

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

Copyright © 2012 Micro Focus. All Rights Reserved. 記載の会社名、製品名は、各社の商標または登録商標です。

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。

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 ES-CUSTID
ACCESS MODE RANDOM
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 <u>https://localhost:12001/javaee_admin</u> 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。

CInterstage Java EE管理コンソール V11.0 -	Windows Internet Explo	
🕒 🗢 🖉 https://localhost:12001/javaee_a	admin/	IIII 書のエラー 日本 IIII 書のエラー 日本 IIII 書のエラー 日本 IIII 書のエラー 日本 IIII 書のエラー 日本 IIII 書のエラー IIII 書の IIII 書 IIII IIII IIII IIII IIII IIII IIII IIII III II II
<u>会</u> お気に入り 🛛 🚖 👜 おすすめサイト・ 🔊 Web	スライス ギャラリー・	
<i>後</i> Interstage Java EE管理コンソール V11.0		🚹 • 🔜 - 🖃 🖶 • ページ(P)・ セーフティ(S)・ ツール(O)・ 🕢
- Interstage Java E	E管理コンソール -	ユーザーadministrator ドメイン:interstage サーバー:localhost バージョン ヘルグ ログアウト FUJITSU
== 共通操作	アプリケーション > コネク	7タモジュール
 	エンタープライズ つ 配備するアプリケーション	アプリケーション/モジュールを配備 の場所を指定します。アブリケーションの形式は、war、ear、jar、rarなどのパッケージファイルです。
	タイプ: □ネクタモジ: 場所: [*] ⊙ サーバー C:Progra	ュール (.rar) ・ ・ にアップロードされるパッケージファイル m Files (x88)Wicro FocusiEnterprise Developer) 参照…
- @ Web サービス ▼ ₩ リソース ▶ ☆ JDBC	O Applicati	on Server からアクセス可能なローカルのパッケージファイルまたはディレクトリ
▶ 🛃 JMSリソース	アブリケーション名:*	f mfcobol-notx
- 📻 JavaMail セッション	状態:	☑ 有効
▶ (m) JNDI ▶ (m) コネクタ - 789 クラスタ	ベリファイアを実行:	二 有効 配備モジュールの検証を行います。 検証検乳ペリカ先ディレクトリ-Stoom sun.aas.das.instanceRootl/loos/verifier-results
 ► □ スタンドアロンインスタンス ► □ ノードエージェント 	スレッドブール ID:	コキクタモジュールのスレッドブール D (リソースアダブタ)
	説明:	
4	•	このセッションをあとで簡単に見つけられるようにします

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。

🖉 Interstage Java EE管理コンソール V11.0 - Windows Internet	Explorer
COO - Attps://localhost:12001/javaee_admin/	▼ 😵 証明書のエラー 🕺 🍫 🗙 🕞 Bing 🖉 マ
😭 お気に入り 👍 🖻 おすすめサイト・ 🖻 Web スライス ギャラリー・	
🏉 Interstage Java EE管理コンソール V11.0	🚹 • 🗟 - 🖃 🖶 • ページ(P)・ セーフティ(S)・ ツール(O)・ 🕢・
- Interstage Java EE管理コンソール -	ユーザー:administrator ドメイン:interstage サーバー:localhost バージョン ヘルプ ログアウト FUJITSU
++>3+6//-	
「「「大」」、「「「」」、「」」、「」、「」、「」、「」、「」、「」、「」、「」、「」、	
- ◎ ドメイン - □ マゴリケーション	iel/metンコールの視識で行います。 検証結果出力先ディレクトリ:\${com.sun.aas.das.instanceRoot}/logs/verifier-results
→ → → → → → → → → → → → → → → → → → →	
- 🛅 Web アプリケーション	コネクタモシュールのスレッドフール ID (リソースアタフタ)
- 🛅 EJB モジュール	このセッションをあとで簡単に見つけられるようにします
□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □	
- <u> <u> </u> Web サービス 利用可能な </u>	ターゲット: 選択したターゲット:
▼ 🔓 リンース	Add > server
	< Remove
	<< Remove All
► 🛄 JNDI	
▶ 高 コネクタ	
▶ 📑 設定	
ページが表示されました	

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。

🖉 Interstage Java EE管理コンソール V11.0 -	Windows Internet Explorer		
COO - Contraction - Contractio	admin/	🗾 😒 証明書のエラー 🕺 🍫 🗙 🗔 B	ing P •
👍 お気に入り 👍 🙆 おすすめサイト・ 🔊 Web	> スライス ギャラリー・		
<i>後</i> Interstage Java EE管理コンソール V11.0		🟠 • 🔊 - 🖃 🖷	" ・ ページ(P)・ セーフティ(S)・ ツール(O)・ ?
- Interstage Java E	E管理コンソール -	ユーザー:administrator ドメイン パージョン	ンinterstage サーバー:localhost FUJITSU
三 共通操作	アプリケーション > コネクタモジュール		
- 💮 FX12		🥑 コネクタモジュールが配備されました。	
 ▼ プリケーション - ニンターブライズアプリケーション - Web アプリケーション - EIB モジュール 	コネクタモジュール コネクタモジュールは企業情報システム (EIS) す。	への接続に使用され、RAR (Resource Adapter Archive) ファイルまたはディレクトリに バッケージ化されま
 □ コネクタモジュール 	配備されているコネクタモジュール (1)		
	お 日 配備… 配備取消し 有効	1 無効 表示: サマリ 🗸	
- 🛅 アブリケーションクライアントモジュール	名前	状態	アクション
─ <u>♀</u> Web サービス	mfcobol-notx	すべてのターゲット上で有効	再配備
🔻 🥁 リソース			
► 📄 JDBC			
▶ 🖨 JMSリソース			
- 🖂 JavaMail セッション			
► 🛄 JNDI			
▶ ⁽¹⁾ □ ² □ ² □ ² □ ²			
▶ □ スタンドアロンインスタンス			
			ット 保護モード: 無効 🛛 🖓 🔹 100% 🔹 🏸

