Visual COBOL チュートリアル

Apache Tomcat と JVM COBOL コンポーネントを利用した RESTful Web サービス開発

1 目的

Visual COBOL は、他言語・他システムとの連携手段として、「Enterprise Server」を利用したサービス連携だけでは なく、COBOL プログラムをそのまま利用し、Java 技術と連携可能な JVM COBOL 機能を提供しています。本機能を 利用することで、Java, Scala などの Java 言語で作成した RESTful Web サービス上で COBOL を利用するといっ た方法が可能です。

このドキュメントでは JVM COBOL 機能を利用して Java で実装する RESTful Web サービスと COBOL の連携 方法について説明します。

2 前提条件

本チュートリアルは、下記環境を前提に作成されています。

OS	Windows 11 Enterprise Edition
COBOL 環境製品	Visual COBOL 9.0J for Eclipse Patch Update 2
Java アプリケーションサーバー	Apache Tomcat 10.0.27

※ Visual COBOL 製品は Apache Tomcat 10.0 の EOS にともない、10.1 をサポートしています。しかし、製品 で使用する Eclipse IDE では 10.0 のみ選択できます。これは、Eclipse IDE 側のプラグインの挙動によります。このチ ュートリアルで使用する 10.0.27 は製品機能紹介目的であり、運用環境においては 10.1 の使用を推奨します。

Java 言語で作成する RESTful Web サービスは JAX-RS という Web サービスを構築するための Java API を使用しています。本ドキュメントでは、Tomcat 10 を Java アプリケーションサーバーとして使用して動作確認を行いますが、 Apache Tomcat に依存せず標準的な API 仕様に基づいたサンプルアプリケーションとなります。このため、本ドキュメントで紹介するようなサービスを、他の Java アプリケーションサーバー上にて開発・利用することも可能です。

また、Eclipse の Tomcat プラグインに関する設定および動作などにつきましては、弊社製品外であり、インターネットなどの 各種文献をご参照ください。

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。 このサンプルプログラムは、COBOL で作成された簡単な書籍情報を管理するアプリケーションであり、索引ファイルを利用し ています。以降の手順では、C:¥jvmRESTfulWSTutorialを解凍先のフォルダとして説明しています。

チュートリアル用のサンプルファイルのダウンロード

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 プロジェクトの作成と COBOL アプリケーションの確認
 - 3.1.1 プロジェクトの作成
 - 3.1.2 アプリケーションの確認
 - 3.2 JVM COBOL プロジェクトの作成
 - 3.3 Apache Tomcat 上で動作する RESTful Web サービスの作成
 - 3.3.1 Java Web アプリケーションプロジェクトの作成
 - 3.3.2 RESTful Web サービスの確認

3 チュートリアル手順の概要

従来型の書籍情報を管理するコンソールアプリケーションを、JVM COBOL の機能を利用して COBOL プログラムそのま ま流用し、Java クラスを生成します。そして、このクラスを利用して Java ベースの RESTful Web アプリケーションを構 築します。最後に、Web アプリケーションを Apache Tomcat アプリケーションサーバーにデプロイし、実際の動作を確認 します。



3.1 プロジェクトの作成と COBOL アプリケーションの確認

3.1.1 プロジェクトの作成

1) Visual COBOL for Eclipse を起動します。



2) 任意のワークスペースを選択し、[起動(L)] をクリックします。

ディレクトリーをワークスペースとして選択

Eclipseは、ワークスペースディレクトリを使用して、環境設定と開発成果物を保存します。

ワークスペース(<u>W</u>):	C:¥workspace-jvmws	~	参照(<u>B</u>)
□この選択をデフォ ▶ 最近のワークスペ・	ルトとして使用し、今後この質問を表示しない(<u>U</u>) -ス(<u>R</u>)		
		起動(<u>L</u>)	キャンセル

 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。[Window] > [設定] 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] をクリックします。

フィルタ入力	ワークスペース	⟨¬ ▼ □⟩ ▼ 8°
✓ 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、 <u>開始およびシャットダウン</u> を参照	してください。
Schema Associations UI フリーズ・モニタリング	□ ネイティブのフックまたはポーリングを使用して更新(R)	
> User Storage Service Web ブラウザ	 ビ」アグゼス時に更新(S) □ 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 グローバル化	ウィンドウのタイトル	
コンテンツ・タイプ サービス・ポリシー	✓ ワークスペース名を表示(E): workspace-jvmws	
5 - Lス・ホックー > セキュリティー トレース > ネットワーク接続	 □ ハースペクテイン名を表示(1) □ ワークスペースのフル/(スを表示(F): C:¥workspace-jvmws ☑ ブロダクト名を表示 	
ハンドラーをリンク パースペクティブ ゴロジェクレッナ チャ	ブロジェクトを開く際に、参照するブロジェクトを開く: ブロンプト 🗸 🗸	
> ワークスペース > 開始およびシャットタウン > 外観	不明なブロジェクトの性質を以下のように報告(A): 警告 🗸	
検索 比較/パッチ	システム・エクスプローラーを起動するコマンド(X): explorer /E,/select=\${selected_resources	rce_loc}
> Ant Aspect/ Compiler	=キフト・ファイル・エンフード(ア) 新規テキスト・ファイルの行区切	り文字(F)
> Gradle	 ● デフォルト(U) (MS932) ● デフォルト(E) (Windows) 	_
> J2EE > Java	○その他(O): MS932 ∨ ○その他(H): Windows ∨	/
> Java 永続化 JDT Weaving ✓	デフォルトの復元(1	ī) 適用(L)
? 🛓 🖌 💿 😨	適用して閉じる	キャンセル

Preference Recorder のダイアログが表示された場合は、[キャンセル] をクリックします。

4) [ファイル(F)] > [新規(N)] > [COBOL プロジェクト] を選択します。

ファイル(F)	編集(E)	リファクタリング	ナビゲート(N)	検索	プロジェ	7Ի(P)	実行(R)	ウィンドウ(W)	ヘルプ(H)	
新規	(N)		A	lt+シフト	+N>]	2	COBOLプロ	リジェクト		
						- 0				

