Micro Focus Visual COBOL 2.1 J for Windows Interstage Application Server V 11.0.0 動作検証結果報告書

2012年12月11日
マイクロフォーカス株式会社

1　検証概要、目的及びテスト方法

1.1　検証概要

Micro Focus Visual COBOL 2.1 J の Enterprise Server が提供する JavaEE Connector 機能は、JCA仕様準拠のコンテナとして多くの JavaEE準拠アプリケーションサーバーについて動作検証がなされています。本報告書は、富士通のInterstage Application Server(以降Interstage ASと略記)との JavaEE Connector の接続性を検証し、報告するものです。

1.2　目的及びテスト方法

Micro Focus Visual COBOL 2.1 J の Enterprise Server が提供する JavaEE Connector は、現在 WebSphere, WebLogic, JBoss などとの連携が動作保証されています。しかし Enterprise Serverは、JCA仕様準拠のコンテナとして、設計上は JCA仕様に準拠したすべてのアプリケーションサーバーとの連携が可能です。
Interstage ASはJCAの仕様に準拠しており、理論的には Micro Focus Enterprise Server のEISとも連携するはずです。今回、以下のテストプログラムを実行することによって、このことを実際に検証しました。

(1) 渡された数字パラメタのキーから索引ファイルを読み、関連データを返すCOBOLサブルーチンを使用
(2) Interface Mapping Toolkit が自動生成した EJB と ServletクライアントをInterstage AS上で運用し、COBOLを呼び出す

2　使用ハードウェア及びソフトウェア一覧

Dell Latitude E6410 Intel Core i5 CPU M520 2.40GHz　Memory 2Gb
Windows Server 2008 R2 SP1 x64
Micro Focus Visual COBOL 2.1
Interstage AS Standard-J Edition (64bit) V11.0.0

作業用環境として Windows Server 2008 PC を使用し、Internet Explorer 8を利用

3　テスト内容

以下に実施したテストの概要を述べます。詳細な手順については補足に記載します。

(1) 使用した COBOLロジック
渡された1つの数字パラメタのキーから索引ファイルを読み、3つの関連データを返す簡単なCOBOLサブルーチンを使用
(2) 使用したリソースアダプタ
C:¥Program Files (x86)¥Micro Focus¥Enterprise Developer¥javaee¥j2ee14¥beaweblogic9¥mfcobol-notx.rar
WebLogic 9.x にディプロイするのに適した形式でパッケージされたものであり、JavaEE仕様に照らして最も標準的な提供形態です。
(3) 使用した Enterprise Server
既定義の ESDEMO をそのまま使用。
(4) 使用した JavaEEアプリケーション
Visual COBOL の Interface Mapping Toolkit がデプロイ時に自動生成する EJB と、自動生成される Webモジュールクライアントを使用。

4　結果

上記のテストを実行した結果、正常に実行されることを確認しました。詳細な結果については補足に記載します。

5　テスト結果及び考察

最新の JavaEE標準をサポートする Interstage ASで、既存の Micro Focus Visual COBOL 2.1 J の JavaEE Connector 接続を問題なく使用できることが検証できました。

以上

補足. 検証の手順

1. 前提条件

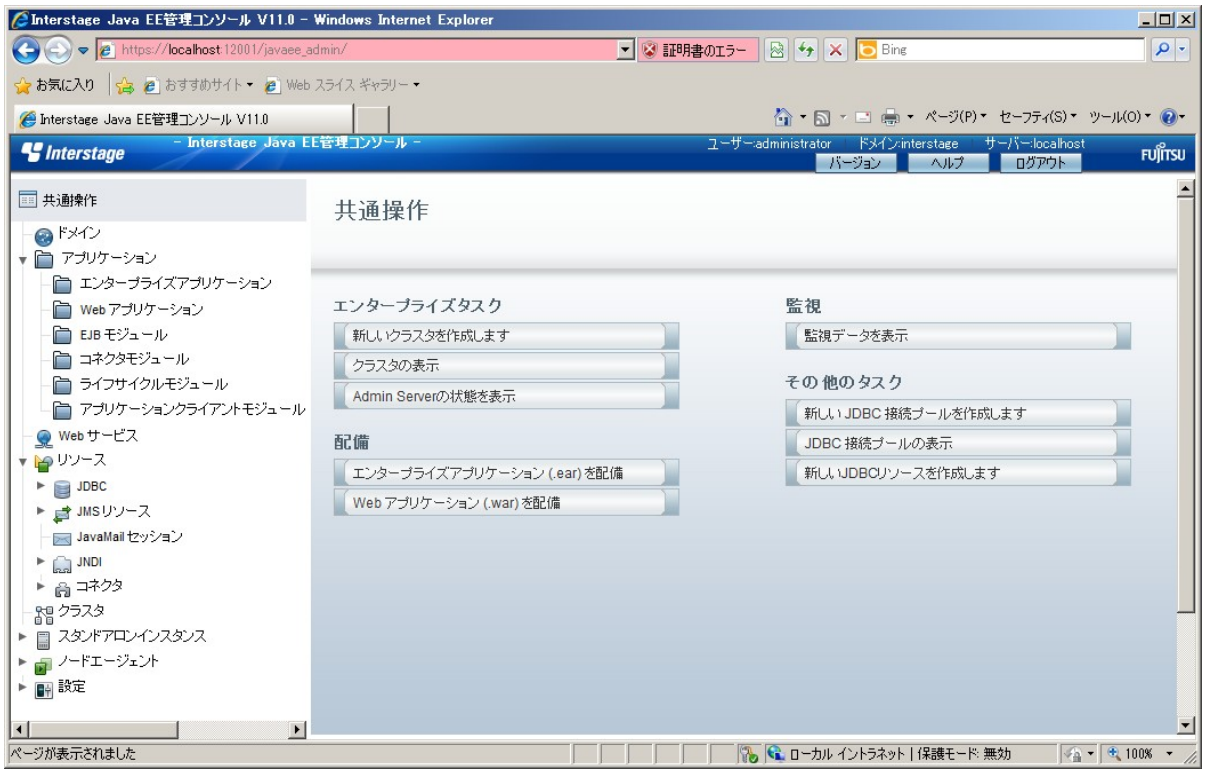
本検証では、各ソフトウェアはデフォルトでインストールされたままの状態になっていることを仮定しています。 Visual COBOL はデフォルトのインストール先に Enterprise Server も含めてインストールされており、出荷時設定のサーバー ESDEMO がそのままの状態で利用可能になっているものとします。 検証を始める前に ESDEMO を開始状態にしておきます。
Interstage AS もデフォルトでインストールされており、OS 管理者ユーザで利用可能になっているものとします。
ここでは、以下の簡単な COBOL 例題プログラム ReadCust.cbl を使用します。第一引数をキーに索引ファイルを読み込み、同一レコードのデータを第二、第三、第四引数に返すというだけのプログラムです：

```
       PROGRAM-ID.   "ReadCust".
       FILE-CONTROL.
            SELECT CUST-MASTER ASSIGN TO "CUST.dat"
                       ORGANIZATION INDEXED RECORD KEY FS-CUSTID
                       ACCESS MODE RANDOM.
       DATA                  DIVISION.
       FILE                  SECTION.
       FD   CUST-MASTER.
       01   CUST-REC.
            05   FS-CustId          PIC X(4)     COMP-5.
            05   FS-CustName        PIC X(30).
            05   FS-CustCompany     PIC X(30).
            05   FS-CustEmail       PIC X(30).
       LINKAGE               SECTION.
       01   CustId          PIC X(4)     COMP-5.
       01   CustName        PIC X(30).
       01   CustCompany     PIC X(30).
       01   CustEmail       PIC X(30).
       PROCEDURE             DIVISION
            USING CustId CustName CustCompany CustEmail.
       1.
            OPEN             I-O CUST-MASTER.
            MOVE             CustId          TO   FS-CustId.
            READ             CUST-MASTER INVALID   CONTINUE
                             END-READ.
            CLOSE            CUST-MASTER.
            MOVE             FS-CustName       TO   CustName.
            MOVE             FS-CustCompany TO   CustCompany.
            MOVE             FS-CustEmail     TO   CustEmail.
            EXIT             PROGRAM.
```

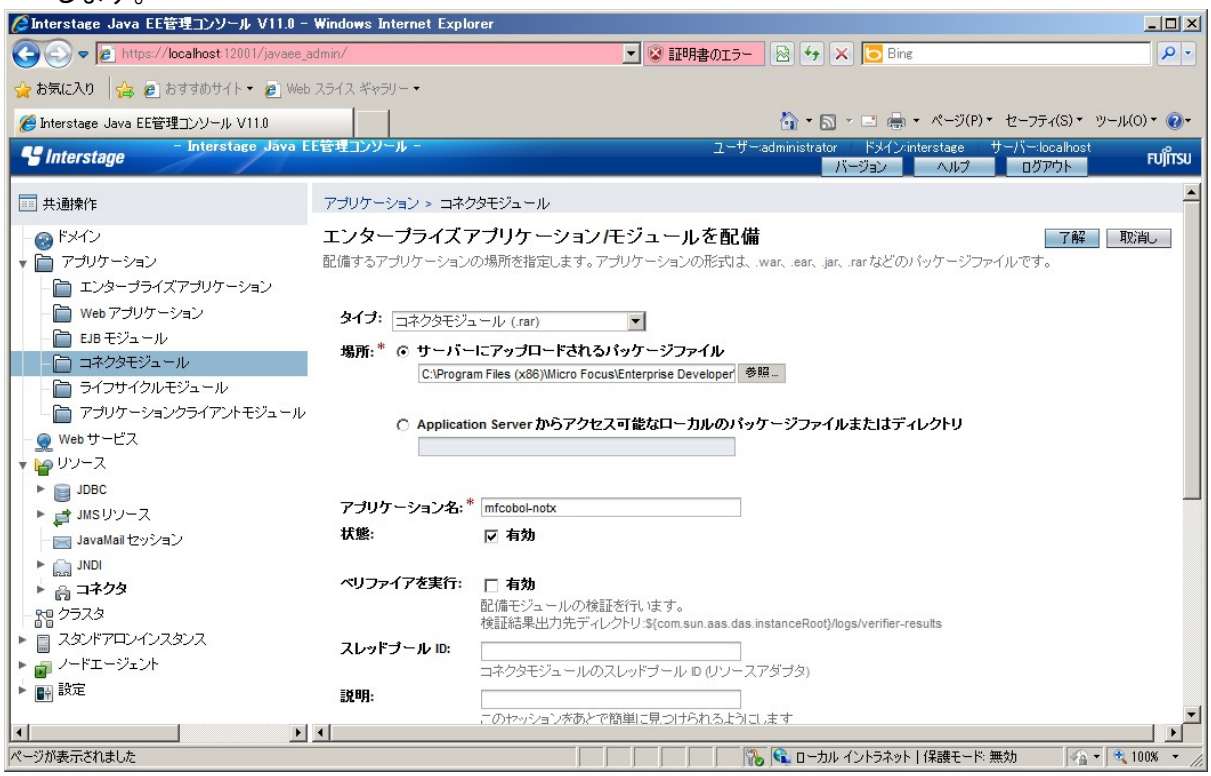
2. リソースアダプタの設定

Visual COBOL 2.1 JのEnterprise ServerへのJCA接続は、WebLogic などのいくつかのJavaEEアプリケーションサーバーで動作保証されており、それらのそれぞれに対応したリソースアダプタが個別に製品に添付されています。これらは基本的に同じ物ですが、アプリケーションサーバーの種類によって必要となるマニフェストやデプロイメントディスクリプタが個別にパッケージ化されています。
今回の検証対象である Interstage AS に対応したものは用意されていませんので、ここでは比較的標準的な内容を持っている WebLogic 向けのリソースアダプタを使用しました。