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。

🖉 Interstage Java EE管理コンソール V	11.0 - Windows Intern	et Explorer
COO - Attps://localhost12001/j	avaee_admin/	🔽 😵 証明書のエラー 🕺 🍫 🗙 🔁 Bing 🖉 🖓
😭 お気に入り 👍 🙆 おすすめサイト・ 🧯	🕑 Web スライス ギャラリー・	
🏉 Interstage Java EE管理コンソール V11.0		🏠 • 🔝 - 🖃 🖶 • ページ(P)・ セーフティ(S)・ ツール(O)・ 🕢・
- Interstage	lāva EE管理コンソール -	ユーザーadministrator ドメインinterstage サーバー:localhost ののつかし FUJITSU
- 🛅 EJB モジュール	リソース > コネクタ >	コネクタ接続プール
▶ 📄 コネクタモジュール	新しいコネクタ格	と 続ブール (手順 1/2) 「次へ」 取消し
- 🛅 ライフサイクルモジュール	コネクタブールを作成し、	関連するリソースアダブタおよび接続定義を選択してから「次へ」をクリックします。
- 🛅 アプリケーションクライアント=		
ー 🧕 Web サービス	* 24	
▼ 🍟 リソース	名則:	eis/MFC000[_v1.5 一章(の名前 (255 文字以内で指定できます。半角英教字または、特殊文字()だけで構成される必
► 📄 JDBC		要があります。先頭には、英数字または_のみ指定可です〉
▶ 📑 JMSリソース	リソースアダプタ:*	mfcobol-notx 💌
- 🖂 JavaMail セッション		配備されたリソースアダプタ (コネクタモジュール) のリストから選択します
► 🚊 JNDI	接続定義:*	javax.resource.cci.ConnectionFactory
🔻 👸 コネクタ		
コネクタリソース		
- 🗋 コネクタ接続ブール		
— 管理オブジェクトリソース		
- <u>Re</u> 2 5 79		
▶ 📄 スタンドアロンインスタンス		
▶ 🚽 ノードエージェント		
▶ ■ 設定 🔽		
ページが表示されました		-

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。

🌈 Interstage Java EE管理コンソール V	11.0 – Windows Internet Explore	۲		<u>- 🗆 ×</u>
C C C R https://localhost:12001/j	avaee_admin/	111明書のエラー 🗟 😏 🗙 🖡	o Bing	₽ -
🙀 お気に入り 🛛 👍 🙋 おすすめサイト・ 👔	🧃 Web スライス ギャラリー ▼			
🏉 Interstage Java EE管理コンソール V11.0		🏠 • 🔝 • 🛛	🖪 🖶 🔹 ページ(P) 🍷 セーフティ(S)・ ツール(0)・ 🕡・
- Interstage	Jáva EE管理コンソール -	ユーザー:administrator バージ	ドメイン:interstage サーバー:local aン ヘルプ ログアウト	host FUĴĨTSU
□□ 1000 / 2007 2000 □□ EJB モジュール	リソース > コネクタ > コネクタ接続	売 プール		-
 ▶ □ コネクタモジュール □ ライフサイクルモジュール □ アブリケーションクライアント² 	新しいコネクタ接続ブー」 接続ブールの設定を検証して各プロ	ル (手順 2/2) ^い ティーの値を定義するプロパティー	戻る 5 を追加し、「完了」をクリックします。	<u>87</u> 取消し
Q Web サービス P リソース D リンース D JDBC JDSC JMS リソース JavaMail セッション D NDI	ー 般設定 名前: eis/MFCobol_ リソースアダプタ: mfcobol-notx 接続定義: javax.resourr 説明:	v1.5 ce.cci.ConnectionFactory		
	ブール設定			
 □ コネクタ接続ブール □ 管理オブジェクトリソース 	初期および最小プールサイズ:	- <mark>8 接続</mark> ブールで維持する接続数の下限お。	よこが初期値	
	最大ブールサイズ:	32 接続 クライアント要求に応じて作成できる	接続数の上限	
 ■ ノードエージェント ■ 設定 	プールサイズ変更量:	2 接続 ブールアイドルタイムアウトが時間も	のれになったときに消去される接続の	の数
ページが表示されました		לא תל-ם 🕵 🗌	、ラネット 保護モード 無効	🐴 🕶 🔍 100% 💌 🎵

