Micro Focus Visual COBOL チュートリアル

JCA による JBOSS EAP 連携の設定と開発

1. 目的

Micro Focus Visual COBOL に付属する COBOL 専用のアプリケーションサーバー「Enterprise Server」は、ネイティブにコンパイルした COBOL のビジネスロジックを EJB として再利用し、J2EE クライアントから呼び出す機能を提供しています。EJB コンポーネントとして呼び出しを行う場合、Java アプリケーションサーバー上の J2EE クライアントは JCA の仕様にもとづいたリクエストを Enterprise Server に渡し、 COBOL のビジネスロジックが処理をして結果を返します。また、Enterprise Server は、XA に準拠しているので同じく XA に準拠しているデータベースや他のシステムと協調してトランザクション処理を行うことができます。

Micro Focus Visual COBOL の Linux/UNIX 版には、Linux/UNIX 環境へインストールし、リモート接続を可能にする Development Hub および開発クライアントとして Windows 環境へインストールする Eclipse 版のライセンスが提供されます。これにより、Windows 上の Eclipse で開発作業を行い、Linux/UNIX 上のソースコードを直接編集し、コンパイルするリモート開発機能が利用できます。
通常、Pro*COBOL を使った Oracle 連携アプリケーションの開発を行う場合、ソースコードをプリコンパイルし、生成された COBOL のソースコードを COBOL コンパイラでコンパイルするという 2 つのステップが必要ですが、Visual COBOL は Pro*COBOL を利用しプリコンパイルからコンパイルまでワンステップで行う COBSQL という技術を提供しています。

このドキュメントでは Red Hat Linux 上の JBOSS EAP と Enterprise Server を JCA による連携を行い、Enterprise Server にディプロイする COBOL アプリケーションは、Oracle データベースを利用してトランザクション連携する方法を説明します。

2. 前提条件

本チュートリアルは、下記の環境を前提に作成されています。サポートしているプラットフォームであれば他の Linux/UNIX でも利用可能です。

- アプリケーションサーバー側 ソフトウェア

OS	Red Hat Enterprise Linux Server 8.2（64bit）
COBOL 開発環境製品	Micro Focus Visual COBOL 8.0 Development Hub
AP サーバー製品	Red Hat JBoss EAP 7.4.6 (CP6 適用)
Oracle DBMS 製品	Oracle 19c クライアント（19.8.0.0.0） （Oracle Pro* COBOL 含む）
Oracle JDK	Oracle JDK 1.8（x64）

- 開発クライアント ソフトウェア

OS	Windows Server 2018 Standard Edition（64bit）
COBOL 開発環境製品	Micro Focus Visual COBOL 8.0 for Eclipse

- データベースサーバー ソフトウェア

OS	Oracle Linux 7.9（64bit）
Oracle DBMS 製品	Oracle 19c（19.3.0.0.0） ※事前に Oracle 提供の Scot のサンプルスキーマとデータが設定済み

- チュートリアル用サンプルプログラム

  下記のリンクから事前にチュートリアル用のサンプルファイルをダウンロードして、任意のフォルダに解凍しておいてください。

  サンプルプログラムのダウンロード

内容

3. チュートリアル手順の概要

3.1. COBOL 環境変数の設定

インストールした製品を COBOL 実行環境に設定するため環境変数を設定します。製品ディレクトリの bin ディレクトリに cobsetenv が用意されていますので、これを一般ユーザーで実行します。

コマンド例）. /opt/microfocus/VisualCOBOL/bin/cobsetenv

実行すると環境変数 COBDIR にインストールした製品のパスが設定されます。

3.2. Oracle 関連の設定

1) Oracle 用オプションファイルの作成

Oracle のライブラリをリンクするためにオプションファイルを生成します。この作業を行うためには Oracle 関連の環境変数が適切に設定されている必要があります。

① set_cobopt_oracle の実行

```
$ $COBDIR/src/oracle/set_cobopt_oracle
Set COBOPT to /home/tarot/cobopt.ora before starting the RDO daemon.
Ensure that you specify both the main entry point name and the output name when linking your user application.
From the command-line, you can do this by passing entry_point -o output_name to cob.
```

3.3. Oracle 用 XA スイッチモジュールの作成

トランザクション処理を伴うデータベース I/O を Enterprise Server 経由で行うには XA スイッチモジュール経由でデータベースと接続することになります。このチュートリアルでは Oracle を使用するので Oracle 用の XA スイッチモジュールを root ユーザーで作成します。

1) XA リソースのコピー

ビルドを行うため、インストールディレクトリ配下の $COBDIR/src/enterpriseserver/xa をディレクトリごと書き込み権限があるパスへコピーします。

コピー元パス例：$COBDIR/src/enterpriseserver/xa

コピー先パス例：/home/tarot/xa

$ cp $COBDIR/src/enterpriseserver/xa/* /home/tarot/xa

2) XA スイッチモジュールのビルド準備

生成する環境の設定を行います。

① COBOL 作業モードの設定

接続するデータベースのビット数に合わせた数値を指定します。XA スイッチモジュールはこの設定値に沿って生成されます。 cobmode コマンドまたは環境変数 COBMODE を使用して設定します。

64 ビット設定例）export COBMODE=64

3) XA スイッチモジュールのビルド実行

① 書き込み権限のあるコピー先パスへ移動します。

コマンド例）cd /home/tarot/xa

② Oracle を使用する場合は下記コマンドを実行し、XA スイッチモジュールを生成します。

コマンド）./build ora

cobmode=64 の場合、下記の 2 ファイルが生成されます。

ESORAXA64_D.so	動的
ESORAXA64.so	静的

cobmode=32 の場合、下記の 2 ファイルが生成されます。

ESORAXA_D.so	動的
ESORAXA.so	静的

3.4. リモート開発用デーモンの起動

root ユーザーに切り替えます。

環境変数を指定します。

export COBOPT=/home/tarot/cobopt.ora

次に startrdodaemon コマンドを実行します。

コマンド例）

$COBDIR/remotedev/startrdodaemon

Starting RSE daemon...

3.5. Micro Focus Directory Server の起動

root ユーザーで mfds (Micro Focus Directory Server) コマンドを実行します。使用する環境によって、明示的に 32-bit 環境用に mfds32 コマンド、64-bit 環境用に mfds64 コマンドを実行することもできます。

コマンド例） mfds &

上記 & を付加すると、前項の環境変数を基にバックグラウンドで mfds のプロセスが起動されます。

補足）
１つの管理画面上で、異なるサーバー上で稼働するディレクトリサーバーを管理することができます。
本チュートリアルでは、localhost 上（Windows 上）の管理画面を用いてリモートサーバー上で稼働するディレクトリサーバーを管理していますが、リモートサーバー上で管理画面を起動し、そちらを利用することもできます。
その場合は、リモートサーバーの管理画面機能を有効にするため、escwa コマンドの実行が必要です。
escwa &
localhost 以外からのアクセスの許可を行う場合は、以下のようなオプションをつけて実行します。
escwa --BasicConfig.MfRequestedEndpoint=tcp:*:10086 &

3.6. リソースアダプターの編集

root ユーザーのまま作業を行います。

1) COBOL Resource Adapter utility の実行

$ COBDIR/javaee へ移動し、ravaluesupdater.sh (COBOL Resource Adapter Utility) を実行します。

例：bash ravaluesupdater.sh

① どのアプリケーションサーバーを使用しているのかの問いには “jboss74EAP” をタイプします。

Please enter the application server you would like to update: jboss74EAP

② どのリソースアダプターを編集するのかの問いには “mfcobol-xa.rar” をタイプします。

Please enter the resource adapter you would like to update: mfcobol-xa.rar

③ サーバーホスト名を変更するかどうかの問いには “n” をタイプします。

④ サーバーポートの変更をするかの問いには ”n” を入力します。

⑤ トレースを取得するかの問いには ”x” を指定します。

⑥ 全ての変更を保存するかどうかの問いには ”y” を指定して終了します。

3.7. リソースアダプターを JBoss EAP 7.4.6 へディプロイ

1) COBOL Resource Adapter utility で編集した「mfcobol-xa.rar」をコピー

$ $COBDIR/javaee/javaee7/jbossEAP74 へ移動し、「mfcobol-xa.rar」ファイルを

$JBOSS_HOME/standalone/deployments へコピーします。

例：

cp -p $COBDIR/javaee/javaee7/jbossEAP74/mfcobol-xa.rar $JBOSS_HOME/standalone/deployments