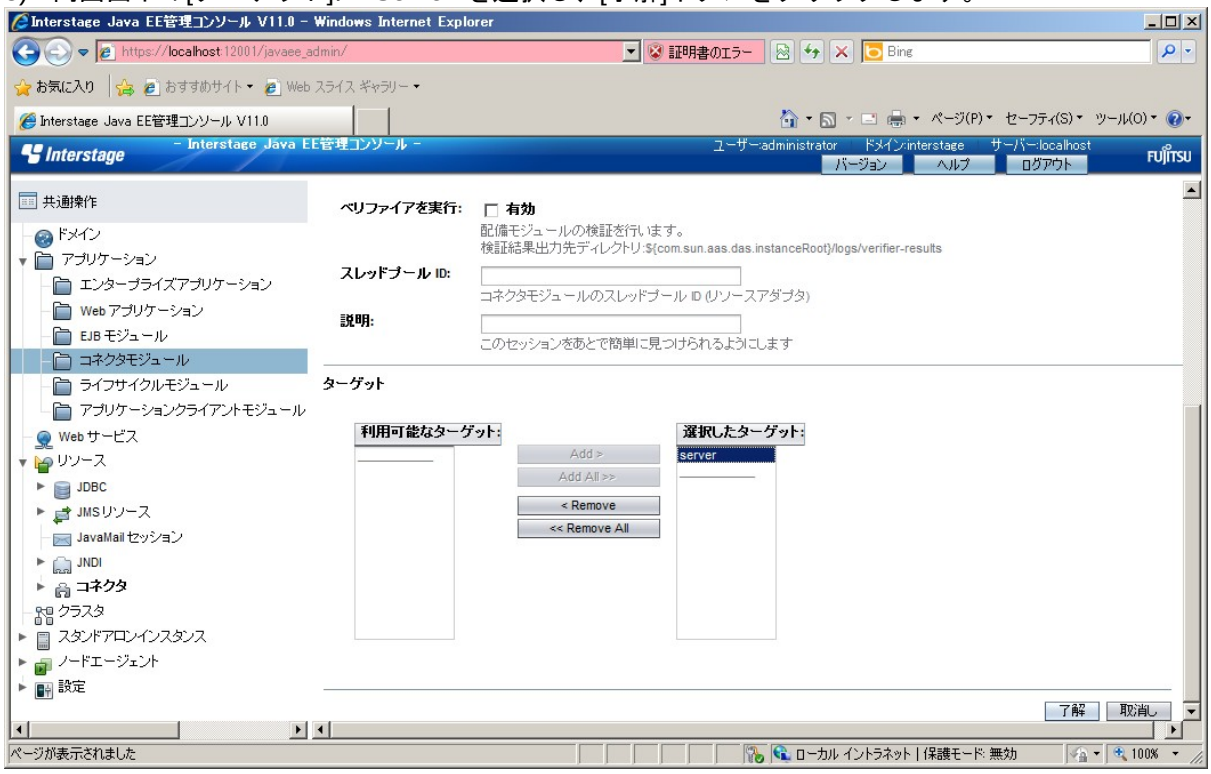
1) Windows PC上の Internet Explorer で https://localhost:12001/javaee_admin を指定し、Interstage AS管理コンソールを開き、ログインすると、下記が表示されます。



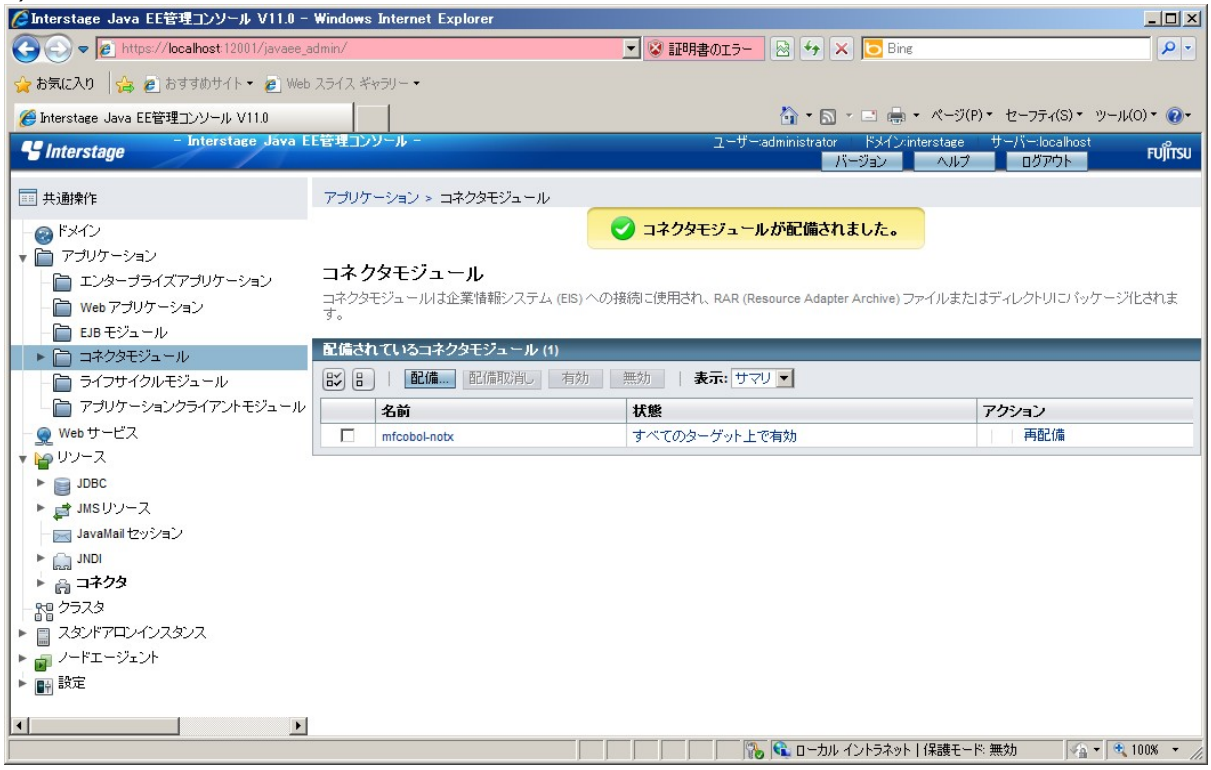
2) 左ペインのツリービューで [アプリケーション] – [コネクタモジュール]をクリックし、[コネクタモジュール] 画面で[配備]ボタンをクリックします。表示された画面で、以下のように[サーバーにアップロードされるパッケージファイル]の[参照]ボタンをクリックし、C:¥Program Files (x86)¥Micro Focus¥Enterprise Developer¥javaee¥j2ee14¥beaweblogic9¥mfcobol-notx.rar を指定します。



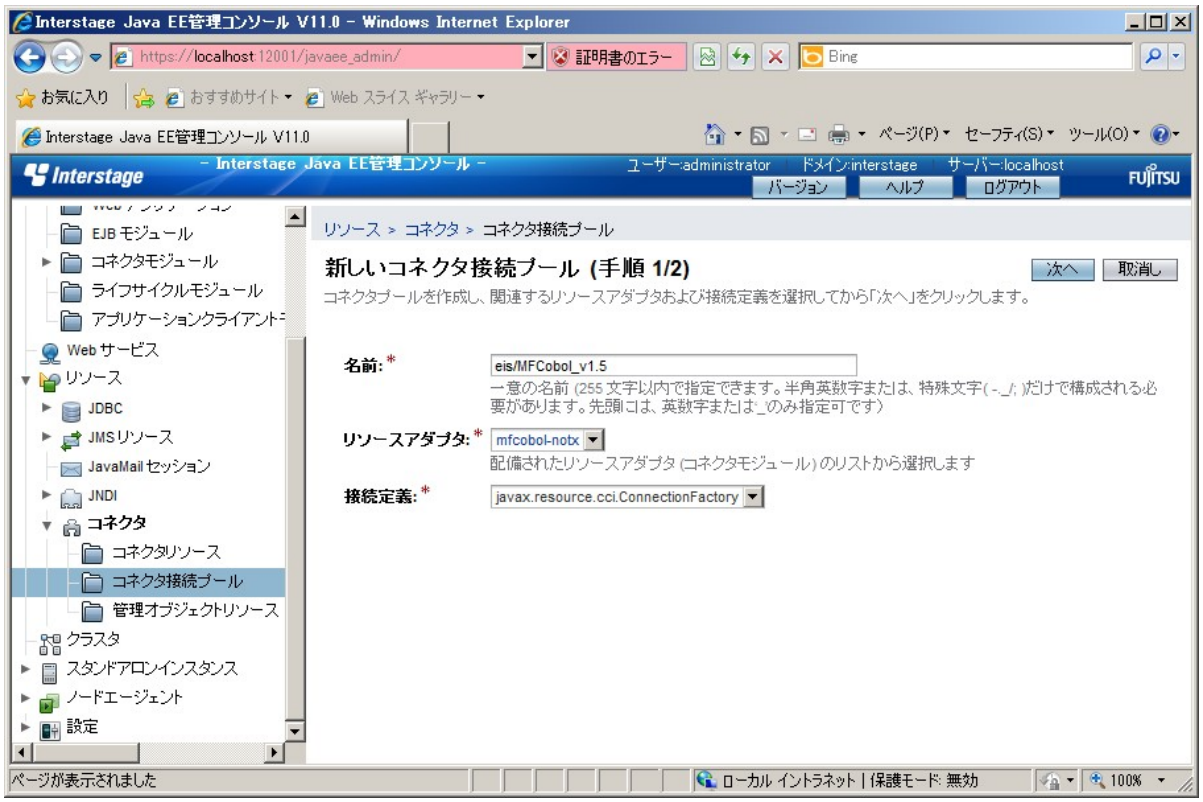
3) 同画面下の[ターゲット]に Server を選択し、[了解]ボタンをクリックします。



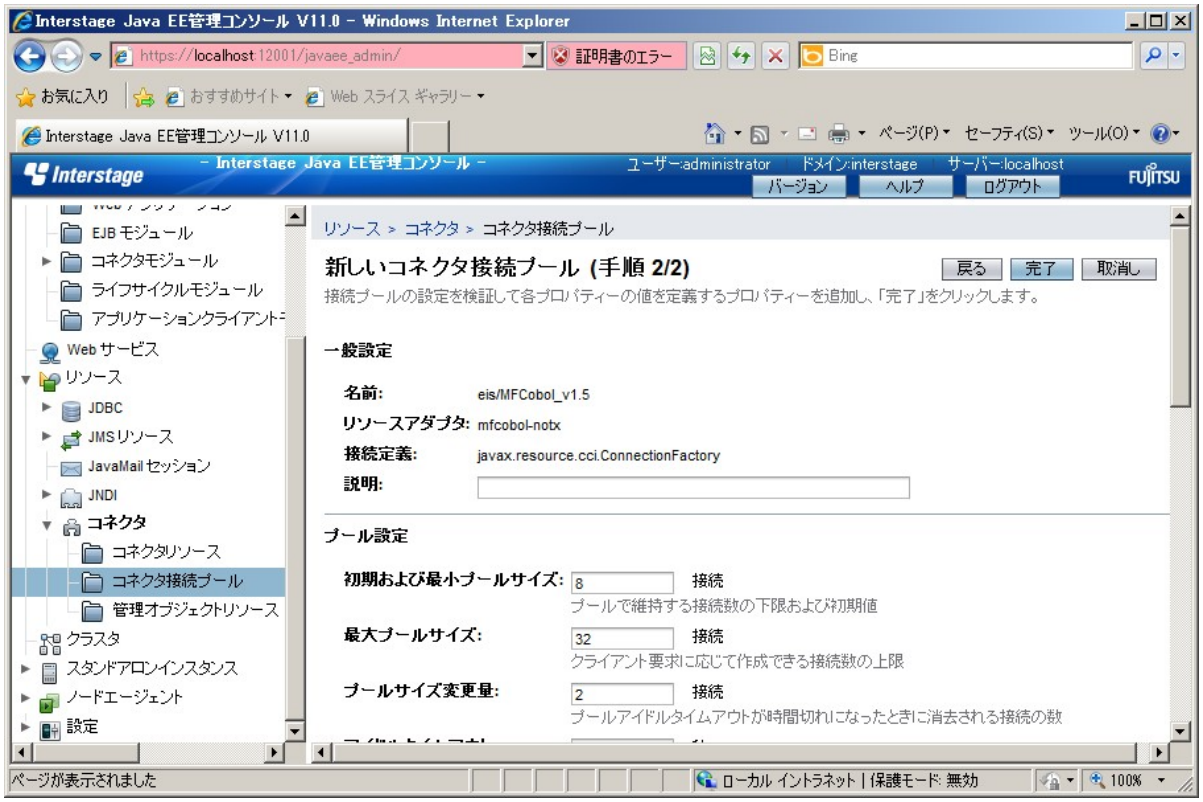
4) 以下のようにリソースアダプタが配備されたことを確認します。



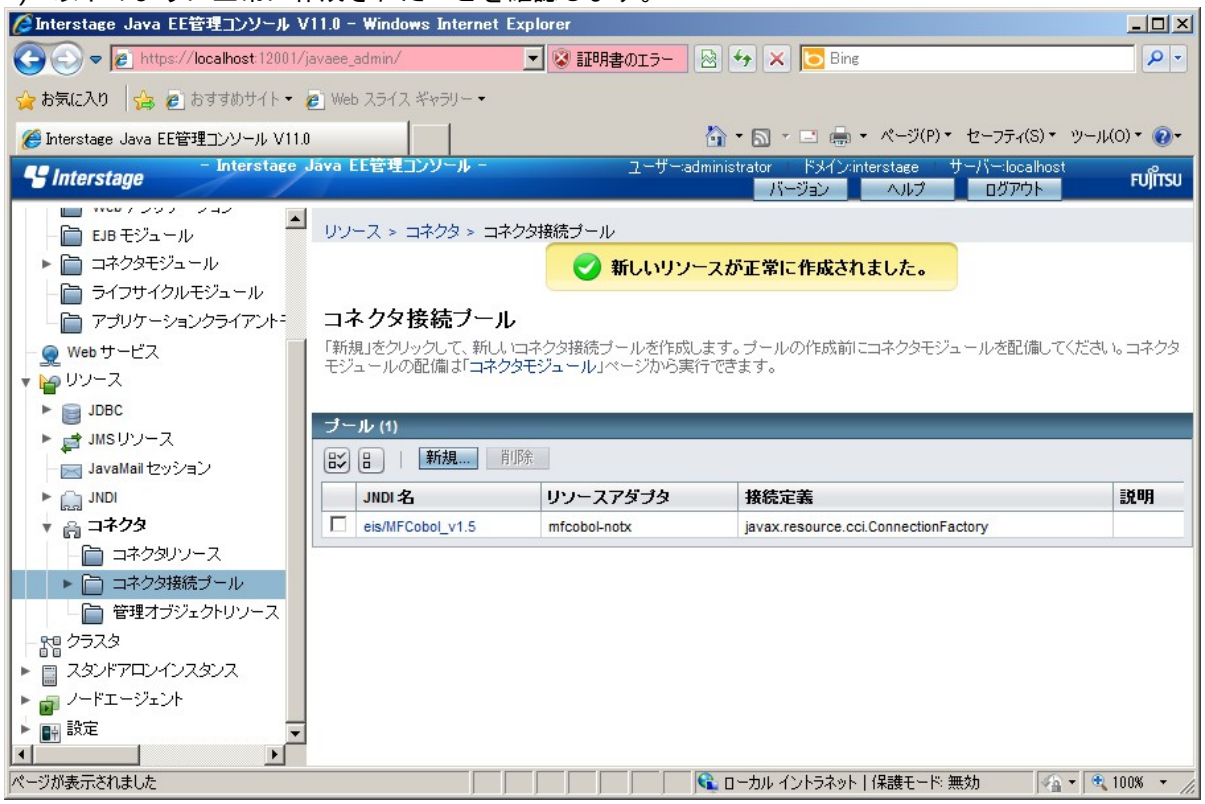
5) 配備されたリソースアダプタのコネクタ接続プールを登録します。管理コンソールのツリービューで [リソース] - [コネクタ接続プール] を選択します。右ペインの [リソースアダプタ] のプルダウンリストから mfcobol-notx を選択し、[名前]に eis/MFCobol_v1.5 と入力します。



