

Visual COBOL チュートリアル

RESTful Web サービスによる COBOL 資産の再利用

Eclipse リモート開発編

1 目的

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

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

2 前提条件

本チュートリアルでは、Linux サーバーとのリモート開発を行います。リモート開発については、「Visual COBOL チュートリアル COBOL 開発：Linux/UNIX 版 リモート開発編」を参照してください。

また、本資料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 Linux/UNIX でも利用可能です。

開発クライアント(Windows) ソフトウェア

OS Windows Server 11

COBOL 開発環境製品 Visual COBOL 11.0 for Eclipse Patch Update 01

開発クライアント(Linux) ソフトウェア

OS Rocky Linux 9.4

COBOL 開発環境製品 Visual COBOL Development Hub 11.0 Patch Update 01

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、任意のフォルダーに解凍しておいてください。

[サンプルプログラムのダウンロード](#)

内容

- 1 目的
- 2 前提条件
- 3 チュートリアル手順
 - 3.1 Windows クライアントでの開発準備作業
 - 3.2 チュートリアルファイルのインポート
 - 3.3 リモートサーバーの Enterprise Server の追加と起動
 - 3.4 デプロイフォルダー、データファイルの準備
 - 3.5 RESTful Web サービスの開発作業
 - 3.6 COBOL アプリケーションを Enterprise Server へデプロイ
 - 3.7 Enterprise Server インスタンスへの環境設定と有効化
 - 3.8 RESTful Web サービスのテスト
 - 3.9 RESTful Web サービスのデバッグ
 - 3.10 サーバーの停止

3 チュートリアル手順

3.1 Windows クライアントでの開発準備作業

1) Visual COBOL for Eclipse を起動

- ① [スタート] メニュー > [すべてのアプリ] > [Rocket Visual COBOL] > [Visual COBOL for Eclipse] を選択します。

ワークスペースには任意のフォルダーを選択してください。

また、起動後、ようこそ画面は閉じてください。

2) ネイティブ COBOL プロジェクトの作成とプログラムソースのインポート

- ① [ファイル(F)]メニュー > [新規(N)] > [リモート COBOL プロジェクト] を選択し、プロジェクト名に “RemoteNativeCOBOL” を入力、ファイル システムに “セキュアシェル (SSH) ファイル システム”を選択して、[次へ(N)] をクリックします。

リモート COBOL プロジェクト

ワークスペースまたは外部にリモート COBOL プロジェクトを作成



プロジェクト名: RemoteNativeCOBOL

ファイル システム

ファイル システムを選択: セキュアシェル (SSH) ファイル システム

セキュアシェル (SSH) ファイル システムを使用すると、SSH 接続サポートのみを使用してリモート プロジェクトを処理できます。ローカル ファイル システム上の場所を指定する必要はありませんが、リモート マシン上の場所のみ指定する必要があります。

リモート ファイル システムの場合、RSE サポートによりリモート プロジェクトで作業できます。ローカル ファイル システムの場所の指定は不要で、リモートマシン上の場所の指定だけです。

ネットワーク ファイル システムの場合は、ローカルマシン上のプロジェクトの場所(マップされたドライブ上のプロジェクトパス)とリモートマシン上のパスを指定する必要があります。

? < 戻る(B) 次へ(N) > 終了(F) キャンセル


- ② [プロジェクト テンプレート] に “Rocket テンプレート (64 ビット)” を選択して、[次へ(N)] をクリックします。

リモート COBOL プロジェクト

ワークスペースまたは外部にリモート COBOL プロジェクトを作成



プロジェクト テンプレートを選択

 Rocket テンプレート [32 ビット]

 Rocket テンプレート [64 ビット]

[テンプレートの設定を構成](#)

☐ 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 ▼



< 戻る(B)

次へ(N) >

終了(F)

キャンセル

- ③ [接続の新規作成] をクリックします。

プロジェクト名: RemoteNativeCOBOL

リモート設定

接続名:



接続の新規作成...


- ④ [Rocket DevHub SSH のみ] を選択し、[次へ(N)] をクリックします。


システム・タイプ:

フィルタ入力



一般

 Rocket DevHub SSH のみ

 Rocket DevHub SSH 使用



< 戻る(B)

次へ(N) >

終了(F)

キャンセル

- ⑤ [ホスト名] に Linux サーバーのホスト名、もしくは、IP アドレスを入力、[接続名] に任意の名前を入力して、[次へ(N)] をクリックします。

接続情報の定義

親プロファイル：	DESKTOP-TMOAELJ
ホスト名：	172.24.146.141
接続名：	Linux
記述/説明：	

☒ ホスト名を検証
[プロキシ設定を構成](#)

- ⑥ [DevHub の場所] に Development Hub 製品のインストールディレクトリを入力し、[終了(F)] をクリックします。

DevHub SSH アクセス

Rocket DevHub のインストール場所の定義

Rocket DevHub インストール ディレクトリの場所はフルパスである必要があります。
 これは 'DevHub Ssh Access' サブシステムのプロパティを使用して後で変更できます。

DevHub の場所: /opt/rocketsoftware/VisualCOBOL

- ⑦ プロジェクトの保存先を指定し、[終了(F)] をクリックします。

リモート COBOL プロジェクト

ワークスペースまたは外部にリモート COBOL プロジェクトを作成



プロジェクト名: RemoteNativeCOBOL

リモート設定

接続名: Linux 接続の新規作成...

リモートのパス: /home/tarot/RemoteNativeCOBOL 参照...

リモートの場所はリモート マシンのプロジェクト パスに設定しなければいけません。

?
< 戻る(B)
次へ(N) >
終了(E)
キャンセル

認証ダイアログが表示されますので、ユーザー ID、パスワードを入力し、[パスワードを保管(C)]にチェックをしたうえで、[OK] をクリックします。

システム・タイプ: Rocket DevHub SSH のみ

ホスト名: 172.24.146.141

接続名: Linux

ユーザー ID: tarot

パスワード(任意)(B): *****

☐ ユーザー ID の保管
☒ パスワードを保管(C)

OK
キャンセル(A)

以下のようなダイアログが表示された場合は、いずれも [はい(Y)] をクリックします。

パスワードヒントのダイアログについては、任意の選択で構いません。

The authenticity of host '172.24.146.141' can't be established.
 ECDSA key fingerprint is 5a:f0:39:0e:28:63:25:87:8b:c2:dc:0a:5b:81:96:02.
 Are you sure you want to continue connecting?

はい(Y)
いいえ(N)

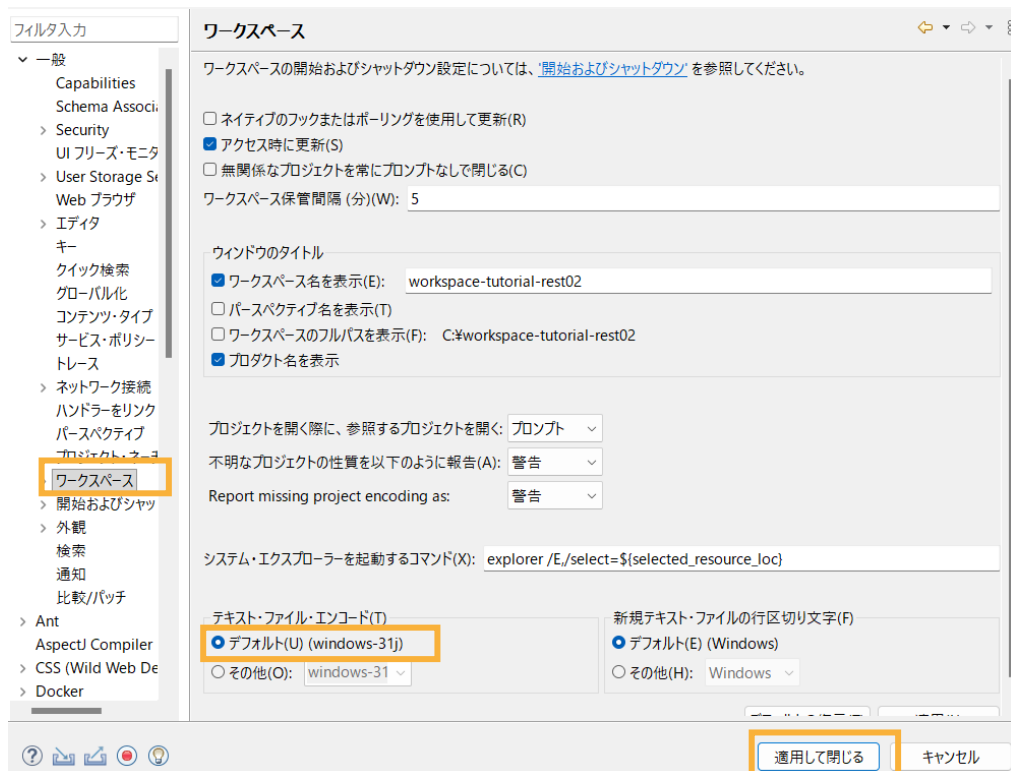
C:\Users\tarot\.ssh\known_hosts does not exist.
Are you sure you want to create it?

はい(Y)

いいえ(N)

3) 文字コードの指定を行います。

- ① 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]メニュー > [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「デフォルト(windows-31j)」もしくは、MS932 に変更し、[適用して閉じる] をクリックします。



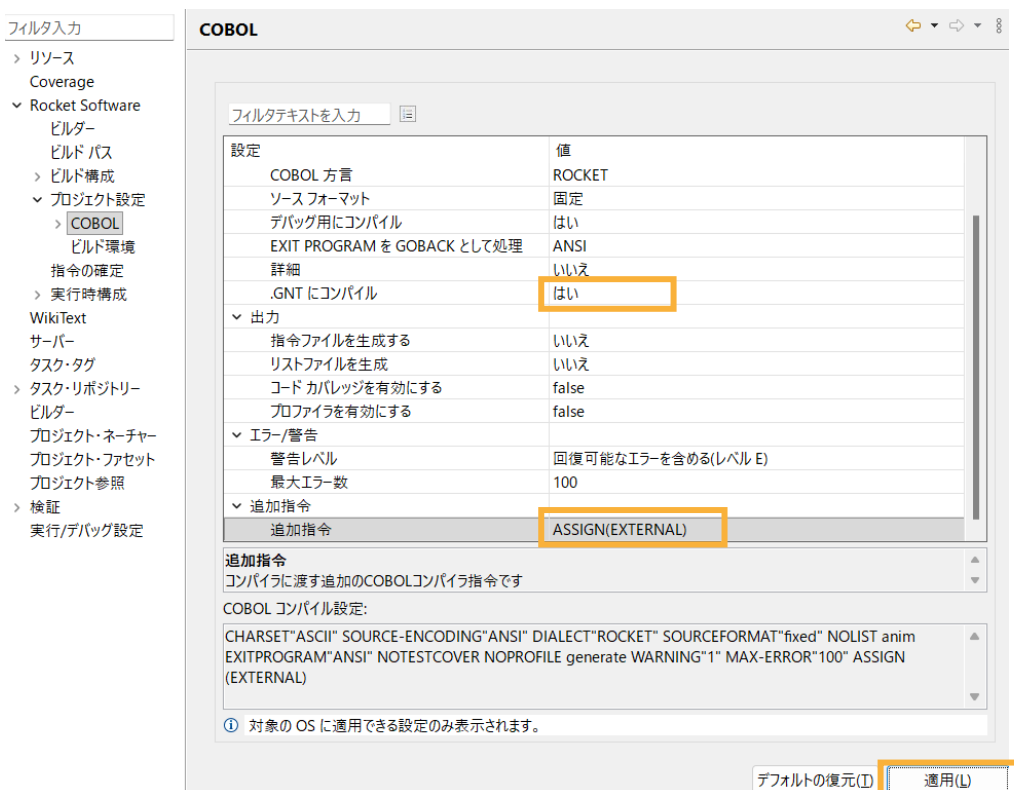
Preference Recorder のダイアログが表示された場合は [キャンセル] を選択してください。

- 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キストメニューを表示し、「プロパティ」を選択します。[Rocket Software] > [プロジェクト設定] > [COBOL] とナビゲートし、[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(L)] をクリックします。

