IBM® WebSphere® Commerce

ストア開発者ガイド

バージョン 5.4

IBM® WebSphere® Commerce

ストア開発者ガイド

バージョン 5.4

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。

本書は以下の製品に適用されます。

- IBM WebSphere Commerce Business Edition for Windows NT and Windows 2000 バージョン 5.4
- IBM WebSphere Commerce Business Edition for AIX バージョン 5.4
- IBM WebSphere Commerce Business Edition for Solaris バージョン 5.4
- IBM WebSphere Commerce Studio, Business Developer Edition for Windows NT and Windows 2000 バージョン 5.4
- IBM WebSphere Commerce Professional Edition for Windows NT and Windows 2000 バージョン 5.4
- IBM WebSphere Commerce Professional Edition for AIX バージョン 5.4
- IBM WebSphere Commerce Professional Edition for Solaris バージョン 5.4
- IBM WebSphere Commerce Studio, Professional Developer Edition for Windows NT and Windows 2000 バージョン 5.4

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)

原 典 :	IBM WebSphere Commerce Store Developer's Guide Version 5.4
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 2002.4

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

はじめに

IBM WebSphere Commerce ストア開発者ガイド には、 WebSphere Commerce ストアのアーキテクチャーおよびストア開発プロセスについての情報が載せられています。特に、以下のトピックについて詳細に説明します。

- さまざまなストア開発方法
- ストア・アーカイブ
- さまざまなストア開発ツール
- ストアフロントの開発
- ストア・データの開発
- ストア・データのアーキテクチャー
- ストア・データの情報モデル
- ストアへのアクセス・コントロールの追加
- ストアのパッケージ化
- ストアの発行
- ストアへの WebSphere Commerce フィーチャーの追加

本書の表記規則

本書では、以下のような強調表示の規則を使用しています。

太文字は、コマンドまたはグラフィカル・ユーザー・インターフェース (GUI) のコントロール (フィールド、ボタン、またはメニュー選択項目の名前など) を表します。

`モノスペース (Monospace)` は、表示どおりに入力するテキストの例、およびディレクトリーのパスを表します。

イタリック は、強調のため、および実際に使用する値に置き換える変数を表すために使用されます。

このアイコンはヒントを表します。これは、タスクを完了するのに役立つ可能性のある追加情報です。

NT は、Windows NT® に固有の情報を示します。

2000 は、 Windows® 2000 に固有の情報を示します。

AIX は、 AIX® に固有の情報を示します。

Solaris は、Solaris オペレーティング環境に固有の情報を示します。

400 は、 IBM e (e-business ロゴ) server iSeries 400® (旧 AS/400®) に固有の情報を示します。

Linux は、Linux に固有の情報を示します。

Business は、 WebSphere Commerce Business Edition に固有の情報を示します。

Professional は、WebSphere Commerce Professional Edition に固有の情報を示します。

最新情報の入手先

本書は、将来、更新される可能性があります。以下の WebSphere Commerce Web サイトで更新情報をチェックしてください。

Business

http://www.ibm.com/software/webservers/commerce/wc_be/lit-tech-general.html

Professional http://www.ibm.com/software/webservers/commerce/wc_pe/lit-tech-general.html

更新では、新しい情報が含まれていることがあります。

第 1 部 概要

第 1 章 ストア・アーキテクチャーの概要

この章では、WebSphere Commerce Server のストア・アーキテクチャーについて概要を説明します。

オンライン・ストアとは

オンライン・ストアとは、インターネットのテクノロジーを使用して、商品やサービスを販売または交換するストアのことです。オンライン・ストアは、商品を表示し、顧客がそれを購入するための一連の Web ページで構成されています。顧客は、ホーム・ページからストアへと案内され、そこで製品やサービスが紹介されます。オンライン・カタログ・ページでは商品が分類されており、商品についての詳細情報が記載されている商品ページに顧客を案内します。企業消費者間取引ストアでは、「ショッピング・カート」のページが物理的なショッピング・カートと同じ役割を果たします。購入したい製品をそこに追加し、「チェックアウト」ページでそのための支払いをします。企業間取引サイトの場合は、特定のページによりオーダーと見積依頼 (RFQ) が送信されます。

ストア・ページは、JavaServer Pages (JSP) テクノロジーを使って作成されます。各ページには、静的コンテンツ、入力データや洗練されたデータ表示のためのクライアント・サイド JavaScript、 WebSphere Commerce Server コマンドおよびその他のビューを呼び出すための URL を含む HTML、そして動的コンテンツを生成するための JSP タグや Java™ コードが含まれています。 JavaServer Pages (JSP) ファイルでは、 WebSphere Commerce Studio および WebSphere Commerce に含まれる一連のコマース・データ bean を利用できます。それを使うことにより、製品の価格や製品の属性など、データベースからの情報にアクセスできます。

ストアには、さらに機能的なストアを作成するために必要なデータベース資産も含まれています。たとえば、機能的なストアには、カタログ、税、配送、および通貨に関するデータが含まれていなければなりません。

ストアの構成

オンライン・ストアは、次の資産で構成されます。

- ストアフロント

  ストアの外部を構成する部分、つまり顧客に対して表示される部分は、ストアフロントと呼ばれます。ストアフロント資産は、 HTML ページ、JSP ファイル、スタイルシート、イメージ、グラフィックス、およびその他のマルチメディア・ファイル・タイプなどの Web 資産で構成されています。

  このマニュアルでは、ストア・ページを構築するために必要な JSP ファイルの作成に関係したさまざまな概念や作業について説明します。詳細については、 17 ページの『第 3 章 ストアフロントの開発』を参照してください。

