

# Visual COBOL チュートリアル

# RESTful Web サービスによる COBOL 資産の再利用

### Visual Studio 編

#### 1 目的

Visual COBOL に付属する COBOL 専用のアプリケーションサーバー「Enterprise Server」は、ネイティブにコンパ イルした COBOL のビジネスロジックを REST API を利用し Web サービスとして呼び出す機能を提供しています。 RESTful の Web サービスとして呼び出しを行う場合、JSON 形式でやり取りが可能であれば呼び出し側のプログラムに 依存することなく連携できるようになります。

このドキュメントでは COBOL のソースコードに一切手を加えることなくビジネスロジックとして Enterprise Server にディプロイし、それを Visual COBOL のクライアント生成機能を使って動作確認用のクライアントを作成し連携する方法を説明します。

#### 2 前提

本チュートリアルは、下記の環境を前提に作成されています。

開発クライアント ソフトウェア
 OS: Windows 11
 COBOL 開発環境製品: Visual COBOL 10.0 for Visual Studio 2022
 IDE: Microsoft Visual Studio Professional 2022 (17.7.4)?

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、任意のフォルダー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
  - 3.1 Windows クライアントでの開発準備作業
  - 3.2 Enterprise Server の設定変更
    - 3.2.1 不要ログの抑止
    - 3.2.2 リスナー構成の変更
    - 3.2.3 Enterprise Server の環境設定
  - 3.3 Enterprise Server の起動
  - 3.4 RESTful Web サービスの開発作業
  - 3.5 Enterprise Server ソリューションのビルド作業
  - 3.6 באלראוטל COBOL האיז באליער באפט באליער באנג באליער ב
  - 3.7 RESTful Web サービスのテスト
    - 3.7.1 RESTful Web サービスクライアントプログラムの利用
    - 3.7.2 Curl コマンドによるテスト
  - 3.8 RESTful Web サービスのデバッグ
  - 3.9 Enterprise Server の停止

**Rocket** software

#### 3 チュートリアル手順

#### 3.1 Windows クライアントでの開発準備作業

- 1) Visual COBOL for Visual Studio を起動
  - [スタート] メニュー > [すべてのアプリ] > [Visual Studio 2022] を選択します。
- 2) [Enterprise Server アプリケーション] プロジェクトの作成
  - ① 「新しいプロジェクトの作成」を選択します。

開始する

<b>→</b>	<b>リポジトリのクローン(C)</b> GitHub や Azure DevOps などのオンライン リボジトリ からコードを取得します
ď	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 .sln ファイル を開きます
Ľ	<b>ローカル フォルダーを開く(F)</b> 任意のフォルダー内のコードに移動して編集します
<del>ث</del>	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

 ② 「新しいプロジェクトの作成」ウィザードが表示されるので [言語] を「COBOL」、[プラットフォーム] を 「Windows」、[プロジェクト タイプ] に「Enterprise Server」を選択し、[次へ(N)] をクリックします。

COBOL	• Windows	✓ Enterprise Server
Enterprise Server Micro Focus Serve す。 COBOL Wind	アプリケーション er 配下で実行される COBOL ア dows Enterprise Server	プリケーションを作成するためのプロジェクトで
WSDL or JSON/YA WSDL または JSON するためのプロジェク COBOL Wind	AML から作成する Web サービス J/YAML スキーマ ファイルから We トです。 dows Enterprise Server	(クライアント アプリケーション eb サービスクライアント アプリケーションを作成 Native コンソール
	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 ストールする
		戻る(B) 次へ <b>(N)</b>

③ [名前(N)] フィールドに "RESTfulCOBOL" を入力して、[OK] をクリックします。[場所(L)] は任意のフォルダー を指定し、[ソリューション名] は、デフォルトのままとし、[作成(C)] をクリックします。



Enterprise Server アプリケーション COBOL Windows Enterprise Server	
カジェクト名(J)	
RESTfulCOBOL	
易所(L)	
C.¥Users¥tarot¥source¥repos •	
/リューション名(M) 🕕	
RESTfulCOBOL	
クリューンヨンとフロンエクトを回し ティレクトット 配置する(U)	
」 クリニーションとフロフエクトを回しティレクトンに 配置する(U) 行Dジェクト は "C:¥Users¥tarot¥source¥repos¥RESTfulCOBOL¥RESTfulCOBOL¥" で作成されます	
ウリーションとフロクリケを回じアイレクトウに 配置 9 @(D) 知ジェクト は "C:¥Users¥tarot¥source¥repos¥RESTfulCOBOL¥RESTfulCOBOL¥" で作成されます	
ウリーションとフロウエシトを回しアイレクトウに 配当する(U) 加ジェクト は "C:¥Users¥tarot¥source¥repos¥RESTfulCOBOL¥RESTfulCOBOL¥" で作成されます	

- 3) コンパイラオプションの指定とソースコードのインポート
  - ① 作成されたプロジェクトの Properties をダブルクリックします。



② [COBOL] をクリックし、画面を下にスクロールして、[追加指令] に "ASSIGN(EXTERNAL)" を指定し、保存したうえで画面を閉じます。

RESTfulCOBOL* 🗢 🗙			
アプリケーション SQL	構成(C): アクティブな (Debug	」) ✓ プラットフォーム(M): アクティブな (x64) ✓	
デバッグ	警告レベル:	回復可能なエラーを含める(レベル E) く	
コピーブック	最大エラー数:	100	
COBOL	出力 ————		
COBOL リンク	11 + 187.	VhisVsCAVDahuaV	
Micro Focus Code Analysis	田力八人:	_+DIN#X04#Deblug# 参照	
	□ 指令ファイルの生成	□ リストファイルを生成	
	□ コード カバレッジを有効	にする □ プロファイラを有効にする	
	追加指令    —		
	ASSIGN(EXTERNAL)		
			•

