# Visual COBOL チュートリアル

# JCA による JBOSS EAP 連携の設定と開発

## 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 EJB として再利用し、Java EE ライアントから呼び出す機能を提供しています。EJB コンポーネントとして呼び出しを行う場合、Java アプリケーションサーバー上の Java EE クライアントは JCA の仕様にもとづいたリクエストを Enterprise Server に渡し、 COBOL のビジネスロジックが処理をして結果を返します。また、Enterprise Server は、XA に準拠しているので同じく XA に準拠して いるデータベースや他のシステムと協調してトランザクション処理を行うことができます。

このドキュメントでは Windows Server 上の JBOSS EAP 7.4 と Enterprise Server を JCA による連携を行い、Eclipse で開発 する方法を説明します。

## 2. 前提条件

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。サポートしているプラットフォームであれば他の Linux/UNIX でも利用可能です。

アプリケーションサーバー側 ソフトウェア

OS	Windows Server 2022 Standard Edition (64bit)
AP サーハー製品	Red Hat JBoss EAP 7.4.6

• 開発クライアント ソフトウェア

OS	Windows Server 2022 Standard Edition	(64bit)
COBOL 開発環境製品	Visual COBOL 9.0 for Eclipse	

チュートリアル用サンプルプログラム

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. 前提条件
- 3. チュートリアル手順の概要
  - 3.1. Visual COBOL for Eclipse を起動してソースコードのインポート作業を行う
  - 3.2. JCA 用のアプリケーションプロファイルを作成する
  - 3.3. ディプロイ先の変更を行う
  - 3.4. データファイルのアクセス指定を行う
  - 3.5. Visual COBOL for Eclipse にてマッピング指定を変更する
  - 3.6. リソースアダプタの編集と JBoss ヘディプロイを行う
  - 3.7. Visual COBOL for Eclipse にて JCA プロファイルを構成する
  - 3.8. テスト用クライアントアプリケーションを生成する
  - 3.9. テスト用アプリケーションの生成とテスト実行
  - 3.10. Enterprise Server にディプロイしたアプリケーションのデバッグ
  - 3.11. JBoss の停止と Enterprise Server インスタンスの停止

## 3. チュートリアル手順の概要

## 3.1. Visual COBOL for Eclipse を起動してソースコードのインポート作業を行う

- 1) Visual COBOL for Eclipse を起動
  - ① [スタート] メニュー > [Micro Focus Visual COBOL] > [Visual COBOL for Eclipse]を選択します。
  - ② ワークスペースの選択画面にて 任意の作業フォルダを指定します。ここでは"C:\#work\#JCA" を指定します。
- 2) 文字コードの指定を行います。
  - Visual COBOL 9.0 より Shift-JIS を指定して日本語を表示する場合、文字コードの指定を明確に行う必要があります。最初に、[Window(W)]メニュー > [設定(P)]より[一般] > [ワークスペース]とナビゲートし、テキストファイルエンコードを「MS932」に変更し、[適用して閉じる]ボタンをクリックします。

7ብሥል አካ	ワークスペース	<> ▼ ⇒ ▼ §
✓ 一般 へ Capabilities	ワークスペースの開始およびシャットダウン設定については、「開始およびシャットダウン」を	参照してください。
Schema Associations	□ さノニ パッコークナキ はぞうけいがた 使用して声が(の)	
UI フリーズ・モニタリング	□ ネイテイノのノックまたはパーリンクを使用して更新(R)	
> User Storage Service	✓ アクセス時に更新(S)	
Web ブラウザ	□ 無関係なプロジェクトを常にプロンプトなしで閉じる(C)	
> IF19 +-	ワークスペース保管間隔(分)(W): 5	
クイック検索	ウィンドウのタイトル	
20-7/14E		
コノテノッ・ツィノ	Cipse-workspace	
> 24-01-24-	□ バースペクティフ名を表示(T)	
> E41574	<ul> <li>ワークスペースのフルバスを表示(F): C:¥Users¥Administrator¥eclipse-works</li> </ul>	pace
> ネットワーク接続	✓ プロダクト名を表示	
ハンドラーをリンク		
パースペクティブ		
プロジェクト・ネーチャー	プロジェクトを開く際に、参照するプロジェクトを開く: プロンプト 🗸	
> ワークスペース 開始わよびシャットダウン	不明なプロジェクトの住間を以下のように報告(A): 警告 >	
> 外観		
検索	システム・エクスプローラーを記載するコマンド(X): explorer /E/select=\$(selected re	source loc}
比較/パッチ		- /
> Ant		_
AspectJ Compiler	テキスト・ファイル・エンコード(T) 新規テキスト・ファイルの行	区切り文字(F)
> Gradle	<ul> <li>デフォルト(U) (MS932)</li> <li>デフォルト(E) (Window</li> </ul>	/s)
> J2EE	○その他(O): MS932 ∨ ○その他(H): Window	/s ~
> Java		
> Java 水榄化		
	デフォルトの役	〔元(T) 適用(L)
, ,		
	· 通用1.7間	じる キャンセル
	Lizhote	117 6/