- バック・オフィス

ストアのうち、顧客からは見えない部分、コマンド、カスタマイズされたコード、および顧客がストアフロントで製品を購入するためのビジネス・ロジックのインプリメンテーションは、バック・オフィスと呼ばれます。

ビジネス・ロジックまたはカスタマイズ済みコードの作成については、 *IBM WebSphere Commerce プログラマーズ・ガイド* を参照してください。

- ストア・データ

  ストアを構成するデータ資産。ストアが正しく動作するためには、顧客のすべてのアクティビティーをサポートするデータがストアに配置されていなければなりません。たとえば、顧客が商品を購入するためには、販売商品のカタログ、オーダー処理のためのプロセス、要求を実行する在庫機能、および配送プロセスが必要です。さらに、支払を処理し、集金するための手段も必要です。

  ストア・データの作成に関係するさまざまな概念や作業については、 39 ページの『第 4 部 ストア・データの開発』で説明します。

ストアのアーキテクチャー

WebSphere Commerce ストアのアーキテクチャーは、以下のコンポーネントで構成されます。

- WebSphere Commerce Server
- WebSphere Commerce Server インスタンス
- ストア構成

WebSphere Commerce Server

WebSphere Commerce Server は、e-commerce ソリューションのうち、ストアおよびコマースに関連したさまざまな機能を処理するサーバーです。ストアフロント資産およびバック・オフィス・ビジネス・ロジックは、 WebSphere Commerce Server 内の Web アプリケーションの中に存在しています。 WebSphere Commerce には、デフォルトで使用できる Web アプリケーション (`WCS Stores`) が用意されていますが、独自に作成することもできます。

1 つの Web アプリケーションには、1 つのストアのための資産が含まれる場合と、複数のストアのための資産が含まれる場合とがあります。 1 つの Web アプリケーションに複数のストアフロントおよびバック・オフィスが含まれる場合、各ストアの資産はストア・ディレクトリー (storedir) によって分離されます。

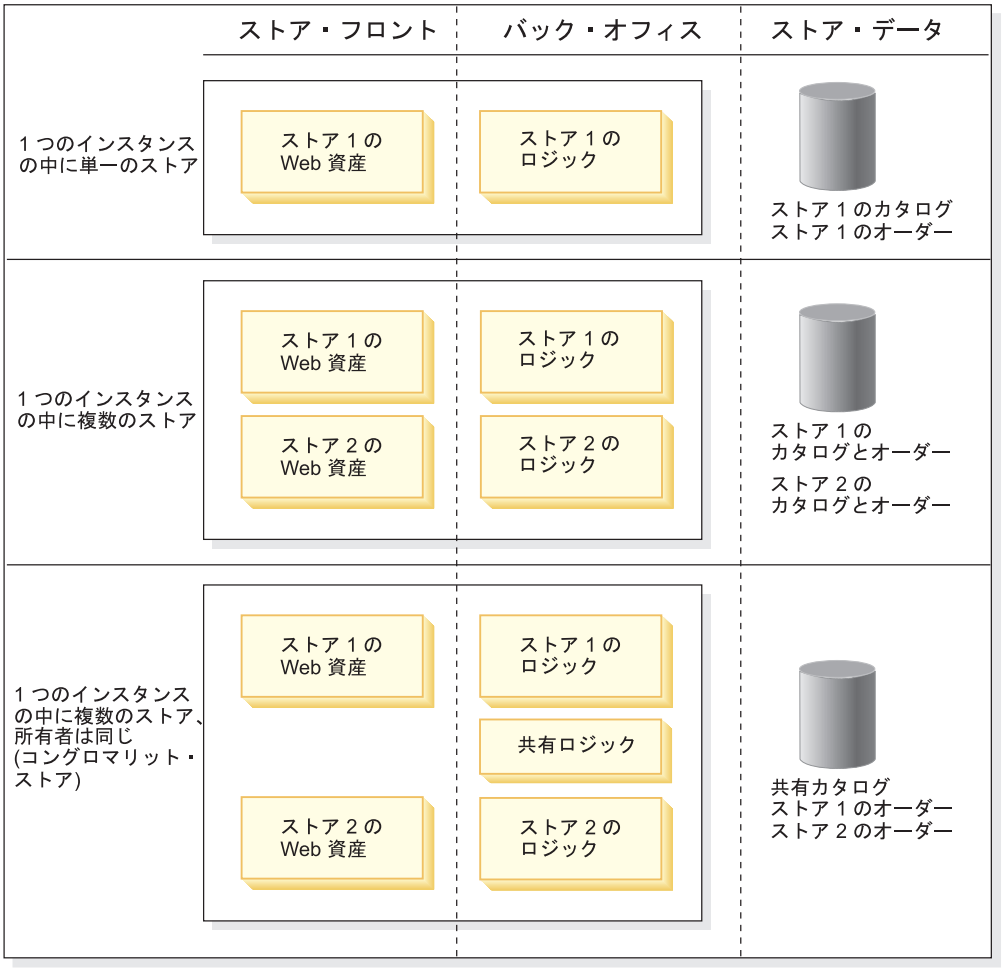
WebSphere Commerce Server インスタンス

WebSphere Commerce Server インスタンスは、関連するデータベースを伴う WebSphere Application Server アプリケーションです。 1 つのインスタンスで、複数のストアをサポートできます。 1 つのインスタンス中のすべてのストアは同じデータベースを共用します。さらにはカタログ、フルフィルメント、または領収証などのある種のデータも共用できます。また、1 つのインスタンスに含まれるすべてのストアは、同じ EJB コンテナーを共用します。