③ エクスプローラーを起動し、サンプルのソースコードを解凍したフォルダーから "BOOK-INFO.cpy" と "BOOK.cbl" をプロジェクトフォルダーにドラッグアンドドロップします。

4

④ ソリューションエクスプローラーから、2つのファイルが正常にロードされていることを確認します。



ソリューション エクスプローラー	- ¶ ×
v	
ソリューション エクスプローラー の検索 (Ctrl+:)	- م
戻 ソリューション 'RESTfulCOBOL' (1/1 のプロジェ	クト)
▲ CBL RESTFUICOBOL	
🎤 Properties	
BOOK.cbl	
BOOK-INFO.cpy	

#### 3.2 Enterprise Server の設定変更

- 1) ディプロイ用フォルダーを作成します。
  - ① 「RESTfulCOBOL」プロジェクト上で右クリックし、コンテクストメニューから [追加(D)]→[新しいフォルダー(D)] を 選択します。
  - ② フォルダー名に "deploy" を指定します。
- 2) 「.mfdeploy」ファイルをインポートします。
  - ① 作成した「deploy」フォルダー上で右クリックし、コンテクストメニューから [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	<u>5</u> -			• <b>∓</b> ₽ ×		
		🖉   💿 = 🖘 🗇 🖆   🌮 🚔   🔻						
		ソリューション エクスプロー	ラーの検索 (Ctrl+:)			- م		
		😽 ソリューション 'RES	TfulCOBOL' (1/1 のプロジェク	<del>۱</del> )				
		RESTfulCOBOL						
		🔑 Properties						
		🛅 deploy						
Ŷ٦	新山頂	」項目(W) Ctrl+Shift+A 追加(D)						
t	□ 既存の項目(G) Shift+Alt+A				ここまでスコープ指定する(S)			
	省 新しいフォルダー(D)			<b>☆</b> ⊐	新しいソリューション エクスプローラーのビュー(N)			

② [すべてのファイル(\*.\*)] に変更した上で、Visual COBOL インストールフォルダー¥deploy 配下にある



- 3) ESCWA 管理画面の設定
  - サーバーエクスプローラーを表示し、[Micro Focus Server」] > [Default] 上で右クリックし、コンテクストメニューから[管理(A)]を選択します。



<u>ا</u> –لا	サーバー エクスプローラー		<b>★</b> +⊐ X	
レポック	Ŭ× \$9\$\$1\$ ₿	e		<ul> <li> <sup>2</sup> ВООК</li> </ul>
גל	Micro Focus Anal	ysis S	erver	*****
S	Micro Focus Serve	er		S INTERNATIONAL I
Lser	▷ 📓 Default ▲ 📑 サーバー		管理(A)	
ver オ	▷ 🗍 win11-v-na		削除(D)	
55	育 データ接続		Enterprise Server を新規	見作成(N)
1 7 1			認証情報をクリア(C)	
エクス			フィルタ(F)	
לם-י		U	最新の情報に更新	
<sub>ラー</sub> サーバー エクスプローラー				D S B-TITLE WITH D S B-AUTHOR WITH I

認証画面が表示された場合は認証情報を入力します。

補足)

インストール直後に有効となっている際の認証情報は以下の手順で確認できます。

スタートメニューより、[Micro Focus Visual COBOL] > [Visual COBOL コマンドプロンプト(64 ビット)] を選択します。

コマンドライン上で "mfsecretsadmin read microfocus/temp/admin"のコマンドを実行します。

以下の場合、ユーザー名は "SYSAD"、パスワードは "LZQe4VS3" です。

C:¥>mfsecretsadmin read microfocus/temp/admin {"mfUser":"SYSAD", "mfPassword":"LZQe4VS3"} C:¥>

② ログイン画面が表示されますので、上記と同じ認証情報を入力して、[ログイン]をクリックします。



ES

● 言語 ∨

# Enterprise Server Common Web Administration

▲ Micro Focus Enterprise Serverでは、インストール後に基本 的なセキュリティ機能がデフォルトで有効になっています。 詳細情報			
ユーザー名			
SYSAD			
パスワード			
パスワード変更 ログオン			

#### 3.2.1 不要ログの抑止

1) [プロパティ] タブをクリックし、画面をスクロールしたのち、モニターセクションの [有効] のチェックを外したうえで、画 面上部にある [適用] をクリックします。



# **Rocket** software

- 3.2.2 リスナー構成の変更
  - 1) [ネイティブ] タブをクリックします。

ES	管理	ダッシュボー	۲	ネイティブ	メインフレーム	セキュリテ	イ
ネイテ	ィブナビゲ	ーション	^	リージョン	ンおよびサーバー	プロパティ	~

2) 次に左側メニューの [Directory Server] をドリルダウンして、[Default] → [ESDEMO64]をクリックします。



3) [一般]メニューが表示されるので横にある下向き記号をクリックし、[リスナー]を選択します。

一般	~	
一般的	プロパティ	
	コントロール	
8844	検証	_
開炉	リスナー	
名前*	サービス	7

4) [通信プロセス1] > [Web] をクリックします。 ネイティブ リスナー ナビゲーション ^



リスナープロパティが表示されます。横のスライドバーを下に下げていくとのカスタム構成情報が表示されています。デ フォルトは「uploads=<ES>/deploy」となっています。この指定により、Visual COBOL インストールディレク トリ配下の deploy フォルダーがディプロイ用フォルダーとして使用されます。通常、Program Files (x86)等のフ ォルダーは管理者権限を持つユーザーしか書き込みできないため、さきほど作成した deploy フォルダーに変更しま す。