※Preference Recorder のダイアログが表示されたら [キャンセル] を選択してください。

- 3) ネイティブ COBOL プロジェクトの作成
  - [ファイル(F)]メニュー > [新規(N)] > [COBOL プロジェクト]を選択し、プロジェクト名に"NativeCOBOL"を入力し、 プロジェクトテンプレートは「64 ビット」を指定して、[終了(F)] ボタンを押します。
  - ② 次に作成した COBOL プロジェクトを選択した状態で、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、「プロパティ」 を選択します。[Micro Focus] > [ビルド構成] > [COBOL]とナビゲートし、「構成の固有な設定を可能にする (C)」にチェックを入れて[一般] > [ソース エンコーディング]を"UTF-8"から"ANSI"に変更し、[適用して閉じる]ボタン をクリックします。

リソース Coverage	New Configuration [使用中]	✓ 構成の管理
Micro Focus		
FIL-9-	一構成の固有な設定を可能にする(C)	
ビルドパス		
↓ ビルド構成	上位レベルの設定を上書きする (マージしない)	
COBOL		
1721	フィルタテキアトを3.カ	
ディプロイ		
ビルド環境	設定	值
> リンク	× 一般	
> プロジェクト設定	文字セット	ASCII
指令の確定	ソース エンコーディング	ANSI
> 実行時構成	COBOL 方言	Micro Focus
930-90 Dice	ソース フォーマット	固定
ビルター ゴロントクレーネーボル	デバッグ用にコンパイル	はい
ブロジェクト・ホーテヤー	EXIT PROGRAM を GOBACK として処理	ANSI
プロジェクト・ノバビット	詳細	いいえ
	.GNT にコンパイル	いいえ
侠雄 実行/だげゅが時史	✓ 出力	
美行/ナハック設定	指令ファイルを生成する	いいえ
	リストファイルを生成	いいえ
	コードカバレッジを有効にする	false
	プロファイラを有効にする	false
	✓ Iラ-/警告	
	警告レベル	回復可能なエラーを含める(レベル E)
	最大エラー数	100
	✓ 追加指令	
	追加指令	
	説明を表示する指令を選択します	^
	COBOL コンパイル設定: <括弧内の継承された設定 >	×
	<charset"ascii" dialect"mf"="" sourceformat"<="" td=""><td>fixed" NOLIST anim EXITPROGRAM"ANSI"</td></charset"ascii">	fixed" NOLIST anim EXITPROGRAM"ANSI"
	NOTESTCOVER NOPROFILE WARNING*1* MAX-ERI	ROR*100*> SOURCE-ENCODING*ANSI*

- 4) プログラムソースのインポート
  - COBOL エクスプローラーのパースペクティブを開き、COBOL エクスプローラービューにて プロジェクトフォルダを右クリック し、コンテクストメニューから [インポート(I)] > [インポート(I)] を選択します。

	インポート(i)	>	Ê,	リモートプロジェクト
4	エクスポート(O)		8	ローカル Micro Focus プロジェクトのリモート プロジェクトへの変換
8	更新(F)	F5	2	Net Express プロジェクトの変換
	プロジェクトを閉じる(S)		è	インポート(I)

② 既存のソースコードをロードします。一般のフォルダを展開し、[ファイル・システム]を選択し、[次へ(N)] ボタンを押します。

Leal インポート	_	Х
選択		
ローカル・ファイル・システムから既存のプロジェクトへリソースをインポートします。		5
Select an import wizard:		
プルタ入力		
▼ 🧀 一般		^
ノアイル・システム つ フォルダーまたはアーカイブ中来のプロジェクト		
□ アイルン などはシ カインロネのクロシェノト  ご 一 一 一 一 一 一 一 一 一 一 一 一 一 一 一 一 一 一 一		
1 設定		

 ③ インポートダイアログが表示されるので[参照(R)] ボタンを押してソースコードを解凍したフォルダを指定します。ここでは "C:¥Work"を指定しています。そこにある「BOOK.cbl」とコピーブック「BOOK-INFO.cpy」を指定し、[終了(F)] ボタンを押します。

# ファイル・システム

フォルダーを指定してください

次のディレクトリーから(Y): C:¥work	
> 🔳 🔁 work	BOOK.cbi     BOOK-INFO.cpy     BOOKSCRN.cbi     BOKSCRN.cbi     BReadme.txt

④ プロジェクトフォルダを展開し、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



- 5) ビルドオプションの変更
  - ① COBOL エクスプローラーにて作成したプロジェクトを右クリックし、[プロパティ(R)] を選択します。
  - ② プロパティ設定ダイアログが表示されます。[Micro Focus] > [ビルド構成] > [COBOL] を選択します。次に [構成の 固有な設定を可能にする(C)] にチェックを入れて、[.GNT にコンパイル] を"いいえ"から"はい"に変更します。次に[追 加指令]の値フィールドに"ASSIGN(EXTERNAL)"を指定します。

✓ Micro Focus ビルダ− ビルドパス ✓ ビルド構成 > COBOL イベント ディプロイ	<ul> <li>✓ 構成の固有な設定を可能にする(C)</li> <li>▲</li> <li>▲</li> <li>▲</li> <li>上位レベルの設定を上書きする (マージしない)</li> <li>フィルタテキストを入力     (三)     </li> </ul>	
ビルド環境 > リンク > プロジェクト設定	設定 ▼ 一般	值 · · · · ·
指令の確定 、 実行時構成	文字ゼット COBOL 方言	ASCII Micro Focus
Project Facets	ソース フォーマット     デバッグ用にコンパイル     ノーニー	固定 はい
Task Tags	EXIT PROGRAM を GOBACK として処理デバック	/用にコンパイル: はい
> Validation WikiText	.GNT にコンパイル	(tu
> タスク・リポジトリー ビルダー	◇ 田力 指令ファイルを生成する	いいえ
プロジェクト・ネーチャー	リストファイルを生成 コードカバレッジを有効にする	いいえ false
フロジェクト参照 実行/デバッグ設定	プロファイラを有効にする	false
	▼ 17-7 書き 警告レベル	回復可能なエラーを含める(レベル E)
	最大Iラ-数 > 違加指令	100
	追加指令	ASSIGN(EXTERNAL)
?		適用して閉じる キャンセル