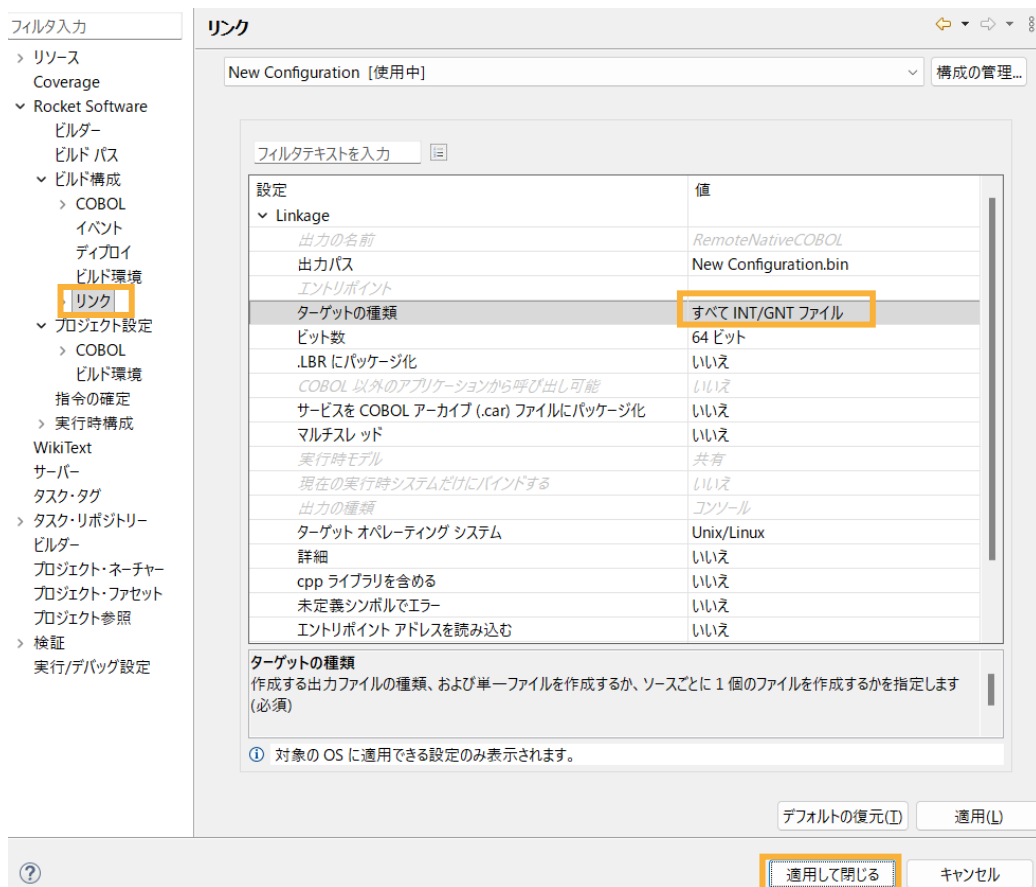


4) ビルドオプションの変更

- ① 現在の画面より、[.GNT にコンパイル] を「はい」に変更し、[追加指令]に "ASSIGN(EXTERNAL)" を指定し、[適用(L)] ボタンをクリックします。



- ② [Rocket Software] > [ビルド構成] > [リンク] をクリックし、[ターゲットの種類] を「すべて INT/GNT ファイル」に変更し、[適用して閉じる] をクリックします。



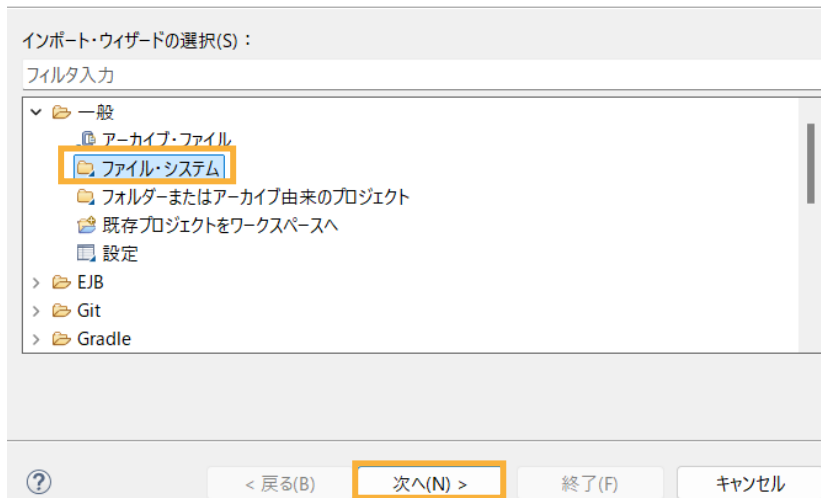
3.2 チュートリアルファイルのインポート

1) チュートリアルファイルのインポート

- ① 作成したプロジェクトを選択し、マウスの右クリックでコンテキ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。
- ② [一般] > [ファイル・システム] を選択し、[次へ] をクリックします。

選択

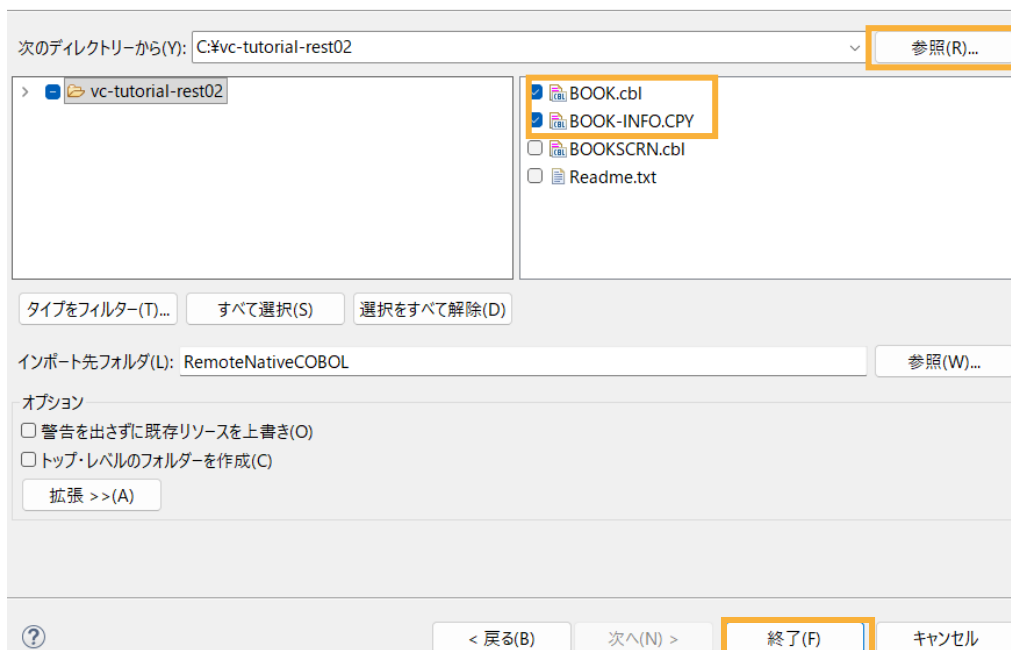
ローカル・ファイル・システムから既存のプロジェクトヘリソースをインポートします。



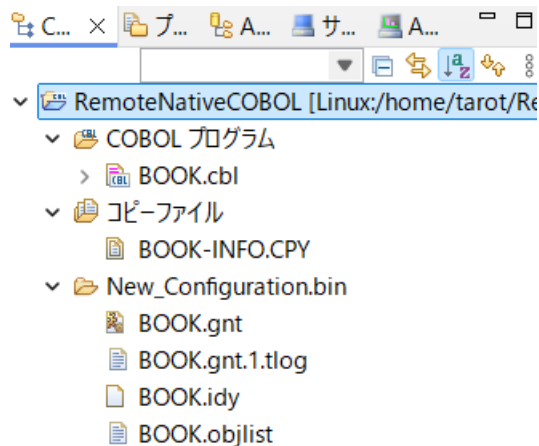
- ③ [参照(R)] ボタンをクリックし、ダウンロードしたサンプルファイルを展開したフォルダーを指定します。下図では “C:\vc-tutorial-rest02” に展開したファイルを指定しています。ここで “BOOK-INFO.cpy” と “BOOK.cbl” を指定し、[終了(F)] をクリックします。

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。



自動的にビルドが行われ、プロジェクト配下が以下ようになります。



3.3 リモートサーバーの Enterprise Server の追加と起動

1) リモートサーバー上でサーバーの起動

- ① ターミナル画面上で、管理者権限を持つユーザーで、リモートサーバーにログインします。
- ② 以下のコマンドを実行します。

```
./opt/rocketsoftware/VisualCOBOL/bin/cobsetenv
```

```
# ./opt/rocketsoftware/VisualCOBOL/bin/cobsetenv
COBDIR set to /opt/rocketsoftware/VisualCOBOL
#
```

Directory Server の起動

以下のコマンドを実行します。

```
mfds&
```

```
# mfds&
[1] 2876
#
```

Enterprise Server Common Web Admin(ESCWA) の起動

以下のコマンドを実行します。

```
escwa --BasicConfig.MfRequestedEndpoint="tcp:*:10086" --write=true &
```

補足)

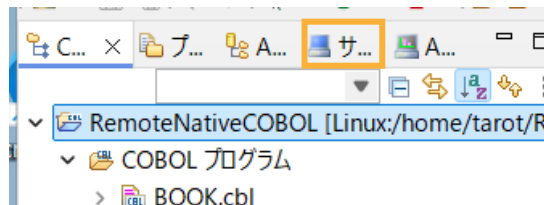
デフォルトでは外部からアクセスすることはできません。外部アクセスを許可するためのオプションを指定しています。

```
# escwa --BasicConfig.MfRequestedEndpoint="tcp:*:10086" --write=true &
[2] 2666
# 2025-11-20 15:32:00.196 Loaded COBOL Run Time Environment Extension
2025-11-20 15:32:00.197 New thread high-water mark: 1 threads are now running
2025-11-20 15:32:00.197 MFCS server "ESCWA" running as process 2666
2025-11-20 15:32:00.197 GK-OS version 2.13.0
2025-11-20 15:32:00.197 GK-Utility version 2.12.3
2025-11-20 15:32:00.198 GkCobGetFuncAddr: 4
2025-11-20 15:32:00.198 ES Common Web Administration version: 7.1.0
```

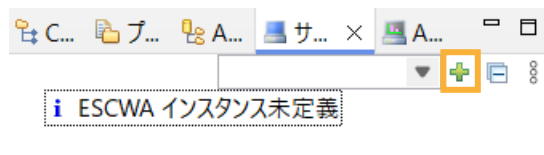
```
2025-11-20 15:32:00.199 Copyright (C) 2019-2025 Rocket Software, Inc. or its affiliates. All rights reserved.
2025-11-20 15:32:00.199 /opt/rocketsoftware/VisualCOBOL/etc
2025-11-20 15:32:00.206 Common Web Admin http endpoint starting on endpoint:
tcp:*:10086
#
```

2) Eclipse 側の設定

- ③ Eclipse に戻り、[サーバーエクスプローラー] タブを選択します。



- ④ [+] ボタンをクリックします。



- ⑤ リモートサーバーの情報を入力し、[OK] をクリックします。

接続の新規作成

既存の Enterprise Server Common Web Administration インスタンスへの接続を新規作成します



名前:	Linux
サーバアドレス (IPv4/ホスト名):	172.24.146.141
サーバポート:	10086
<input type="checkbox"/> TLS 有効	
TLS 設定	
CA 証明書:	
参照...	
<div> ? 終了(F) キャンセル </div>	

認証情報を入力し、[OK] をクリックします。

ESCWA: Linux の接続の詳細を入力してください

☒ サーバーに認証情報が必要

認証情報

ユーザー名: SYSAD

パスワード: *****

☒ 認証情報の保存

☐ 認証情報がクリアされるまで、再度プロンプトを表示しない

OK キャンセル

補足)

この情報は、以下のコマンドで取得できます。

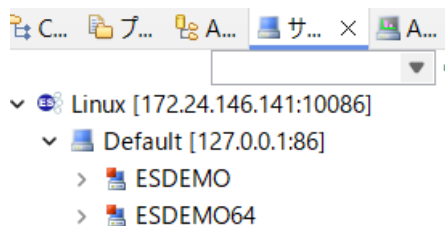
```
mfsecretsadmin read microfocus/temp/admin
```

```
# mfsecretsadmin read microfocus/temp/admin
```

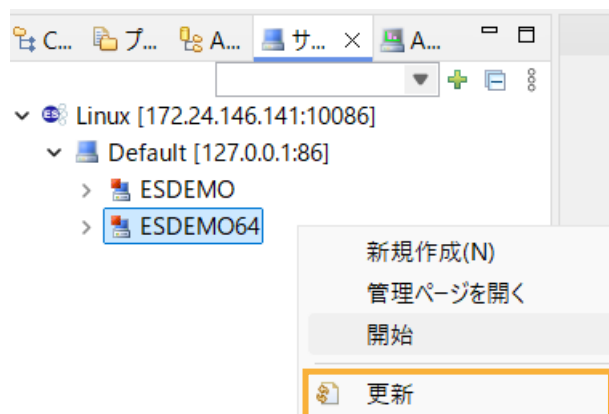
```
{"mfUser":"SYSAD", "mfPassword":"88tudJ8a"}
```

上記の場合、[ユーザー名] に “SYSAD”、[パスワード] に “88tudJ8a” を入力します。

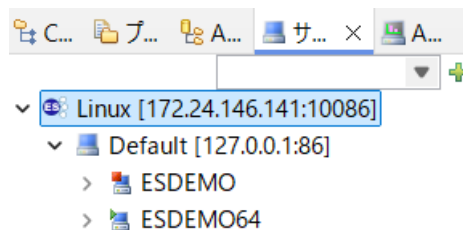
しばらくすると、Directory Server 情報が表示されます。



- ⑥ リモートサーバー配下の「ESDEMO64」を右クリックし、コンテキストメニューから「開始」を選択します。認証ダイアログが表示された場合は、さきほどの認証情報を入力して [OK] を押してください。