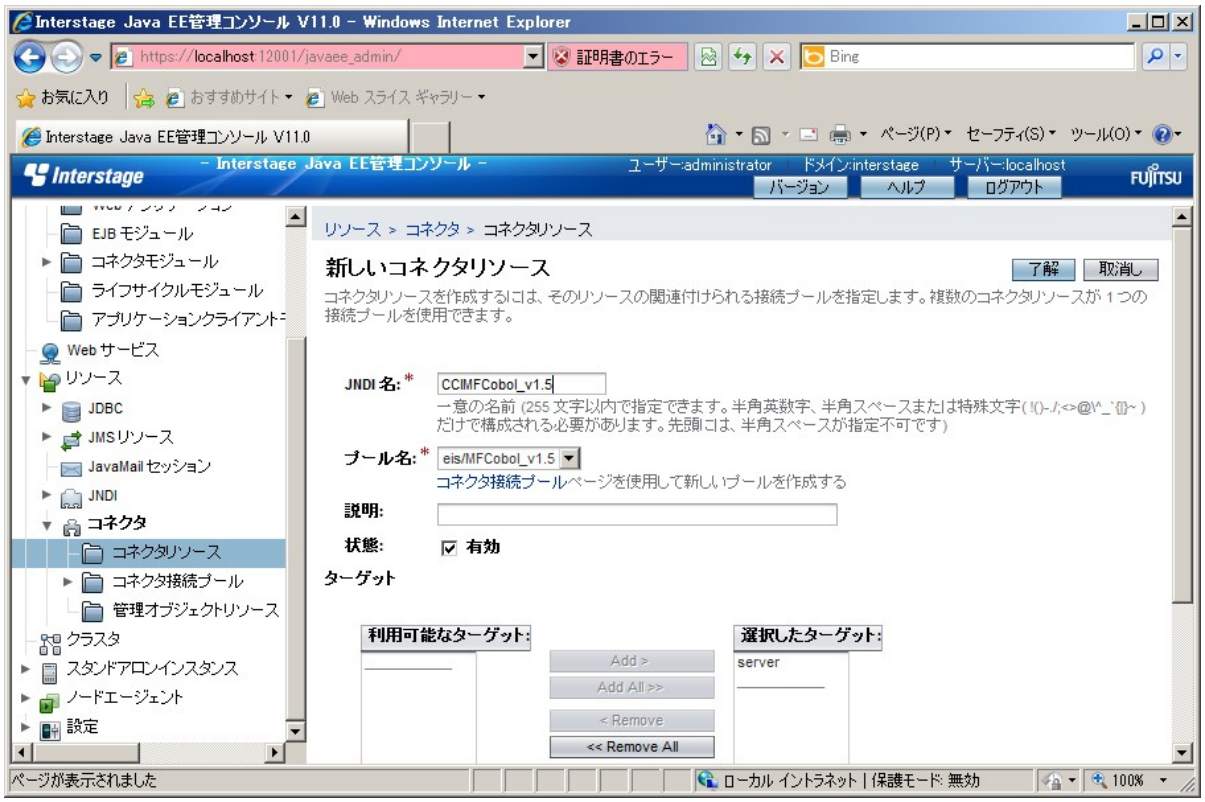
6) [次へ]ボタンをクリックすると、以下が表示されますが、ここでは、そのまま [完了]ボタンをクリックします。



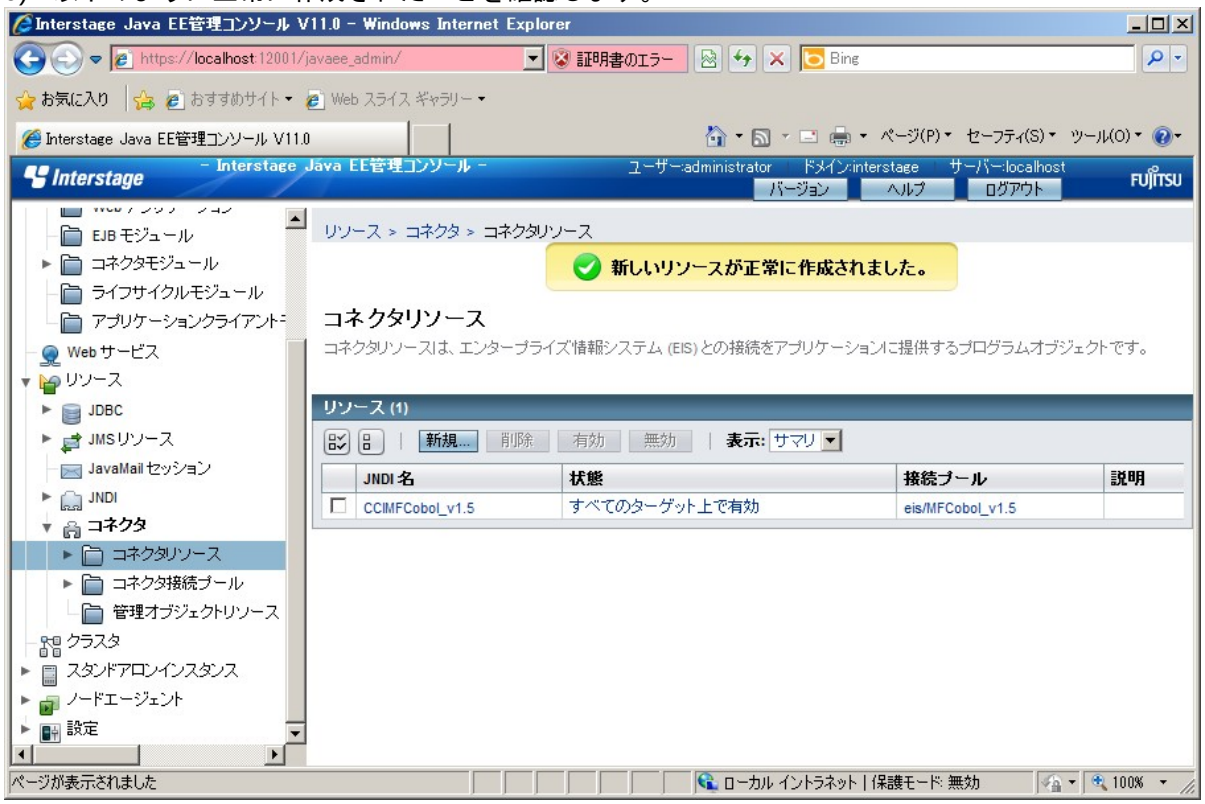
7) 以下のように正常に作成されたことを確認します。



8) 管理コンソールの [リソース] - [コネクタリソース] を選択します。右ペインの [JNDI 名] に CCIMFCobol_v1.5 と入力し、[了解]ボタンをクリックします。（注 : 本来の JNDI 名および接続プール名の指定とは逆になっている模様）



9)　以下のように正常に作成されたことを確認します。



以上で、リソースアダプタの配備は完了しました。

3. COBOLプログラムの準備

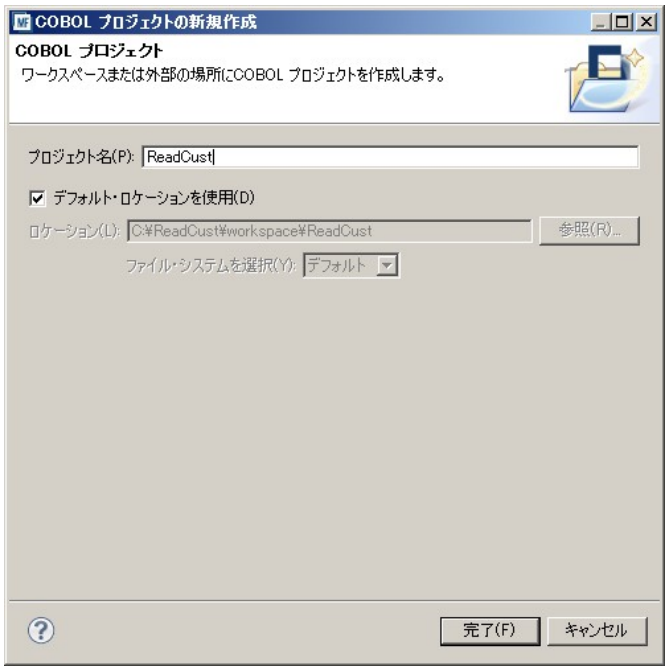
ここでは、検証で使用する COBOL プログラムを Enterprise Server に配備し、同時に EJB ラッパーを自動生成します。

3-1. プロジェクトの新規作成

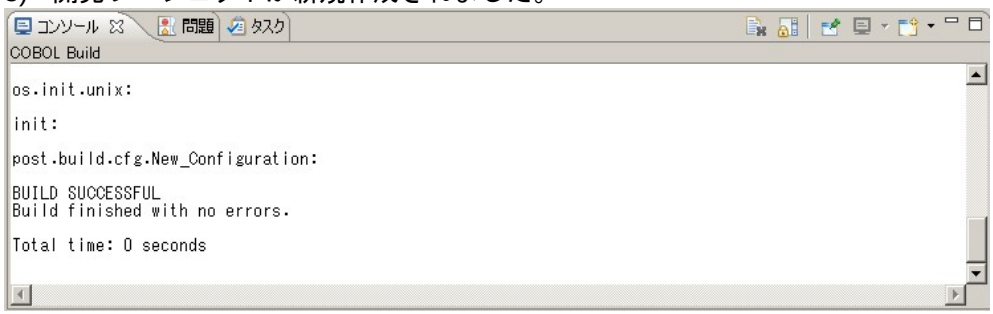
COBOL 開発プロジェクトを新規作成します。
今回は、例題として使用する COBOL プログラムを C:¥ReadCust の下に用意しました。