例: uploads=C: Users U



カスタム構成 💡

```
[virtual paths]
cgi=<ES>/bin
uploads=C:\Users\tarot\source\repos\RESTfulCOBOL\RESTfulCOBOL\deploy
<default>=/dev/null
```

上記は、プロジェクトフォルダーが C:¥Users¥tarot¥source¥repos¥RESTfulCOBOL¥RESTfulCOBOL の場合です。

入力が終わったらスライドバーを上に移動して、[適用]をクリックします。

5) [Web Services and J2EE]のポート番号の変更を行います。

[通信プロセス1] > [Web Services and J2EE] をクリックし、ポート番号を「\*」から「9003」に変更します。 最後に、スライドバーを上に移動して、「適用] をクリックします。

<b>ES</b> 管理   グッシュボード	<b>ネイティブ</b> メインフレーム セキュリティ	Q. D. 🖉 & SYSAD 🗸 U6.0.33
ネイティブナビゲーション ^	一般  ~	ESDEM064 (Default)    と ユーザー 🗸
<ul> <li> プループ</li> <li>&gt; 論理</li> <li>&gt; PAC</li> </ul>	リスナー プロパティ <u>湖</u> 🗎 🧃 溯除	C
<ul> <li>✓ Image: Directory Server</li> <li>✓ Image: Image: Optimized Content</li> <li>✓ Image: Optimized Content</li></ul>	* 入力必須の項目です 名前* ♀	
ESDEMO64	Web Services and J2EE	🔲 レガシー Micro Focus アブリケーション形式 💡
> 🕀 SOR	6 このエンドポイントはネットワーク経由でアクセス可能になり、TLSが無効に	なります。
ネイティブ リスナー ナビゲーション 🏻 ^	プロトコル* 💡 ホスト名またはIP アドレス* 💡	#-r 8
通信サーバーの新規作成	tcp ~	9003
✓ 団 通信プロセス1 #ff HTTP Echo #ff Web #ff Web Services and J2EE	+ TLSBDE	

#### 3.2.3 Enterprise Server の環境設定

1) [一般]をクリックし、[追加設定]フィールドに下記の命令を追加したうえで、[適用] をクリックします。

※ サンプルファイルを c: ¥vs-tutorial に解凍した場合です。環境に合わせてパスを修正してください。



追加設定

構成情報 💡

```
[ES-Environment]
BOOKINFO=C:\vc-tutorial\DAT\BOOKINFO.DAT
```



#### 3.3 Enterprise Server の起動

- 1) Visual Studio に戻ります。[表示(V)] > [サーバーエクスプローラー(V)] を選択します。
- [Micro Focus Server] > [localhost] > [ESDEMO64] と展開します。「ESDEMO64」の上で右クリックし、コン テクストメニューから[開始(S)]を選択します。

サーバーエクスプローラー 🔹 🚽 🗙	
() ×   변 별 1   12	• 😤 воок
Micro Focus Analysis Server	****
<ul> <li>Micro Focus Server</li> </ul>	S INTERNATI(
Default	
SDEMO	
ESDEMO64	*****
▲ 目 サーバー 開始(S)	

[出力] に起動メッセージが表示されます。



正常に開始されると [サーバーエクスプローラー] 上の ESDEMO64 アイコンが起動されたことを示す緑色のアイコンに切り替わります。



#### 3.4 RESTful Web サービスの開発作業

- 1) RESTful Web サービスのプロファイル作成
  - 「RESTfulCOBOL」プロジェクトを右クリックし、コンテクストメニューから [追加(D)]> [新しい項目(W)] を選択します。

<u>*</u> × <del>•</del> ≎	ソリューション エクスプローラー	i i i i i i i i i i i i i i i i i i i	ビルド(U)	
÷	🖉 🐚 • 🖨 🗇 🎤 🛋	Ψ	リビルド(E)	
	ソリューション エクスプローラー の検索 (Ct	trl + :)	クリーン(N)	
	🕞 ソリューション 'RESTfulCOBOL' (	1/1 のプロジ: 🗉	すべての子孫を折りたたむ	Ctrl+左矢印
197 1:	▲ 💷 RESTfulCOBOL		ここまでスコープ指定する(S)	
-	Properties	<del>ث</del>	新しいソリューション エクスプローラーのビュー(N)	
1 新しい項目	∃(W) Ctrl+	Shift+A	追加(D)	•
し既存の項	目(G) Shift-	+Alt+A	既存の COBOL 項目を追加	

② [COBOL] > [Native] アイテムを選択し、[Service Interface] を指定します。

▲ インストール済み	並べ替え:	既定	•	 :=	]
<ul> <li>COBOL</li> <li>Code</li> </ul>	*}* *}*	Service Interface			COBOL
General Native					
▶ オンライン					

③ [名前] に "RESTfulBOOK.svi" を指定し、[追加(A)] をクリックします。



 ④ [サービスインターフェイス] 画面が表示されるので、[インターフェイスの種類] に「Web サービス」を選択し、[転送 形式]は、「JSON (RESTful)」を選択し [OK] をクリックします。

名前:	
RESTfulBOOK	
インターフェイスの種類:	
Web サービス	~
転送形式	
O SOAP	
🔾 JSON (RESTful)	
ソースの種類:	
COBOL プログラム	$\sim$
ОК	キャンセル

[RESTfulBOOK] のオペレーション初期画面が表示されます。



RESTfulBOOK.svi 🏾 🗢 🗙							
Operation		~					
インターフェイスフィールド 名前	- Input 方向	COBOL エンド 名前	リポイント PICTURE	日 日 日 日 日 日 日 日	インターフェイスフ・ 前	ィールド - Output 方向	型
COBOL 割当て COBOL Field Value					再利用フィー 名前 4	ールド	þ