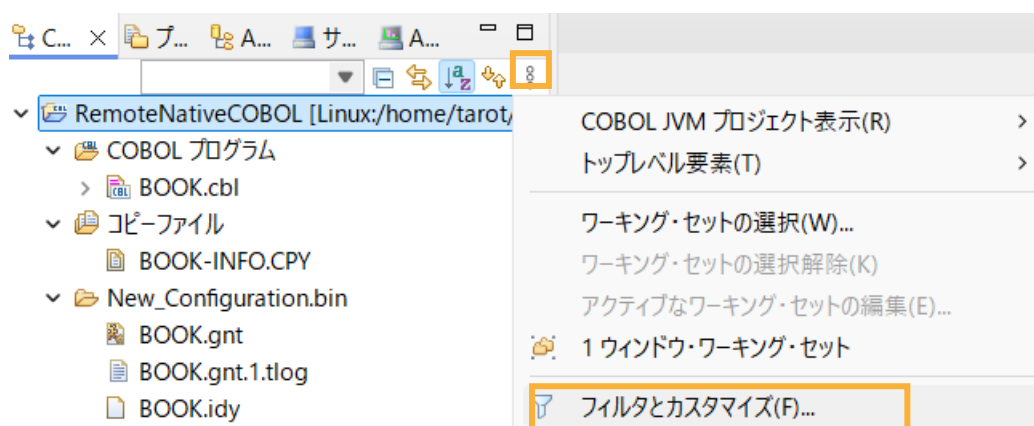
起動すると、ESDEMO64 の左側に緑色のアイコンが表示されます。



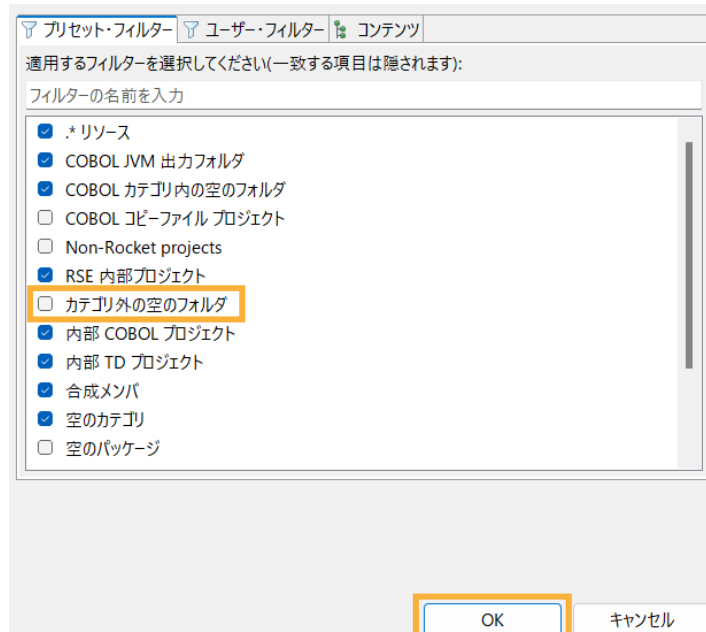
3.4 ディプロイフォルダー、データファイルの準備

1) COBOL エクスプローラーの表示設定

- ① 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)] を選択します。

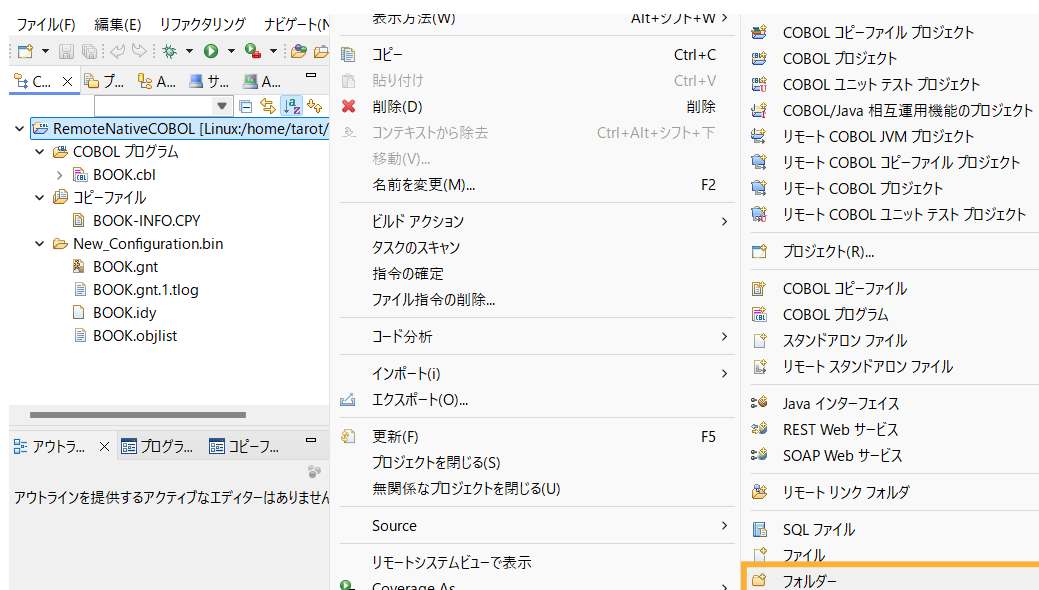


- ③ [カテゴリ外の空のフォルダ] にチェックされている場合は、チェックを外したのち、[OK] ボタンをクリックします。



2) デプロイ用フォルダーの作成

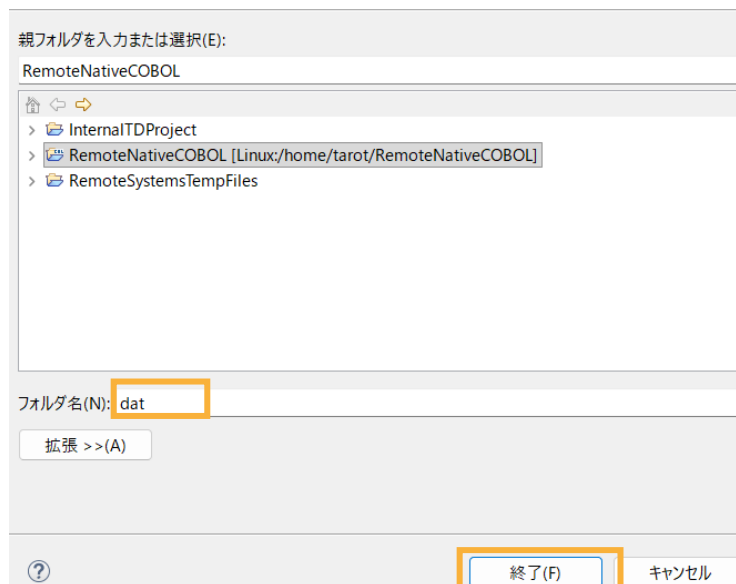
- ① 「RemoteNativeCOBOL」プロジェクトを右クリックし、コンテキストメニューから [新規作成(N)] > [フォルダー] を選択します。



- ② 「RemoteNativeCOBOL」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 "dat" を指定し、[終了 (F)] ボタンをクリックします。

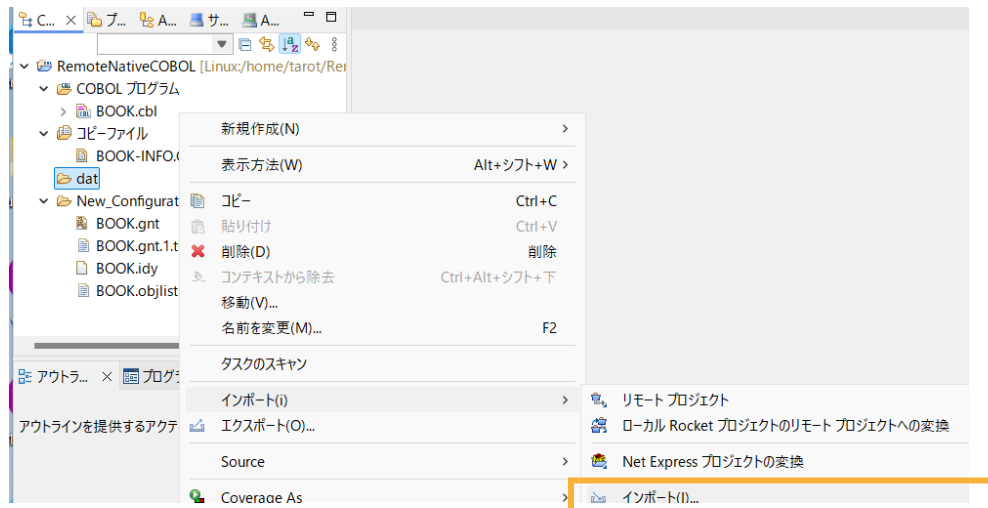
フォルダー

新規のフォルダ・リソースを作成します



3) データファイルのインポート

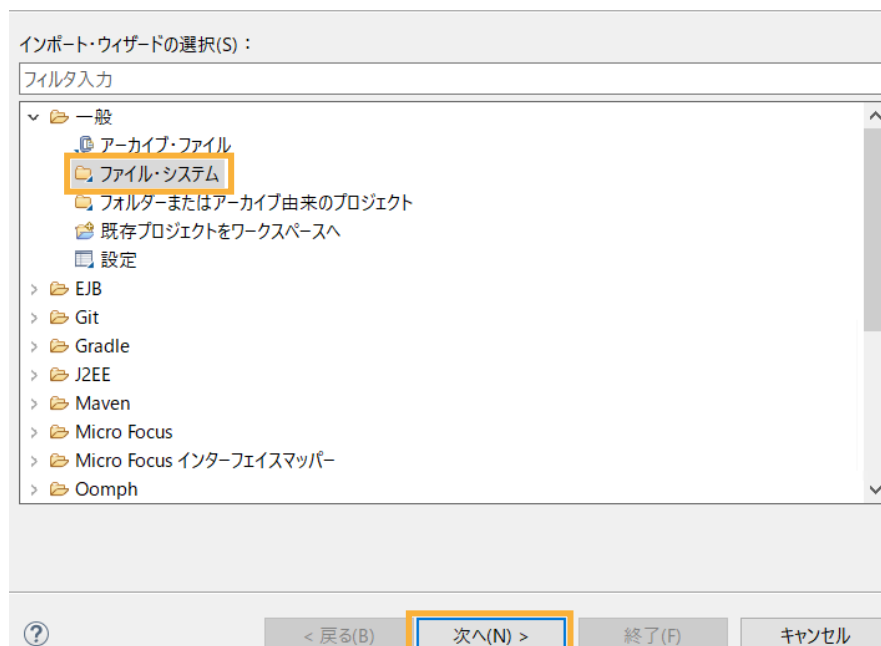
- ① dat フォルダを選択し、マウスの右クリックにてコマンドでコンテキストメニューを開き、[インポート(i)] > [インポート (I)] を選択します。



- ② [一般] > [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ヘリソースをインポートします。



- ③ [参照(R)] をクリックし、チュートリアル用ファイルを展開したフォルダー配下の DAT を指定したうえで、BOOKINFO.DAT にチェックを行い、[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(Y): C:\vc-tutorial-rest02\DAT 参照(R)...

<input checked="" type="checkbox"/> DAT	<input type="checkbox"/> BOOKINFO bk.DAT <input checked="" type="checkbox"/> BOOKINFO.DAT
---	--

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)

インポート先フォルダ(L): RemoteNativeCOBOL/dat 参照(W)...

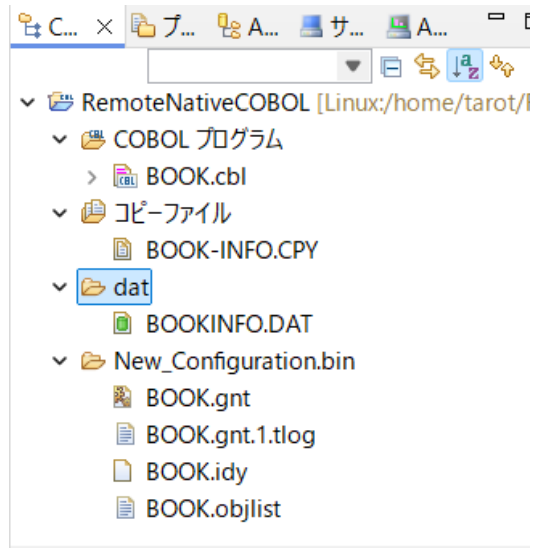
オプション

☐ 警告を出さずに既存リソースを上書き(O)
☐ トップ・レベルのフォルダーを作成(C)

拡張 >>(A)

? < 戻る(B) 次へ(N) > 終了(F) キャンセル

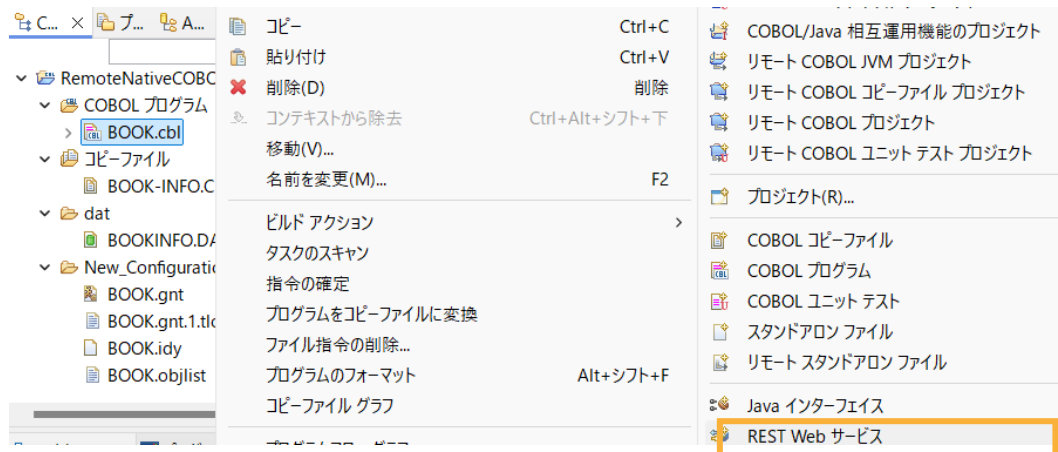
BOOKINFO.DAT が DAT フォルダ配下に表示されます。



3.5 RESTful Web サービスの開発作業

1) RESTful Web サービスのプロファイル作成

- ① RESTful Web サービスとして利用するビジネスロジックを処理するプログラム「BOOK.cbl」を右クリックし、コンテキストメニューから [新規作成(N)] > [REST Web サービス] を選択します。