	新規(N)	Alt+ソフト+N≯		COBOL JUNINE	
	ファイルを開く(.)		2	COBOL コピーファイル プロジェクト	
È,	ファイル・システムからプロジェクトを開く		۲ Ξ	リモート COBOL プロジェ COBOL プロジェクトを作成します。	Ì
	最近のファイル	>	ġ	リモート COBOL コピーファイル プロジェクト	

5) 以下の入力を行い、[終了(F)] をクリックします。

プロジェクト名: "NativeCOBOL"

プロジェクトテンプレート: 32 ビット、64 ビット、いずれを選択しても構いません

OBOL プロジェクト フークスペースまたは外部の場所にCOBOL プロジェクトを作成します。	
プロジェクト名(<u>P</u>): NativeCOBOL	
プロジェクト テンプレートを選択	
<u>テンプ</u> テンプレートの参照	ブレートの設定を構成
場所:	参照
ファイルシステムを選択: default 🗸	
✓ デフォルト・ロケーションの使用(D)	
ロケーション(<u>L</u>): C:¥workspace-jvmws¥NativeCOBOL	参照(<u>R</u>)
ファイル・システムを選択(<u>Y</u>): デフォルト ~	

 NativeCOBOL プロジェクトを選択し、マウスの右クリックにてコンテクストメニューを開き、[インポ -ト(I)] > [インポート(I)] を選択します。

	インポート(i)	\rightarrow	٩,	リモート プロジェクト
4	エクスポート(O)		寄	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
\$	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)		2	インポート(I)

7) 一般フォルダ配下の [ファイル・システム] を選択して、[次へ(N)] をクリックします。

インポート・ウィザードの選択(<u>S</u>):	
フィルタ入力	
 → 一般	
> 🔁 Oomph	
C TextMate	

8) C:¥jvmRESTfulWSTutorial¥COBOL 配下の3つのファイルを選択し、[終了(F)] をクリックします。

ファイル・システム ローカル・ファイル・システムオ	ゔらリソースをインポートしま	きす。				
次のディレクトリーから(<u>Y</u>):	C:¥jvmRESTfulWSTut	orial¥COBOL			~	参照(<u>R</u>)
● COBOL Ø1ブをフィルター(D)	すべて選択(5)	選択をすべて解除()		Ccbl (-INFO.cpy (SCRN.cbl]	
インポート先フォルダ(止):	NativeCOBOL					参照(<u>₩</u>)
オプション	リソースを上書き(<u>Q</u>) ダーを作成(<u>C</u>)					
?			< 戻る(<u>B</u>)	次へ(<u>N</u>) >	終了(E)	キャンセル

9) NativeCOBOL プロジェクトを選択し、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た上で、 [プロパティ(R)] を選択します。



10) [Micro Focus] > [ビルド構成] > [リンク] を選択し、「エントリポイント」に "BOOKSCRN" を入力したうえで、[適用(L)] をクリックします。

フィルタ入力	リンク	← → ⇒ < §
> リソース Coverage	New Configuration [使用中]	~ 構成の管理
 Micro Focus ビルダー ビルド パス ビルド構成 	71ルタテキストを入力 (三)	
> COBOL イベント	設定 Maintage	値
ディプロイ	◆ Linkage 出力の名前	NativeCOBOL
ビルト環境	出力パス	New Configuration bin
> リノク 、 プロジェクト設定	エントリポイント	BOOKSCRN
指令の確定	ターゲットの種類	千 天口の8827177
> 実行時構成	ビット数	64 ビット
WikiText		レルズ
サーバー タスク・タグ タスク・リボジトリー ビルダー プロジェクト・ネーチャー プロジェクト・マーチャー プロジェクト・ショート プロジェクトや照 * 検証 **********************************	エントリポイント ビルドファイル内のデフォルトのエントリポイント る	指定します
実行/テバック設定		適用して閉じる キャンセル

- [Micro Focus] > [プロジェクト設定] > [COBOL] を選択し、以下の入力を行ったうえで、
 [適用して閉じる] をクリックします。
 [ソースエンコーディング]: "ANSI"
 - [追加指令]: "ASSIGN(EXTERNAL)"

設定	値
✔ 一般	
文字セット	ASCIL
ソース エンコーディング	ANSI
COBOL 方言	Micro Focus
ソース フォーマット	固定
デバッグ用にコンパイル	はい
EXIT PROGRAM を GOBACK として処理	ANSI
詳細	いいえ
.GNT にコンパイル	いいえ
・出力	
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
・ エラー/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	ASSIGN(EXTERNAL)
加指令	~
ソパイラに渡す追加のCOBOLコンパイラ指令です	
)BOL コンパイル設定:	
HARSET"ASCII" SOURCE-ENCODING"ANSI" DIALE KITPROGRAM"ANSI" NOTESTCOVER NOPROFILE V :XTERNAL)	CT"MF" SOURCEFORMAT"fixed" NOLIST anim VARNING"1" MAX-ERROR"100" ASSIGN
	デフォルトの復元(I) 適用(<u>L</u>

3.1.2 アプリケーションの確認

1) NativeCOBOL プロジェクトを選択し、[実行(R)] > [実行構成(N)] を選択します。

実行	(R) ウィンドウ(W)	ヘルプ(H)
2	実行点をリセット	
Q	実行(R)	Ctrl+F11
椮	デバッグ(D)	F11
	実行履歴(T)	>
0	実行(S)	>
	実行構成(N)	

 COBOL アプリケーションを選択し、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た上で、[新 規構成(W)]をクリックします。

構成の作成、管理、および実行

COBOL プログラムを実行します



3) 「名前」に "NativeCOBOL" を入力します。

構成の作成、管理、および実行

COBOL プログラムを実行します

📑 🖻 🐌 🗎 🗶 🖻 🍸 🗸	名前(N): NativeCOBOL	
フィルタ入力	🕟 一般 🧤 ソース 🌄 環境 🔲 共通(C) 🔎 実行時	🏷 デバッグシンボル
Apache Tomcat	▼ COBOL プロジェクト(P)	
Aspect/ Java Application	NativeCOBOL	参照

4) [環境]タブを選択し、[追加(A)] をクリックします。



5) 以下の入力を行い、[OK] をクリックします。

変数: "BOOKINFO"