 ③ 次に [Micro Focus] > [ビルド構成] > [リンク] を展開します。[ターゲットの種類] を「すべて INT/GNT ファイル」 に変更します。

✓ Micro Focus ビルダー		
ビルド パス <b>~</b> ビルド構成	フィルタテキストを入力	
> COBOL	設定	値
イベント	✓ Linkage	
ディプロイ	出力の名前	NativeCOBOL
ビルド環境	出力パス	New Configuration.bin
→ リンク	エントリポイント	
追加のリンクファイル	ターゲットの種類	すべて INT/GNT ファイル 💌
> プロジェクト設定	ビット数	64 ビット
指令の確定	IDD I-IPoutr_SIAV	11/1-7

- ④ 全ての設定が終了したら [適用して閉じる] ボタンを押します。
- ⑤ COBOL エクスプローラーにて"New\_Configuration.bin"を展開して下記のファイルが作成されていることを確認しま

す。

🗸 🔁 NativeCOBOL
> 垣 COBOL プログラム
> 🔑 コピーファイル
✓
🖹 BOOK.gnt
BOOK.gnt.1.tlog
BOOK.idy

# 3.2. JCA 用のアプリケーションプロファイルを作成する

- 1) EJB/JCA アプリケーションのプロファイルを作成
  - ① EJB/JCA アプリケーションとして利用する「BOOK.cbl」を右クリックし、コンテクストメニューから [新規作成(N)] > [Java インターフェイス]を選択します。

	新規作成(N)	>	曾	COBOL JVM プロジェクト
	開<(O)		Je≎ EU	COBOL JVM ユニット テスト プロジェクト
	表示方法(W)	Alt+シフト+W>	2	COBOL コピーファイル プロジェクト
	アプリケーションから聞く	>	2	COBOL プロジェクト
			8	COBOL ユニット テスト プロジェクト
	JK-	Ctrl+C	e:	リモート COBOL JVM プロジェクト
Ē	貼り付け	Ctrl+V	۲ġ	リモート COBOL コピーファイル プロジェクト
×	削除(D)	削除	<u>ک</u>	リモート COBOL プロジェクト
<u>_</u>	コンテキストから除去	Ctrl+Alt+シフト+下	1	リモート COBOL ユニット テスト プロジェクト
	移動(V)		₩U O	
	名前を変更(M)	F2	Ľ	フロジェクト(R)
	ビルド アクション	>	ß	COBOL コピーファイル
	タスクのスキャン		ð	COBOL プログラム
	指令の確定		<b>B</b>	COBOL ユニット テスト
	プログラムをコピーファイルに変換		<b>P</b>	スタンドアロン ファイル
			L\$	リモート スタンドアロン ファイル
		AU	- A	
	ノロクラムのノオーマット	Alt+シノト+ト	20	Java 1 79-JI1 X

 [Java インターフェイス名] 欄には "bookEJB" を入力し、「マッピング] 欄は「無し」を選択し、マッピングするプログ ラムは「BOOK.cbl」が選択されていることを確認し「終了] ボタンを押します。

## 3.3. ディプロイ先の変更を行う

- 1) COBOL エクスプローラー表示設定の変更
  - COBOL エクスプローラーに戻ります。
  - ② COBOL エクスプローラー右上の「↓↑」アイコンの右横にあるアイコンをクリックし、「フィルタとカスタマイズ(F)] を選択しま

#### JCA による JBOSS EAP 連携の設定と開発



③ [カテゴリ外の空のフォルダ] にチェックされている場合は、チェックを外したのち、[OK] ボタンをクリックします。



- 2) ディプロイ用のフォルダの作成
  - ① 「NativeCOBOL」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 [新規作成(N)]→[フォルダー] を選択しま す。
  - ② 「NativeCOBOL」プロジェクトが選択されていることを確認の上、フォルダ名に "deploy" を指定し、[終了(F)] ボタ ンをクリックします。
- 3) 「.mfdeploy」ファイルのインポート
  - ① 作成した「deploy」フォルダ上で右クリックし、コンテクストメニューから [インポート]→[インポート] を選択します。
  - ② 「一般」配下の「ファイル・システム」を選択し、[次へ(N)] ボタンをクリックします。

## 選択

ローカル・ファイル・システムから既存のプロジェクトへリソースをインポートします。

インポート・ウィザードの選択(S):
フィルタ入力
<ul> <li>              ←般             ・</li></ul>

- ③ [参照(R)] ボタンをクリックし、Visual COBOL インストールフォルダ¥deploy を選択します。
- ④ [.mfdeploy] ファイルにチェックを行い、[終了(F)] ボタンをクリックします。

