xUnit 系の単体テストフレームワークです。xUnit はオブジェクト指向型の単体テストフレーム ワーク SUnit に起源を持つ JUnit や RUnit 等の単体テストフレームワークの総称です。MFUnit は xUnit の設計アーキテクチャーや 仕組みは取り入れつつも COBOL 開発者にとって扱いやすい手続き型の COBOL を対象とした単体テストフレームワークという設計思想 の下、開発されました。

MFUnit は COBOL 開発作業に以下の利点を提供します。

- テストを繰返し実行させることができるため、修正作業時などのテスト工数の削減が見込める
- Jenkins などの継続的インテグレーション (Continuous Integration) ツールと連携によりテストの自動化が行え、DevOps サイクルの導入が足がかりを作れる

2. 前提

- 本チュートリアルで使用したマシン OS : Windows Server 2022
- Visual COBOL 9.0 for Visual Studio 2022 がインストール済みであること

本資料は、.NET COBOL に対する単体テストフレームワークの利用方法を記載したチュートリアルです。ネイティブ COBOL の単体テ スト実現方法については、別チュートリアルを参照ください。

下記のリンクから事前にチュートリアル用のサンプルファイルをダウンロードして、任意のフォルダに解凍しておいてください。 サンプルプログラムのダウンロード

内容

- 1. 目的
- 2. 前提
- 3. チュートリアル手順の概要
- 3.1. IDE からの実行
 - 3.1.1. Visual Studio の起動
 - 3.1.2. チュートリアルプロジェクトの作成
 - 3.1.3. MFUnit テストの作成
 - 3.1.4. MFUnit テストの実行
- 3.2. コマンドラインからの実行

3. チュートリアル手順の概要

3.1. IDE からの実行

3.1.1. Visual Studio の起動

1) スタートメニューより、Visual Studio 2022 を起動します。

3.1.2. チュートリアルプロジェクトの作成

- 1) [新しいプロジェクトの作成]をクリックします。
- 2) 言語に "COBOL"、プロジェクトの種類は "テスト" を選択し、「ユニットテストライブラリ(.NET Framework)」を選択した 上で、[次へ(N)] ボタンをクリックします。

COBOL	-	Windows	-	ታスト	-
CBL	ユニット テスト ライブラリ MFUnit テスト ライブラリ COBOL Windows	(.NET Framework) を作成するための .NET プ デスクトップ テスト	ロジェクトです。 、		
	ユニット テスト ライブラリ MFUnit テスト ライブラリ COBOL Windows	を作成するためのネイティン ネーティブ テスト	ブロジェクトです。 ライブラリ		
		探しているものが見つか さらにツールと機能をイン	らない場合 ノストールする		
				戻る(B)	次へ(N)

3) プロジェクト名に "VCTutManageMFUnit"を入力し、[作成(C)] 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			
ユニット テスト ライブラリ (.NET Framework) COBOL Windows デスクトップ テスト			
プロジェクト名(J)			
VCTutManageMFUnit			
場所(L)			
C:¥work¥ManageMFUnit¥ •			
ソリューション名(M) ①			
VCTutManageMFUnit			
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)			
フレームワーク(F)			
.NET Framework 4.7.2 🔹			
プロジェクトは "C:¥work¥ManageMFUnit¥VCTutManageMFUnit¥VCTutManageMFUnit¥" で作成されます			
戻る(B)	f	乍成(C)	

新規プロジェクトが作成されます。



 4) VCTutManageMFUnit ソリューション名を選択した状態で、マウスの右クリックにてコンテクストメニューを表示し、[追加 (D)] > [新しいプロジェクト(N)] を選択します。

© ⊂ 🏠 🛱 - To - S @ 🗿 🗡 🗕 13
リューション エクスプローラー の検索 (Ctrl+:)
🕡 ソリューション 'VCTutManageMFUnit' (1/1 プロジェクト)
CTutManageMFUnit Properties

5) 言語に "COBOL" を選択し、プロジェクトの種類は「ライブラリ」を選択し、表示された一覧から「クラスライブラリ (.NET Framework)」を選択し、[次へ(N)] ボタンをクリックします。