值: "C:¥jvmRESTfulWSTutorial¥DAT¥BOOKINFO.DAT"

Ē	
DKINFO.DAT	
ОК	キャンセル
	DKINFO.DAT

6) BOOKINFO 変数が追加されたことを確認して、[実行(R)] をクリックします。

 高 一般 (注: ここで定: 環境スクリプ) 	ソース ■ 環境 □ 共通(C) ● 実行時 □ デ 美された変数は、任意の現在の設定値、または任意の ト内の設定値を上書きします。)	(ッグシンボル 🛃 動的分析 崉 CTF 🥒 コン 指定された	/テナー	
変数 BOOKINFC	2	値 C:¥jvmRESTfulWSTutorial¥DAT¥BOOKIN	NFO.DAT	追加(A) 編集(E) 削除(R)
_実行する環 場所: パラメータ: □ 関連付けら	境スクリプト: ファイルがプロジェクト内にある場合、絶対パスは相対パ: 、 、 れたプロジェクトのビルド環境から値を継承	ನಿದ್ದರುಕ್ಕೆ		参照
		ŝ	前回保管した状態に戻す(V) 実行(R)	適用(Y) 閉じる

以下のような画面が表示されます。

an NativeCOBOL	-		×
FUNCTION: [_] READ=1, ADD=2, DELETE=3 NEXT=4, GROSSSALES=S,	QUIT	=9	
ストック番号: []			
タイトル: []		
ジャンル: []			
著者: []]			
小売価格: [0] 仕入れ: [0] 売上: [000000000]			
STATUS: []			

以下の入力を行った後、Enter キーを押すと、該当書籍情報が表示されます。 FUNCTION: "1" ストック番号: "1111"

C:1.	NativeCOBOL	-		×
	PUNUTION: [1] REAU-1, ADD-2, DELETE-3 NEXT-4, GRUSSSALES-S, ストック番号: [1111]	QUII-	9	
	タイトル: [LOAD OF THE RING]		
	ジャンル: [FANTASY]			
	著者: [TOLKIEN]			
	小売価格: [1500] 仕入れ: [2000] 売上: [000001000]			
	STATUS: [00]			

現在、ストック番号:4444 は未登録のため、以下の入力を行い、Enter キーをクリックします。 FUNCTION: "2" ストック番号: "4444" タイトル: "鏡の国のアリス" ジャンル: "ファンタジー" 著者: "ルイス・キャロル" 小売価格: "1200" 仕入れ: "3000" 売上: "4000" NativeCOBOL - 0 FUNCTION: [2] READ=1, ADD=2, DELETE=3 NEXT=4, GROSSSALES=S, QUIT=9 ストック番号: [4444] タイトル: [鏡の国のアリス ジャンル: [ファンタジー 著者: [ルイス・キャロル 小売価格: [1200] 売上: [000004000] 仕入れ: [3000] STATUS: [00]

```
実行後、以下の入力を行い、書籍情報が登録されていることを確認してください。
FUNCTION: "1"
ストック番号: "4444"
確認後、以下の入力を行い、Enter キーを押すことで登録した書籍情報が削除されます。
FUNCTION: "3"
ストック番号: "4444"
```



実行後、READ 機能でストック番号: 4444 が削除されていることを確認してください。

FUNCTION=Sの GROSSSALES 機能は登録された全書籍情報の売上額を集計します。



確認後、「FUNCTION」に"9"を入力し、Enter キーを押してプログラムを終了します。

3.2 JVM COBOL プロジェクトの作成

本節では、さきほど確認した従来の COBOL プログラムを JVM COBOL としてコンパイルを行います。

Visual COBOL for Eclipse メニューより、[ファイル(F)] > [新規(N)] > [COBOL JVM プロジェクト] を選択します。

ファイ	ル(F) 編集(E)	リファクタリング	ナビゲート(N)	検索	プロジェ	/ / (P)	実行(R)	ウィンドウ(W)	ヘルプ(H)
	新規(N)			Alt+シフ	ト+N>)	2	COBOLプロ	リジェクト	
	ファイルを開く(.)					📸	COBOL I	ーファイル プロジ	ェクト
	ファイル・システムた	いらプロジェクトを	開く			ġ	IJモ−ト COE	BOL プロジェクト	
	最近のファイル				>	ġ	IJモ−ト COE	BOL コピーファイノ	レプロジェクト
	問じる(0)			Ctr		Ē	COBOL 1	ニット テスト プロシ	バェクト
	すべて閉じる(1)		(`trl+シフト	+W	2	COBOL JVN	M プロジェクト	

