IBM WebSphere Commerce - Express Developer Edition

マイグレーション・ガイド

バージョン 5.5

IBM WebSphere Commerce - Express Developer Edition

マイグレーション・ガイド

バージョン 5.5

ご注意!

本書および本書で紹介する製品をご使用になる前に、特記事項に記載されている情報をお読みください。

本書の内容は、新版で特に指定のない限り、IBM WebSphere Commerce - Express Developer Edition for Windows 2000 (プロダクト番号 5724-A18) のバージョン 5.5 以降のすべてのリリースおよびモディフィケーションに適用されます。製品のレベルにあった版を使用していることをご確認ください。

本マニュアルに関するご意見やご感想は、次の URL からお送りください。今後の参考にさせていただきます。

http://www.ibm.com/jp/manuals/main/mail.html

なお、日本 IBM 発行のマニュアルはインターネット経由でもご購入いただけます。詳しくは

http://www.ibm.com/jp/manuals/ の「ご注文について」をご覧ください。

(URL は、変更になる場合があります)

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原　典： IBM WebSphere Commerce - Express Developer Edition
Migration Guide
Version 5.5

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2003.10

この文書では、平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、平成角ゴシック体™W5、および平成角ゴシック体™W7を使用しています。この(書体*)は、(財) 日本規格協会と使用契約を締結し使用しているものです。フォントとして無断複製することは禁止されています。

注* 平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、
平成角ゴシック体™W5、平成角ゴシック体™W7

目次

まえがき v
本書の更新 v
本書の表記規則 v
詳細情報の入手方法 vi

第 1 部 マイグレーション前 1

第 1 章 WebSphere Commerce 開発環境のマイグレーションの前に 3
マイグレーションの前提条件 3
前提条件となるハードウェア 3
前提条件となるソフトウェア 4
マイグレーション環境 4
環境 A — 純粋な開発環境 6
環境 B — 開発環境と WebSphere Commerce が共存するが資産は共用しない 7

第 2 章 WebSphere Commerce 開発環境へのマイグレーションに伴うコードの変更 9
WebSphere Commerce 5.4 および WebSphere Commerce Studio 5.4 のセットアップ環境 9
見積要求 (RFQ) 9
コラボレイティブ・ワークスペース 10
データベース・スキーマの差異 13
API の差異 14
Enterprise Bean の 1.0 から 1.1 仕様へのマイグレーション 14
MSGTYPES テーブルの固有索引の検査 16
WebSphere アプリケーション・コマンド・フレームワーク 16
J2EE Connector Architecture 16

第 2 部 マイグレーション・フロー 19

第 3 章 WebSphere Commerce 開発環境のマイグレーション 21
WebSphere Commerce 資産および WebSphere Commerce 開発環境資産のバックアップ 21
DB2 製品レベルのアップグレード 21
WebSphere Commerce 開発環境のアップグレード . . 22
タスク・コマンド、コントローラー・コマンド、および Data Bean の JAR ファイルへのエクスポート 23
Enterprise Bean の JAR ファイルへのエクスポート 24
IBM WebSphere Commerce 5.5.0.2 フィックスパックの適用 24
WebSphere Commerce 開発環境データベース・スキーマのマイグレーション 25
WebSphere Commerce インスタンスのマイグレーション 25
カスタマイズまたは拡張されたコードの変換 . . . 30
VisualAge for Java と WebSphere Studio Application Developer の違い 30
拡張または変更された WebSphere Commerce ビジネス・ロジックおよびコードの変換 31
決済インスタンスおよび決済データベースのマイグレーション 41

第 3 部 マイグレーション後 43

第 4 章 マイグレーション後のアクション 45
環境固有のマイグレーション後アクション 46
環境 B 46

第 4 部 コードの変換のためのチュートリアル 47

第 5 章 チュートリアルのマイグレーション・フロー 49

第 6 章 チュートリアル A: 拡張またはカスタマイズされたコントローラー・コマンドのマイグレーション 51
Enterprise Bean をマイグレーションする 51
VisualAge for Java から Enterprise Bean をエクスポートする 51
WebSphereCommerceServerExtensionData フォルダーの ExtensionJDBCHelperBean を削除する . . . 51
Enterprise Bean を WebSphere Studio Application Developer にインポートする 52
メソッドを編集して java.rmi.RemoteException エラーが起きないようにする 52
アクセス分離レベルを変更する 52
デプロイメント・コードを生成する 53
カスタマイズされた JSP ファイルをマイグレーションする 53
拡張コントローラー・コマンドをマイグレーションする 54
新規ロジックを Express Store サンプル・ストアでテストする 55

第 7 章 チュートリアル B: 拡張/カスタマイズされた Entity Bean およびタスク・コマンドのマイグレーション 57
拡張タスク・コマンドをマイグレーションする . . 57

拡張 Data Bean をマイグレーションする 58
Enterprise Bean をマイグレーションする. 59
bonusPoint フィールドを User エンティティーにマイグレーションする. 59
スキーマおよびテーブル・マッピングを更新して BONUS テーブルを組み込む. 60
デプロイメント・コードを生成する 61
カスタマイズされた JSP ファイルをマイグレーションする 62
新規ロジックを Express Store サンプル・ストアでテストする 62

第 5 部 付録 65

特記事項 67
商標. 69

まえがき

本書では、WebSphere® Commerce 開発環境を WebSphere Commerce Studio バージョン 5.4 から WebSphere Commerce - Express Developer Edition バージョン 5.5 にマイグレーションする方法を説明しています。

WebSphere Commerce バージョン 5.4 用のフィックスパックまたは Commerce Enhancement Pack を適用済みで、そのバージョンの WebSphere Commerce で WebSphere Commerce Studio バージョン 5.4 を使用している場合は、そのコード・レベルからのマイグレーションもサポートされています。

本書の更新

このマイグレーション・ガイドおよびその更新版は、 WebSphere Commerce Technical Library から入手可能です。

本書の完成後に本製品に加えられた変更を確認するには、その更新済み製品の README ファイルを参照してください。このファイルは上記の Web サイトにも用意されています。 WebSphere Commerce Studio バージョン 5.4 および対応の製品がインストールされていないマシンに WebSphere Commerce 開発環境をインストールする場合は、ご使用のデータベース用の「*WebSphere Commerce Studio インストール・ガイド*」を参照してください。

本書の表記規則

本書では以下の強調表記規則を使用します。

- **太文字**は、コマンドまたは、フィールド名、アイコン、メニュー選択などのグラフィカル・ユーザー・インターフェース (GUI) コントロールを示します。
- `モノスペース (Monospace)` は、ファイル名、ディレクトリーのパスや名前などの、示されたとおりに正確に入力する必要のあるテキストの例を示します。
- *イタリック* は、語を強調するために使用します。イタリックは、ご使用のシステムの該当する値に置換しなければならない名前も示します。以下のいずれかの名前が出てきたら、説明するとおりに、ご使用のシステムの値に置換してください。

  host_name
  WebSphere Commerce 開発環境マシンの完全修飾ホスト名。たとえば、ibm.com は完全修飾名です。

  drive 当該製品またはコンポーネントがインストールされているドライブを表す文字。たとえば C: です。

  WC_installdir
  WebSphere Commerce - Express 5.5 のインストール・パス。デフォルトのインストール・パスは、¥Program Files¥WebSphere¥CommerceServer55 です。たとえば、c:¥Program Files¥WebSphere¥CommerceServer55 になります。

WCDE_installdir

WebSphere Commerce 開発環境のインストール・パス。デフォルトのインストール・パスは、¥WebSphere¥CommerceDev55 です。たとえば、c:¥WebSphere¥CommerceDev55 になります。

WSAD_installdir

WebSphere Studio Application Developer バージョン 5.1 のインストール・パス。デフォルトのインストール・パスは、¥WebSphereStudio です。たとえば、c:¥WebSphereStudio になります。

WSAD_workspace

ご使用の WebSphere Studio Application Developer バージョン 5.1 ワークスペースのインストール・パス。デフォルトのインストール・パスは、¥WebSphere¥workspace_db2 です。たとえば、c:¥WebSphere¥workspace_db2 になります。

このアイコンは、ヒント (作業を完了するために役立つ追加情報) を表すマークです。

詳細情報の入手方法

WebSphere Commerce 開発環境に関する詳細情報は、WebSphere Commerce 開発環境サイトを参照してください。

WebSphere Commerce についての詳細は、以下の Web サイトを参照してください。

- WebSphere Commerce Technical Library。ここには、ガイドおよびオンライン・ヘルプなど、更新済みドキュメンテーションのダウンロードに関するセクションがあります。
- WebSphere Commerce Support
- WebSphere Commerce - Express の概説

第 1 部 マイグレーション前

この部では、 WebSphere Commerce 開発環境へのマイグレーションを行う前に確認しておくべき重要な情報を解説します。ハードウェアおよびソフトウェアの前提条件、セットアップ環境および可能なマイグレーション環境、および重要なコード変更に関する情報がこの部に記載されています。

第 1 章 WebSphere Commerce 開発環境のマイグレーションの前に

この章では、WebSphere Commerce Studio バージョン 5.4 から WebSphere Commerce - Express Developer Edition バージョン 5.5 へマイグレーションする前に必要な注意事項と準備作業を説明します。

マイグレーションの前提条件

このガイドに記載されているマイグレーションの手順を実行するには、ご使用の WebSphere Commerce Studio バージョン 5.4 のマシンが、以下のセクションで述べるハードウェアおよびソフトウェアの要件を満たしていなければなりません。

前提条件となるハードウェア

- 専用の Pentium® III 733 MHz を搭載した IBM 互換のパーソナル・コンピューターで、以下の要件を満たすもの。
  - 512 MB 以上のランダム・アクセス・メモリー (RAM)。
  - 選択してインストールするオプションごとに、以下のディスク・スペース容量:
    - ストアフロント資産開発の場合:
      - WebSphere Studio — 300 MB
      - ストア・アーカイブ・ツール — 25 MB
      - Blaze Advisor および Blaze Innovator Workbench — 55 MB
    - バックエンド開発の場合:
      - WebSphere Application Developer for Java™ Enterprise — 1.2 GB
      - IBM® DB2® Universal Database — 300 MB
      - WebSphere Commerce 開発環境 — 300 MB
      - Blaze Advisor™ サーバー — 45 MB
    - IBM 分散デバッガー — 160 MB
    - アプレット・デザイナー — 15 MB
    - Page Detailer — 10 MB

    この他に、さらに C: ドライブに 50 MB 必要になります。使用マシンの区画が FAT でフォーマットされており、その区画が 1.024 GB を超えている場合は、その 2 倍の空きディスク・スペースが必要になります。インストールの際に、必要な十分な空きディスク・スペースがあるかどうかのチェックが行われ、十分なスペースがない場合は警告が出されます。
  - CD-ROM ドライブ。
  - 表示色が 256 色以上で、画面解像度の設定が 800x600 以上のグラフィックス対応モニター。
  - マウスまたはその他のポインティング・デバイス。

– TCP/IP プロトコルをサポートしているローカル・エリア・ネットワーク (LAN) アダプター。

前提条件となるソフトウェア

WebSphere Commerce - Express Developer Edition バージョン 5.5 へのマイグレーションを開始する前に、少なくとも以下のソフトウェア要件を満たしておいてください。

- Windows® 2000 Server Edition または Advanced Server Edition をインストール済み (Service Pack 3 を適用) であることを確認してください。更新情報を http://www.microsoft.com から入手できます。

> **重要**
>
> WebSphere Commerce Studio バージョン 5.4 を Windows NT® で使用している場合は、 WebSphere Commerce - Express Developer Edition バージョン 5.5 へのマイグレーションを行う前に、そのオペレーティング・システムを Windows 2000 にアップグレードする必要があります。このバージョンの WebSphere Commerce 開発環境は、 Windows NT 上ではサポートされません。

- WebSphere Commerce Studio へのアクセスに使用する各マシンに Internet Explorer 5.5 Service Pack 1 をインストールし、重要なセキュリティーに関する最新の更新ファイルを適用してください。 Microsoft® Internet Explorer のコピーは、 Microsoft Web サイト (http://www.microsoft.com) からダウンロードして入手することができます。
- WebSphere Commerce 開発環境のインストール後は、IBM WebSphere Commerce 5.5.0.2 フィックスパックをインストールすることが必要です。このフィックスパックには、マイグレーション・プロセスの修正および機能拡張が組み込まれています。このフィックスパックは、WebSphere Commerce サポート・サイトから入手することができます。使用するエディションのリンクをクリックし、ソフトウェアのダウンロードのセクションを参照します。このフィックスパックをダウンロードして、示されるインストールの手順に従います。 (WebSphere Studio Application Developer バージョン 5.1 から WebSphere Studio Application Developer バージョン 5.1 へのアップグレードを計画している場合には、その手順もフィックスパックの資料に記載されています。) 本書のマイグレーション・ステップ (インスタンスまたはデータベースのマイグレーションなど) を続行する前に、必ずこのフィックスパックをインストールします。

マイグレーション環境

WebSphere Commerce Studio バージョン 5.4 から WebSphere Commerce - Express Developer Edition バージョン 5.5 へのマイグレーションで、 WebSphere Commerce 開発環境を新規インストールに相当する環境にマイグレーションしたようにすることができます。つまり、WebSphere Commerce 開発環境のマイグレーションが完了した後に、マシンから不要な WebSphere Commerce 資産を除去することができます。

WebSphere Commerce のローカル・インスタンスを保持し続けたい場合は、それらの資産を除去する必要はありません。この場合は、WebSphere Commerce のローカル・インスタンスを WebSphere Commerce 5.5 レベルにマイグレーションすることをお勧めします。

WebSphere Commerce 開発環境のマイグレーションの結果の最終的な環境として、以下がサポートされます。

- 環境 A — 純粋な開発環境。
- 環境 B — 開発環境と WebSphere Commerce が共存するが資産は共用しない。

これらのシナリオでのマイグレーション・パスを、以降のセクションで説明します。いずれの場合の環境でも、リモート WebSphere Commerce データベースかローカル (スタンドアロン構成) データベースを使用することができます。また、いずれの環境でも、ストアフロント資産 (ストアのルック・アンド・フィールなど) の開発、ストアのバック・オフィスのロジック (カスタム・コマンドや Enterprise Bean など) の開発、あるいはストアフロント資産とバック・オフィスのロジックの組み合わせの開発を行うことができます。

環境 A または B を使用する WebSphere Commerce - Express Developer Edition バージョン 5.5 にマイグレーションするステップの概要は、以下のようになります。

__ 1. 3 ページの『マイグレーションの前提条件』および 4 ページの『マイグレーション環境』に記載されていることを再確認します。

__ 2. 21 ページの『WebSphere Commerce 資産および WebSphere Commerce 開発環境資産のバックアップ』の説明に従って、既存の WebSphere Commerce Studio バージョン 5.4 資産と WebSphere Commerce 5.4 資産のバックアップをとります。

__ 3. 21 ページの『DB2 製品レベルのアップグレード』 の概説に従って、DB2 - Express バージョン 8.1 にアップグレードします。

__ 4. WebSphere Commerce Studio バージョン 5.4 から WebSphere Commerce - Express Developer Edition バージョン 5.5 にアップグレードします。インストールおよびアップグレードの詳細なステップを、 22 ページの『WebSphere Commerce 開発環境のアップグレード』に記載しています。これには、WebSphere Studio Application Developer バージョン 5.1 のインストールも含まれます。

__ 5. IBM WebSphere Commerce 5.5.0.2 フィックスパックをインストールします。このフィックスパックには、マイグレーション・プロセスの修正および機能拡張が組み込まれています。このフィックスパックは、WebSphere Commerce サポート・サイトから入手することができます。使用するエディションのリンクをクリックし、ソフトウェアのダウンロードのセクションを参照します。このフィックスパックをダウンロードして、示されるインストールの手順に従います。本書のマイグレーション・ステップ (インスタンスまたはデータベースのマイグレーションなど) を続行する前に、必ずこのフィックスパックをインストールします。

__ 6. 既存の WebSphere Commerce Studio バージョン 5.4 データベース・スキーマを、 25 ページの『WebSphere Commerce 開発環境データベース・スキーマのマイグレーション』の説明に従って WebSphere Commerce - Express Developer Edition バージョン 5.5 レベルにマイグレーションします。

__ 7. 25 ページの『WebSphere Commerce インスタンスのマイグレーション』の説明に従って、既存の WebSphere Commerce インスタンスを WebSphere Commerce 5.5 レベルにマイグレーションします。

