

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 10
COBOL 環境製品	Visual COBOL 10.0 for Eclipse
Java アプリケーションサーバー	Apache Tomcat 10.1.28

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 10.1 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、任意のフォルダー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ー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:¥i	workspace-jvmws	~	参照(<u>B</u>)
 □ この選択をデフォルトと ▶ 最近のワークスペース(R 	として使用し、今後この質問を表示しない(<u>U)</u> 3)		
	-	起動(L)	キャンセル

© Rocket Software, Inc. or its affiliates 1990–2024. All rights reserved. Rocket and the Rocket Software logos are registered trademarks of Rocket Software, Inc. Other product and service names might be trademarks of Rocket Software or its affiliates.



SJIS 資産を利用する場合、文字コードの指定を明確に行う必要があります。[Window(W)] > [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「デフォルト(windows-31j)」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 Capabilities Schema Associa	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。
> Security	□ ネイティブのフックまたはボーリングを使用して更新(R)	
UI フリーズ・モニタ	✓ アクセス時に更新(S)	
> User Storage Se	□ 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)	
Web ブラウザ	ワークスペース保管間隔 (分)(W): 5	
> エディタ		
+-	ウィンドウのタイトル	
クイック検索	マロークスパーフタを表示(F): workspaceivmws	
グローバル化		
コンテンツ・タイプ		
サービス・ポリシー	□ ワークスペースのフルハスを表示(F): C:¥workspaceJvmws	
トレース	ノロダクト名を表示	
> ネットワーク接続		
ハンドラーをリンク	プロジェクトを閉く際に 参照するプロジェクトを閉く プロソプト シ	
バースベクテイノ	クロクエアトを開く旅に、 参照するクロクエアトを開く、 クロクアト	
フロジェクト・ネーナ	不明なプロジェクトの性質を以下のように報告(A): 警告 ~	
	Report missing project encoding as: 警告 ~	
> 開始わよびンヤツ		
2 2 下航 检索		
通知	システム・エクスブローラーを起動するコマンド(X): explorer /E,/select=\${selected_resource_lo	c}
比較/パッチ		
> Ant	テナフト・ファイル・エンフード(T) 新規テキスト・ファイルの行区切り文字((F)
AspectJ Compiler	● デフォルト(U) (windows-31i) ● デフォルト(E) (Windows)	
> CSS (Wild Web De	$\bigcirc \mathcal{Z} \oplus (\mathbb{A})$; windows-31 \checkmark	
> Gradle		
> HTML (Wild Web I 🗡	デフォルトの復元(T)	適用(1)
< >	77/10/01/20(1)	x=ris(=)
? 눱 👍 🖗	適用して閉じる	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	Ί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)		ļ	Alt+シフト	≻+N>]	₹.	COBOLプロ	コジェクト		
	ファイ	ルを開く(.).					2	COBOL I	ニーファイル プロ	リジェクト	
\sim	ファイ	ル・システム	からプロジェクトを	開<			e	リモ−ト COE	BOL 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				>	۲ ۲	リモ−ト COE	30L 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: 64 ビット

COBOL プロジェクト ワ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。	Ð
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	
 ※ Micro Focus テンプレート [32 ビット] ※ Micro Focus テンプレート [64 ビット] 	
<u>קרעד</u>	<u>トの設定を構成</u>
□ テンプレートの参照	
場所:	参照
ファイルシステムを選択: default \vee	
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>): <mark>デフォルト</mark> ~	
? 終7(E)	キャンセル

6) 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、[インポート(I)] > [インポ

ート(I)] を選択します。





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

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。	Ľ
インポート・ウィザードの選択(<u>S</u>):	
7ብሥንአታ	
 ◇ ◇ 一般 ① アーカイブ・ファイル ② ファイル・システム ③ フォルダーまたはアーカイブ由来のブロジェクト ◇ 読 既存プロジェクトをワークスペースへ ③ 設定 > ◇ EB > ◇ Git > ◇ Git > ◇ Gradle > ◇ J2EE > ◇ Micro Focus > ◇ Micro Focus インターフェイスマッパー > ◇ Oomph < △ TevtMate 	
(ア) (ア) 終了(F)	キャンセル