- 2) 書籍検索用のオペレーションを作成
  - 下図のように Visual Studio 内で「RESTfulBOOK.svi」が開いている状態で [拡張機能] > [オペレーション

(P)] メニュー	>	[新規作成(N)]	を選択します。
-----------	---	-----------	---------

ファイル(F) 編集(E) 表示(V)	) Git(G) プロジェク	クト(P) ビルド(B)	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)	ヘルプ(H)	🔎 検索 ▼	RESTfulCOBOL
) • 🕘   🛅 • 🚅 💾 📳   🔈	- 🤍 - Debug	▼ x64	- 🕨 RESTfr	uICOBOL - ▷	📭   🗖 📮	オペレーショ	ン(P)	<u> </u>	新規作成(N)	
						フィールド(I)		•	再配置(R)	
RESTfulBOOK.svi → ×						マッピング(A	0	•	条件を編集(E)	· ·
Operation		~				割当て(S)		•	削除(D)	
						四 拡張機能の	)管理(M)		プロパティ(P)	
						メニューの力	スタマイズ(C)		_	
インターフェイスフィー	ルド – Input	COBOLエン	トリポイント		ンターフェ	イスフィール	F - Output			
		名前	PICTURE							
名別	方向			1 4	80		万回	型		

 ② [オペレーションプロパティ] ダイアログが表示されるので [名前] に "SearchBOOK" を入力し、[プログラム/コピー ブックを選択] では、「BOOK」を選択、[Select entry point] も「BOOK」を選択し、[OK] をクリックします。

全般 パス/HTTP ユーザー出口	API リソース (無効)	
名前 SearchBOOK		
プログラム/コピーブックを選		
BOOK-INFO		
Select entry point (parameter	s shown at right):	
ВООК	Name	Picture
	LNK-FUNCTION	X
	LNK-FILE-STATUS	X(2)
答 敢 味 フッピンガ 友 作 成・	APIリソースを有効にする	
	ALL Y MERMEYS	



- 3) COBOL と RESTful Web サービス間の変数型変換マッピングを定義(書籍情報検索用オペレーション)
  - ① COBOL エントリポイントが展開されますので「LNK-FUNCTION」を [COBOL 割当て] ペインにドラッグ&ドロップ します。



② [値] フィールドには "1" を入力し、[OK] をクリックします。

名前:	LNK-FUNCTIO	N		~
値:	1			
			ОК	キャンセル

③ 次に「LNK-B-STOCKNO」を [インターフェイスフィールド - Input] ペインにドラッグ&ドロップします。

KESTIUBOOK.	SVI ~ ~ X						
Operation	SearchBOOK		✓ HTTP Method:	POST Path:	ogram: BOOK		
インター	フェイスフィールド -	Input	COBOL エントリポイン 名前	ト: BOOK PICTURE	インターフェイスフィール	ド - Output	Ŧ
名前	4	方向	LINK-FUNCTION     LINK-B-DETAILS     LINK-B-TEXT-DET     LINK-B-TITLE     LINK-B-TITLE     LINK-B-TYPE     LINK-B-ATITLE     LINK-B-STOCKNO     LINK-B-STOCKNO     LINK-B-CONHAND     LINK-B-SOLD     LINK-FILE-STATUS	X X(50) X(20) X(50) X(4) 9(5) comp-3 9(5) comp-3 9(5) comp-3 X(2)	-C 81	(A) (C)	-
COBOL # COBOL Fiel	N발 C d Value ICTION 1 -		4	Þ	再利用フィールド       名前		

④ [インターフェイスフィールド - Input]にある「LNK-B-STOCKNO」上で右クリックから[グループ化]を選択し、[グループロパティ]ウィンドウにて "input\_root" と入力して、[OK] をクリックします。



インタ *Namele	ーフェイスフィー ess top-level value in	リレド - Input JSON message	COBOL エントリポ 名前
名前	K_B_STOCKNO		□ LNK-FUNCTION □ 日 LNK-B-DETAILS
			VI 078/1881F126(IN)
		グループ化.(G)	
名前:	input_root		
OCCUR	0		
	方向 〇 入力 〇 出力	-場所 ●ボディ ●パス ● クェリ	
		ОК	キャンセル

 ⑤「LNK-B-DETAILS」と「LNK-FILE-STATUS」を順に[インターフェイスフィールド - Output]ペインにドラッグ アンドドロップします。



⑥ [インターフェイスフィールド - Output]にある「LNK-B-DETAILS」及び「LNK-FILE-STATUS」を選択した状態で右クリックから [グループ化]を選択し、[グループプロパティ] ウィンドウにて "output\_root" と入力して、[OK]をクリックします。



インターフェイスフィール *Nameless top-level value in JS	・ド ー Output ON message		ソリューション エクスプローラー ()           マリューション 'RESTfulk           ▶ ■ RESTfulBOOKClie           ▲ ■ RESTfulCOBOL
名前	方向	型	🎤 Properties
	出力		🕨 🛅 deploy
LNK_FILE_STATUS	出力	インターフ:	ェイスフィールドの新規作成(N)
		切り取り( 貼り付け( <b>削除(D).</b>	C) (P)
		グループ化	ζ(G)



