# Micro Focus Visual COBOL チュートリアル

## JBoss アプリケーションサーバーと 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 では、他言語・他システムとの連携手法として、「Enterprise Server」を利用したサービス連携だけではなく、COBOL プログラムに一切の変更を行わずに、 Java 技術と連携可能な JVM COBOL 機能も提供しています。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も可能となります。

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 Server 2019 Standard Edition (64bit) |
| COBOL 開発環境製品 | Micro Focus Visual COBOL 7.0J for Eclipse（Patch Update2 適用）<br>JBoss AS, Wildfly & EAP Server Tools プラグイン導入済 |
| J2EE アプリケーションサーバー | JBoss EAP 7.3.8 |

RESTful Web サービスの実装やテストクライアントなどにおいて、Java、 JX-RS、 jQuery などの技術を利用したチュートリアルとなっておりますが、これらの技術に関する説明は本チュートリアルでは行っておりません。別途資料やオンライン文献などを参照ください。

- 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。

  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。

  サンプルプログラムのダウンロード

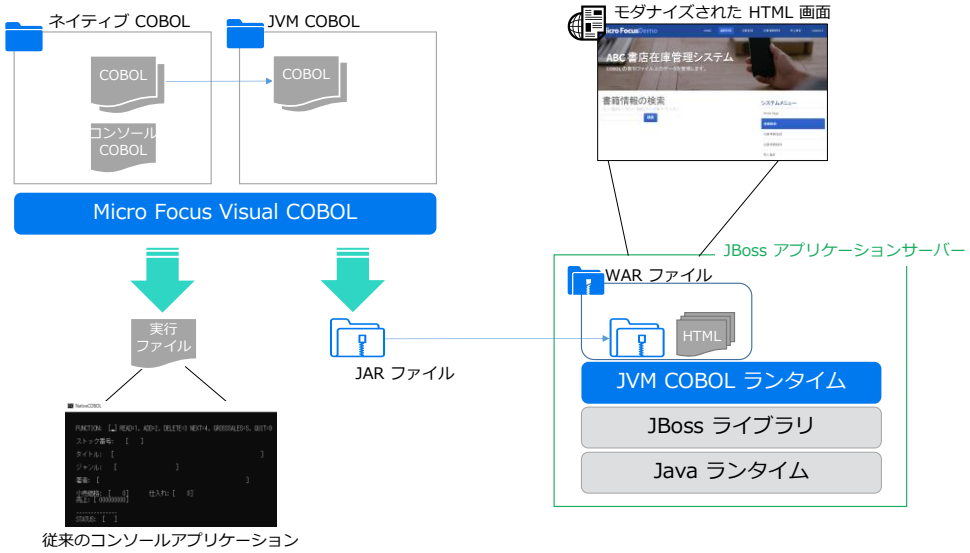
# 内容


1. 目的
2. 前提条件
3. チュートリアル手順の概要
   - 3.1. 今回作成するアプリケーション概要
   - 3.2. プロジェクトの作成と COBOL アプリケーションの確認
     - 3.2.1. プロジェクトの作成
     - 3.2.2. アプリケーションの確認
   - 3.3. JVM COBOL プロジェクトの作成
   - 3.4. JBoss EAP 上で動作する RESTful Web サービスの作成
     - 3.4.1. Web プロジェクトの作成
     - 3.4.2. JBoss EAP サーバー設定
     - 3.4.3. Java プログラムのインポート
     - 3.4.4. JBoss EAP サーバーを Eclipse 上から起動

# 3. チュートリアル手順の概要

## 3.1. 今回作成するアプリケーション概要

従来のコンソールアプリケーションである書籍情報を管理するアプリケーションを、JVM COBOL の機能を利用して COBOL プログラムを変更することなく Java クラスを生成、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 JBoss アプリケーションサーバーにディプロイし、実際の動作を確認します。

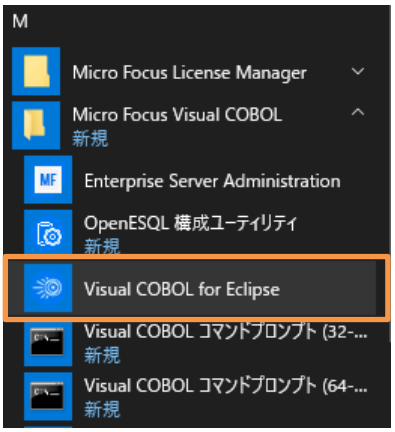



## 3.2. プロジェクトの作成と COBOL アプリケーションの確認

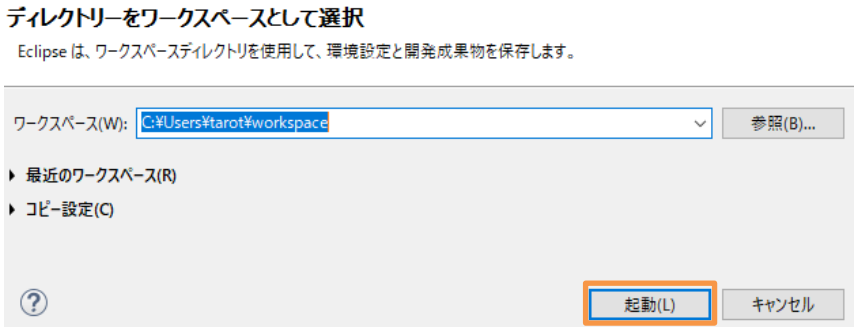
RESTful Web サービスと連携する前に、今の COBOL アプリケーションの状態を確認します。

### 3.2.1. プロジェクトの作成

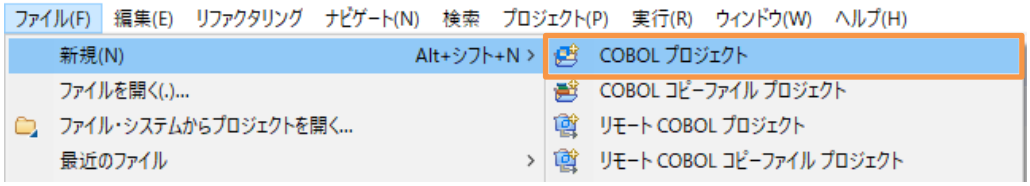
1) Visual COBOL for Eclipse を起動します。




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。

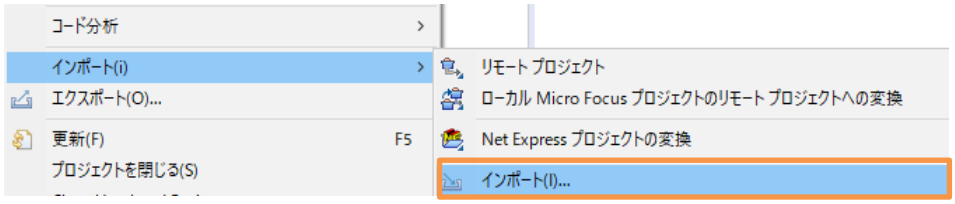



3) [ファイル(F)] > [新規(N)] > [COBOL プロジェクト] を選択します。

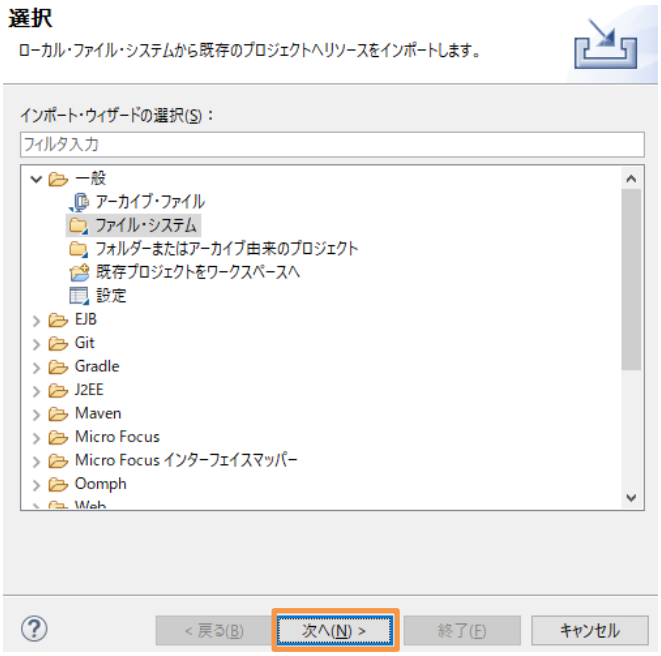



4) プロジェクト名に “NativeCOBOL” を入力し、[終了(F)] をクリックします。

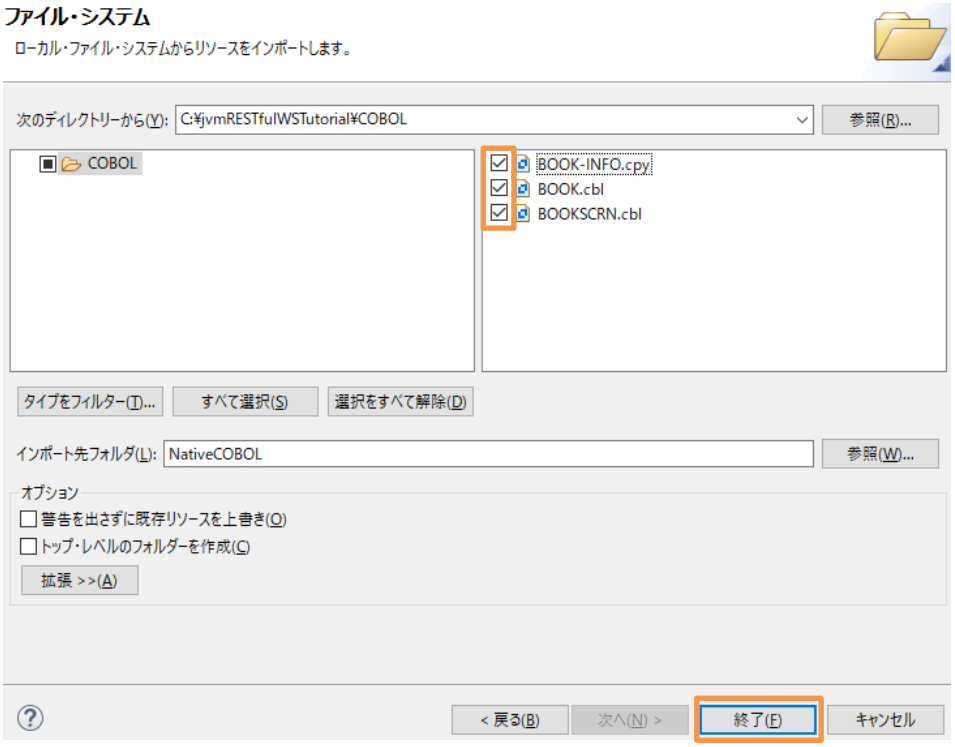

5) NativeCOBOL プロジェクトを選択し、マウスの右クリックにてコンテクストメニューを表示した上で、[インポート(I)] > [インポート(I)] を選択します。