2) 「プロジェクト名」に "BookLibrary" を入力して、[終了(F)] をクリックします。

プロジェクトキ(P): BookLibrary ダブオルト・ロケーションの使用(D) ロケーション(D): ロケーション(D): CWworkspace-jernwarWBookLibrary プロジェクト・テンプレートを選択	COBOL JVM プロジェクト ワークスペースまたは外部の場所に COBOL JVM プロジェクトを作成しま	す。			E
□ アージョン(1): [LiWorkspace-jumws¥BookLibrary 季酉(R) プロジェクト テンプレートを豊沢 □ アンブレートの登録 ■ アンブレートの登録 場所:	プロジェクト名(P): BookLibrary				
リケーション(リ:) CV#workspace-jvmws4BookLibrary 参照(P) プロジェクト キンブレートを選択 「ごがい- ForCus テンプレート アンプレートの設定を進成 アンパルシステムを選択: ● 気行環境 RE の使用(N): ● 気行環境 RE の使用(S): ● 人口タングーを知んごのでのいしん、and workspace compiler preferences アーキング・セットにプロジェクトを追加(T) アーキング・セット(O):	✓ デフォルト・ロケーションの使用(D)				
プロジェクト テップレートを選択 デンプレートの設定を提成。 「テンプレートの設定を提成。 「テンプレートの設定を提成。 アンプレートの設定を提成。 アーキング・セットにプロジェクトを追加(T) アーキング・セットにプロジェクトを追加(T) アーキング・セット(O): 第提(W) アーキング・セット(O):	ロケーション(L): C:¥workspace-jvmws¥BookLibrary				参照(R)
アンプレートの設定を進め。 アンプレートの参照 「日 アンプレートの参照 アンプレートの参照 「日 アングレットのでのに、 アングレートの考知 アングレートの使用(N): 「日 中ングーレー・ アング・セット(ロ): 第111111111111111111111111111111111111	プロジェクト テンプレートを選択				
JRE ● 実行環境 JRE の使用(V): ○ プロジェクト回有の JRE を使用(S): AdoptOpenJDK ○ Use default JRE 'AdoptOpenJDK' and workspace compiler preferences アーキング・セット □ ワーキング・セットにプロジェクトを追加(T) 第規(W) ワーキング・セット(C):	 Micro Focus テンプレート テンプレートの参照 場所: ファイル・システィ(宏谟好: default w) 			<u>テンプレー</u>	- <u>の設定を構成</u> 参照
 ● 実行環境 JRE の使用(V):	JRE				
○ プロジェクト国有の JRE を使用(S): AdoptOpenJDK ○ Use default JRE 'AdoptOpenJDK' and workspace compiler preferences JRE を構成 ワーキング・セット □ ワーキング・セットにプロジェクトを追加(T) 新規(W) ワーキング・セット(C): 選択(E) ② <定う(B) 次へ(N) > 終了(F) キャンセル	 実行環境 JRE の使用(V): 		JavaSE-17		~
○ Use default JRE 'AdoptOpenJDK' and workspace compiler preferences RE を構成 ワーキング・セット □ ワーキング・セットにプロジェクトを追加(T) 新規(W) ワーキング・セット(O):	〇 プロジェクト固有の JRE を使用(S):		AdoptOpenJDK		\sim
ワーキング・セット □ ワーキング・セットにプロジェクトを追加(1) □ テーキング・セット(0):	O Use default JRE 'AdoptOpenJDK' and workspace compile	er preferences			<u>JREを構成</u>
(?) <戻る(B) 次へ(N) > 終了(F) キャンセル	ワーキング・セット ワーキング・セットにプロジェクトを追加(T) ワーキング・セット(O):			~	新規(W) 選択(E)
	(?)	< 戻る(B)	次へ(N) >	終了(F)	キャンセル

BookLibrary プロジェクトを選択し、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たうえで、[プロパティ(R)]を選択します。

プロパティ(R)	Alt+Enter
ソース(S)	>
構成	>
比較対象(A)	>
チーム(E)	>

4) [Micro Focus] > [ビルド構成] を選択し、「追加指令」に "ASSIGN(EXTERNAL)
 ILSMARTLINKAGE ILNAMESPACE(com.microfocus.sample)"を設定し、[適用して閉じる] をクリックします。



補足)

ILSMARTLINKAGE コンパイラ指令により、COBOL と Java 言語でのデータ型の差異を吸収する ラッパークラスがコンパイル時に自動生成されます。本指令の詳細については、製品マニュアルトップより、 [リファレンス] > [コンパイラ指令] > [コンパイラ指令 - アルファベット順一覧] > [ILSMARTLINKAGE] を参照してください。

ILNAMESACE コンパイラ指令は、JVM COBOL の出力先のパッケージ階層を指定するもので、今回 は、com.microfocus.sample パッケージ配下に出力します。本指令の詳細については、製品マニュ アルトップより、[リファレンス] > [コンパイラ指令] > [コンパイラ指令 - アルファベット順一覧] > [ILNAMESPACE] を参照してください。

5) BookLibrary プロジェクト配下の src フォルダを選択し、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た上で、[新規作成(N)] > [COBOL JVM パッケージ]を選択します。

■▲ 参照ライブ		新規作成(N)	>	솔	COBOL JVM プロジェクト
> 🛃 NativeCOBO		表示方法(W) 型階層を開く	Alt+シフト+W > Alt+シフト+H	COBOL JVM ユニットテストプロ を COBOL コピーファイル プロジェク	COBOL JVM ユニット テスト プロジェクト COBOL コピーファイル プロジェクト COBOL プロジェクト
		コピー(C) 修飾名のコピー(Y)	Ctrl+C		COBOL フニット テスト プロジェクト リモート COBOL JVM プロジェクト
	Ē	貼り付け(P)	Ctrl+V	1	リモート COBOL コピーファイル プロジェクト
	×	削除(D)	削除	官	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
		リファクタリンク(T)	>		プロジェクト(R)
		タスクのスキャン		D\$	COBOL วษี-วฐศ แ
		インポート(i)	>		COBOL JE 77 17
	4	エクスポート(O)		C	スタンドアロン ファイル
	\$	更新(F)	F5	Ľ	リモート スタンドアロン ファイル
アウトラインを提供する	Q.	Coverage As	>	Ø	COBOL JVM Delegate
	0	実行(R)	>	G	COBOL JVM Enum
	*	デバッグ(D)	>	Ø	COBOL JVM Value Type
		プロファイル(P)	>	O	COBOL JVM インターフェイス
		ローカル履歴から復元(Y)		G	COBOL JVM 252
		Maven	>	A Y	
	\checkmark	検証		₩.	COBOL JVMI // 9/7-9

6) 「名前」に "com.microfocus.sample" を入力して、[終了(F)] をクリックします。

COBOL JVM /(COBOL JVM /(ም	ッケージ ケージを新規作成します。		
パッケージに対応す	るフォルダを作成します。		
ソ−ス フォルダ(<u>D</u>):	BookLibrary/src		参照(<u>o</u>)
名前(<u>M</u>):	com.microfocus.sample		
?		終了(白	キャンセル

src フォルダ配下に、空のパッケージが作成されます。

Es COB	0 🛛	琉 Navig	eg Applic
∽ ピ B > 🛋	ookLibi COBO	閉じる L JVM 実行時3	
> 🛋	JRE シ	ステム・ライブラリ・	- [JavaSE-11]
 	src 🖁		
	🕀 coi	m.microfocus.s	ample
=	参照う	イブラリー	

 さきほどの NativeCOBOL プロジェクトの「コピーファイル」配下の "BOOK-INFO.cpy"を選択し、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たうえで、[コピー]を選択します。