⑦ 下記の図のようになっていることを確認して、保存します。



- 4) 書籍データ追加機能のオペレーションを追加
  - ① さきほどと同じ手順で「AddBOOK」オペレーションを作成し、オペレーションの項目マッピングでは、以下の手順を実施し、保存します。
    - 「LNK-FUNCTION」項目を [COBOL 割当て] にドラッグ、値に "2" を入力
    - 「LNK-B-DETAILS」項目を [インターフェイスフィールド Input] にドラッグ ドラッグした項目からコンテクストメニューを開き、[グループ化] を選択。"input\_root" を作成
    - 「LNK-FILE-STATUS」項目を [インターフェイスフィールド Output] にドラッグ ドラッグした項目からコンテクストメニューを開き、[グループ化] を選択。"output\_root" を作成 作業完了後、以下のようになります。





- 5) プロジェクトと Enterprise Server「ESDEMO64」を関連付ける
  - サーバーエクスプローラーにて、[Micro Focus Servers] > [localhost] > [ESDEMO64] を右クリックし、コンテ クストメニューから [プロジェクトと関連付ける] > [RESTfulCOBOL] を選択します。



#### 3.5 Enterprise Server ソリューションのビルド作業

1) メニューより、[ビルド(B)] > [ソリューションのビルド(B)] を選択します。

	ビルド(B)	デバッグ(D)	テスト(S)	分析(N)	ツール(T)				
	בעי 📩	ーションのビルド	(B)	Ctr	rl+Shift+B				
Ì	 ソリューションのリビルド(R)								

#### 3.6 コンパイルした COBOL アプリケーションを Enterprise Server ヘディプロイ

- 1) ディプロイする COBOL プログラムの指定
  - ソリューションエクスプローラーにて「RESTfulBOOK.svi」を右クリックし、コンテクストメニューから [プロパティ] を選択します。
  - ② プロパティペインの [ディプロイするアプリケーションファイル] 右横の空欄をクリックしたうえで表示される [...] をクリック します。

プロパティ	₹ Д
RESTfulBOOK.svi ファイル プロパティ	
1 🛛 🎾	
サービス バージョン	1.0
サービス ベース パス	/temppath/
サービス名	RESTfulBOOK
ディプロイされたアプリケーション パス	C:¥Users¥tarot¥source¥repos¥F
ディプロイするアプリケーションファイル	
トランザクション管理	Application
ユーザ名/パスワードが必要	No
許可されるオリジン	

③ [項目の追加/削除] ウィンドウが表示されるので [ファイル追加] を押します。

項目の追加/削除するにはボタンを使い、再配置する	っには項目をドラッグ
	ファイル追加
	削除
OK	キャンセル

- ④ エクスプローラーから「VisualStudio プロジェクトフォルダー¥RESTfulCOBOL¥bin¥x64¥Debug」まで移動し、
   「BOOK.dll」及び「BOOK.idy」を指定します。ファイルが存在しない場合は、[ビルド(B)] > [ソリューションのリビルド(R)]を実行してください。
- ⑤ [項目の追加/削除] ウィンドウにファイルがセットされるので [OK] をクリックします。

項目の追加/削除するにはボタンを使い、再配置する(	こは項目をドラッグ
¥RESTfulCOBOL¥bin¥x64¥Debug¥BOOK.DLL ¥RESTfulCOBOL¥bin¥x64¥Debug¥BOOK.idy	
	ファイル追加
	削除
ОК	キャンセル

⑥ デフォルトではオリジン間リソース共有は許可されていません。もしこれに関するエラーが発生する場合、許可設定を行います。ここでは[許可されるオリジン]に "\*" を入力します。



プロパティ	▼ Д
	1
RESTfulBOOK.svi ノアイル ノロ	バテイ
□ 全般	
アプリケーションが HTTP ステー	No
エグジットポイントハンドラー名	
サービス エンドポイント URL	http://localhost:9003
サービス バージョン	1.0
サービス ベース パス	/temppath/
サービス名	RESTfulBOOK
ディプロイされたアプリケーション	C:¥Users¥tarot¥source¥repos¥RESTful
ディプロイするアプリケーションフ	C:¥Users¥tarot¥source¥repos¥RESTfu
トランザクション管理	Application
ユーザ名/パスワードが必要	No
許可されるオリジン	*

- 2) RESTful Web サービスのコンポーネント一式を Enterprise Server ヘディプロイ
  - ① 「RESTfulBOOK.svi」を右クリックし、コンテクストメニューから [検査(V)]を実行し、問題ないことを確認したうえで、

[ディプロイ(D)] 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 (Ct	rl+:)				
マンション 'RESTfulCOBOL' (1	/1 Ø	プロジェクト)			
▲ CBL RESTfulCOBOL					
🔑 Properties					
BOOK.cbl					
BOOK-INFO.cpy		RE (/ A)			
RESTfulBOOK.svi	0	開<(O)			
		ファイルを開くアプリケーションの選択(N)			
		コードのクリーンアップ	•		
ソリューション エクスプローラー Git 変更	>	タスク ランナー エクスプローラー			
プロパティ		外部ツールの構成			
RESTfulBOOK.svi Jアイル ノロバテイ ここまでスコープ指定する(S)					
🔡 🔛 🎤 🛅 新しいソリューション エクスプローラーのビュー(N)					
□ 全般		比較する(W)			
アプリケーションが HTTP ステータスを返		プロジェクトから除み(ハ			
エグジットボイントハンドラー名		ノロシェクトから味りたり			
	X	切り取り(T)	Ctrl+X		
リーヒス ハーンヨン	þ	⊐ピ–(Y)	Ctrl+C		
リーレス ハース ハス サービフ タ	$\times$	削除(D)	Del		
ディプロイされたアプリケーション パス	≡Į́i	名前の変更(M)	F2		
トランザクション管理					
ユーザ名/パスワードが必要		実行時環境を構成(C)			
許可されるオリジン		検査(V)			
許可されるオリジン		<i>ร</i> ั₁/ี่ปิๅ(D)			