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




7) チュートリアルファイル解凍フォルダ¥COBOL フォルダ配下の 3 ファイルを選択し、[終了(F)] をクリックします。

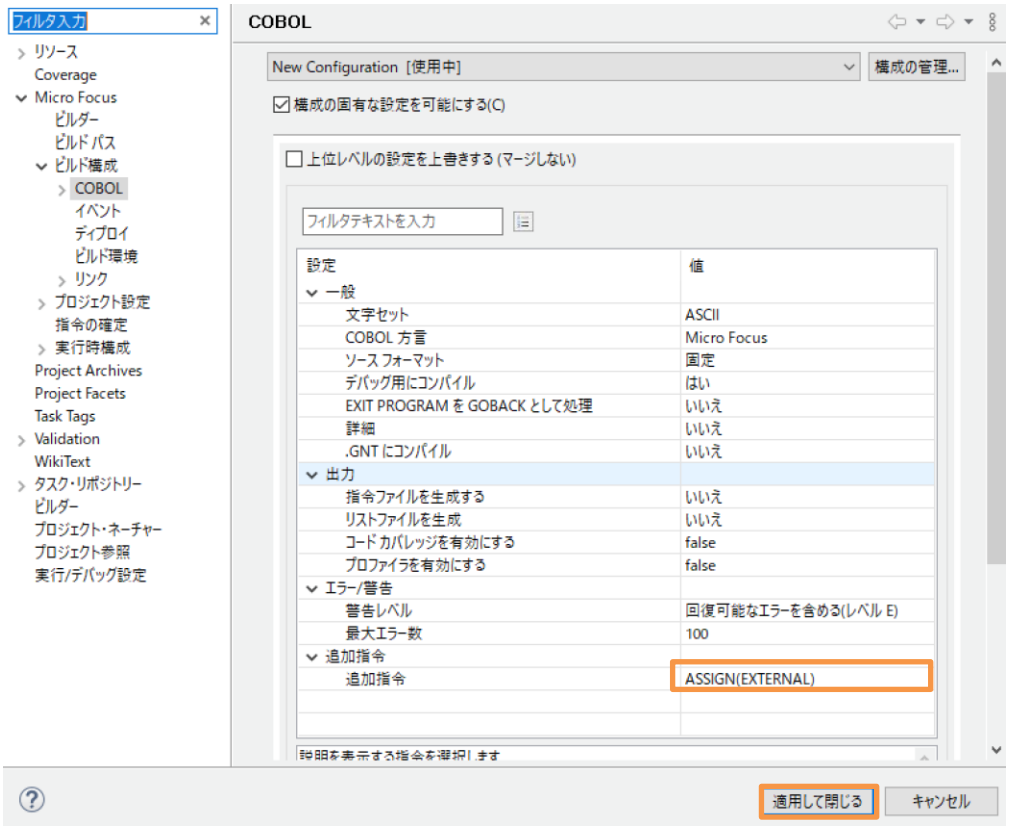



8) チュートリアルファイル解凍フォルダ 配下の DAT フォルダを任意の場所にコピーします。本チュートリアルでは、C ドライブ直下にコピーします。

9) NativeCOBOL プロジェクト名を選択し、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。

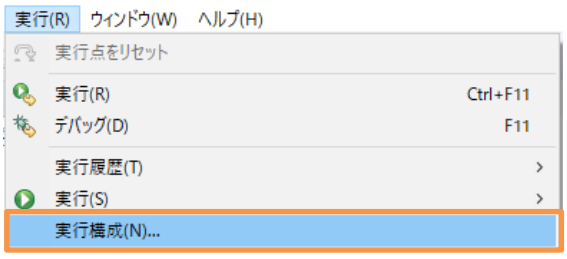

10) [Micro Focus] > [ビルド構成] > [リンク] 配下の「エントリポイント」に “BOOKSCRN” を入力します。

11) [Micro Focus] > [プロジェクト設定] > [COBOL] を選択し、「追加指令」に “ASSIGN(EXTERNAL)” を入力した上で、[適用して閉じる] ボタンをクリックします。

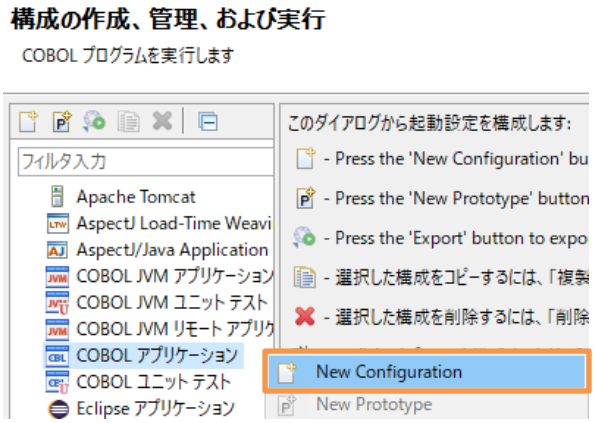



### 3.2.2. アプリケーションの確認

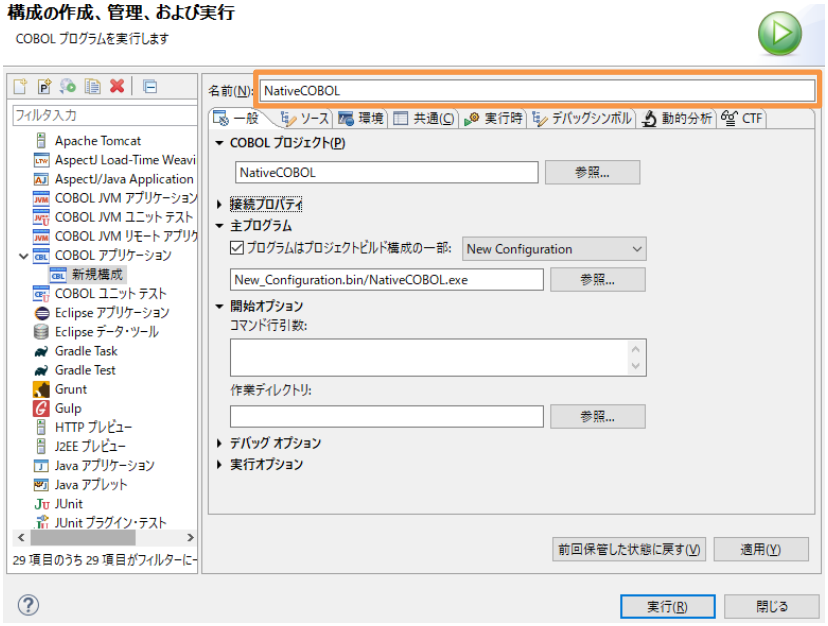
1) NativeCOBOL プロジェクト名を選択し、[実行(R)] > [実行構成(N)] を選択します。



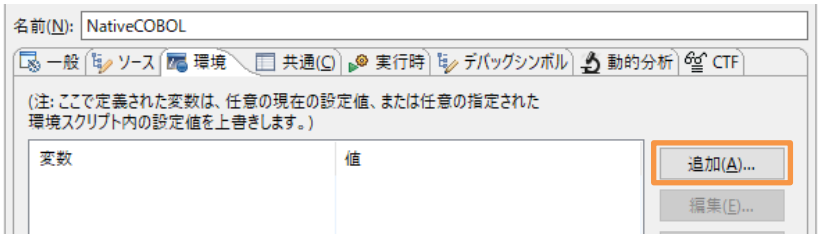

2) 「COBOL アプリケーション」を選択し、マウスの右クリックにてコンテクストメニューを表示した上で、[New Confugiration] を選択します。




3) 「名前」に “NativeCOBOL” を入力します。



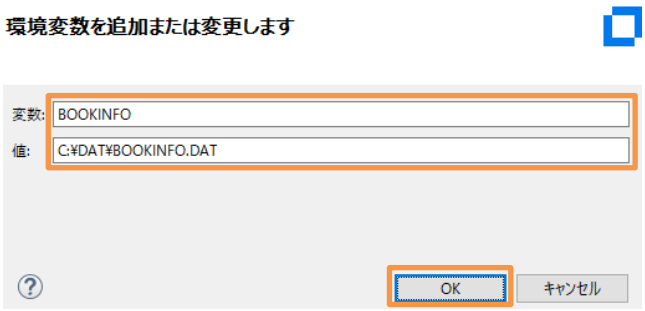

4) 「環境」タブを選択し、[追加(A)] をクリックします。



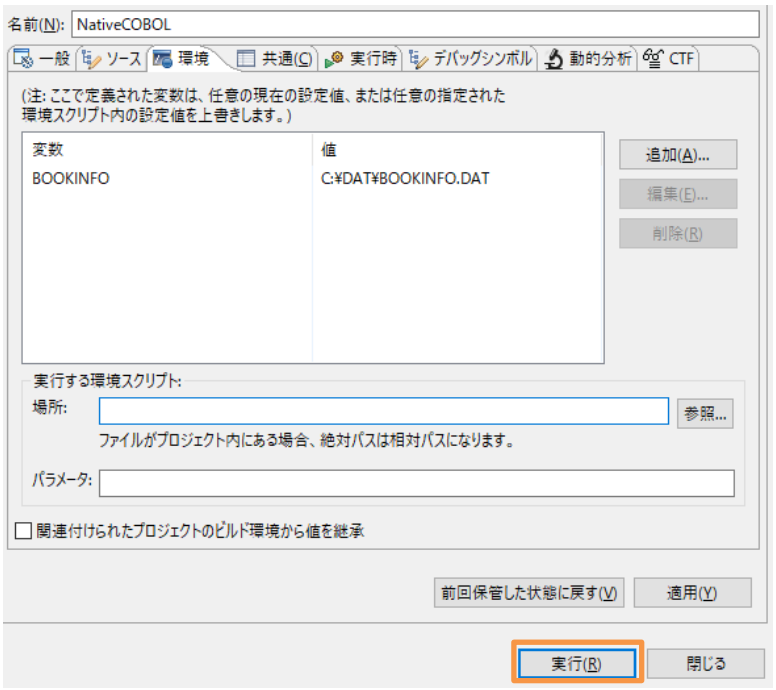

5) 以下の入力を行い、[OK] をクリックします。

変数： “BOOKINFO”