3.8. JBoss EAP 7.4.6 の設定と起動

1) JBoss の設定ファイルを XA 用のリソースアダプター向けに編集

一般 ユーザーでログインし、JBoss の Standalone サーバー向け設定ファイルをエディタ等で開いて編集します。編集内容は Visual COBOL のマニュアルを参照し、「mfcobol-xa.rar」をリソースアダプターに追加します。マニュアル参照箇所 ：

https://www.microfocus.co.jp/manuals/VC80/Eclipse/vc80indx.html

ディプロイ > Enterprise Server へのディプロイ >モダナイズ済みのアプリケーションのディプロイおよび構成 > EJB およびリソース アダプターのディプロイ > EJB のディプロイ - 概要 > JBoss へのディプロイ

2) JBoss の起動

下記のコマンドを実行し JBoss EAP 7.4.6 を起動します。例にあるようなメッセージが表示されていれば正しく起動されリソースアダプターもディプロイされています。

$JBOSS_HOME/bin/standalone.sh -b 0.0.0.0 -bmanagement=0.0.0.0

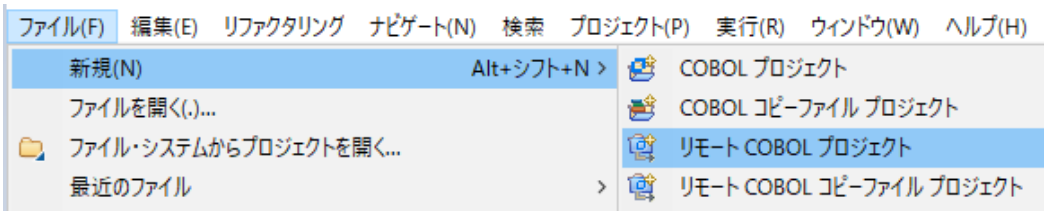
```
# $JBOSS_HOME/bin/standalone.sh -b 0.0.0.0 -bmanagement=0.0.0.0

=========================================================================

  JBoss Bootstrap Environment

  JBOSS_HOME: /home/jboss/EAP-7.4

  JAVA: /usr/java/jdk1.8.0_181-amd64/bin/java

  JAVA_OPTS:        -server    -verbose:gc    -Xloggc:"/home/jboss/EAP-7.4/standalone/log/gc.log"    -XX:+PrintGCDetails    -
XX:+PrintGCDateStamps  -XX:+UseGCLogFileRotation  -XX:NumberOfGCLogFiles=5  -XX:GCLogFileSize=3M  -XX:-TraceClassUnloading  -
Xms1303m      -Xmx1303m      -XX:MetaspaceSize=96M      -XX:MaxMetaspaceSize=256m      -Djava.net.preferIPv4Stack=true      -
Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true

=========================================================================

途中省略

16:13:29,269  INFO    [org.jboss.as.connector.deployment]  (MSC  service  thread  1-5)  WFLYJCA0002:  Jakarta  Connectors
ConnectionFactory [java:/eis/MFCobol_v1.5] をバインドました

16:13:31,121 INFO  [org.infinispan.CONTAINER] (ServerService Thread Pool -- 73) ISPN000128: Infinispan version: Infinispan
'Corona Extra' 11.0.15.Final-redhat-00001

16:13:31,609 INFO  [org.infinispan.CONFIG] (MSC service thread 1-3) ISPN000152: Passivation configured without an eviction
policy being selected. Only manually evicted entities will be passivated.

16:13:31,611 INFO  [org.infinispan.CONFIG] (MSC service thread 1-3) ISPN000152: Passivation configured without an eviction
policy being selected. Only manually evicted entities will be passivated.

16:13:31,682 INFO  [org.infinispan.PERSISTENCE] (ServerService Thread Pool -- 73) ISPN000556: Starting user marshaller
'org.wildfly.clustering.infinispan.spi.marshalling.InfinispanProtoStreamMarshaller'

16:13:32,304 INFO  [org.jboss.as.clustering.infinispan] (ServerService Thread Pool -- 73) WFLYCLINF0002: ejb コンテナーから http-
remoting-connector キャッシュを開始しました。

16:13:33,155 INFO  [org.jboss.as.server] (ServerService Thread Pool -- 42) WFLYSRV0010: "mfcobol-xa.rar" (runtime-name :
"mfcobol-xa.rar") をデプロイしました。

16:13:33,670 INFO  [org.jboss.as.server] (Controller Boot Thread) WFLYSRV0212: サーバーを再開しています

16:13:34,248 INFO  [org.jboss.as] (Controller Boot Thread) WFLYSRV0025: JBoss EAP 7.4.6.GA (WildFly Core 15.0.15.Final-
redhat-00001) は 26619 ミリ秒で開始しました - サービス 615 個のうち 413 個を開始しました (348 のサービスはレイジー、パッシブ、またはオンデマンドです)。

16:13:34,249 INFO  [org.jboss.as] (Controller Boot Thread) WFLYSRV0060: http://0.0.0.0:9990/management 上でリッスンする HTTP 管理イ
ンターフェース

16:13:34,249 INFO  [org.jboss.as] (Controller Boot Thread) WFLYSRV0051: 管理コンソールは http://0.0.0.0:9990 をリッスンしています。
```

3.9. Windows クライアントでの開発作業

1) Visual COBOL for Eclipse を起動し、リモート COBOL プロジェクトを作成

① Visual COBOL を起動します。ワークスペースは任意のフォルダを指定してください。

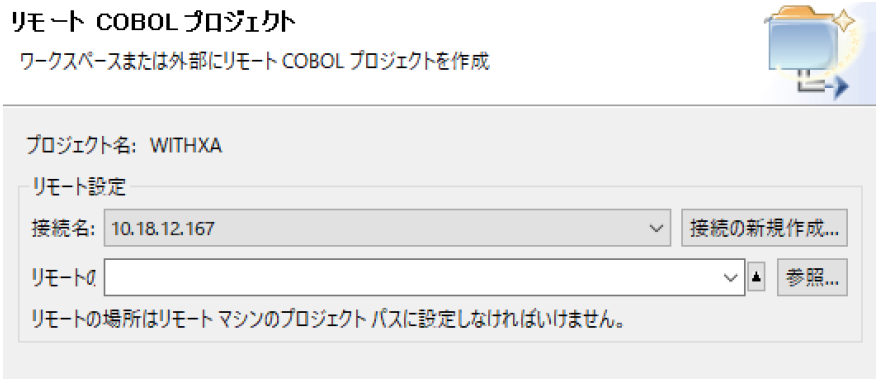
② [ファイル(F)] メニュー > [新規(N)] > [リモート COBOL プロジェクト] を選択します。



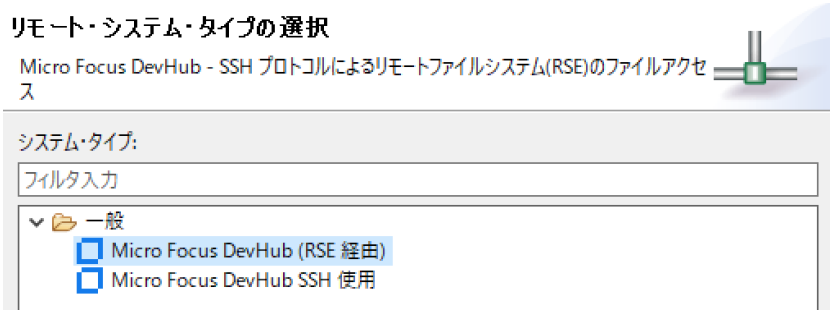
③ プロジェクト名 "WITHXA" を指定し、[ファイル システムを選択]は「リモートファイルシステム(RSE)」を選んで [次へ(N)] ボタンをクリックします。