② ディプロイが完了すると下図のようなメッセージが出力されます(ここでは警告は無視して構いません)。

出力
出力元(S): Generation -   呈   雪 韋   雪   準   ⑤
UU2U (2U24/U8/27 13:59:U3): Adding service and package objects to directory 0021 (2024/08/27 13:59:03): Using directory at mpri://127.0.0.1:86 0030 (2024/08/27 13:59:03): ES server "ESDEMO64" notified service "/temppath/RESTfulBOOK/1.0#SeachBOOK" is available 0030 (2024/08/27 13:59:03): ES server "ESDEMO64" notified service "/temppath/RESTfulBOOK/1.0#AddBook" is available 0030 (2024/08/27 13:59:03): Installation of package "RESTfulBOOK.car" finished with 3 warnings ディブロイメントが完了 ディブロイメントに完了
Iラー一覧 「九ジェクト詳細」 出力



- 3) ESCWA(Enterprise Server Common Web Administration) からディプロイされたことを確認
  - ① ブラウザー上の ESCWA に切り替えます。
  - ② [一般メニュー]から [サービス]をクリックします。
  - ③ 画面下にスクロールしていくと最下行にディプロイした RESTful Web サービスが追加されていることを確認します。

_	般~	モニター <b> ~</b>			⊳es	DEMO64 (Defa
サー	ビス	ハンドラおよびパッケージ				
	最終ス	<テータス ∨				
	□表	示列 🗸				
		名前	最終	リス	バツ	ハン
	ଞ	Deployer	Available	Web@C		
	ଞ	ES	Available	Web Ser		
	ල්	✓ /temppath/RESTfulBOOK/1.0				
	89°	#SeachBOOK	Available	Web Ser	/temppath	MFRHJSON
	ଞ	#AddBook	Available	Web Ser	/temppath	MFRHJSON

#### 3.7 RESTful Web サービスのテスト

#### 3.7.1 RESTful Web サービスクライアントプログラムの利用

- 1) RESTful Web サービステスト用のクライアント生成
  - ① 「RESTfulBOOK.svi」を右クリックし、コンテクストメニューから [クライアントの生成(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+:)			- م
🛛 🖂 ソリューション 'RESTfulCOBOL' (1/1 のプロジ:	፤ / בי		
<ul> <li>RESTfulCOBOL</li> </ul>			
🔑 Properties			
deploy			
BOOK.cbl	$\rightarrow$	閚<(∩)	
BOOK-INFO.cpy		ファイルを聞くアプリケーションの選択(N)	
*** RESTILIBOOK.SVI			
		コートのクリーンアップ	•
9リューション 10入ノローフー Git 変更	>	タスク ランナー エクスプローラー	
プロパティ		外部ツールの構成	
RESTfulBOOK.svi ファイル プロパティ		ここまでスコープ指定する(S)	
	<del>ث</del>	新しいソリューション エクスプローラーのビュー(N)	
□		比較する(W)	
(名前)			
1 ノジーフェ1 ス ジ1 ノ ソースタイプ	V		CLL X
□ その他	~	5J54X5(T)	Ctrl+X
ファイル名		JE-(Y)	Ctrl+C
完全パス	X	削除(D)	Del
□ 高度	Ęļ	名前の変更(M)	F2
カスタムツール		プロジェクトのデフォルトを使用	
リスダム ソールの名前空間 ビルド アクション		実行時環境を構成(C)	
出力ディレクトリにコピー		検査(V)	
		ディプロイ(D)	
(右則)		WSBIND の生成(G)	
		クライアントを牛成(G)	



- 2) テスト用アプリケーションの実行準備
  - ① RESTfulBOOK-app.cbl をダブルクリックして開きます。
  - テスト用クライアントは実行を終了すると DOS 画面から消えてしまうため、186 行目にブレークポイントを設定します。

знивоок-арр.сы 📲	KESTIUIBOOK.SVI					•
RESTfulBOOKClientApp	<ul> <li>RESTFULBOOK-APP</li> </ul>	<ul> <li>Procedure</li> </ul>	Division	-		÷
v when v when v when v end-ev call " call v usir	rform wsc-op-1 2 rform wsc-op-2 other splay "Invalid operation" aluate close-RESTfulB00K-proxy" CBL_CANCEL" g value 1 reference z"RESTfulB00K-	proxy″				
stop r	un.				2	And the state
1個の参照 V wsc-op-1	ą. •				1	
displa	y "Body Parameters:"					and the second
displa	у ″: ″				8	
displa no a	y ".LNK_B_STOCKNO: " dvancing					
accept	LNK-B-STOCKNO					-
0% - ◎問題は見つ	T IDDUT=FOOT かりませんでした	•	行:186	文字:10	SPC	CR

- ③ メニューより、[ビルド(B)] > [ソリューションのリビルド(R)]を選択します。
- ④ 「RESTfulBOOKClientApp]プロジェクトを右クリックし、コンテクストメニューから[スタートアッププロジェクトに 設定(A)]を選択します。



- 3) 生成したテスト用 COBOL クライアントの実行(登録処理)
  - ツールバーにて RESTfulBOOKClientApp の [開始] アイコンをクリックし、アプリケーションを起動します。
     DOS プロンプトでアプリケーションが起動します。