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・システムからリソースをインポートします。		
次のディレクトリーから(<u>Y</u>): C:¥jvmRESTfulWSTutorial¥COBOL COBOL	 ✓ BOOK.cbl ✓ BOOK.INFO.cpy ✓ BOOKSCRN.cbl 	参照(<u>B</u>)
タイプをフィルター(①) すべて選択(S) 選択をすべて解除(D) インポート先フォルダ(L): NativeCOBOL		参照(<u>W</u>)
オブション □ 答告を出さずに既存リソースを上書き(<u>O</u>) □ トップ・レベルのフォルダーを作成(<u>C</u>) 拡張 >>(<u>A</u>)		
?	< 戻る(<u>B</u>) 次へ(<u>N</u>) > 終了(<u>F</u>)	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。

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

© Rocket Software, Inc. or its affiliates 1990–2024. All rights reserved. Rocket and the Rocket Software logos are registered trademarks of Rocket Software, Inc. Other product and service names might be trademarks of Rocket Software or its affiliates.



 [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 ✓ Micro Focus	New Configuration [使用中]	~ 構成の管理
ビルダー ビルド パス 〜 ビルド構成	フィルタテキストを入力 匡	
> COBOL イベント	設定	値
ディプロイ ビルド環境	◆ Linkage 出力の名前	NativeCOBOL
> リンク > 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 ビット数	キー 天口 318277 170 64 ビット
WikiText	. <i>LBR にパッケージ化</i> サービスを COBOL アーカイブ (.car) フ	レルズ ロークション レークション レーン レーン レーン レーン レーン レーン レーン レーン レーン レー
9-7 タスク・タグ > タスク・リポジトリー ビルダー	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を	▲
プロジェクト・ネーチャー プロジェクト・ファセット		· ·
プロジェクト参照 > 検証 実行/デバッグ設定		デフォルトの復元(1) 適用(1)
?		適用して閉じる キャンセル

11) [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)"
```

BX.AC	値
< −般	
文字セット	ASCI
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)
追加指令 コンパイラに渡す追加のCOBOLコンパイラ指令です COBOL コンパイル設定: CHARSET"ASCII" SOURCE-ENCODING"ANSI" DIALE	CT"MF" SOURCEFORMAT" fixed" NOLIST anim
EXITPROGRAM"ANSI" NOTESTCOVER NOPROFILE EXTERNAL)	WARNING"1" MAX-ERROR"100" ASSIGN



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)
R	実行点をリセット	
Q	実行(R)	Ctrl+F11
核	デバッグ(D)	F11
	実行履歴(T)	>
0	実行(S)	>
	実行構成(N)	

2) [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

1 🖻 🕫 🔛 🗶 🖪 🏹 👻			
フィルタ入力			
🗄 Apache Tomcat	^		
🖾 AspectJ/Java Application			
📼 AspectJ Load-Time Weaving Application			
國 COBOL/Java 相互運用機能のアプリケーション			
🚾 COBOL JVM アプリケーション			
🚾 COBOL JVM ユニット テスト			
🚾 COBOL JVM リモート アプリケーション			
■ COBOL アプリケーション	1		
🔄 COBOL ユニット テスト			

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	🗟 — A	🖞 🦭 ソース 🚾 環境 🔲	共通(C) 🔎	実行時	🏷 デバッグシンボル
Apache Tomcat		OL プロジェクト(P)			
Aspect/Java Application	Na	ativeCOBOL			参照

4) [環境]タブを選択し、[追加(A)] をクリックします。

名前(N): NativeCOBOL								
🗔 一般 🧤 ソース 🌄 環境 🔲 共通(C) 🏓 実行時 🍢 デバッ	🗔 一般 💱 ソース 🐻 環境 🔲 共通(C) 🔊 実行時 💱 デバッグシンボル 🧕 動的分析 🔮 CTF 🥒 ユンテナー							
(注:ここで定義された変数は、任意の現在の設定値、または任意の指 環境スクリプト内の設定値を上書きします。)	定された							
変数	値	追加(A)						
		編集(E)						

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"



環境	変数を追加または変更します		0
変数:	BOOKINFO		
値:	C:¥jvmRESTfulWSTutorial¥DAT¥BOOKINFO.DAT	 	
?		ок	キャンセル

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。

名前(N): NativeCOBOL	9 実行時 15g デバッグシンボル 4 動的分析 22 CTF 2 コンテナ− 2 値、または任意の指定された
変数 BOOKINFO	值 C:¥jvmRESTfulWSTutorial¥DAT¥BOOKINFO.DAT 編集(E) 削除(R)
ま行する環境スクリプト: 場所: ファイルがプロジェクト内にある場合、 パラメータ: □ 関連付けられたプロジェクトのピルド環境から値:	絶対/(スは相対/(スになります。 乾継承
	前回保管した状態に戻す(V) 適用(Y) 実行(R) 閉じる

以下のような画面が表示されます。

C:\.	NativeCOBOL					-		×
	FUNCTION: [_] READ=1, ADD=2, DELE	ETE=3	NEXT=4,	GROSSSAL	ES=S,	QUIT	=9	
	ストック留号: L 」 タイトル: [ジャンル: 「	٦]		
	著者: [小売価格: [0] 仕入れ: [0]]			
	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	-		\times
	FUNCTION: [1] READ=1, ADD=2, DELETE=3 NEXT=4, GROSSSALES=S,	QUIT:	=9	
	ストック番号: [1111]			
	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]
```