④ プロジェクトテンプレートを選択する画面では「Micro Focus テンプレート[64 ビット]」を選択し [次へ(N)] ボタンをクリックします。

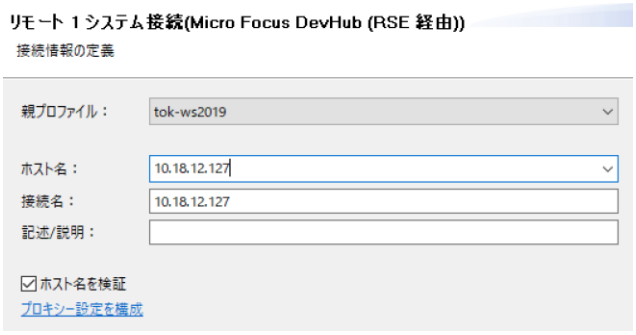
⑤ 「リモート COBOL プロジェクト」のダイアログが表示されます。[接続の新規作成] ボタンをクリックします。



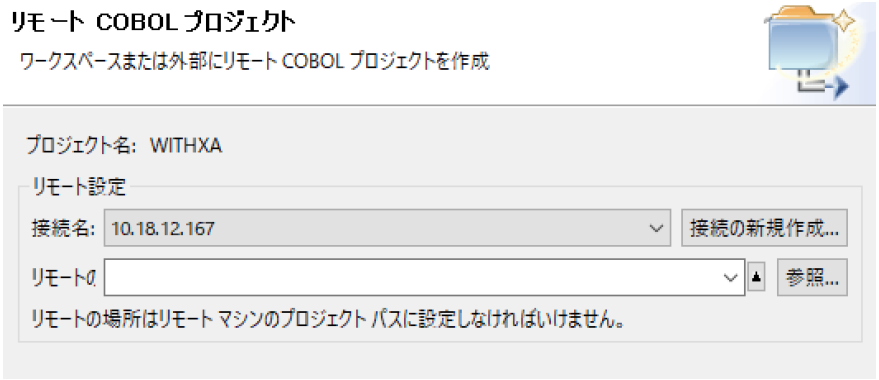
⑥ [Micro Focus DevHub(RSE 経由)] を選択し、[次へ(N)] ボタンをクリックします。



⑦ [ホスト名] 欄に Linux サーバーの IP アドレスを入力し、[終了(F)] ボタンをクリックします。



⑧ リモート COBOL プロジェクトの画面に戻ってくるので[リモートの場所] 横にある[参照] ボタンをクリックします。



⑨ 「マイ・ホーム」の左にある展開アイコンをクリック

⑩ Linux/UNIX 側でソースや生成されるモジュール等を格納するプロジェクトディレクトリとして利用するディレクトリをツリーで選択し、[OK] ボタンをクリックします。

⑪ ユーザー認証に関するポップアップが出たら Linux サーバーのユーザーの認証情報を入力し、[Save Password] にチェックを入れ、[OK] ボタンをクリックします。

⑫ Connection 接続先名 has not been secured using SSL. Proceed anyway? のポップアップに対しては「Do not show this message again」にチェックを入れ [はい] ボタンをクリックします。

⑬ Linux サーバー上でリモート開発のプロジェクトディレクトリとして利用するディレクトリをツリーで指定し、[OK] ボタンをクリックします。

⑭ [終了] ボタンをクリックします。

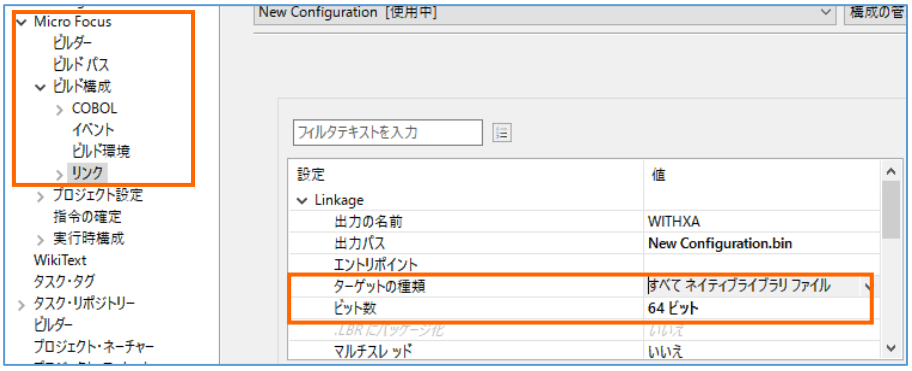
⑮ Linux サーバー上に Eclipse の COBOL プロジェクトが生成されます。また、同時にリモート接続先のフォルダにもプロジェクトファイルが作成されます。

2) 64-bit の Oracle 連携アプリケーションを生成するようプロジェクトを構成

① COBOL エクスプローラにて作成したプロジェクトを右クリックし、[プロパティ(R)] を選択します。

② [Micro Focus] > [ビルド構成] > [リンク] を展開します。

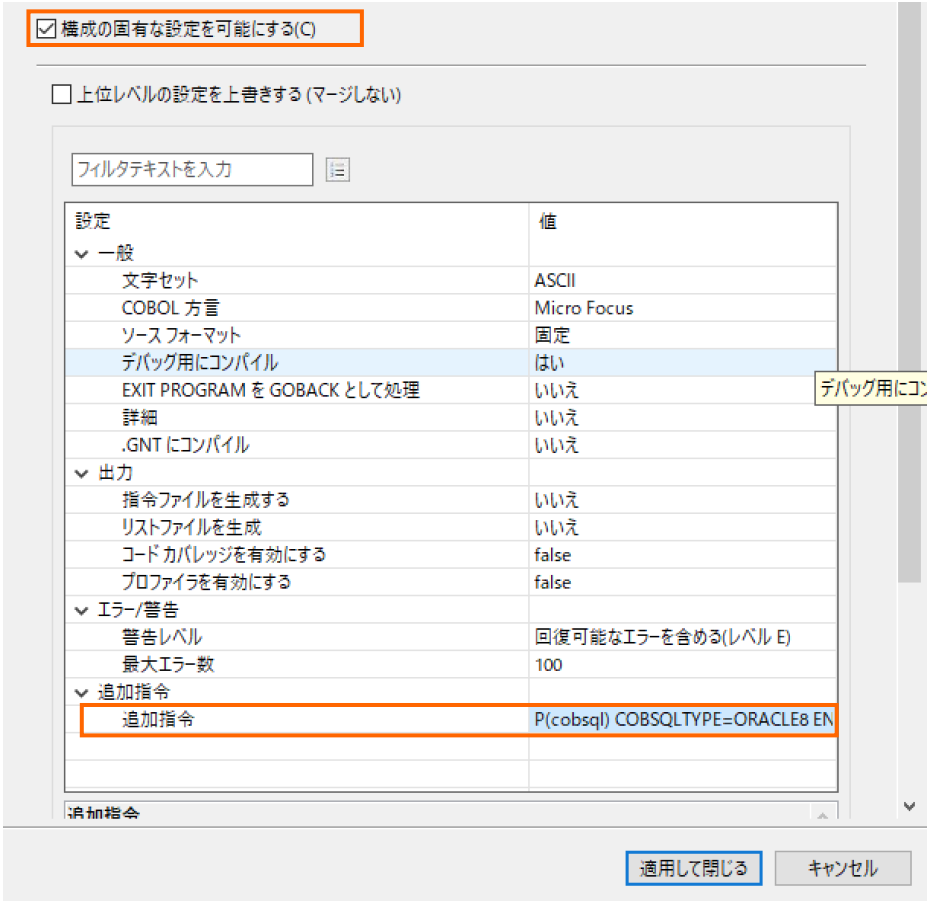
③ [ターゲットの種類] を「すべてネイティブライブラリ ファイル」に、[ビット数] 欄が [64 ビット] になっていることを確認します。