ストア構成

WebSphere Commerce では、複数のストア構成がサポートされています。つまり、WebSphere Commerce を使うと、 1 つのインスタンスに含まれる単一のストアを作成したり、 1 つのインスタンスの中に複数のストアを作成し、それぞれが別個のストアフロント、バック・オフィス、およびストア・データを使用するようにしたり

できます。あるいは、1 つのインスタンスの中に、それぞれ別個のストアフロントを持ち、バック・オフィスとカタログを共用する複数のストアを作成することもできます。次の図は、可能性のあるいくつかのストア構成を示しています。



注: 各ストアには、それぞれ独自の ID があります。 WebSphere Commerce のライセンスでは、作成できるストアの数について特定の限界が設定されています。限界を超える分については、購入することができます。詳しくは、ライセンス証書を参照してください。

第 2 章 ストア開発

この章では、WebSphere Commerce におけるストア開発プロセスの概要を説明します。

さまざまなストア開発方法

WebSphere Commerce には、ストアを開発するためのいくつかのオプションが用意されています。

- サンプルに基づいてストアを作成する
- 新しいストア資産を開発することによってストアを作成する
- サンプル・ストアと新しいストア資産の組み合わせを使用してストアを作成する

サンプルに基づいてストアを作成する

WebSphere Commerce でオンライン・ストアを最も短時間に、また最も簡単に作成する方法は、 WebSphere Commerce に含まれているサンプル・ストアの 1 つをコピーし、それを実際の必要に合わせてカスタマイズする方法です。

サンプル・ストア

WebSphere Commerce には、独自のストアを作成するための基礎として活用できる、十分な機能を備えたサンプル・オンライン・ストアがいくつか用意されています。それらのサンプルには、企業消費者間取引ストアと企業間取引ストアの両方が含まれています。それらには、代表的な電子商取引サイトで現在一般的に使用されている機能のほとんどがインプリメントされており、必要なストア資産がすべて含まれています。 WebSphere Commerce に含まれているオンライン・ストアについては、 WebSphere Commerce のオンライン・ヘルプを参照してください。

なぜサンプル・ストアからスタートするのか?: WebSphere Commerce でオンライン・ストアをまったく新規に作成することも可能ですが、作成するストアの基礎としてサンプル・ストアの 1 つをコピーして使用すれば、実働ストアをさらに短時間で作成できます。

WebSphere Commerce では、実働ストアを作成するために特定のデータを WebSphere Commerce Server データベースにロードし、そのデータをスキーマによって決定されるオーダー中にロードする必要があります。サンプル・ストアには WebSphere Commerce Server データベースで必要なオーダーや構造のデータがすべて含まれているため、独自に作成するストアの基礎としてその 1 つを使用するなら、作成作業の初期に必要な時間を大幅に節約できます。

サンプル・ストアのコピーを作成してから、それをストアのニーズに合わせていくらか編集することができます。たとえば、WebSphere Commerce で利用できるツールを使ってデータを編集したり、 WebSphere Commerce Studio を使ってストア・ページのルック・アンド・フィールを変更したりするだけで十分かもしれません。あるいはデータに対して広範囲に渡る変更を加えるために XML ファイルまたはデータベースを直接編集したり、ストアのフローとフィーチャーを変更するためにスト

ア・ページを再作成したりする必要がある場合もあります。ストアの編集の詳細については、 WebSphere Commerce オンライン・ヘルプのトピック『ストア・データベース資産の変更』 を参照してください。

WebSphere Commerce には参照ストアも用意されています。これは、重要なフィーチャーのサンプル・コードとして使用するために設計されています。参照ストアは、オンライン・ストアの選択したフィーチャーの全機能のコードを含むオンライン・ストアです。たとえば、クーポンなどがその一例です。参照ストアは以下の URL から入手できます。

Business

http://www.ibm.com/software/webservers/commerce/wc_be/downloads.html

Professional http://www.ibm.com/software/webservers/commerce/wc_pe/downloads.html

サンプルを基にしたストアの作成の詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

新規ストア資産の開発によるストアの作成

誰もがサンプル・ストアを基にしてストアを作成することを望むとは限りません。たとえば、ストア・ページのフローが、用意されているサンプルのいずれのものとも大きく異なる場合、または WebSphere Commerce Server データベース・スキーマを大幅にカスタマイズする計画の場合は、独自のストアフロント、バック・オフィス、およびストア・データ資産を開発して、実際のストアを作成することもできます。 WebSphere Commerce に用意されているツールのリストについては、 10 ページの『ストア開発ツール』を参照してください。

サンプル・ストア資産と新規ストア資産の組み合わせを使用したストアの作成

ストア開発に、サンプル・ストア資産と、開発する新規ストア資産の組み合わせを使用すると、最も有効なストア開発の方法になる場合があります。たとえば、あるサンプル・ストアの中のデータベース資産のいくつかが実際のストアのニーズによく似ているが、実際のストアのページのフローについてはそうではない場合、そのストアからコピーしたデータベース資産をカスタマイズし、しかもまったく新規の Web 資産を開発することができます。

ストア・アーカイブ

WebSphere Commerce に組み込まれているサンプル・ストアは、ストア・アーカイブの形式で提供されます。ストア・アーカイブ・ファイル (.sar) は、ストアの作成に必要なすべての資産の入った圧縮アーカイブ・ファイル (ZIP ファイルなど) です。これは、第一に簡単にコピーできる形式でストアをパッケージ化し、送付するための媒体として使用され、また、新規ストア作成の基礎として使用されます。ストア・アーカイブを WebSphere Commerce Server に発行するだけで、表示、ブラウズ、および買い物を行える機能ストアを作成できます。