7) 以下のように正常に作成されたことを確認します。

🌈 Interstage Java EE管理コンソール \	/11.0 - Windows Internet I	Explorer		
C	javaee_admin/	💽 😵 証明書のエラー 🖉	3 😽 🗙 🔽 Bing	₽ -
🖕 お気に入り 🛛 👍 🙋 おすすめサイト・	🦻 Web スライス ギャラリー・			
🏉 Interstage Java EE管理コンソール V11.	0	ľ	🕥 • 🔝 - 🖃 🖶 • ページ(P)• セーフ	アティ(S)▼ ツール(O)▼ 🕡▼
- Interstage	Java EE管理コンソール -	ユーザー:adm	inistrator ドメイン:interstage サーバー パージョン ヘルプ ロイ	:localhost ቻፖウト FUĴĨTSU
■ 1007 / 047 ■ EJB モジュール	リソース > コネクタ > コネ	クタ接続ブール		
► □ コネクタモジュール		🛛 🥑 新しいリソー:	スが正常に作成されました。	
	コネクタ接続ブール	,		
- <u>@</u> Web サービス	「新規」をクリックして、新しい モジュールの配備は「コネク	ロネクタ接続ブールを作成しま タモジュール」ページから実行	ます。プールの作成前にコネクタモジュールを できます。	を配備してください。コネクタ
► ■ JDBC	,, (4)			
▶ 🚅 JMSリソース	び日 新規 肖	『除		
► ja jndi	JNDI 名	リソースアダプタ	接続定義	説明
▼ 🔒 コネクタ	eis/MFCobol_v1.5	mfcobol-notx	javax.resource.cci.ConnectionFactory	
コネクタリソース				
▶ □ コネクタ接続ブール				
└── 管理オブジェクトリソース				
▶ ■ スタンドアロンインスタンス				
▶ 🗊 ノードエージェント				
▶ ● 設定 ▼				
ページが表示されました			ローカル イントラネット 保護モード: 無効	🖓 🔹 🔍 100% 🔹 🎢

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)

🖉 Interstage Java EE管理コンソール V	11.0 - Windows Internet Explorer
🕞 🗢 💌 https://localhost:12001/j	avaee_admin/ 🗾 😵 証明書のエラー 🕺 😏 🗙 🕞 Bing 🖉 🗸
🙀 お気に入り 🛛 👍 🔊 おすすめサイト・ 🧯	Web スライス ギャラリー▼
🏉 Interstage Java EE管理コンソール V11.0	🚹 * 🗟 * 🖻 * ページ(P)* セーフティ(S)* ツール(O)* 🕖・
- Interstage	J ava EE管理コンソール - ユーザー:administrator ドメイン:interstage サーバー:localhost FUJITSU
- ☐ EJB モジュール	
 ▶ □ コネクタモジュール □ ライフサイクルモジュール □ アプリケーションクライアント² 	新しいコネクタリソース ア解 取消 コネクタリソースを作成するはは、そのリソースの関連付けられる接続ブールを指定します。複数のコネクタリソースが1つの 接続ブールを使用できます。
Web サービス ▼	JNDI名: [*] CCIMFCobol_v1.5 一意の名前(255文字以内で指定できます。半角英数字、半角スペースまたは特殊文字(!()- <i>t</i> ,⇔@\^_'0~) だけで構成される必要があります。先頭コよ、半角スペースが指定不可です)
▶ ∰ JMS リワース ────────────────────────────────────	ブール名: * eis/MFCobol_v1.5 ▼ コネクタ接続プールページを使用して新しいプールを作成する
▼ 🔒 コネクタ	
 □ コネクタリソース □ コネクタ接続ブール □ ロネクタ接続ブール □ 管理オブジェクトリソース 	
	利用可能なターゲット: Add > server
 ► □ ノードエージェント 	Add All >>
→ → 設定 →	< Remove

9) 以下のように正常に作成されたことを確認します。

💋 Interstage Java EE管理コンソール \	/11.0 - Windows Internet Explo	rer			
COO - 12001/	/javaee_admin/	😵 証明書のエラー 🛛 🗟	🔸 🗙 🔽 Bing		P -
🙀 お気に入り 🛛 👍 🙋 おすすめサイト 🔹	🥑 Web スライス ギャラリー 🔹				
🏉 Interstage Java EE管理コンソール V11.	0	6	• • • • •	ページ(P)・ セーフティ(S)・	ツール(0) * 🕡 •
- Interstage	Jáva EE管理コンソール -	ユーザー:admir	nistrator ドメイン:inter: パージョン	stage サーバー:localhos ヘルプ ログアウト	t FUĴĨTSU
	リリース > コネクタ > コネクタリ	ノース			
► □ コネクタモジュール		🥑 新しいリソース	が正常に作成されま	した。	
	ーナカカリリーフ				
ー 〒 アブリケーションクライアント-	」インメリノース 」 コネクタリリースは、エンターゴライ	イズ"情報システム (FIS)と(の接続をアプリケーション	に提供するプログラムオラ	ジェクトです。
		у (наты) у (у та (сно) Со	71x02C7 77772 747		2271 C70
▶ 📄 JDBC	リソース (1)				
▶ 🛃 JMSリソース	♥ 品 新規 削除	有効無効	表示: サマリ 💌		
ー 🖂 JavaMail セッション	JNDI 名	状態		接続ブール	説明
	CCIMFCobol_v1.5	すべてのターゲット上1	で有効	eis/MFCobol_v1.5	
▼ 前 コキンス					
▶ 🗋 コネクタ接続ブール					
└── 管理オブジェクトリソース					
- 8ª 27729					
▶					
▶ 🔐 ノードエージェント					
]				
ページが表示されました			ローカル イントラネット 係	こう 無効 🦷	- 🔍 100% - //