- ② REST Web サービスの新規作成ウィザードが表示されます。[Web サービス名] 欄に “BOOKREST” を指定します。[マッピング] 欄は「無し」を選択、[マップするプログラム] 欄には「RemoteNativeCOBOL/BOOK.cbl」が選択されていることを確認し [終了(F)] ボタンをクリックします。

REST Web サービスの新規作成

このページで REST Web サービスを新規作成します



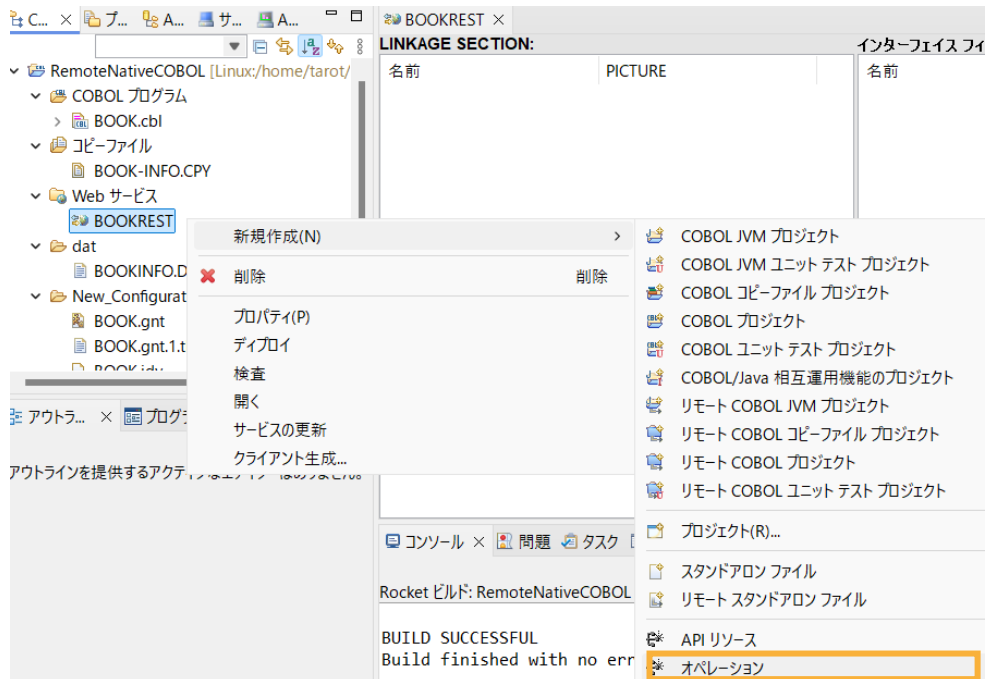
Web サービス名: BOOKREST

マッピング: ☐ デフォルト ☒ 無し

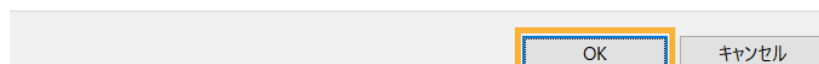
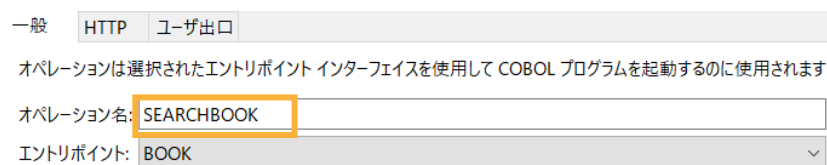
マップするプログラム: RemoteNativeCOBOL/BOOK.cbl

2) 書籍データ検索機能オペレーションの作成

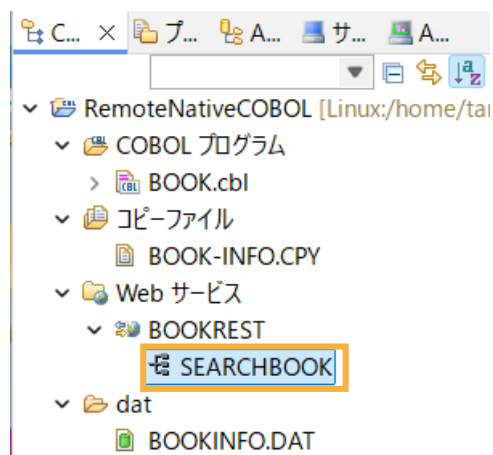
- ① 1) で作成した Web サービスプロファイル「BOOKREST」を右クリックし、コンテキストメニューから [新規作成 (N)] > [オペレーション] を選択します。



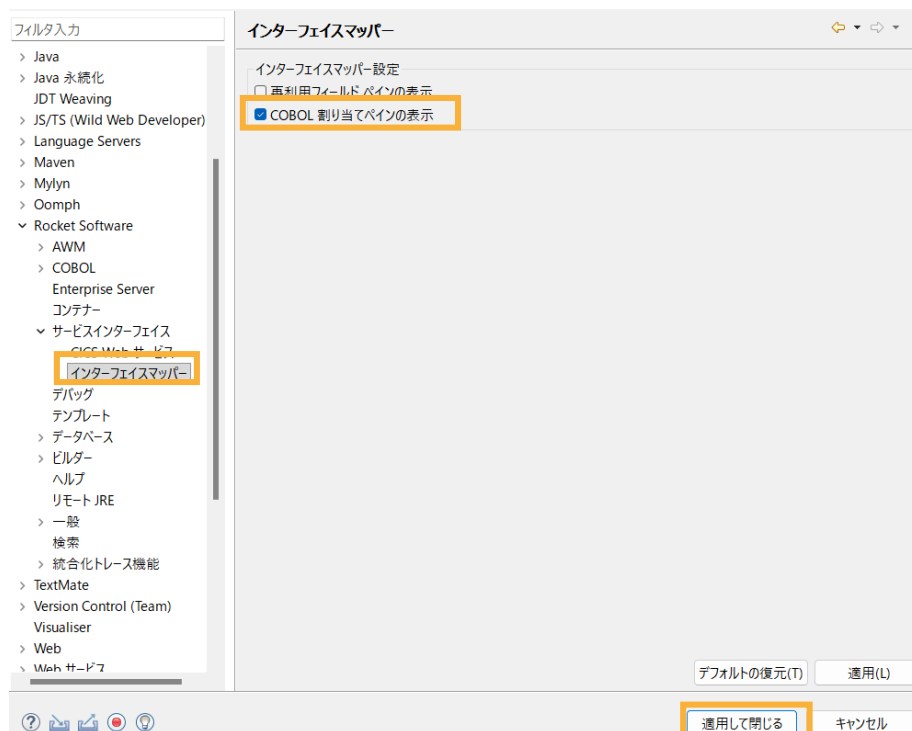
- ② オペレーションプロパティウィンドウが表示されます。[オペレーション名] 欄に “SEARCHBOOK” を入力して、[OK] をクリックします。



- ③ COBOL エクスプローラーより、[BOOKREST] > [SEARCHBOOK] をダブルクリックします。



- ④ Eclipse IDE メニューから [ウィンドウ(W)] -> [設定(P)] を選択し、[Rocket Software] > [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。

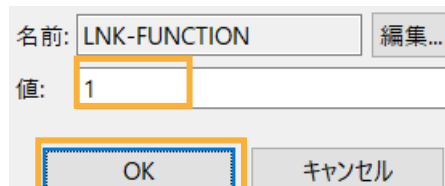


※Preference Recorder のダイアログが表示されたら [キャンセル] を選択してください。

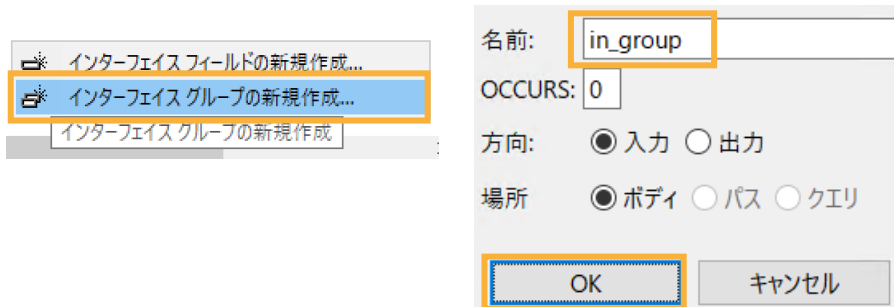
- ⑤ LINKAGE SECTION の COBOL の変数「LNK-FUNCTION」を [COBOL 割り当て] にドラッグ&ドロップします。



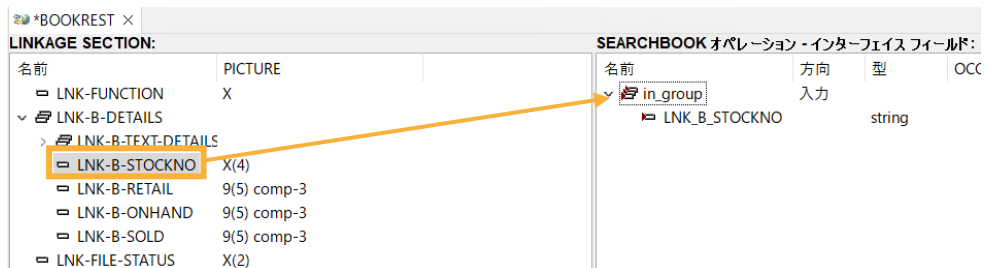
- ⑥ [COBOL 割り当てプロパティ] ダイアログが表示されるので [値] に "1" を設定して [OK] ボタンをクリックします。



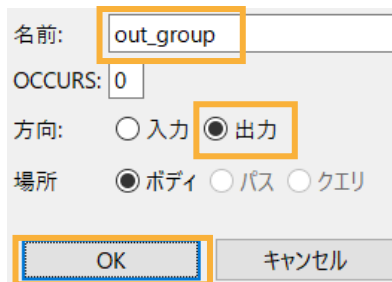
- ⑦ [SEARCHBOOK オペレーション - インターフェイスフィールド] にて右クリックから[インターフェイスグループの新規作成]を選択し、[名前]に“in_group”と入力し、[OK]ボタンをクリックします。



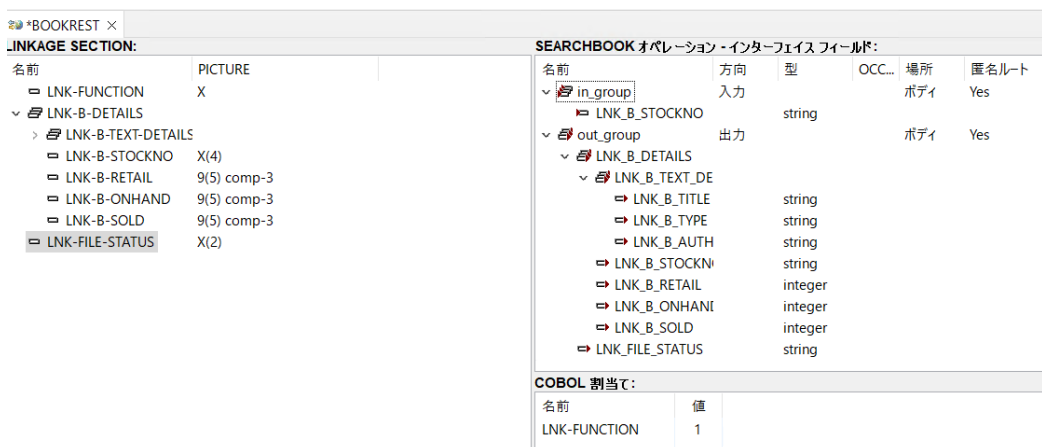
- ⑧ 作成したグループに [LNK-B-DETAILS] > [LNK-B-STOCKNO] をドラッグ&ドロップします。



- ⑨ 同じく [SEARCHBOOK オペレーション - インターフェイスフィールド] にて右クリックから[インターフェイスグループの新規作成]を選択し、[名前]に“out_group”と入力し、[方向]を「出力」に変更し、[OK]ボタンをクリックします。



- ⑩ 作成したグループに「LNK-B-DETAILS」と「LNK-FILE-STATUS」をドラッグ&ドロップします。
最終的には下のイメージのような構成になります。



3) 書籍データ追加機能オペレーションの追加

- ① 再度、Web サービスプロファイル「BOOKREST」を右クリックし、コンテキストメニューから [新規作成(N)] > [オペレーション] を選択します。
- ② オペレーションプロパティウィンドウが表示されます。[オペレーション名] 欄に "ADDBOOK" を入力し、[OK] をクリックします。

一般 HTTP ユーザ出口

オペレーションは選択されたエントリポイント インターフェイスを使用して COBOL プログラムを起動するために使用されます

オペレーション名:

エントリポイント:

- ③ [BOOKREST] > [ADDBOOK] をダブルクリックします。
- ④ 「LNK-FUNCTION」を COBOL 割当てにドラッグ&ドロップし、[値] には "2" を指定します。
- ⑤ 書籍データ検索機能オペレーションの作成と同様の手順で「in_group」を作成し、「LNK-B-DETAILS」をドラッグ&ドロップします。
- ⑥ 書籍データ検索機能オペレーションの作成と同様の手順で「out_group」を作成し[方向]を出力に変更してから、「LNK_FILE_STATUS」をドラッグ&ドロップします。

最終的には、以下のようになります。

BOOKREST

LINKAGE SECTION:

名前	PICTURE
LNK-FUNCTION	X
LNK-B-DETAILS	
LNK-FILE-STATUS	X(2)

ADDBOOK オペレーション - インターフェイス フィールド:

名前	方向	型	OCC...	場所	匿名ルート
in_group	入力			ボディ	Yes
LNK_B_DETAILS					
LNK_B_TEXT_DE					
LNK_B_TITLE		string			
LNK_B_TYPE		string			
LNK_B_AUTH		string			
LNK_B_STOCKN		string			
LNK_B_RETAIL		integer			
LNK_B_ONHANI		integer			
LNK_B_SOLD		integer			
out_group	出力			ボディ	Yes
LNK_FILE_STATUS		string			

COBOL 割当て:

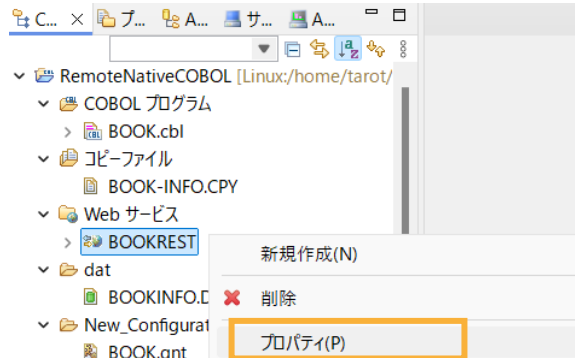
名前	値
LNK-FUNCTION	2

- ⑦ Ctrl + S を押して保存します。

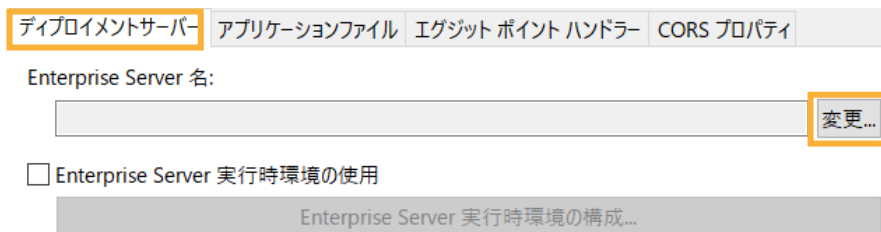
3.6 COBOL アプリケーションを Enterprise Server ヘッドプロイ

1) Enterprise Server へのディプロイ情報を指定

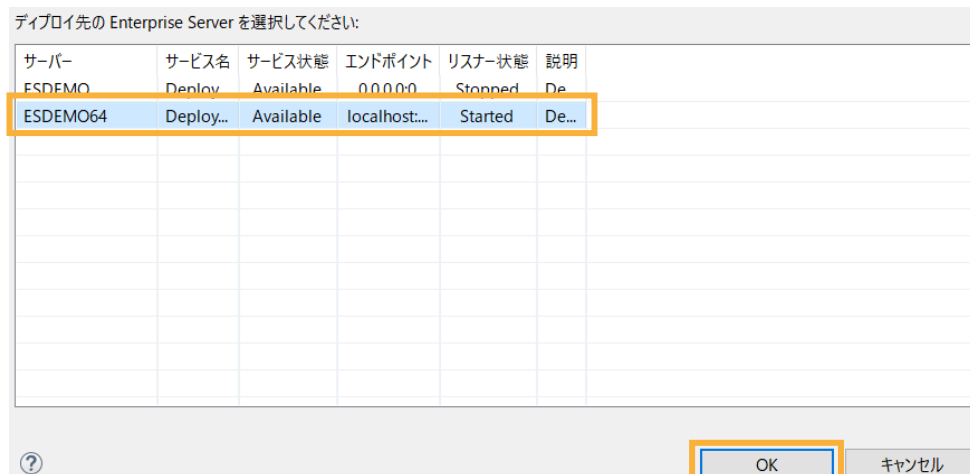
- ① COBOL エクスプローラーにて追加した Web サービス「BOOKREST」を右クリックし、コンテキストメニューから [プロパティ(P)] を選択します。



- ② [ディプロイメントサーバー] タブを選択し、[変更] ボタンをクリックします。

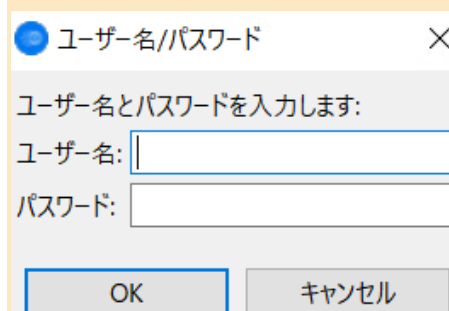


- ③ 起動済みの Enterprise Server 「ESDEMO64」を選択し、[OK] ボタンをクリックします。



注意)

[変更] をクリックした際に、以下のダイアログが表示された際は、ユーザー名/パスワードには、ESCWA 画面で入力した mfsecretadmin の情報を入力してください。



上記にかかわらず、Enterprise Server 名に何も表示されない場合は、以下の手順のいずれかを実施してください。

1) ESCWA セキュリティを無効化する

\$COBDIR/bin/DisableESDefaultSecurity.sh を実行

2) \$COBDIR/etc/mf-client.dat を編集する

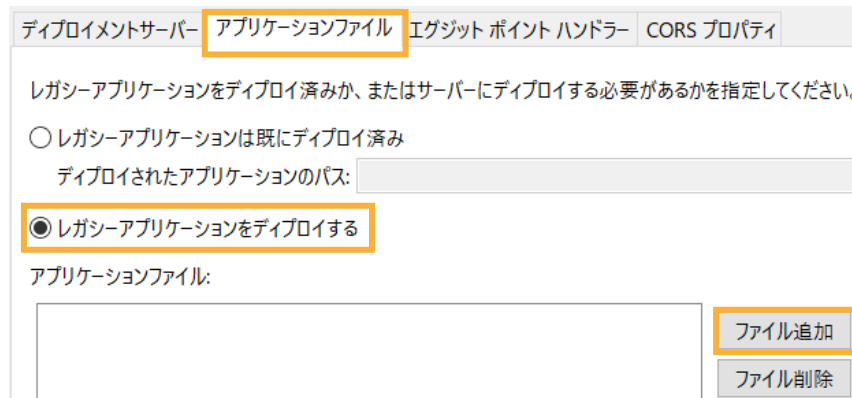
[mldap] セクションに username=SYSAD と userpassword=xxxxxxx 項目を追加

いずれも、mfds を再起動してください。

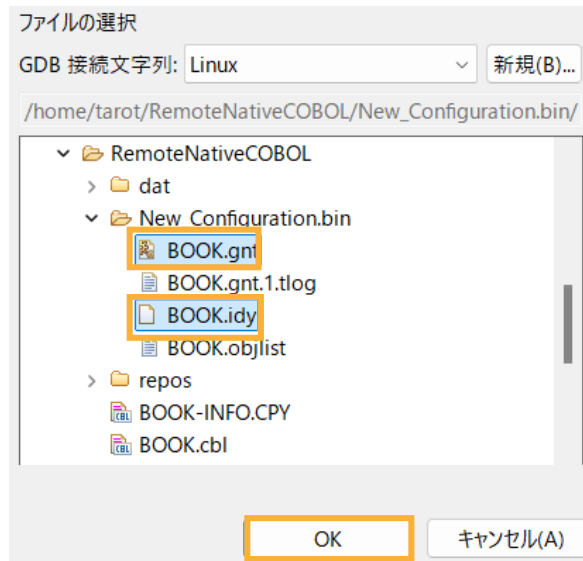
④ [ディプロイする場合はユーザー名/パスワードが必要] にチェックします。



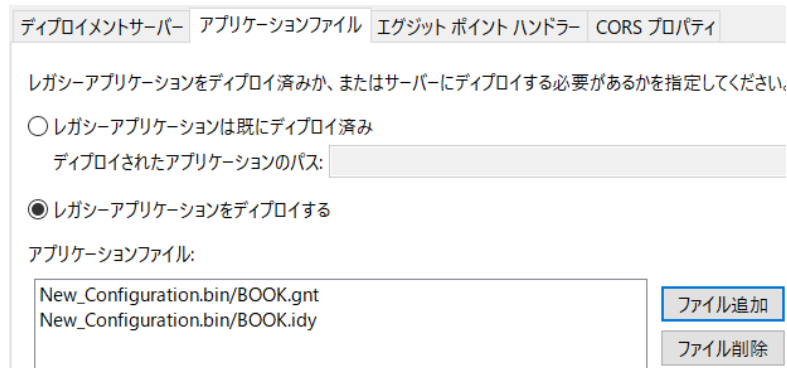
⑤ [アプリケーションファイル] タブを選択し、「レガシーアプリケーションをディプロイする」を選択したうえで、[ファイル追加] をクリックします。



プロジェクトディレクトリ配下の「New_Configuration.bin」に生成された「BOOK.gnt」および「BOOK.idy」を選択し、[OK] ボタンをクリックします。



選択したファイルが追加されます。



2) オリジン間リソース共有（CORS）を許可

デフォルトではオリジン間リソース共有は許可されていません。もしこれに関するエラーが発生する場合、許可設定を行います。

- ① [CORS プロパティ]タブを選択し、運用用途に合わせた設定を行います。ここでは「アクセス制限チェックなしでクロスオリジンでリソースのフェッチを許可する(*)」を選択したうえで、[OK] をクリックします。

ディプロイメントサーバー アプリケーションファイル エグジット ポイント ハンドラー **CORS プロパティ**

オリジンを許可する

☐ クロスオリジン要求を許可しない (null)

☒ アクセス制御チェックなしでクロスオリジンでリソースのフェッチを許可する (*)

☐ 特定のオリジンからクロスオリジンでリソースのフェッチを許可する

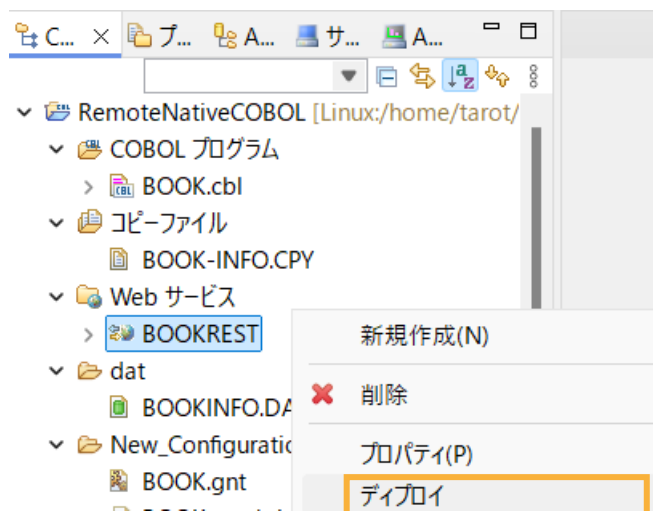
Allowed Origin:

公開するヘッダー

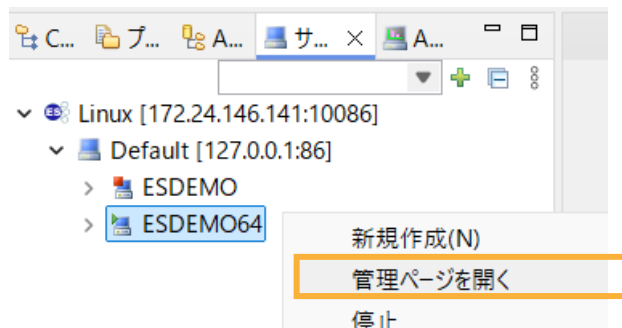
☐ 資格情報を許可する

3) RESTful Web サービスを Enterprise Server ヘディプロイ

- ① COBOL エクスプローラーにて作成した Web サービス「BOOKREST」を右クリックし、コンテキストメニューから [ディプロイ] を選択します。



- ② Eclipse IDE 上からサーバーエクスプローラーを選択します。
認証画面が表示された場合は、3.3 と同じ情報を入力します。
- ③ 「ESDEMO64」を選択し、右クリックでコンテキストメニューを表示して、「管理ページを開く」をクリックし、ブラウザで管理画面を開きます。



- ④ [一般]メニューから[サービス]をクリックします。

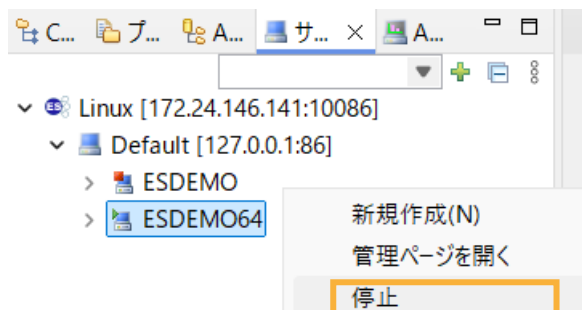


画面を下にスクロールしていくと最下行にデプロイした RESTful Web サービスが表示されます。

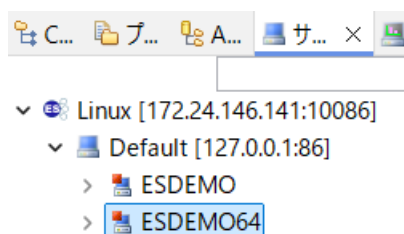
📁	▼ /temppath/BOOKREST/1.0				
🔗	#SEARCHBOOK	Available	Web Services and J2EE@CP1	/temppath/...	MFRHJSON
🔗	#ADDBOOK	Available	Web Services and J2EE@CP1	/temppath/...	MFRHJSON