<b>ファイル・システム</b> ローカル・ファイル・システムからリソースをインポートします。	
次のデイレクトリーから(Y): C:¥Program Files (x86)¥Micro Focus¥Visual COBOL¥deploy ~	参照(R)
■ 🗁 deploy	
タイプをフィルター(T) すべて選択(S) 選択をすべて解除(D)	
インボート先フォルダ(L): NativeCOBOL/deploy	参照(W)

#### 補足)

上記手順完了後も COBOL エクスプローラー上の NativeCOBOL¥deploy フォルダ配下に .mfdeploy ファイルは 表示されませんが、これはフィルタによるものです。

正しくインポートされたことを確認する場合は、さきほど同様、[フィルタとカスタマイズ]を選択し [.\* リソース]のチェックを 外すことで表示されるようになります。しかし、リソースに関する設定情報が表示されるようになるため、通常はチェックを 行い、非表示状態とすることを推奨します。

- 4) ディプロイ先の変更設定
  - ① [サーバーエクスプローラー] タブを選択します。
  - ② [ローカル localhost:86] 上で右クリックし、コンテクストメニューから [管理ページを開く] を選択します。



③ ESCWA (Enterprise Server Common Web Administration) 画面にてダッシュボードが表示されます。

ES 管理 ダッ	シュボード <b>キイティブ</b> ES_NET メインフレーム セキュリティ & v U5.1.5 S5.1.3
✓ <sup>1</sup> グループ	⑦ 一般   ~ & ユーザー ~ ESDEMO64 (Default) 〇
> 論理 > PAC	ー般的なプロパティ   C 適用 面削除 ⑦
<ul> <li>✓ ☐ Directory Server</li> <li>✓ ☐ ☐ Default</li> <li>③ ESDEMO</li> </ul>	開始オプション
ESDEMO64 > ∲ SOR	<ul> <li>入力必須の項目です</li> <li>名前。 Q</li> <li>システムディレクトリ Q</li> </ul>
	ESDEMO64 共有メモリ ペー* ② 512 🔷 ページ数(4k): 共有メモリ クッ * ③ 32 🔷 ページ数(4k): ジ数 ション
	SEP数* 2 コンソールログ * 2 サイズ k
	□ ローカル コンソー ♀ □ 動的デバッグを許 ♀ □ システム起動時に ♀ ルを表示 □ 開始する
	■ 64ビット作業モー <b>♀ □ 以前のログを削除 ♀</b> ド

④ [ネイティブ]タブメニューをクリックします。



- ⑤ 次に左側メニューの[Directory Server]をドリルダウンして、[Default]  $\rightarrow$  [ESDEMO64]をクリックします。

~	යි 🖯	Default	
	Ξ.	ESDEMO	
	Ξ,	ESDEMO64	$\triangleright$

⑥ [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ます。

ダッ	ッシュボー	×	ネイテ	ィブ	ES .NET
	0	-	般	$ $ $\checkmark$	
		₹⊐	<b>9</b> -	$\sim$	

⑦ ドロップダウンメニューから[リスナー]をクリックします。

✓ プロパティ



⑧ [通信プロセス1]の下に見えている[Web]リスナーをクリックします。



- ・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。この場合、Visual COBOL インストールディレクトリ配下のdeploy フォルダがディプロイ用フォルダとして使用されます。通常、Program Files (x86)等のフォルダは管理者権限を持つユーザーでないと書き込みできないので変更を行います。
   下記の例のように設定を変更し、[適用] ボタンをクリックします。
   例:uploads=C:/work/JCA/NativeCOBOL/deploy
- 10 正常に更新された旨のポップアップが表示されます。

※Enterprise Server を自分で作成した場合、[Web リスナー]を「Disable」から「Started」に変更してください。

① 次に [Web Services and J2EE] リスナーをクリックします。

?	一般   ~	&ユーザー >
通信フ	<b>゜ロセス</b>   C	* 通信サーバーの
✓ 🗉	通信プロセス 1 af HTTP Echo af Web	
	ar i Web Services a	Ind J2EE

22 ポート番号を"\*"から"9003"に変更し、スライドバーを上に移動して、[適用] ボタンをクリックします。

~		
8		
b Servi	ces and J2EE ロジェクティング	シー Micro Focus アプリ ション形式
	Q	
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で	、TLSが有効ではありません。
・⊐ル *	ホスト名またはIP アドレス* ♀	ボート* 💡
	*	9003

## 3.4. データファイルのアクセス指定を行う

- 1) Enterprise Server がアクセスするデータファイルを指定
  - ESCWA 画面の[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ます。
     [ES-Environment]

BOOKINFO=C:\u00e4work\u00e4DAT\u00e4BOOKINFO.DAT

設定
構成情報 ♀
[ES-Environment] BOOKINFO=C:\work\DAT\BOOKINFO.DAT

このパスは

① 上にスクロールし、[適用] ボタンをクリックします。

?	一般			₹ <b>二</b> ター ~	, ,	<u>&amp;</u> д-
一般的	はプロノ	(ティ	С	適用	<b>茴 削除</b>	
884/1-4-						

② 更新された旨のポップアップメッセージが表示されます。

~	ネイティブ リージョンの更新 ネイティブ リージョン "ESDEMO64" が更新されました。 この変更内容はリージョンの再起 動時に適用されます。
	25-01-22-13-21-1-01-3-0

- 2) Enterprise Server の起動
  - ① [サーバーエクスプローラー] タブを選択します。

<mark>≌ COBOL</mark> … ສິ	₨. ナビゲーター	Provide Applica	💻 サーバー	
			✓ □	エクスプローラー
V 🔁 NativeCOE	30L ゴロガニル			