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。

COBOL プロジェクト名(P): ReadOust プロジェクト名(P): ReadOust プロジェクト名(P): ReadOust プロジェクト名(P): ReadOust クロジェクト名(P): ReadOust クロジェクト名(P): ReadOust クロジェクト名(P): ReadOust クロジェクト名(P): ReadOust クロジェクト名(P): アティル・シュンを使用(D) ファイル・システムを選択(Y): デフォルト・マ	🚾 COBOL プロジェクトの新規作成	
プロジェクト名(P): ReadCust ▼ デフォルト・ロケーションを使用(D) ロケーション(1): ©*ReadCust#workspace*ReadCust ファイル・システムを選択(Y): デフォルト ▼	COBOL プロジェクト ワークスペースまたは外部の場所にCOBOL プロジェクトを作成しま	⁹ .
デフォルト・ロケーションを使用(D) ロケーション(L): ○、半ReadCust半workspace半ReadCust アァイル・システムを選択(Y): デフォルト ▼	プロジェクト名(P): ReadCust	
ロケーション(L): [C:¥ReadCust¥workspace¥ReadCust ファイル・システムを選択(Y): 「デフォルト ▼	デフォルト・ロケーションを使用(D)	
ファイル・システムを選択(Y): デフォルト -	ロケーション(L): C:¥ReadCust¥workspace¥ReadCust	参照(R)
	ファイル・システムを選択(Y): デフォルト 🔽	
(ア) 元了(F) キャンセル	?	完了(F) キャンセル

3) 開発プロジェクトが新規作成されました。

🖳 コンソール 😫 【 問題) 🧔 タスク	- P	I - 📬 - 🗆 🗆	1
COBOL Build			1
os.init.unix:		<u> </u>	1
init:			
post.build.cfg.New_Configuration:			
BUILD SUCCESSFUL Build finished with no errors.			
Total time: O seconds			
a l		•	1

3-2. プロジェクトヘ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。	
屋 プロパティー: ReadCust		_O×
フィルター入力	COBOL	
 □-リソース □ Micro Focus COBOL □ Project 設定 □ ビルド パス □ ビルド構成 □ □ COBOL □ - イペント □ □ リンク □ 実行時構成 □ ビルダー - プロジェクト参照 - リファクタリング・ヒストリー 実行・ゲバッグ設定 	New Configuration [使用中] 出力の名前: ReadCust 出力パス: New Configuration bin エントリポイント ターゲット設定 ターゲット設定 ターゲット設定 単一実行可能ファイル 単一実行可能ファイル アスドロビアイル マンドメディアイジョン マンドメディアイジョン マンドメディジョン マンドメディジョン マンドメディジョン マンドメディジョン マンドメディジョン マンドメディジョン マンドメディション マンドメディション マンドメディション マンドメリー マンドメリー マンドメリー マンドメリー マンド マンド マンド コンド コンド マンド マンド マンド マンド マンド </td <td> ■ 構成の管理 参照 </td>	 ■ 構成の管理 参照
		复元(D)適用(A)
?	0	K キャンセル

- 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。

M インボート	_ <u> </u>
ファイル・システム ソースは空にできません。	
次のディレクトリーから(^):	▲ 参照(R)
	ディレクトリーからインボート 🗵
タイブのフィルター(T)_ すべて選択(S) 選択をすべて解 宛先フォルダー(L): オブションー オブションー 警告を出さずに既存リソースを上書き(O) 〇 Create top-level folder(C) 拡張(A) >>	インボート元のディレクトリーを選択します。
?	< 戻る(B) 次へ(N) > 完了(F) キャンセル

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。

層 インボート		
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ボートします。		
次のディレクトリーから(Y): C¥ReadCust		参照(R)
🖲 🗖 🦢 ReadCust	☐ @ CUST.dat ☐ @ CUST.idx ☑ © ReadCust.cbl ☑ © ReadCustMain.cbl	
, タイブのフィルター(T) すべて選択(S) 選択をすべて解除((D)	
宛先フォルダー(L): ReadCust		参照(W)
「オブション」 「 警告を出さずに既存リソースを上書き(O) 「 Create top-level folder(C) 拡張(A) >>		
0	< 戻る(B) 次へ(N) > 完了(F)	キャンセル

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。

📃 コンソール 🙁 🖹 問題 🥝 タスク	📑 🙀 🛃 🛃 - 📬 - 🗖 🗖
COBOL Build	
os.init.unix:	-
init:	
post.build.cfg.New_Configuration:	
BUILD SUCCESSFUL Build finished with no errors.	
Total time: 2 seconds	
	Þ