 ✓ 浸 NativeCOBOL > / / / 回 COBOL プログラム マ () コピーファイル 		
BOOK-INFO.cpy > > Prev_Configuration.bi	新規作成(N)	>
	開く(O) 表示方法(W) コンテキスト内で開く アプリケーションから開く	Alt+シフト+W > > >
	וייי	Ctrl+C
	1 貼り付け	Ctrl+V

 BookLibrary プロジェクト配下の src¥com.microfocus.sample フォルダを選択し、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たうえで、[貼り付け(P)]を選択します。

 ✓ ピ BookLibrary > ▲ COBOL JVM 実行時システム > ▲ JRE システム・ライブラリー [JavaSE-1 > 伊 src 	1]		
 Gom.microfocus.sample 参照ライブラリー 参照ライブラリー ※ NativeCOBOL		新規作成(N) 表示方法(W) 型階層を開く	> Alt+シフト+W> Alt+シフト+H
		コピー(C) 修飾名のコピー(Y)	Ctrl+C
	(ii)	貼り付け(P)	Ctrl+V

この操作により、BOOK-INFO.cpy が com.microfocus.sample フォルダ配下にコピーされます。

 さきほどと同様の手順で、NativeCOBOL プロジェクトの「COBOL プログラム」配下の "BOOK.cbl" を BookLibrary プロジェクト配下の src¥com.microfocus.sample フォルダにコピーしてください。 コピー後は以下のように表示されます。



デフォルトで自動的にビルドが有効になっている場合、BOOK.cbl をコピーした段階でコンパイルが行われ、プロジェクトが保存されたフォルダ配下の bin¥com¥sample フォルダに、以下の .class ファイル、 すなわち、Java バイトコードが生成されます。 « workspace-jvmws > BookLibrary > bin > com > microfocus > sample

BOOK\$_MF_LCTYPE_1.class

- BOOK.cbldat
- BOOK.class
- LnkBDetails.cbldat
- LnkBDetails.class
- LnkFileStatus.cbldat
- LnkFileStatus.class
- LnkFunction.cbldat
- LnkFunction.class

補足)

自動的にビルドする設定は、[プロジェクト(P)] > [自動的にビルド(M)] で確認できます。チェックされて いる場合は自動的にビルドが行われます。 プロジェクト(P) 実行(R) ウィンドウ(W) ヘルプ(H)

	705	イエクト(E) 美口(E) クイフトウ(M	
9		プロジェクトを開く(E) プロジェクトを閉じる(S)	
•••		フロジェクドを同じる(3) 	Ctrl+B
-	010	プロジェクトのビルド(B)	Cui+b
		ワーキング・セットのビルド(W)	>
		クリーン(N)	
	~	自動的にビルド(M)	

3.3 Apache Tomcat 上で動作する RESTful Web サービスの作成

本節を実行するにあたり、Visual COBOL for Eclipse 上で、予めサーバー設定を実施してください。この設 定は、以降の手順で使用する J2EE パースペクティブのデフォルトで表示されるサーバービューで行うことができ ます。また、異なるパースペクティブを使用している際でも、Eclipse のメニューより、[ウィンドウ(W)] > [ビュー の表示(V)] > [その他(O)] を選択した上で表示されるダイアログ上で、以下のビューを選択することで行えま す。

🧿 ビューの表示	—		×
<u> 7ብሥንእታ</u>			×
> 📂 Mylyn			
> 🥭 Oomph			
Version Control (Team)	9		
> 🗁 visualiser			
			- 11
↓ ↓ ↓ − パー			
> (=> ターミナル			-11
> 🗁 デバッグ			
> 🗁 データ管理			
> 🗁 プラグイン開発			
> 🗁 ヘルプ			
> 📂 リモート・システム			
> 🗁 その他			
開<(o)		キャンセ	JU
本手順では 以下のように	Tomca	at 10	のサーノ
🚜 サーバー 🕴 📃 コンソール	2 🔝 問	題 瘪	タスク [
📙 Tomcat 10 [停止, 再分	公開]		

3.3.1 Java Web アプリケーションプロジェクトの作成

 [ウィンドウ(W)] > [パースペクティブ(R)] > [パースペクティブを開く(O)] > [その他(o)] を選 択します。

ウィンドウ(<u>W</u>) ヘルプ(<u>H</u>)			
新規ウィンドウ(N)			
エディター	>		
外観	>		
ビューの表示(V)	>		
パースペクティブ(R)		書 パースペクティブを開く(O) →	参 デバッグ
ナビゲーション(G)	>	パースペクティブのカスタマイズ(Z)	
設定(P)		パースペクティブの別名保管(A)	目目 リモート・システム・エクスフローラー
BARE(F)	_	パースペクティブのリセット(R)	その他(o)
		パースペクティブを閉じる(C) すべてのパースペクティブを閉じる(L)	

2) 「J2EE」を選択して、[開く(o)] をクリックします。

Aspect Visualization				
💷 COBOL (デフォルト)				
6월 CTF				
🔚 Git		_		
₽ [®] J2EE				
🎳 Java				
^と Java の型階層				
🔊 Java 参照				
♦ JPA 開発				
🮯 Web				
X XML				
≝ ⁰ チーム同期化				
☆ デバッグ				
₿ データベース・デバッグ				
🔓 データベース開発				
◆プラグイン開発				
各リソース			 	
		BB //)	de reducter II	
	l	開((0)	キャンセル	