② ツリーを展開し、Visual COBOL にビルドインされている Enterprise Server 「ESDEMO64」を右クリックし、[開始]を選択します。もし、ダイアログが表示されたらそのまま[OK]ボタンを押してください。

マ ❹ ローカル [localho マ 昌 Default [12]	ost:10086] 7.0.0.1:86]	
> 🛃 ESDEMO		
> 🛃 ESDEM		>
_	管理ページを開く	
	開始	

③ Eclipse の Secure Storage に関するダイアログが表示された場合、[いいえ] を選択してください。開始処理の状況 は、[コンソール] ビューでモニターできます。

230817	13424619			CASCD0127I SEP 00002 created for ES ESDEM064, process-id = 5128 13:42:46
230817	13424628	5128	ESDEMO64	CASSI1500I SEP initialization started 13:42:46
230817	13424633	144	ESDEMO64	CASSI1600I SEP initialization completed successfully 13:42:46
230817	13424635	144	ESDEMO64	CASSI4100I PLTPI Phase 1 Processing Starting 13:42:46
230817	13424636	144	ESDEMO64	CASSI4106I PLTPI Phase 1 starting File initialization program 13:42:46
230817	13424637	144	ESDEMO64	CASSI5040I Active SEP memory strategy set to x'00000001', retain count 100 13:42:46
230817	13424681	4896	ESDEMO64	CASCS5100I Communications Process instance 01 is ready to accept requests 13:42:46
230817	13424721			CASCD1071I Administration SEP created for Server ESDEM064, process-id = 5712 13:42:4
230817	13424726	5712	ESDEMO64	CASSI1500I SEP initialization started 13:42:47
230817	13424735	5712	ESDEMO64	CASSI1600I SEP initialization completed successfully 13:42:47
230817	13424739	5128	ESDEMO64	CASSI1600I SEP initialization completed successfully 13:42:47

 ④ 正常に開始されると [サーバーエクスプローラー] ビュー上の「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。

>	SDEMO	
>	📇 ESDEMO64	

## 3.5. Visual COBOL for Eclipse にてマッピング指定を変更する

- 1) COBOL と Java の間の変換マッピングを定義、編集
  - ① [ウィンドウ(W)] メニューから [設定(P)] を選択し、[Micro Focus] > [サービスインターフェイス] > [インターフェイスマッ

パー]を選択します。[COBOL 割り当てペインの表示] にチェックを入れ [適用して閉じる] ボタンをクリックします。

※Preference Recorder のダイアログが表示されたら [キャンセル] を選択してください。



② 生成された bookEJB を右クリックし、コンテクストメニューから [新規作成(N)] > [オペレーション] を選択します。

V 💦 Java 1	(ンダ	<sup>7</sup> -フェイス			
> 🧀 Ne		新規作成(N)	>	曾	COBOL JVM プロジェクト
> 🗁 rep y	۲.	削除	削除		COBOL JVM ユニット テスト プロジェクト
		-fn (f= /(n)		2	COBOL コピーファイル プロジェクト
		ノロハナ1(P) ニィプロノ		2	COBOL プロジェクト
≝ / X @		ア1ノ⊔1 ☆本		Ē	COBOL ユニット テスト プロジェクト
		伸直		4	リモート COBOL JVM プロジェクト
アウトラインを計		用く		Ê	リモート COBOL コピーファイル プロジェクト
712-18-028		27177下生成		Ê	リモート COBOL プロジェクト
				1	リモート COBOL ユニット テスト プロジェクト
			Micro Focus ビルド: NativeCOBOL		プロジェクト(R)
			BUILD SUCCESSFUL Build finished with no errors.	<b>P</b>	スタンドアロン ファイル
			"201 10	Ľ	リモート スタンドアロン ファイル
			Total time: 0 seconds ビルド完了	<del>C</del> *	オペレーション

- ③ [オペレーション名] 欄に "searchBook" を指定し、[OK] ボタンを押します。
- ④ 生成されたオペレーション「searchBook」をダブルクリックします。
- ⑤ 生成されたインターフェイスマッピングを編集します。[LINKAGE SECTION] の COBOL 変数「LNK-FUNCTION」を[COBOL割当て] にドラッグ&ドロップします。



⑥ [COBOL 割当てプロパティ] ダイアログが表示されるので [値] に "1" を設定して[OK]ボタンをクリックします。

😑 COBOL 割当てプロパティ 🚽 🗆 🗙					
名前: LNK-FUNCTIO	N			編集	
値: 1					
	ОК		キャンさ	211	

す。

⑦ 次に「LNK\_B\_STOCKNO」を [searchBook オペレーション – インターフェースフィールド] にドラッグ&ドロップしま



- ⑧ 同様の手順で、個別に以下のフィールドをドラッグ&ドロップします。
   「LNK-B-TITLE」、「LNK-B-TYPE」、「LNK-B-AUTHOR」、
   「LNK-B-RETAIL」、「LNK-B-ONHAND」、「LNK-B-SOLD」、「LNK-FILE-STATUS」
- ⑨「LNK\_B\_TITLE」をダブルクリックします。フィールドプロパティが表示されるので〔入力〕→〔出力〕に変更して [OK]ボタンを押します。