注: インスタンスのマイグレーションでは、 VisualAge® for Java のストアフロントまたは Web 資産 (JSP ファイル、HTML ページ、グラフィックス) を、 WebSphere Studio Application Developer に自動的にコピーします。ただし、 WebSphere Commerce Studio バージョン 5.4 のストアフロント用に作成されたすべての資産についてアカウントを持っていることが必要ですので、このことも確認してください。

既存の WebSphere Commerce インスタンスを WebSphere Commerce 5.5 レベルにマイグレーションするには、以下を行います。

- instance.xml ファイルをマイグレーションする。
- VisualAge for Java のストアフロントまたは Web 資産を WebSphere Studio Application Developer Web プロジェクトにコピーする。該当するものとして、ストアフロントで使用していた JSP ファイル、HTML ページ、あるいはグラフィックスがあります。
- Enterprise Bean 以外のカスタム・コードを含む JAR ファイルを、WebSphereCommerceServerExtensionsLogic という名前の WebSphere Studio Application Developer Java プロジェクトに抽出する。
- インスタンス・マイグレーション・スクリプトを実行したときにコピーされた JSP ファイルをマイグレーションして JSP 1.2 仕様に適合するようにする。

__ 8. VisualAge for Java に含まれる WebSphere Commerce のカスタマイズ・コードを WebSphere Studio Application Developer バージョン 5.1 用に変換します。この詳細ステップは、 30 ページの『カスタマイズまたは拡張されたコードの変換』で説明します。

__ 9. 41 ページの『決済インスタンスおよび決済データベースのマイグレーション』の説明に従って、 WebSphere Commerce 決済データベースおよびインスタンスをマイグレーションします。 WebSphere Commerce 5.4 では、WebSphere Commerce が置かれているマシンとは別のマシンに決済データベースをインストールする必要がありました。 WebSphere Commerce 5.5 では、WebSphere Commerce 開発環境と同じマシンにローカルに決済データベースをインストールすることを選択できます。

__ 10. 45 ページの『第 4 章 マイグレーション後のアクション』の説明に従って、マイグレーション後の各ステップを完了します。このとき WebSphere Commerce のいくつかのコンポーネントのアンインストールも行います。

環境 A — 純粋な開発環境

純粋な開発環境上のマイグレーションの場合、WebSphere Commerce 開発環境はストア資産とロジックの開発に使用し、実動ストアのホストとしては使用しません。この種のマイグレーションの最終的な構造は、 WebSphere Commerce 開発環境のまったく新規のインストールの場合と同じようになります。つまり、純粋な開発環境では、マイグレーション後はマシン上に WebSphere Commerce は存在せず、マシン

上に存在するのは WebSphere Commerce 開発環境のみです。この場合、既存の開発ストアはマイグレーションされ、 FashionFlow サンプル・ストアがインストールされます。

環境 B — 開発環境と WebSphere Commerce が共存するが資産は共用しない

開発環境と WebSphere Commerce が共存するが資産は共用しない場合の最終的な構造は、開発環境が WebSphere Commerce 5.5 のローカル・インスタンスから完全に分離されて搭載された 1 台のマシンになります。つまり、この環境では、WebSphere Commerce と WebSphere Commerce 開発環境の両方が 1 台のマシン上に存在します。開発環境と WebSphere Commerce の両方がデータベースを使用し、それぞれのデータベースの内容は同じですが共用されません。マイグレーションの完了後は、それぞれのデータベースは同じ内容になるようにしておき、 XML ファイルも同じになるようにしておく必要があります。この場合、データベースを 2 回マイグレーションする必要はありません。その代わりに、WebSphere Commerce 開発環境をマイグレーションした後に、マイグレーション済みのデータベースのバックアップをとり、次に新規データベース (例: `Mall2`) を作成してから、その新規データベースに復元します。これについての詳細は、ステップ 3 (46 ページ) を参照してください。

第 2 章 WebSphere Commerce 開発環境へのマイグレーションに伴うコードの変更

マイグレーションの前提条件とマイグレーションが可能な環境を確認したら、WebSphere Commerce Studio バージョン 5.4 のマイグレーションに関係のあるコードの変更を検討する必要があります。この章では、WebSphere Commerce 開発環境のマイグレーションを行う前に注意しておかなければならないコードの変更について解説します。

WebSphere Commerce 5.4 および WebSphere Commerce Studio 5.4 のセットアップ環境

WebSphere Commerce Studio バージョン 5.4 では、 WebSphere Commerce 開発環境と WebSphere Commerce を同一マシンに共存させてインストールすることを選択できました。また、VisualAge for Java をインストールするときに、 WebSphere Test Environment で使用するサンプル・ストアをインストールする選択肢もありました。サンプル・ストアを選択した場合は、以下のものが作成されています。

- データベース。たとえば、デフォルトの VAJ_DEMO データベースです。インストール時にこれを変更することができます。
- ストアのディレクトリー構造。
- インスタンス XML ファイル。たとえば、デフォルトの vaj_demo.xml インスタンス XML ファイルです。インストール時にこれを変更することができます。

これらの資産は WebSphere Test Environment によって使用されます。 WebSphere Commerce Studio バージョン 5.4 が使用できるサンプル・ストアは InFashion ストアであり、WebSphere Commerce 開発環境のインストール・プログラムはマスター XML ファイルを使用してサンプル・ストアをデータベースに取り込みました。

WebSphere Commerce Studio バージョン 5.4 では、WebSphere Commerce 開発環境と WebSphere Commerce を別々のマシンに置くこともできます。

見積要求 (RFQ)

見積要求 (RFQ) は、バイヤー組織が特定の商品または特定の商品のセットの適切な価格を得るために、販売組織にオファーを求めるプロセスです。バイヤー組織のバイヤーは販売組織のカタログで希望の商品を見つけられない場合があります。

WebSphere Commerce 5.5 では、バイヤーが「オーダー」アイテムの RFQ を作成できるように RFQ 要求ツールが変更されました。「オーダー」アイテムの属性を完全に指定できるように、他の RFQ でのアイテムと同様、Personalization 属性が使用されます。新規または変更した (コマンドまたはデータベース) 資産を含めて RFQ のマイグレーションの詳細、および RFQ 要求ツールのマイグレーション方法の詳細については、「*WebSphere Commerce* マイグレーション・ガイド」の『見積要求 (RFQ) のマイグレーション』のセクションを参照してください。

さらに、サポートされる API、推奨されない API、予約済み API、および廃止された API については、 WebSphere Commerce 5.5 のオンライン・ヘルプを参照してください。

コラボレイティブ・ワークスペース

WebSphere Commerce 5.4 では、コラボレイティブ・ワークスペースの機能には CollabManagerBean Enterprise Bean が使用されていました。 WebSphere Commerce 5.5 では、この Enterprise Bean はクラス CollabManager に置き換えられました。CollabManagerBean Enterprise Bean を使用するカスタマイズ・コードがある場合、そのコードを新規の CollabManager クラスと連動するようにマイグレーションするには、以下のようにします。

1. クラスまたはパッケージをインポートするステートメントを以下のように編集して、新規の CollabManager クラスを参照するようにします。
   a. コードの中の CollabManagerBean クラスを参照するインポート・ステートメント (例: `import com.ibm.commerce.collaboration.objects.CollabManagerBean;`) を除去します。
   b. CollabManager クラスを参照する新規のインポート・ステートメント (例: `import com.ibm.commerce.collaboration.manager.*;`) を挿入します。
2. CollabManager タイプを使用するための宣言およびインスタンス・ステートメントを編集します。
3. オブジェクトによって呼び出されるメソッドを、下記の API マッピングの表に従って編集します。新規 API をどのようにマップするかで、さらにコードを追加することが必要な場合があります。

   注: 対応する API のパラメーターや戻りタイプが CollabManagerBean と同じではないものがあります。詳しくは、CollabManager クラスの API の解説で確認してください。

   次の表は、CollabManagerBean クラスと CollabManager クラスの API のマッピングです (Enterprise Bean 仕様として必要な各インターフェースはここには含まれていないので、ご注意ください)。

表 1. CollabManagerBean クラスと CollabManager クラスの API のマッピング

CollabManagerBean でのメソッド	CollabManager クラスでの対応するメソッド
addAuthorGroupMember (CollabSpaceBean csbean)	次のいずれか: • addCWMembers(CollabWorkspaceInfo csbean) • addCWMembers(String collabWorkspaceId,Vector memberDNs,String role) • addCWMember(String collabWorkspaceId,String memberDN,String role)

表 1. *CollabManagerBean* クラスと *CollabManager* クラスの *API* のマッピング *(続き)*

ddManagerGroupMember (CollabSpaceBean csbean)	次のいずれか: • addCWMembers(CollabWorkspaceInfo csbean) • addCWMembers(String collabWorkspaceId,Vector memberDNs,String role) • addCWMember(String collabWorkspaceId,String memberDN,String role)
addReaderGroupMember (CollabSpaceBean csbean)	次のいずれか: • addCWMembers(CollabWorkspaceInfo csbean) • addCWMembers(String collabWorkspaceId,Vector memberDNs,String role) • addCWMember(String collabWorkspaceId,String memberDN,String role)
createCollabSpace (CollabSpaceBean cspaceBean, java.lang.String creatorDN)	createCollabWorkspace(CollabWorkspaceInfo cspaceBean)
deleteAuthorGroupMember (CollabSpaceBean csbean)	次のいずれか: • removeCWMembers(CollabWorkspaceInfo csbean) removeCWMembers(String cwId, Vector memberDNs) • removeCWMember(String cwId, String memberDN)
deleteCollabSpace (java.lang.String collabSpaceID)	removeCollabWorkpace(String collabSpaceID)
deleteManagerGroupMember (CollabSpaceBean csbean)	次のいずれか: • removeCWMembers(CollabWorkspaceInfo csbean) • removeCWMembers(String cwId, Vector memberDNs) • removeCWMember(String cwId, String memberDN)
deleteReaderGroupMember (CollabSpaceBean csbean)	次のいずれか: • removeCWMembers(CollabWorkspaceInfo csbean) • removeCWMembers(String cwId, Vector memberDNs) • removeCWMember(String cwId, String memberDN)
deleteCollabSpace (java.lang.String collabSpaceID)	getCollabWorkspaceDetails(String collabSpaceID)
isUserDNAuthorGroupMember (java.lang.String userDN, java.lang.String collabSpaceID)	getCWMemberRole(String memberDN, String cwId)
getCWMemberRole (String memberDN, String cwId)	getCWMemberRole(String memberDN, String cwId)

表 1. *CollabManagerBean* クラスと *CollabManager* クラスの *API* のマッピング *(*続き*)*

isUserDNReaderGroupMember (java.lang.String userDN, java.lang.String collabSpaceID)	getCWMemberRole(String memberDN, String cwId)
listAllCollabSpaces()	listCollabWorkspaces()
listAuthorGroupMembers (java.lang.String collabSpaceID)	listCWMembers(String collabSpaceID, String role)
listCollabSpaceForUserDN (java.lang.String userDN)	listCollabWorkspaces(String userDN)
listCollabSpaceTemplates()	listCWTemplates()
listCWTemplates()	listCWTemplates()
listReaderGroupMembers (java.lang.String collabSpaceID)	listCWMembers(String collabSpaceID, String role)
modifyAuthorGroupMembers (CollabSpaceBean csbean)	setCWMemberRole(String memberDN, String cwId, String role)
modifyCollabSpaceDescription (java.lang.String collabSpaceID, java.lang.String description)	changeCWDescription(String collabSpaceID,String description)
changeCWDescription (String collabSpaceID,String description)	changeCWDescription(String collabSpaceID,String description)
changeCWDescription (String collabSpaceID,String description)	setCWMemberRole(String memberDN, String cwId, String role)
TESTaddAuthorGroupMembers (java.lang.String csID)	不適用 (対応するメソッドなし)
TESTaddAuthorGroupMembers (java.lang.String csID, java.lang.String userDN) TESTaddManagerGroupMembers (java.lang.String csID)	不適用 (対応するメソッドなし)
TESTaddManagerGroupMembers (java.lang.String csID, java.lang.String userDN)	不適用 (対応するメソッドなし)
TESTaddReaderGroupMembers (java.lang.String csID) TESTaddReaderGroup Members(java.lang.String csID, java.lang.String userDN)	不適用 (対応するメソッドなし)
TESTcreateCollabSpace (java.lang.String bProcType, java.lang.String bProcID)	不適用 (対応するメソッドなし)
TESTdeleteAuthorGroupMembers (java.lang.String csID)	不適用 (対応するメソッドなし)
TESTdeleteAuthorGroupMembers (java.lang.String csID, java.lang.String userDN)	不適用 (対応するメソッドなし)
TESTdeleteManagerGroupMembers (java.lang.String csID)	不適用 (対応するメソッドなし)

表 1. CollabManagerBean クラスと CollabManager クラスの API のマッピング (続き)

TESTdeleteManagerGroupMembers (java.lang.String csID, java.lang.String userDN)	不適用 (対応するメソッドなし)
TESTdeleteReaderGroupMembers (java.lang.String csID)	不適用 (対応するメソッドなし)
TESTdeleteReaderGroupMembers (java.lang.String csID, java.lang.String userDN)	不適用 (対応するメソッドなし)
TESTmodifyAuthorGroupMembers()	不適用 (対応するメソッドなし)
TESTmodifyManagerGroupMembers()	不適用 (対応するメソッドなし)
TESTmodifyReaderGroupMembers()	不適用 (対応するメソッドなし)

次の表は、コラボレイティブ・ワークスペース用のその他の API のマッピングです。

表 2. コラボレイティブ・ワークスペース用のその他の API のマッピング

CollabManagerBean でのメソッド	**対応するインプリメンテーション**
getCollabAppServerIP()	WcsApp.configProperties.getValue (ECConstants.EC_COLLAB_QP_HOST)
getCollabAppServerPort()	WcsApp.configProperties.getValue (ECConstants.EC_COLLAB_QP_HTTP_PORT)

4. CollabManager は Enterprise Bean ではないため、 Enterprise Bean が起こす例外を処理するためにこれまで Enterprise Bean のコーディングに使われていた try および catch のエントリーをすべて除去します。 try および catch のエントリーを除去するのは、コンパイル・エラーにならないようにするためです。
5. コードをコンパイルして、IDE に起因するコンパイル・エラーがあれば修正します。

データベース・スキーマの差異

WebSphere Commerce 5.5 および WebSphere Commerce 開発環境のデータベース・スキーマについての詳細は、 WebSphere Commerce のオンライン・ヘルプを参照してください。このオンライン・ヘルプを起動したら、「**WebSphere Commere 開発情報**」 > 「**参照**」 > 「**データ**」 > 「**データベース・スキーマ**」を選択します。ここから、**「データベース・テーブル」**を選択し、すべてのデータベース・スキーマ情報のアルファベット順リストを表示します。**「このリリースでのデータベース変更」**を選択して、バージョン 5.5 のスキーマ変更を表示します。このオンライン・ヘルプには、データ・モデル情報もあります。

オンライン・ヘルプのデータベース・スキーマ情報の更新を含む、WebSphere Commerce の最新バージョンのドキュメンテーションを、WebSphere Commerce Technical Library (http://www.ibm.com/software/commerce/library/) にアクセスして確認してください。

API の差異

WebSphere Commerce 5.5 および WebSphere Commerce 開発環境の API についての詳細は、 WebSphere Commerce のオンライン・ヘルプを参照してください。このオンライン・ヘルプを起動したら、「**WebSphere Commere 開発情報 (WebSphere Commere Development information)**」 > 「**参照**」 > 「**API 情報 (API information)**」を選択します。このオンライン・ヘルプには、以下の API 情報もあります。

- バージョン 5.5 でサポートされている API。
- バージョン 5.5 で変更された API。これらの API の中にこれまで使用していたものまたはカスタマイズしていたものがあれば、変更されたバージョンを必ず確認してください。 WebSphere Commerce 5.5 および WebSphere Commerce 開発環境で使用するには、 Enterprise Bean レベル 1.1 にマイグレーションしておきます。このこと、および Enterprise Bean の Finder Helper のインプリメンテーションについての詳細は、 『Enterprise Bean の 1.0 から 1.1 仕様へのマイグレーション』および 30 ページの『VisualAge for Java と WebSphere Studio Application Developer の違い』を参照してください。
- IBM の内部使用に予約済みで参照のみを目的とする API。
- バージョン 5.5 では推奨されない API。推奨されない API のいずれかをこれまで使用していた場合は、バージョン 5.5 で使用できる代わりの API をオンライン・ヘルプで参照してください。
- Version 5.5 から廃止されてサポートされなくなった API。
- 5.5 との互換性はなくなったがバージョン 5.1 および 5.4 でサポートされていた API。