一般に、ストア・アーカイブは以下のファイルで構成されます。

- Web 資産: HTML ファイル、JSP ファイル、イメージ、グラフィックス、および組み込みファイルなど、ストア・ページの作成に使用するファイル。 Web 資産は、ストア・アーカイブ内で圧縮ファイルとして分類されます。
- プロパティー・リソース・バンドル (オプション): ストア・ページのテキストが入っています。ストアが複数の言語をサポートする場合、リソース・バンドルには複数のバンドル (言語ごとに 1 つ) があります。
- ストア・データ資産: データベースにロードされるデータ。ストア・データ資産には、キャンペーン、カタログ・エントリー、通貨、フルフィルメント情報、価格設定、配送、ストア、税情報などのデータが含まれます。ストア・データ資産の詳細なリストについては、 39 ページの『第 4 部 ストア・データの開発』を参照してください。

  WebSphere Commerce で提供されるサンプル・ストア・アーカイブにあるストア・データベース資産は、ローダー・パッケージにとって妥当な、整形式の XML ファイルです。ストア・アーカイブ XML ファイルは移植可能であることが意図されており、データベースの特定のインスタンスに固有な、生成された 1 次鍵を含めるべきではありません。その代わりに、ストア発行時に ID リゾルバーによって解決される内部別名が使用されます。これらの規則を使用すると、サンプル・ストア・アーカイブを何度もコピーして発行することができます。詳細については、 313 ページの『付録 B. データの作成』を参照してください。

  ローダー・パッケージについて詳しくは、 225 ページの『第 7 部 ストアの発行』を参照してください。
- 支払資産: IBM Payment Manager の構成情報。
- ディスクリプター: XML ファイル `sarinfo.xml`。 Web 資産圧縮アーカイブ・ファイル、リソース・バンドル、およびストア・データベース資産 XML ファイルの名前を含むストア・アーカイブについて記述します。 `sarinfo.xml` ファイルには、組み込みファイルと整合性検査ファイルの名前、および発行処理中に必要なアーカイブ・ファイルについての情報が含まれます。ストア・アーカイブで必須のファイルは `sarinfo.xml` だけです。

注: Business サンプル・ストア・アーカイブ ToolTech および NewFashion のアーカイブには、以下のファイルも含まれています。

- tools_properties.zip
- tools_xml.zip
- runtime_xml.zip

これらのファイルは、ストアを構成するためにストア・サービスによって使用されます。これらのファイルは、経したり削除したり、他のストアにコピーしたりしないでください。

サンプル・ストア・アーカイブ

サンプル・ストア・アーカイブ・ファイル (.sar) は、新しいストア作成の基礎としてコピーして使用するためのストア・アーカイブです。サンプル・ストア・アーカイブには、それを何度もコピーして発行することができるようにするための、いくつかの規則が含まれています。それらの規則には、以下のものが含まれます。

- 生成された 1 次鍵または外部キーへの参照がないこと。サンプル・ストア・アーカイブには、データベースの特定のインスタンスに対して生成された固有の 1 次

鍵は含まれていません。その代わりに、ストア発行時に ID リゾルバーによって解決される内部別名が使用されます。詳細については、 313 ページの『付録 B. データの作成』を参照してください。

WebSphere Commerce に付属のサンプル・ストアは、サンプル・ストア・アーカイブです。それらのストアは、ストア・サービスの「ストア・アーカイブの作成」ページの**「Samples (サンプル)」**リストから利用できます。

どのような場合にストア・アーカイブを使用するかを決める

ストア・アーカイブは、ストアをパッケージ化して送付するための手段として設計されています。ストアの使用目的が、これをサンプルとして他人に送付すること、別のサーバーまたは別のプラットフォームにデプロイすること、あるいは他のストアを作成する基盤として使用することであるなら、これをストア・アーカイブの形でバンドルすることを考慮してください。

自分のストアとサンプル・ストアの 1 つとの対応が緊密で、アーカイブに多くの変更を加える必要がない場合にも、ストア・アーカイブを使用できます。

ストア・アーカイブを使用することにして、サンプル・ストアを使ってストアを作成した場合、ストアはおのずとストア・アーカイブ形式になります。その後、ストアの開発中に加える変更をすべてストア・アーカイブに反映するようにするなら、新しいストアをストア・アーカイブ形式で保守することができます。

他の方法を使って作成したストアもストア・アーカイブとしてパッケージ化できます。ストア・アーカイブの作成の詳細については、 215 ページの『第 6 部 ストアのパッケージ化』を参照してください。

ストア・アーカイブを使用したくない場合

作成するストア・インスタンスが 1 つの場合、または既存の WebSphere Commerce スキーマに対して広範囲にわたる変更を加える場合は、ストア・アーカイブを使用しないようにすることができます。 WebSphere Commerce スキーマに対する変更は、ストア・アーカイブおよび WebSphere Commerce ストア開発ツールでは直接はサポートされていません。広範囲にわたるスキーマの変更を行った後でストア・アーカイブを保守する場合は、 IBM 担当員に連絡して詳細な情報を入手してください。

ストア開発ツール

WebSphere Commerce には、ストアの開発に役立つ様々なツールが用意されています。どのツールを使用するかは、実際のストアを開発してパッケージ化する際の選択によって異なります。

ストアフロント開発用のツール

ストアフロント資産の開発には、サンプル・ストア・ページをカスタマイズすること、そのページを独自の既存のページに置き換えること、新規ページを作成すること、あるいはこれらの 3 つすべてを組み合わせて実行することが含まれるかもしれません。