3) [ファイル(F)] > [新規(N)] > [その他(o)] を選択します。

ファイ	ル(F) 編集(E) ナビゲート(N) 検索	プロジェクト(P) 実	[行(R)	ゥ	ィンドウ(W) ヘルプ(H)	
	新規(N)	Alt+シフト+M		1	Maven Project	
	ファイルを開く(.)		- (Ċ	エンタープライズ・アプリケーション・プロジェクト	
	ファイル・システムからプロジェクトを開く		(T\$	動的 Web プロジェクト	
	最近のファイル		>	`	EJB プロジェクト	
	- 閉じる(C)	Ctrl+V	V	a	コネクター・プロジェクト	
	すべて閉じる(L)	Ctrl+シフト+V	v	R	アプリケーション・クライアント・プロジェクト	
			(Ŷ	静的 Web プロジェクト	
	保存(S)	Ctrl+	S	H	JPA プロジェクト	
	別名保存(A)		1	Ĵ	プロジェクト(R)	
6	すべて保管(E)	Ctrl+シフト+	S	s	CSS ファイル	
	前回保管した状態に戻す(T)				JavaScript ファイル	
	移動(V)			*	JSP ファイル	
2	名前を変更(M)	F	2	8	サーブレット	
\$	更新(F)	F	5 [ŝ	Session Bean (EJB 3.x)	
	行区切り文字の変換(D)		> [-	Message-Driven Bean (EJB 3.x)	
	印刷(P)	Ctrl+	P .	1	Web サービス	
			- (Ŷ	フォルダー	
è	インボート(I)			C [°]	ファイル	
4	エクスホート(O)			A	Aspect	
	プロパティ(R)	Alt+Ente	er	Ŷ	サンプル(X)	
	ワークスペースの切り替え(W)		>	•	この他の	Ctrl+N
	= 00		L	<u> </u>	CO/18(O/	Cur+IN

4) [Maven] > [Maven Project] を選択し、[次へ(N)]をクリックします。



以下の入力を行い、[終了(F)]をクリックします。

Group Id: "com.microfocus.sample"

Artifact Id: "JVMRESTfulWS"

Version: "0.0.1-SNAPSHOT"

Packaging: "war"

New Maven project

Configure project

~	5
V. I	
_	

Artifact	
Group Id:	com.microfocus.sample ~
Artifact Id:	JVMRESTfulWS
Version:	0.0.1-SNAPSHOT V
Packaging	war V
Name:	~
Description:	:
Parent Proje	ect
Group Id:	~
Artifact Id:	~
Version:	Browse Clear
Advanced	
?	< 戻る(<u>B</u>) 次へ(<u>N</u>) > 終了(<u>F</u>) キャンセル

JVMRESTfulWS プロジェクトが作成されます。



6) JVMRESTFulWS プロジェクトを選択し、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たう

えで、[プロパティ(R)]を選択します。

「 プロパティ(R)	Alt+Enter
ソース(S)	>
構成	>
比較対象(A)	>
チーム(E)	>

7) [プロジェクト・ファセット]を選択し、「Java」に "11"を選択し、「JavaScript」と「動的 Web モジュール」にチェックを入れて[適用(L)]をクリックします。※[プロジェクト・ファセット]を選択した際、「Convert to faceted from…」リンクが表示される場合は、リンクをクリックしてください。



8) [Java のビルド・パス] から [ライブラリー] タブを選択します。「クラスパス」を選択したうえで、

[ライブラリーの追加(i)] をクリックします。

7ィルタ入力	Java のビルド・パス	⟨¬ ▼ ¬ ▼
> Javaエディタ > Javaコード・スタイル > Javaコンパイラー	アース(S) プロジェクト(P) オーブラリー(L) シース(S) プロジェクト(P) マール依存開 ビルド・パス上の JAR およびクラス・フォルダー(I):	係(M)
Java のビルド・バス JSP フラグメント	✓ ♣ モジュールパス ■ JPF 277 A·ライブラリー [JavaSE-11]	JAR の追加(J)
> Maven Web コンテンツの設定値 Web プロジェクトの設定	✓ ♦₀ / סלגולג > = may may be pendencies	外部 JAR の追加(<u>X</u>) 変数の追加(<u>V</u>)
Web ページ・エディター WikiText		ライブラリーを追加(i)
> XDoclet > クライアント・サイド JavaScri		クラス・フォルダの追加(<u>C</u>)…
<i>₩</i> - <i>N</i> -		外部クラス・フォルダーを追加(<u>D</u>)

9) [COBOL JVM 実行時システム] を選択したうえで、[次へ(N)] をクリックします。



次の画面では、そのまま [終了(F)] をクリックすると、「COBOLJVM 実行時システム」が追加 されます。



10) [プロジェクト]タブを選択し、「クラスパス」を選択したうえで、[追加(D)] をクリックします。

● プロパティ: JVMRESTfulWS		— 🗆 X
フィルタ入力	Java のビルド・バス	← → ⇒ 8
> リソース Coverage Deployment Assembly Javador ロケーション	(通 ソース(S) G ブロジェクト(P) ▲ ライブラリー(L) % 順序およびエクスポート(O) G モジュール依存関係(M) ビルド・パス上に必要なプロジェクト(B):	
> JavaIデイタ		追加(<u>D</u>)
> Java コート・スタイル > Java コンパイラー		編集(<u>E</u>)
Java のビルド・パス JSP フラグメント		除去(<u>M</u>)

11)「BookLibrary」プロジェクトにチェックを入れたうえで、[OK] をクリックします。

💿 必要なプロジェクトの選択	—		×
追加するプロジェクトを選択してください:			
 ✓ [™] BookLibrary □ [™] NativeCOBOL 			
すべて選択(S)	選拶	そをすべて解	¥除(D)
ОК		キャンセ	zJV

BookLibrary プロジェクトが追加されたことを確認したうえで、[確認(L)] をクリックします。

Java のビルド・バス	← → ⇒ 8
(巻 ソース(S) ラゴジェクト(P) 「ライブラリー(L) 🍫 順序およびエクスポート(O) G モジュール依存関係(M)	
ビルド・パス上に必要なプロジェクト(<u>R</u>):	
	追加(<u>D</u>)
> C BookLibrary	編集(<u>E</u>)
	除去(<u>M</u>)
	適用(<u>L</u>)

12) [Deployment Assembly] を選択し、[追加(D)] をクリックします。

フィルタ入力	Web 展開アセンブリ		← → ⇒ 8
> UV-X	Define packaging structure for this Java EE We	o Application project.	
Deployment Assembly	· ソース ^	デプロイ・パス	追加(<u>D</u>)
Javadoc D / - / 3/	🗀 /src/main/java	WEB-INF/classes	(5.5 (D)
> Javaエディタ	/src/main/resources	WEB-INF/classes	福朱(上)
> Java コード・スタイル	/src/main/webapp	□ /	除去(R)
> Java コンバイラー	/target/m2e-wtp/web-resources	□ /	
Java のビルド・バス	Maven Dependencies	🗀 WEB-INF/lib	
> Maven			