⑩ 同様の手順で「LNK\_B\_STOCKNO」以外の項目は全て方向を出力に変更してください。最終的には下のイメージのような構成になります。

searchBook オペレーション	៸ - インターフェイス フィー	ルド
名前	方向	型
LNK_B_STOCKNO	入力	String
LNK_B_TITLE	出力	String
LNK_B_TYPE	出力	String
LNK_B_AUTHOR	出力	String
LNK_B_RETAIL	出力	int
LNK_B_ONHAND	出力	int
LNK_B_SOLD	出力	int
LNK_FILE_STATUS	出力	String

# 3.6. リソースアダプタの編集と JBoss ヘディプロイを行う

- 1) リソースアダプタの編集とディプロイ
  - Visual COBOL のマニュアルに従い、JBoss EAP の構成ファイル「standalone.xml」にリソースアダプタの指定をエ ディタ等で記述します。
  - 2 <u>https://www.amc.rocketsoftware.co.jp/manuals/VC90/Eclipse/vc90indx.html</u>
  - ③ <archive-validation> タグを修正します。
  - ④ <subsystem> タグにてリソースアダプター mfcobol-notx.rar に関連する情報を指定します。
  - ⑤ Visual COBOL イントールディレクトリ配下にある javaee フォルダ配下の javaee7 → jboss74EAP にある 「mfcobol-notx.rar」ファイルを JBOSS インストールディレクトリ配下の standalone フォルダ → deployments 配下にコピーします。
  - ⑤ JBoss EAP を起動するために Windows メニューを表示して [JBoss プラットフォーム] > [サーバーの起動(スタンドアローン]を選択します。



⑦ リソースアダプタ「mfcobol-notx.rar」ファイルが適切にデプロイされていることをコンソールメッセージで確認します。

>ら http-remoting-connector キャッシュを開始しました。 |:40:18,984 INFO [org.jboss.as.server] (DeploymentScanner-threads - 2) WFLYSRV0010: "mfcobol-notx.rar" (runtime-name : "mfcobol-notx.rar") をデプロイしました。

※standalone.xml の編集の詳細はマニュアルの [ディプロイ] > [Enterprise Server へのディプロイ] > [モダナイズ

済みのアプリケーションのディプロイおよび構成] > [EJB およびリソース アダプターのディプロイ] > [EJB のディプロイ - 概要] > [JBoss へのディプロイ] を参照してください。

## 3.7. Visual COBOL for Eclipse にて JCA プロファイルを構成する

- 1) EJB/JCA アプリケーションのプロファイルを構成
  - 「3.5 Visual COBOL for Eclipse にてマッピング指定を変更する」で作成したプロファイル「bookEJB」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 [ディプロイメントサーバー] タブをクリックします。
  - ③ [Enterprise Server 名] 欄にて [変更] ボタンを押します。
  - ④ ディプロイする Enterprise Server の一覧が表示されるので [ESDEMO64] 選択して [OK] ボタンを押します。

ディプロイ先の Er	nterprise Serv	erを選択してく	ください:		
サーバー	サービス名	サービス状態	エンドポイント	リスナー <del>状</del> 態	説明
ESDEMO	Deployer	Available	0.0.0.0:0	Stopped	Deployment
ESDEMO64	Deployer	Available	localhost:53595	Started	Deployment

- ⑤ [アプリケーションファイル] タブをクリックし、[レガシーアプリケーションをディプロイする] を選択します。
- ⑥ [ファイル追加] ボタンを押します。
- ⑦ Ctrl を押しながら、「BOOK.gnt」及び「BOOK.idy」を選択し [開く] ボタンを押します。
- ⑧ [EJB 生成] タブをクリックします。
- ③ [アプリケーションサーバー] 欄にて、「JEE 7」及び「JBoss EAP 7.4」を選択し、「トランザクション可能] のチェックを外します。

一般	ディプロイメント	サー	バー	アこ	プリケーションファイル	EJB 生
アプリケ	ーション サーバー	JI	EE 7	$\sim$	WebLogic 12.2.1	$\sim$
トラン	ッザクション可能				WebSphere 9.0	
-EJB 属	性				WebSphere Libert WebLogic 12.2.1	ty
Bean	名:	bo	okEJI	В	JBoss EAP 7.4	

- ① [Java コンパイラ] 欄は、インストールされている JDK を指定します。例えばここでは、チュートリアル用に別途インストールした "C:¥Java¥JDK1.8.0¥bin"を指定しています。
- ① [Java EE クラスパス] 欄は [参照] ボタンを押し、エクスプローラーにて下記の3つのクラスファイルを指定します。
   C:/JBoss/EAP-7.4.0/modules/system/layers/base/javax/ejb/api/main/jboss-ejb-api\_3.2\_spec 2.0.0.Final-redhat-00001.jar
- 12 11の要領で

 $\label{eq:c:/JBoss/EAP-7.4.0/modules/system/layers/base/javax/resource/api/main/jboss-connector-api_1.7\_spec-2.0.0.Final-redhat-00001.jar$ 

13 11の要領で

C:/JBoss/EAP-7.4.0/modules/system/layers/base/javax/servlet/api/main/jboss-servlet-api\_4.0\_spec-2.0.0.Final-redhat-00001.jar

④ [J2EE クラスパス] 欄に jar ファイルが3つセミコロン「;」で区切られて追加されていることが確認できたら、[OK] ボタンを押します。

- 2) Enterprise Server の再起動
  - ① Eclipse IDE に戻り、「サーバーエクスプローラー」に切り替えます。
  - 「ESDEMO64」上で右クリックし、コンテクストメニューから [再起動] を選択し、Enterprise Server を停止/起動 します。