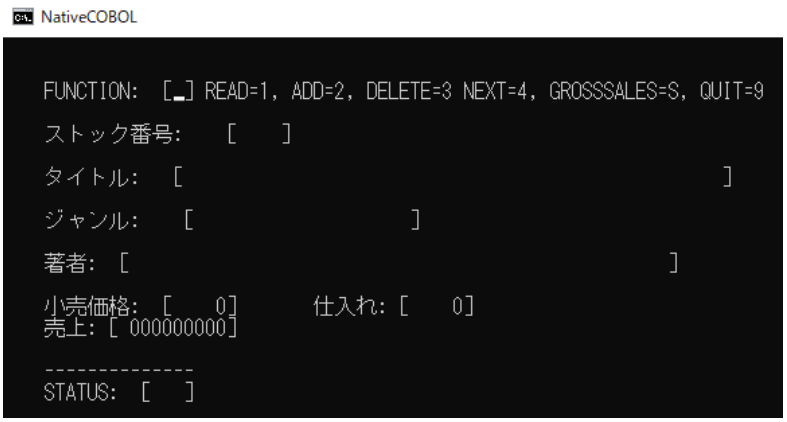
値： “C:¥DAT¥BOOKINFO.DAT”




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




以下のような画面が表示されます。



以下の入力を行った後、Enter キーを打鍵すると、書籍情報が表示されます。

FUNCTION: “1”

ストック番号： “1111”

```
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 の書籍は未登録のため、以下の情報を入力し、Enter キーを打鍵することで、書籍を追加します。

FUNCTION: “2”

ストック番号： “4444”

タイトル： “ブレイブストーリー”

ジャンル： “FANTASY”

著者： “宮部みゆき”

小売価格： “3000”

仕入れ： “1000”

売上： “900”

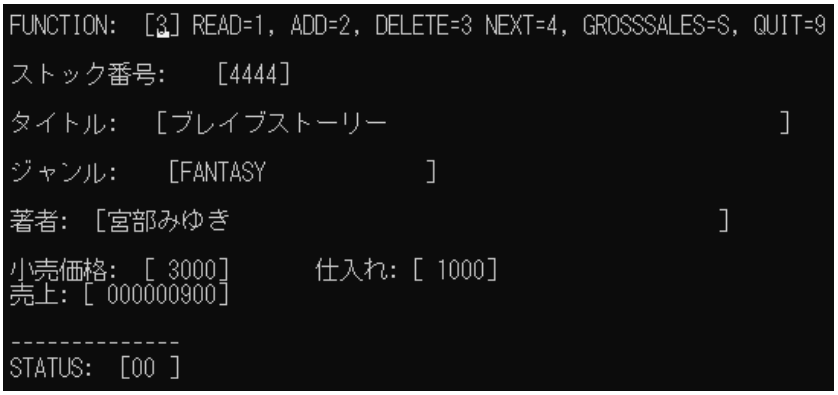
```
FUNCTION:  [2] READ=1, ADD=2, DELETE=3 NEXT=4, GROSSSALES=S, QUIT=9
ストック番号:    [4444]
タイトル:   [ブレイブストーリー                                  ]
ジャンル:    [FANTASY            ]
著者: [宮部みゆき                                        ]
小売価格:  [ 3000]       仕入れ: [ 1000]
売上: [ 000000900]