Data Bean、Enterprise Bean、データベース・テーブルの相互参照情報、およびコマンド、タスク、データベース・テーブルの相互参照情報もあります。

オンライン・ヘルプの API 情報の更新を含む、WebSphere Commerce の最新バージョンのドキュメンテーションを、 WebSphere Commerce Technical Library (http://www.ibm.com/software/commerce/library/) にアクセスして確認してください。

Enterprise Bean の 1.0 から 1.1 仕様へのマイグレーション

1.0 から 1.1 仕様への移行はアクセス Bean 層またはご使用のコードには影響を与えないため、これまでご使用のコードを WebSphere Commerce 開発環境でも使うことができます。 1.0 から 1.1 への移行は、ご使用の Enterprise Bean に影響を与えます。次の表は WebSphere Commerce 開発環境での Enterprise Bean の変更内容をまとめたものです。これらは 1.0 から 1.1 仕様へのマイグレーションに影響します。

表 3. Enterprise Bean の変更内容と WebSphere Commerce 開発環境への影響

WebSphere Commerce Studio バージョン 5.5 での変更	**変更による Enterprise Bean への影響**
Enterprise Bean の「impl」基本クラスが「bean」基本クラスに名前が変更されました。	名前とパッケージ化の規則に関係する変更であり、影響はありません。

表 3. Enterprise Bean の変更内容と WebSphere Commerce 開発環境への影響 (続き)

BeanBase クラスから「implements java.io.Serializable」が除去されました。	この変更は Enterprise Bean 1.1 仕様に直接の関連はありませんが、ご使用の Enterprise Bean がマイグレーション後に適切に機能するにはこの変更を行う必要があります。
シリアル・バージョン UID が除去されました。	この変更は Enterprise Bean 1.1 仕様に直接の関連はありませんが、ご使用の Enterprise Bean がマイグレーション後に適切に機能するにはこの変更を行う必要があります。
ejbCreate() が基本キー・オブジェクトを戻すようになりました。	この変更を行って Enterprise Bean 1.1 仕様に適合させる必要があります。
ejbCreate()、ejbPostCreate()、ejbActivate()、ejbPassivate()、 ejbStore()、 ejbRemove()、およびすべてのリモート・メソッドから RemoteException が除去されました。	この変更を行って Enterprise Bean 1.1 仕様に適合させる必要があります。
bean または bean 基本クラスから getEntityContext()、setEntityContext()、unsetEntityContext() が除去されました。	内部のクリーンアップ修正であり、影響はありません。
ejbCreate() および ejbPostCreate() コードがすべて bean 基本クラスに移動しました。	内部のクリーンアップ修正であり、影響はありません。
ejbPostCreate() および ejbCreate() メソッドのバランスをとりました。	この変更を行って Enterprise Bean 1.1 仕様に適合させる必要があります。
WCSecurity 役割が追加されました。	この変更は Enterprise Bean 1.1 仕様に直接の関連はありませんが、ご使用の Enterprise Bean がマイグレーション後に適切に機能するにはこの変更を行う必要があります。 WebSphere Studio Application Developer でセキュリティーがオンになっている場合は、Enterprise Bean のセキュリティー役割が必要です。
Enterprise Bean のトランザクション設定が「必要 (Required)」に指定されます。	この変更は Enterprise Bean 1.1 仕様に直接の関連はありませんが、ご使用の Enterprise Bean がマイグレーション後に適切に機能するにはこの変更を行う必要があります。 Enterprise Bean のこのトランザクション設定は必須です。値は Enterprise Bean の使用法によって異なります。
Enterprise Bean ごとに Enterprise Bean の分離レベルが「反復可能読み取り (Repeatable Read)」に指定されます。	この変更は Enterprise Bean 1.1 仕様に直接の関連はありませんが、ご使用の Enterprise Bean がマイグレーション後に適切に機能するにはこの変更を行う必要があります。 Enterprise Bean のこの分離レベルは必須です。値は使用されるデータベース・タイプによって異なります。
JNDI 名が変更されました。	名前とパッケージ化の規則に関係する変更であり、影響はありません。

表 3. *Enterprise Bean* の変更内容と *WebSphere Commerce* 開発環境への影響 (続き)

xyzBeanFinderHelper インターフェースが削除されました。	この変更は Enterprise Bean 1.1 仕様に直接の関連はありませんが、ご使用の Enterprise Bean がマイグレーション後に適切に機能するにはこの変更を行う必要があります。 BeanFinderHelper インターフェースが廃止されました。ファインダーはすべて ibm-ejb-jar-ext.xmi で定義されています。
xyzBeanFinderObjectImpl の名前が xyzBeanFinderObjectBase に変更されました。	この変更は Enterprise Bean 1.1 仕様に直接の関連はありませんが、ご使用の Enterprise Bean がマイグレーション後に適切に機能するにはこの変更を行う必要があります。 BeanFinderHelper インターフェースが廃止されました。ファインダーはすべて ibm-ejb-jar-ext.xmi で定義されています。

MSGTYPES テーブルの固有索引の検査

MSGTYPES テーブルの新規の列として MSGTYPE_ID と NAME があります。MSGTYPE_ID 列は基本キーであり、NAME 列は固有索引です。独自のメッセージ・タイプを作成済みの場合は、それらがこの新しいものと競合しないことを確認してください。競合する場合は、MSGTYPE_ID 値を番号に 10,000 を追加して変更するか、 NAME 値を変更します。カスタム・メッセージ・タイプを参照するコマンドを再コンパイルする必要があります。これを行わないと、 WebSphere Commerce 5.4 のデータをマイグレーションするときに問題が発生する場合があります。

WebSphere アプリケーション・コマンド・フレームワーク

WebSphere Commerce 5.4 では San Francisco コマンド・フレームワークが使われました。しかし WebSphere Commerce 5.5 では WebSphere アプリケーション・コマンド・フレームワークを使用するようになりました。そのため、カスタム・コードや拡張コードをマイグレーションするときは、次の点に十分ご注意ください。

- WebSphere Commerce 5.4 のコントローラー・コマンドまたはタスク・コマンドから拡張したカスタム・コマンドはすべて、 WebSphere Application Server バージョン 5 の新規ランタイム・ライブラリーを使用するよう再コンパイルすることが必要です。
- コントローラー・コマンドとタスク・コマンドの呼び出し元は、コマンド・コンテキスト・セットを持っている必要があります。

J2EE Connector Architecture

WebSphere Commerce Suite 5.1 および WebSphere Commerce 5.4 では Common Connector Framework が使われました。しかし WebSphere Commerce 5.5 では J2EE Connector Architecture を使用するようになりました。 WebSphere Commerce Suite 5.1 または WebSphere Commerce 5.4 とのコネクターを使用していた場合は、それ

を再作成する必要があります。 J2EE フレームワークおよび J2EE Connector Architecture についての詳細は、 Sun Microsystem の Web サイト (http://java.sun.com/j2ee/connector) を参照してください。

なお WebSphere Commerce 開発環境には、 J2EE Connector Architecture のアダプター (コネクター) のサンプルとして、 Sample Adapter というサンプル・コネクターも付属しています。

第 2 部 マイグレーション・フロー

この部では、WebSphere Commerce Studio バージョン 5.4 から WebSphere Commerce - Express Developer Edition バージョン 5.5 にマイグレーションするときの標準的なマイグレーション・フローを説明します。マイグレーション・プロセスには一連のステップがあります。各マイグレーション・ステップを実行する手順を、この部の各章で説明します。

第 3 章 WebSphere Commerce 開発環境のマイグレーション

この章では、WebSphere Commerce 開発環境から WebSphere Commerce - Express Developer Edition へアップグレードするための詳細なステップを説明します。

注: WebSphere Commerce 開発環境をアップグレードする前に、ハードウェアおよびソフトウェア要件の要点を含めて、1 ページの『第 1 部 マイグレーション前』に記載されている要点を確認してください。

WebSphere Commerce 資産および WebSphere Commerce 開発環境資産のバックアップ

WebSphere Commerce 開発環境レベルにソフトウェアをアップグレードする場合またはカスタム・コードを変換する場合は、以下のようにして事前に WebSphere Commerce 資産と WebSphere Commerce 開発環境資産のバックアップをとっておいてください。

- 次の WebSphere Commerce 資産のバックアップをとります。
  - WebSphere Commerce ディレクトリー・ツリー
  - WebSphere Commerce データベース

  WebSphere Commerce ディレクトリー・ツリーおよびデータベースのバックアップをとるステップについては、マイグレーションする WebSphere Commerce のバージョンに合う「*WebSphere Commerce マイグレーション・ガイド*」の中のセクションまたは『WebSphere Commerce 5.4 のバックアップ』を参照してください。
- WebSphere Commerce 開発環境資産のバックアップをとります。

DB2 製品レベルのアップグレード

データベース管理システムとして DB2 を使用している場合は、 WebSphere Commerce 資産および WebSphere Commerce Studio 資産のバックアップをとった後に、 WebSphere Commerce 開発環境へのアップグレードを行う前に、DB2 - Express バージョン 8.1 にアップグレードして構成する必要があります。 DB2 バージョン 8.1 を WebSphere Commerce Studio データベースとしてインストールして構成するには、以下のステップの概要に従う必要があります。

1. DB2 マイグレーションの前に、データベースの構成設定を記録します。
2. 診断エラー・レベルを変更します。
3. DB2 マイグレーション用の DB2 サーバーをオフラインにします。
4. DB2 マイグレーション用のデータベースの準備ができていることを確認します。
5. データベースのバックアップをとります。
6. オプション: 複製を使用する場合は、すべての DB2 ログ・ファイルを保存する必要があります。
7. DB2 Enterprise Server Edition バージョン 8.1 をインストールします。

8. データベースをマイグレーションします。
9. オプション: DB2 Explain 表をマイグレーションします。

上記のステップの詳細については、「*DB2 サーバー機能 概説およびインストール*」の第 2 部の『DB2 サーバーの移行』を参照してください。この資料は、DB2 Technical Spport Library (http://www.ibm.com/software/data/db2/library/) から入手することができます。 DB2 テクニカル・サポート バージョン 8 インフォメーション・センターにアクセスする必要があります。

WebSphere Commerce 開発環境のアップグレード

WebSphere Commerce 資産および WebSphere Commerce 開発環境資産のバックアップをとり終えたら、WebSphere Commerce 開発環境をアップグレードすることができます。このとき WebSphere Studio Application Developer バージョン 5.1 のインストールも行われます。また、インストール・プロセスによってシステムは DB2 Universal Database™ Enterprise Edition バージョン 8.1 にアップグレードされます。

注: WebSphere Commerce Studio バージョン 5.4 で使用していた VisualAge for Java や以前のバージョンの WebSphere Studio Application Developer をアンインストールしないでください。これらのアプリケーションから資産とプロジェクトを WebSphere Studio Application Developer バージョン 5.1 にエクスポートすることが必要になります。ただし、WebSphere Commerce - Express Developer Edition バージョン 5.5 は、VisualAge for Java や以前のバージョンの WebSphere Studio Application Developer とは別のディレクトリーにインストールして、競合しないようにしてください。また、VisualAge for Java プロジェクトは、WebSphere Studio Application Developer バージョン 5.1 ワークスペースには作用しないので注意してください。

WebSphere Commerce - Express Developer Edition バージョン 5.5 のインストール・プログラムでは、以下のタスクが行われます。

- WebSphere Commerce 開発環境のコードをインストールします。これには WebSphere Studio Application Developer バージョン 5.1 も含まれています。また、データベース管理システムとして DB2 を使用している場合は、インストール・プロセスによってシステムは DB2 Universal Database Enterprise Edition バージョン 8.1 にアップグレードされます。
- 新規ディレクトリー構造 ¥WebSphere¥CommerceDev55 を作成します。ここには次のサブディレクトリーがあります。
  - ¥properties

    このディレクトリーには、WebSphere Commerce 5.5 レベルのコードのためのプロパティー・ファイルが入ります。
  - ¥bin

    このディレクトリーにはデータベース・マイグレーション・スクリプトが入ります。
- 元の wcs_instances ファイルを新規 ¥WebSphere¥CommerceDev55¥instances ディレクトリー構造にコピーして、それらのファイルを更新して新規構造を反映します。

WebSphere Commerce 開発環境をインストールするには、以下のようにします (インストール手順の詳細は、「*WebSphere Commerce Studio インストール・ガイド*」を参照してください)。

__ 1. CD-ROM ドライブに WebSphere Commerce - Express Developer Edition CD 1 を挿入します。オートプレイが使用可能になっていれば、インストール・プロセスが始まります。そうでない場合は setup.exe を実行してください。

__ 2. システム環境パスから ¥WebSphere¥CommerceDev¥bin ディレクトリー・パスを除去します。このディレクトリーを除去するには、以下のようにします。

a. **「システム・プロパティー (System Properties)」**ウィンドウを開きます。

b. **「環境変数 (Environment Variables)」**を選択します。**「システム変数 (System variables)」**で**「パス」**を選択します。

c. このパスを編集して、前の Commerce Studio Development のパス (例: WebSphere¥CommerceDev¥bin) を除去します。

d. **「OK」** をクリックします。

__ 3. インストール・ウィザードで出されるプロンプトに従います。

__ 4. WebSphere Commerce 開発環境のインストールが完了したら、作成したカスタム・コードが VisualAge for Java に含まれている場合は、そのカスタム・コードを WebSphere Studio Application Developer バージョン 5.1 にマイグレーションする必要があります。

注: このステップが必要なのは、コマンドまたは Data Bean などのバックエンド資産をマイグレーションする場合のみです。

『タスク・コマンド、コントローラー・コマンド、および Data Bean の JAR ファイルへのエクスポート』および 24 ページの『Enterprise Bean の JAR ファイルへのエクスポート』で概説されているように、このプロセスを簡単にするには、VisualAge for Java を始動して、現在の以下のプロジェクトとグループを JAR ファイルにエクスポートします。 (カスタム・コードおよびロジックの変換の詳細については、 30 ページの『VisualAge for Java と WebSphere Studio Application Developer の違い』で説明しています。)

タスク・コマンド、コントローラー・コマンド、および Data Bean の JAR ファイルへのエクスポート

VisualAge for Java を開始して、新規または拡張タスク・コマンド、コントローラー・コマンド、または Data Bean のためのカスタム・コードを含む VisualAge for Java プロジェクトをエクスポートします。この場合、このコードを含むプロジェクト (複数の場合あり) を、以下の手順で JAR ファイルにエクスポートします。

1. VisualAge for Java の「ワークベンチ (Workbench)」ウィンドウで、プロジェクト名 (複数の場合あり) を選択して右マウス・ボタンでクリックし、**「エクスポート」**をクリックします。
2. **「JAR ファイル (Jar file)」**ラジオ・ボタンを選択して**「次へ」**をクリックします。
3. JAR ファイルの名前を指定します。
4. Java ファイルをエクスポートするために**「.java」**チェック・ボックスを選択します。

5. 必要に応じて他のフィールドを埋めます。このタスクの実行方法の詳細は、VisualAge for Java のオンライン・ヘルプを参照してください。

Enterprise Bean の JAR ファイルへのエクスポート

VisualAge for Java を開始して、カスタム Enterprise Bean を以下のように JAR ファイルにエクスポートします (カスタム Enterprise Bean を含むすべての Enterprise Bean グループを、単一の JAR ファイルにエクスポートする必要があります。)

カスタム Enterprise Bean が少ない場合

カスタム Enterprise Bean が非常に少ない場合 (10 またはそれ以下) は、WebSphere Studio Application Developer バージョン 5.1 を使用して Enterprise Bean を再作成することをお勧めします。この方法の詳細は、「*WebSphere Studio Application Developer バージョン 5.1 マイグレーション・ガイド*」を参照してください。

カスタム Enterprise Bean が多い場合

VisualAge for Java を開始し、Export Tool for Enterprise Java Beans 1.1 を使用して、カスタム Enterprise Bean をエクスポートします。以下の手順で、カスタム Enterprise Bean 1.1 を含むすべての Enterprise Bean グループを、単一の JAR ファイルにエクスポートする必要があります。