④ [Micro Focus] > [ビルド構成] > [COBOL] を展開します。

⑤ [構成の固有な設定を可能にする(C)] をチェックします。

⑥ [追加指令] 欄に COBSQL の指令を指定し、[OK] ボタンをクリックします。



指定値：「P(cobsql) COBSQLTYPE=ORACLE8 END-C ENDP」

⑦ 最後に[適用して閉じる] ボタンをクリックします。

3) Oracle 上のデータを照会/更新する埋め込み SQL 文の入った COBOL プログラムをプロジェクトに追加

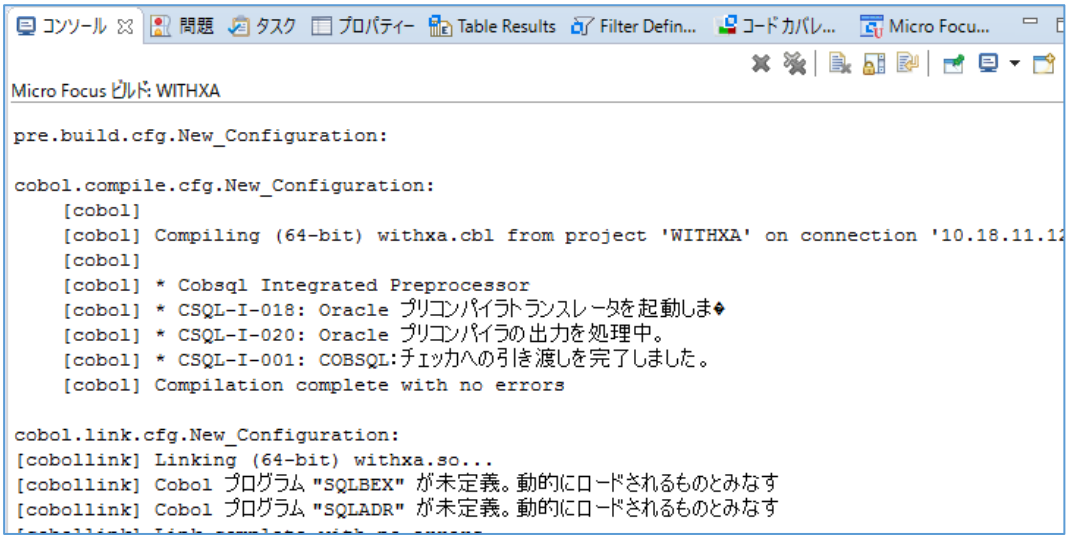
① COBOL エクスプローラにてプロジェクトを右クリックし、[インポート(i)] > [インポート(I)] を選択します。

② インポート用のダイアログが表示されるので [一般] > [ファイル・システム] を指定し、[次へ(N)] ボタンをクリックします。

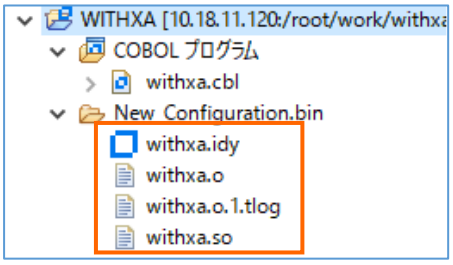
③ [参照] ボタンを押して Windows のコモンダイアログからチュートリアル用のソースコードが保存されているフォルダまで移動します。

④ 対象のソースコード「withxa.cbl」にチェックを入れて [終了(F)] ボタンをクリックします。

⑤ Windows から Linux へプログラムソースがダイレクトに転送され、Linux 上の Pro*COBOL プリコンパイラ及び Visual COBOL コンパイラを使ってビルド処理されます。



⑥ COBOL エクスプローラビューにて、Linux サーバー環境に呼び出し可能な共有オブジェクトが生成されていることを確認できます。



⑦ 端末エミュレータでも実際に生成されていることを確認できます。

```
# ls
withxa.idy   withxa.o   withxa.o.1.tlog   withxa.so
```

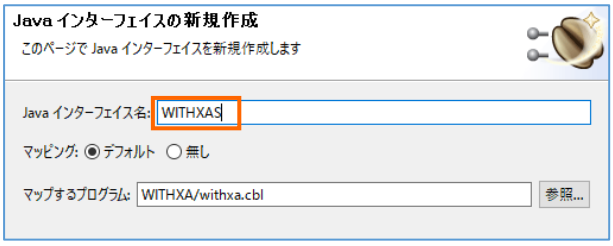
4) インポートしたプログラムの概要

- Oracle Database への接続は XA スイッチモジュールを経由しているのでプログラム中には CONNECT 文等の接続関連の命令は記述しません。
- 更新前の EMP.ENAME を取得後、LINKAGE 経由で受け取った値に基づき、EMP テーブルを更新します。
- LINKAGE パラメータ L-COMMIT-OR-ROLLBACK に「R」が渡されると意図的に添え字範囲外の実行時エラーを発生させます。

5) アプリケーションの COBOL-Java 変換マッピングを作成

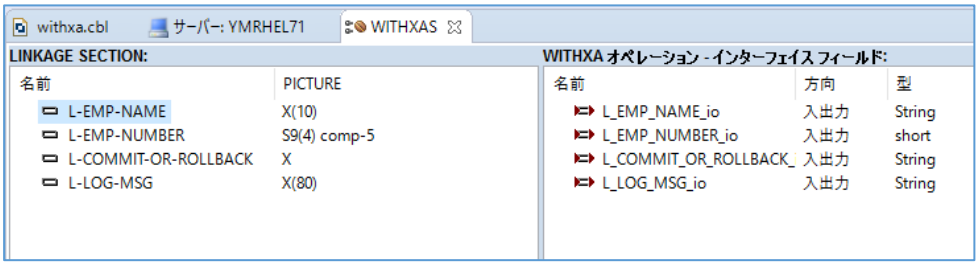
① COBOL エクスプローラにて「withxa.cbl」を右クリックし、コンテクストメニューから [新規作成] > [Java インターフェイス] を選択します。

② Java インターフェイス名には “WITHXAS” を入力し、[終了] ボタンをクリックするとデフォルトのインターフェイスマッピングが生成されます。

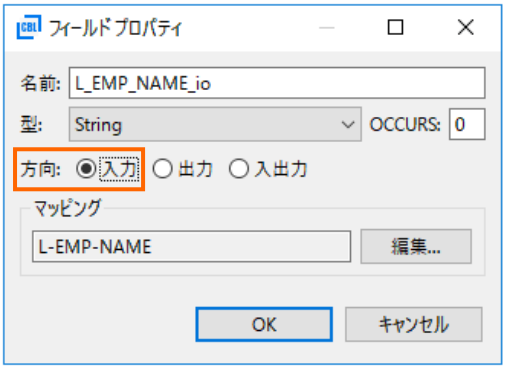


6) 生成されたインターフェイスマッピングを編集

① 「WITHXA オペレーション - インターフェイスフィールド：」を編集します。



② L_EMP_NAME_io をクリックして [方向] を “入力” に変更し、[OK] ボタンをクリックします。



③ 同様の作業を「L_EMP_NUMBER_io」、「L_COMMIT_OR_ROLLBACK_io」に行います。

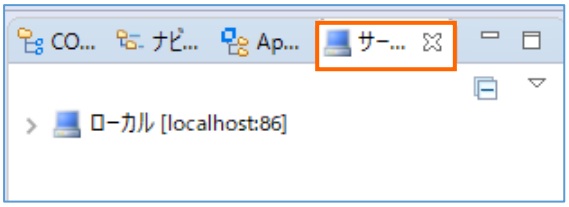
④ 「L_LOG_MSG_io」は ”出力” に変更します。

⑤ CTRL+S キーを押して設定を保存します。

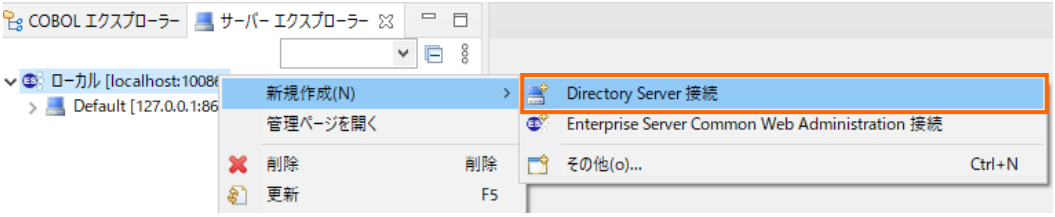
3.10. Enterprise Server のインスタンス起動準備

1) Linux サーバー上で稼動する Micro Focus Directory Server をサーバーエクスプローラビューへ追加

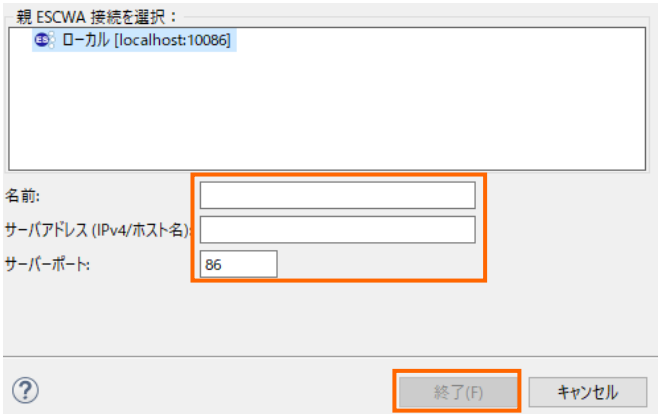
① Eclipse 上でサーバーエクスプローラビューを表示します。



② 一番上の「ローカル [localhost:10086]」上で右クリックし、コンテクストメニューから[新規作成(N)] → [Directory Server 接続] を選択します。