### 3.8. テスト用クライアントアプリケーションを生成する

- 1) ディプロイした Java サービスをテストするための J2EE アプリケーションを生成する
  - ① "bookEJB"を右クリックし、コンテクストメニューから[ディプロイ]を選択します。

🔄 🛛 🖂 Java -	(ンター	・フェイス		LNK-B-AUTHOR
⊿ ≋⊚ bo	ok E 1 P	1	-	LNK-B-STOCKNO
-6		新規作成(N)		•
⊳ 🗁 New_	×	削除(D)		
b > repos		プロパティ(P)		
		รังวับง		
		検査		

- ② さきほどブラウザー上で開いていた「管理画面」に切り替えます。
- ③ [一般]メニューから[サービス]をクリックします。
- ④ 画面を下にスクロールしていくと最下行にディプロイした JCA サービスが追加されていることを確認します。

Image: SearchBook         Available         Web Services and J         BOOK         MFRHBINP         created 16:24:44 Image: 0.023	෯	✓ BOOK						
	ଞ	.searchBook	Available	Web Services and J	воок	MFRHBINP	created 16:24:44 �� 2023-04	
								_

## 3.9. テスト用アプリケーションの生成とテスト実行

- 1) ディプロイした Java サービスをテストするための Java EE アプリケーションを生成する
  - ① 「bookEJB」を右クリックし、コンテクストメニューから[クライアント生成...]を選択します。

✓ 💦 Java インターフェ	<u>ال</u> م		LNK-B-TYPE	X(20)
🗸 📽 bookEJ <sup>p</sup>				×//-0
🗄 sear	新規作成(N)			>)
> 🗁 deploy	削除			削除
BOOK.	プロパティ(P)			
BOOK.	รัสวิอิส			)
ר 📼 🕫 ריק	検査			
	開く			
	クライアント生成			
トライ フタ提供す コアクエ	イフルエエイツーはめり	<b>T</b>		

② [サービスインターフェイス コンソール] ビューに正常にクライアントアプリケーションが生成された旨のメッセージが出力され

ます。	
-----	--

```
マニフェストが追加されました

bookEJB.jarを追加中です(入=3056)(出=2747)(10%収縮されました)

bookEJB.warを追加中です(入=19427)(出=18513)(4%収縮されました)

mfejblib.jarを追加中です(入=11614)(出=10418)(10%収縮されました)

META-INF/application.xmlを追加中です(入=527)(出=261)(50%収縮されました)

META-INF/jboss-deployment-structure.xmlを追加中です(入=509)(出=201)(60%収縮されました)

C:\Program Files (x86)\Micro Focus\Visual COBOL\bin64

クライアント生成が正常終了しました。
```

 ③ 生成されたクライアントアプリケーション「bookEJB.ear」をJBoss EAP インストールディレクトリ配下の standalone フォルダ → deployments 配下にコピーします。



④ 正常にディプロイされたことを JBoss EAP の起動コンソール画面で確認します。

16:30:30,431 INFO [org.jboss.as.ejb3.deployment] (MSC service thread 1-4) WFLYEJB0473: デプロイメントユニット 'subdepl yment "bookEJB.jar" of deployment "bookEJB.ear" の 'bookEJBEJB' という名前のセッション Bean の JNDI バインディングは次0 とおりです:
java:global/bookEJB/bookEJB/bookEJBEJB!com.mypackage.bookEJB.BookEJB java:app/bookEJB/bookEJBEJB!com.mypackage.bookEJB.BookEJB java:module/bookEJBEJB!com.mypackage.bookEJB.BookEJB java:jboss/exported/bookEJB/bookEJB/BookEJBBUB!orm.mypackage.bookEJB.BookEJB ejb:bookEJB/bookEJB/bookEJBEJB!com.mypackage.bookEJB.BookEJB java:global/bookEJB/bookEJBEJB!com.mypackage.bookEJB.BookEJB java:apc/bookEJB/bookEJBEJB java:apc/bookEJB/bookEJBEJB

- 2) ディプロイした Java アプリケーションのテストを行う
  - ブラウザーを起動します。
  - ② アドレスバーに http://localhost:8080/bookEJB/searchBook.jsp を入力します。
  - ③ アプリケーションの動作を確認します。
  - ④ LNK\_B\_STOCKNO\_io に "1111" を指定して [Go!] ボタンを押します。
  - ⑤ ブラウザーに COBOL プログラムがデータファイルより読み込んだ内容が返されます。

searchBook_LNK_B_STOCKNO : 1	111
	Go!