エクスポート・プロセス中に、「**ターゲット・データベース・タイプ (Target database type)**」選択ボックスが使用不可であれば、エンタープライズ・スキーマ・マッピングが消失していることを示しています。

1. VisualAge for Java で、**「Export Tool for Enterprise Java Beans 1.1」**フィーチャーをワークスペースに追加します。
2. 「ワークベンチ」の EJB ページで Enterprise Bean グループ (複数の場合あり) を右マウス・ボタンでクリックし、**「エクスポート」** > **「EJB 1.1 JAR」**をクリックします。
3. 必要に応じて他のフィールドを埋めます。**「.java」**チェック・ボックスを選択します。
4. 使用データベースを選択します。
5. **「終了」**をクリックします。
6. JAR ファイルを WebSphere Studio Application Developer バージョン 5.1 にインポートします。

IBM WebSphere Commerce 5.5.0.2 フィックスパックの適用

WebSphere Commerce 開発環境のインストール後は、IBM WebSphere Commerce 5.5.0.2 フィックスパックをインストールすることが必要です。このフィックスパックには、マイグレーション・プロセスの修正および機能拡張が組み込まれています。このフィックスパックは、WebSphere Commerce サポート・サイトから入手することができます。使用するエディションのリンクをクリックし、ソフトウェアのダウンロードのセクションを参照します。このフィックスパックをダウンロードして、示されるインストールの手順に従います。本書のマイグレーション・ステップ

(インスタンスまたはデータベースのマイグレーションなど) を続行する前に、必ずこのフィックスパックをインストールします。

WebSphere Commerce 開発環境データベース・スキーマのマイグレーション

WebSphere Commerce 開発環境をアップグレードした後の次のステップとして、WebSphere Commerce 開発環境データベース・スキーマを WebSphere Commerce - Express Developer Edition バージョン 5.5 レベルにマイグレーションします。

WebSphere Commerce 開発環境データベース・スキーマをマイグレーションするには、 migratedb54.bat と呼ばれるデータベース・スキーマ・マイグレーション・スクリプトを実行します。

__ 1. データベース・スキーマのマイグレーション・スクリプトを実行する前に、setenv.bat ファイルの中の JAVA_HOME 環境変数が、使用マシン上の既存のバージョンの IBM Developer Kit, Java Technology Edition を指すようにします。 setenv.bat ファイルは、通常は *WCDE_installdir*¥Commerce¥bin ディレクトリーにあります。

   たとえば、IBM Developer Kit, Java Technology Edition が使用マシンの ¥eclipse¥jre ディレクトリーにインストールされている場合は、 JAVA_HOME を JAVA_HOME=*WCDE_installdir*¥eclipse¥jre にします。デフォルトでは、JAVA_HOME は WebSphere Studio 5.1 のインストール・ディレクトリーを指します (たとえば、JAVA_HOME=C:¥WebSphere¥Studio5¥runtimes¥base_v5¥java¥jre)。

__ 2. データベース・スキーマのマイグレーション・スクリプトを実行する前に、カスタマイズした外部キー参照をすべて除去しておいてください。こうすることによって、マイグレーション時にさまざまなデータベース・エラーが発生するのを回避します。それらの外部キー参照は、データベース・スキーマのマイグレーション後に手動で再ビルドする必要があります。

__ 3. データベース・マイグレーション・スクリプトを実行するには、「*WebSphere Commerce* マイグレーション・ガイド」の中のセクション『データベースのマイグレーション』を参照してください。

   注:

   a. WebSphere Commerce 開発環境では、データベース・スキーマのマイグレーション・スクリプトは *WCDE_installdir*¥commerce¥bin ディレクトリーにあります。

   b. 「*WebSphere Commerce* マイグレーション・ガイド」に記載されているように、 premigratedb スクリプトはコマンド・プロンプトで実行できますが、 migratedb スクリプトは DB2 Command Line Processor ウィンドウから実行する必要があります。

WebSphere Commerce インスタンスのマイグレーション

WebSphere Commerce 開発環境データベース・スキーマを WebSphere Commerce - Express Developer Edition レベルにマイグレーションした後の次のステップとして、WebSphere Commerce インスタンスをマイグレーションします。

WebSphere Commerce インスタンスのマイグレーション・プロセスで、以下のタスクを完了します。

- instance.xml ファイルをマイグレーションします。
- VisualAge for Java のストアフロントまたは Web 資産を WebSphere Studio Application Developer Web プロジェクトにコピーします。該当するものとして、ストアフロントで使用していた JSP ファイル、HTML ページ、あるいはグラフィックスがあります。
- Enterprise Bean 以外のカスタム・コードを含む JAR ファイルを、WebSphereCommerceServerExtensionsLogic という名前の WebSphere Studio Application Developer Java プロジェクトに抽出します。
- データベース・マイグレーション・スクリプトを実行したときにコピーされた JSP ファイルをマイグレーションして JSP 1.2 仕様に適合するようにします。

注: インスタンスのマイグレーションでは、 VisualAge for Java のストアフロントまたは Web 資産 (JSP ファイル、HTML ページ、グラフィックス) を、WebSphere Studio Application Developer に自動的にコピーします。ただし、WebSphere Commerce Studio バージョン 5.4 のストアフロント用に作成されたすべての資産についてアカウントを持っていることが必要ですので、このことも確認してください。

インスタンスをマイグレーションするには、以下の手順で wstudioimenv.bat と wstudioim.bat の 2 つのインスタンス・マイグレーション・スクリプトを実行します。

1. ストアフロント資産をマイグレーションする場合は、 WebSphere Studio Application Developer バージョン 5.1 を開始してストア資産用の Web プロジェクトを作成し、そのプロジェクトの名前をメモしておきます。このプロジェクトは、 wstudioim.bat インスタンス・マイグレーション・スクリプトを実行する前に作成しておく必要があります。このスクリプトを実行するときに、パラメーター値として Web プロジェクト名を指定することが必要になるためです。

   注: バックエンド資産のみをマイグレーションする場合は、このステップはスキップしてもかまいません。

2. *WC_installdir*¥xml ディレクトリーにある product.xml ファイル内のマイグレーション・パスの値がすべて適切に設定されていることを確認します。 product.xml のこれらの値は WebSphere Commerce 5.5 のインストール・プログラムによって設定されますが、これらの値を調べてご使用の環境に合った値になっているのを確認することをお勧めします。特に、<migrationFrom> エレメントとこのエレメントのすべての子エレメントに値が入っていることを確認してください。そうなっていない場合は、値を手動で挿入する必要があります。値は WebSphere Commerce 5.4 で使用されていた product.xml ファイルから求めることができます。そのファイルの <CommerceServer> タグを参照してください。

   <migrationFrom> エレメントは以下の形式になっています。

```
<migrationFrom>
  <edition>
    <name>name</name>
  </edition>
  <version>5</version>
  <release>rel</release>
  <modification>mod</modification>
```

```
  <fixpak>fixpak</fixpak>
  <path>path</path>
  <altpath>long_path</altpath>
</migrationFrom>
```

各パラメーターの意味は次のとおりです。

name マイグレーション元の WebSphere Commerce のエディション。この名前は次のいずれかになります。

- **Pro** — WebSphere Commerce Professional Edition からのマイグレーションであることを示します。
- **Business** — WebSphere Commerce Buesiness Edition からのマイグレーションであることを示します。

rel マイグレーション元の WebSphere Commerce のリリース。 WebSphere Commerce 5.4 に **4** を指定します。

mod マイグレーション元の WebSphere Commerce の修正レベル。この修正レベルの値は常に **0** です。たとえば、WebSphere Commerce 5.4.**0**.1 からマイグレーションする場合、product.xml ファイルでの修正レベルは **0** です。

fixpak マイグレーション元の WebSphere Commerce のフィックスパック・レベル。たとえば、WebSphere Commerce 5.4.0.**4** からマイグレーションする場合は、 product.xml ファイルでのフィックスパック・レベルは **4** です。

path マイグレーション元のバージョンの WebSphere Commerce のインストール・パス。この値は通常、*long_path* の値と同じです。たとえば、WebSphere Commerce 5.4 からマイグレーションする場合は、このパスは *WC54_installdir* (*c:¥WebSphere¥WebSphere Commerce* など) になります。

long_path
マイグレーション元のバージョンの WebSphere Commerce の絶対つまり長いインストール・パス。この値は通常、*path* の値と同じです。たとえば、WebSphere Commerce 5.4 からマイグレーションする場合は、このパスは *WC54_installdir* (*c:¥WebSphere¥WebSphere Commerce* など) になります。

参考例として、 WebSphere Commerce 5.4.0.1 Business Edition からマイグレーションする場合に product.xml ファイルになければならないものを次に示します。

```
<migrationFrom>
  <edition>
    <name>Business</name>
  </edition>
  <version>5</version>
  <release>4</release>
  <modification>0</modification>
  <fixpak>1</fixpak>
  <path>c:¥WebSphere¥WebSphere Commerce</path>
  <altpath>c:¥WebSphere¥WebSphere Commerce</altpath>
</migrationFrom>
```

3. *WCDE_installdir*¥Commerce¥bin¥setenv.bat ファイルを開いて、次の各設定の値が正しいことを確認します。

- WAS_HOME は *WAS_installdir*¥runtimes¥base_v5 に設定されています。
- JAVA_HOME は、WebSphere Commerce 開発環境マシン上の ¥eclipse¥jre ディレクトリーに設定されます。たとえば、*WCDE_installdir*¥runtimes¥base_v5¥java¥jre または C:¥WebSphere¥Studio5¥runtimes¥base_v5¥java¥jre です。
- WCS_HOME は *WC_installdir* に設定されています。これは WebSphere Commerce 5.5 がインストールされているホーム・ディレクトリーです。たとえば C:¥Program Files¥WebSphere¥CommerceServer55 です。

4. *WC_installdir*¥bin¥wstudioimenv.bat ファイルを編集して、次の各設定が正しい値になるようにします。
   - SET WCIM_MIGRATE_FROM は SET WCIM_MIGRATE_FROM=54 にします。
   - SET WC_PATH を *WC_installdir* に設定します。これは WebSphere Commerce 5.5 がインストールされているディレクトリーです。たとえば C:¥Program Files¥WebSphere¥CommerceServer55 です。
   - SET ANT_PATH を *WSAD_installdir*¥runtimes¥base_v5¥lib に設定します。
   - SET WORK_DIR を *WCDE_installdir* の一時作業ディレクトリーに設定します。たとえば *WCDE_installdir*¥temp にします。
   - SET LOG_FILE を、インスタンスのマイグレーションに関する情報をログに記録するファイルの名前に設定します。
   - SET INSTANCE は WebSphere Commerce 5.5 インスタンスの名前です。
5. コマンド・プロンプトで *WCDE_installdir*¥bin ディレクトリーに移動し、次を入力して最初のスクリプトを実行します。

   ```
   wstudioimenv.bat
   ```

6. コマンド・プロンプトで *WCDE_installdir*¥bin ディレクトリーに移動し、次を 1 行で入力して 2 つ目のスクリプトを実行します。

   ```
   wstudioim.bat WSAD_workspacedir VAJ_installdir
     custom_code_JAR_file store_properties_dir
   ```

   各パラメーターの意味は次のとおりです。

   WSAD_workspacedir
   : WebSphere Studio Application Developer バージョン 5.1 ワークスペースのパス。これは WebSphere Commerce 開発環境をインストールしたときに指定したパスです。たとえば、c:¥WebSphere¥workspace_db2 になります。

   VAJ_installdir
   : VisualAge for Java のインストール・パス。デフォルトのインストール・パスは、¥Visual Age for Javaです。たとえば、d:¥Visual Age for Java になります。

   custom_code_JAR_file
   : カスタマイズされた Enterprise Bean、コマンド、 Data Bean などのカスタム・コードを含む JAR ファイル。たとえば、d:¥myCustomCode.jar になります。

   store_properties_dir
   : これまでのバージョンの WebSphere Commerce のストアのプロパティ

ー・ファイルのパス。このパスはたとえば
c:¥WebSphere¥CommerceServerDev¥wc.ear¥wcstores.war
¥WEB-INF¥classes です。

注: ストアフロント資産のみをマイグレーションする場合は、custom_code_JAR_file パラメーターにはブランク値を指定してください (二重引用符を 2 つ使用してブランク値であることを示します)。たとえばここでの例では、次のようにしてインスタンス・マイグレーション・スクリプトを実行します。

```
wstudioim.bat WSAD_workspacedir VAJ_installdir ""
store_properties_dir
```

7. ログ・ファイル *WCDE_installdir*/temp/logs/wstudioim.log を表示して、プログラムが正常に実行されたかどうか確認します。
8. マイグレーションされたインスタンス XML ファイルを、次のように編集します。
   a. WorkspacePath="" を検索します。
   b. 引用符の間に WebSphere Commerce 開発環境のワークスペースのパスを追加します (例: WorkspacePath="D:¥WEBSPH~1¥WORKSP~1")。
9. カスタム Enterprise Bean を、 WebSphere Studio Application Developer バージョン 5.1 内の WebSphereCommerceServerExtensionsData Enterprise Bean プロジェクトにインポートします。 WebSphere Studio Application Developer バージョン 5.1 を開始し、 24 ページの『Enterprise Bean の JAR ファイルへのエクスポート』 ページで VisualAge for Java からエクスポートした Enterprise Bean JAR ファイル (複数の場合あり) を、以下の手順でインポートします。
   a. WebSphere Studio Application Developer の「J2EE」パースペクティブで、**「ファイル」 > 「インポート」 > 「EJB JAR ファイル (EJB JAR file)」**を選択します。**「次へ」**をクリックします。
   b. 使用する Enterprise Bean を含んでいる JAR ファイルを指定します。
   c. Enterprise Bean がインポートされる Enterprise Bean プロジェクトを指定します。具体的には、 **WebsphereCommerceServerExtensionsData** という既存のプロジェクトを選択します。
   d. **「次へ」**をクリックして、 JAR ファイルからすべての Enterprise Bean ファイルを選択します。
   e. **「終了」**をクリックします。エラーが発生した場合は (エラーは「タスク」ビューに表示されます)、35 ページの『WebSphere Studio Application Developer への Enterprise Bean のマイグレーションでのトラブルシューティングのヒント』を参照してトラブルシューティングを行ってください。 Enterprise Bean をインポートしたら、 36 ページの『Enterprise Bean 情報の場所』の手順に従って、 Enterprise Bean およびメソッドのプロパティー、使用するスキーマ、マップなどの場所を決定します。

カスタマイズまたは拡張されたコードの変換

WebSphere Commerce インスタンスをマイグレーションした後の次のステップとして、 VisualAge for Java ワークスペースに含まれる拡張または変更されたビジネス・ロジックおよびコードを、 WebSphere Studio Application Developer バージョン 5.1 レベルに変換します。カスタム・コードまたは拡張コードを変換する際の複雑なタスクに役立つように、本ガイドにチュートリアルを 2 つ用意しています。カスタマイズまたは拡張されたコードのマイグレーションについての詳細は、 47 ページの『第 4 部 コードの変換のためのチュートリアル』を参照してください。

VisualAge for Java と WebSphere Studio Application Developer の違い

VisualAge for Java ワークスペースから WebSphere Studio Application Developer にコードを変換する前に、この 2 つのソフトウェア・アプリケーションの違いを理解しておくことが重要です。以下に、VisualAge for Java と WebSphere Studio Application Developer の主な違いを示します。

- Enterprise Java Beans (EJB) の仕様レベルが 1.0 から 1.1 に変わりました。ただし、WebSphere Studio Application Developer バージョン 5.1 を使用する WebSphere Commerce 開発環境用は 1.1 仕様です。
- Web アプリケーション用の JSP のレベルが 1.1 から 1.2 に変わりました。
- Web アプリケーション用のサーブレットのレベルは 2.3 のままです。
- サポートされている Java 2 プラットフォームのレベルが 1.2 から 1.3 に変わりました (コンパイラーは 1.4 のコード生成をターゲットにすることができますが、 WebSphere Application Server ランタイム環境はまだ 1.3 です)。
- Visual Composition Editor が Visual Editor for Java に置き換わりました。
- VisualAge for Java のバージョン管理と固有ソース・コード・リポジトリーが、ソフトウェア構成管理 (SCM) プラグインのサポートに置き換わりました。VisualAge for Java Tools API が、 WebSphere Studio Workbench プラグイン・アーキテクチャーに置き換わりました。
- VisualAge for Java XML ツールが WebSphere Studio Application Developer XML ツールに置き換わりました。
- VisualAge for Java のプロジェクトの概念が、 WebSphere Studio Application Developer の複数のタイプのプロジェクトに置き換わりました。
- VisualAge for Java EJB AccessBeans (Data AccessBeans ではない) にあるコンバーターは、 WebSphere Studio Application Developer EJB AccessBeans では使用できません。代わりに、基礎になっている Enterprise Bean にある Enterprise Bean のコンバーターおよびコンポーザーを使用してください。 Enterprise Bean のコンバーターおよびコンポーザーについての詳細は、オンライン・ヘルプを参照してください。