1)　[ファイル]メニュー > [新規] > [COBOL プロジェクト]を選択します。

2)　[プロジェクト名]欄に「ReadCust」と入力し[完了]ボタンをクリックします。



3)　開発プロジェクトが新規作成されました。

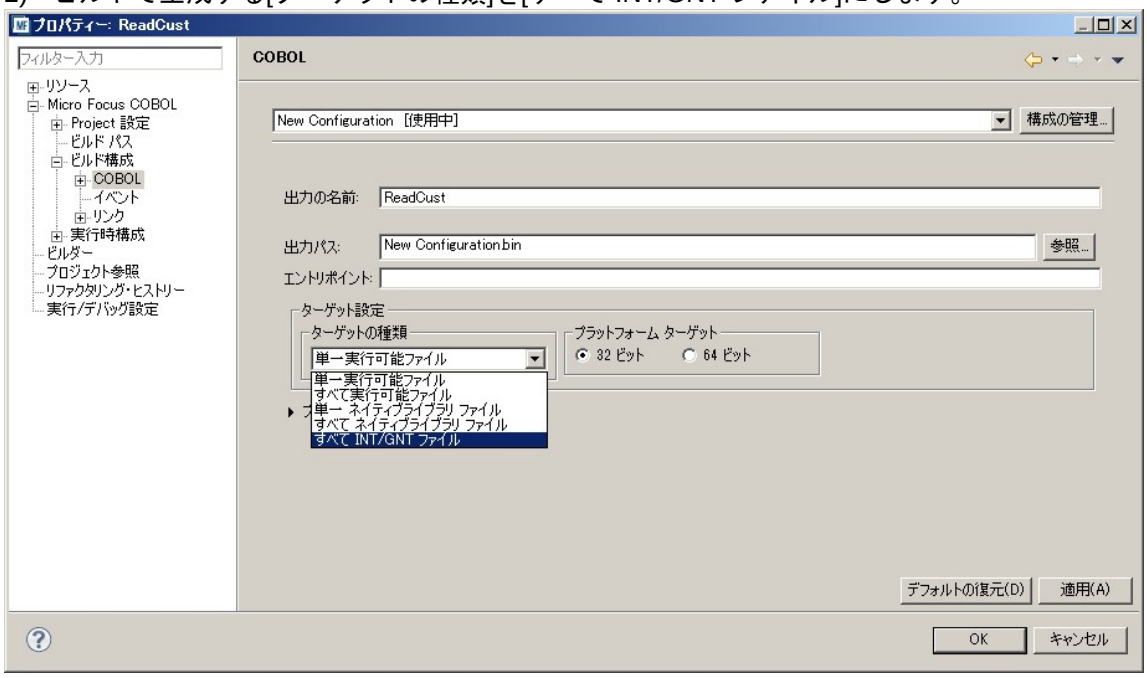


3-2. プロジェクトへメンバーを追加及びコンパイル

[COBOL エクスプローラ]に表示されるプロジェクトのツリーに現在 COBOL プログラムが追加されていない状態のため、プログラムソース等を追加します。

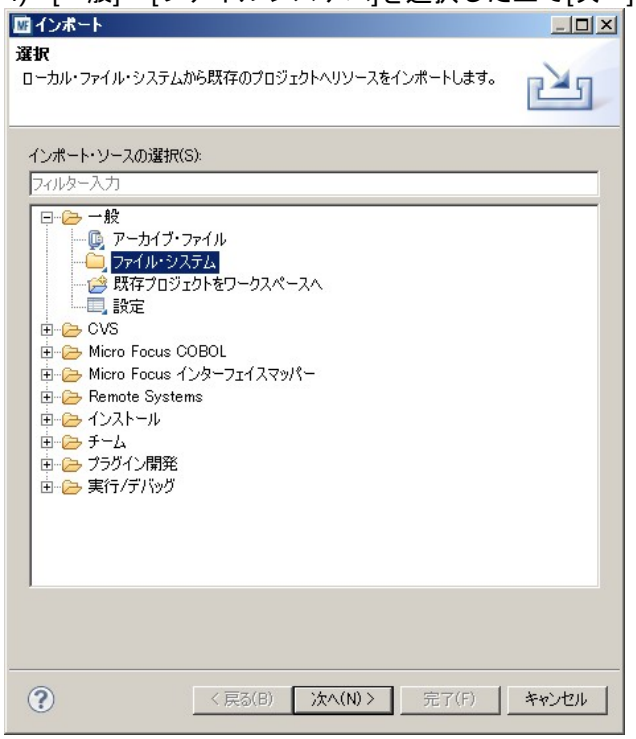
1)　ReadCust プロジェクトの[プロパティー]をクリックします。

2)　ビルドで生成する[ターゲットの種類]を[すべて INT/GNT ファイル]にします。

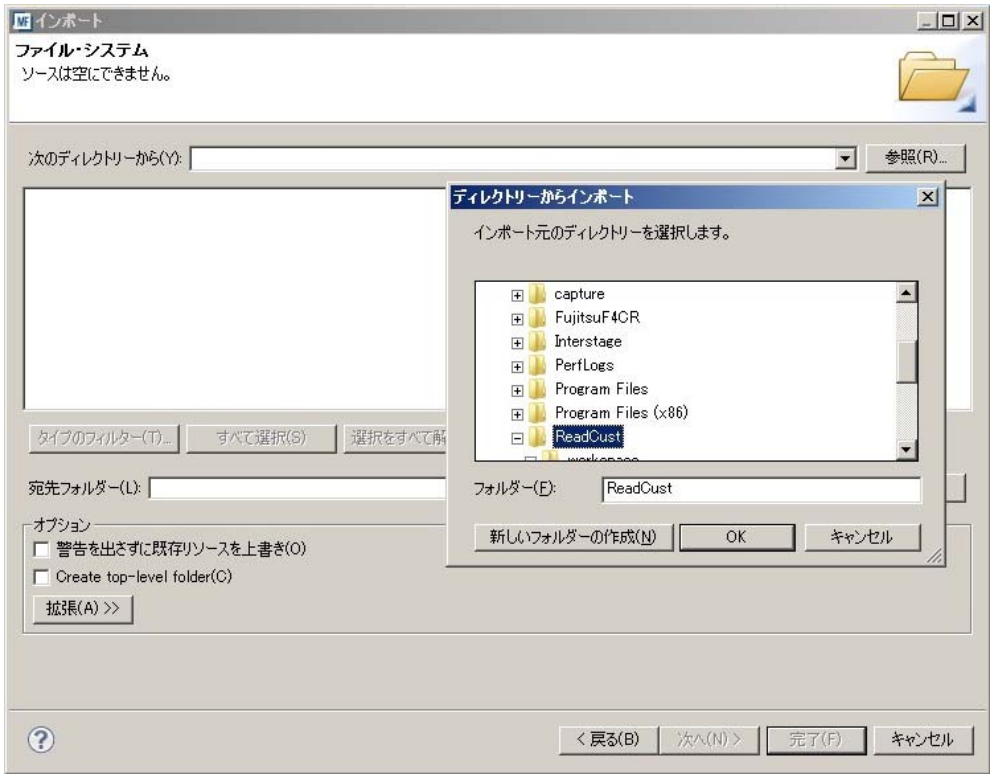


3)　[ファイル]メニュー > [インポート] を選択します。

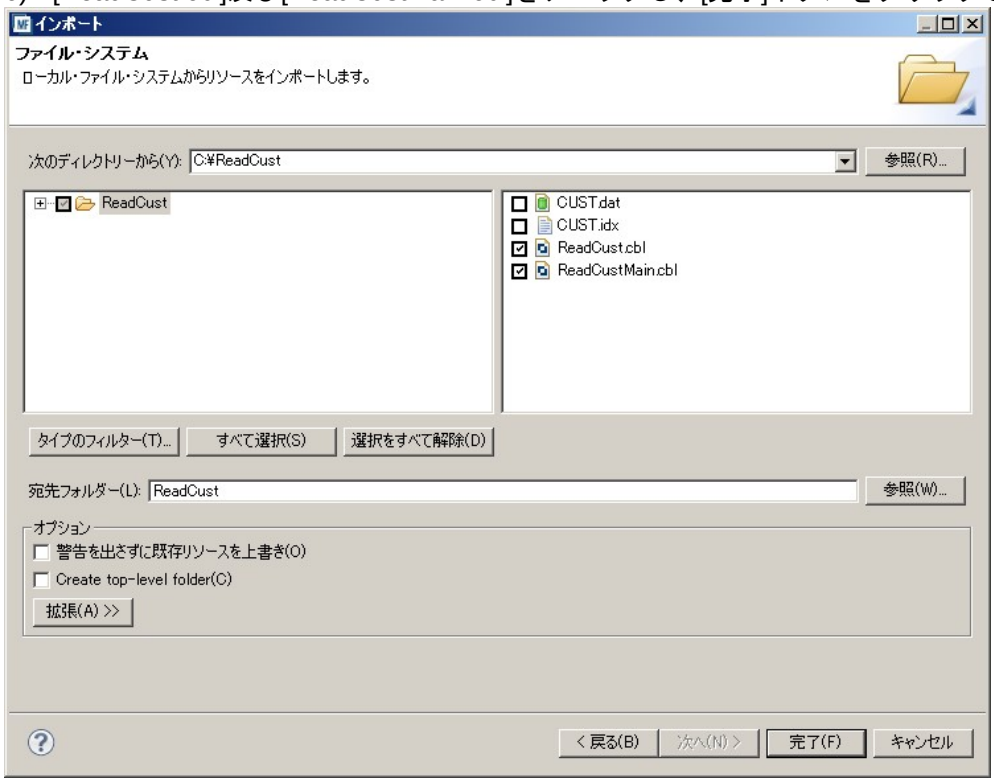
4)　[一般] > [ファイルシステム]を選択した上で[次へ]ボタンをクリックします。



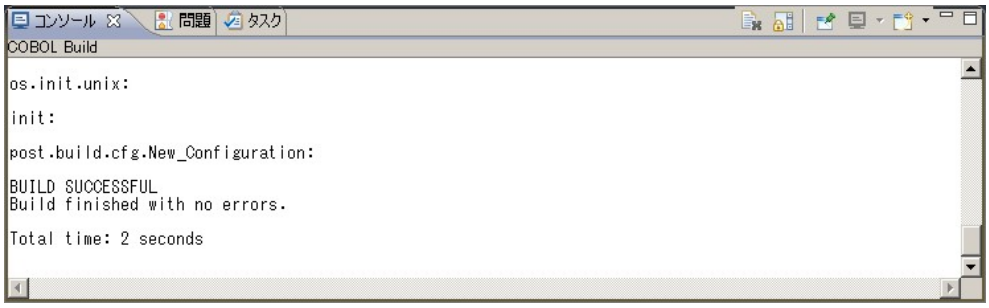
5)　[参照]ボタンをクリックし[C:¥ReadCust]を選択します。



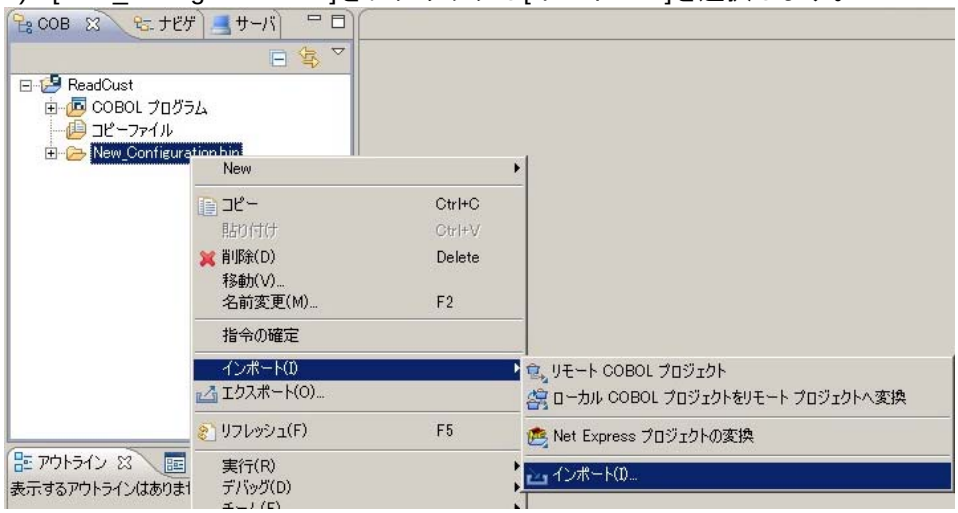
6)　[ReadCust.cbl]及び[ReadCustMain.cbl]をチェックし、[完了]ボタンをクリックします。



7)　右下の[コンソール]画面にてビルドが正常に終了していることを確認します。



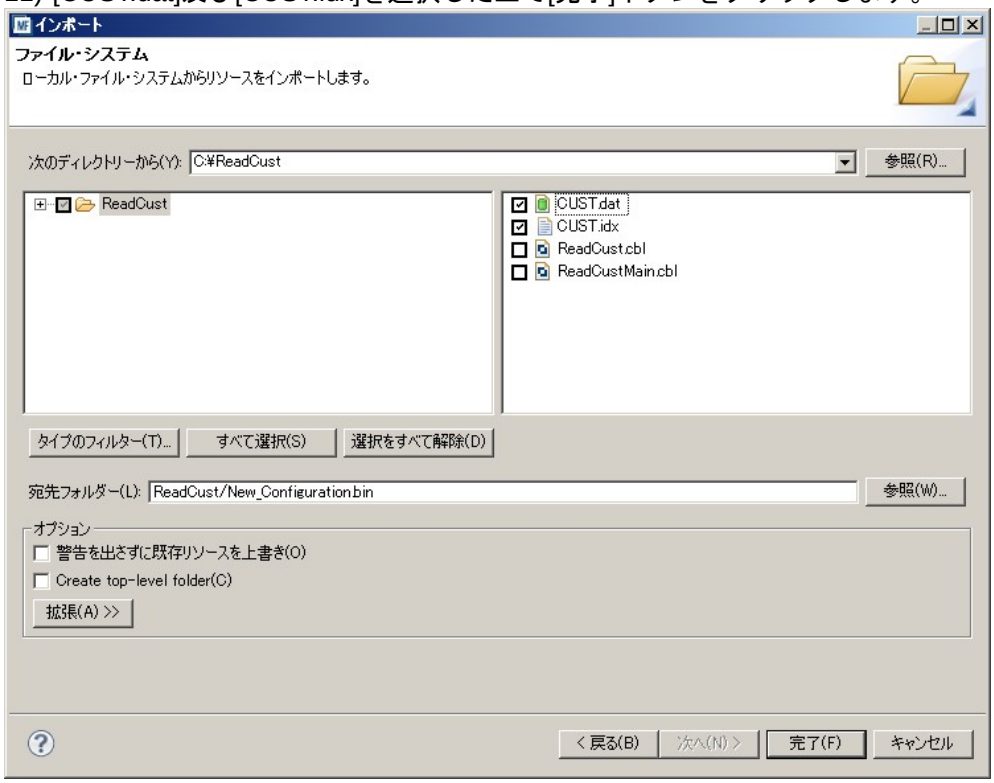
8)　[New_Configuration.bin]を右クリックし[インポート]を選択します。