--------------
STATUS:  [00 ]
```

実行後、READ 機能を用いて、追加した情報が参照できていることを確認してください。

確認した後、以下の入力後、Enter キーを打鍵することで、情報を削除します。

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。

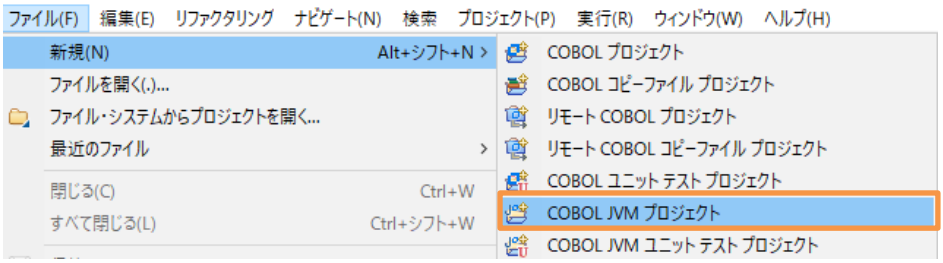
FUNCTION=S の GROSSSALES 機能は登録された全書籍情報の売上額を集計します。


確認後、FUNCTION=9 でアプリケーションを終了してください。

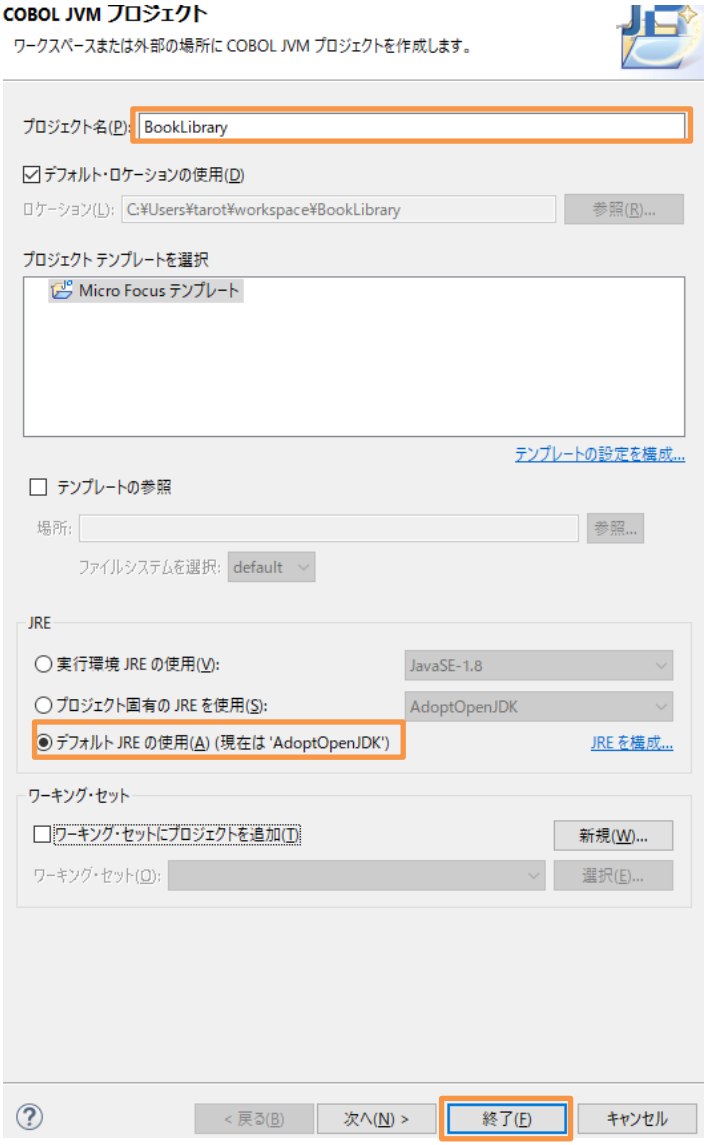
## 3.3. JVM COBOL プロジェクトの作成

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



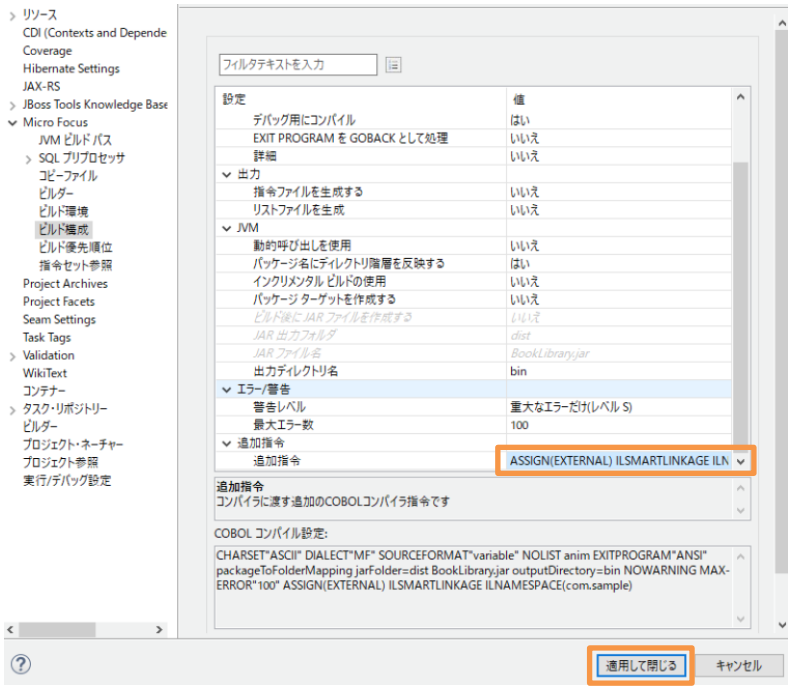

2) プロジェクト名に “BookLibrary” を入力し、「デフォルト JRE の使用」を選択の上、[終了(F)] をクリックします。




補足）
JRE 項目は環境に合わせて変更頂いても構いません。

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上で、[プロパティ(R)] を選択します。


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

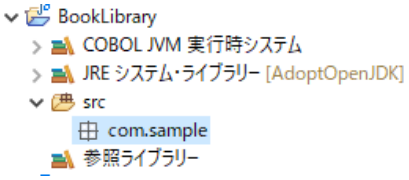



補足）
ILSMARTLINKAGE コンパイラ指令により、COBOL と Java 言語でのデータ型の差異を吸収するラッパークラスがコンパイル時に自動生成されます。
本指令の詳細については、製品マニュアルトップより、[リファレンス] > [コンパイラ指令] > [コンパイラ指令 - アルファベット順一覧] > [ILSMARTLINKAGE] を参照してください。
ILNAMESACE コンパイラ指令は、JVM COBOL の出力先のパッケージ階層を指定するもので、今回は、com.sample パッケージ配下に出力します。
本指令の詳細については、製品マニュアルトップより、[リファレンス] > [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。

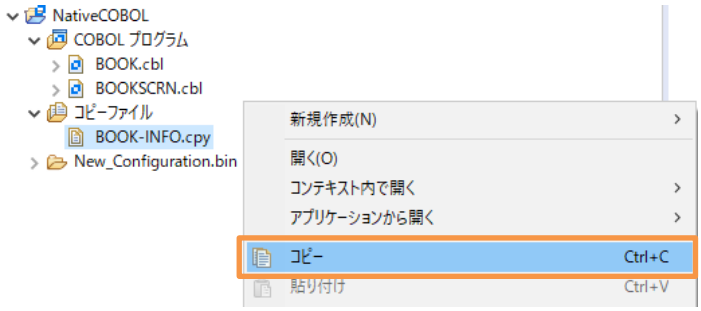
5) BookLibrary プロジェクト配下の src フォルダを選択し、マウスの右クリックにてコンテクストメニューを表示した上で、[新規作成(N)] > [COBOL JVM パッケージ] を選択します。


6) 名前に “com.sample” を指定して、[終了(F)] をクリックします。

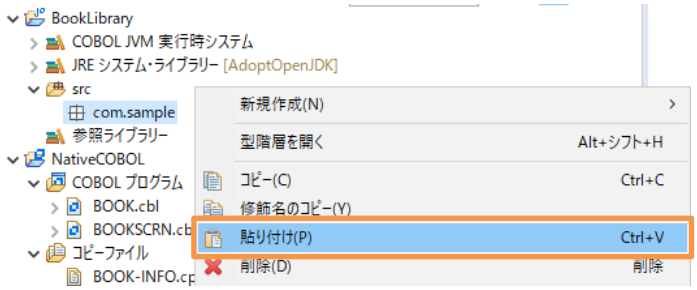

src フォルダ配下に空のフォルダが作成されます。




7) NativeCOBOL プロジェクトのコピーファイルフォルダ配下の 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上で、[コピー] を選択します。



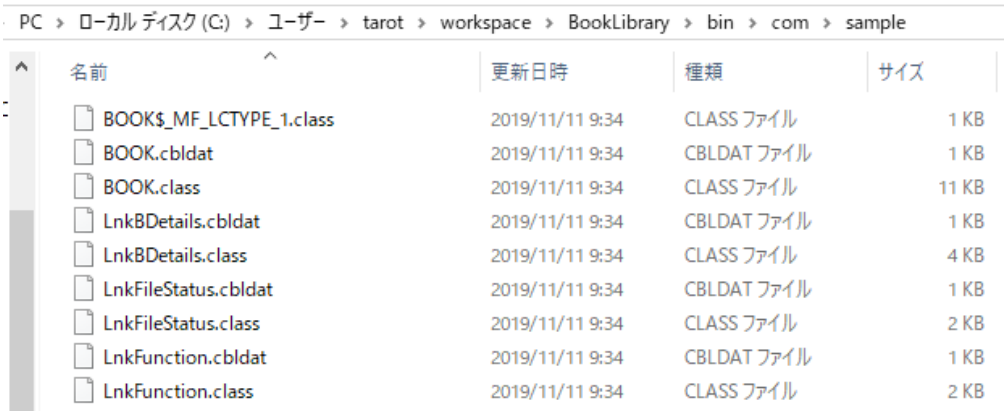

8) BookLibrary プロジェクト配下の src¥com.sample フォルダを選択し、マウスの右クリックにてコンテクストメニューを表示した上で、[貼り付け(P)] を選択します。




この操作により、src¥com.sample フォルダ内に BOOK-INFO.cpy がコピーされます。

9) さきほどと同様の手順にて、NativeCOBOL プロジェクトの COBOL プログラムフォルダ配下の BOOK.cbl を BookLibrary プロジェクト配下の src フォルダにコピーしてください。

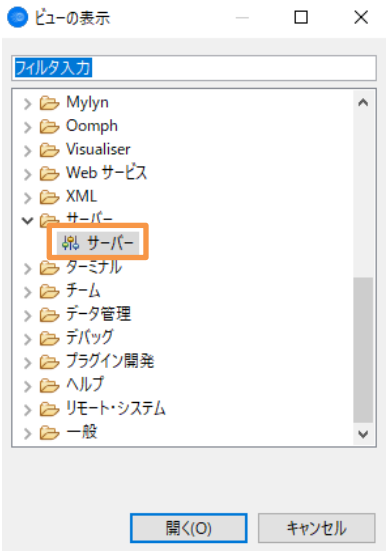