3.7 Enterprise Server インスタンスへの環境設定と有効化

- 1) 変更内容を反映させるために ESDEMO64 の再起動が必要となります。このため、まずは停止を行います。いったん、Eclipse IDE に戻り、[サーバーエクスプローラー] ビューの [ESDEMO64] を選択し、マウスの右クリックにてコンテキストメニューを開き、[停止] をクリックします。

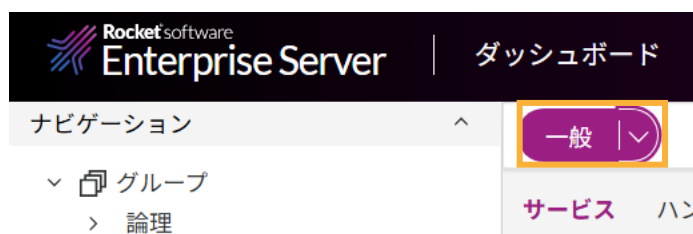


停止後、アイコンが赤色になったことを確認して、ESCWA 管理画面に戻ってください。



- 2) データファイルへの環境変数設定

- ① [一般] をクリックします。



- ② [構成情報] に以下の情報を設定したうえで、画面上部の [適用] をクリックします。

[ES-Environment]

BOOKINFO=/home/tarot/RemoteNativeCOBOL/dat/BOOKINFO.DAT

一般 | モニター | ログ

一般的なプロパティ **適用** 削除

名前*

共有メモリ ページ数 ページ数(4k): 共有メモリ クッション ページ数(4k):

SEP数 コンソール ログ サイズ k

☐ ローカル コンソールを表示 ☐ 動的デバッグを許可

☒ 64ビット作業モード ☐ 以前のログを削除

☐ システム起動時に開始する

自動実行制御用 Enterprise Server 認証情報:

ユーザーID パスワード

追加設定

構成情報

3) リスナーポートの設定

- ① [一般] > [リスナー] をクリックします。

メインフレーム セキュリティ

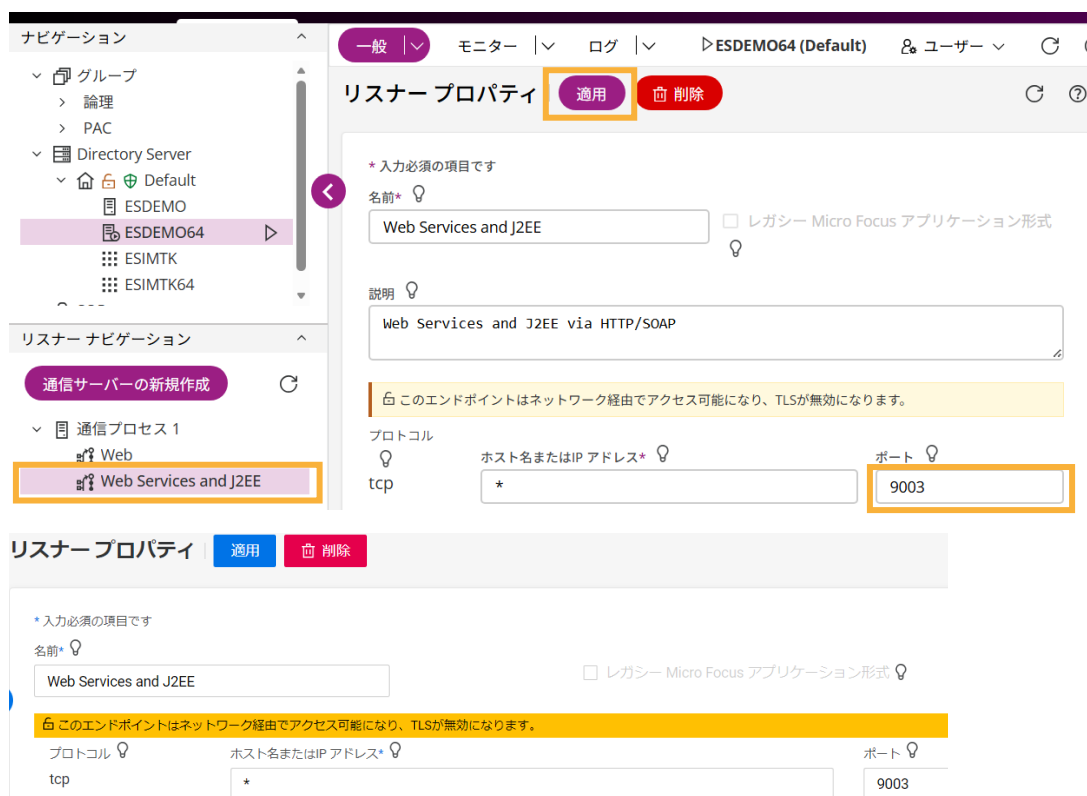
一般 | モニター | ログ

一般的 プロパティ
コントロール
検証
リスナー
サービス

開始

名前*

- ② 左下より [Web Services and J2EE] をクリックし、右側の [ポート] に “9003” を入力したうえで、[適用] をクリックします。



ナビゲーション

- グループ
 - 論理
 - PAC
- Directory Server
 - Default
 - ESDEMO
 - ESDEMO64**
 - ESIMTK
 - ESIMTK64

リスナー ナビゲーション

通信サーバーの新規作成

通信プロセス 1

- Web
- Web Services and J2EE**

リスナー プロパティ

一般 | モニター | ログ | ESDEMO64 (Default) | ユーザー | 適用 | 削除

* 入力必須の項目です

名前* Web Services and J2EE

説明 Web Services and J2EE via HTTP/SOAP

このエンドポイントはネットワーク経由でアクセス可能になり、TLSが無効になります。

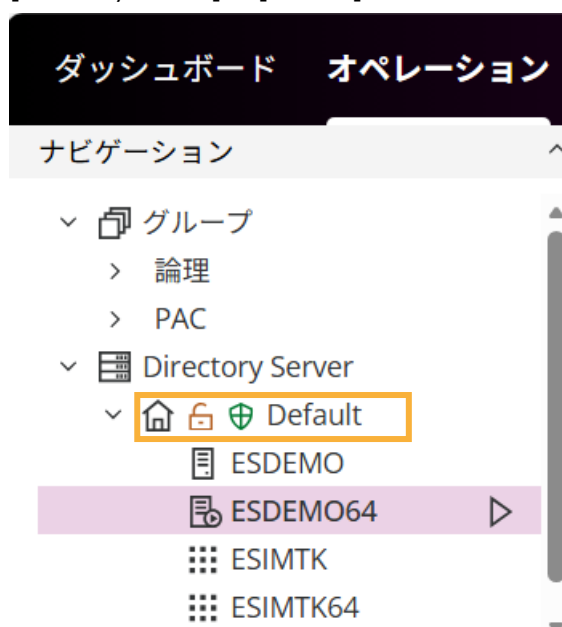
プロトコル tcp

ホスト名またはIP アドレス* *

ポート 9003

4) 不要ログの停止

- ① [Directory Server] > [Default] をクリックします。

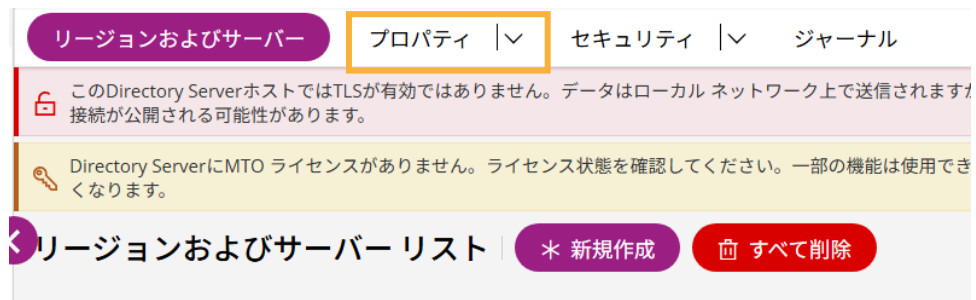


ダッシュボード オペレーション

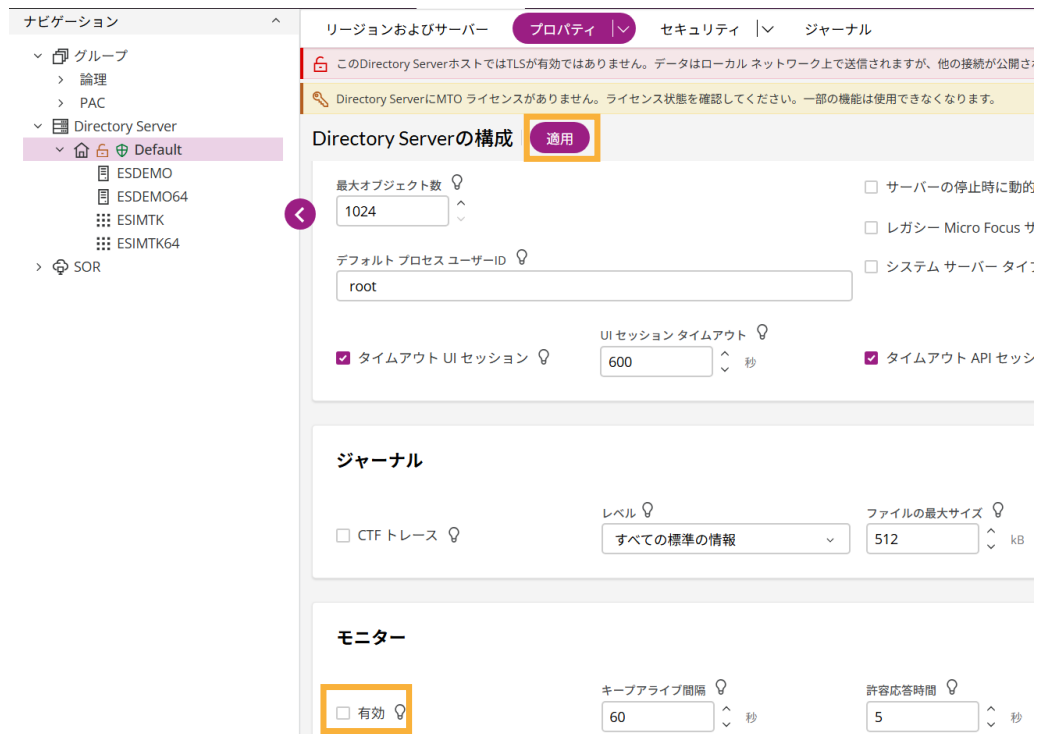
ナビゲーション

- グループ
 - 論理
 - PAC
- Directory Server
 - Default**
 - ESDEMO
 - ESDEMO64**
 - ESIMTK
 - ESIMTK64

- ② [プロパティ] をクリックします。

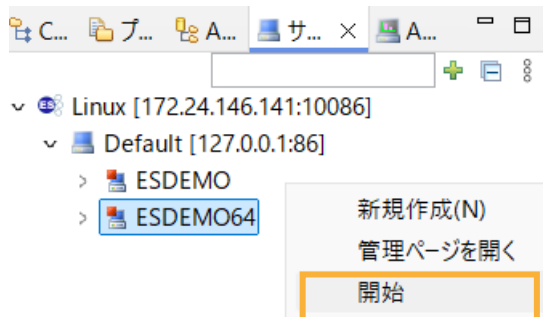


- ③ 一番下にスクロールし、[モニター] 欄の [有効] のチェックを外し、[適用] をクリックします。



5) Directory Server の開始

- ④ Eclipse IDE に戻り、[サーバーエクスプローラー] ビューの [ESDEMO64] を右クリックにてコンテキストメニューを開き、[開始] をクリックします。

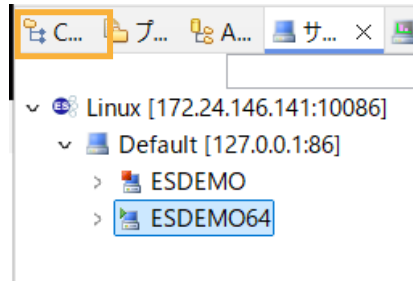


3.8 RESTful Web サービスのテスト

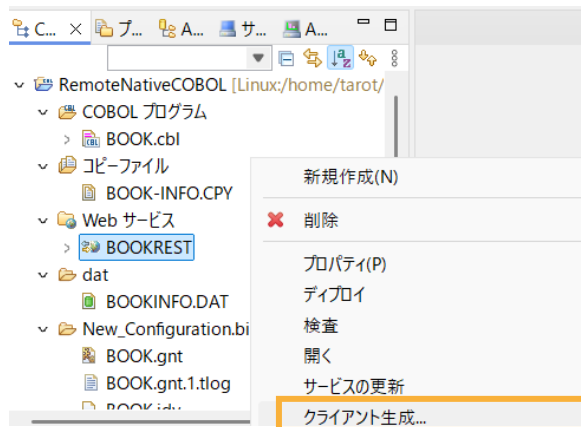
1) テストクライアントアプリケーションによる確認

製品には、登録したサービスのテストクライアントアプリケーションを自動生成する機能が搭載されています。こちらを利用して、書籍データ検索機能オペレーションのテストを行います。