拡張または変更された WebSphere Commerce ビジネス・ロジックおよびコードの変換

WebSphere Commerce Studio バージョン 5.4 から WebSphere Commerce - Express Developer Edition バージョン 5.5 へのマイグレーションでは、使用する VisualAge for Java 環境に関係するステップをいくつか実行します。これらのステップには、以下のものがあります。

- VisualAge for Java から Java ファイルとプロジェクト・リソース・ファイルをエクスポートする。
- WebSphere Studio Application Developer を開始して、コードを収容するプロジェクトを新規作成する。
- Java ファイルとプロジェクト・リソース・ファイルを WebSphere Studio Application Developer にインポートする。
- Enterprise Bean を WebSphere Studio Application Developer にマイグレーションする。
- カスタマイズされた WebSphere Commerce Server Enterprise Bean を WebSphere Studio Application Developer にマイグレーションする。
- テスト環境をセットアップして、マイグレーション済みアプリケーションをテストする。

注:

1. WebSphere Commerce 5.4 では Common Connector Framework が使われました。しかし WebSphere Commerce 5.5 では WebSphere アプリケーション・コマンド・フレームワークを使用するようになりました。そのため、カスタム・コードや拡張コードをマイグレーションするときは、次の点に十分ご注意ください。
   - WebSphere Commerce 5.4 のコントローラー・コマンドまたはタスク・コマンドから拡張したカスタム・コマンドはすべて、 WebSphere Application Server バージョン 5 の新規ランタイム・ライブラリーを使用するよう再コンパイルすることが必要です。
   - コントローラー・コマンドとタスク・コマンドの呼び出し元は、コマンド・コンテキスト・セットを持っている必要があります。
2. サポートされる API、推奨されない API、予約済み API、および廃止された API については、 WebSphere Commerce 5.5 のオンライン・ヘルプを参照してください。

VisualAge for Java から Java ファイルとプロジェクト・リソース・ファイルをエクスポートする

コードを WebSphere Studio Application Developer バージョン 5.1 レベルに変換する最初のステップは、 VisualAge for Java から Java ファイルとプロジェクト・リソース・ファイルをエクスポートすることです。

注:

1. VisualAge for Java リポジトリーからのバージョン化されたプロジェクトおよびリソースのバルク・マイグレーションはサポートされていません。ただし、VisualAge for Java ワークスペースにあるプロジェクトおよびリソースはマイグレーションすることができます。プロジェクトまたはリソースのバージョン化さ

れたコピーを WebSphere Studio Application Developer にマイグレーションする場合は、それを一旦 VisualAge for Java ワークスペースに取り込んでからマイグレーションする必要があります。

2. プロジェクトに複数の種類のデータ (例: Enterprise Bean と Java ソース・コード・ファイル) が含まれている場合は、そのデータをタイプに基づいて別々の JAR に分割しなければなりません。

プロジェクトを JAR ファイルにエクスポートするには、以下のステップで行います。

1. エクスポートするプロジェクトが現在 VisualAge for Java ワークスペースにない場合はワークスペースに追加します。
2. VisualAge for Java の「ワークベンチ (Workbench)」ウィンドウでプロジェクト (複数の場合あり) を選択して右マウス・ボタンでクリックし、**「エクスポート」**をクリックします。
3. **「JAR ファイル (Jar file)」**ラジオ・ボタンを選択して**「次へ」**をクリックします。
4. JAR ファイルの名前を入力します。
5. Java ファイルをエクスポートするために**「.java」**チェック・ボックスを選択し、リソース・ ファイルをエクスポートするために**「リソース (resources)」**チェック・ボックスを選択します。
6. 必要に応じて他のフィールドを埋めます。各フィールドでの指定内容についての詳細は、VisualAge for Java のオンライン・ヘルプを参照してください。

WebSphere Studio Application Developer を開始して、コードを収容するプロジェクトを新規作成する

Java ファイルとプロジェクト・リソース・ファイルを VisualAge for Java からエクスポートした後の、 WebSphere Studio Application Developer へのコードの変換の次のステップとして、 WebSphere Studio Application Developer を開始して適切なプロジェクトを作成します。ファイルのインポート先の WebSphere Studio Application Developer のプロジェクトの種類を決定するのに役立つ、マイグレーションの一般的なガイドラインを以下に示します。プロジェクトを作成したりコードをインポートしたりする前に、 WebSphere Studio Application Developer のオンライン・ヘルプを読んで、 WebSphere Studio Application Developer のプロジェクトの各種類をよく理解しておくことを強くお勧めします。

- コードが Web アプリケーションの一部である場合は、そのコードは以下のようにして Web プロジェクトにインポートします。
  - コントローラー・コマンドやタスク・コマンドなどの Java ファイルはすべて、 Web プロジェクトのソース・ディレクトリーにインポートします (それぞれのパッケージ・ステートメントに基づく適切な階層が WebSphere Studio Application Developer によって自動的に作成されます)。
  - JSP ページなどのリソース・ファイルはすべて、 Web プロジェクトの Web コンテンツ・ディレクトリーにインポートします。
- コードが純粋な Java である場合は、そのコードは Java プロジェクトにインポートします。
- コードが Enterprise Bean である場合は、そのコードは Enterprise Bean プロジェクトにインポートします。

Java ファイルとプロジェクト・リソース・ファイルを WebSphere Studio Application Developer にインポートする

WebSphere Studio Application Developer を開始してコードを収容するプロジェクトを新規作成した後の、 WebSphere Studio Application Developer へのコードの変換の次のステップとして、 Java ファイルとプロジェクト・リソース・ファイルを WebSphere Studio Application Developer にインポートします。

注: ファイルを WebSphere Studio Application Developer にインポートするときは、適切なディレクトリーにインポートしてください。コードをインポートする前に、 WebSphere Studio Application Developer のオンライン・ヘルプを読んで、WebSphere Studio Application Developer のプロジェクトの各種類をよく理解しておくことをお勧めします。どの種類のコードをどのフォルダーに含めるかを決定するのに役立ちます。

Java ファイルとプロジェクト・リソース・ファイルを WebSphere Studio Application Developer にインポートするには、以下のステップで行います。

1. WebSphere Studio Application Developer を開いて、「リソース (Resource)」パースペクティブに切り替えます。
2. **「ファイル」** > **「インポート」** > **「ZIP ファイル (Zip file)」**を選択します。**「次へ」**をクリックします。
3. 該当する JAR ファイルをブラウズします。
4. インポートするファイルと、ファイルを入れるプロジェクトまたはフォルダーを選択します。

Enterprise Bean を WebSphere Studio Application Developer にマイグレーションする

Java ファイルとプロジェクト・リソース・ファイルを WebSphere Studio Application Developer にインポートした後の、 WebSphere Studio Application Developer へのコードの変換の次のステップとして、 Enterprise Bean を WebSphere Studio Application Developer にマイグレーションします。このタスクには、 VisualAge for Java から Enterprise Bean をエクスポートする、 Enterprise Bean を WebSphere Studio Application Developer にインポートする、デプロイメント・コードを生成する、という 3 つのサブタスクがあります。

VisualAge for Java から Enterprise Bean をエクスポートする:

1. VisualAge for Java で、**「Export Tool for Enterprise Java Beans 1.1」**フィーチャーをワークスペースに追加します (まだ追加していない場合)。
2. 「ワークベンチ」の EJB ページで Enterprise Bean グループ (複数の場合あり) を右マウス・ボタンでクリックし、**「エクスポート」** > **「EJB 1.1 JAR」**をクリックします。
3. 必要に応じて他のフィールドを埋めます。**「.java」**チェック・ボックスを選択します。
4. 使用データベースを選択します。
5. **「終了」**をクリックします。

Enterprise Bean を WebSphere Studio Application Developer にインポートする:

1. WebSphere Studio Application Developer で、EJB プロジェクトとエンタープライズ・アプリケーション・アーカイブ・プロジェクトを新規作成します。「J2EE」パースペクティブに自動的に切り替わります。
2. **「ファイル」** > **「インポート」** > **「EJB JAR ファイル (EJB JAR file)」**を選択します。**「次へ」**をクリックします。
3. JAR ファイル、EJB プロジェクト、および EAR プロジェクトを選択します。
4. **「次へ」**をクリックして、 JAR ファイルからすべての EJB ファイルを選択します。
5. **「次へ」**をクリックして、従属プロジェクトをすべて選択します。依存関係があるプロジェクトが不確かな場合は、プロジェクトをすべて選択してください。
6. **「終了」**をクリックします。エラーが発生した場合は (エラーは「タスク」ビューに表示されます)、 35 ページの『WebSphere Studio Application Developer への Enterprise Bean のマイグレーションでのトラブルシューティングのヒント』を参照してトラブルシューティングを行ってください。 Enterprise Bean をインポートしたら、 36 ページの『Enterprise Bean 情報の場所』の手順に従って、 Enterprise Bean およびメソッドのプロパティー、使用するスキーマ、マップなどの場所を決定します。

メソッドを編集して java.rmi.RemoteException エラーが起きないようにする:

1. WebSphere Studio Application Developer で「J2EE」パースペクティブに移動します。
2. 「プロジェクト・ナビゲーター (Project Navigator)」ビューで、WebSphereCommerceServerExtensionData¥ejbModule¥com.ibm.commerce.sample.objects¥*enterprise_bean* を開きます。

   enterprise_bean は、 WebSphere Studio Application Developer にインポートした Enterprise Bean の名前です。たとえば MyEnterpriseBean.java のようになります。
3. 次の各メソッドを変更して、これらが java.rmi.RemoteException エラーを throw しないようにします。

   ```
   ejbLoad()
   ejbStore()
   unsetEntityContext()
   ```

   たとえば `public void ejbLoad() throws java.rmi.RemoteException` を `public void ejbLoad()` に変更します。
4. 変更内容を保存します。

アクセス分離レベルを変更する:

1. WebSphere Studio Application Developer で「J2EE」パースペクティブに移動します。
2. 「プロジェクト・ナビゲーター (Project Navigator)」ビューで、¥WebSphereCommerceServerExtensionData¥ejbModule¥META-INF の下の ejb-jar.xml を開きます。
3. **「アクセス」**をクリックします。
4. **「分離レベル (Isolation Level)」**の下にある**「コミット済み (Committed)」**ノードを選択して、**「除去」**をクリックします。

5. 「**追加**」をクリックして「**反復可能読み取り (Repeatable read)**」を選択し、「**次へ**」をクリックします。
6. Enterprise Bean の名前を選択して「**次へ**」をクリックします。
7. Enterprise Bean の名前を拡張表示して、すべてのホーム・メソッド (Enterprise Bean の名前の下の最初のアイコン) のチェック・ボックスを選択します。
8. 「**終了**」をクリックします。
9. 「**反復 (Repeat)**」ノードが「**分離レベル (Isolation Level)**」の下にあることを確認します。
10. 変更内容を保存します。

デプロイメント・コードを生成する:

1. WebSphere Studio Application Developer の「J2EE 階層 (J2EE Hierarchy)」ビューで、「EJB モジュール (EJB Modules)」フォルダーを拡張表示して、新規にインポートした EJB JAR ファイルを選択します。
2. 「**EJB JAR**」ポップアップ・メニューから、「**生成**」 > 「**デプロイメントおよび RMIC コード (Deploy and RMIC Code)**」をクリックします。使用するすべての Enterprise Bean のコードを生成するように選択します。

WebSphere Studio Application Developer への Enterprise Bean のマイグレーションでのトラブルシューティングのヒント: Enterprise Bean を WebSphere Studio Application Developer にマイグレーションするときに役立つことがあるトラブルシューティングのヒントを以下に示します。

- メソッド・ファインダー・ヘルパーをマイグレーションすると、ファインダー・ヘルパー・インターフェースがなくなります。これはファインダー・ヘルパー・インターフェースが XML 記述ファイルに移動するためです。ファインダー・ヘルパー・オブジェクトは通常、このインターフェースをインプリメントしているため、このことが原因で問題になります。この場合はそのインプリメンテーションを除去する必要があります。
- Enterprise Bean で継承を使用する場合、独自のカスタム・ファインダーをビルドすると壊れることがあります。これは、WebSphere Studio Application Developer で生成されるコード内の各フィールドのマッピングが異なるためです。これを解決するには、`EJSCMPxxxxxx` 生成クラスを調べて、 `findbyPrimaryKey` に対応して生成された Select ステートメントを見つけ、これをテンプレートとして使用します。
- WebSphere Studio Application Developer の「設定」ページをセットアップする場合、変更を保管するたびに自動ビルドが行われないようにするには、以下のステップを実行して Enterprise Bean のデプロイメント・コードおよび RMIC スタブを生成する必要があります。
  1. Enterprise Bean プロジェクトを選択して右マウス・ボタンでクリックし、「**生成**」 > 「**デプロイメントおよび RMIC コード (Deploy and RMIC code)**」を選択します。
  2. RMIC コンパイラーが認識できるのはコンパイル済みコードのみですが、生成された Java クラスはまだコンパイルされていないため、エラーになります。このエラーが表示された後、 Java パースペクティブに切り替えてプロジェクトをビルドします。
  3. ステップ 1 を繰り返します。今度はエラーにならないはずです。

4. ステップ 2 を繰り返して、新規に生成されたスタブをコンパイルします。これで、Enterprise Bean をデプロイしてもランタイム・エラーにならなくなります。

Enterprise Bean 情報の場所: 次の表は、Enterprise Bean の各項目と、 Enterprise Bean を WebSphere Studio Application Developer にマイグレーションした後のそれぞれの場所をまとめたものです。

表 4. WebSphere Studio Application Developer バージョン 5.1 へのマイグレーション後の Enterprise Bean 情報の場所

項目	場所
Enterprise Bean (フィールド、クラス)	「J2EE 階層 (J2EE Hierarchy)」ビューで「EJB モジュール (EJB Modules)」フォルダーを拡張表示します。
Enterprise Bean (ファインダー・クラス、Access Bean、生成クラス)	「ナビゲーター (Navigator)」ビューで「EJB モジュール (EJB Modules)」フォルダーを拡張表示し、 `com` ディレクトリー (または使用しているパッケージ構成) に移動します。
関連情報	「J2EE 階層 (J2EE Hierarchy)」ビューで「EJB モジュール (EJB Modules)」フォルダーを拡張表示し、目的の Enterprise Bean JAR ファイルを選択します。そのポップアップ・メニューから、**「開く (Open With)」** > **「EJB Deployment Descriptor」**を選択します。「Deployment Descriptor」で、**「概要」**タブを選択します。
ファインダー・ヘルパーの記述	「J2EE 階層 (J2EE Hierarchy)」ビューで「EJB モジュール (EJB Modules)」フォルダーを拡張表示し、目的の Enterprise Bean JAR ファイルを選択します。そのポップアップ・メニューから、**「開く (Open With)」** > **「EJB Deployment Descriptor」**を選択します。「Deployment Descriptor」で、**「Beans」**タブを選択します。
マッピングまたはスキーマの記述	「J2EE 階層 (J2EE Hierarchy)」ビューで「EJB モジュール (EJB Modules)」フォルダーを拡張表示し、目的の EJB JAR ファイルを選択します。そのポップアップ・メニューから、**「生成」** > **「EJB と RDB のマッピング (EJB to RDB mapping)」**を選択します。
トランザクション区分	「J2EE 階層 (J2EE Hierarchy)」ビューで「EJB モジュール (EJB Modules)」フォルダーを拡張表示し、目的の Enterprise Bean JAR ファイルを選択します。そのポップアップ・メニューから、**「開く (Open With)」** > **「EJB Deployment Descriptor Editor」**を選択します。「EJB Deployment Descriptor Editor」で、**「Beans」**タブを選択します。

表 4. WebSphere Studio Application Developer バージョン 5.1 へのマイグレーション後の Enterprise Bean 情報の場所 (続き)