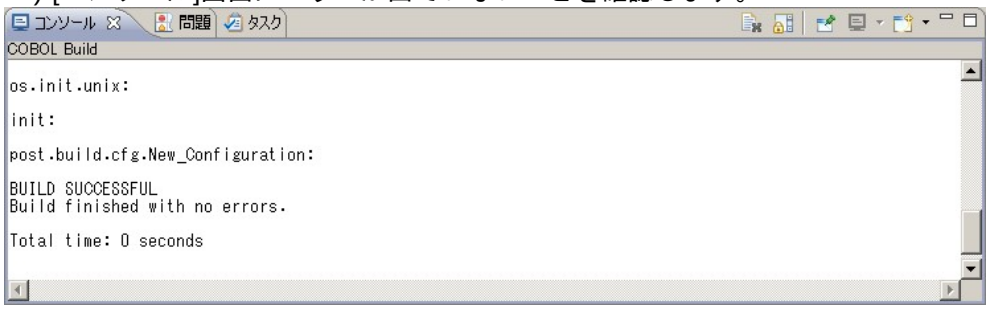
9)　4)の手順同様、[一般] > [ファイルシステム]を選択した上で[次へ]ボタンをクリックします。

10) 5)の手順同様[参照]ボタンをクリックし[C:¥ReadCust]を選択します。

11) [CUST.dat]及び[CUST.idx]を選択した上で[完了]ボタンをクリックします。



12) [コンソール]画面にエラーが出ていないことを確認します。



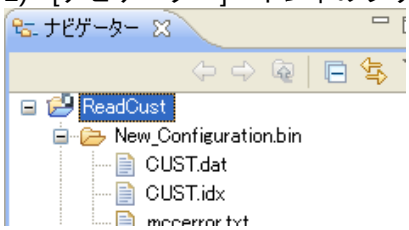
3-3. JCA のインタフェースマッピングの作成

JavaEE Connector としての配備に進みます。ReadCust.cbl のビジネスロジックをそのまま再利用して JCA サービス化します。

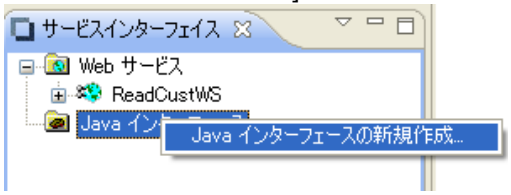
まず　ReadCust.cbl の LINKAGE パラメタに対するインタフェースマッピングを作成します。

1)　[インターフェースマッパー]パースペクティブへ切り替えます。

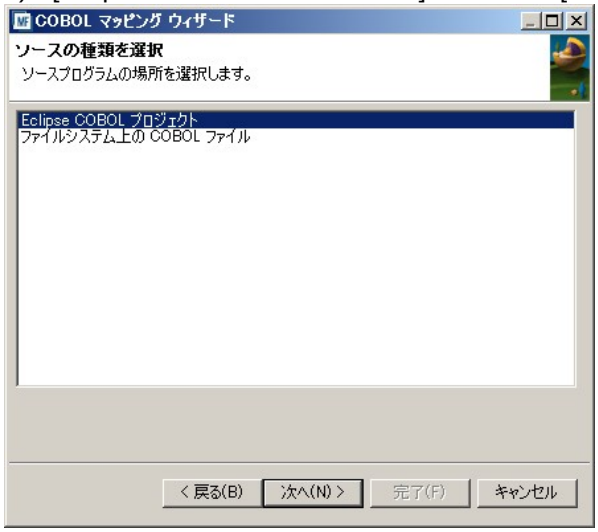
2)　[ナビゲーター]ペイン中のツリーにて[ReadCust]を選択します。



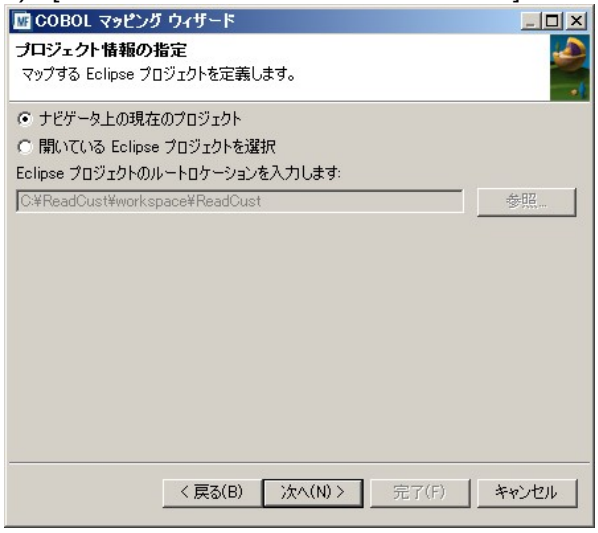
3)　[サービスインターフェース]ペイン中の[Java インターフェース]を右クリックし、[Java インターフェースの新規作成]を選択します。



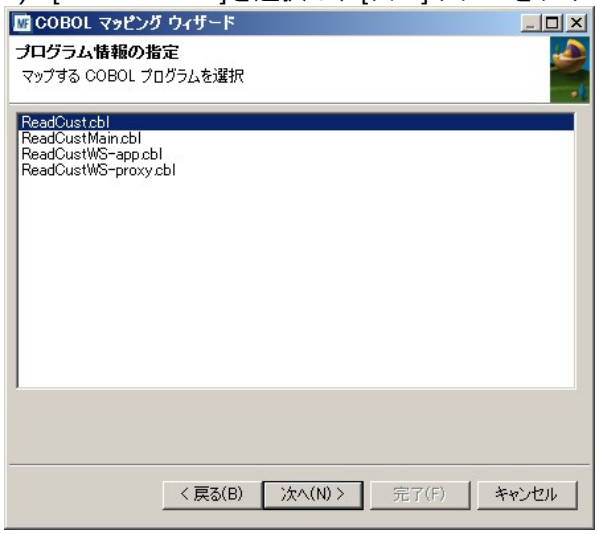
4)　[Eclipse COBOL プロジェクト]を選択し、[次へ]ボタンをクリックします。



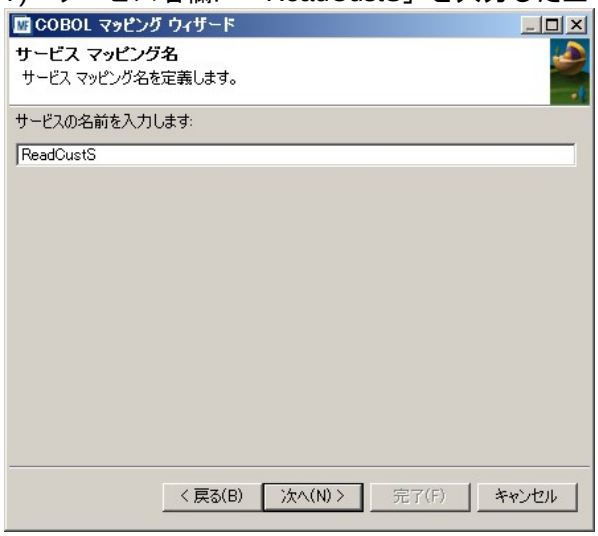
5)　[ナビゲーター上の現在のプロジェクト]を選択した上で[次へ]ボタンをクリックします。



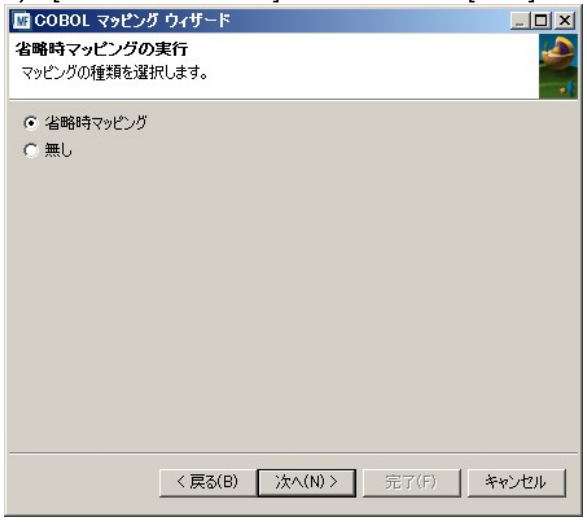
6)　[ReadCust.cbl]を選択し、[次へ]ボタンをクリックします。



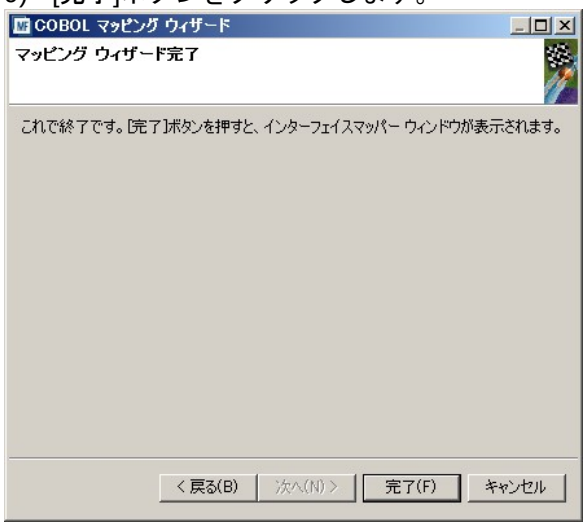
7)　サービス名欄に「ReadCustS」と入力した上で、[次へ]ボタンをクリックします。



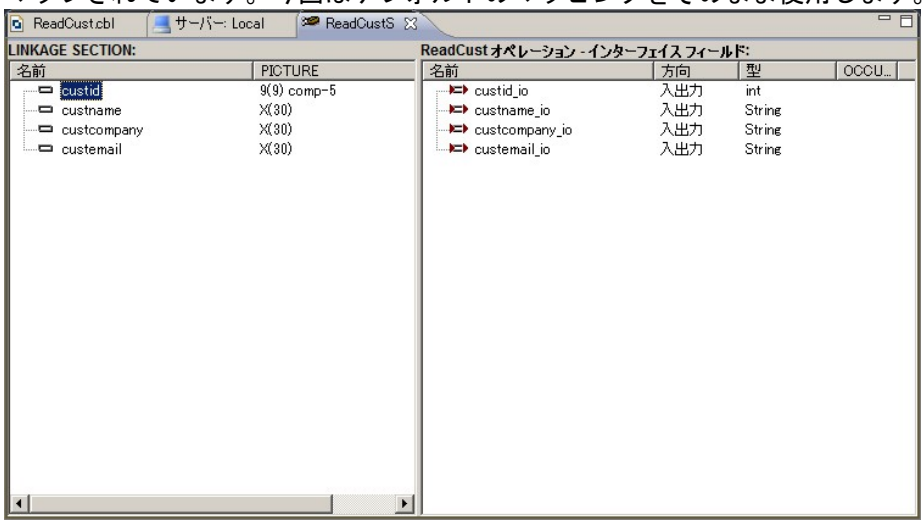
8)　[省略時マッピング]を選択した上で[次へ]ボタンをクリックします。



9)　[完了]ボタンをクリックします。



10) 以下のようなマッピングツールが表示されます。
左側のペインには、マッピングの対象となる　COBOL プログラムの LINKAGE パラメタの一覧が表示されています。右側のペインではそれらのパラメタを　Java 側からどのようなインターフェースで見せるかを定義します。EJB マッピングのデフォルトでは、すべての基本項目パラメタが[入出力]としてマップされています。今回はデフォルトのマッピングをそのまま使用します。