デフォルトで自動的にビルドが有効になっているため、BOOK.cbl をコピーした段階でコンパイルが行われ、プロジェクトが保存されたフォルダ配下の bin¥com¥sample フォルダに、以下の .class ファイル、すなわち、Java バイトコードが生成されます。



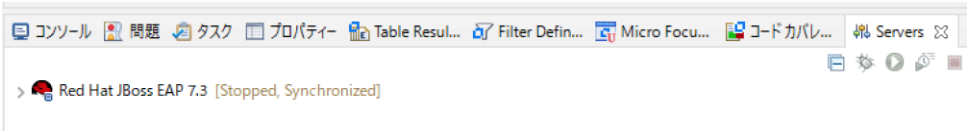
## 3.4. JBoss EAP 上で動作する RESTful Web サービスの作成

本節を実行するにあたり、Visual COBOL for Eclipse 上で、予め JBoss EAP のサーバー設定を実施してください。

これは、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。



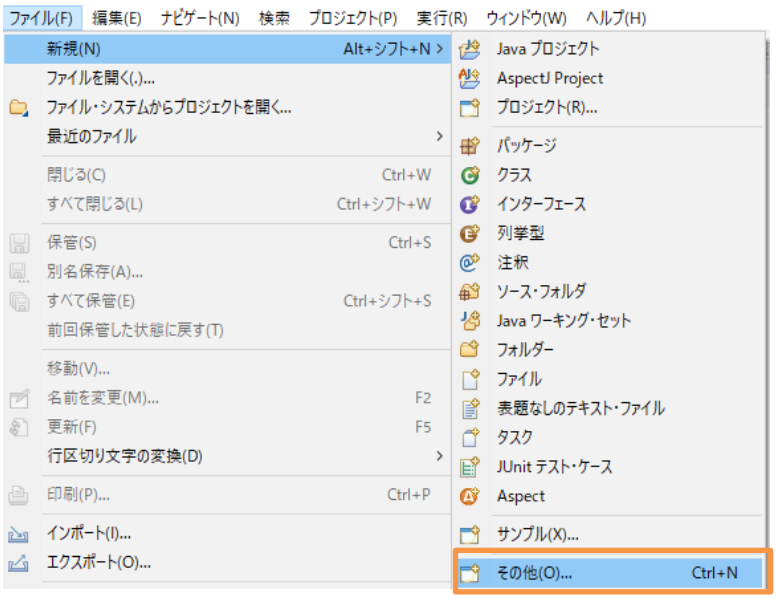

本チュートリアルでは、"JBoss AS, Wildfly & EAP Server Tools" プラグインを Eclipse に導入した上で、当該プラグインを利用してサーバーを作成しています。



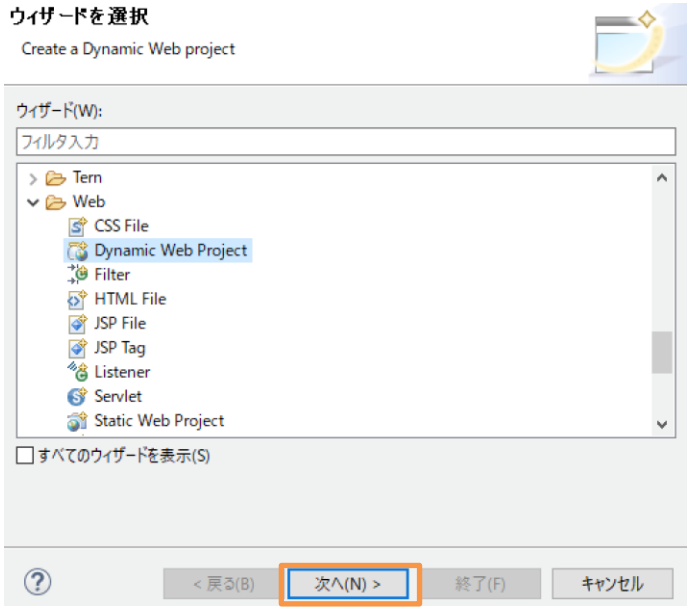

なお、プラグインの導入方法につきましては、別途オンライン文献などを参照ください。

### 3.4.1. Web プロジェクトの作成

1) Visual COBOL for Eclipse メニューより、[ファイル(F)] > [新規(N)] > [その他(O)] を選択します。



2) Web フォルダ配下の「Dynamic Web Project」を選択し、[次へ(N)] をクリックします。

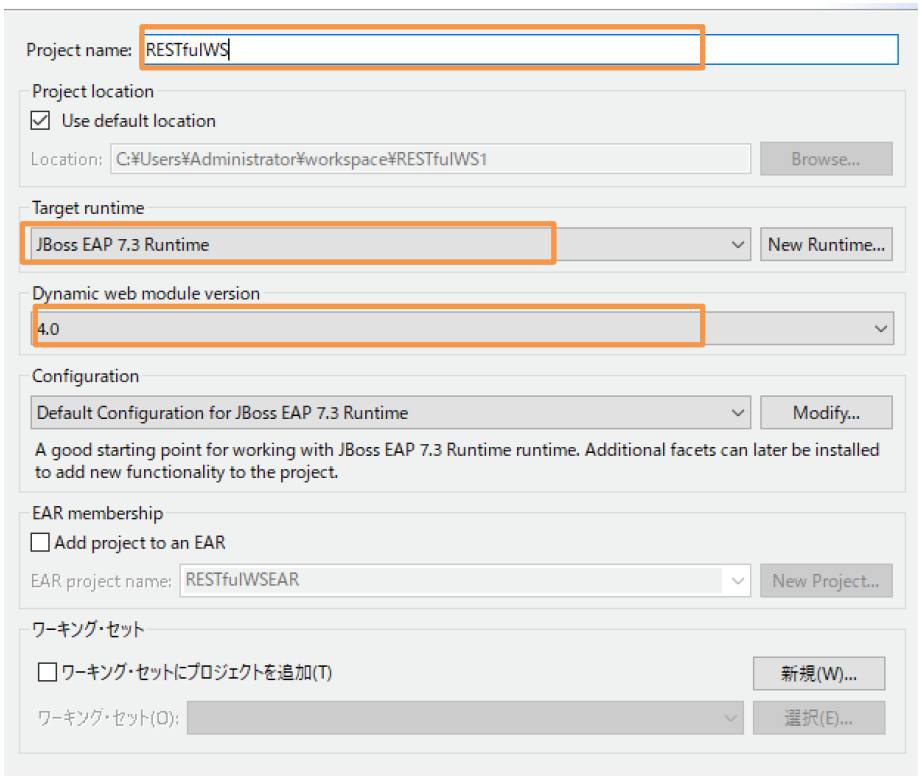



3) 以下の入力を行い、[次へ(N)] をクリックしながらプロジェクトの設定を行います。

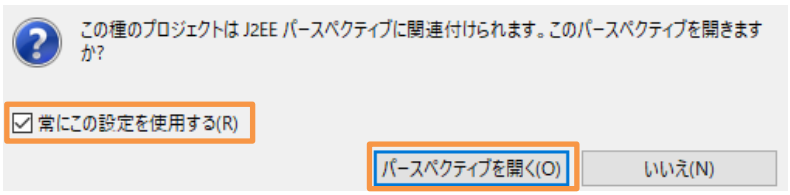
Project name: “RESTfulWS”

Target runtime:「JBoss EAP 7.3 Runtime」

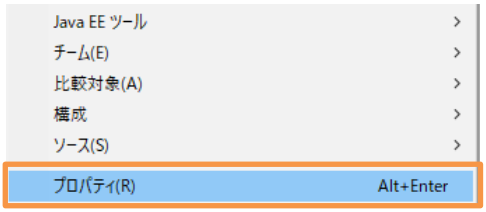
Dynamic web module version:「4.0」



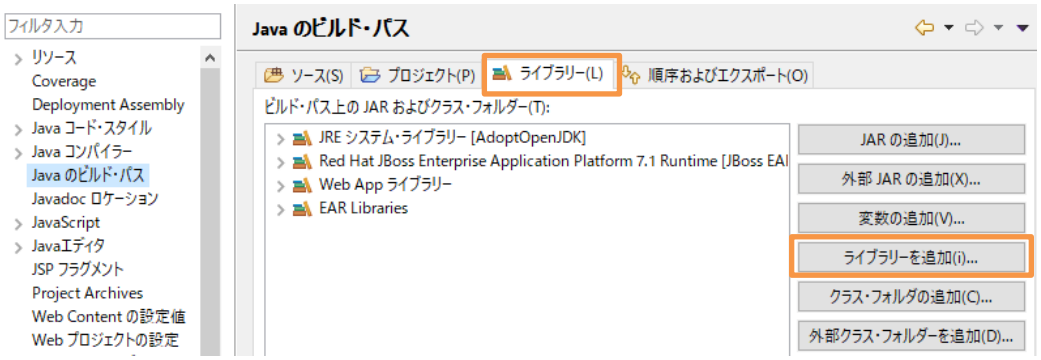

以下のダイアログが表示された場合、「常にこの設定を使用する」にチェックをした上で、[パースペクティブを開く(O)] をクリックします。



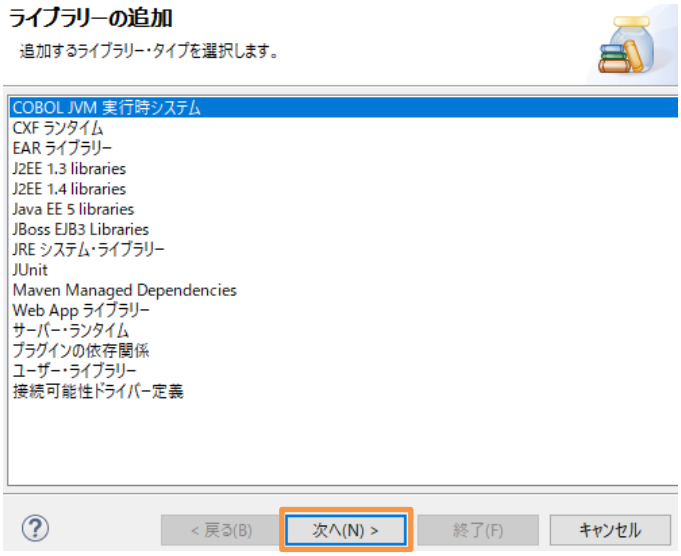

4) RESTfulWS プロジェクト名を選択し、[プロパティ(R)] を選択します。




5) 「Java のビルド・パス」を選択し、「ライブラリー」タブを選択した上で、[ライブラリーを追加(i)] をクリックします。

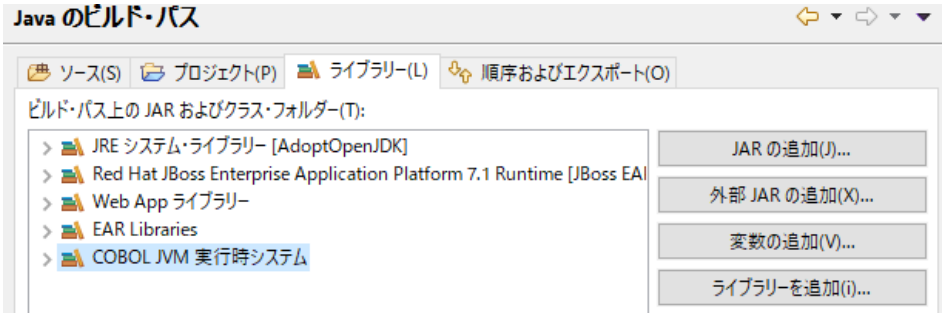



6) 「COBOL JVM 実行時ランタイム」を選択し、[次へ(N)] をクリックします。

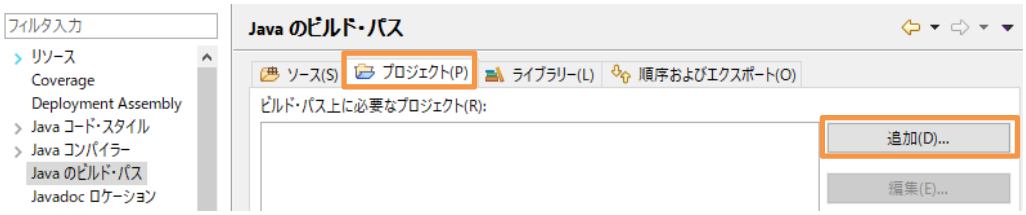


そのまま、[終了(F)] をクリックします。

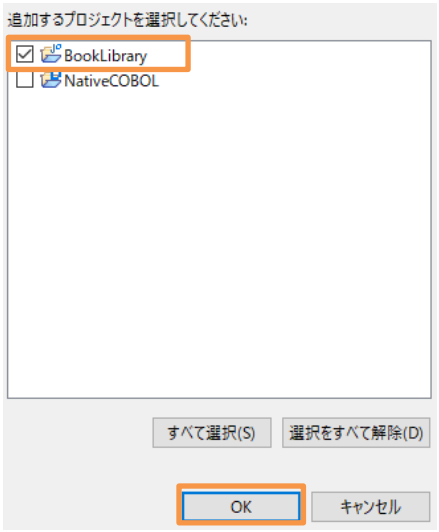

「COBOL JVM 実行時ランタイム」が追加されます。



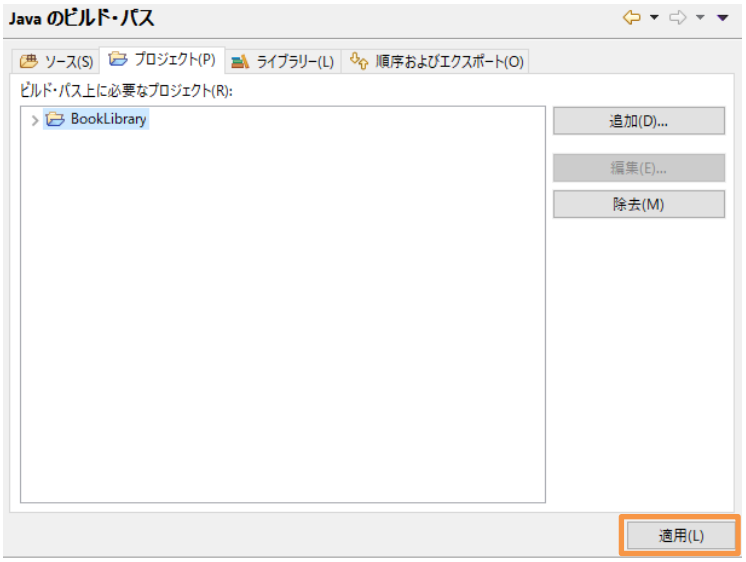

7) 「プロジェクト」タブを選択し、[追加(D)] をクリックします。



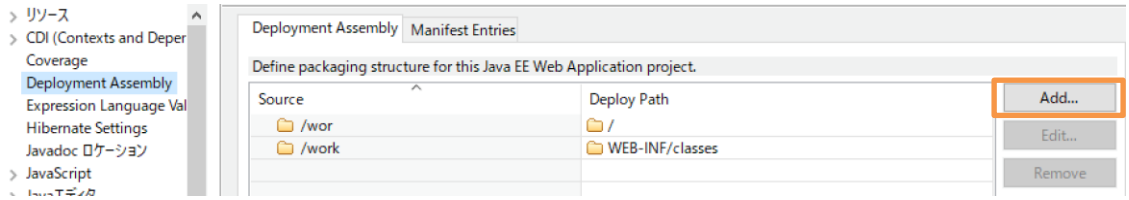
「BookLibrary」にチェックを入れ、[OK] をクリックします。



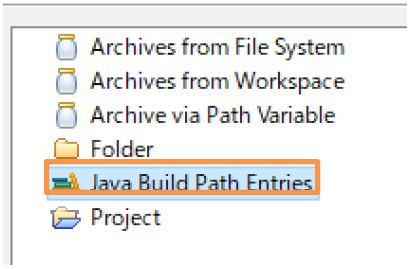

「BookLibrary」が追加されたことを確認し、[適用(L)] をクリックします。



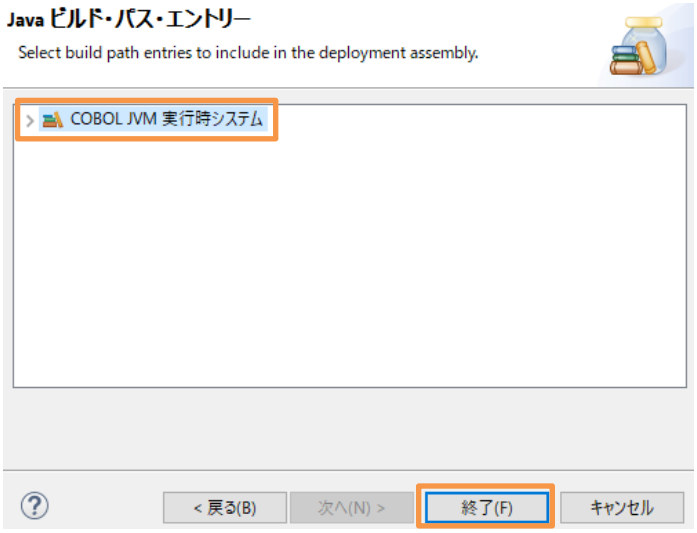

8) 「Deployment Assembly」を選択し、[Add...] をクリックします。



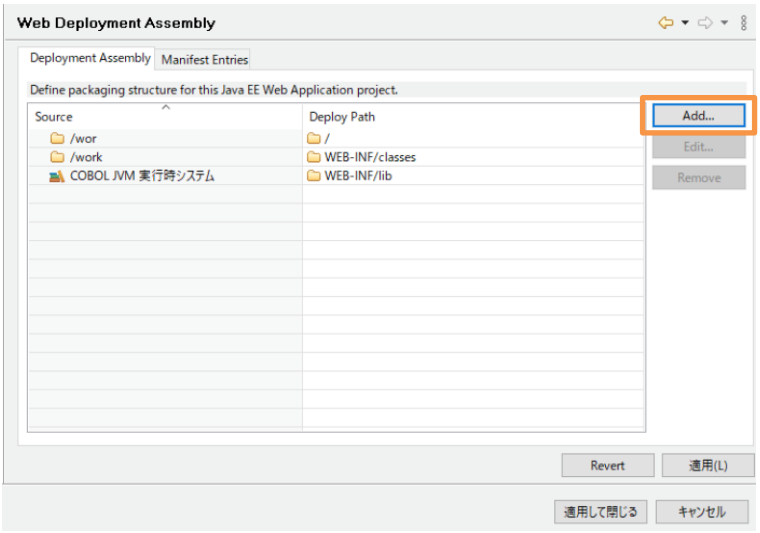

9) 「Java Build Path Entries」を選択し、[次へ(N)] をクリックします。