6) プロジェクト名に "AircodeLibrary" と入力し、[作成(C)] 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	
クラス ライブラリ (.NET Framework) COBOL Windows ライブラリ	
プロジェクト名(J)	
AircodeLibrary	
場所(L)	
C:¥work¥ManageMFUnit¥VCTutManageMFUnit •	
ンレームノーフ(F) .NET Framework 4.7.2 ・ プロジェクトは "Ci¥work¥ManageMFUnit¥VCTutManageMFUnit¥AircodeLibrary¥" で作成されます	
	戻る(B) 作成(C)

プロジェクトが作成されます。



AircodeLibrary プロジェクト配下の Class1.cbl は不要のため、削除してください。

AircodeLibrary プロジェクト名を選択した状態で、マウスの右クリックにてコンテクストメニューを表示し、[追加(D)] > [既存の項目(G)]を選択します。

ソリューション エクスプローラー					
◎ ◎ 🏠 📩 • 🐻 • ≒ 🗗 🗿					
ソリューション エクスプローラー の検索 (Ctrl+:)					ビルドの依存関係(B)
👦 ソリューション 'VCTutManageMFUnit'	*ם	新しい項目(W)	Ctrl+Shift+A		追加(D)
GIL AircodeLibrary	t a	既存の項目(G)	Shift+Alt+A	1	既存の COBOL 項目を追加
▶ Properties ▶ ■■ 参照設定	*-	新しいフォルダー(D)		Ť.	NuGet パッケージの管理(N)
VCTutManageMFUnit		参照(R)		Ф	スタートアップ プロジェクトに設定(

サンプルファイルを展開したフォルダ内の AirportDemoMFUnit フォルダ配下を選択し、"すべてのファイル(*.*)"を選択した
 た結果、表示される全てのファイルを選択した上で、[追加(A)] ボタンをクリックします

				0
7 11	百戒口時	価格	#17	8== ▼
	文初口时	12.72	91.4	
aircode	2019/05/15 22:22	COBOL ソースファイル	8 KB	
AirCodeWrapper	2018/11/29 14:40	COBOL ソースファイル	4 KB	
airparams	2019/05/15 22:22	COBOL コピーブック	2 KB	
airports.dat	2019/05/15 22:22	DAT ファイル	670 KB	
airrec	2019/05/15 22:22	COBOL コピーブック	1 KB	
e) readme	2019/05/15 22:22	XML ドキュメント	2 KB	
_				
N): "aircode" "AirCodeWrapper" "airparams" "	airports.dat" "airrec" "r	eadme"	✓ すべての :島力D	(7ァイル (*.*)

プロジェクトが、以下のようになります。



aircode.cbl はレガシーなネイティブ COBOL プログラムです。AirCodeWrapper.cbl は .NET 言語から容易に COBOL プログラムを呼び出すための Wrapper クラスです。

 AircodeLibrary プロジェクト名を選択した状態で、Visual Studio IDE メニューより、[ビルド(B)] > [AircodeLibrary のビルド(U)] を選択します。

ĽIV	ŀč(B)	デバッグ(D)	テスト(S)	分析(N)	ツール(T)			
.	בעע	ーションのビルド	(B)	Ct	rl+Shift+B			
	בעצ	ーションのリビル	۲̈́(R)					
	ソリューションのクリーン(C)							
	ソリューションでコード分析を実行(Y) Alt+F11							
*	Airco	odeLibrary のど	゚゚゚ルド(U)	Ct	rl+B			
AircodeLibrary のリビルド(E)								
	AircodeLibrary のクリーン(N)							

プロジェクトのビルド処理が行われ、DLL が作成されます。

出力	
出力元(S): ビルド	- <u>-</u> <u>-</u> = <u>-</u> = <u>-</u> = <u>-</u>
1> ビルド開始: プロジェクト: AircodeLibrary,構成 1> * C:¥Users¥tarot¥source¥repos¥VCTutManageMFUnit¥Air 1> * C:¥Users¥tarot¥source¥repos¥VCTutManageMFUnit¥Air 1> * 生成中 aircode 1> * 生成中 ManagedAirportLinkLibrary.AirCodeWrapper 1> AircodeLibrary -> C:¥Users¥tarot¥source¥repos¥VCTut ========= ビルド: 正常終了または最新の状態 1、失敗 0、	: Debug Any CPU codeLibrary¥aircode.cbl のコンパイル中 codeLibrary¥AirCodeWrapper.cbl のコンパイル中 ManageMFUnit¥AircodeLibrary¥bin¥Debug¥AircodeLibrary.dll スキップ 0 ========