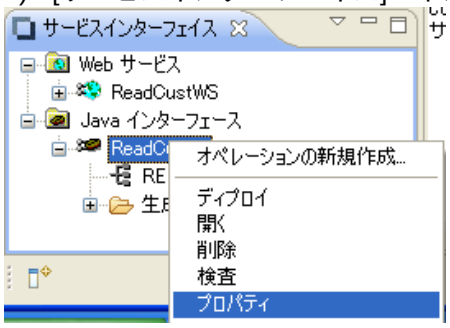


11) [サービスインターフェイス]ペイン中にて ReadCustS が追加されたことを確認します。

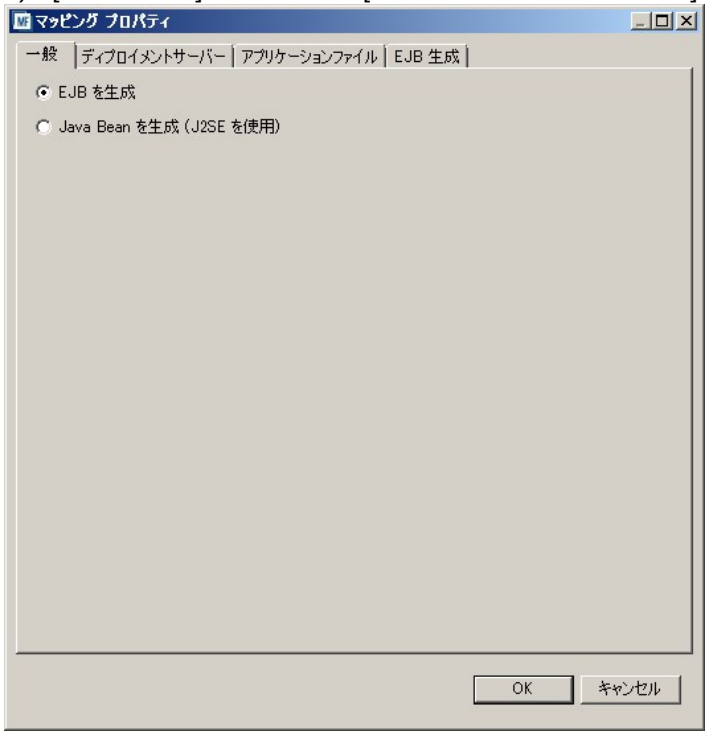
以上で EJB マッピングのサービスインターフェースの定義が完了しました。

3-4. JCA マッピングのディプロイ

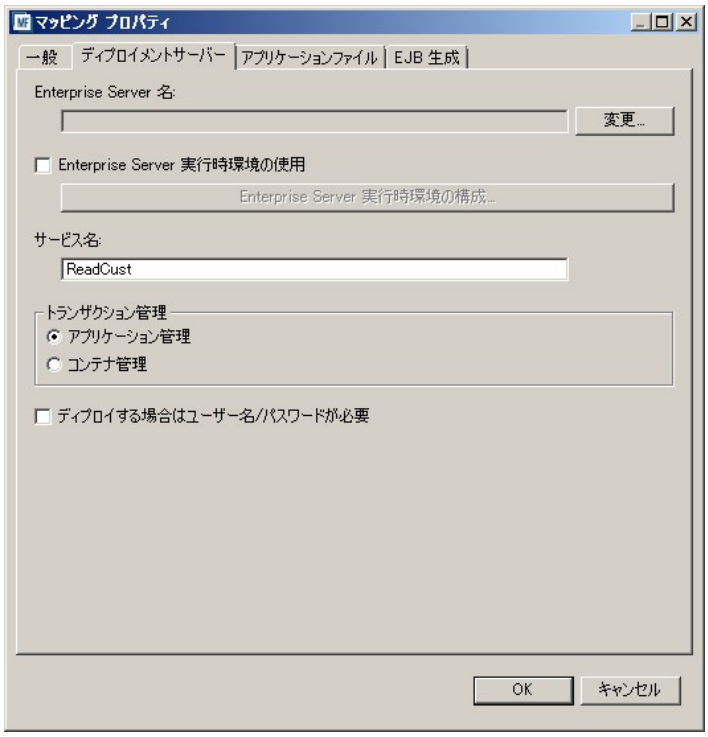
1) [サービスインターフェイス]ペインにて[ReadCustS]を右クリックし[プロパティ]を選択します。



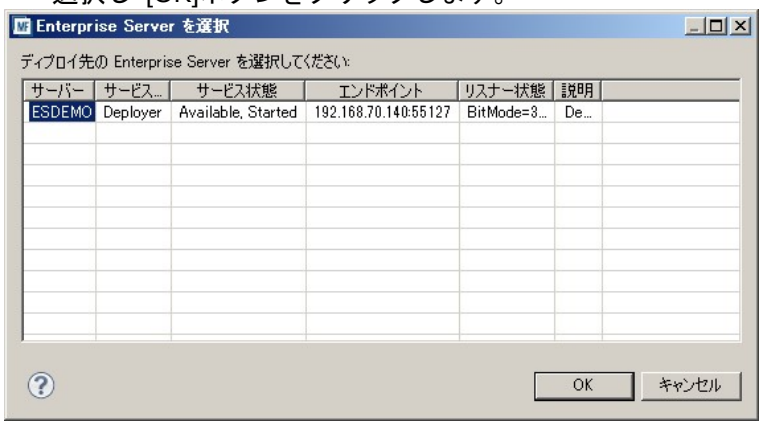
2) [EJB を生成]を選択の上、[ディプロイメントサーバー]タブをクリックします。



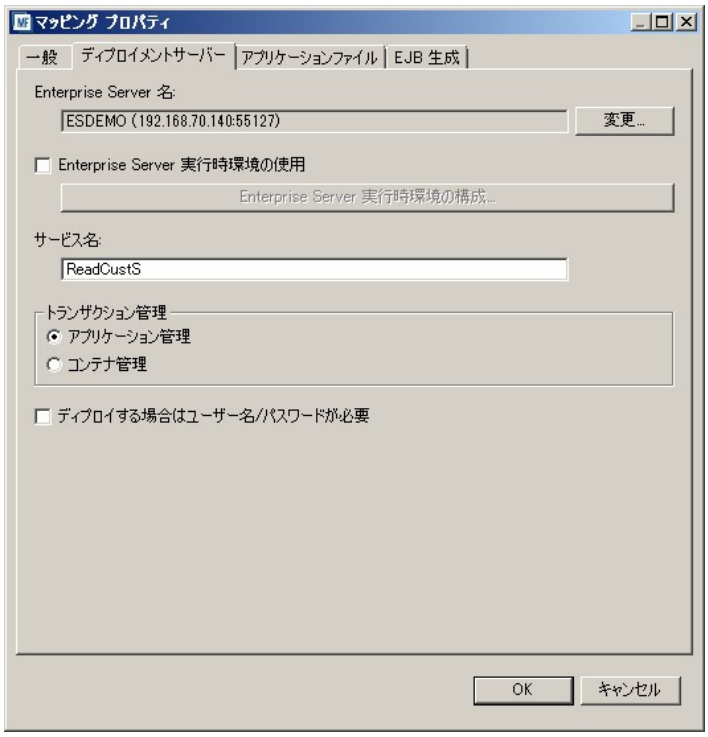
3) [変更]ボタンをクリックします。



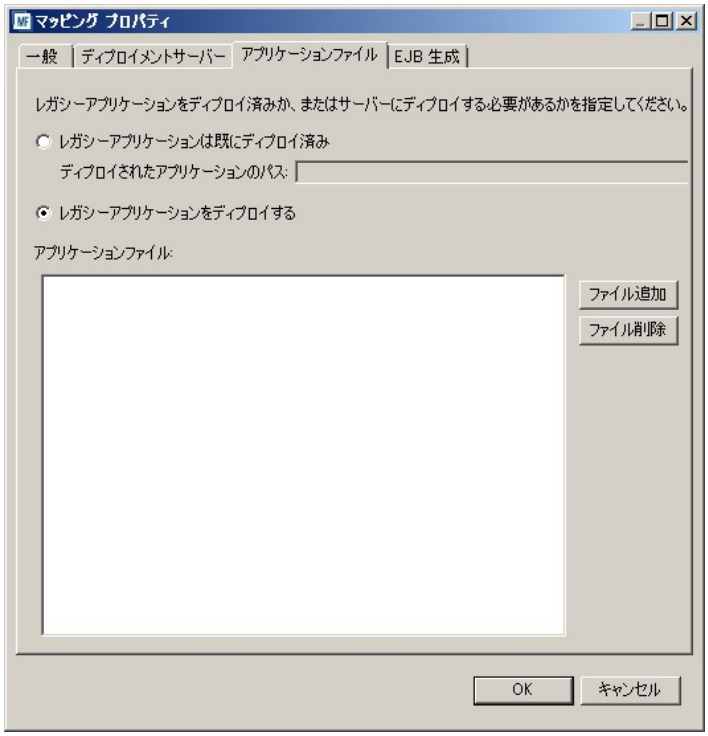
4) 3)により開発環境から参照可能な Enterprise Server の一覧が表示されています。既に開始している ESDEMO というサーバーインスタンスが見えており、[Started] の状態となっています。これを選択し [OK]ボタンをクリックします。



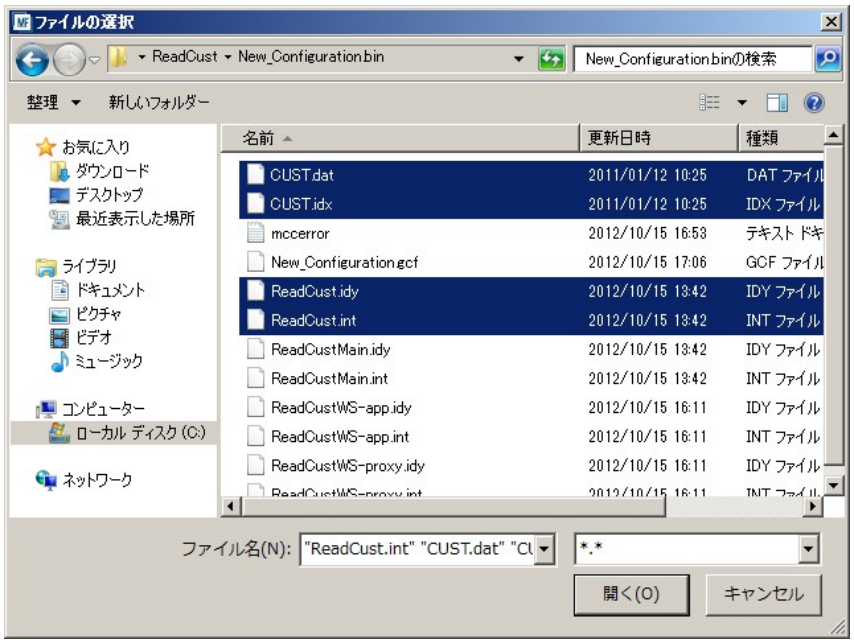
5) [サービス名]欄に「ReadCustS」を入力の上、[アプリケーションファイル]タブをクリックします。



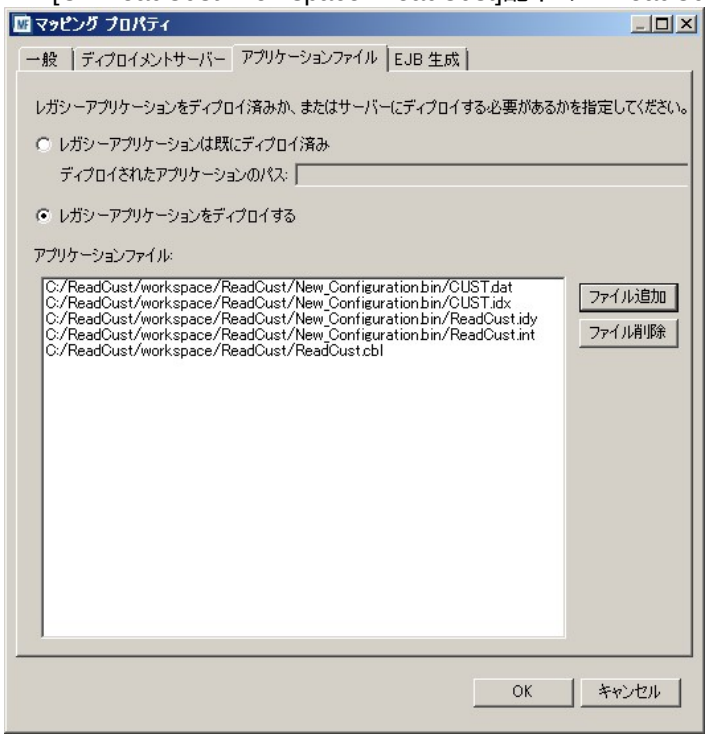
6) [レガシーアプリケーションをディプロイする]を選択の上[ファイルを追加]ボタンをクリックします。



7) [C:¥ReadCust¥workspace¥ReadCust¥New_Configuration.bin]下の「CUST.dat」、「CUST.idx」、「ReadCust.idy」、「ReadCust.int」を[Ctrl]キーを打鍵しつつ選択し、[開く]ボタンをクリックします。



8) アプリケーションファイル欄にそれらのファイルが追加されます。同じ要領で [C:¥ReadCust¥workspace¥ReadCust]配下の「ReadCust.cbl」を追加します。