項目	場所
分離レベル/更新の検索/読み取り専用メソッドのマーキング	「J2EE 階層 (J2EE Hierarchy)」ビューで「EJB モジュール (EJB Modules)」フォルダーを拡張表示し、目的の EJB JAR ファイルを選択します。そのポップアップ・メニューから、**「開く (Open With)」** > **「EJB Deployment Descriptor」**を選択します。「EJB Deployment Descriptor」で、**「アクセス」**タブを選択します。
EJB 環境変数	「J2EE 階層 (J2EE Hierarchy)」ビューで「EJB モジュール (EJB Modules)」フォルダーを拡張表示し、目的の Enterprise Bean JAR ファイルを選択します。そのポップアップ・メニューから、**「開く (Open With)」** > **「EJB Deployment Descriptor」**を選択します。「EJB Deployment Descriptor」で、**「Beans」**タブを選択します。

EJB Access Bean のマイグレーション: Export Tool for Enterprise Java Beans 1.1 を使用して VisualAge for Java Enterprise Edition バージョン 4.0 から Enterprise Bean をエクスポートすると、関連する Java bean ラッパーおよびコピー・ヘルパー Access Bean のメタデータもエクスポートされます。行セット Access Bean は WebSphere Studio Application Developer ではサポートされていないため、その Access Bean のメタデータはエクスポートされません。

顧客ファインダー・ヘルパーのマイグレーション: Export Tool for Enterprise JavaBeans™ 1.1 を使用して VisualAge for Java Enterprise Edition バージョン 4.0 から Enterprise Bean をエクスポートする場合、または Deployment Tool for Enterprise JavaBeans (つまり EJB Deploy Tool) を使用して 1.0 JAR ファイルをデプロイする場合、ファインダー・ヘルパー・インターフェースが拡張文書にマイグレーションされます。 JAR ファイルが Enterprise Bean JAR ファイルの場合は、そのメタデータもファインダー・ヘルパー・インターフェースから拡張文書にマイグレーションされます。ただし、拡張文書へのファインダー・ヘルパー記述子のマイグレーションが行われるのは、 JAR ファイル内のファインダー記述子が拡張文書にない場合のみです。 Export Tool for Enterprise Java Beans 1.1 を使用して Enterprise Bean をエクスポートすると、エクスポートされた JAR の冗長クラスがフィルターに掛けられます。 Export Tool for Enterprise Java Beans 1.1 を使わないでエクスポートした Enterprise Bean を、冗長クラスも一緒にインポートした場合、そのクラスは単に無視されます。

カスタマイズされた WebSphere Commerce Server Enterprise Bean を WebSphere Studio Application Developer にマイグレーションする

Enterprise Bean を WebSphere Studio Application Developer にマイグレーションした後の、 WebSphere Studio Application Developer へのコードの変換の次のステップとして、カスタマイズされた WebSphere Commerce Server Enterprise Bean を

WebSphere Studio Application Developer にマイグレーションします。このタスクには、コードとメタデータの変更を WebSphere Studio Application Developer にある既存の WebSphere Commerce Server プロジェクトにマイグレーションする、デプロイメント・コードと Access Bean を生成する、という 2 つのサブタスクがあります。

コードとメタデータの変更を WebSphere Studio Application Developer にある既存の WebSphere Commerce Server プロジェクトにマイグレーションする:

コードとメタデータの変更を、 WebSphere Studio Application Developer にある既存の WebSphere Commerce Server プロジェクトにマイグレーションするには、以下のようにします。

1. WebSphere Studio Application Developer で、変更を加える WebSphere Commerce Server Enterprise Bean プロジェクトを選択します。
2. Java コードの変更をマージします。これを行うために必要な Enterprise Bean プロジェクトに Java コードの変更を加える方法については、 WebSphere Studio Application Developer の資料を参照してください。
3. テーブル定義または列定義をマージします。これを行うために必要な Enterprise Bean プロジェクト内の既存のデータベース・テーブル定義を変更する方法については、 WebSphere Studio Application Developer の資料を参照してください。
4. コンテナー管理パーシスタンス (CMP) フィールドをマージします。 Enterprise Bean の CMP フィールドを追加または変更する方法については、 WebSphere Studio Application Developer の資料を参照してください。
5. ファインダー・メソッドをマージします。 Enterprise Bean プロジェクトのデプロイメント記述子内のファインダー・メソッドを追加または変更する方法については、 WebSphere Studio Application Developer の資料を参照してください。
6. 変更する WebSphere Commerce Server Enterprise Bean プロジェクトごとに、上記のステップを繰り返します。

注: コードをマージする過程で、警告またはエラーになることがあります。これは正常な状況です。デプロイメント・コードと Access Bean を生成する段階でエラーはなくなります (61 ページの『デプロイメント・コードを生成する』を参照)。

デプロイメント・コードを生成する

デプロイメント・コードを再生成するには、以下のようにします。

1. WebSphere Studio Application Developer の「J2EE」パースペクティブで、 J2EE 階層から**「EJB モジュール (EJB Modules)」**を拡張表示します。
2. 新規作成した EJB JAR ファイルを右マウス・ボタンでクリックして、**「生成」** > **「デプロイメントおよび RMIC コード (Deploy and RMIC Code)」**を選択します。
3. **「すべての EJB (All EJBs)」**を選択して、**「終了」**をクリックします。

注: WebSphere Studio Application Developer への Enterprise Bean のマイグレーションに関して、さらに以下の情報も参照してください。

- 35 ページの『WebSphere Studio Application Developer への Enterprise Bean のマイグレーションでのトラブルシューティングのヒント』
- 36 ページの『Enterprise Bean 情報の場所』

- 37 ページの『EJB Access Bean のマイグレーション』
- 37 ページの『顧客ファインダー・ヘルパーのマイグレーション』

テスト環境をセットアップして、マイグレーション済みアプリケーションをテストする

Enterprise Bean ファイルを WebSphere Studio Application Developer にマイグレーションした後の、 WebSphere Studio Application Developer へのコードの変換の最後のステップとして、テスト環境をセットアップしてマイグレーション済みアプリケーションをテストします。

サーバー・インスタンスを開始するには、以下のステップで行います。

1. WebSphere Studio Application Developer で、ローカル Web サーバーまたは WebSphere Application Server サーバーをクローズします。こうしておかないと WebSphere Commerce が正常に開始されません。
2. データ・ソースのプロパティーのページで、デフォルトのパスワードを、これまで設定していたデータ・ソースのパスワードに変更します。デフォルトでは、**「デフォルト・パスワード (Default password)」**フィールドはブランクです。
3. データ・ソースの設定を行い、データ・ソースの URL の設定が正しいことを確認します。 URL が正しくないと Commerce Server を開始するときにエラーになり、データ・ソースを正常に作成できなくなります。 DB2 の場合、デフォルトの JDBC データ・ソース URL は `jdbc:db2:`*`database_name`* です。たとえば `jdbc:db2:demo_dev` のようになります。データ・ソース URL がこのようになっていることを確認してください。
4. データベースの正しいユーザー・パスワードを指定して、 WebSphere Commerce Server のデータ・ソースを以下のように更新します。
   a. **「スタート」** > **「プログラム」** > **「WebSphere Studio の開始 (Start WebSphere Studio)」** > **「Application Developer 5.0」**と選択して WebSphere Studio Application Developer を開始します。
   b. 「J2EE」パースペクティブの「J2EE 階層 (J2EE Hierarchy)」ビューで、**「サーバー」**を拡張表示して **「WebSphereCommerceServer」**をダブルクリックします。「WebSphereCommerceServer」ビューが表示されます。
   c. 「WebSphereCommerceServer」ビューで**「データ・ソース (Data sources)」**ページを選択します。
   d. 「データ・ソース (Data sources)」ページで、**「サーバーの設定 (Server Settings)」**を拡張表示します。
   e. **JDBC プロバイダー・リスト**・テーブルで、**DB2 JDBC ドライバー**を選択します。

      JDBC プロバイダーを選択すると、 JDBC プロバイダー・リスト・テーブルの下のテーブルが更新されます。
   f. **「上で選択された JDBC プロバイダーで定義されているデータ・ソース (Data source defined in the JDBC provider selected above)」**テーブルで、 **jdbc/WebSphere Commerce DB2 DataSource** *instance_name* を選択します。

      ここで *instance_name* は WebSphere Commerce Studio インスタンスの名前です。デフォルトのインスタンス名は Demo_Dev です。

g. **「編集」**をクリックします。「データ・ソースの変更 (Modify Data Source)」ウィザードが開きます。

h. データ・ソース JDNI 名が以下のようになっていることを確認してください。

```
jdbc/WebSphere Commerce DB2 DataSource instance_name
```

たとえば WebSphere Commerce インスタンス名が **demo** の場合、 JDNI 名は jdbc/WebSphere Commerce DB2 DataSource demo である必要があります。データ・ソース JDNI 名の値が以下のようになっていないと WebSphere Commerce Server は開始されません。その場合はデータ・ソース JDNI 名を変更して、 instance.xml ファイルでの値に合わせる必要があります。

i. 「データ・ソースの変更 (Modify Data Source)」ウィザードで、**「デフォルト・ユーザー・パスワード (Default user password)」**フィールドの値を、**「デフォルト・ユーザー ID (Default user ID)」**フィールドに表示される ID のパスワードに置き換えます。

j. **「終了」**をクリックして、情報を更新して「データ・ソースの変更 (Modify Data Source)」ウィザードを閉じます。

k. **「ファイル」 > 「WebSphereCommerceServer の保管 (Save WebSphereCommerceServer)」**を選択して、更新内容を構成に保管します。

5. 「サーバー」ビューを開きます。パースペクティブで「サーバー」ビューが見つからない場合は、メインメニューで、**「パースペクティブ (Perspective)」 > 「ビューの表示 (Show View)」 > 「サーバー」**を選択します。
6. サーバー・インスタンスを右マウス・ボタンでクリックして、**「発行」**を選択します。
7. 発行が完了したら、**「OK」**をクリックしてポップアップ・メニューを閉じます。
8. サーバー・インスタンスを右マウス・ボタンでクリックして、**「開始」**を選択します。「コンソール (Console)」ビューが表示されます。
9. インスタンスの始動方法に従います。

これで、ストアを発行してマイグレーション済みコードをテストできる状態になりました。 WebSphere Commerce 管理コンソールを使用してストアへのアクセスとセットアップを行うには、以下のステップで行います。

1. ブラウザーのウィンドウを開いて、次の URL を入力します。
   https://*host_name*/webapp/wcs/admin/servlet/ToolsLogon?XMLFile=adminconsole.AdminConsoleLogonhttps
2. 「管理コンソールのログオン」ページで、ユーザー名に `wcsadmin` を入力してパスワードを指定し、**「ログオン」**をクリックします。 (デフォルトとして、ユーザー名とパスワードはいずれもインストール時に `wcsadmin` に設定されますが、このパスワードは最初に WebSphere Commerce 管理コンソールを使用するときに変更しなければなりません。)
3. WebSphere Commerce 管理コンソールにログインしたら、**「サイト」**を選択します。
4. WebSphere Commerce 管理コンソールのメイン・ページにある**「ストア」**で、**「発行」**を選択します。

5. このウィザードに必要な情報を指定し、**「次へ」**をクリックしてページを進めます。
6. 「オプション」ページで**「終了」**をクリックすると、ストアが発行されます。

VisualAge for Java の WebSphere Test Environment と違い、この場合はセットアップの結果次第で HTTPS 要求を実行できることに注目してください。

決済インスタンスおよび決済データベースのマイグレーション

カスタム・コードを WebSphere Studio Application Developer バージョン 5.1 に変換した後の次のステップとして、 WebSphere Commerce Payments (以前の Payment Manager) で使用される WebSphere Commerce の決済インスタンスおよびデータベースをマイグレーションします。外部の決済システムを使用している場合はこのセクションに従う必要はありませんが、使用している決済システムが WebSphere Commerce 5.5 と WebSphere Commerce 開発環境で正しく機能することを確認してください。

WebSphere Commerce 5.4 では、 WebSphere Commerce が置かれているマシンとは別のマシンに決済データベースをインストールする必要がありました。 WebSphere Commerce 5.5 では、WebSphere Commerce 開発環境と同じマシンにローカルに決済データベースをインストールすることを選択できるようになっています。したがって、バージョン 5.5 では決済に関して以下の 2 つのシナリオがあります。

- 決済データベースをリモートに保持する場合は、 Commerce Payments インスタンスおよびデータベースを、 WebSphere Commerce Instance Migrator ツールを使用してマイグレーションする方法の詳細について、「*WebSphere Commerce マイグレーション・ガイド*」を参照してください。
- WebSphere Commerce 開発環境が置かれているマシンと同じマシンに決済データベースをインストールする場合は、決済データベースをマイグレーションして、WebSphere Studio Application Developer バージョン 5.1 で作成する決済インスタンスがマイグレーション済みの決済データベースを使用するようにする必要があります。

第 3 部 マイグレーション後

この部では、WebSphere Commerce 開発環境をマイグレーションした後の必要なステップを説明します。

第 4 章 マイグレーション後のアクション

19 ページの『第 2 部 マイグレーション・フロー』で説明したマイグレーションの各ステップを完了したら、 WebSphere Commerce Studio バージョン 5.4 からのマイグレーションの場合は、以下を完了する必要があります。

- ストアに含まれる JavaServer Page テンプレートを、 Sun Microsystems によって作成された JavaServer Page 1.2 仕様に準拠するようにします。 WebSphere Commerce 5.4 で実行されるストアで使用される JSP テンプレートは、 JSP 1.1 仕様をサポートする必要がありました。ストアを WebSphere Commerce 5.5 にマイグレーションしたら、ストアに含まれる JSP テンプレートを、 Sun Microsystems によって作成された JavaServer Page 1.2 仕様に準拠させる必要があります。 JavaServer Page 1.2 仕様については、 Sun Microsystems の Java Web サイト (www.java.sun.com) を参照してください。
- JavaServer Pages 仕様 1.2 では、サポートされている言語は「java」のみです。したがって、JSP での次に示すページ言語宣言は有効ではなくなります。

  ```
  <%@ page language="JAVA" %>
  ```

  なお、WebSphere Commerce Instance Migrator ツール (決済のマイグレーションに使用) は、 <%@ page language=″JAVA″ %> があれば、ユーザーに代わってそれをすべて <%@ page language=″java″ %> に変換します。
- AbstractAccessBean.getInitContext() を使用して JSP 内の初期コンテキストを検索している場合は、それを AbstractAccessBean.getInitContext(null,null) に変更することをお勧めします。
- JSP ファイルの <jsp:root> および </jsp:root> のセクションが有効なのは JSP ファイルが XML 文書の場合だけであるため、これらのセクションを除去します。 JSP ファイルが XML 文書でない場合 (つまり JavaServer Pages 形式の場合) は、その JSP ファイルから <jsp:root>...</jsp:root> セクションを除去する必要があります。これについて、および XML 文書についての詳細は、 Sun Microsystems の「*JavaServer Pages Specification (Version 1.2)*」の『JSP.5.2 JSP Documents』のセクション、および「*WebSphere Commerce* マイグレーション・ガイド」の『WebSphere Commerce 5.4 JSP ファイルの更新』のセクションを参照してください。
- java.util.* パッケージをインポートします。 WebSphere Application Server バージョン 5 では、「Vector」ディレクティブを使用する JSP ファイルには、java.util.Vector パッケージを明示的に組み込む必要があります。 JSP に次の行がある場合は、 WebSphere Commerce 5.5 で機能させるためにその JSP を変更する必要はありません。

  ```
  <%@ page import="java.util.*" %>
  ```

  たとえば ToolTech サンプル・ストアをマイグレーションした場合、 JSP ページが適切に表示されるようにするには、 JSP ページ (例: CatalogItemAdd.jsp ページ) の先頭に次の行を追加します。

  ```
  <@ page import="java.util.Vector"%>
  ```

java.util パッケージをインポートしないでそのパッケージ内のクラスを使用する場合は、 JSP ファイルに対して以下の変更を行う必要があります。 WebSphere Commerce で使用される共通クラスについては次のとおりです。

Enumeration

この特定のクラスをインポートするには、次を使用します。

```
<%@ page import="java.util.Enumeration" %>
```

Vector

この特定のクラスをインポートするには、次を使用します。

```
<%@ page import="java.util.Vector" %>
```

ResourceBundle

この特定のクラスをインポートするには、次を使用します。

```
<%@ page import="java.util.ResourceBundle" %>
```