3.1.3. MFUnit テストの作成

さきほど追加した AircodeWrapper.cbl に対する単体テストを作成します。

1) VCTutManageMFUnit プロジェクト配下の「参照設定」を選択した状態で、マウスの右クリックにてコンテクストメニューを

表示し、[参照の追加(R)] を選択します。



2) プロジェクトタブを選択し、AircodeLibrary 項目のチェックを行った後、[OK] ボタンをクリックします。

参照の追加				?	×
▶ アセンブリ			検索 (Ctrl+E)		ρ-
▲ プロジェクト	名前 【】 Aircraft library	パス	名前:		
ソリューション	AircodeLibrary	C:#Users#tarot#source	AircodeLibrary		
▶ COM					
▶ 参照					
		参照(B) OK	キャンも	211

参照設定に、AircodeLibrary が追加されます。



3) VCTutManageMFUnit プロジェクト名を選択し、[追加(D)] > [新しい項目(W)] を選択します。

▲ I VCTutManageMFUnit		Saladin. Site Albumi Site Albumi	ビルドの依存関係(B)
🔑 Properties	*□ 新しい項目(W)	Ctrl+Shift+A	追加(D)
▶ ■■ 参照設定	* 回 既存の項目(G)	Shift+Alt+A	既存の COBOL 項目を追加

4) 「テスト プログラム」を選択し、[追加(A)] ボタンをクリックします。

新しい項目の追加 - VCTutManageMFUnit						? ×
▲ インストール済み	並べ替え	; 既定	· # E			検索 (Ctrl+E) ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・
▲ COBOL プロジェクト項目 Code		COBOL クラス		COBOL プロジェクト項目		種類: COBOL プロジェクト項目 テスト セットアップ、テスト ティアダウンおよびテスト
General Test	D	COBOL プログラム		COBOL プロジェクト項目	I	ケースのエントリ ポイントを含む MFUnit テスト プ ログラムです。
WCF Windows Forms	••	インタフェース		COBOL プロジェクト項目	I	
WPF	D	コピーブック		COBOL プロジェクト項目	I	
▶ オンライン	Z	テストプログラム		COBOL プロジェクト項目		
	==	Windows フォーム		COBOL プロジェクト項目	I	
		ログイン フォーム		COBOL プロジェクト項目	I	
		ב-ש-עכ -ש		COBOL プロジェクト項目	l	
		Installer クラス		COBOL プロジェクト項目	l	
	O.	WCF サービス		COBOL プロジェクト項目		
		アイコン ファイル		COBOL プロジェクト項目		
		ウィンドウ (WPF)		COBOL プロジェクト項目		
	N	カーソル ファイル		COBOL プロジェクト項目		
	~~	ビットマップ ファイル		COBOL プロジェクト項目	-	
名前(N): TestProgram1.cbl						
						追加(A) キャンセル

- 5) 新規のテストケース (羽田・ロンドンヒースロー空間間の距離 (km) のテスト) を追加した上で、実行を行ないます。サンプ ルファイルを展開したフォルダ内の TestProgram1.cbl で、現在の TestProgram1.cbl を上書きしてください。 これは、テストケース "testDistance" を途中まで作成したものになります。プログラムを確認すると、MFU-TC-SETUP-PREFIX, MFU-TC-PREFIX, MFU-TC-TEARDOWN-PREFIX から始まる "testDistance" の 3 entry が定義さ れていることが分かります。MFUnit では、テストを下記のように決められた手順で実行しています。
 - ① entry MFU-TC-SETUP-PREFIX & "testDistance"
 - ② entry MFU-TC-PREFIX & "testDistance"
 - ③ entry MFU-TC-TEARDOWN-PREFIX & "testDistance"