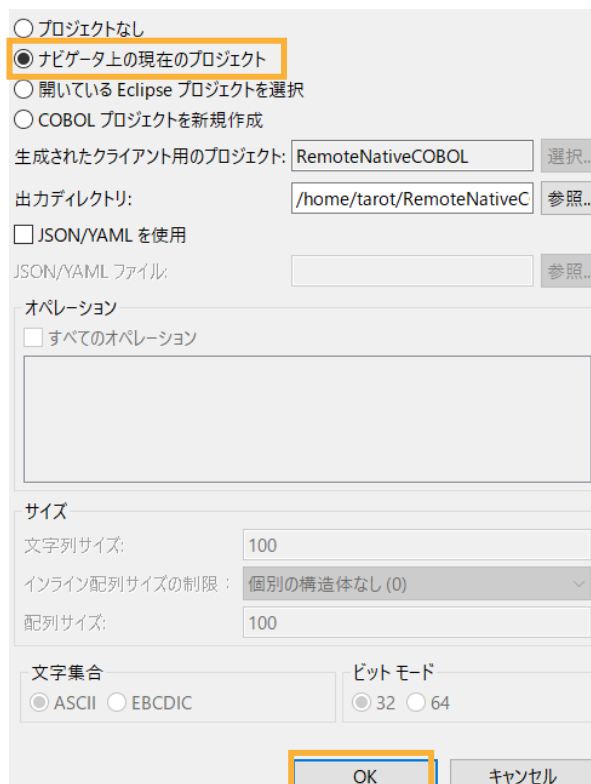
- ① Eclipse IDE に戻り、[COBOL エクスプローラー] をクリックします。



- ② [BOOKREST] を選択し、マウスの右クリックにてコンテキストメニューを開き、[クライアント生成] をクリックします。

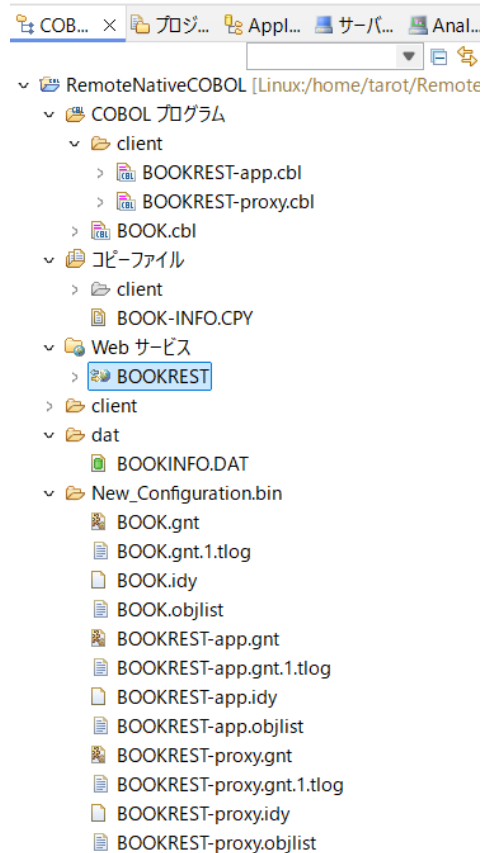


- ③ [ナビゲーター上の現在のプロジェクト] を選択し、[OK] をクリックします。



クライアントプログラムが生成され、自動的にビルドされます。

BOOKREST-app.cbl がテストクライアントプログラムになります。



- ④ リモートサーバーにターミナル上でログインを行い、以下のコマンドを実行します。

```
. /opt/rocketsoftware/VisualCOBOL/bin/cobsetenv
```

```
$ . /opt/rocketsoftware/VisualCOBOL/bin/cobsetenv
COBDIR set to /opt/rocketsoftware/VisualCOBOL
$
```

- ⑤ リモートサーバー上の RemoteNativeCOBOL/New_Configuration.bin ディレクトリにターミナル上で移動します。

- ⑥ 以下のコマンドを実行します。

```
cobrun BOOKREST-app.gnt
```

次の入力を行ってください。

「Service Address (Enter = http://localhost:9003):」では、そのまま Enter キーを押します

「Supplemental Query String (optional):」では、そのまま Enter キーを押します

「Username (optional):」では、そのまま Enter キーを押します

「Password (optional):」では、そのまま Enter キーを押します

「Operation (1 = SEARCHBOOK, 2 = ADDBOOK):」では “1” を入力します

「.LNK_B_STOCKNO:」では、“1111” を入力します

```
Supplemental Query String (optional):
Username (optional):
Password (optional):
Operation (1 = SEARCHBOOK, 2 = ADDBOOK): 1
Body Parameters:
:
.LNK_B_STOCKNO: 1111
:
.LNK_B_DETAILS:
..LNK_B_TEXT_DETAILS:
...LNK_B_TITLE: LORD OF THE RINGS
...LNK_B_TYPE: FANTASY
...LNK_B_AUTHOR: TOLKIEN
..LNK_B_STOCKNO: 1111
..LNK_B_RETAIL: 01500
..LNK_B_ONHAND: 04000
..LNK_B_SOLD: 03444
.LNK_FILE_STATUS: 00
```

書籍データが戻されていることが確認できます。

2) curl コマンドによる確認

- ⑦ Windows のスタートメニューより、コマンドプロンプトを起動し、サンプルファイルを解凍したフォルダーに移動します。
- ⑧ 検索機能をテストするため、以下のコマンドをコマンドプロンプト上で実行します。

```
curl http://IP アドレス:9003/temppath/BOOKREST/1.0/SEARCHBOOK -d
@json¥search1111.txt
```

注意)

IP アドレスはリモートサーバーのホスト名やアドレスなどに修正してください。

```
C:¥vc-tutorial-rest02>curl -XPOST http://172.24.146.141:9003/temppath/BOOKREST/1.0/SEARCHBOOK -d @json¥search1111.txt
{
  "LNK_B_DETAILS" :
  {
    "LNK_B_TEXT_DETAILS" :
    {
      "LNK_B_TITLE" : "LORD OF THE RINGS",
      "LNK_B_TYPE" : "FANTASY",
      "LNK_B_AUTHOR" : "TOLKIEN"
    },
    "LNK_B_STOCKNO" : "1111",
    "LNK_B_RETAIL" : 1500,
    "LNK_B_ONHAND" : 4000,
    "LNK_B_SOLD" : 3444
  },
  "LNK_FILE_STATUS" : "00"
}
C:¥vc-tutorial-rest02>
```

- ⑨ 追加機能をテストするため、以下のコマンドをコマンドプロンプト上で実行します。

まずは、既存データがない事を検索機能で確認します。

```
curl http://IP アドレス:9003/temppath/BOOKREST/1.0/SEARCHBOOK -d
```

```
@json¥search9999.txt
```

```
C:¥vc-tutorial-rest02>curl -XPOST http://172.24.146.141:9003/temppath/BOOKREST/1.0/SEARCHBOOK -d @json¥search9999.txt
{
  "LNK_B_DETAILS" :
  {
    "LNK_B_TEXT_DETAILS" :
    {
      "LNK_B_TITLE" : "*****"
    },
    "LNK_B_TYPE" : "*****",
    "LNK_B_AUTHOR" : "*****"
  },
  "LNK_B_STOCKNO" : "",
  "LNK_B_RETAIL" : 0,
  "LNK_B_ONHAND" : 0,
  "LNK_B_SOLD" : 0
},
"LNK_FILE_STATUS" : "23"
}
```

続いて、追加機能でデータを追加します。

```
curl http://IP アドレス:9003/temppath/BOOKREST/1.0/ADDBOOK -d @json¥add9999.txt
```

```
C:¥vc-tutorial-rest02>curl -XPOST http://172.24.146.141:9003/temppath/BOOKREST/1.0/ADDBOOK -d @json¥add9999.txt
{
  "LNK_FILE_STATUS" : "00"
}
```

検索機能で、正しく追加されたことを確認します。

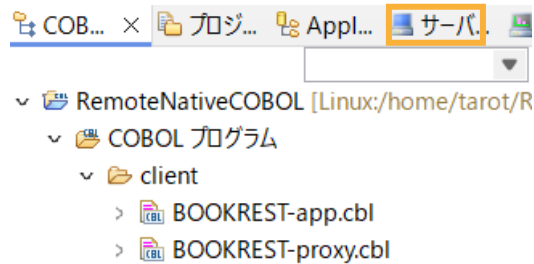
```
curl http://IP アドレス:9003/temppath/BOOKREST/1.0/SEARCHBOOK -d
@json¥search9999.txt
```

```
C:¥vc-tutorial-rest02>curl -XPOST http://172.24.146.141:9003/temppath/BOOKREST/1.0/SEARCHBOOK -d @json¥search9999.txt
{
  "LNK_B_DETAILS" :
  {
    "LNK_B_TEXT_DETAILS" :
    {
      "LNK_B_TITLE" : "ALICE'S ADVENTURES IN WONDERLAND",
      "LNK_B_TYPE" : "FANTASY",
      "LNK_B_AUTHOR" : "LEWIS CARROLL"
    },
    "LNK_B_STOCKNO" : "9999",
    "LNK_B_RETAIL" : 100,
    "LNK_B_ONHAND" : 200,
    "LNK_B_SOLD" : 300
  },
  "LNK_FILE_STATUS" : "00"
}
```

3.9 RESTful Web サービスのデバッグ

1) サーバー側のデバッグ設定

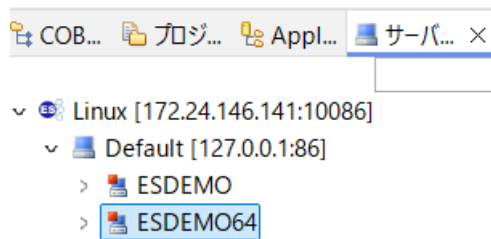
- ① Eclipse IDE に戻り、[サーバーエクスプローラー] をクリックします。