13) [Java ビルド・パス・エントリー] を選択し、[次へ(N)] をクリックします



[COBOL JVM 実行時システム] を選択し、[終了(F)] をクリックします。

J ava ビルド・パス Select build path er	・エントリー tries to include in the deplo	oyment assem	bly.	a
> 🛋 COBOL JVM	実行時システム			
?	< 戻る(B) 次へ	(N) >	終了(F)	キャンセル

14) さきほど同様、[追加(D)] をクリックします。

Veb 展開アセンブリ		
Define packaging structure for this Java EE We	b Application project.	
у-д ^	デプロイ・パス	追加(<u>D</u>)
🗀 /src/main/java	WEB-INF/classes	(四年(四)
/src/main/resources	WEB-INF/classes	福未(三)
🗀 /src/main/webapp	🗀 /	除去(R)
/target/m2e-wtp/web-resources	<u> / / / / / / / / / / / / / / / / / </u>	
🛋 COBOL JVM 実行時システム	🗀 WEB-INF/lib	
🛋 Maven Dependencies	🗀 WEB-INF/lib	

15)「プロジェクト」を選択し、[次へ(N)] をクリックします。

ディレクティ アセンブリーディ	プの種類を選択 レクティブを新規追加しま	ξġ.		
■ Java つ ファイ) つ フォル つ フォル つ フ・ク つ ワーク つ 変数	ビルド・パス・エントリー レ・システム由来のアーカ・ ダー エクト スペース由来のアーカイブ	(7		
?	< 戻る(B)	次^(N) >	終了(F)	キャンセル

「BookLibrary」を選択し、[終了(F)] をクリックします。

プロジェクト Select projects to	include in the deplo	yment assembly.		
🔁 BookLibra	у			
?	< 戻る(B)	次へ(N) >	終了(F)	キャンセル

以下のように追加した項目が表示されていることを確認して、[適用して閉じる]をクリックします。

fine packaging structure for this laws FF Web	Application project	
	Application project.	:自力(D)
		迎加(0)
/src/main/java	WEB-INF/classes	編集(E)
/src/main/resources	U WEB-INF/classes	
/src/main/webapp		除去(R)
/src/test/java	WEB-INF/classes	-
src/test/resources	U WEB-INF/classes	-
/target/m2e-wtp/web-resources		-
BookLibrary	WEB-INF/lib/BookLibrary.jar	
N COBOL JVM 実行時システム	WEB-INF/lib	-
Maven Dependencies	WEB-INF/IID	
		-
		-
		-
	前回保管した状態に戻す(V)	適用(L)
	適用して閉じる	キャンヤル

16) C:¥jvmRESTfulWSTutorial¥Java¥src 配下の com フォルダをクリップボードにコピーした

うえで、JVMREstfulWS プロジェクト配下の src/main/java フォルダを選択した状態でマウ

		/3
✓ W JVMRESTfulWS > 図 デプロイメント記述 > 2 JAX-WS Web サ・	子 : -ビス	
> (= src/main/java > (= src/main/resou > (= src/test/java	新規(N) 次へジャンプ(I)	>
> M src/rest/resourc > A JRE システム・ライ > A COBOL JVM 実: > C Deployed Resource	型階層を開く(N) 表示方法(W) ローカル・ターミナルで表示	F4 Alt+シフト+W > >
> 🦢 target	┣ 」ピー(C) ● 修飾名のコピー(Y)	Ctrl+C
> 🤃 NativeCOBOL > 🔁 サーバー	幅 貼り付け(P)★ 削除(D)	Ctrl+V 削除

スの右クリックにてコンテクストメニューを表示して、[貼り付け(P)]を選択します。

この時点ではエラーになりますが、ここでは無視します。

- 17) C:¥jvmRESTfulWSTutorial¥Java¥webapps 配下の WEB-INF フォルダをクリップボー ドにコピーしたうえで、JVMRESTfulWS プロジェクト配下の src¥main¥webapps フォルダを 選択し、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たうえで、[貼り付け(P)] を選択しま す。
- 18) JVM RESTfulWS プロジェクト配下の pom.xml の中身を以下のファイルで上書き保存します。 C:¥jvmRESTfulWSTutorial¥Java¥pom.xml



19) pom.xml を選択し、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たうえで、[Maven] >

[Update Project] を選択します。

蓉	デバッグ(D) プロファイル(P)	> >	記述/説明 > Java 例外ブレークポイ
	Maven	>	Add Dependency
\checkmark	検証		Add Plugin
	<i>Ŧ−</i> <u>/</u> ₄ (E)	>	New Maven Module Project
	比較対象(A)	>	Update Project

JVMRESTfulWS にチェックがついていることを確認したうえで [OK] をクリックしてください。

Update Maven Project

Select Maven projects and update options		
Available Maven Codebases		
VI V		Select All
		Add out-of-date
		Deselect All
		Expand All
		Collapse All
Offline		
Update dependencies		
Force Update of Snapshots/Releases		
Update project configuration from pom.xml		
Clean projects		
?	ОК	キャンセル

20) サーバービューを開き、「Tomcat 10」サーバーをダブルクリックします。

🔝 र-म	- 🗆 🗆	プロパティー	劇サ	-11- 🛙
🔓 To	omcat 10) [停止, 再	[公開]	

21) [起動構成を開く] リンクをクリックします。

+++++++++++++++++++++++++++++++++++++++	
侃モ	
	儆要

一般情報		
ホスト名とその	他の共通設定を指定します。	
サーバー名:	Tomcat 10	
ホスト名:	localhost	
<u>52974:</u>	Tomcat 10	~
構成パス:	/サーバー/Tomcat 10-config	参照
起動構成を	<u>開く</u>	

22) [環境]タブを開き、[追加(A)] をクリックします。

起動構成プロパティの編集

名前(N): Tomcat 10	🍫 クラスパス 📴 ソース 🚾 環境 🛄 共通(C)	
設定する環境変数(S) 変数	値	追加(A)
		編集(D)

- 23) 以下の入力を行い、[OK] をクリックします。
 - 名前: "BOOKINFO"