WebSphere Commerce には、ストアフロント資産を作成または編集するための以下のツールが用意されています。

- WebSphere Commerce Studio: Commerce Studio には、ストアフロント資産の作成と編集に必要なツールが含まれています。ストアフロント資産には、HTML、グラフィックス、マルチメディア、および JavaServer Pages (JSP) の各ファイルが含まれます。 Page Designer (Commerce Studio に付属) を使用すると、アニメーション・イメージに加えて、HTML または JSP を作成できるようになります。また、自分で選んだ別の Web 開発ツールを使用するよう WebSphere Commerce Studio を構成することもできます。独自のツールを WebSphere Commerce Studio に登録する方法の詳細については、WebSphere Studio オンライン・ヘルプを参照してください。

  ストア・アーカイブ形式でストアを処理する計画の場合、WebSphere Commerce Studio を使用するなら、ストア・アーカイブの構造をそのままの状態に保ちつつ、ストア・アーカイブから Studio プロジェクトへ Web 資産をインポートできます。 Studio ツールを使用して JSP ファイル、HTML ファイル、およびイメージに変更を加え、それからファイルを WebSphere Commerce Server 上のストア・アーカイブに再びエクスポートし、 Web 資産を再発行することができます。

  ストア・アーカイブを保守しない場合は、 WebSphere Commerce Studio を使用して実働ストアに直接ファイルを発行することができます。
- ストア・サービス: ストア・サービスの「Web 資産」ダイアログを使用して、ストア・アーカイブにある Web 資産の圧縮アーカイブ・ファイルを別のセットの Web 資産に置き換えたり、既存の Web 資産を任意の場所へダウンロードすることができます。ストア・アーカイブ形式でストアを処理する場合は、変更した資産を「Web 資産」ダイアログを使ってストア・アーカイブに戻すことができます。「Configure stores (ストアの構成)」ページを使うと、発行済みストアにおいて JSP ファイルのさまざまなフィーチャーを有効にしたり無効にしたりできます。現在のところストア・サービスでサポートされているのは、コラボレーション構成フィーチャー (コラボレーション・ワークスペースおよびカスタマー・ケア) だけです。それらのフィーチャーは、 Business ToolTech および NewFashion のサンプル・ストアに基づくストアでの構成に関してのみ利用できます。

WebSphere Commerce Studio およびストア・サービスのツールを使用してストアフロント資産を作成および編集する方法の詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。 WebSphere Commerce でストアフロントを作成する方法の詳細については、 15 ページの『第 2 部 ストアフロントの開発』を参照してください。

ストア・データ開発用のツール

ストアのデータベース資産を開発および編集するための選択肢がいくつかあります。

- ストア・サービス

  ストア・サービスは、ブラウザー・ベースのツール・セットで、ストア・アーカイブに対して使用します。ストア・サービスを使用すると、WebSphere Commerce に備わっているサンプルに基づいて、ストア・アーカイブを迅速に作成することができます。ストア・アーカイブを作成すると、ストア・サービスによって、以下のタスクを実行できます。

– ストア・アーカイブを発行して、機能ストアを作成する。
– 「税」ノートブックを使用して、税設定を変更する。
– 「配送」ノートブックを使用して、配送設定を変更する。
– 「ストア・プロファイル」ノートブックを使用して、一般ストア設定を変更する。

ストア・アーカイブ内のストア・データ資産をすべてストア・サービスで編集できるわけではありません。ストア・サービスを使用して編集できる資産のリストについては、 WebSphere Commerce オンライン・ヘルプのトピック『ストア・データベース資産の変更』 を参照してください。ストア・アーカイブ内のその他の資産を編集するには、XML 資産を直接編集してください。

ストア・サービスの使い方の詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

いつストア・サービスを使用するか: サンプル・ストア・アーカイブをコピーする場合、およびストア・アーカイブ形式のデータベース資産を編集する場合は、ストア・サービスを使用してください。

- WebSphere Commerce ローダー・パッケージ

  WebSphere Commerce ローダー・パッケージは、 WebSphere Commerce データを準備してロードするユーティリティーで構成されます。 WebSphere Commerce データベースに大量のデータをロードしたり、データベースのデータを更新したりする場合には、ローダー・パッケージを使用してください。このパッケージ内のローダー・ユーティリティーは、有効かつ形式化された XML を使用します。XML 文書のエレメントはデータベースのテーブル名にマップされ、エレメント属性は列にマップされます。

  ローダー・パッケージを使用してデータ資産を開発およびロードする方法の詳細については、 225 ページの『第 7 部 ストアの発行』を参照してください。

  いつ WebSphere Commerce ローダー・パッケージを使用するか: データベース資産を初めて WebSphere Commerce データベース資産にロードするとき、およびそれらを更新するときに WebSphere Commerce ローダー・パッケージを使用します。

  重要: データベース・スキーマを変更した場合は、ローダー・パッケージを使用しないとデータをデータベースにロードできません。

- WebSphere Commerce Accelerator

  WebSphere Commerce Accelerator は、さまざまなストア操作でオンライン・ストアを保守するのに使用される、オンライン・ツールのワークベンチです。とはいえ、WebSphere Commerce Accelerator はデータベース内の既存のデータを編集できるため、サンプル・ストア・データであれ、自分で作成したデータであれ、データベースに初めてデータを読み込んだ後は、これをストア開発ツールとして使用できます。 WebSphere Commerce Accelerator を使用して編集できるデータベース資産のリストについては、WebSphere Commerce オンライン・ヘルプのトピック『ストア・データベース資産の変更』 を参照してください。

  いつ WebSphere Commerce Accelerator を使用するか: WebSphere Commerce データベースにデータを読み込んだ後、 WebSphere Commerce Accelerator を使用します。