WebSphere Commerce 5.5 が提供するクラスについて詳しくは、 WebSphere Commerce のオンライン・ヘルプを参照してください。

環境固有のマイグレーション後アクション

マイグレーション後のアクションの一部は、以下に説明するように、使用環境によって異なります。

環境 B

環境 B (開発環境と WebSphere Commerce が共存するが資産は共用しない環境) の場合は、以下を実行します。

1. *instance*.xml ファイルの名前を、 (識別のみのために) 固有の名前に変更します。たとえば、このファイルの名前を demo.xml から demo2.xml に変更します。
2. WebSphere Commerce インスタンスのディレクトリー (WebSphere¥WCS¥instances) にある wcs_instances ファイルを、新しい XML ファイルを指すように変更します。
3. CommerceServerDev¥instances¥*instance*¥xmlディレクトリーにある *instance*.xml ファイルを編集して、データベース用の `name` エントリーを新しい名前 (例: `Mall2`) に更新します。以下に例を示します。

```
<Database>
<DB DBMSName="DB2"
    RunDB2SG="true"
    SummaryTable="true"
    OraUserID=""
    DBAName=""
    RemoteDB="false"
    DBAPwd=""
    DBServerPort=""
    DBNode=""
    DBHost=""
    DBUserID=""
    DBUserPwd=""
    name="Mall2"
    active="true"
    StagingEnable="false" />
</Database>
```

第 4 部 コードの変換のためのチュートリアル

以下のチュートリアルでは、カスタマイズまたは拡張されたコードを効果的にマイグレーションする方法を説明します。各チュートリアルは、「*WebSphere Commerce プログラミング・ガイドとチュートリアル*」に記載されているチュートリアルを拡張したものです。たとえば、プログラミング・ガイドのチュートリアルではコントローラー・コマンドを拡張する方法を説明していますが、このマイグレーション・チュートリアルでは拡張コマンドを WebSphere Studio Application Developer にマイグレーションする方法を説明します。

チュートリアルを実際に試してみる場合は、ご使用の WebSphere Studio Application Developer 開発環境に、すべての VisualAge for Java ストア資産を事前にマイグレーションしておいてください。 WebSphere Studio Application Developer に JSP ファイルをマイグレーションするときのステップは、以下のようになります。

1. WebSphere Studio Application Developer を開いて、「リソース (Resource)」パースペクティブに切り替えます。
2. **「ファイル」** > **「インポート」** > **「フォルダー」**をクリックして、**「次へ」**をクリックします。
3. ディレクトリー *VAJ_installdir*¥ide¥project_resources¥IBM WebSphere Test Environment¥hosts¥default_host¥default_app¥web をブラウズします。
4. サブフォルダーも含めて、インポートするファイルをすべて選択します。
5. そのファイルを入れるフォルダーとして、フォルダー ¥Stores¥Web Content を選択します。

第 5 章 チュートリアルのマイグレーション・フロー

以下は、このガイドで要点を説明するチュートリアルのエンドツーエンドのシナリオ、またはフローをハイレベルで示したものです。ここではシナリオとして 3 台のマシンを使用します。マシン A は実動が目的のマシンで WebSphere Commerce 5.4 が稼働しています。マシン B は空のマシンでマイグレーションのためのものです。マシン C は VisualAge for Java を実行する開発マシンです。以下はこのシナリオに必要なステップです。

1. 必要なソフトウェア・スタックをマシン B にインストールします。
2. マシン A から B へのインスタンスのマイグレーションを実行します。このインスタンス・マイグレーションはリモートで行うこともできます。
3. マシン A のデータベースをマシン B にコピーします。
4. マシン B でデータベース・マイグレーションを実行します。
5. WebSphere Studio Application Developer バージョン 5.1 をマシン C にインストールします。
6. カスタム・コードおよび Enterprise Bean を VisualAge for Java からエクスポートして WebSphere Studio Application Developer にインポートします。
7. カスタム・コードおよび Enterprise Bean をマイグレーションして単体テストを行います。
8. カスタム・コードおよび Enterprise Bean を、マシン A から B へのインスタンス・マイグレーションを実行したステップで得られたマイグレーション済み EAR ファイルにデプロイ (アセンブル) します。デプロイメントについての詳細は、「*WebSphere Commerce プログラミング・ガイドとチュートリアル*」ガイドを参照してください。
9. EAR をデプロイ (インストール) します。デプロイメントについての詳細は、「*WebSphere Commerce プログラミング・ガイドとチュートリアル*」ガイドを参照してください。
10. すべてのカスタム・コードおよび Enterprise Bean が適切に機能することをテストします。
11. マシン A での実動をシャットダウンします。
12. マシン A のデータベースをマシン B にコピーします。
13. マシン B でデータベース・マイグレーションを実行します。
14. 実動サーバーとしてマシン B を使用するように構成を切り替えます。
15. マシン B で WebSphere Commerce 5.5 を開始します。

第 6 章 チュートリアル A: 拡張またはカスタマイズされたコントローラー・コマンドのマイグレーション

以下のチュートリアルでは、拡張コントローラー・コマンドのマイグレーションを行います。このマイグレーションには、コントローラー・コマンド自体をマイグレーションする、コマンドを表示する JSP ファイルをマイグレーションする、コマンドに関連した Enterprise Bean をマイグレーションする、すべてのものが正常にマイグレーションされていることをテストする、というサブステップがあります。

WebSphere Commerce 5.4 向けの「*WebSphere Commerce プログラマーズ・ガイド*」に記載されているチュートリアルでは、既存の OrderProcess コントローラー・コマンドを、購入に基づいて蓄積されている合計ボーナス・ポイントが「オーダーの確認」ページに表示されるように拡張する方法を例示しています。ここで行うチュートリアルでは、コントローラー・コマンドのマイグレーション手順のモデルとして、新規の拡張コマンド OrderProcessCmdBonusImpl を作成します。

Enterprise Bean をマイグレーションする

前記の「*WebSphere Commerce プログラミング・ガイドとチュートリアル*」ガイドのチュートリアルでは、 BonusBean Enterprise Bean を作成しました。この Enterprise Bean を WebSphere Studio Application Developer にマイグレーションする場合、 Enterprise Bean を 1.1 JAR としてエクスポートする、WebSphereCommerceServerExtensionData フォルダーにある ExtensionJDBCHelperBean を削除する、 Enterprise Bean を WebSphere Studio Application Developer にインポートする、デプロイメント・コードを生成する、というサブステップがあります。

VisualAge for Java から Enterprise Bean をエクスポートする

1. VisualAge for Java で、**「Export Tool for Enterprise Java Beans 1.1」**フィーチャーをワークスペースに追加します (まだ追加していない場合)。
2. 「ワークベンチ」の EJB ページで **「WCSSamplesEntity Bean」** Enterprise Bean グループを右マウス・ボタンでクリックし、**「エクスポート」** > **「EJB 1.1 JAR」**をクリックします。
3. 必要に応じて他のフィールドを埋めます。**「.java」**チェック・ボックスを選択します。
4. データベースに UDB DB2 を選択します。
5. 使用データベースを選択します。
6. **「終了」**をクリックします。

WebSphereCommerceServerExtensionData フォルダーの ExtensionJDBCHelperBean を削除する

1. WebSphere Studio Application Developer で「J2EE」パースペクティブに移動します。

2. 「J2EE 階層 (J2EE Hierarchy)」ビューで**「EJB モジュール (EJB Modules)」**を拡張表示します。
3. **「WebSphereCommerceServerExtensionsData」** フォルダーを拡張表示します。
4. **「Session Beans」**を拡張表示します。
5. **「ExtensionJDBCHelperBean」**を右マウス・ボタンでクリックし、**「削除」**をクリックします。
6. **「全選択」**をクリックして**「OK」**をクリックします。

Enterprise Bean を WebSphere Studio Application Developer にインポートする

1. WebSphere Studio Application Developer で「J2EE」パースペクティブに移動します。
2. **「ファイル」** > **「インポート」** > **「EJB JAR ファイル (EJB JAR file)」**を選択します。**「次へ」**をクリックします。
3. **WebSphereCommerceServerExtensionsData** という EJB プロジェクトを選択します。
4. **WebSphereCommerceServer** という EAR プロジェクトを選択します。
5. **「次へ」**をクリックして、 JAR ファイルからすべての EJB ファイルを選択します。
6. **「次へ」**をクリックして、従属プロジェクトをすべて選択します。依存関係があるプロジェクトが不確かな場合は、プロジェクトをすべて選択してください。
7. **「終了」**をクリックします。

メソッドを編集して java.rmi.RemoteException エラーが起きないようにする

1. WebSphere Studio Application Developer で「J2EE」パースペクティブに移動します。
2. 「プロジェクト・ナビゲーター (Project Navigator)」ビューで、**WebSphereCommerceServerExtensionData¥ejbModule ¥com.ibm.commerce.sample.objects¥BonusBean.java** を開きます。
3. 次の各メソッドを変更して、これらが java.rmi.RemoteException エラーを throw しないようにします。

   ```
   ejbLoad()
   ejbStore()
   unsetEntityContext()
   ```

   たとえば `public void ejbLoad() throws java.rmi.RemoteException` を `public void ejbLoad()` に変更します。
4. 変更内容を保存します。

アクセス分離レベルを変更する

1. WebSphere Studio Application Developer で「J2EE」パースペクティブに移動します。

2. 「プロジェクト・ナビゲーター (Project Navigator)」ビューで、¥WebSphereCommerceServerExtensionData¥ejbModule¥META-INF の下の ejb-jar.xml を開きます。
3. **「アクセス」**をクリックします。
4. **「分離レベル (Isolation Level)」**の下にある**「コミット済み (Committed)」**ノードを選択して、**「除去」**をクリックします。
5. **「追加」**をクリックして**「反復可能読み取り (Repeatable read)」**を選択し、**「次へ」**をクリックします。
6. **「ボーナス (Bonus)」**を選択して**「次へ」**をクリックします。
7. **「ボーナス (Bonus)」**を拡張表示して、すべてのホーム・メソッド (**「ボーナス (Bonus)」**の下の最初のアイコン) のチェック・ボックスを選択します。
8. **「終了」**をクリックします。
9. **「反復 (Repeat)」**ノードが**「分離レベル (Isolation Level)」**の下にあることを確認します。
10. 変更内容を保存します。

デプロイメント・コードを生成する

コードを変更したら、そのデプロイメント・コードを再生成する必要があります。

デプロイメント・コードを再生成するには、以下のようにします。

1. WebSphere Studio Application Developer で「J2EE 階層 (J2EE Hierarchy)」ビューに移動します。
2. **「EJB モジュール (EJB Modules)」**フォルダーを拡張表示します。
3. **WebSphereCommerceServerExtensionsData**を右マウス・ボタンでクリックして、**「生成」** > **「デプロイメントおよび RMIC コード (Deploy and RMIC Code)」**を選択します。
4. **「すべての EJB (All EJBs)」**を選択して、**「終了」**をクリックします。

カスタマイズされた JSP ファイルをマイグレーションする

前記の「*WebSphere Commerce プログラミング・ガイドとチュートリアル*」ガイドのチュートリアルでは、 confirmation.jsp テンプレートを変更して、オーダー処理ビジネス・ロジックに追加した新規ビジネス・ロジックを表示するようにしました。カスタマイズ JSP ファイルを WebSphere Studio Application Developer にマイグレーションするには、以下のステップで行います。

1. WebSphere Studio Application Developer を開いて、「リソース (Resource)」パースペクティブに切り替えます。
2. **「ファイル」** > **「インポート」** > **「フォルダー」**をクリックして、**「次へ」**をクリックします。
3. ディレクトリー *VAJ_installdir*¥ide¥project_resources¥IBM WebSphere Test Environment¥hosts¥default_host¥default_app¥web¥store_directory をブラウズします。
4. インポートする confirmation.jsp を選択します。

5. そのファイルを入れるフォルダーとして、フォルダー ¥Stores¥Web Content を選択します。

拡張コントローラー・コマンドをマイグレーションする

拡張コントローラー・コマンドをマイグレーションするには、まずプロジェクトを JAR ファイルにエクスポートし、次に Java ファイルとリソース・ファイルを WebSphere Studio Application Developer にインポートした後に、マイグレーション済みコードをコンパイルします。

_WCSamples プロジェクトを JAR ファイルにエクスポートするには、以下のステップで行います。

1. VisualAge for Java の「ワークベンチ (Workbench)」ウィンドウで **「_WCSamples」**プロジェクトを選択して右マウス・ボタンでクリックし、**「エクスポート」**をクリックします。
2. **「JAR ファイル (Jar file)」**ラジオ・ボタンを選択して**「次へ」**をクリックします。
3. JAR ファイルの名前を指定します。
4. Java ファイルをエクスポートするために**「.java」**チェック・ボックスを選択します。
5. 必要に応じて他のフィールドを埋めます。このタスクの実行方法の詳細は、VisualAge for Java のオンライン・ヘルプを参照してください。

Java ファイルとリソース・ファイルを WebSphere Studio Application Developer にインポートするには、以下のステップで行います。

1. WebSphere Studio Application Developer を開いて、「リソース (Resource)」パースペクティブに切り替えます。
2. **「ファイル」** > **「インポート」** > **「ZIP ファイル (Zip file)」**を選択します。**「次へ」**をクリックします。
3. 上記で作成したものに該当する JAR ファイルをブラウズします。
4. JAR に含まれるインポート対象の .java ファイルをすべて選択します。
5. そのファイルを入れるフォルダーとして、フォルダー ¥WebSphereCommerceServerExtensionsLogic¥src を選択します。

Web プロジェクトに含まれるマイグレーション済みコードをコンパイルするには、以下のステップで行います。

1. WebSphere Studio Application Developer を開いて、「J2EE」パースペクティブに切り替えます。
2. 「プロジェクト・ナビゲーター (Project Navigator)」ビューで、**WebSphereCommerceExtensionsLogic** を右マウス・ボタンでクリックします。
3. **「プロパティー」**を選択します。
4. **「Java ビルド・パス (Java Build Path)」**を選択します。
5. **「プロジェクト」**タブを選択します。
6. **「WebSphereCommerceExtensionsData」**を選択します。

7. 「**WebSphereCommerceServerExtensionsLogic**」プロジェクトを選択します。
8. 右マウス・ボタンでクリックして、「**プロジェクトのビルド (Build Project)**」を選択します。

新規ロジックを Express Store サンプル・ストアでテストする

新規ビジネス・ロジックを検証する前に、サーバーを開始しておく必要があります。サーバーを開始するには、「サーバー」ビューから「**WebSphereCommerceServer**」を右マウス・ボタンでクリックして、「**開始**」を選択します。

WebSphere Studio Application Developer で実行する Express Store で新規ビジネス・ロジックを検証するには、以下のようにします。

1. WebSphere Commerce Server インスタンスが WebSphere Studio Application Developer 内で開始されたら、ブラウザーを開いて、使用ストアのホーム・ページの URL を入力します。たとえば次の URL を入力します。

   http://hostname/webapp/wcs/stores/servlet/StoreCatalogDisplay?storeId=*store_Id*&catalogId=*catalog_Id*&langId=-1

   ここで *store_Id* はストアの ID、 *catalog_Id* はストアのカタログの ID です。
2. 商品を選択して購入します。
3. 商品を購入すると、そのオーダーで獲得したボーナス・ポイント数が「オーダーの確認」に表示されます。

第 7 章 チュートリアル B: 拡張/カスタマイズされた Entity Bean およびタスク・コマンドのマイグレーション

以下のチュートリアルでは、拡張 Entity Bean と拡張タスク・コマンドのマイグレーションを行います。このマイグレーションには、タスク・コマンド、拡張 Data Bean、コマンドに関連した Enterprise Bean、コマンドを表示する JSP ファイル、をそれぞれマイグレーションするサブステップの他に、すべてのものが正常にマイグレーションされていることをテストするサブステップがあります。 WebSphere Commerce 5.4 向けの「*WebSphere Commerce プログラマーズ・ガイド*」に記載されているチュートリアルの後半では、既存のタスク・コマンド GetProductContractUnitPriceCmd と、それに対応するインターフェースをカスタマイズする方法を例示しています。ここで行うチュートリアルでは、顧客のボーナス・ポイントの残ポイント数に基づいて割引価格を計算する新規ビジネス・ロジックを作成します。この計算は新規タスク・コマンドによって行われます。このマイグレーション・チュートリアルのモデルとして、新規拡張コマンド GetNewProductContractUnitPriceCmd と、そのインターフェース GetNewProductContractUnitPriceCmdImpl を使用します。