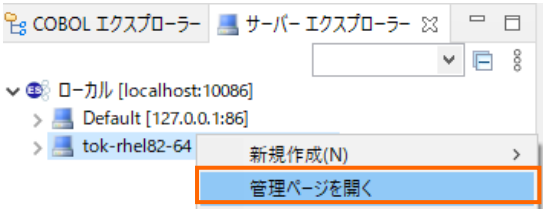
③ Linux サーバーの名前とアドレスを指定し、[終了(F)]ボタンをクリックします。



3.11. ESCWA よりデバッグ情報とポートを設定

1) ESCWA（Enterprise Server Common Web Administration）の起動

① サーバーエクスプローラーより、ローカル先の接続を展開し、さきほど設定した Linux サーバー上で右クリックし、コンテクストメニューから「管理ページを開く」を選択します。



② ブラウザが起動し、ESCWA （Enterprise Server Common Web Administration）の画面が表示されます。

③ 「ESDEMO64」をクリックし、「一般」メニューから「プロパティ」をクリックします。

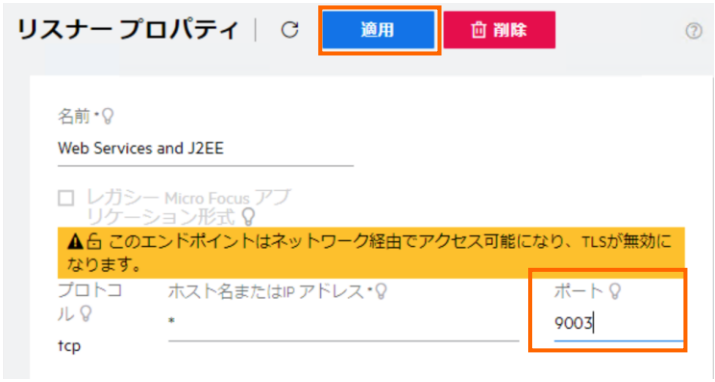
④ 下にスクロールし、[動的デバッグを許可] にチェックを入れます。

⑤ 上にスクロールし、[適用] ボタンをクリックします。

⑥ 次に、[一般]メニューから[リスナー]を選択します。

⑦ 「通信プロセス１」より [Web Services and J2EE] をクリックします。

⑧ リスナープロパティのポートに”9003”を入力し、[適用] ボタンをクリックします。

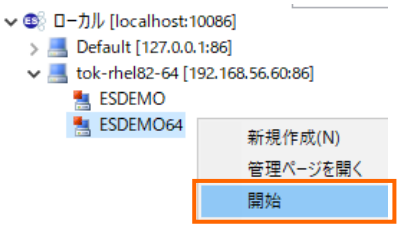


※このポート番号はリソースアダプターと Enterprise Server が通信を行う時に使用されます。

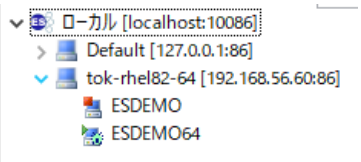
3.12. 64 ビット Enterprise Server のインスタンス起動

1) インスタンスの起動

① Eclipse IDE に戻り、サーバーエクスプローラーのツリーを展開し、「ESDEMO64」上で右クリックしコンテクストメニューから [開始] を選択し、Enterprise Server を起動します。



② しばらくするとサーバーが起動されていることが確認できます。



3.13. XA リソースを設定

1) ブラウザより ESCWA にて XA 関連の情報を設定

① 「ESDEMO64」をクリックし、「一般」メニューから「XA リソース」をクリックします。

② 「新規作成」をクリックし、「一般」メニューから「XA リソース」をクリックすると、「XA リソースの構成」画面が表示されます。

③ 必要項目を入力後［保存］ボタンをクリックします。

項目名	説明
ID	ORACLE のインスタンス名を指定します。ここでは ORCL を指定します。
名前	XA リソース名として任意の名前を指定します。 例：ORACLE19C
モジュール	前項で作成した XA スイッチモジュールのパスとファイル名を指定します。 【Oracle 使用時の例】 動的登録 /home/ユーザー名/ディレクトリ名/ESORAXA64_D.so を入力します。
OPEN 文字列	対象データベースのオープン文字列を指定します。 【Oracle 使用時の例】 Oracle_XA+SesTm=100+SqlNet=ORCL-19C+Acc=P/scott/tiger を入力します。 ※接続文字列の詳細については Oracle ドキュメントを参照するか Oracle 管理者に確認してください。
有効	有効、無効切り替えチェックを指定します。ここではオンを指定します。

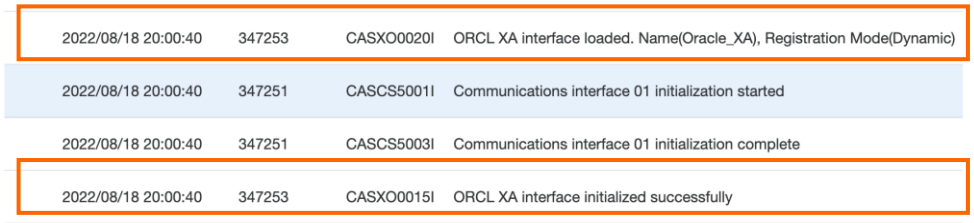
2) Enterprise Server インスタンスの再起動

③ サーバーエクスプローラーのツリーを展開し、「ESDEMO64」上で右クリックしコンテクストメニューから［停止］を選択し、Enterprise Server を停止、その後再度起動します。

④ ESCWA 上にて起動ログをチェックします。「モニター」メニューから「ログ」→「コンソールログ」に遷移します。

注意）
モニターメニューが表示されない場合は、ESCWA とリスナーの通信が正しく行えていない可能性があります。この場合は、3.11 でリスナー設定を変更した際、「ホスト名、または、IP アドレス」に Windows から通信可能な ホスト名や IP アドレスを設定してください。また、リスナーメニューの「通信プロセス 1 」を選択し、[構成] をクリックして、同様にホスト名や IP アドレスを設定してください。

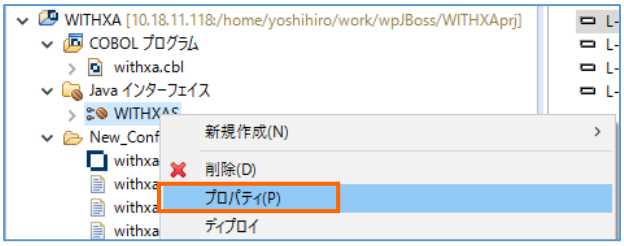
⑤ 下記のようなメッセージの中に XA 関連の初期化が正常に行われているかどうか確認します。



3.14.コンパイルした COBOL アプリケーションを Enterprise Server へディプロイ

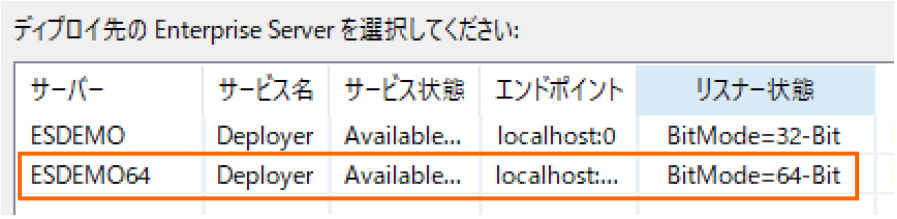
1) Enterprise Server へのディプロイ情報を指定

① COBOL エクスプローラにて追加した Java インターフェイス「WITHXAS」を右クリックし、コンテクストメニューから［プロパティ(R)］を選択します。