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"

STATUS: [00]



0:5.	NativeCOBOL	-	
	FUNCTION: [2] READ=1. ADD=2. DELETE=3 NEXT=4. GROSSSALES=S.	QUIT=	9
	ストック番号: [4444]		Ŭ
	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: [000004000]		
	STATUS: [00]		

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。

C:4.	NativeCOBOL	-	
	FUNCTION: [S] 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: [017000000]		
	STATUS: [00]		

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。

1) 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	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	エクト	•(P)	実行(R)	ウィンドウ(W)	ヘルプ(H)
	新規(N)		Alt	t+シフト	+N >		COE	BOL プロジ	ェクト	
	ファイノ	レを開く(.)					2	COE	BOL JR-1	ファイル プロジェク	フト
0	ファイノ	レ・システム	からプロジェクトを	開く			Ê	IJ£-	- ト СОВО	L プロジェクト	
	Recei	nt Files				>	Ŷ	IJŦ	- ト COBO	Lコピーファイル:	プロジェクト
							曾	COE	3OL/Java	相互運用機能	のプロジェクト
	閉じる	6(C)			Ctrl	+W	儒	COE	3OL ユニッ	ト テスト プロジェ	クト
	すべて	閉じる(L)		Ctrl	+シフト	+W	1	COE	BOL JVM (プロジェクト	
g	保存(S)			Ctr	1+5	R	IJŦ-	- ト COBO	L ユニット テスト	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。

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
プロジェクトዲ(P): 図 デフォルト・ロケーションの使用(D) ロケーション(L): ○¥workspace-jvmws¥BookLibrary プロジェクト テンプレートを選択 び Micro Focus デンプレート		Ē(R)
「 テンプレートの参照 場所: ファイルシステムを選択: default 〜 IPE	<u>テンプレートの時</u> 参覧	<u>定を構成</u>
 ● 東行環境 JRE の使用(V): ○ プロジェクト固有の JRE を使用(S): ○ Use default JRE 'AdoptOpenJDK' and workspace compiler pr 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) 	JavaSE-17 AdoptOpenJDK eferences JRE 新規(1	 ✓ <u>を構成</u> W)
ワーキング・セット(0): ②	<	E)

3) 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。

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

4) [Micro Focus] > [ビルド構成] を選択し、以下の入力をしたうえで、[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) ILSMARTLINKAGE ILNAMESPACE(com.microfocus.sample)"

フィルタ入力	ビルド構成					
> リソース						
Coverage						
✓ Micro Focus JVM ビルド パス	フィルタテキストを入力					
> SQL プリプロセッサ	設定	値				
コピーファイル	▼ 一般					
ビルダー	文字セット	ASCIL				
ビルド理情	ソース エンコーディング	ANSI				
ビルド構成	COBOL 方言	Micro Focus				
ビルド優先順位	ソース フォーマット	可変				