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

COB CB CB COBOL プログラ D COBOL プログラ D D COBOL プログラ			
H-B New_Configurat	New		•
	<u>רש</u> בצ-	Ctrl+C	
	貼り付け	Ctrl+V	
	💢 削除(D)	Delete	
	移動(V) 名前変更(M)	F2	
	指令の確定		
	インポート(1)		▶ 転 リモート COBOL プロジェクト
	🛃 エクスポート(0)		🧟 ローカル COBOL ブロジェクトをリモート プロジェクトへ変換
	🧞 リフレッシュ(F)	F5	👝 Net Express プロジェクトの変換
アウトライン 🛿 📰	実行(R) デバッグ(D)		と インボート(0

- 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。

図 インボート		_ 🗆 🗙
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ボートします。		
ン☆のディレクトリーから(Y): [C¥ReadCust ᠃- □	 CUSTdat CUSTdat CUSTidx ReadCustcbl ReadCustMaincbl 	参照(R)
タイブのフィルター(T) すべて選択(S) 選択をすべて解除(D) 宛先フォルダー(L): ReadCust/New_Configuration.bin オブション		参照(W)
?	< 戻る(B) 次へ(N) > 完了(F)	キャンセル

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。

📃 コンソール 🙁 🔡 🗐 🖉 タスク	🕞 🔂 🛃 🗐 - 😁 - 🗖 🗋
COBOL Build	
os.init.unix:	<u>•</u>
init:	
post.build.cfg.New_Configuration:	
BUILD SUCCESSFUL Build finished with no errors.	
Total time: O seconds	
<u> </u>	×

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。

🔤 COBOL マッピング ウィザード	
ソースの種類を選択 ソースプログラムの場所を選択します。	
Eclipse COBOL プロジェクト ファイルシステム上の COBOL ファイル	
< 戻る(B) 次へ(N) > 完了(F)	キャンセル

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。

📴 COBOL マッピング ウィザード	
プロジェクト情報の指定 マップする Eclipse プロジェクトを定義します。	
◎ ナビゲータ上の現在のプロジェクト	
○ 開いている Eclipse プロジェクトを選択	
Eclipse プロジェクトのルートロケーションを入力します:	
C:¥ReadCust¥workspace¥ReadCust	参照
< 戻る(B) 次へ(N) > 完了(F)	キャンセル

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

III COBOL マッピング ウィザード	
プログラム情報の指定 マップする COBOL プログラムを選択	9
ReadCust cbl ReadCustWS-app.cbl ReadCustWS-proxy.cbl	
< 戻る(B) 次へ(N) > 完了(F)	キャンセル

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。

🔤 COBOL マッピング ウィザード	
サービス マッピング名 サービス マッピング名を定義します。	
サービスの名前を入力します:	
ReadCustS	
< 戻る(B) 次へ(N) >	完了(F) キャンセル

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。

📠 COBOL マッピング ウィザード	
省略時マッピングの実行 マッピングの種類を選択します。	4
● 省略時マッピング	
○ 無し	
< 戻⊚(B) /갔へ(N) 元](7 7777

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。

NKAGE SECTION:		ReadCust オペレーション - イン:	ターフェイスフィー	ルド:	
名前	PICTURE	名前	方向	型	OCCU
🖵 custid	9(9) comp-5	custid_io	入出力	int	
🗖 custname	X(30)	🕨 🖚 custname_io	入出力	String	
🗖 custcompany	X(30)	custcompany_io	入出力	String	
🗖 custemail	X(30)	custemail_io	入出力	String	





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。

四マッピンク プロパティ	
一般 ディプロイメントサーバー アプリケーションファイル EJB 生成	
● EJB を生成	
〇 Java Bean を生成 (J2SE を使用)	
OK +	ンセル

3) [変更]ボタンをクリックします。

國 マッピング プロパティ	
→般 ディプロイメントサーバー アプリケーションファイル EJB 生成	
Enterprise Server 名:	
	変更
☐ Enterprise Server 実行時環境の使用	
Enterprise Server 実行時環境の構成	
サービス名:	
ReadCust	
「トランザクション管理	
 アプリケーション管理 	
○ コンテナ管理	
□ ディプロイする場合はユーザー名/パスワードが必要	
	OK キャンセル

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。

-K- SDEMO	サービス Deployer	サービス状態 Available Started	エンドポイント 102 168 70 140:55127	リスナー状態 RitMode=3	説明	
SDEMO	Deployer	Available, Starteu	192.100.70.140.00127	DICMODE-3	De	
					_	

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。

 一般 ディプロイメントサーバー アプリケーションファイル EJB 生成 Enterprise Server 名: [ESDEMO (192.168.70.140:55127) 変更 Enterprise Server 実行時環境の使用 Enterprise Server 実行時環境の構成
Enterprise Server 名: ESDEMO (192.168.70.140:55127)変更 Enterprise Server 実行時環境の使用 Enterprise Server 実行時環境の構成
ESDEMO (192.168.70.140:55127) 変更 Enterprise Server 実行時環境の使用 Enterprise Server 実行時環境の構成
Enterprise Server 実行時環境の使用 Enterprise Server,実行時環境の構成
Enterprise Server 実行時環境の構成
サービス名:
ReadCustS
● アプリケーション管理
○ コンテナ管理
□ ディプロイする場合はユーザー名/パスワードが必要
OKキャンセル