MFU-TC-SETUP-PREFIX で始まる entry にて、テストの前処理を定義できます。前処理の代表例としては、ファイルを あらかじめオープンしておくなどが考えられます。一方、MFU-TC-TEARDOWN-PREFIX で始まる entry では、テスト実 行後の処理を定義できます。前処理でオープンしたファイルをクローズするような処理が該当します。前処理、後処理ともに 省略可能です。

注意)

25行目に、airports.datへのパスが指定されているため、環境に合わせて修正してください。

COBOL 開発: Visual Studio - .NET COBOL の単体テスト

テスト本体である MFU-TC-PREFIX を確認すると、下記のように結果検証コードが実装されていません。

```
entry MFU-TC-PREFIX & "testDistance".
    *> set environment variable
    display "dd_airports" upon environment-name.
    display """C:¥path-to-datafile¥airports.dat""" upon environment-value.
    declare clazz = new AircodeLibrary.AirCodeWrapper() ;
    invoke clazz::OpenFile() ;
    invoke clazz::GetDistance("HND", "LHR") returning distance-km.
    invoke clazz::CloseFile() ;.
```

このテストを実装するため、以下のコードを上記 entry 句の最終位置に含まれるよう、挿入してください。

```
set exp-distance-km to 9591.
if exp-distance-km = distance-km
then
goback returning MFU-PASS-RETURN-CODE
else
string
"expected "
exp-distance-km
", but "
distance-km
into err-msg
end-string
call MFU-ASSERT-FAIL-Z using err-msg
end-if.
```

```
補足)
```

```
テスト失敗時の記述方法として、成功時同様に、戻り値で返す方法は、以下の通りです。
         if exp-distance-km = distance-km
         then
            goback returning MFU-PASS-RETURN-CODE
         else
            string
               "expected "
               exp-distance-km
               ", but "
               distance-km
               into err-msg
            end-string
            display err-msg
            goback returning MFU-FAIL-RETURN-CODE
         end-if.
戻り値 MFU-FAIL-RETURN-CODE を利用する場合、テスト結果を確認するためにエラー情報を、display などで出
力する必要があります。
```

3.1.4. MFUnit テストの実行

本テスト対象のプログラムは、環境変数で設定された空港情報が保存されたデータファイルを参照するため、手順内で設定を行い ます。

1) Visual Studio IDE メニューより、[表示(V)] > [Micro Focus Unit Testing] を選択します。



2) Micro Focus Unit Testing ビューより、[全て実行] ボタンをクリックします。

Micro Focus Unit Testing				
▶ コードカバレッジで実行	▶ すべて実行	直前を実行	▶ 失敗を実行	▶ 選択を実行
				テスト結果

以下のように全て緑色のマークが設定されます。緑色は、テストに成功したことを示します。



3) エラーケースを確認します。「TestProgram1.cbl」をエラーとなるように修正した上で、再度、Micro Focus Unit Testing ビューより、「全て実行」ボタンをクリックします。

なお、本例では、3.1.3 で作成したテストプログラム内に記載されていた期待値 9591 を 9590 に修正しています。

```
entry MFU-TC-PREFIX & "testDistance".
    *> set environment variable
    display "dd_airports" upon environment-name
    display """C:¥Users¥Public¥Documents¥Micro

    declare clazz = new AirCodeWrapper() ;
    invoke clazz::OpenFile() ;
    invoke clazz::CloseFile() ;
    set exp-distance-km to 9590.
    if exp-distance-km = distance-km
        then
            goback returning MFU-PASS-RETURN-CODE
    else
        string
            "exp-distance-km
            into err-msg
            end-string
            call MFU-ASSERT-FAIL-Z using err-msg
    end-if.

Micro Focus UnitTesting
```

MFUT_testDistance のテストで、エラーが発生したことが一覧から判断できます。

▶ すべて実行 (コードカバレッジ) ▶ すべて実行 ▶ 直前を実行 ▶ 失敗を実行



3.2. コマンドラインからの実行

MFUnit によるテストは、Visual Studio 上の画面からではなく、コマンドライン上からも行なうことができます。従来のスタイルでの テスト作業の効率化を図ることができ、Jenkins などの CI ツールと連携する事で、テストの自動実行を行なえるため、品質担保 や作業工数の削減が見込めます。