	適用して閉じる キャンセル
	デフォルトの復元(T) 適用(L)
CHARSET"ASCII" SOURCE-ENCODING"UTF8 packageToFolderMapping jarFolder=dist Bo (EXTERNAL) ILSMARTLINKAGE ILNAMESPAC	" DIALECT"MF" SOURCEFORMAT"variable" NOLIST anim EXITPROGRAM"ANSI" okLibraryjar outputDirectory=bin NOWARNING MAX-ERROR"100" ASSIGN 2E(com.microfocus.sample)
COBOL コンパイル設定:	
ソース エンコーディング SOURCE-ENCODING はソース プログラムのエンコ	-ディングをコンパイラに渡します。その後、RUNTIME-ENCODING 指令が指定されていな v
追加指令	ASSIGN(EXTERNAL) ILSMARTLINKAGE ILNAMESPACE(com.microfocus.sample)
✔ 追加指令	
最大エラー数	100
警告レベル	重大なエラーだけ(レベル S)
▼ Iラ-/警告	

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上で、 [新規作成(N)] > [COBOL JVM パッケージ]を選択します。

² ₈ C × ¹ ₆ J ² / ₈ A ² ₂ □							
 ✓ E SookLibrary 							
> 🛋 COBOL J	VM 9	実行時システム					
> 🛋 JRE システ		新規作成(N)	>	않	COBOL JVM プロジェクト		
> <mark>巻 src</mark> 私 参照うイン > 1号 NativeCOBC		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 COBOL コピーファイル プロジ COBOL プロジェクト 	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 ユニット テスト プロジェクト COBOL/Java 相互運用機能のプロジェクト		
	1 1 2	貼り付け(P) 削除(D) コンテキストから除去	Ctrl+V 削除 Ctrl+Alt+シフト+下	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	リモート COBOL JVM プロジェクト リモート COBOL コピーファイル プロジェクト リモート COBOL プロジェクト		
<		リファクタリング(T)	>	R	リモート COBOL ユニット テスト プロジェクト		
🏽 ア 🗙 📰 プロ		タスクのスキャン			プロジェクト(R)…		
アウトラインを提供す	4	インポート(i) エクスポート(O)…	>	ii ii	COBOL コピーファイル COBOL コログラム		
はありません。	\$	更新(F) F5 プロジェクトを閉じる(S)		Ľ	スタンドアロン ファイル リモート スタンドアロン ファイル		
		無関係なプロジェクトを閉じる(U)		0	COBOL JVM Delegate		
		Source		C)	COBOL JVM Enum COBOL JVM Value Type		
	 Coverage As 実行(R) デバッグ(D) 		>	© ©	COBOL JVM インターフェイス COBOL JVM クラス COBOL IVM ソースフォルダ		
	8	プロファイル(P)	>	₿	COBOL JVM パッケージ		

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。		
ソース フォルダ(<u>D</u>):	BookLibrary/src		参照(<u>o</u>)
名前(<u>M</u>):	com.microfocus.sample		
?		終了(E)	キャンセル

src フォルダー配下に、空のパッケージが作成されます。



7) 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。



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





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

9) 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 フォルダー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ー配下の bin¥com¥sample フォルダーに、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。

© Rocket Software, Inc. or its affiliates 1990–2024. All rights reserved. Rocket and the Rocket Software logos are registered trademarks of Rocket Software, Inc. Other product and service names might be trademarks of Rocket Software or its affiliates.



プロ	ジェクト(<u>P)</u> 実行(<u>R</u>) ウィンドウ	(<u>W) ヘルプ(H</u>)
P 	プロジェクトを開く(E) プロジェクトを閉じる(S)	
_ 10	ー すべてビルド(A) プロジェクトのビルド(B) ワーキング・セットのビルド(WA)	Ctrl+B
~	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>フィルタ入力</u>			×		
 > ゆ Mylyn > つ Oomph > Version Control (Tex) > Version Control (Tex) > Web サービス > ひ Web サービス > ひ サーバー ※ サーバー ※ サーバー > クーミナル > ゆ デーダ管理 > プラグイン開発 > クーリエー > リモート・システム > ご その他 	am)				
開<	0)	キャンセ	L		
本手順では 以下のよう(Tomca	at 10.1	のサ・	-バー構成を行っていま	す。
🖹 マーカー 🔲 プロパティ	イー 棉サ	- <i>I</i> (- ×		データ・ソース・エクスプロ・	
🖥 localhost Ø Tor	ncat v10	.1 Serve	r [停	止, 再公開]	