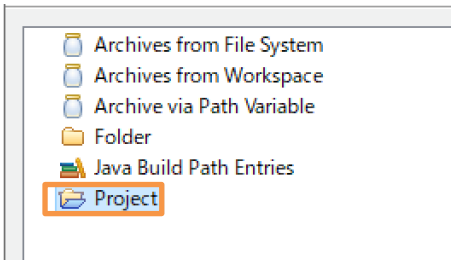
10)「COBOL JVM 実行時ランタイム」を選択し、[終了(F)] をクリックします。



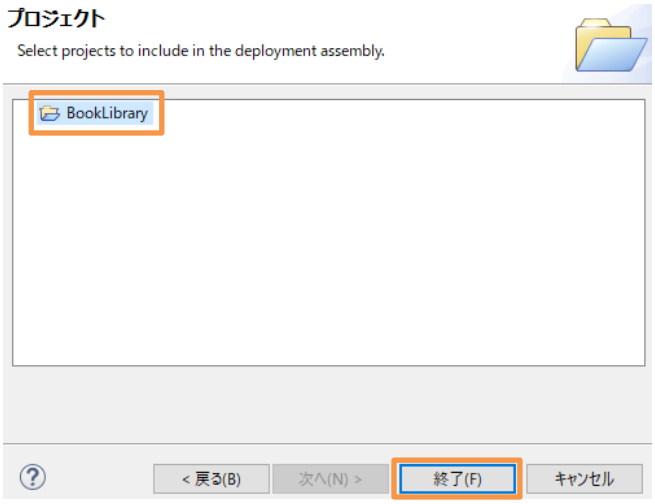
「COBOL JVM 実行時システム」が追加されたことを確認した上で、再度［Add...］をクリックします。



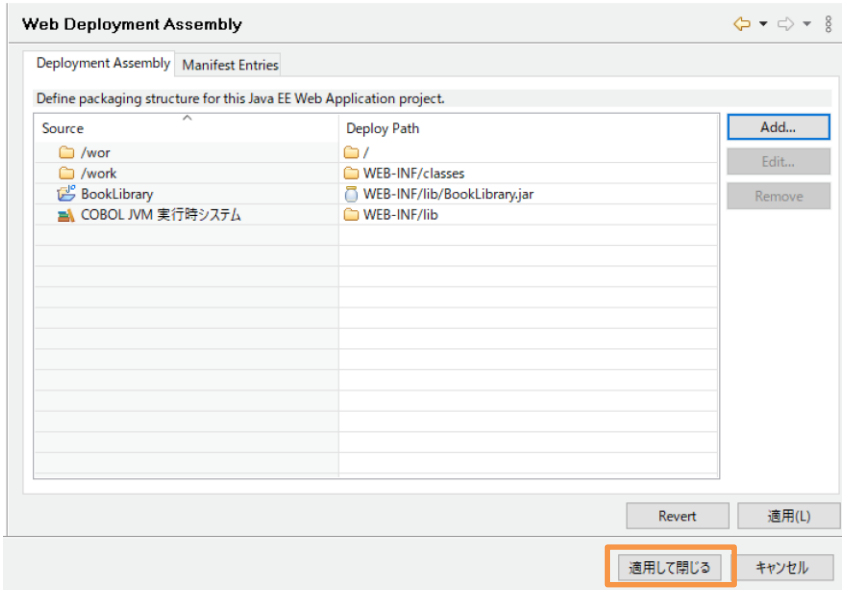

11)「Project」を選択し、[次へ(N)] をクリックします。




12)「BookLibrary」 を選択し、[終了(F)] をクリックします。

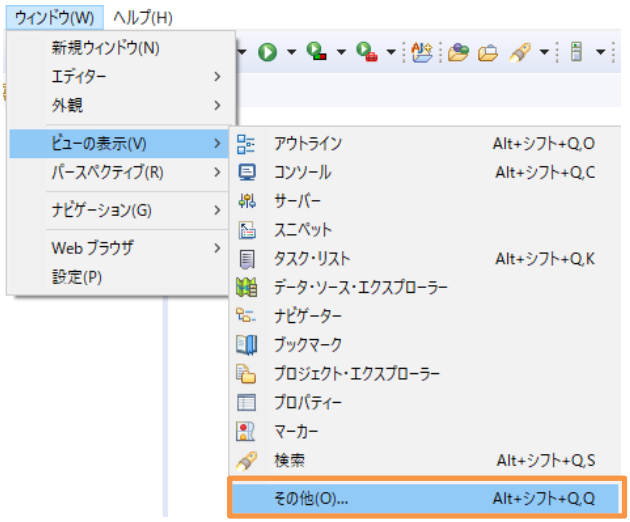



13) BookLibrary が追加されていることを確認し、[適用して閉じる] ボタンをクリックします。

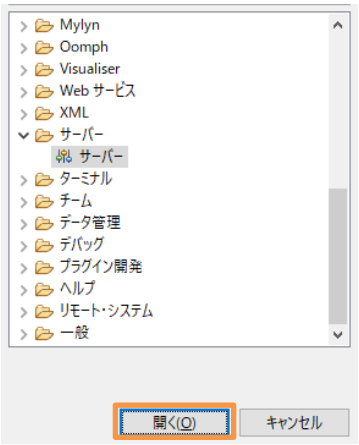



### 3.4.2. JBoss EAP サーバー設定

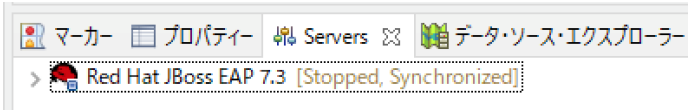
1) Eclipse のメニューより、[ウィンドウ(W)] > [ビューの表示(V)] > [その他(O)] を選択します。




2) サーバーフォルダ配下の「サーバー」を選択し、[開く(O)] をクリックします。

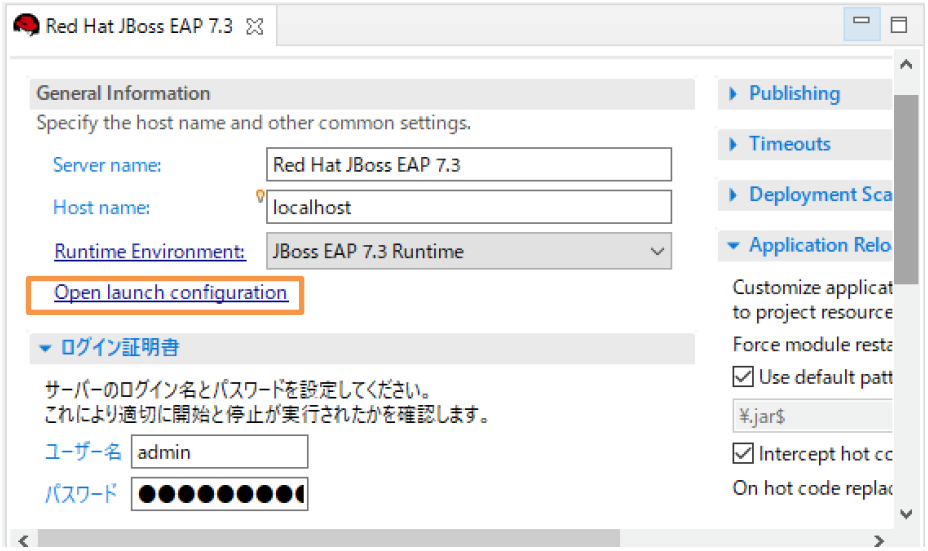


サーバービューが開きます。

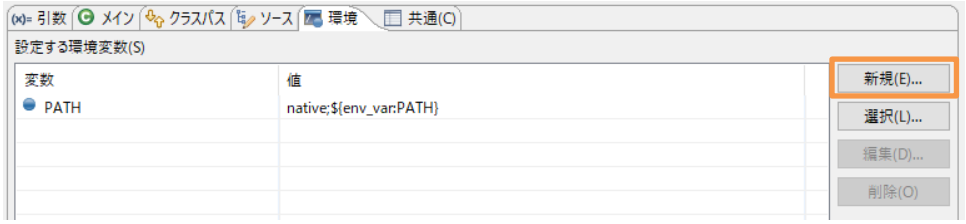



3) 登録した JBoss EAP サーバー情報をダブルクリックします。

4) 「Open launch configuration」をクリックします。



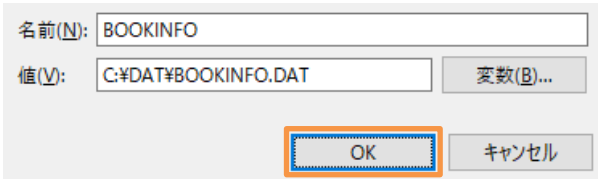

5) 「環境」タブを選択し、[新規(E)] をクリックします。



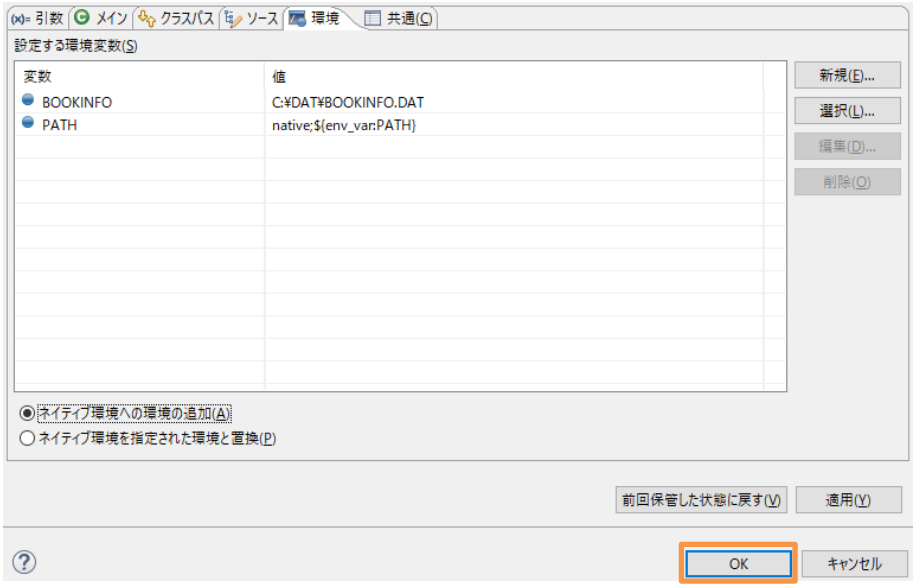

6) 以下の入力を行い、[OK] をクリックします。

名前： “BOOKINFO”

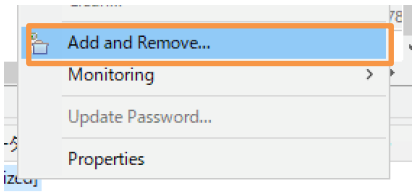
値： “C:¥DAT¥BOOKINFO.DAT”



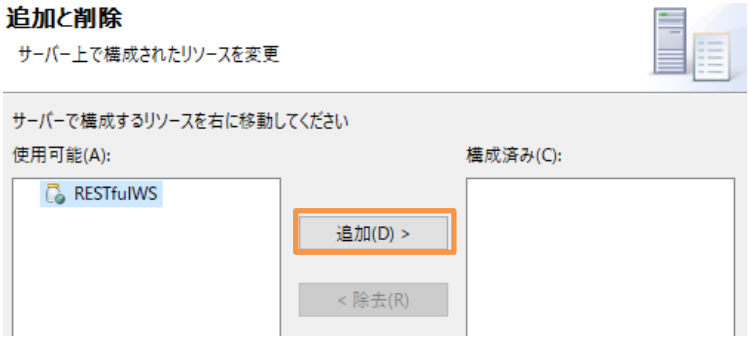

7) BOOKINFO 環境変数が追加されたことを確認し、[OK] をクリックします。



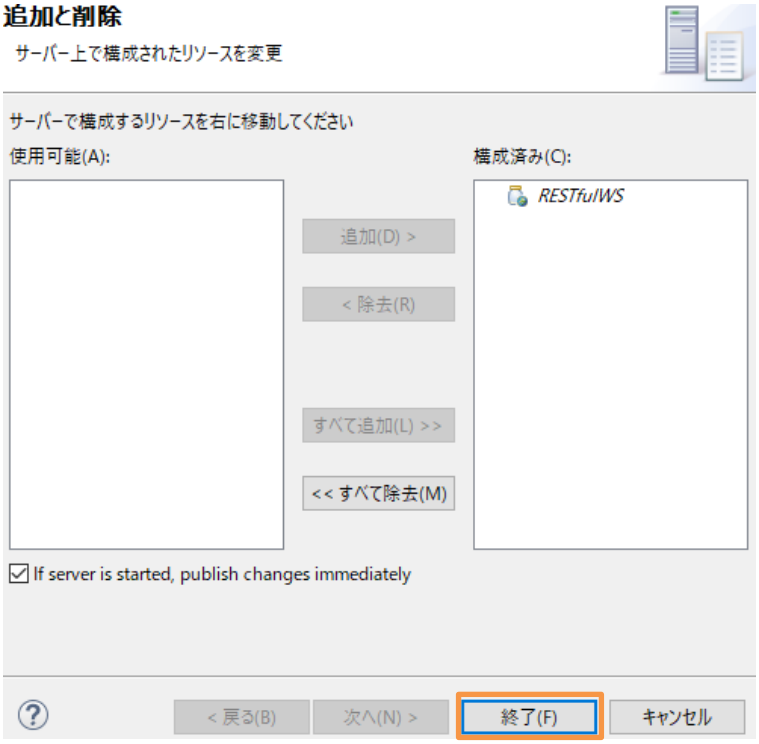

8) サーバービューより、JBoss EAP サーバーを選択し、マウスの右クリックにてコンテクストメニューを表示した上で、[Add and Remove...] をクリックします。




使用可能欄にある「RESTfulWS」を選択し、[追加(D)] をクリックします。



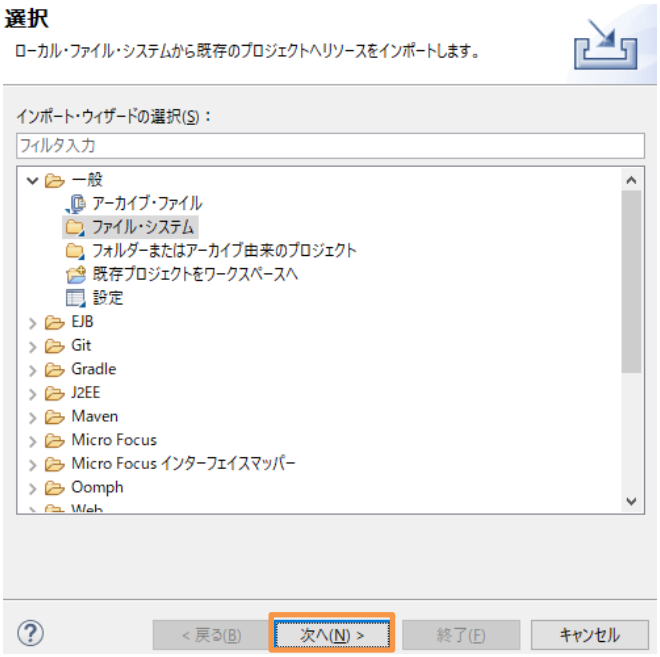
構成済み欄に移動したことを確認し、[終了(F)] をクリックします。




### 3.4.3. Java プログラムのインポート

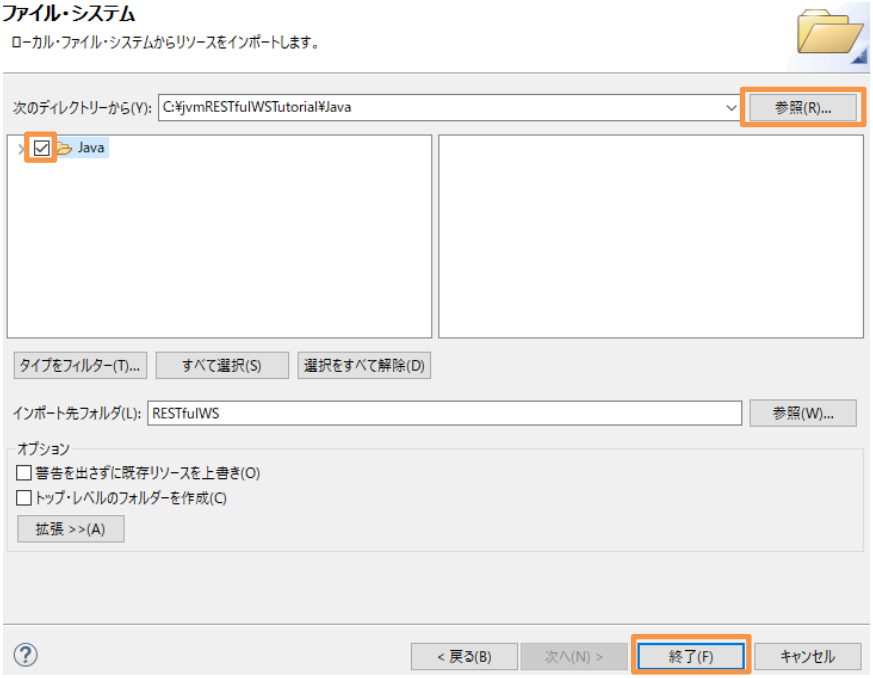
1) RESTfulWS プロジェクト配下の [Java Resources] > [work] を選択し、マウスの右クリックにてコンテクストメニューを表示した上で、[インポート(I)] > [インポート(I)] を選択します。

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

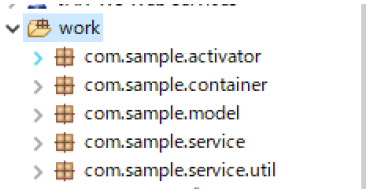



3) [参照(R)] をクリックし、チュートリアルファイル解凍フォルダ¥Java フォルダを選択した後、「Java」にチェックを行った上で、

[終了(F)] をクリックします。




Java ファイルがインポートされます。




4) Java から COBOL を呼び出している個所を確認するため、BookSalesService.java をダブルクリックします。