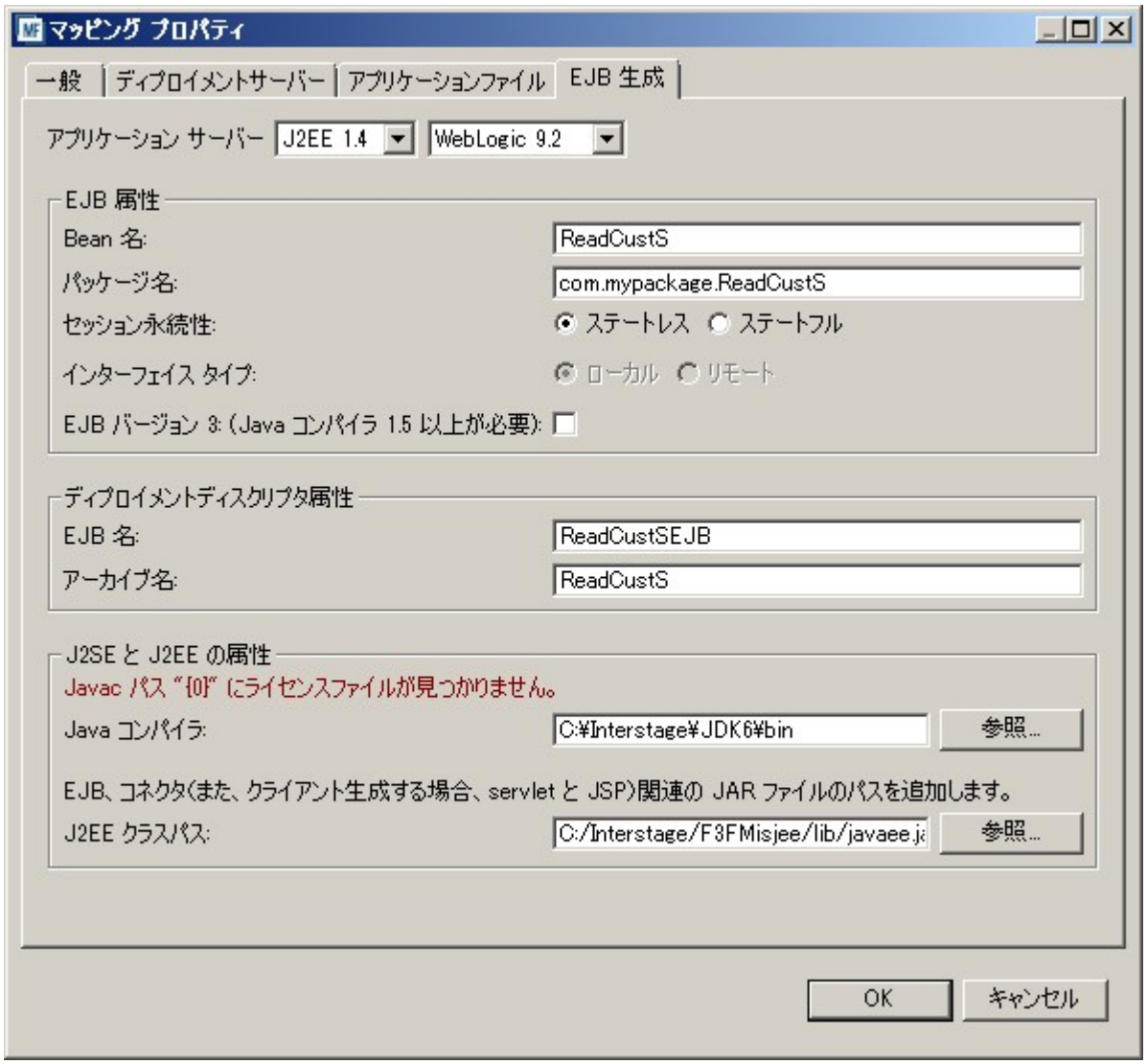


9) [EJB を生成]タブをクリックします。

10) [アプリケーションサーバー]欄にて、[J2EE 1.4]及び[WebLogic 9.2]を選択します。

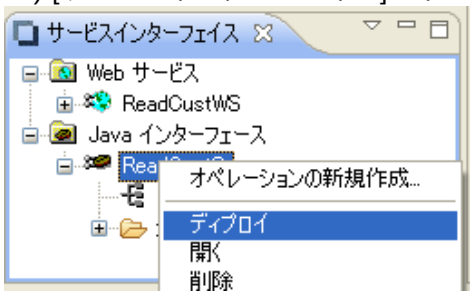
11) [Java コンパイラ]欄右の[参照]ボタンをクリックし、“C:¥Interstage¥JDK6¥bin” にナビゲートしてそのパスを入力します。

12) [J2EE クラスパス]欄右の[参照]ボタンをクリックし、 “C:/Interstage/F3FMisjee/lib” にナビゲートして、使用する JavaEE の提供する javaee.jar を指定します。



13) [OK]ボタンをクリックします。

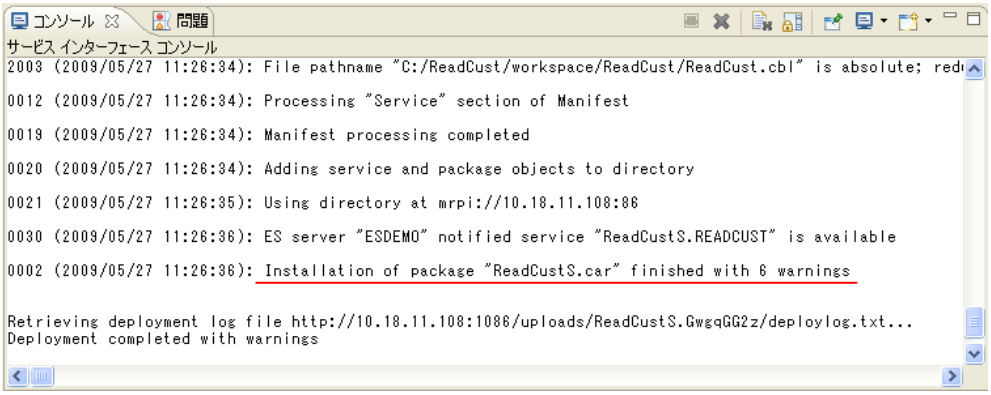
14) [サービスインターフェイス]ペイン中の[ReadCustS]を右クリックし[ディプロイ]を選択します。



ここでは以下の 2 点について処理されます：

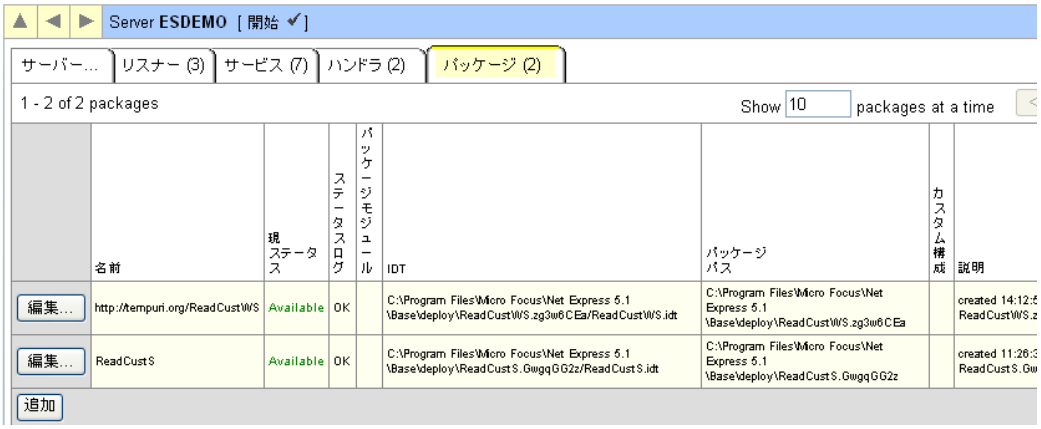
① 指定された Enterprise Server への COBOL サービスのディプロイ
② ディプロイされた COBOL サービスを呼び出すためのラッパーEJB の生成

処理が正常に完了しますと完了した旨のメッセージが以下のようにサービスインターフェースコンソールに表示されます。

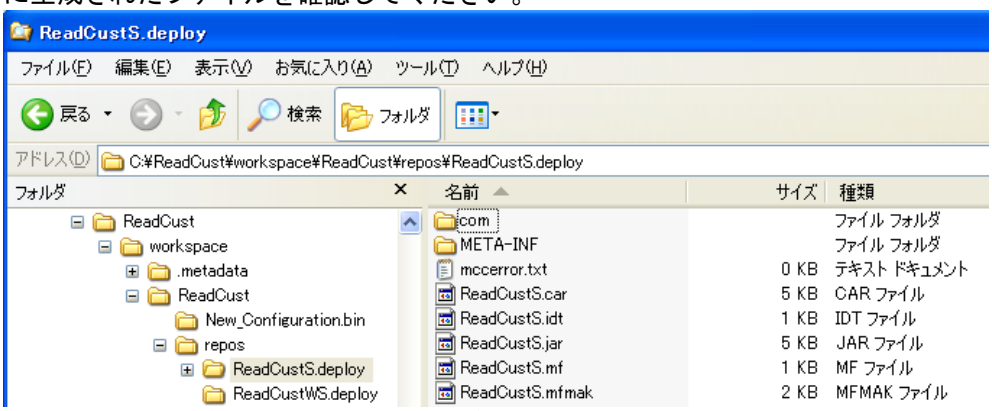


3-5. ディプロイ結果の確認

Enterprise Server の管理ツールで ESDEMO の[パッケージ]の詳細表示で、今ディプロイされたサービス ReadCustS が追加されていることを確認することができます。



Windows のエクスプローラで、C:¥ReadCust¥workspace¥ReadCust¥repos¥ReadCustS.deploy の下に生成されたファイルを確認してください。



以下のファイルが生成されていることがわかります:

ReasCustS.jar : ディプロイされたサービスを呼び出す EJB パッケージ
ReadCustS.car: COBOL サービスのディプロイメントパッケージ。これが Enterprise Server にファイル転送されて配備済み

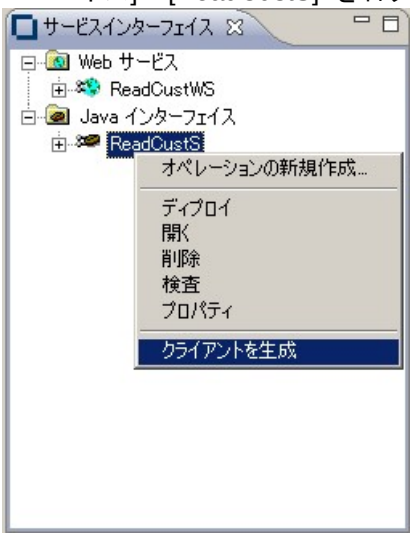
以上で、レガシーCOBOL プログラムは、JCA サービスとして利用可能な状態で用意ができました。

4. テスト用 Webクライアントの生成

インターフェイスマッピングツールキットのクライアント生成機能を使用すると、 対話型でパラメタの値を受け取り、EJB のメソッドを呼び出して結果を表示するような、 簡単な Servlet モジュールを

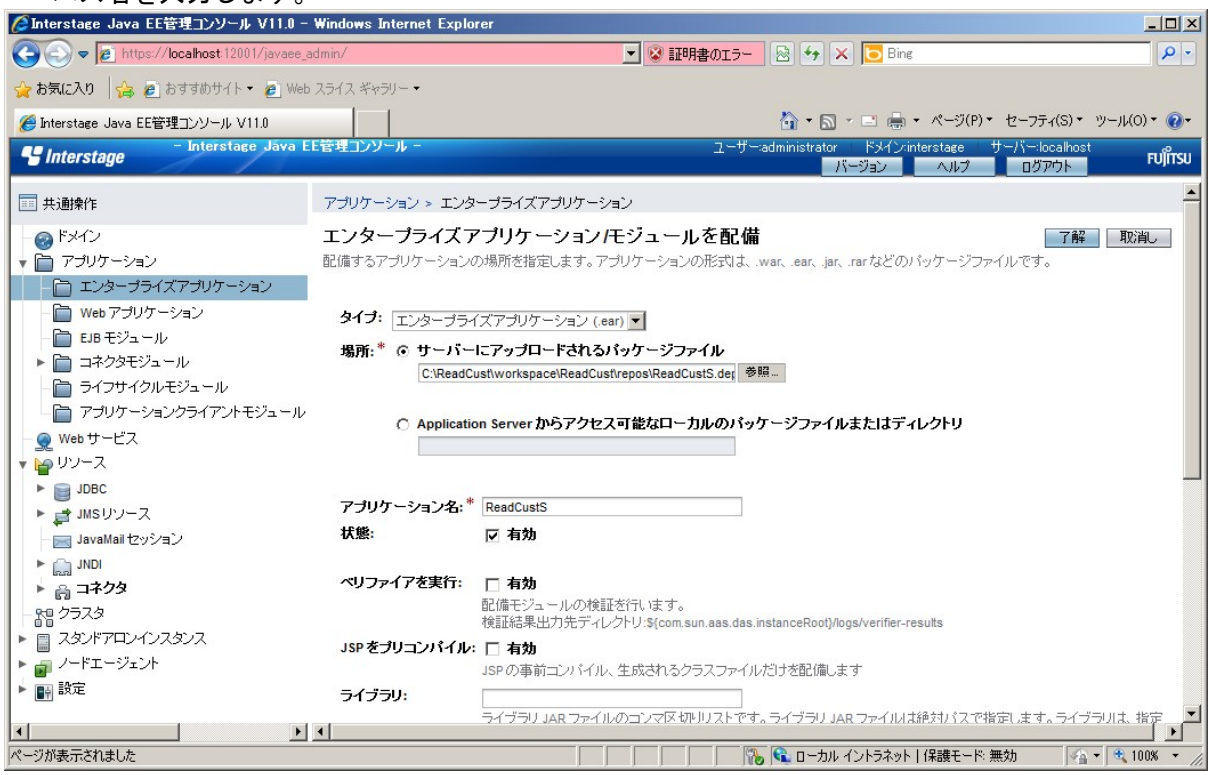
含む、.ear パッケージを作成できます。 ここでは、これを使用して Interstage AS 上の Web クライアントからの呼び出しを行います。

1) COBOL サービスを配備したインターフェイスマッピングツールキットに戻り、[Java インターフェイス] > [ReadCustS] を右クリックして [クライアント生成] を選択します。