3.3.1 Java Web アプリケーションプロジェクトの作成

1) [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。

新規ワインドウ(N)	P	🔲 🕺 🔁 📭 🔄 😎
エディター	>	Q i 🔂 🛛 🛍 🔞 🖓 🔤 🖓
外観	>	🖓 🗗 📴 アウトライン × 💿 🕴 🖓 🗖
ビューの表示(V)	>	アウトラインを提供するアクティブなエ
パースペクティブ(R)	> 🖪	パースペクティブを開く(O) > 🌇 Aspect Visualization
ナビゲーション(G)	>	パースペクティブのカスタマイズ(Z)
Web ブラウザ 設定(P)	>	名前を付けてパースペクティブを保存(A) W Web パースペクティブのリセット(R) W Web パースペクティブのリセット(R) や デバッグ パースペクティブを閉じる(C) ゆ リソース

2) 「J2EE」を選択して、[開く(o)] をクリックします。

^
~
_

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

ファイ	イル(F) 編集(E) ナビゲート(N) 検索	プロジェクト(P) 実	行(R	R)	ウィンドウ(W) ヘルプ(H)	
	新規(N)	Alt+シフト+N	۷ > ۱	ß	Maven Project	
	ファイルを開く(.)			Ċ		
۵.	ファイル・システムからプロジェクトを開く			3	動的 Web ブロジェクト	
	Recent Files		>	€ŭ .~~>	EJB ノロシエクト	
	閉じる(C)	Ctrl+W	V	₩₩ 	コインツー・ノロンエント アプリケーション・クライアント・プロジェクト	
	すべて閉じる(L)	Ctrl+シフト+W	V	<u>.</u>	静的 Web プロジェクト	
	保存(S)	Ctrl+S	s -	æ	JPA プロジェクト	
	名前を付けて保存(A)			Ľ	プロジェクト(R)…	
	すべて保存(E)	Ctrl+シフト+S	S	S	CSS ファイル	
	前回保存した状態に戻す(T)			翁 JavaScript ファイル		
	移動(V)			6	サーブレット	
	名前を変更(M)	F2	2		Session Bean (EJB 3.x/4.x)	
66	更新(F)	F5	5	⊵ ⊛ _≁	Message-Driven Bean (EJB 3.x/4.x)	
	行区切り文字の変換(D)		>	22" 59		
۵	印刷(P)	Ctrl+F	P	x¢	XMLファイル	
2	インポート(I)			Ċ	フォルダー	
4	エクスポート(O)…			Ľ	ファイル	
	プロパティ(R)	Alt+Ente	r		JSP ファイル Aspect	
	ワークスペースの切り替え(W)		>	-		
	再開				リンフル(X)	
	終了(X)			P	その他(o)	Ctrl+N

© Rocket Software, Inc. or its affiliates 1990–2024. All rights reserved. Rocket and the Rocket Software logos are registered trademarks of Rocket Software, Inc. Other product and service names might be trademarks of Rocket Software or its affiliates.



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



© Rocket Software, Inc. or its affiliates 1990–2024. All rights reserved. Rocket and the Rocket Software logos are registered trademarks of Rocket Software, Inc. Other product and service names might be trademarks of Rocket Software or its affiliates.



j: "war"	
i project	
oject	
com.microfocus.sample	
JVMRESTfulWS	_
0.0.1-SNAPSHOT V	
war v	
ect	
	_
Browse Cle	ar
	<pre> 3. "War" a project oject com.microfocus.sample JVMRESTfulWS 0.0.1-SNAPSHOT war cet cet cet cet cet cet cet cet cet cet</pre>

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 <u>ム</u> (E)	>

 アロジェクト・ファセット]を選択し、「Java」に"17"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。