```
public Response readBookInfo(@PathParam("stockNo") final String stockNo) {
    BOOK book = new BOOK();
    LnkFunction lnkFunction = new LnkFunction();
    LnkBDetails lnkBDetails = new LnkBDetails();
    LnkFileStatus lnkFileStatus = new LnkFileStatus();

    RunUnit runUnit = new RunUnit();
    runUnit.addInstance(book);
    runUnit.addInstance(lnkFunction);
    runUnit.addInstance(lnkBDetails);
    runUnit.addInstance(lnkFileStatus);
    lnkFunction.setLnkFunction("1");
    lnkBDetails.setLnkBStockno(stockNo);
    book.BOOK(lnkFunction, lnkBDetails, lnkFileStatus);

    BookInfo bookInfo = new BookInfo();
    bookInfo.setStockNo(lnkBDetails.getLnkBStockno());
    bookInfo.setTitle(lnkBDetails.getLnkBTitle());
    bookInfo.setAuthor(lnkBDetails.getLnkBAuthor());
    bookInfo.setGenre(lnkBDetails.getLnkBType());
    bookInfo.setRetail(lnkBDetails.getLnkBRetail());
    bookInfo.setOnHand(lnkBDetails.getLnkBOnhand());
    bookInfo.setSold(lnkBDetails.getLnkBSold());