拡張タスク・コマンドをマイグレーションする

拡張タスク・コマンドをマイグレーションするには、まずプロジェクトを JAR ファイルにエクスポートし、次に Java ファイルとリソース・ファイルを WebSphere Studio Application Developer にインポートした後に、マイグレーション済みコードをコンパイルします。

_WCSamples プロジェクトを JAR ファイルにエクスポートするには、以下のステップで行います。

1. VisualAge for Java の「ワークベンチ (Workbench)」ウィンドウで **「_WCSamples」**プロジェクトを選択して右マウス・ボタンでクリックし、**「エクスポート」**をクリックします。
2. **「JAR ファイル (Jar file)」**ラジオ・ボタンを選択して**「次へ」**をクリックします。
3. JAR ファイルの名前を指定します。
4. Java ファイルをエクスポートするために**「.java」**チェック・ボックスを選択します。
5. 必要に応じて他のフィールドを埋めます。このタスクの実行方法の詳細は、VisualAge for Java のオンライン・ヘルプを参照してください。

Java ファイルとリソース・ファイルを WebSphere Studio Application Developer にインポートするには、以下のステップで行います。

1. WebSphere Studio Application Developer を開いて、「リソース (Resource)」パースペクティブに切り替えます。
2. **「ファイル」** > **「インポート」** > **「ZIP ファイル (Zip file)」**を選択します。**「次へ」**をクリックします。

3. 上記で作成したものに該当する JAR ファイルをブラウズします。
4. JAR に含まれるインポート対象の .java ファイルをすべて選択します。
5. そのファイルを入れるフォルダーとして、フォルダー ¥WebSphereCommerceServerExtensionsLogic¥src を選択します。

Web プロジェクトに含まれるマイグレーション済みコードをコンパイルするには、以下のステップで行います。

1. WebSphere Studio Application Developer を開いて、「J2EE」パースペクティブに切り替えます。
2. 「プロジェクト・ナビゲーター (Project Navigator)」ビューで、**WebSphereCommerceServerExtensionsLogic¥src¥com¥ibm¥commerce ¥sample¥commands¥GetNewBaseUnitPriceCmdImpl.java** を開きます。
3. `new BigDecimal(dblBonusPrice)` を検索して、これを `new java.math.BigDecimal(dblBonusPrice)` に変更します。
4. ファイルを保管します。
5. **「WebSphereCommerceServerExtensionsLogic」**プロジェクトを選択します。
6. 右マウス・ボタンでクリックして、**「プロジェクトのビルド (Build Project)」**を選択します。

拡張 Data Bean をマイグレーションする

このチュートリアルの次のステップは、拡張 Data Bean の NewProductDataBean.java のマイグレーションです。拡張 Data Bean をマイグレーションするには、まずプロジェクトを JAR ファイルにエクスポートし、次に Java ファイルとリソース・ファイルを WebSphere Studio Application Developer にインポートした後に、マイグレーション済みコードをコンパイルします。

`_WCSamples` プロジェクトを JAR ファイルにエクスポートするには、以下のステップで行います。

1. VisualAge for Java の「ワークベンチ (Workbench)」ウィンドウで**「_WCSamples」**プロジェクトを選択して右マウス・ボタンでクリックし、**「エクスポート」**をクリックします。
2. **「JAR ファイル (Jar file)」**ラジオ・ボタンを選択して**「次へ」**をクリックします。
3. JAR ファイルの名前を指定します。
4. Java ファイルをエクスポートするために**「.java」**チェック・ボックスを選択します。
5. 必要に応じて他のフィールドを埋めます。このタスクの実行方法の詳細は、VisualAge for Java のオンライン・ヘルプを参照してください。

Java ファイルとリソース・ファイルを WebSphere Studio Application Developer にインポートするには、以下のステップで行います。

1. WebSphere Studio Application Developer を開いて、「リソース (Resource)」パースペクティブに切り替えます。

2. **「ファイル」** > **「インポート」** > **「ZIP ファイル (Zip file)」**を選択します。**「次へ」**をクリックします。
3. 上記で作成したものに該当する JAR ファイルをブラウズします。
4. JAR に含まれるインポート対象の .java ファイルをすべて選択します。
5. そのファイルを入れるフォルダーとして、フォルダー ¥WebSphereCommerceServerExtensionsLogic¥src を選択します。

Web プロジェクトに含まれるマイグレーション済みコードをコンパイルするには、以下のステップで行います。

1. **「WebSphereCommerceServerExtensionsLogic」**プロジェクトを選択します。
2. 右マウス・ボタンでクリックして、**「プロジェクトのビルド (Build Project)」**を選択します。

Enterprise Bean をマイグレーションする

前記の「*WebSphere Commerce プログラミング・ガイドとチュートリアル*」ガイドのチュートリアルでは、アプリケーションには BONUS テーブルの BONUSPOINT 列が USERS テーブルの実際の列に見えるように User Enterprise Bean をカスタマイズしています。 USERS テーブルに新規レコードが作成されると、それに対応するレコードが BONUS テーブルに自動的に挿入されます。このテーブル結合を作成するためには、新しいコンテナー管理パーシスタンス (CMP) フィールドを User Entity Bean に追加しなければなりません。この CMP フィールドは、 Secondary Table Map フィーチャーによって、 BONUS テーブルの BONUSPOINT 列にマップされます。この Enterprise Bean を WebSphere Studio Application Developer にマイグレーションする場合、以下で説明するように、CMP フィールドを User Entity Bean にマイグレーションする、 USER データベースのスキーマおよびテーブル・マッピングを BONUS テーブルを含むように更新する、 User Entity Bean 用のデプロイメント・コードと Access Bean を生成する、というサブステップがあります。

bonusPoint フィールドを User エンティティーにマイグレーションする

このセクションでは、Enterprise Bean のツールを使用して、 CMP フィールドを Entity Bean にマイグレーションします。このフィールドは bonusPoint というフィールドで、最終的に BONUS テーブルの BONUSPOINT 列にマップされます。この CMP フィールドを User Entity Bean に再作成するには、以下のようにします。

1. WebSphere Studio Application Developer で「J2EE」パースペクティブに移動します。
2. J2EE 階層で、**「EJB モジュール (EJB Modules)」**を拡張表示します。
3. **「Member-MemberManagementData EJB モジュール (Member-MemberManagementData EJB module)」** > **「Entity Beans」** > **「メンバー」**を拡張表示します。
4. **「ユーザー」**を右マウス・ボタンでクリックして、**「開く (Open With)」** > **「Deployment Descriptor Editor」**を選択します。
5. 「Bean」ページを開いて**「User EJB」**を選択します。

6. 右にある**「追加」**をクリックすると CMP 属性ウィザードが開きます。そこに以下の情報を指定します。
   - **名前**: bonusPoint
   - **タイプ**: int
   - **getter および setter メソッドによるアクセス (Access with getter and setter methods)**: 使用可能
   - **リモート・インターフェースでの getter および setter メソッドの使用 (Promote getter and setter methods to remote interface)**: 使用不可
7. **「OK」** をクリックします。
8. EJB Deployment Descriptor の変更を保管します (キーボードの**「Ctrl」キー + 「S」キー**を押します)。

bonusPoint フィールドが「属性」ウィンドウに表示されます。警告が表示されますが、 Entity Bean のコードを再生成する段階で警告はなくなります (38 ページの『デプロイメント・コードを生成する』を参照)。

スキーマおよびテーブル・マッピングを更新して BONUS テーブルを組み込む

ここでは、User スキーマを BONUS テーブルを含むように更新し、新規テーブルの外部キー関係を作成して、 User Entity Bean のフィールドと BONUS テーブルの列とのテーブル・マップを作成します。

テーブル・スキーマを作成するには、以下のようにします。

1. WebSphere Studio Application Developer で「J2EE」パースペクティブに移動します。
2. 「J2EE 階層 (J2EE Hierarchy)」ビューで**「データベース」**を拡張表示します。
3. **「User データベース・スキーマ (User database schema)」**を拡張表示し、次に**「NULLID」**を拡張表示します。
4. **「テーブル」**を右マウス・ボタンでクリックして、**「新規」 > 「新規テーブルの定義 (New Table Definition)」**を選択します。テーブル定義エディターが開きます。
5. テーブル名に BONUS と入力して、**「次へ」**をクリックします。
6. 「テーブル列 (Table Columns)」ページで、次の 2 つの列を作成します。
   - memberid - BIGINT (キー列)
   - bonusPoint - INTEGER (ヌル可能)
7. **「次へ」**を 2 度クリックして「外部キー (Foreign Keys)」ページに進みます。
8. **「追加 (Add Another)」**をクリックし、以下の情報を指定して外部キーを新規作成します。
   - **外部キー名 (Foreign key name)**: F_User_Bonus
   - **削除 (On Delete)**: CASCADE
   - **ターゲット・テーブル (Target table)**: NULLID.USERS
9. **「ソース列 (Source Columns)」**で**「member_Id」**を選択して、**「>」**ボタンをクリックします。

10. **「終了」**をクリックすると新規テーブルが作成されます。

BONUS テーブル・マップを作成するには、以下のようにします。

1. WebSphere Studio Application Developer で「J2EE」パースペクティブに移動します。
2. 「J2EE 階層 (J2EE Hierarchy)」ビューで**「EJB モジュール (EJB Modules)」**を拡張表示します。
3. **「Member-MemberManagementData EJB モジュール (Member-MemberManagementData EJB module)」**を右マウス・ボタンでクリックして、**「開く (Open With)」** > **「マッピング・エディター (Mapping Editor)」**を選択します。
4. 左上のペイン (Enterprise Bean ペイン) で、**「メンバー」**を拡張表示して**「User Bean」**を選択します。
5. **「Ctrl」**キーを押しながら、右上隅ペインの**「MEMBER」**、**「USERS」**、および**「BONUS」**テーブルを選択します。これらの 3 つのテーブルとも強調表示されます。
6. プルダウン・メニューから、**「マッピング (Mapping)」** > **「マッピングの作成 (Create Mapping)」**を選択します。
7. プルダウン・メニューから、**「マッピング (Mapping)」** > **「名前による突き合わせ (Match by Name)」**を選択します。
8. 「概要 (Overview)」ウィンドウで、**「User Bean」**を選択します。
9. 「プロパティー」ウィンドウに移動して、**「Bean 対テーブルのストラテジー (Bean to Table Strategy)」**を拡張表示します。
10. User の**「Descriptor 値 (Descriptor Value)」**を U に変更します。
11. EJB マッピングを保管します (キーボードの**「Ctrl」キー + 「S」キー**を押します)。

この段階の User Bean には、インプリメントされていない抽象クラスがあるというエラーが含まれています。このエラーは、デプロイメント・コードが生成される段階でなくなります (『デプロイメント・コードを生成する』を参照)。

デプロイメント・コードを生成する

User Entity Bean 用にコードを変更したら、次はそのデプロイメント・コードを再生成する必要があります。

デプロイメント・コードを再生成するには、以下のようにします。

1. WebSphere Studio Application Developer で「J2EE」パースペクティブに移動します。
2. 「J2EE 階層 (J2EE Hierarchy)」ビューで**「EJB モジュール (EJB Modules)」**を拡張表示します。
3. **「Member-MemberManagementData EJB モジュール (Member-MemberManagementData EJB module)」**を右マウス・ボタンでクリックして、**「生成」** > **「デプロイメントおよび RMIC コード (Deploy and RMIC Code)」**を選択します。
4. **「すべての EJB (All EJBs)」**を選択して、**「終了」**をクリックします。

カスタマイズされた JSP ファイルをマイグレーションする

前記の「*WebSphere Commerce プログラミング・ガイドとチュートリアル*」ガイドのチュートリアルでは、 ProductDisplay.jsp テンプレートを変更して、オーダー処理ビジネス・ロジックに追加したカスタマイズ価格を顧客が見ることができるようにしました。カスタマイズ JSP ファイルを WebSphere Studio Application Developer にマイグレーションするには、以下のステップで行います。

1. WebSphere Studio Application Developer を開いて、「リソース (Resource)」パースペクティブに切り替えます。
2. **「ファイル」** > **「インポート」** > **「フォルダー」**をクリックして、**「次へ」**をクリックします。
3. ディレクトリー *VAJ_installdir*¥ide¥project_resources¥IBM WebSphere Test Environment¥hosts¥default_host¥default_app¥web¥store_directory をブラウズします。
4. インポートする ProductDisplay.jsp を選択します。
5. そのファイルを入れるフォルダーとして、フォルダー ¥Stores¥Web Content を選択します。

新規ロジックを Express Store サンプル・ストアでテストする

新規ビジネス・ロジックを検証する前に、サーバーを開始しておく必要があります。サーバーを開始するには、「サーバー」ビューから**「WebSphereCommerceServer」**を右マウス・ボタンでクリックして、**「開始」**を選択します。

WebSphere Studio Application Developer で実行する Express Store で新規ビジネス・ロジックを検証するには、以下のようにします。

1. WebSphere Commerce Server インスタンスが WebSphere Studio Application Developer 内で開始されたら、ブラウザーを開いて、使用ストアのホーム・ページの URL を入力します。たとえば次の URL を入力します:
   http://*host_name*/webapp/wcs/stores/servlet/StoreCatalogDisplay?storeId=*store_Id*&catalogId=*catalog_Id*&langId=-1
   各パラメーターの意味は次のとおりです。

   host_name
   ご使用の WebSphere Commerce Studio マシンの完全修飾ホスト名。

   store_Id
   ストアの ID。

   catalog_Id
   ストアのカタログの ID。
2. **「サービス」**ヘッダーの下の**「登録」**リンクをクリックしてから、**「新規の顧客 (New Customer)」**ヘッダーの下の**「登録」**をクリックします。
3. E メール・アドレス `wctester@wc` とパスワード `wctester1` を使用して新規の顧客を登録します。
4. 他のフィールドにテスト値を指定し、**「送信」**をクリックします。ブラウザーは開いたままにしておきます。

5. DB2 Command Center を開いて、以下のようにします。
   a. **「インタラクティブ (Interactive)」**タブをクリックします。
   b. **「コマンド」**フィールドで、以下のようにします。
      1) 次を入力します。

         ```
         connect to your_database_name
         ```

         your_database_name は、WebSphere Commerce の使用データベースの名前です。
      2) **「実行」**アイコンをクリックします。
      3) 次を入力します。

         ```
         select users_id from userreg where logonid = 'wctester@wc'
         ```

      4) **「実行」**アイコンをクリックします。
   c. **「参照結果 (Query Results)」**タブに、前のステップで登録した顧客のエントリーが表示されます。その顧客の USERS_ID の値をメモしておきます。
   d. 新規登録した顧客のボーナス・ポイントの残ポイント数を更新します。
   e. **「インタラクティブ (Interactive)」**タブをクリックして、**「コマンド」**フィールドに次を入力します。

      ```
      update BONUS set BONUSPOINT = 1000 where MEMBERID = users_id
      ```

      users_id は、ステップ 5c で表示された値です。
   f. **「実行」**アイコンをクリックします。
6. ブラウザーで**「メンズ」**リンクをクリックして、ストアのメンズ・ファッションのセクションを表示します。
7. お買い得のリンクをクリックして商品ページを表示します。このページに、正規価格と、その顧客のボーナス・ポイントの残ポイント数に基づいた割引価格が表示されます。

第 5 部 付録

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものであり、本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

本文書中において IBM プログラムについて言及している場合、該当 IBM プログラムのみが使用可能であることを意味するものではありません。 IBM 製品、プログラムまたはサービスに代えて、IBM の知的所有権を侵害することのない機能的に同等のプログラムまたは製品を使用することができます。ただし、IBM によって明示的に指定されたものを除き、他社の製品と組み合わせた場合の動作の評価と検証はお客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-0032
東京都港区六本木 3-2-31
IBM World Trade Asia Corporation
Licensing

以下の保証は、国または地域の法律に沿わない場合は、適用されません。

IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム（本プログラムを含む）との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Canada Ltd.
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
Canada

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書はプランニング目的としてのみ記述されています。記述内容は製品が使用可能になる前に変更になる場合があります。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

この製品で使用されているクレジット・カードのイメージ、商標、商号は、そのクレジット・カードを利用して支払うことを、それら商標等の所有者によって許可された人のみが、使用することができます。

商標

以下は、IBM Corporation の商標です。

DB2 IBM VisualAge WebSphere

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

Printed in Japan

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12