- データベースを直接編集する

SQL の挿入 (INSERT) を使用してデータベースを直接編集することも、いつでも可能です。

注: SQL は、データベース固有のものです。 Oracle で必要とされる SQL 構文は、異なっています。 SQL ステートメントには必然的にデータベース固有の値が含まれることになり、そのような SQL ステートメントは、別の WebSphere Commerce Server インスタンスで使用できないことがあります。

バック・オフィス開発用のツール

バック・オフィス開発用のツール (コマンドの作成と拡張、カスタマイズされたコードの作成、およびビジネス・ロジックのインプリメントを含む) は、 *IBM WebSphere Commerce プログラマーズ・ガイド* に説明されています。

ストア開発者の役割

ストア開発者は、3 種類のストア資産をすべて開発します。ストアフロント資産 (JavaServer Pages (JSP) ファイルおよびバック・オフィスを含む) を設計し、インプリメントします。また、新しいコマンドや必要なカスタマイズ・コードの作成も行います。ストア・データの作成も行いますし、 WebSphere Commerce に組み込まれている標準機能すべてを変更することもできます。

ストアフロントおよびストア・データを作成するストア開発者は、Java、JavaScript、 HTML、JSP テクノロジーのプログラミング・スキルがなければならず、WebSphere Commerce ストア・アーキテクチャー、ストア・データ、およびストア・アーカイブに精通していなければなりません。

バック・オフィスを作成するストア開発者は、Java、JavaBeans、 VisualAge® for Java、 J2EE プログラミングのプログラミング・スキルおよび WebSphere Commerce プログラミング・モデルとオブジェクト・モデルに精通している必要があります。 *IBM WebSphere Commerce プログラマーズ・ガイド* には、バック・オフィスのカスタマイズに関する詳細が説明されています。

ストア開発者はデータベース開発者および Web デザイナーと協働することになるかもしれません。データベース開発者は、カスタマイズされたストア機能をインプリメントしたり、既存のデータベース情報と統合したりするために、WebSphere Commerce データベース・スキーマの修正や拡張を行います。このメンバーは DB2® または Oracle のデータベース管理者のスキルを持つのが普通です。

Web デザイナーは、サイトのルック・アンド・フィールを作成し、ストア開発者と協働してストア・ページを作成します。 Web デザイナーには、マルチメディア・ツールの使用経験や、HTML と JavaScript のスキルが必要です。また、JSP テクノロジーに精通している必要があります。

注: WebSphere Commerce Server には、データベース開発者の役割と Web デザイナーの役割は定義されません。必要な場合は、データベース開発者と Web デザイナーにストア開発者のアクセス権を割り当ててください。

ストア・アーカイブが作成されると、ストア開発者は手動、あるいは「ストア・プロファイル」、「税」、および「配送」のノートブックを使用してそのストア・ア

ーカイブに対して変更を加える権限があります。しかし、ストア・アーカイブを WebSphere Commerce サーバーに発行する権限はありません。

第 2 部 ストアフロントの開発

第 3 章 ストアフロントの開発

この章では、WebSphere Commerce ストアフロント・アーキテクチャーの概要を示します。これには、ストアの外側の部分となる Web 資産 (HTML ページ、JSP ファイル、スタイルシート、イメージ、グラフィックス、およびその他のマルチメディア・ファイル・タイプなど) が、顧客に対してどのように表示されるかが含まれます。

ストアフロント・アーキテクチャー

WebSphere Commerce では、*コマンド* と*ビュー* によって、ストアフロント内の Web 資産を顧客に表示します。

- *コマンド* は、特定のビジネス処理 (ショッピング・カートへの商品の追加、オーダーの処理、顧客の住所録の更新、特定の商品ページの表示など) を実行します。アクションが完了すると、コマンドはビューを戻します。
- *ビュー* はコマンドとユーザー・アクションの結果を表示します。つまり、ビューはストア・ページ (JSP ファイル) を顧客に表示します。ビューが JSP ファイルを呼び出すためには、JSP ファイル名がビュー・レジストリー (VIEWREG) テーブルのビューに登録されていなければなりません。対応する JSP ファイルは、`WCS Stores` webapp doc ルートの下にあるストアのサブディレクトリー (storedir) に、JSP ファイル名で保管されます。

コマンドとビューはどちらも URL を使用して呼び出されます。たとえば、顧客がサンプル・ストア内で**「ショッピング・カート」**をクリックすると、URL `https://hostname/path/OrderItemDisplay?` が呼び出されます。この URL は、WebSphere Commerce Server に渡されます。 WebSphere Commerce Server は OrderItemDisplay コマンドを呼び出し、ショッピング・カートのページが顧客に表示されます。

顧客がサンプル・ストア内で**「ヘルプ」**をクリックすると、URL `https://hostname/path/HelpView?` が呼び出されます。この URL は、WebSphere Commerce Server に渡されます。 WebSphere Commerce Server は HelpView を呼び出します。これは「ヘルプ」ページを戻します。

また、WebSphere Commerce Server では複数のコマンドを 1 つの URL にマップすることができます。これにより、オプションで、コマンドのインプリメンテーションをストアごとに独自のものにすることができます。

同様に、WebSphere Commerce Server は、複数の JSP ファイルを単一のビューにマップすることもできます。これにより、オプションで、各ストアはデバイス・タイプごとに異なる JSP ファイル名を登録することができます。

注: 商品の表示コマンドおよびカテゴリーの表示コマンドは、ビューとともに JSP ファイル名を戻します。商品およびカテゴリーを表示するこれらの JSP ファイル名は、カタログ・データに保管されます。詳細については、82 ページの『ストア・カタログ資産の表示』を参照してください。オプションで、商品および