    ResponseData rd = new ResponseData();
    rd.setBookInfo(bookInfo);
    rd.setFileStatus(ServiceUtil.convFileStatus(lnkFileStatus));

    runUnit.stopRunUnit();
    return Response.status(200)
                   .entity(rd)
                   .build();
}
```

JVM COBOL では、従来の COBOL プログラムから PROGRAM-ID 名でクラス、および、メソッドが自動作成されます。今回の BOOK.cbl は PROGRAM-ID が “BOOK” であるため、BOOK クラスとなります。また、前手順で設定した ILSMARTLINKAGE コンパイラ指令により、LINKAGE SECTION に指定されていた LNKFUNCTION,LNK-FILE-STATUS, LNK-B-DETAILS に対応するラッパークラスも合わせて作成されます。このラッパークラスを利用することで、Java 言語と COBOL のデータ型の差異を意識することなく、一般的な Java のコーディングで記述できていることが確認できます。

```
BOOK book = new BOOK();
LnkFunction lnkFunction = new LnkFunction();
LnkBDetails lnkBDetails = new LnkBDetails();
LnkFileStatus lnkFileStatus = new LnkFileStatus();

RunUnit runUnit = new RunUnit();
runUnit.addInstance(book);
runUnit.addInstance(lnkFunction);
runUnit.addInstance(lnkBDetails);
runUnit.addInstance(lnkFileStatus);
lnkFunction.setLnkFunction("1");
lnkBDetails.setLnkBStockno(stockNo);
book.BOOK(lnkFunction, lnkBDetails, lnkFileStatus);
```

補足）
多くの COBOL プログラムは、シングルスレッドでの動作を前提としています。しかし、Web アプリケーションは一般的にマルチスレッドのため、WORKING-STORAGE SECTION のようなプロセス共有資源のスレッド間衝突による問題が発生する可能性があります。 com.microfocus.cobol.runtime.core.RunUnit は、これらの共有資源を各スレッドで独立して利用できるようにします。

### 3.4.4. JBoss EAP サーバーを Eclipse 上から起動

1) サーバービューより、JBoss EAP サーバーを選択し、ビューの右上にある開始アイコンをクリックします。

コンソールビューで、サーバーが正常に稼働したことを確認します。

```
.4:40:58,817 INFO  [org.jboss.as.ejb3] (MSC service thread 1-1) WFLYEJB0482: 厳格なプール mdb-strict-max-po
.4:40:58,820 INFO  [org.jboss.as.ejb3] (MSC service thread 1-1) WFLYEJB0481: 厳格なプール slsb-strict-max-p
.4:40:58,992 INFO  [org.wildfly.extension.undertow] (MSC service thread 1-3) WFLYUT0012: サーバー default-s
.4:40:59,012 INFO  [org.jboss.remoting] (MSC service thread 1-2) JBoss Remoting version 5.0.23.Final-redh
.4:40:59,121 INFO  [org.wildfly.extension.undertow] (MSC service thread 1-1) WFLYUT0018: ホスト default-hos
.4:40:59,214 INFO  [org.wildfly.extension.undertow] (MSC service thread 1-3) WFLYUT0006: Undertow HTTP リ
.4:40:59,618 INFO  [org.jboss.as.ejb3] (MSC service thread 1-4) WFLYEJB0493: EJB サブシステム中断が完了しました
.4:40:59,786 INFO  [org.jboss.as.connector.subsystems.datasources] (MSC service thread 1-4) WFLYJCA0001:
.4:41:03,182 INFO  [org.jboss.as.patching] (MSC service thread 1-1) WFLYPAT0050: JBoss EAP 累積パッチ ID: j
.4:41:03,210 WARN  [org.jboss.as.domain.management.security] (MSC service thread 1-4) WFLYDM0111: キーストア
.4:41:03,227 INFO  [org.jboss.as.server.deployment.scanner] (MSC service thread 1-2) WFLYDS0013: ディレクトリ
.4:41:03,668 INFO  [org.wildfly.extension.undertow] (MSC service thread 1-4) WFLYUT0006: Undertow HTTPS !
.4:41:05,187 INFO  [org.jboss.ws.common.management] (MSC service thread 1-4) JBWS022052: Starting JBossWS
.4:41:05,536 INFO  [org.jboss.as.server] (Controller Boot Thread) WFLYSRV0212: サーバーを再開しています
.4:41:05,543 INFO  [org.jboss.as] (Controller Boot Thread) WFLYSRV0025: JBoss EAP 7.3.8.GA (WildFly Core
.4:41:05,551 INFO  [org.jboss.as] (Controller Boot Thread) WFLYSRV0060: http://127.0.0.1:9990/management
.4:41:05,552 INFO  [org.jboss.as] (Controller Boot Thread) WFLYSRV0051: 管理コンソールは http://127.0.0.1:999
```

2) Web ブラウザーで以下のアドレスを開きます。

http://localhost:8080/RESTfulWS/bookmgnt/books/1111


Java で作成した Web サービスが COBOL プログラムを呼び出し、その結果を正しく戻していることが確認できます。

なお、ブラウザーの種類によっては、JSON ファイルのダウンロードとなることがありますが、これはブラウザー側の挙動によるものです。

3) チュートリアルファイル解凍フォルダ配下の WebClient フォルダにある SearchBook.html をブラウザーで開きます。


こちらは、今回作成した RESTful Web サービスを呼び出す Web 画面となります。さきほど確認したように REST API を介して JSON 形式でデータの送受信を行うモダンなシステムインターフェースとなっています。

書籍検索機能などを確認した後、Eclipse IDE のサーバービューの JBoss EAP サーバーを選択し、停止アイコンをクリックします。

補足）

本チュートリアルでは、Eclipse より JBoss EAP サーバーを起動しましたが、JBoss EAP サーバーの deployments

フォルダ内に WAR がコピーされており、ディプロイされています。このため、通常のサーバー起動方法からも今回作成したサービスを利用することができます。
なお、standalone サーバーで設定した場合、ディプロイフォルダは、以下になります。
JBoss EAP サーバーインストールフォルダ¥standalone¥deployments

## 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。