プロンプト画面では以下の入力を行います。 Service Address: そのまま Enter キーを押す Supplemental Query String: そのまま Enter キーを押す Username: そのまま Enter キーを押す Password: そのまま Enter キーを押す Operation: "2"を入力し、Enter キーを押す LNK\_B\_TITLE: "PLANET OF THE APES"を入力し、Enter キーを押す LNK\_B\_TYPE: "SCIENCE FICTION"を入力し、Enter キーを押す LNK\_B\_AUTHOR: "PIERRE BOULLE"を入力し、Enter キーを押す LNK\_B\_STOCKNO: "5555"を入力し、Enter キーを押す LNK\_B\_RETAIL: "1000"を入力し、Enter キーを押す LNK\_B\_ONHAND: "3000"を入力し、Enter キーを押す LNK\_B\_SOLD: "2333"を入力し、Enter キーを押す

েব. Service Address (Enter = http://localhost:9003): Supplemental Query String (optional): Username (optional):

Password (optional): Operation (1 = SearchBOOK, 2 = AddBook): 2 Body Parameters:

.LNK\_B\_DETAILS:

..LNK\_B\_TEXT\_DETAILS: ...LNK\_B\_TITLE: PLANET OF THE APES ...LNK\_B\_TYPE: SCIENCE FICTION ...LNK\_B\_AUTHOR: PIERRE BOULLE ..LNK\_B\_STOCKNO: 5555 ..LNK\_B\_RETAIL: 1000 ..LNK\_B\_ONHAND: 3000 ..LNK\_B\_SOLD: 2333

#### .LNK\_FILE\_STATUS: 00

③ ブレークポイントで止まっているのでツールバーから [続行:]をクリックして処理を終了させます。

🕨 続行(C) 🕶 📴 🜄 🖕



- 4) 生成したテスト用 COBOL クライアント実行(検索処理)
  - 再度、ツールバーにて [RESTfulBOOKClientApp] アイコンをクリックし、アプリケーションを起動します。 DOS プロンプトでアプリケーションが起動します。 プロンプト画面では以下の入力を行います。
     Service Address: そのまま Enter キーを押す
     Supplemental Query String: そのまま Enter キーを押す
     Username: そのまま Enter キーを押す
     Password: そのまま Enter キーを押す
     Operation: "2"を入力し、Enter キーを押す
     LNK\_B\_STOCKNO: "5555" を入力し、Enter キーを押す

#### さきほど登録した情報が表示されます。

Service Address (Enter = http://localhost:9003):
Supplemental Query String (optional):
Username (optional):
Password (optional):
Operation $(1 = \text{SearchBOOK}, 2 = \text{AddBook})$ : 1
Body Parameters:
.LNK_B_STOCKNO: 5555
.LNK_B_DETAILS:
LNK_B_TEXT_DETAILS:
LNK_B_TITLE: PLANET OF THE APES
LNK_B_TYPE: SCIENCE FICTION
LNK_B_AUTHOR: PIERRE BOULLE
LNK_B_STOCKNO: 5555
LNK_B_RETAIL: 01000
LNK_B_ONHAND: 03000
LNK_B_SOLD: 02333
.LNK_FILE_STATUS: 00

② さきほど同様、[続行] アイコンをクリックして、デバッグを終了します。

#### 3.7.2 Curl コマンドによるテスト

作成した RESTful Web サービスは、一般的な仕様であるため、テスト用の COBOL アプリケーションからだけではな く、Java などの他の開発言語などからもアクセスできます。

ここでは、curl コマンドを用いてリクエストが行えることを確認します。

- 1) Windows のコマンドプロンプト画面を開き、チュートリアル用のファイルを解凍したフォルダーに移動します。ここまで の手順では、C:¥vc-tutorial に解凍しています。
- 2) 以下のコマンドを実行し、さきほど登録した書籍情報を検索します。
  - curl http://localhost:9003/temppath/RESTfulBOOK/1.0/SearchBOOK -d
     @json¥search5555.txt

```
C:\u00e4vc-tutorial>curl http://localhost:9003/temppath/RESTfulBOOK/1.0/SearchBOOK -d
@json¥search5555.txt
 "LNK_B_DETAILS" :
 {
   "LNK_B_TEXT_DETAILS" :
     "LNK_B_TITLE" : "PLANET OF THE APES",
     "LNK_B_TYPE" : "SCIENCE FICTION",
     "LNK_B_AUTHOR" : "PIERRE BOULLE"
   },
   "LNK_B_STOCKNO" : "5555",
   "LNK_B_RETAIL" : 1000,
   "LNK B ONHAND" : 3000,
   "LNK_B_SOLD" : 2333
 },
 "LNK_FILE_STATUS" : "00"
C:¥vc-tutorial>
```

#### 3.8 RESTful Web サービスのデバッグ

- 1) Visual Studio に戻り、[サーバーエクスプローラー] タブを選択します。
- [Micro Focus Server] > [Default] を選択したうえで、マウスの右クリックでコンテクストメニューを開き、[管理(A)] を 選択し、ESCWA 管理画面にログインします。



ر–لا	サーバー エクスプローラー		▼ -⊐ X	_	
দিশিখ	Ů× 혐별 ⊫			~	
57 SQL	<ul> <li>Micro Focus Analysis Se</li> <li>Micro Focus Server</li> <li>Default</li> </ul>	rver			
Serv			管理(A)		
er オブジェクト エクスブ	<ul> <li>Q ESDEMO64</li> <li>▲ 目 サーバー</li> <li>▷ Q win11-v-na</li> <li>データ接続</li> </ul>		削除(D) Enterprise Server 認証情報をクリア( フィルタ(F)	r を新規作成(N) C)	
□- <u>-</u> -		U	最新の情報に更新	f	Į,
- サーバー エクスプローラー					L L K

3) [Directory Server] > [Default] > [ESDEMO64] を選択し、[動的デバッグを許可] にチェックしたうえで、[適用] をクリックします。