Result:

Variable	Value
LNK_B_TITLE	LORD OF THE RINGS
LNK_B_TYPE	FANTASY
LNK_B_AUTHOR	TOLKIEN
LNK_B_RETAIL	1500
LNK_B_ONHAND	4000
LNK_B_SOLD	3444
LNK_FILE_STATUS	00

## 3.10. Enterprise Server にディプロイしたアプリケーションのデバッグ

- 1) 動的デバッグの許可
  - ① さきほどブラウザー上で開いた管理画面の左側より「ESDEMO64」の情報をクリックします。
    - - 🗸 🏠 🔓 Default
        - ESDEMO
  - ② 画面の[一般」をクリックし、[動的デバッグを許可]をチェックしたうえで、[適用]ボタンをクリックします。

② <u>−<sub>般</sub>   ∨</u> 一般的なプロパティ	モニター   〜 C _ 適用 _ 値 削除	& ユ−ザ−	
開始オプション			
* 入力必須の項目です			
名前* 💡	システムディレクトリ 🤇	}	
ESDEMO64			
共有メモリ ページ数* 💡	512 🗘 ページ数(4	(): 共有メモリ クッション*	<b>32</b> ページ数(4k):
SEP数* 💡 2			
コンソール ログサイズ* 💡	0 🗘 k を表 🖓 🗹 動的デバ	ッグを許可 💡 🗌	システム起動時に開始する 💡

- ③ Eclipse IDE に戻り、「サーバーエクスプローラー」に切り替えます。
- ④ 「ESDEMO64」上で右クリックし、コンテクストメニューから [再起動] を選択し、Enterprise Server を再起動します。
- 2) アプリケーションのデバッグ準備
  - ① [実行(R)]メニューより、[デバッグの構成(B)]を選択します。

	デバッグ履歴(H)				
称	デバッグ(G)	>			
	デバッグの構成(B)				

② 左側より、「COBOL Enterprise Server」上で右クリックし、コンテクストメニューから [新規構成(W)] を選択しま

す。

📑 🖻 💫 🗎 🗶 🖻 🍸 🕶		このダイ
フィルタ入力		📑 - i
🗄 Apache Tomcat	^	P - 1
AspectJ/Java Application		🔊 - j
AspectJ Load-Time Weaving Applicatio		<b>B</b> . <b>3</b>
<ul> <li>COBOL/Java 相互運用機能のアプリケーシ</li> </ul>		
COBOL Enterprise Server	et A	- I AO
JVM COBOL JVM アプリケーション 🏼 新規構	πx(V	(V)

③ 以下の入力を行い、[デバッグ(D)]をクリックします。

JCA による JBOSS EAP 連携の設定と開発

#### 名前: "NativeCOBOL"

Enterprise Server を選択し、 [参照] をクリックして、 「ESDEMO64」 を選択

名前( <u>N</u> ): Nat	tiveCOBOL			
い 一般 🍕	> ソース 🔲 共通((	) 🧤 デバッグシンボル 🥑 コンテナー		
Enterprise Se	erver 上でデバッグセ	ッションを開始して、COBOL プログラムの起	副を待機します。	
	ロジェクト( <u>P</u> )			
NativeCO	DBOL			参照
	Server			
● 接続				
ESCWA:	ローカル			
MFDS:	Default			
リージョン	ESDEMO64			
				参照

#### デバッグの種類で「Java」を選択

▼ デバッグの種類	
Web サービス Java	
Java サービス名 (空白の場合はすべてのサービスをデバッグ)	
▼ デバッグオプション	
□ ブレークポイントでのみ停止	
□ リバース デバッグを有効にする(Linux x86/64 プラットフォームでのみサポート)	
□ メイン エントリ ポイント上のプログラム ブレークポイントのみ	
☑ ブログラム シンボル (.idy) はブログラムと一致する必要がある	<b>~</b>
	前回保管した状態に戻す(火) 適用(火)
	デバッグ( <u>D</u> ) 閉じる

④ 以下のダイアログが表示された場合は、[設定を保存] にチェックをして [はい(Y)] をクリックします。

S - N − 2	スペクティブの切り替えの確認	×			
2	この種類の起動では、開始時にデバッグパースペクティブを開くように構成します。				
	このデバッグ・パースペクティブは、アプリケーションのデバッグをサポートするように設計されています。 これには、デバッグ・スタック、変数、およびブレークポイント管理を表示するビューが組み込まれてい ます。				
	今パースペクティブを開きますか?				
☑設定	を保存 はい(Y) いいえ(N)				

- ⑤ さきほど同様、ブラウザー上で <u>http://localhost:8080/bookEJB/searchBook.jsp</u> にアクセスし、"1111"を入力して、[Go!] をクリックします。
- ⑥ Eclipse IDE を見ると、デバッグパースペクティブ上で、BOOK.cbl で停止していることが分かります。



通常のコンソールアプリケーションと同様、項目値の確認や、ステップ実行などが行えます。

デバッグを終え、画面に応答結果を戻すためには、[実行(R)]メニューより [再開(M)] を選択してください。

1	実行(R)		
Ì		再開(M)	F8
1		中断(S)	
2		終了(T)	Ctrl+F2

⑦ デバッグ作業を終了する際は、[実行(R)] メニューより [終了(T)] を選択してください。

実行(R)	-		
छ : 🔏 🛙		再開(M)	F8
~		中断(S)	
		終了(T)	Ctrl+F2
	59	切断	

## 3.11. JBoss の停止と Enterprise Server インスタンスの停止

- 1) Enterprise Server の停止
  - ① 「サーバーエクスプローラー」にて「ESDEMO」上で右クリックし、コンテクストメニューから [停止] を選択します。
- 2) JBoss EAP の停止
  - ① Windows メニューを表示して [JBoss プラットフォーム] > [サーバーの終了(スタンドアローン)]を選択します。



② JBoss が起動されたコマンドプロンプト、JBoss を停止したコマンドプロンプトの両方の DOS プロンプトを閉じます。

#### 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。

JCA による JBOSS EAP 連携の設定と開発

### 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。