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。

🚾 マッピング プロパティ 📃 🗆 🗙
一般 ディプロイメントサーバー アプリケーションファイル EJB 生成
レガシーアプリケーションをディプロイ済みか、またはサーバーにディプロイする必要があるかを指定してください。
アプリケーションファイル
ファイル追加 ファイル追加 ファイル追加
OK キャンセル

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。

🔤 ファイルの選択			×			
🕞 🕞 🗸 🕨 🗸 ReadCust	New_Configuration bi	New_Configuration binの検索 🛛 🔎				
整理 ▼ 新しいフォルダー		:==	- 🔟 🕡			
🚖 お気に入り	名前 ▲	更新日時	種類 ▲			
🕠 ダウンロード	CUST.dat	2011/01/12 10:25	DAT ファイル			
■ デスクトップ		2011/01/12 10:25	IDX ファイル			
🧾 販近表示した場所	mccerror	2012/10/15 16:53	テキスト ドキ			
🔚 ライブラリ	New_Configuration.gcf	2012/10/15 17:06	GCF ファイル			
 ■ ドキュメント ■ ピクチャ ■ ピグチャ ■ ビデオ ♪ ミュージック 	📄 ReadCust.idy	2012/10/15 13:42	IDY ファイル			
	📄 ReadCust.int	2012/10/15 13:42	INT ファイル			
	ReadCustMain.idy	2012/10/15 13:42	IDY ファイル			
	ReadCustMain.int	2012/10/15 13:42	INT ファイル			
📮 コンピューター	ReadCustWS-app.idy	2012/10/15 16:11	IDY ファイル			
🚢 ローカル ディスク (C:)	ReadCustWS-app.int	2012/10/15 16:11	INT ファイル			
A +	ReadCustWS-proxy.idy	2012/10/15 16:11	IDY ファイル―			
📭 ሐምቦፓቸታ	ReadOuctWS-provv int	2012/10/15 16:11	INT 774 II.			
ファイ	イル名(N): 「ReadCust.int" "CUST.dat" "Cl	*.*	•			
		開<(0)	キャンセル			

 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。

歴 マッピング プロパティ									
一般 ディプロイメントサーバー アプリケーションファイル EJB 生成									
レガシーアプリケーションをディプロイ済みか、またはサーバーにディプロイする必要があるた)を指定してください。								
○ レガシーアプリケーションは既にディプロイ済み									
ディプロイされたアプリケーションのパス:									
● レガシーアプリケーションをディプロイする									
アプリケーションファイル									
C:/ReadCust/workspace/ReadCust/New_Configuration.bin/CUST.dat C:/ReadCust/workspace/ReadCust/New_Configuration.bin/CUST.idx C:/ReadCust/workspace/ReadCust/New_Configuration.bin/ReadCust.idy C:/ReadCust/workspace/ReadCust/New_Configuration.bin/ReadCust.idy C:/ReadCust/workspace/ReadCust/New_Configuration.bin/ReadCust.idy C:/ReadCust/workspace/ReadCust/ReadCust.cbl	ファイル追加 ファイル削除								
ОК	++>>tun								

- 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。

國 マッピング プロパティ					
──般 ディプロイメントサーバー アプリケーションファイル	EJB 生成				
アプリケーション サーバー J2EE 1.4 💌 WebLogic 9	1.2 💌				
┌EJB 属性					
Bean 名:	ReadCustS				
パッケージ名:	com.mypackage.ReadCustS				
セッション永続性:	 ● ステートレス ● ステートフル 				
インターフェイス タイプ:	● ローカル ● リモート				
EJB バージョン 3: (Java コンパイラ 1.5 以上が必要):					
ディプロイメントディスクリプタ属性					
EJB 名:	ReadCustSEJB				
アーカイブ名:	ReadCustS				
J2SE と J2EE の属性 Javac パス "{0}" にライセンスファイルが見つかりません Java コンパイラ:	∞ C:¥Interstage¥JDK6¥bin 参照…				
EJB、コネクタ(また、クライアント生成する場合、servle	at と JSP)関連の JAR ファイルのパスを追加します。				
J2EE クラスパス:	C:/Interstage/F3FMisjee/lib/javaee.jk 参照				
	OK キャンセル				

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。

ال 🖃 🗐 📓 📓 🗐 🗐 🕹 🗐 🗐 🗐 🗐 🗐 🗧							
サービス インターフェース コンソール							
2003 (2009/05/27 11:26:34): File pathname "C:/ReadCust/workspace/ReadCust/ReadCust.cbl" is absolute;	red 🔼						
0012 (2009/05/27 11:26:34): Processing "Service" section of Manifest							
0019 (2009/05/27 11:26:34): Manifest processing completed							
0020 (2009/05/27 11:26:34): Adding service and package objects to directory							
0021 (2009/05/27 11:26:35): Using directory at mrpi://10.18.11.108:86							
0030 (2009/05/27 11:26:36): ES server "ESDEMO" notified service "ReadCustS.READCUST" is available							
0002 (2009/05/27 11:26:36): Installation of package "ReadCustS.car" finished with 6 warnings							
Retrieving deployment log file http://10.18.11.108:1086/uploads/ReadCustS.GwgqGG2z/deploylog.txt Deployment completed with warnings							
	>						