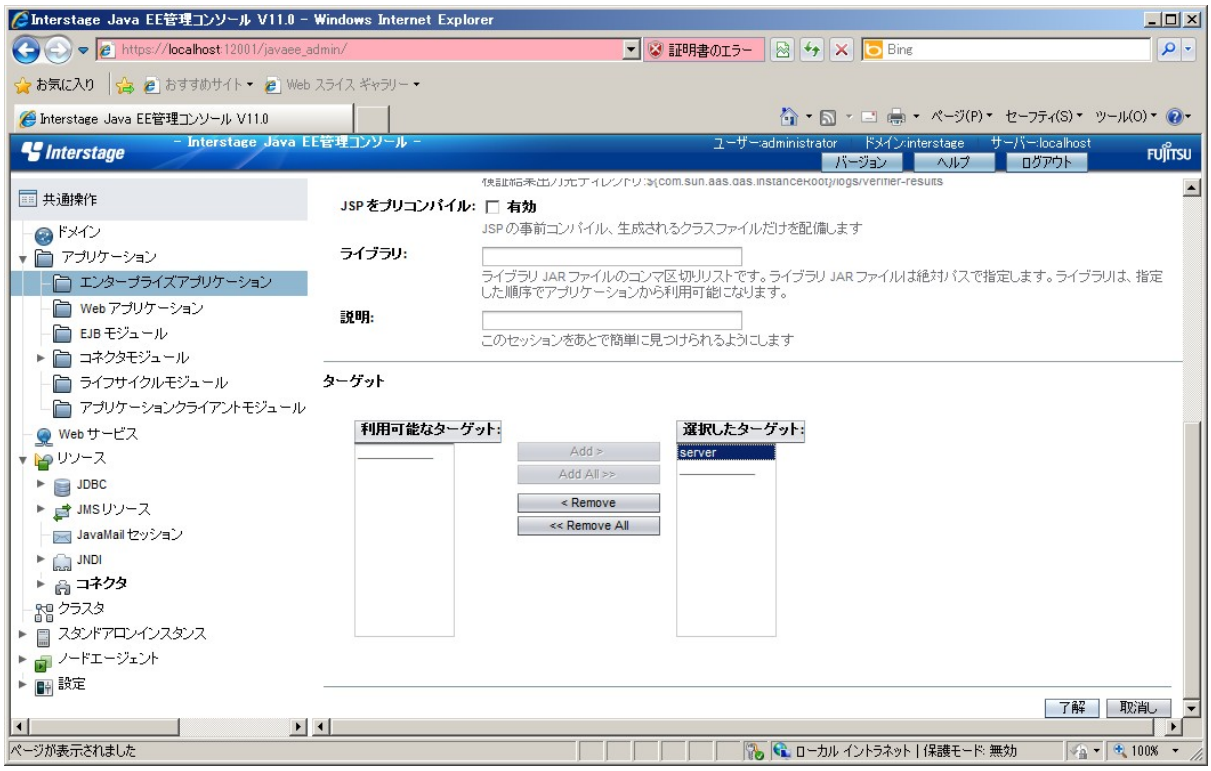


2) この時作業用ディレクトリの C:¥ReadCust¥workspace¥ReadCust¥repos¥ReadCustS.deploy にパッケージ ReadCustS.ear が自動生成されていますので確認します。

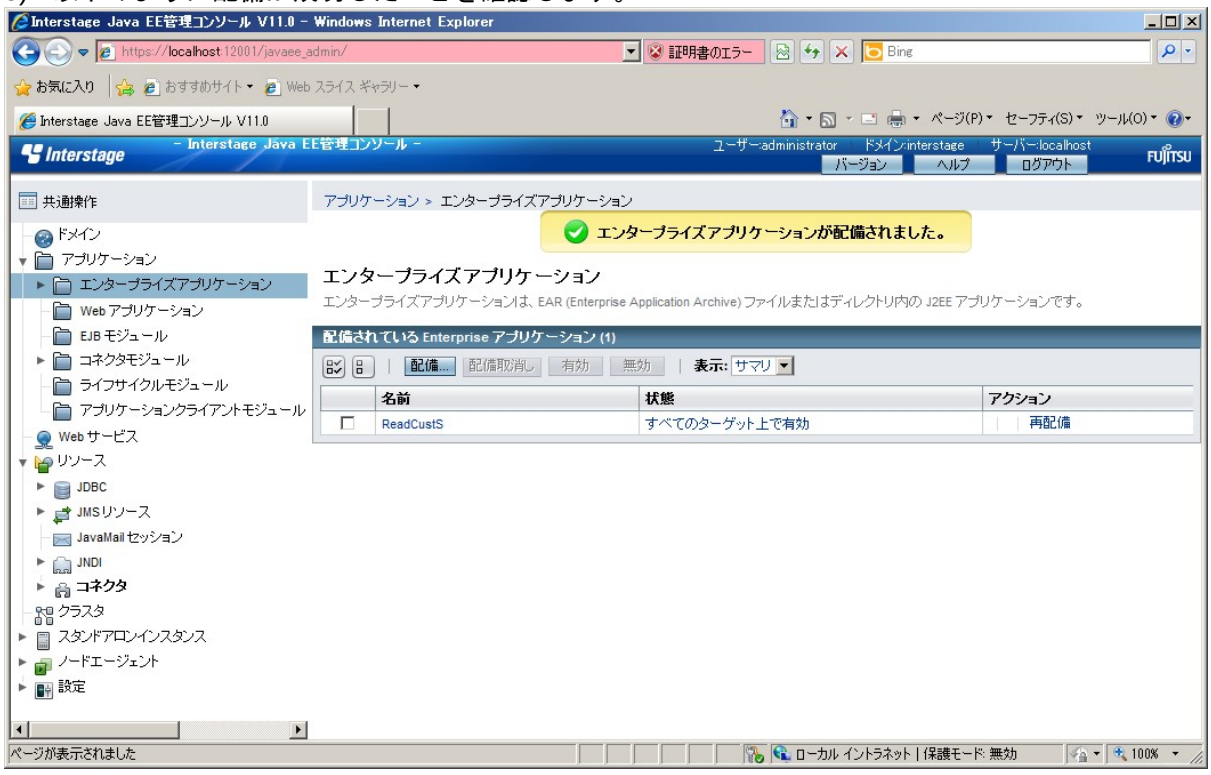
3) Interstage AS 管理コンソールに戻り、左ペインのツリービューで [アプリケーション] – [エンタープライズアプリケーション]を選択します。 右ペインで以下のように、[サーバーにアップロードされるパッケージファイル] の [参照]ボタンをクリックし、上記で生成された ReadCustS.ear のパス名を入力します。



4) 同ページ下で[利用可能なターゲット]から Server を選択し、[Add]ボタンをクリックすると、[選択したターゲット]に追加されます。[了解]ボタンをクリックします。

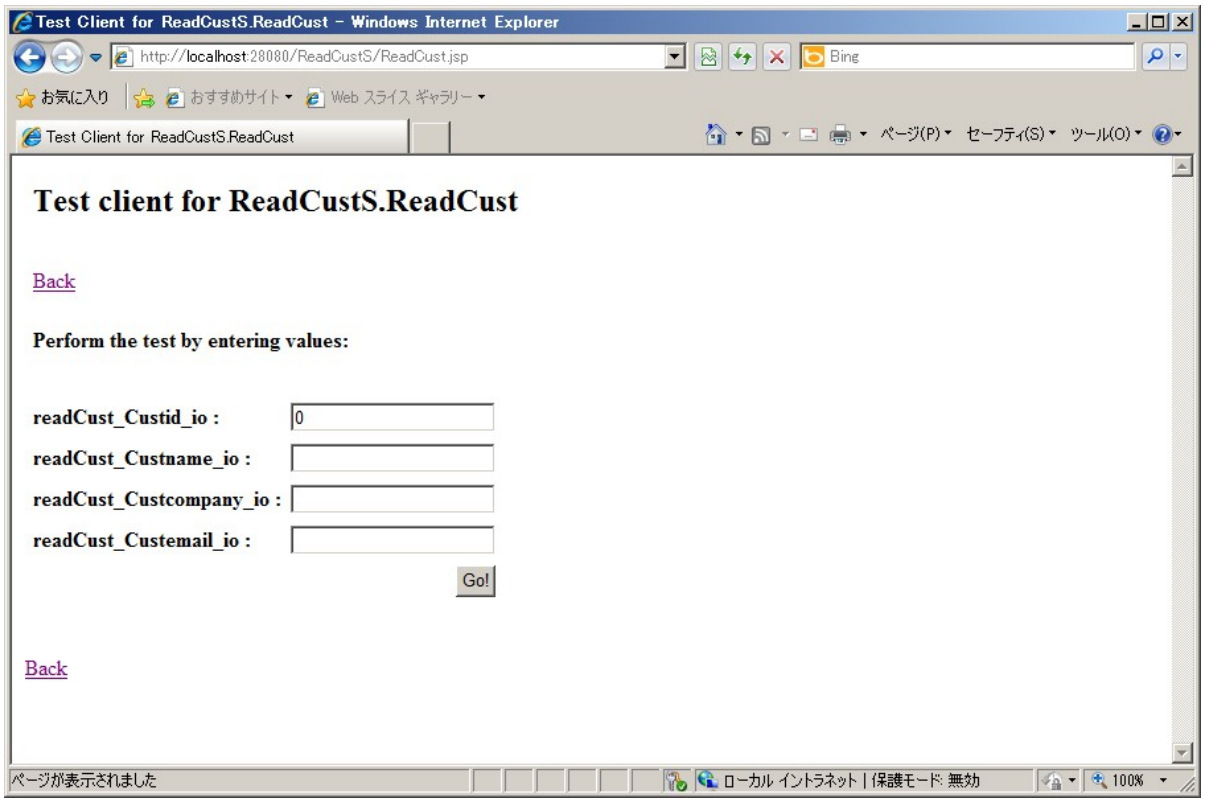


5) 以下のように配備が成功したことを確認します。



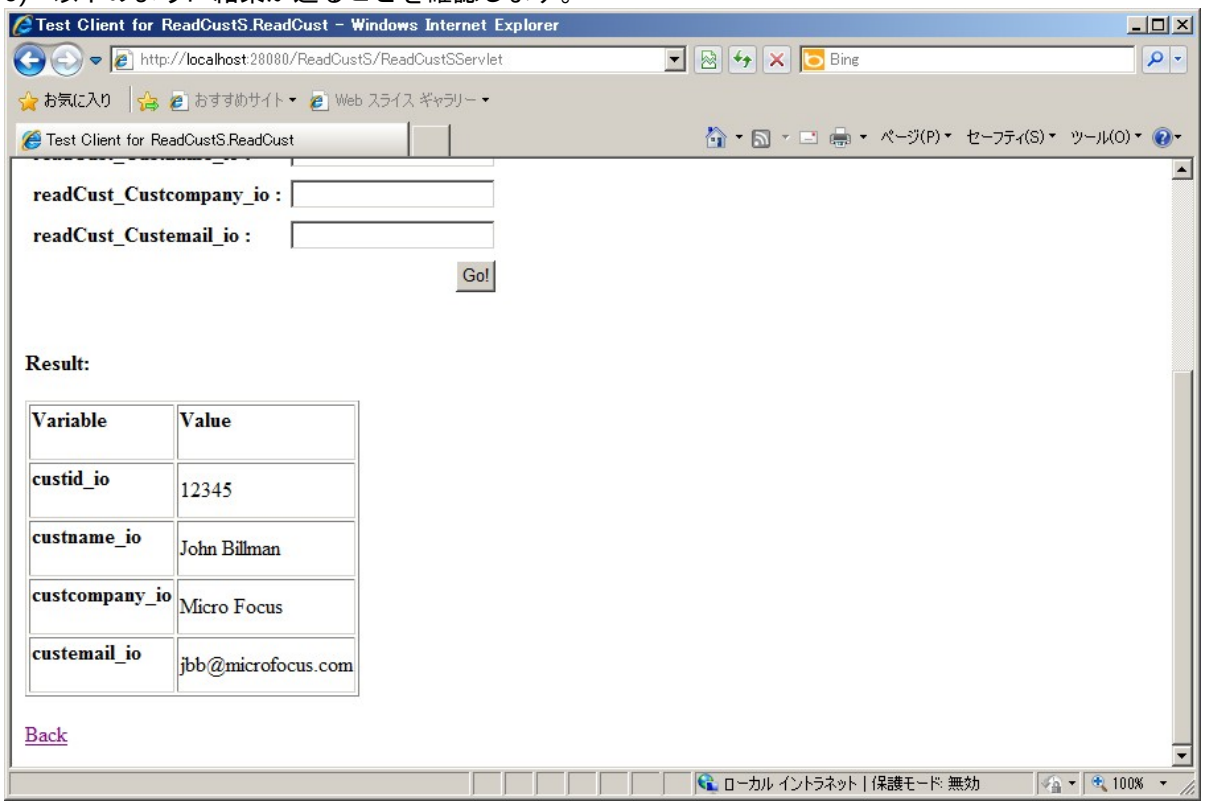
6) 以上で、EJB と Web アプリケーションが同時に配備されました。

7) Web ブラウザを開き、"http://localhost:28080/ReadCustS/ReadCust.jsp" を開きます。



8) 入力フィールド[readcust_custid_io]に 12345 と入力し、[Go!] をクリックします。

9) 以下のように結果が返ることを確認します。



以上