カテゴリーを表示するために割り当てる JSP ファイルを、ストアでサポートされるメンバー・グループまたは言語ごとに変えることができます。

デフォルト・コマンドおよびデフォルト・ビュー

WebSphere Commerce には、ストアで使用できるデフォルトのコマンドと、デフォルトのビューが備わっています。これらのデフォルト・コマンドとデフォルト・ビューは、 wcs.bootstrap.xml ファイルにリストされています。ブートストラップ・ファイルは以下のディレクトリーに置かれています。

- NT `drive:¥WebSphere¥CommerceServer¥schema¥xml`
- 2000 `drive:¥Program Files¥WebSphere¥CommerceServer¥schema¥xml`
- AIX `/usr/WebSphere/CommerceServer/schema/xml`
- Solaris `/opt/WebSphere/CommerceServer/schema/xml`
- Linux `/opt/WebSphere/CommerceServer/schema/xml`
- 400 `/qibm/proddata/WebCommerce/schema/xml`

必要なコマンドまたはビューが提供されていない場合は、独自に作成することができます。コマンドとビューの作成についての詳細は、 *IBM WebSphere Commerce プログラマーズ・ガイド* を参照してください。

ストア・ページの作成

ストアフロントを作成する上で最大のタスクは、実際のストア・ページを作成することです。ストア・ページの開発作業を開始する前に、以下の計画アクティビティーを完了する必要があります。

- 必要なストア・ページのリストの開発
- コマンドとビューの URL のリストの開発
- JSP ファイル名とビューの関連付け

ストア・ページのリストの開発

ストアを作成するために必要なページのリストを開発するには、ストアのビジネス要件や機能要件、および定義されているビジネス・プロセスのすべてを知っていなければなりません。

ユース・ケースからの作業

多くの人々は、要件をユース・ケースの形式で収集します。ユース・ケースは、ストアのビジネス・プロセスを、顧客と提案システムの間の相互作用の形で定義します。オンライン・ストアの場合、顧客がストアに登録したり、カタログをブラウズしたり、アイテムをオーダーしたりする方法を、ユース・ケースで定義できるかもしれません。

オンライン・ヘルプに、サンプル・ストアのビジネス・プロセスを詳述している一連のユース・ケースが収められています。これらのユース・ケースは、サンプル・ストアのフローをより良く理解するのに役立ちますし、独自のストアのユース・ケースを作成するためのガイドとしても使用できます。

以下に、登録ユース・ケースの例を挙げます。

登録ユース・ケース: 登録処理により顧客はデータベースに個人情報を入力することができます。

実行者:

- 顧客

メイン・フロー: 顧客はサイドバーから**「登録」**を選択します。次に、システムは以下のフィールドのあるページを表示します。

- E メール
- パスワード
- 確認パスワード
- 名
- 姓
- 年齢 (オプション)
- 性別 (オプション)

顧客は上記フィールドに該当する情報を入力し、**「送信」**を選択します。システムはシステムに新規顧客を作成し、顧客の情報を保存します (E1、E2、E3)。システムは顧客に、個人アカウント管理のユース・ケースの手順に従い、アカウントを管理するよう促すプロンプトを出します。

代替フロー: なし

例外フロー: E1: E メール・アドレスがすでに存在する場合

- E メール・アドレスがすでにシステムに存在する場合、システムは、ユーザーに別の E メール・アドレスの入力を求めるエラー・メッセージを表示します。そしてそのユース・ケースを最初から再開します。

E2: 必須フィールドが抜けている場合

- 以下のフィールド (E メール、パスワード、確認パスワード、名、姓) の 1 つでも指定されていない場合、システムはエラー・メッセージを表示します。そしてそのユース・ケースを最初から再開します。

E3: パスワードが無効な場合

- パスワードが無効であるか確認パスワードと一致しない場合、システムは警告を表示します。

ストア・ショッピング・フローの決定: ユース・ケースを開発してストアのビジネス・プロセスを示すか、別の方法を使用するかにかかわりなく、いったんビジネス・プロセスが使用可能になったら、ストアのショッピング・フローを作成することができます。