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。

		► Server ESDEMO [開	始 🖌]								
t.	サーバー 】 リスナー (3)】 サービス (7)】 ハンドラ (2) <mark>パッケージ (2)</mark>										
1 - 2 of 2 packages Show 10 packages at a ti									a time 🛛 🧧		
ステータ ワ ル ステータ ワ ル			パッケージモジュール	IDT	パッケージ パス	カスタム構成	記忆時月				
C	編集	http://tempuri.org/ReadCustW/S	Available	ок		C:\Program Files\Micro Focus\Net Express 5.1 \Base\deploy\ReadCust\WS.zg3w6CEa/ReadCust\WS.idt	C:\Program Files\Micro Focus\Net Express 5.1 \Base\deploy\ReadCust\WS.zg3w6CEa		created 14:12:6 ReadCustWS.z		
C	編集	ReadCustS	Available	ок		C:\Program Files\Moro Focus\Net Express 5.1 \Base\deploy\ReadCustS.GwgqGG2z/ReadCustS.idt	C:\Program Files\Micro Focus\Net Express 5.1 \Base\deploy\ReadCustS.GwgqGG2z		created 11:26:3 ReadCustS.Gw		
G											

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。

😂 ReadCustS.deploy									
ファイル(E) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)									
G 戻る ・									
アドレス(D) 🛅 C:¥ReadCust¥workspace¥ReadCust¥repos¥ReadCustS.deploy									
7 ง มงจี	×	名前 🔺	サイズ	種類					
🖃 🚞 ReadCust	^	Com		ファイル フォルダ					
🖃 🚞 workspace		🛅 META-INF		ファイル フォルダ					
표 🚞 .metadata		🗐 mccerror.txt	0 KB	テキスト ドキュメント					
🖃 🚞 ReadCust		💼 ReadCustS.car	5 KB	CAR ファイル					
🚞 New_Configuration.bin		🖻 ReadCustS.idt	1 KB	IDT ファイル					
🖃 🛅 repos		💼 ReadCustS.jar	5 KB	JAR ファイル					
🗉 🗀 ReadCustS.deploy		🖻 ReadCustS.mf	1 KB	MF ファイル					
🛅 ReadCustWS.deploy		🖻 ReadCustS.mfmak	2 KB	MFMAK ファイル					

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。
- 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。