- ② [ESDEMO64] を選択し、マウスの右クリックにてコンテキストメニューを開き、[停止] をクリックします。

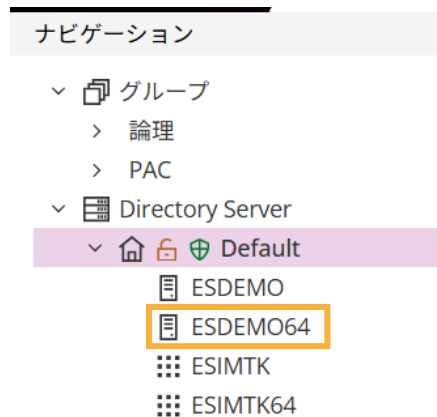


停止すると、赤色のアイコンが表示されます。



- ③ ESCWA 画面に戻り、左下より [ESDEMO64] をクリックします。

この時、認証画面が再度表示されることがありますので、前回と同じ情報を入力してください。



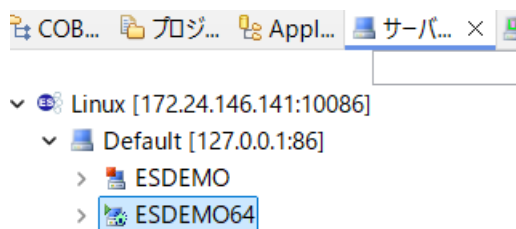
- ④ [動的デバッグを許可] にチェックを行い、[適用] をクリックします。



- ⑤ Eclipse IDE に戻り、[ESDEMO64] を選択し、マウスの右クリックにてコンテキストメニューを開き、[開始] をクリックします。

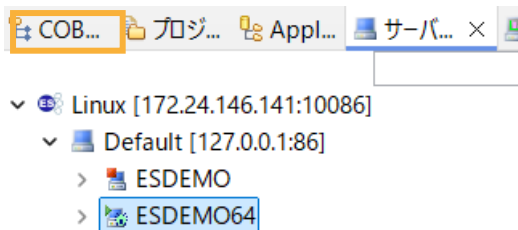


開始すると、緑のアイコンが表示されます。

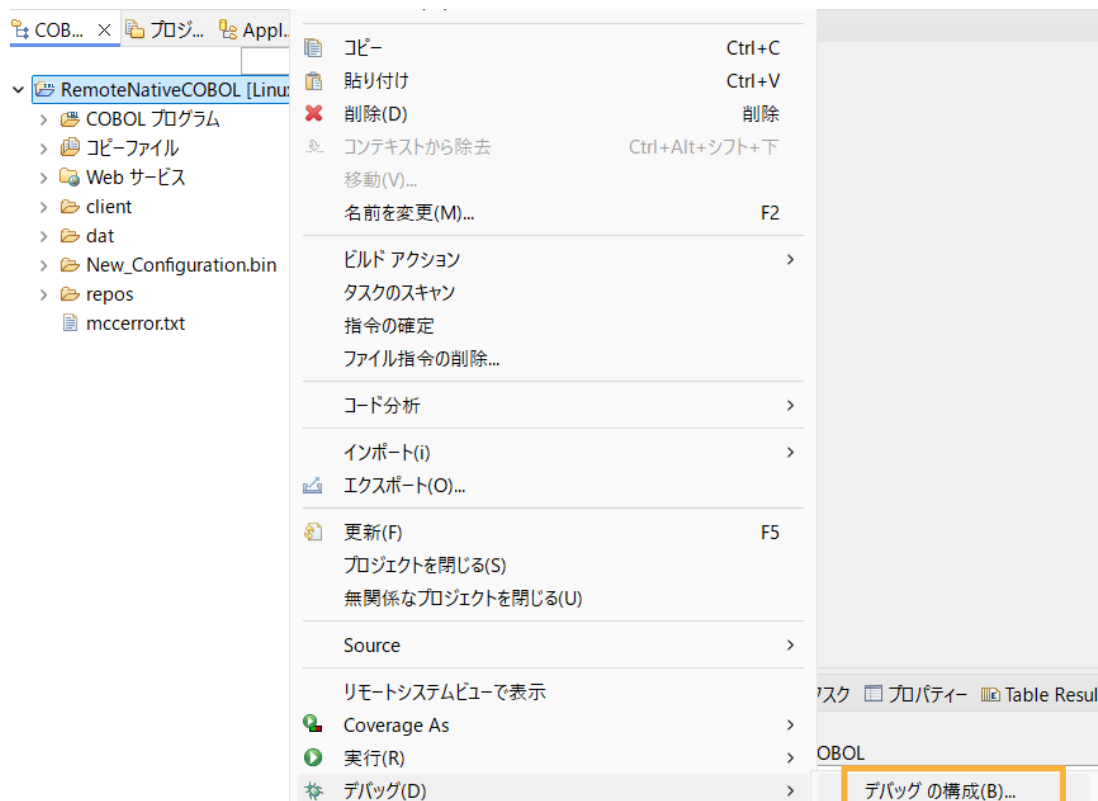


2) デバッガーのアタッチ

- ① [COBOL エクスプローラー] をクリックします。



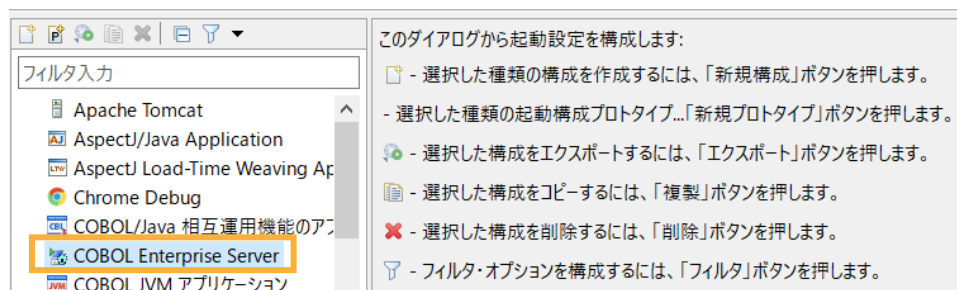
- ② RemoteNativeCOBOL プロジェクトを選択し、マウスの右クリックにてコンテキストメニューを表示した上で、[デバッグ(D)] > [デバッグの構成(B)] をクリックします。



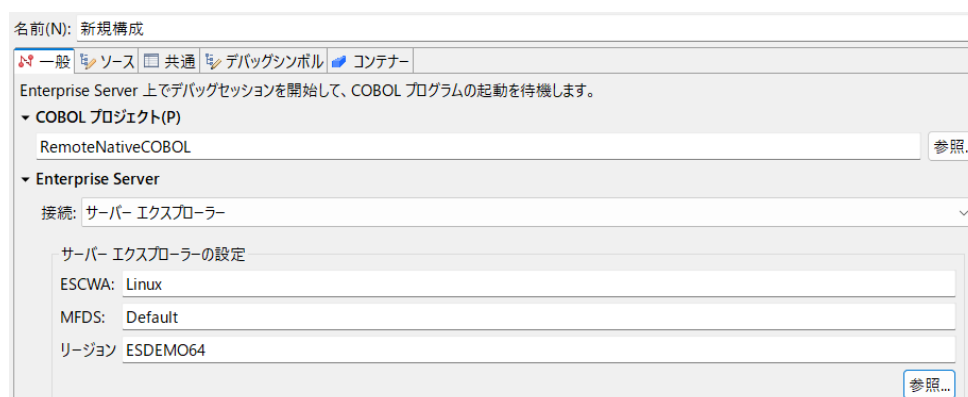
- ③ 画面左側より、[COBOL Enterprise Server] をダブルクリックします。

構成の作成、管理、および実行

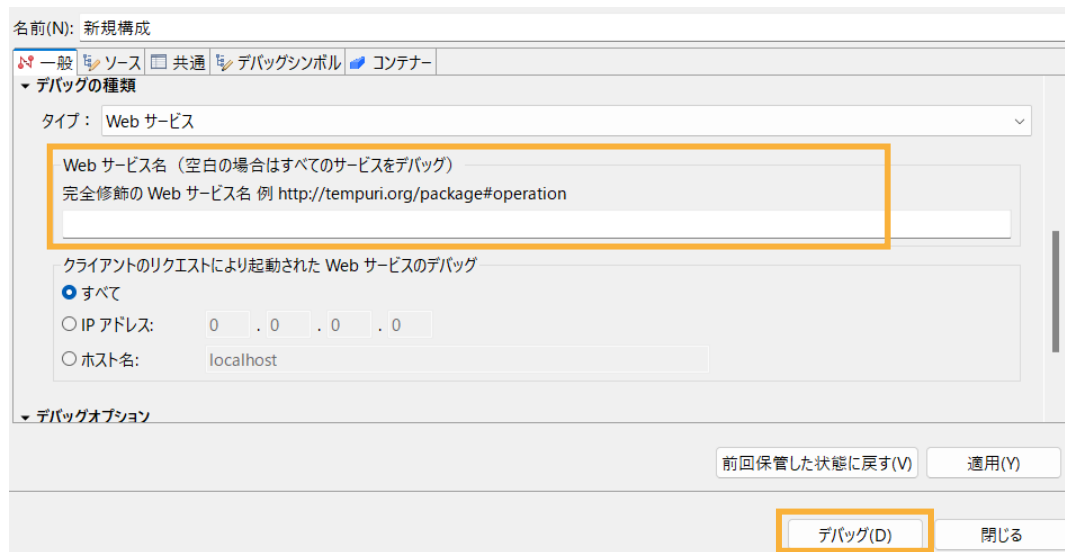
Enterprise Server アプリケーションへの接続とデバッグ



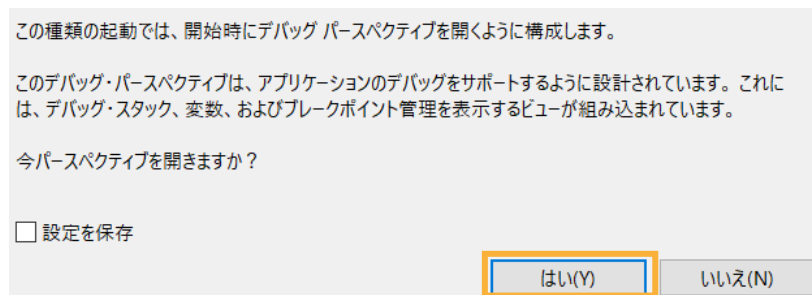
- ④ [Enterprise Server] セクションの [参照] をクリックして、Linux > ESDemo64 を選択します。選択後は、以下のように表示されます。



- ⑤ 画面を下にスクロールを行い、[デバッグの種類] セクションで [Web サービス] タブを選択し、[Web サービス名] が空欄になっていることを確認したうえで、[デバッグ(D)] をクリックします。



以下のダイアログが表示された場合は、[はい(Y)] をクリックします。



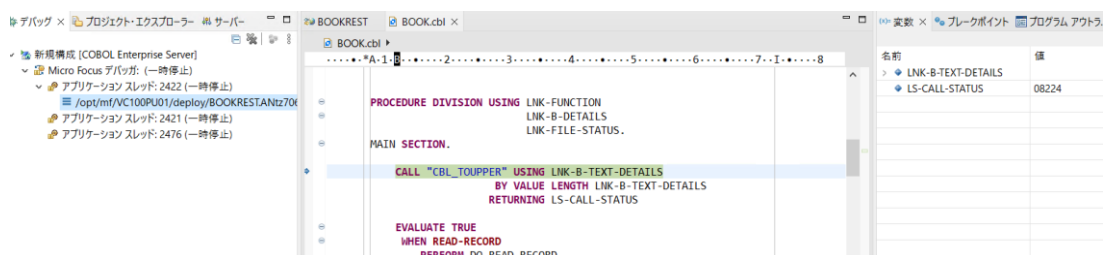
- ⑥ さきほどの curl コマンドを用いたリクエストを行います。以下のコマンドをコマンドプロンプト上で実行します。

```
curl http://IP アドレス:9003/temp-path/BOOKREST/1.0/SEARCHBOOK -d
@json¥search1111.txt
```

「IP アドレス」は、環境に合わせて変更してください。

- ⑦ Eclipse IDE に戻ると、デバッガーが起動し、プログラムが停止しています。

通常のコンソールアプリケーションのデバッグと同様、[実行] メニュー配下の [再開] や [ステップイン] などといった操作が利用できます。



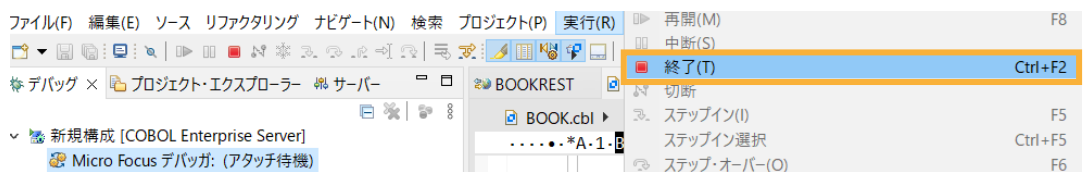
[ステップイン] にて、1 行毎にプログラムを進めることができます。

[再開] など、デバッグが終了すると、コマンドラインに結果が戻されます。


```
C:\vc-tutorial-rest02>curl http://172.24.146.141:9003/tempopath/BOOKREST/1.0/SEARCHBOOK -d @json¥search1111.txt
{
  "LNK_B_DETAILS" :
  {
    "LNK_B_TEXT_DETAILS" :
    {
      "LNK_B_TITLE" : "LORD OF THE RINGS",
      "LNK_B_TYPE" : "FANTASY",
      "LNK_B_AUTHOR" : "TOLKIEN"
    },
    "LNK_B_STOCKNO" : "1111",
    "LNK_B_RETAIL" : 1500,
    "LNK_B_ONHAND" : 4000,
    "LNK_B_SOLD" : 3444
  },
  "LNK_FILE_STATUS" : "00"
}
```

C:\vc-tutorial-rest02>

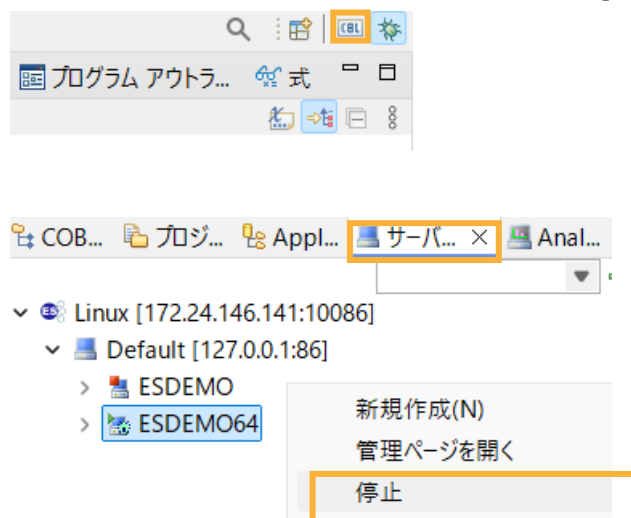
- ⑧ デバッグの終了を行う場合は、[実行(R)] > [終了(T)] をクリックします。



3.10 サーバーの停止

1) Enterprise Server インスタンスの停止

- ① 画面右上のアイコンより、「COBOL エクスプローラー」ビューに切り替え、「サーバーエクスプローラー」に切り替えうえで、ESDEMO64 上で右クリックし、コンテキストメニューから [停止] を選択します。



- 2) escwa, mfdss を実行しターミナル上で、以下のコマンドを実行します。

コマンド引数の情報は、先に取得・使用していた認証情報です。

> Enterprise Server Common Web Administration (ESCWA) サービスの停止

escwa -p SYSAD 88tudJ8a

```
# escwa -p SYSAD 88tudJ8a
2025-11-21 10:50:08.003 Loaded COBOL Run Time Environment Extension
2025-11-21 10:50:08.003 New thread high-water mark: 1 threads are now running
2025-11-21 10:50:08.004 MFCS server "ESCWA" running as process 3273
2025-11-21 10:50:08.004 GK-OS version 2.13.0
2025-11-21 10:50:08.004 GK-Utility version 2.12.3
2025-11-21 10:50:08.004 GkCobGetFuncAddr: 4
2025-11-21 10:50:08.004 ES Common Web Administration version: 7.1.0
2025-11-21 10:50:08.004 Copyright (C) 2019-2025 Rocket Software, Inc. or its affiliates.
All rights reserved.
2025-11-21 10:50:08.004 /opt/rocketsoftware/VisualCOBOL/etc
2025-11-21 10:50:08.006 Shutting down ESCWA instance at: http://localhost:10086
2025-11-21 10:50:08.028 Shutdown request successful.
```

> Directory Server の停止

mfdss /s 2 SYSAD 6YaafasP

```
# mfdss /s 2 SYSAD 88tudJ8a
Processing -s option...
Copyright (C) 1984-2025 Rocket Software, Inc. or its affiliates. All rights reserved.
Enterprise Directory Server daemon: Version 1.31.25
Request sent...
#
```

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。