<b>ES</b> 管理   ダッシュボード	<b>ネイティブ</b> メインフレーム セキュリティ	,
ネイティブナビゲーション ^	一般   ✓ モニター   ✓	
<ul> <li>         ・</li></ul>	一般的なプロパティ 適用 面削	除
<ul> <li>✓ I Directory Server</li> <li>✓ I G G ⊕ Default</li> <li>□ ESDEMO</li> </ul>	開始オプション	* 入力必須の項目です
B ESDEMO64 ▷	名前* <b>2</b> システムディレクトリ <b>2</b>	
› 슈 SOR	ESDEMO64	
	共有メモリページ数* 🛇 共有メモリ	クッション* 🎗
	512 ・ページ数(4k): 32	▲ ページ数(4k):
	SEP数* Q コンソール	ログサイズ* 💡
	2	ç k
	□ ローカル コンソールを表示 💡 🔽 動的テ	バッグを許可

 4) Visual Studio に戻り、サーバーエクスプローラーの [Micro Focus Server] > [Default] > [ESDEMO64] を選 択し、マウスの右クリックでコンテクストメニューを開き、[再起動(R)] を選択します。





5) ソリューションエクスプローラーより、[RESTfulCOBOL] プロジェクトを選択し、コンテクストメニューを開き、[スタートアッププロジェクトに設定(A)]を選択します。

ソリューション エクスプローラー		ビュー(W)
🖉 🐻 • 🖕 🗇 🗗 🎤 📑		すべての子孫を折りたたむ
ソリューション エクスプローラー の検索 (0		ここまでスコープ指定する(S)
😡 ソリューション 'RESTfulCOBOL'		新しいソリューション エクスプローラーのビュー(N)
▷ CBL RESTfulBOOKClientApp		ビルドの依存関係(B)
▲ RESTFUICOBOL		
Properties		追加(D)
deploy		既存の COBOL 項目を追加
BOOK.cbl		NuGet パッケージの管理(N)
BOOK-INFO.cpy		スタートアップ プロジェクトの構成
RESTfulBOOK.svi		スタートアップ プロジェクトに設定(A)

6) [RESTfulCOBOL] のデバッグアイコンをクリックして、デバッグを開始します。

🕨 RESTfulCOBOL 🝷 ▷

画面左下に、"準備完了"という文字が表示され、デバッグ待機状態になります。

ウオッチ 1	
検索 (Ctrl+E)	▶ ← → 検索の詳細度: ▼
名前	値
自動   ローカル <mark>  ウォッチ 1</mark>	
➡ 準備完了	



7) 0 と同様のリクエストを curl コマンドで送信します。

プロンプト画面ではリクエストがすぐに戻らず、Visual Studio 側では最初の処理で停止しています。

補足	<u>]</u> )	
loca	alhost が	デバッグ対象にならないことがあります。この場合は、localhost を 127.0.0.1 に変更してください
◎ () ◎ () ○ ()	) • ⊖   約 コセス: [N/A]	<ul> <li>              ■ ○ · ○ · □ Debug · x64 · ● 続行(C) · □ □ □ · □ □ □ · ○ → ↓      </li> <li>             ESDEMO64 · □ □ うイフサイクル イベント · スレッド: [6664] &lt;名前がありません&gt; · □ □ ∩ ▷ □ □ ○ → ↓      </li> </ul>
BOC	K.cbl +⊨ ×	RESTfulBOOK.svi
CBL R	ESTfulCOBO	L 🗸 🎸 BOOK
Þ	-	88 READ-RECORD VALUE "1". 88 ADD-RECORD VALUE "2". 88 DELETE-RECORD VALUE "3". 88 NEXT-RECORD VALUE "4". 01 LNK-FILE-STATUS PIC XX. COPY "BOOK-INF0.CPY" REPLACING ==(PREFIX)== BY ==LNK-B==.
		PROCEDURE DIVISION USING LNK-FUNCTION LNK-B-DETAILS LNK-FILE-STATUS. 0個の参照 MAIN SECTION.
⇔		CALL "CBL_TOUPPER" USING LNK-B-TEXT-DETAILS BY VALUE LENGTH LNK-B-TEXT-DETAILS RETURNING LS-CALL-STATUS
	->->	EVALUATE TRUE WHEN READ-RECORD PERFORM DO-READ-RECORD

この状態では、通常のプログラム同様、ステップインやブレークポイントなどが利用できます。 [続行(C)]を押して、処理を完了させると、プロンプト画面に結果が戻されます。

8) デバッグ終了のアイコンをクリックして、デバッグを終了します。

▶ 続行(C) ▼ | 📴 | 🔜 🚽 🚺 🔳 🚺

#### 3.9 Enterprise Server の停止

1) サーバーエクスプローラーを選択し、[Micro Focus Server] > [Default] > [ESDEMO64] を選択し、コンテクストメ ニューから [停止] を選択します。





停止ログが出力され、アイコンが赤色になります。

	出力								
	出力元(S):	Enterprise Server		-	ab⊐	Ŀ			
CASCD1005I C:¥Users¥tarot¥Documents¥Micro Focus User¥Visual COBOL¥WORKAREA¥ES CASCD0050I ES "ESDEM064" initiation is starting 15:08:22 ESDEM064 開始の成功 停止中 ESDEM064 userid: password: CASST0005I Shutdown of ES ESDEM064 starting 15:41:20 Return code: 0 ESDEM064 停止の成功									
	Iラー一覧	プロジェクト詳細 出力							
	サーバー エクス	(プローラー 🔹 🤫 🗙							
	Ŭ ×   Ÿ								
	<ul> <li>Mic</li> <li>Mic</li></ul>	ro Focus Analysis Serve ro Focus Server Default P ESDEMO							
		SDEMO64							

#### 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。