值: "C:/jvmRESTfulWSTutorial/DAT/BOOKINFO.DAT"



BOOKINFO 変数が追加されたことを確認して、[OK] をクリックします。

_

	ヘ//ヘ ♥ 2-/ ■ 境場 [□ 共進(5)]	
変数	値	追加(A)
OOKINFO	C:/jvmRESTfulWSTutorial/DAT/BOOKINFO.DAT	選択(L)
		編集(D)
		削除(O)
		⊐ピ-(C)
		貼り付け(P)
)ネイティブ環境への環境の追)ネイティブ環境を指定された	カJ(A) 零境と置換(P)	
	前回保管した	:状態に戻す(V) 適用(Y)

24) [モジュール]タブを選択し、[Web モジュールの追加] をクリックします。

Tomcat 10 🔀				-
Web モジュール				
Web モジュール				
このサーバーで構成する W	eb モジュール。			
パス	文書ベース	モジュール	自動再ロード	Web モジュールの追加…
				外部 Web モジュールの追加
				編集
				削除

概要 モジュール

JVMRESTfulWS を選択し、[OK] をクリックします。

モジュール(M):	G JVMRESTfulWS
文書ベース(D):	JVMRESTfulWS
パス(A):	/JVMRESTfulWS
	ОК + +>>セル

Eclipse IDE のメニューより [ファイル(F)] > [保存(S)] を選択し、変更を保存してください。

771	ル(<u>F</u>)	編集(<u>E</u>)	ナビゲート(<u>N</u>)	検索	プロジェクト(<u>P</u>)	実行(<u>R</u>)
	新規 ファイ ファイ 最近	(N) ルを開く(.). ル・システム のファイル	 からプロジェクト	を開く	Alt+シフ	⊦+N > >
	閉じる すべて	ō(C) て閉じる(L)			Ctr Ctrl+シフト	l+W ∽+W
H	保存	(S)			Ci	trl+S

3.3.2 RESTful Web サービスの確認

1) 「Tomcat 10」サーバーを選択し、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たうえで、 [起動(S)] を選択します。

概要 モジュール		起動(S)	Ctrl+Alt+R
	Ď	プロファイル(F)	
		停止(T)	Ctrl+Alt+S
> 📙 Iomcat 10 [停止, 再公開]		公開	Ctrl+Alt+P

起動すると、ステータスが "始動済み" に変更されます。

> 🖥 Tomcat 10 [始動済み, 同期済み]
また、コンソールビューでもサーバーの起動を確認できます。
શ マーカー 🔲 プロパティー 🖏 サーバー 🏙 データ・ソース・エクスプローラー 📔 スニペット 📮 コンソール 🙁
Tomcat 10 [Apache Tomcat] C:¥Program Files (x86)¥Micro Focus¥Visual COBOL¥AdoptOpenJDK¥bin¥javaw.e
2月 24, 2023 2:51:44 午後 org.glassfish.jersey.server.wadl.WadlFeature configure 警告: JAXBContext implementation could not be found. WADL feature is disabled. 2月 24, 2023 2:51:45 午後 org.glassfish.jersey.internal.Errors logErrors
警告: The following warnings have been detected: WARNING: The (sub)resource method WARNING: The (sub)resource method getBookGrossSalesAmount in com.microfocus.sampl
2月 24, 2023 2:51:45 午後 org.apache.coyote.AbstractProtocol start 情報: プロトコルハンドラー ["http-nio-8080"] を開始しました。
2月 24, 2023 2:51:45 午後 org.apache.catalina.startup.Catalina start 情報: サーバーの起動 [4369] ミリ秒

2) C:¥jvmRESTfulWSTutorial¥WebClient 配下の Search.html を Web ブラウザーで 開きます。

Micro FocusDemo	HOME	書籍検索	在庫登録	在庫情報削除	売上集計	CONTACT
	1 mars			1		
ABC 書店在原	車管	理シ	ステ	4		
and I	1					1
書籍情報の検索				システ		_
ストック番号を入力の上、[検索]ボタンを	押下してくフ	ださい。		Home Pa	ge	
横寨				書籍検索	5	
				在庫情報	發登録	
				在庫情報	剥除	
				売上集計	t	
				Contact		

Apache Tomcat と JVM COBOL コンポーネントを利用した RESTful Web サービス開発

3) ストック番号に "1111" を入力し、[検索] をクリックすると、先に確認したように結果が画面上 に戻されます。

書籍情報の	検索 Material ACTION	システムメニュー
		Home Page
1111	模案	書籍検索
ストック番号	1111	在庫情報登録
タイトル	LOAD OF THE RING	在庫情報削除
ジャンル	FANTASY	売上集計
著者	TOLKIEN	Contact
小売価格	1500	
仕入れ数	2000	
売り上げ数	1000	
IF表示 api EndPoint http://localhost:8080/JVMRE Request (NO DATA) Response ("bookInfo": { "stockNo": "1111", "title": "LOAD OF THE RIN "author": "TOLKIEN "gener": "FANTASY	STfulWS/bookmgnt/books/1111 IG ",	

このように、Web 画面から Java で作成した RESTful Web サービスを介して COBOL ロ ジックを呼び出すことが確認できます。このインターフェースは、一般的な RESTful Web サービ ス仕様に基づいているため、COBOL 資産を活用した他システムとのデータ連携が容易に行え るようになります。

Eclipse IDE のサーバービューに戻り、[Tomcat 10] を選択し、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たうえで、[停止(T)] をクリックしてサーバーを停止します。

ms [[] _ m		再開(R)	Ctrl+Alt+R
🔝 マーカー 🔲 プロパティー 鵫 サーバー 🛛 🇌	Ď	プロファイルで再始動	
> 🗄 Tomcat 10 [始動済み, 同期済み]		停止(T)	Ctrl+Alt+S
		公開	Ctrl+Alt+P

補足)

今回は、Eclipse IDE 上から Tomcat アプリケーションサーバーを起動し、RESTful Web サービスを稼働させていましたが、Eclipse IDE を使用せず、Tomcat をはじめとしたアプリケー ションサーバー上でも今回のサービスを稼働させることができます。

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。

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。