フィルタ入力	プロジェクト・ファセット		← ▼ ⇒ ₹ §
> Java Iディタ ^	構成(C): <カスタム>		✓ 別名保存(S) 削除(D)
> Java コンパイラー	プロジェクト・ファセット	バージョン	詳細(I) ランタイム(<u>R</u>)
Java のビルド・バス JSP フラグメント	> □ ■ Axis2 Web サービス □ ■ CXF 2.x Web サービス	1.0	🛽 Java 17
> Maven Web コンテンツの設5	□ □ EAR □ □ ▲ EJB モジュール	6.0 - 3.1 -	Adds support for writing applications using Java programming language.
Web ブロジェクトの影 Web ページ・エディタ	EJBDoclet (XDoclet)	1.2.3 • 17 •	
Wiki lext > XDoclet	JavaScript JavaServer Faces D IAV BC (BEET Much # 167)	1.0 2.3 -	
サーバーサードス・ポリシー	→ JAXB	1.1 + 2.2 +	
ターゲット・ランタイム タスク・タグ	□ ■ JCA ビジュール □ ↔ JPA 開発	2.2 -	
> タスク・リポジトリー ビルダー	$\square \blacksquare WebDoclet (XDoclet)$ $\square \blacksquare PTU/T-V=Y-7-TV-F=TV-TV-TV-TV-TV-TV-TV-TV-TV-TV-TV-TV-TV-T$	1.2.3 • 6.0 •	
プロジェクト・ネーチャ- プロジェクト・ファセット	□ □ ユーティリティー・モジュール □ □ ユーティリティー・モジュール □ □ □ ユーティリティー・モジュール	2.5	
ノロンエクト参照 > 検証	□ = 2010 the 2011 // □ ■ 静的 Web モジュール		
実行/デバッグ設定 く >			回保存した状態に戻適用(上)
?			適用して閉じる キャンセル

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。

フィルタ入力	Java のビルド・パス	← ← ⇒ 8
> Javaエディタ へ > Javaコード・スタイル > Javaコンパイラー	ピックス(S) G プロジェクト(P) あ ライブラリー(L) や 順序およびエクスポート(Q) G モジュール依存関係(M) ビルド・パス上の JAR およびクラス・フォルダー(D):	
Java のビルド・パス	 ◆ モジュールパス → 105 2172 / = / (100 05 17) 	JAR の追加(<u>」</u>)
> Maven	○ No. RE 2774 A· 71 J 7 J − [JavaSE-17] ◆ ゆ クラスパス	外部 JAR の追加(<u>X</u>)
Web コンテンツの設5	> Maven Dependencies	変数の追加(⊻)
Web ノロシェクトの影 Web ページ・エディタ		ライブラリーを追加(<u>i</u>)…
WikiText		クラス・フォルダの追加(<u>C</u>)…
> XDoclet > クライアント・サイド Ja		外部クラス・フォルダーを追加(D)

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。



 ライブラリー・タイブを選択します。

 追加す るライブラリー・タイブを選択します。

 COBOL VVM 実行時システム

 CAF ランダイム EAR ライブラリー JUnit Maven Managed Dependencies Web App ライブラリー サーバー・ランダイム ブラグインの依存関係 コーザー・ライブラリー 接続可能性ドライバー定義

 ②
 <戻る(B)</td>
 次へ(N) >
 終了(F)
 キャンセル

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。

Java のビルド・パス	← ➡ ⇒ §
(き プロジェクト(P) 🛋 ライブラリー(L) 🌭 順序およびエクスポート(Q) O 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>):	
♣ モジュールパス ♣ クラフパフ	追加(<u>D</u>)
	編集(<u>E</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				
±//7	387 HB (C)	₩旧た	オバア級限	÷(D)
970	通り((3)	370 22	9 个 し府中将	F(U)
	ОК		キャンセノ	١

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・パス	← → ⇒ %
😕 ソース(S) 🗁 プロジェクト(P) 🛋 ライブラリー(L) 🍫 順序およびエクスポート(O) 🟮 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>):	
 ♣ モジュールパス ▲ クラスパス 	追加(<u>D</u>)
> 🔁 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。

ጋィルタ入力	Web 展開アセンブリ		← ▼ ⇒ §
> UV-X ^	Define packaging structure for this Java EE W	eb Application project.	
Deployment Assembly	א-ע	デプロイ・パス	追加(<u>D</u>)
Javadoc ロフージョン	🗀 /src/main/java	WEB-INF/classes	
> Javaエディタ	/src/main/resources	WEB-INF/classes	施兵(上)
> Java コード・スタイル	/src/main/webapp	□ /	除去(图)
> Java コンパイラー	/target/m2e-wtp/web-resources	□ /	
Java のビルド・パス	Maven Dependencies	C WEB-INF/lib	
JSP フラグメント			

^{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



ディレクティブの アセンブリーディレクラ	種類を選択 すブを新規追加します	o		
■ Java ビルド つ ファイル・シン つ フォルダー テ ブロジェクト	・パス・エントリー ステム由来のアーカイブ	r		
□ ワークスペー □ 変数	ス由来のアーカイブ			
?	< 戻る(B)	次へ(N) >	終了(F)	キャンセル