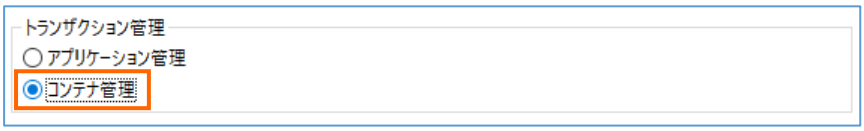


② [ディプロイメントサーバー] タブを選択し、[変更] ボタンをクリックします。

③ Linux サーバーで稼動する「ESDEMO64」を選択し [OK] ボタンをクリックします。



④ [トランザクション管理] フィールドにて「コンテナ管理」を選択します。



⑤ 次に [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」を選択します。

⑥ [ファイル追加] ボタンを押して、 プロジェクトディレクトリ配下の「New_Configuration.bin」に生成された withxa.so 及び withxa.idy を選択し、 [OK] ボタンをクリックします。

⑦ 次に [EJB 生成] タブを選択し、[アプリケーションサーバー] フィールドにて「J2EE7」、「JBoss EAP 7.4」を選択します。

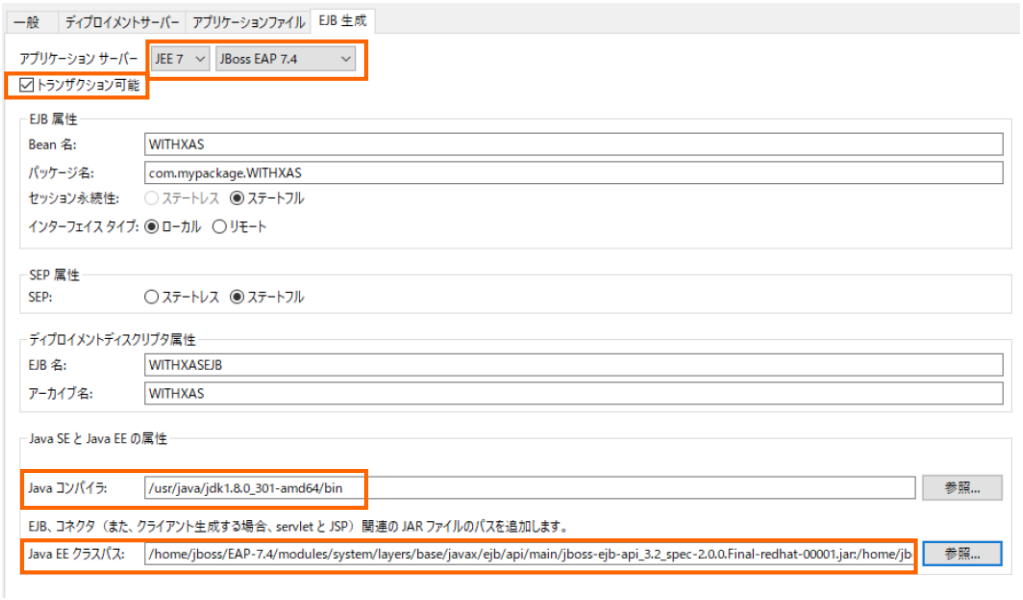
⑧ [トランザクション可能] にチェックされていることを確認します。

⑨ 「J2EE と J2EE の属性」欄にて [Java コンパイラ] フィールドに使用している JDK のパスを入力します。

⑩ 同様に下記の J2EE クラスパスを設定します。

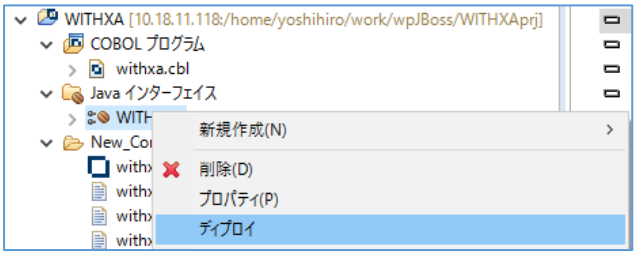
- $JBOSS_HOME/modules/system/layers/base/javax/ejb/api/main/jboss-ejb-api_3.2_spec-2.0.0.Final-redhat-00001.jar
- $JBOSS_HOME/modules/system/layers/base/javax/servlet/api/main/jboss-servlet-api_4.0_spec-2.0.0.Final-redhat-00001.jar
- $JBOSS_HOME/modules/system/layers/base/javax/resource/api/main/jboss-connector-api_1.7_spec-2.0.0.Final-redhat-00001.jar

⑪ すべての情報が下記のイメージのように入力されているか再度チェックします。チェックに問題が無ければ[OK]ボタンをクリックします。

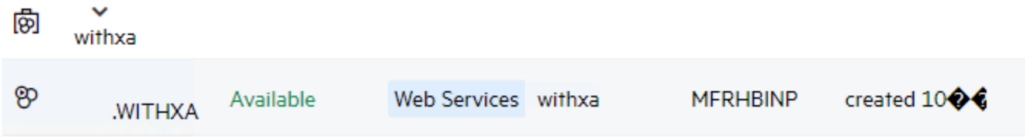


2) 作成した Java サービスを Enterprise Server へディプロイする

① COBOL エクスプローラにて作成した Java インターフェイス「WITHXAS」を右クリックし、コンテクストメニューから[ディプロイ] を選択します。



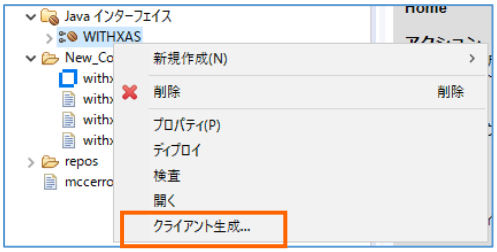
② ESCWA 画面より 右側の Directory Server ツリーから「ESDEMO64」をクリックし、メイン画面の「ネイティブ」→[サービス] にて正常にディプロイできたことを確認することができます。



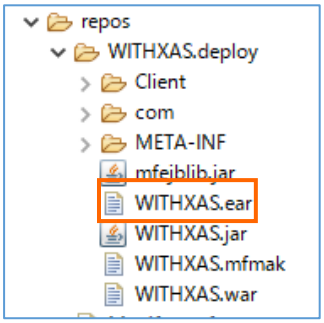
3.15. テスト用クライアントアプリケーションを JBoss EAP にディプロイ

1) ディプロイした Java サービスをテストするための J2EE アプリケーションを生成する

① COBOL エクスプローラにて Java インターフェイスを右クリックして［クライアント生成］を選択します。



② 正常に処理されると＜プロジェクトディレクトリ＞/repos/＜サービス名＞.deploy 配下に拡張子 .ear 形式にアーカイブされた J2EE アプリケーションが生成されます。



2) 生成された J2EE アプリケーションを JBoss EAP 7.4 へデプロイ

① Telnet で Visual COBOL 作業用ディレクトリに生成された "WITHXAS.ear" を $JBOSS_HOME/standalone/deployments 配下にコピーする。

例：

cp -p WITHXAS.ear $JBOSS_HOME/standalone/deployments/.