ここでは、3.1.1 で作成したテストプログラムをコマンドラインから実行する方法について学びます。

1) スタートメニューより、実行環境に合わせた [Visual COBOL コマンドプロンプト] をクリックします。



2) 作業フォルダを作成し、作成したフォルダに移動します。

C:¥>mkdir VCCommandTutorial C:¥>cd VCCommandTutorial C:¥VCCommandTutorial>

- 3) 3.1 で作成したソリューションが保存されているフォルダを VS_SOLUTION_PATH に指定した上で、下記コマンドを実行 します。
 - set VS_SOLUTION_PATH=c:¥vs_solution_path
 - cobol %VS_SOLUTION_PATH%¥AircodeLibrary¥AircodeWrapper.cbl
 ilsource(%VS_SOLUTION_PATH%¥AircodeLibrary¥aircode.cbl) iloutput(.) ilgen(sub);
 - cobol %VS_SOLUTION_PATH%¥VCTutManageMFUnit¥TestProgram1.cbl iloutput(.) ilgen(sub) sourceformat(variable) ilref(AircodeWrapper.dll);
 - 注意)

VS_SOLUTION_PATH は、各環境に合わせて修正してください

C: ¥VCCommandTutorial > set VS_SOLUTION_PATH=c: ¥vs_solution_path

C:¥VCCommandTutorial>cobol %VS_SOLUTION_PATH%¥VCTutManageMFUnit¥TestProgram1 .cbl iloutput(.) ilgen(sub) sourceformat(variable) ilref(AircodeWrapper.dll);

Micro Focus COBOL

Version 9.0 (C) Copyright 1984-2023 Micro Focus or one of its affiliates.

* チェック終了:エラーはありません

C:¥VCCommandTutorial>cobol %VS_SOLUTION_PATH%¥VCTutManageMFUnit¥TestProgram1 .cbl iloutput(.) ilgen(sub) sourceformat(variable) ilref(AircodeWrapper.dll);

Micro Focus COBOL

Version 9.0 (C) Copyright 1984-2022 Micro Focus or one of its affiliates.

・チェック終了:エラーはありません

以下のファイルが作成されていることを確認します。

C:¥VCCommandTutorial>dir										
ドライブ C のボリューム ラベルがありません。										
ボリューム シリアル番号は 969A-9F98 です										
C:¥VCCommandTutorial のディレクトリ										
2023/10/04	16:03	<dir></dir>								
2023/10/04	16:03	<dir></dir>								
2023/10/04	16:02		14,336	AircodeW	/rapper.dll					
2023/10/04	16:03		8,704	TestProgr	am1.dll					
	2 個のファ	ァイル		23,040	バイト					
	2 個のデ	ィレクトリ	34,364,	583,936	バイトの空き領	貢域				

- 4) プロンプト上で下記コマンドを実行し、MFUnit を実行します。
 - set dd_airports=%VS_SOLUTION_PATH%¥AircodeLibrary¥airports.dat
 - mfurunil -report:junit -outdir:mfunit_result TestProgram1.dll

補足)

.NET COBOL プログラムに対する単体テストは、mfurun ではなく、mfurunil を使用します。



outdir オプションにより、出力結果を mfunit_result フォルダに保存されていることを確認してください。



WHAT'S NEXT

● 本チュートリアルで学習した技術の詳細については製品マニュアルをご参照ください。

免責事項

ここで紹介したソースコードは、機能説明のためのサンプルであり、製品の一部ではございません。ソースコードが実際に動作するか、御社業務に適合するかなどに関しまして、一切の保証はございません。 ソースコード、説明、その他すべてについて、無謬性は保障されません。 ここで紹介するソースコードの一部、もしくは全部について、弊社に断りなく、御社の内部に組み込み、そのままご利用頂いても構いません。 本ソースコードの一部もしくは全部を二次的著作物に対して引用する場合、著作権法の精神に基づき、適切な扱いを行ってください。