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

Java ビルド・パス・エントリー Select build path entries to include in the deployment assembly.	a
> 🛋 COBOL JVM 実行時システム	
(?) <戻る(B)	キャンセル

14) さきほど同様、[追加(D)] をクリックします。

pplication project.	
デプロイ・パス	追加(<u>D</u>)
WEB-INF/classes	信告(の)
WEB-INF/classes	福朱(上)
<u> </u>	除去(R)
<u> </u>	
C WEB-INF/lib	
🗀 WEB-INF/lib	
	Application project. デブロイ・パス WEB-INF/classes WEB-INF/classes WEB-INF/classes イ WEB-INF/lib WEB-INF/lib 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。



ディレクティブの アセンブリーディレクティ	重類を選択 げを新規追加しま	ġ.		
■ Java ビルド・ つ ファイル・シス つ フォルダー デ プロジェクト つ ワークスハーク ○ 変数	パス・エントリー テム由来のアーカイ (由来のアーカイブ	J		
(?)	< 戻る(B)	次へ(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 inc	lude in the deplo	yment assembly.		
🔁 BookLibrary				
?	< 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。



ine packaging structure for this Java EE Web	Application project.	
-7 ^	รัプロイ・パス	追加(D)
🗀 /src/main/java	WEB-INF/classes	177 100 100
/src/main/resources	WEB-INF/classes	編集(E)
/src/main/webapp		除去(R)
/src/test/java	WEB-INF/classes	10/2011/14
/src/test/resources	WEB-INF/classes	
/target/m2e-wtp/web-resources		
BookLibrary	WEB-INF/lib/BookLibrary.jar	
■ COBOL JVM 実行時システム	🗀 WEB-INF/lib	
🛋 Maven Dependencies	🗀 WEB-INF/lib	
	前回保管した状態に戻す(V)	週用(L)

 C:¥jvmRESTfulWSTutorial¥Java¥src 配下の com フォルダー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 [Java Resources] > [src/main/java] 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て、[貼り付け(P)] を選択します。

<u></u> ¹ プロジェクト・エクスプロ × □ 🕏 🎖		新規(N) 次へジャンプ(I)	>
BookLibrary Definition		型階層を開く(N) 表示方法(W)	F4 Alt+シフト+W>
 ・ B デプロイメント記述子: ・ ・		コピー(C) 修飾名のコピー(Y)	Ctrl+C
→ Z JAX-WS Web J-L	ĥ	貼り付け(P)	Ctrl+V
 B src/main/java B src/main/resource 	×	削除(D) コンテキストから除去 ビルド・パス(B)	削除 Ctrl+Alt+シフト+下 >
> src/test/java > src/test/resource > Libraries		ソース(S) リファクタリング(T)	Alt+シフト+S > Alt+シフト+T >

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 フォルダー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、[貼り付け(P)] を選択します。



눱 プロジェクト・エクスプロ	×	
E \$	7	6
> 📂 BookLibrary		
> 🗁 InternalTDProject		新規(N)
✓ ₩ JVMRESTfulWS		次ヘジャンプ(I)
> 눱 デプロイメント記述		表示方法(W)
> 🧟 JAX-WS Web サ-		718 (0)
🗸 蹄 Java Resources		」と-(C)
> 🧬 src/main/java		1②肺治(/) ドー(Y)
> 🕭 src/main/resc		<u>賄リ110(P)</u> 問U(全(D))
> 🏼 src/test/java	<u>></u>	月川(ホ(D) コンテキフトから除土
> 쁠 src/test/reso		コノノ エストからは云 ビルド・パフ(P)
> 🛋 Libraries		L///*//A(D) 投動()/)
> 🗟 Deployed Resou		(♥動(♥)
🗸 🖶 src		名刖を変史(IVI)
🗸 🔂 main		タスクのスキャン
> 🔂 java		インポート(i)
🗁 resources	4	エクスポート(O)
> 🗁 webapp	5	
> 🗁 test	\$ <u>`</u>	史析(F)

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



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。

>	0 0 0 0 0	リモートシステムビューで表示 Coverage As 実行(R) デパッグ(D)	> > >	¹ ・ソース・エクスプロー 哈スニペット <i>男</i> ターミナル 旦 コン ¥ 嶺 風 配 ₪
	01	Maven	>	Add Dependency
/	\checkmark	検証		Add Plugin
nom xml - IVMRE	ml - IVMRE チーム(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。

© Rocket Software, Inc. or its affiliates 1990–2024. All rights reserved. Rocket and the Rocket Software logos are registered trademarks of Rocket Software, Inc. Other product and service names might be trademarks of Rocket Software or its affiliates.