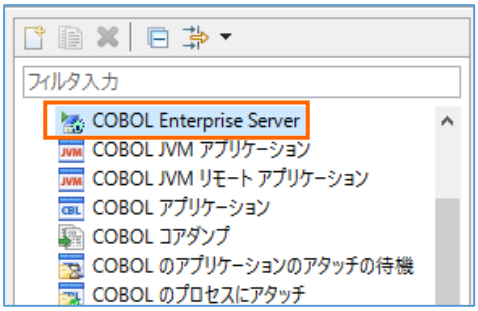
② 正常にディプロイされたことを JBoss を起動した Linux のターミナルから確認ができます。

```
21:15:07,166 INFO  [org.jboss.weld.deployer] (MSC service thread 1-4) WFLYWELD0003: Weld デプロイメント WITHXAS.ear を処理しています
21:15:08,542 INFO  [org.jboss.as.connector.deployment] (MSC service thread 1-1) WFLYJCA0007: 接続ファクトリ java:/eis/MFCobol_v1.5 を登録しました
21:15:08,552 WARN  [org.jboss.as.connector.deployers.RaXmlDeployer] (MSC service thread 1-1) IJ020016: Missing <recovery> element. XA recovery disabled for:
java:/eis/MFCobol_v1.5
21:15:08,555 INFO  [org.jboss.as.connector.deployers.RaXmlDeployer] (MSC service thread 1-1) IJ020002: Deployed: file:/home/jboss/EAP-
7.4/standalone/tmp/vfs/temp/tempd80ac60195bdf7d6/content-91c4c0edb34a2895/contents/
21:15:08,576 INFO  [org.jboss.as.connector.deployment] (MSC service thread 1-6) WFLYJCA0002: Jakarta Connectors ConnectionFactory [java:/eis/MFCobol_v1.5] をバインド
ました
21:15:09,461 INFO  [org.hibernate.validator.internal.util.Version] (MSC service thread 1-4) HV000001: Hibernate Validator 6.0.23.Final-redhat-00001
21:15:11,650 INFO  [org.jboss.weld.deployer] (MSC service thread 1-6) WFLYWELD0003: Weld デプロイメント WITHXAS.war を処理しています
21:15:12,263 INFO  [org.jboss.weld.deployer] (MSC service thread 1-1) WFLYWELD0003: Weld デプロイメント WITHXAS.jar を処理しています
21:15:12,265 INFO  [org.jboss.as.ejb3.deployment] (MSC service thread 1-1) WFLYEJB0473: デプロイメントユニット 'subdeployment "WITHXAS.jar" of deployment
"WITHXAS.ear""の 'WITHXASEJB' という名前のセッション Bean の JNDI バインディングは次のとおりです:

        java:global/WITHXAS/WITHXAS.jar/WITHXASEJB!com.mypackage.WITHXAS.WITHXASLocal
        java:app/WITHXAS.jar/WITHXASEJB!com.mypackage.WITHXAS.WITHXASLocal
        java:module/WITHXASEJB!com.mypackage.WITHXAS.WITHXASLocal
        java:global/WITHXAS/WITHXAS.jar/WITHXASEJB
        java:app/WITHXAS.jar/WITHXASEJB
        java:module/WITHXASEJB

21:15:12,903 INFO  [org.jboss.weld.Version] (MSC service thread 1-3) WELD-000900: 3.1.6 (redhat)
21:15:15,267 INFO  [org.infinispan.CONTAINER] (ServerService Thread Pool -- 73) ISPN000128: Infinispan version: Infinispan 'Corona Extra' 11.0.15.Final-redhat-00001
21:15:16,119 INFO  [org.infinispan.CONFIG] (MSC service thread 1-8) ISPN000152: Passivation configured without an eviction policy being selected. Only manually evicted
entities will be passivated.
21:15:16,121 INFO  [org.infinispan.CONFIG] (MSC service thread 1-8) ISPN000152: Passivation configured without an eviction policy being selected. Only manually evicted
entities will be passivated.
21:15:16,806 INFO  [org.infinispan.PERSISTENCE] (ServerService Thread Pool -- 73) ISPN000556: Starting user marshaller
'org.wildfly.clustering.infinispan.spi.marshalling.InfinispanProtoStreamMarshaller'
21:15:19,162 INFO  [org.jboss.as.clustering.infinispan] (ServerService Thread Pool -- 73) WFLYCLINF0002: ejb コンテナーから http-remoting-connector キャッシュを開始しました。
21:15:20,571 INFO  [org.wildfly.extension.undertow] (ServerService Thread Pool -- 75) WFLYUT0021: 登録された web コンテキスト: '/WITHXAS' (サーバー 'default-server' 用)
21:15:21,281 INFO  [org.jboss.as.server] (ServerService Thread Pool -- 42) WFLYSRV0010: "WITHXAS.ear" (runtime-name : "WITHXAS.ear") をデプロイしました。
```

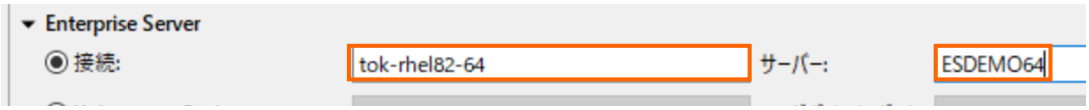
3) Enterprise Server をデバッグ待機する

① COBOL エクスプローラにてプロジェクトを右クリックし、コンテクストメニューから [デバッグ(D)] > [デバッグの構成(B)] を選択します。

② [COBOL Enterprise Server] をダブルクリックします。



③ [一般] タブの Enterprise Server フィールドの [参照] ボタンをクリックし、Linux サーバー上で稼働する「ESDEMO64」を選択します。



④ [Java] タブをクリックし、Java サービス名が空白（全てのサービスがデバッグ対象）となっていることを確認します。

⑤ [デバッグ] ボタンをクリックし、[パースペクティブの切り替えの確認] のプロンプトに対しては [はい] を選択します。

⑥ デバッグパースペクティブにてアタッチ待機状態になっていることが確認できます。

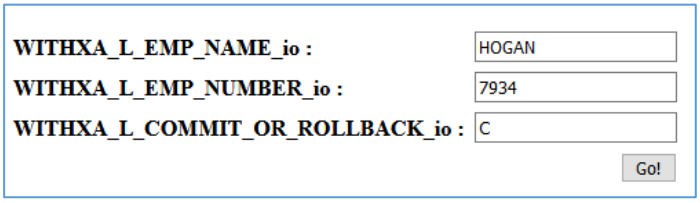
3.16. テスト用アプリケーションを経由して COBOL アプリケーションを呼び出し

1) ディプロイした J2EE アプリケーションをデバッグ実行する（成功パターン）

① ブラウザを起動し、JBoss 実行中のサーバーの IP アドレスを入力し、アプリケーションを起動します。

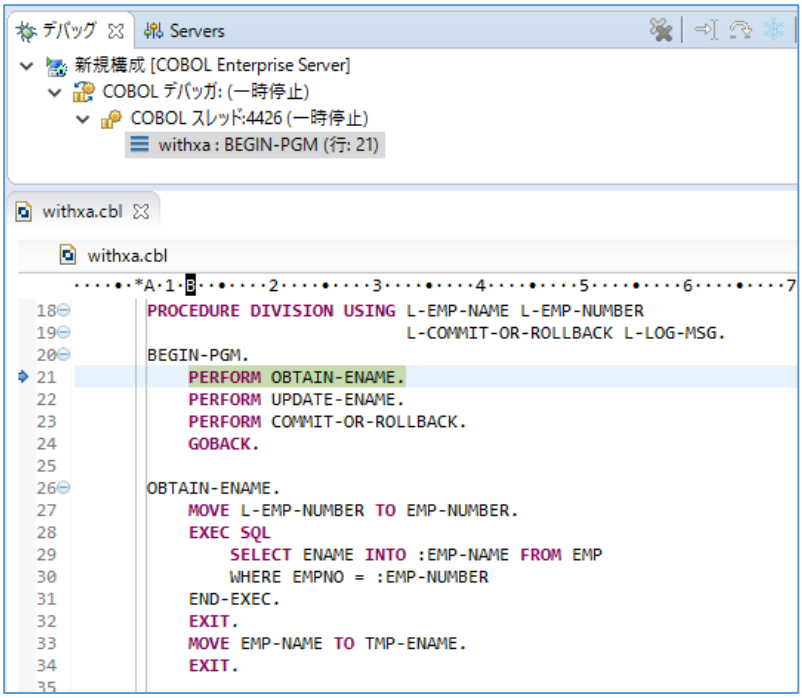
例：10.18.11.118:8080/WITHXAS/WITHXA.jsp

② ３つのパラメータを入力し、[Go!] ボタンをクリックして、アプリケーションを実行します。



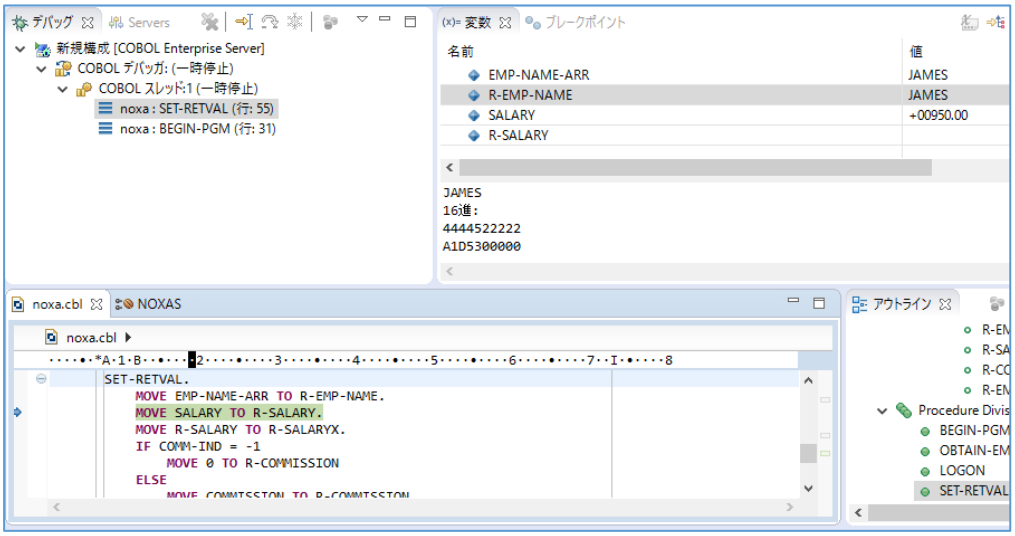
③ 処理が Enterprise Server に渡り、Eclipse のデバッガーが起動します。