6) 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。

	Canterstage gava FE.E.HTTTAAN ALL'0 -	windows internet Explorer							
	COO - Market 12001/javaee_	admin/	🗾 😒 証明書のエラー 🕺 😏 🗙 🗔	Bing 🖉 🗸					
 ● Interstace Java EE管理ユンソール V11.0 ● Interstace Java EE管社コンソール - 2-ヴーadministrator 下パイン(necrtace サーバー-localhost TV) - 10770h ● Interstace Java EE管社コンソール - 2-ヴーadministrator 下パイン(necrtace サーバー-localhost TV) - 10770h ● ボメロター ● ボメロター ● アブリケーション ● エンターブライズアブリケーション ● エンターブライズアブリケーション ● Coll EE モジュール ● コネクタモジュール ● ライフサイクルモジュール ● ライクサイクルモジュール ● アブリケーション ■ Addiai Tervica ● Mach ● Mach ● Mach ● Coll Ei ● Mach ● Coll Ei 	🔆 お気に入り 👍 👩 おすすめサイト・ 👩 Web	スライス ギャラリー・							
● Interstage ● Interstag	<i>後</i> Interstage Java EE管理コンソール V11.0		🟠 • 🔊 - 📼	🖶 • ページ(P)• セーフティ(S)• ツール(O)• 🕡•					
■ 共通雑作 アガリケーション エンターブライズアブリケーション ● ドメイン ● エンターブライズアブリケーション ● アガリケーション ● ンターブライズアブリケーション ● ひゃり アブリケーション ■ ンターブライズアブリケーション ● ひゃり アブリケーション ■ ンターブライズアブリケーション(1) ● コネクタモジュール ● シーブサイクルモジュール ● ライブサイクルモジュール ● シーブサイクルモジュール ● アブリケーションクライアントモジュール ● 副 配価 配価取消し 有効 無効 ● 表示: サマリ ● ● ジリンース ● 名前 ● 文ペ のターグット上で有効 ● JDBC ● 副 MSU リース ● JDBC ● JNSU リース ● JDBC ● JNSU リース ● JNSU リース ● JNSU リース ● JNSU +	- Interstage Java E	E管理コンソール -	ユーザー:administrator ドメ <u>パージョン</u>	インinterstage サーバー:localhost FUjinsu					
 ● ドメイン ● アクリケーション ● エンタープライズアプリケーション ■ エンタープライズアプリケーション ■ エンタープライズアプリケーション ■ エンタープライズアプリケーション ■ エンタープライズアプリケーション ■ エンタープライズアプリケーション ■ コネクタモジュール ● ライフサイクルモジュール ● ライフサイクルモジュール ● アブリケーションクライアントモジュール ● アブリケーションクライアントモジュール ● アブリケーションクライアントモジュール ● アブリケーションクライアントモジュール ● DBC ● JBCC ● (A (A	== 共通操作	アプリケーション > エンタープライズ	アプリケーション						
 ▶ アブリケーション ▶ エンターブライズアブリケーション エンターブライズアブリケーション エンターブライズ エンターブリケージョン エンターブライン エンターブライン エンターブライズ エンターブリケージョン エンターブライン エンターブン エンターブライン エンターブライン<td>- 🚱 FX12</td><td></td><td>✓ エンターブライズアプリケーションが配備され</td><td><mark>いました。</mark></td>	- 🚱 FX12		✓ エンターブライズアプリケーションが配備され	<mark>いました。</mark>					
 □ EUB モジュール ▲ パクタモジュール □ オ々クタモジュール □ ライフサイクルモジュール □ アブリケーションクライアントモジュール ■ アリケーションクライアントモジュール ■ アリケース ■ DBCC ■ JINSU ソース □ Javalkal セッション ► Mission 1000 	 ▼ つ アブリケーション ▶ □ エンターブライズアブリケーション □ Web アブリケーション 	エンタープライズアプリケー エンタープライズアプリケーションは、	ー ション EAR (Enterprise Application Archive) ファイルまたはディレクト	リ内の JZEE アブリケーションです。					
 ■ コネジラミシュール ● ライフサイクルモジュール ● アブリケーションクライアントモジュール ● Web サービス ● Web サービス ● BDBC ● JBC ● (Late and CustS) ●	- EJB モジュール	配備されている Enterprise アブリケーション (1)							
▲前 状態 アグション ● アブリケーションクライアントモジュール ▲前 状態 アグション ● Web サービス ■ ReadCustS すべてのターゲット上で有効 再配備 ● DDBC ● ● JMSU ソンース ● ● ● JMSU ソンース ● ● ● ● ● ● JMSU ソンース ● ● ● ● ● JMSU ソンース ● ● ● ● ● JMSU ソンース ● ● ● ● ● JNDI ● ● ● ●		■ 1 配備取消し 有効 無効 表示: サマリ ▼							
ReadCustS すべてのターグット上で有効 再配備 ● ピリンース > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > <	□ アプリケーションクライアントモジュール	名前	状態	アクション					
 ▼ ● リソース ▶ ● DBC ▶ ● MSUソース ■ JavaMai セッション ▶ ● MDI 	- 🔵 Web サービス	ReadCustS	すべてのターゲット上で有効	冉昭仁備					
 ▶ (a) コネクタ > (a) クラスタ ▶ (a) スタンドアロンインスタンス ▶ (a) ノードエージェント ▶ (b) (b) (c) (c) (c) (c) (c) (c) (c) (c) (c) (c	 ● リソース ● JDBC ● JDBC ● JMSリソース ■ JavaMail セッション ● ☆ JAPJ <								
ーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーー	ページが表示されました			ネット 保護モード 無効 🛛 🖓 🔹 100% 🔹					

1 milest

- 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。

CTest Client for ReadCustS.ReadCust - Windows Internet Explorer		<u>_ </u>
	💌 🗟 🍫 🗙 🔽 Bing	₽ -
😭 お気に入り 🛛 🍰 🔊 おすすめサイト・ 🔊 Web スライス ギャラリー・		
ℰ Test Client for ReadCustS.ReadCust	🏠 • 🔂 → 🖃 🖶 • ページ(P) • セーフティ(S) •	ツール(0) * 🔞 •
Test client for ReadCustS.ReadCust		*
Back		
Perform the test by entering values:		
readCust_Custid_io : 0 readCust_Custname_io :		
readCust Custcompany io :		
readCust_Custemail_io :		
Go!		
Back		
パーソル表示されました		• • • • • · · · · · · · · · · · · · · ·

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。

Test Client for	ReadCustS.ReadCust - W	indows Internet Explorer		
😋 🕤 🗢 😰 http://localhost:28080/ReadCustS/ReadCustSServlet 🛛 🔽 🐼 🐓 🗙 🔁 Bing				
🔓 お気に入り 🛛 👍	🦻 おすすめサイト・ 🤌 Web	スライス ギャラリー・		
🏉 Test Client for R	eadCustS.ReadCust		🟠 • 🔊 - 🖃 📥 • ページ(P)・ セーフティ(S	・ ツール(0) ・ 🕡・
roadCust Cust	company io :			•
Teaucust_cust				
readCust_Cust	emaii_io :	1000		
		Go!		
Result:				
				
Variable	Value			
custid_io	12245			
	12343			
custname_io	John Billman			
custcompany_i	0 Minus France			
	Micro Focus			
custemail_io	jbb@microfocus.com			
L				
Back				_
			● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●	🔨 - 🔍 100% - 🗸