Available iviaven Couebases	
VI 🚰 JVMRESTfulWS	Select All
	Add out-of-date
	Deselect All
	Expand All
	Collapse All

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



21) [起動構成を開く] リンクをクリックします。

🛛 概要

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:	localhost				
ホスト名:	localhost				
<u> דעאלב סיק ד</u>	Apache Tomcat v10.1				
<u>構成パス:</u>	/サーバー/localhost の Tomcat v10.1 Server-config				
起動構成を	開く				

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



- 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"

名前(<u>N</u>):	BOOKINFO	
値(<u>V</u>):	STfulWSTutorial/DAT/BOOKINFO.DAT	変数(<u>B</u>)
	,	
	ОК	キャンセル

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。

定する環境変数(S)		
変数	値	追加(A)
OOKINFO	C:/jvmRESTfulWSTutorial/DAT/BOOKINFO.DAT	
		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)ネイティブ環境を指定され)追加(A) た環境と置換(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。



1	🗄 localhost Ø Tomcat v10.1 Server ×					
6	Web モジュ-	-JL				
N 2	/eb モジュール のサーバーで構成∃	する Web 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	概要「モジュール」					
-						

🖹 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 🖉 ターミナル 📮 コンソール

JVMRESTfulWS を選択し、[OK] をクリックします。

モジュール(M):	JVMRESTfulWS
文書ベース(D):	JVMRESTfulWS
パス(A):	/JVMRESTfulWS
C	OK キャンセル



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。

ファイ	イル(F)	編集(E)	ナビゲート(N)	検索	プロジェクト(P)	実行(I
<u>,</u>	新規(ファイ) ファイ) Recer	N) レを開く(.) レ・システム: nt Files	からプロジェクト	を開く	Alt+シフ	≻+N>
	閉じる すべて	i(C) 閉じる(L)			Ctr Ctrl+シフト	l+W `+W
	保存(S)			Ct	rl+S

3.3.2 RESTful Web サービスの確認

 サーバービューより、Tomcat 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。

	~	則际 名前変更(N)	則际 F2
概要 モジュール	蓉	デバッグ(D)	Ctrl+Alt+D
図マーカー □ プロパティー 跳サーバー × № データ・ソース・T	0	起動(S)	Ctrl+Alt+R
→ La localhost の Tomcat v10.1 Server [停止, 再公開]		プロファイル(F)	Challs Albert
	1	公開	Ctrl+Alt+S

起動すると、ステータスが "始動済み" に変更されます。

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。

マーカー □プロパティー 端サーバー 純 データ・ソース・エクスプローラー ≧ スニペット 夢 ターミナル ■ コンソール × localhost の Tomcat v10.1 Server [Apache Tomcat] C:¥Program Files (x86)¥Micro Focus¥Visual COBOL¥Adopl 警告: The following warnings have been detected: WARNING: The (sub)resource method getBookGrossSalesAmount in com.microfocus.sam

8月 22, 2024 10:20:47 午前 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。 8月 22, 2024 10:20:47 午前 org.apache.catalina.startup.Catalina start 情報: サーバーの起動 [1334] ミル沙

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
ABC書店在原	車管	理シ	ィステ	4		
	甲下してくフ	ださい。		システ Home Pa 書籍検索	-ムメニュ ^{ge}	-
				在庫情報	登録	
				在庫情報	削除	
				売上集計	t	
				Contact		

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	システムメニュー
ストック番号を人刀の上、「検索	引ボタンを押下してくたさい。	Home Page
1111	横条	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 apiEndPoint http://localhost:8080/JVMRES Request (NO DATA) Response { "bookInfo": { "stockNo": "1111", "title": "LOAD OF THE RIN "author": "TOLKIEN "genere": "FANTASY "	TfulWS/bookmgnt/books/1111 G ", ",	

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 サーバー選択し、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。





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。

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。