④ Enterprise Server にディプロイした COBOL プログラムの最初の行を実行する前で処理が一時停止していることが確認できます



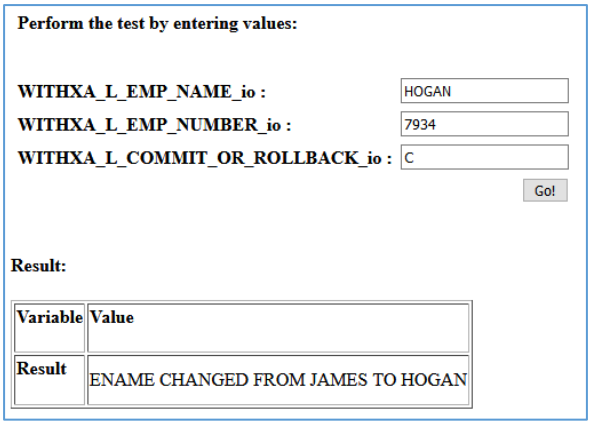
⑤ F5 キー打鍵で 1 ステップずつ処理を進めることができます。尚、このプログラムは COBSQL を利用してコンパイルしているため、プリコンパイル後のソースではなく埋め込み SQL 文が入ったプリコンパイル前のソースでデバッグができます。

⑥ 変数ビューでは、現在のステップで参照している変数の中身を確認できます。



⑦ 本プログラムはトランザクションマネージャーが確立した接続を利用するため、プログラムから CONNECT 文は発行していませんが、正常に SQL 文を実行しています。

⑧ 処理を最後まで進めると Java 側に処理が戻り、COBOL から返された値を戻します。



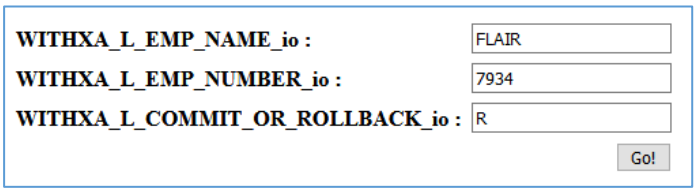
⑨ アプリケーションが処理したレコードを SQL*Plus で確認します。トランザクションが COMMIT されたので値が更新されていることが確認できます

```
SQL> select * from emp where empno=7934;

     EMPNO ENAME                JOB                       MGR HIREDATE

---------- -------------------- ------------------ ---------- --------

      SAL       COMM     DEPTNO

---------- ---------- ----------

      7934 HOGAN                CLERK                     7782 82-01-23

      1300                    10
```

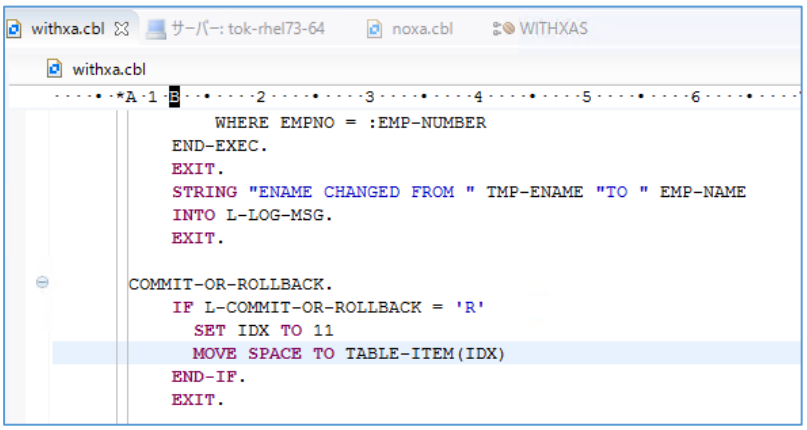
2) ディプロイした J2EE アプリケーションをデバッグ実行する（実行時エラーで終了するパターン）

① ブラウザを起動し、JBoss 実行中のサーバーの IP アドレスを入力し、アプリケーションを起動します。

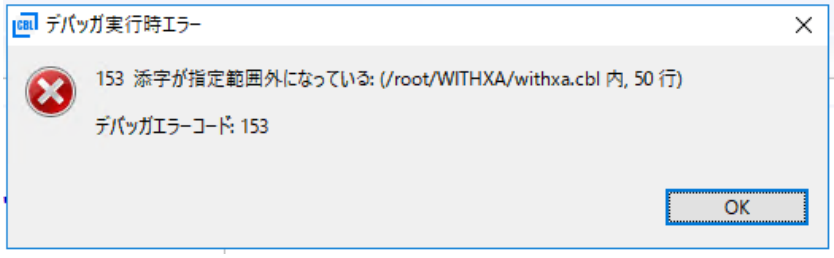
② ３つのパラメータを入力し、[Go!] ボタンをクリックして、アプリケーションを実行します。



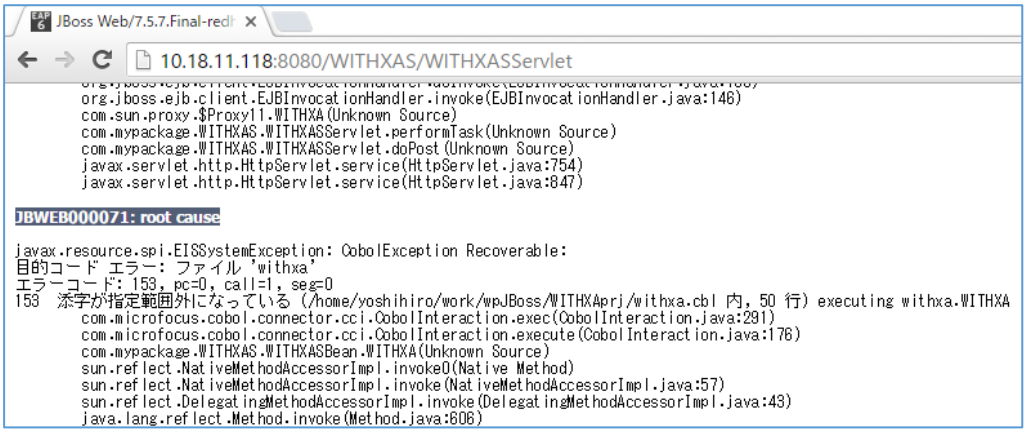
③ 今回は L-COMMIT-OR-ROLLBACK に「R」を格納したため、UPDATE 文実行後の下図の IF 文は真と評価されます。



④ IF 文中の MOVE 文を実行すると添え字範囲外の実行時エラーが発生します。



⑤ デバッガー側で処理を止めずに処理を進める場合、 Java 側へも COBOL の処理で例外が発生したことが伝播されブラウザにもエラーが発生した旨が表示されます。



⑥ アプリケーションが処理したレコードを SQL*Plus で確認します。トランザクションがロールバックされたので Update 文による値の更新は取り消されています。

```
SQL> select * from emp where empno=7934;

     EMPNO ENAME                JOB                       MGR HIREDATE

---------- -------------------- ------------------ ---------- --------

       SAL       COMM     DEPTNO

---------- ---------- ----------

      7934 HOGAN                CLERK                     7782 82-01-23

      1300                    10
```

3.17. インスタンスの停止

1) Enterpise Server の停止

① 「サーバーエクスプローラー」にて「ESDEMO64」上で右クリックし、コンテクストメニューから［停止］を選択します。

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