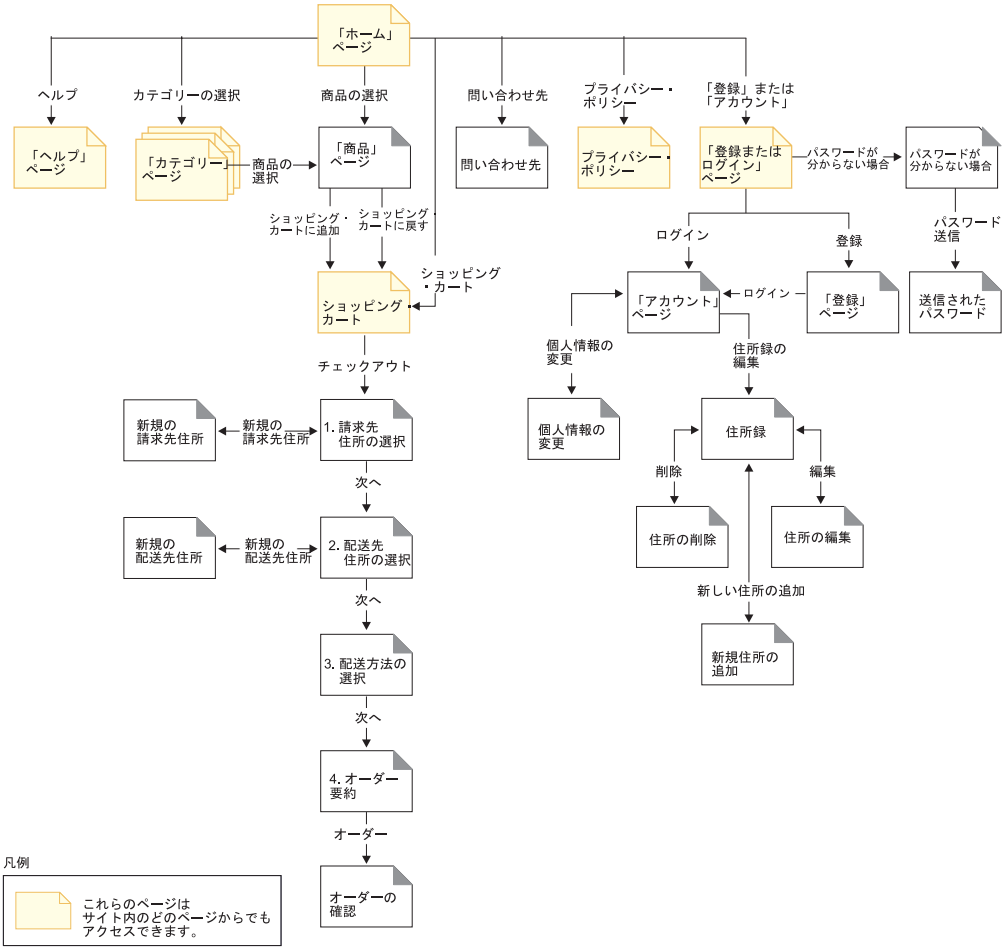
注: ユース・ケースにはしばしば、「顧客が**「送信」**を選択すると、「オーダー」ページが表示されます」などといったフロー情報が含まれているため、ショッピング・フロー・ダイアグラムを作成する上で役立つ情報となることがあります。

ショッピング・フローは、顧客がストアの中をどのように移動するかを示すにあた

り、ストアに定義されている要件やビジネス・プロセスを反映します。たとえば、ホーム・ページからサイトに入ってきた顧客に対し、カタログをブラウズする前に登録を済ませるよう促すこともできますし、登録しなくてもゲストとしてカタログを見られるようにすることもできます。また、顧客が「クイック・チェックアウト」できるショッピング・フローもあれば、購入のたびにすべてのチェックアウト・ステップを完了することの必要なショッピング・フローもあります。あるいは、顧客が両方のチェックアウトから選択できるようなショッピング・フローもありえます。

ストアのフロー・ダイアグラムが完成したかどうかを確かめるため、ストアのユース・ケースの全ステップをストアのフロー・ダイアグラムに表示するようにしてください。

InFashion サンプル・ストアのショッピング・フローに関する以下のダイアグラムのように、ショッピング・フローを視覚的にマップすると、顧客がストアをどのように移動するかを見ることができます。



InFashion のショッピング・フローのダイアグラムは非常に簡潔です。ストアの中の顧客の移動を示すメイン・フローはありますが、エラー・シナリオは何もありません。たとえば、間違ったパスワードを使って顧客がログインした場合、あるいは間違ったクレジット・カード番号を入力した場合は、どうなるのでしょうか。しかし、このような簡潔なダイアグラムでも、ストアに必要なページのリストを開発す

ることが可能です。まず、ショッピング・フローのダイアグラムにリストされているすべてのページのために、ビューを作成する必要があります。

たとえば、InFashion の図と同じショッピング・フローを使用してストアを作成する場合、以下のページを作成する必要があるでしょう。

注: 以下の表には、InFashion ストアで使用されるビュー名がリストされています。

InFashion ショッピング・フロー・ダイアグラムのページ (顧客の側から見たもの)	**対応するビュー**
「ホーム」ページ	StoreCatalogDisplayView
「ヘルプ」ページ	HelpView
「問い合わせ先」	ContactView
「プライバシー・ポリシー」	PrivacyView
「登録またはログイン」ページ	LogonForm
「パスワードが分からない場合」	LogoffView
「送信されたパスワード」	ResetPasswordForm
「アカウント」ページ	LogonForm
「個人情報の変更」	UserRegistrationForm
住所録	AddressBookForm
「新規住所の追加」	AddressForm
「住所の削除」	AddressBookForm
「住所の編集」	AddressForm
「登録」ページ	UserRegistrationForm
「ショッピング・カート」	OrderItemDisplayViewShiptoAssoc
「請求先住所の選択」	OrderItemDisplayViewShiptoAssoc
「新規の請求先住所」	OrderItemDisplayViewShiptoAssoc
「配送先住所の選択」	OrderItemDisplayViewShiptoAssoc
「新規の配送先住所」	AddressForm
「配送方法の選択」	OrderItemDisplayViewShiptoDsp
「オーダー要約」	OrderDisplayPendingView
「オーダーの確認」	OrderOKView

注: InFashion で使用されるビューの多くは、特に InFashion 用に作成されたものです。これらのビューは、InFashion ストア・アーカイブにある `command.xml` ファイルにリストされています。詳細については、53 ページの『WebSphere Commerce でのコマンド、ビュー、および URL の登録』を参照してください。

上記の表は、作成する必要のある一連の基本的なページを暗に示しているにすぎません。他に作成する必要のあるページを決めるためには、ユース・ケース、またはビジネス・プロセスを定義するために使用する別の方法を詳しく調べてみてください。

エラー・ページ: ユース・ケースの中の例外フローも、ストアで作成する必要のあるエラー・ページを決めるのに役立ちます。 InFashion の登録ユース・ケースでは、以下の例外フローが指定されています。

- E メール・アドレスがすでに存在する: E メール・アドレスがすでにシステムに存在する場合、システムは、ユーザーに別の E メール・アドレスの入力を求めるエラー・メッセージを表示します。そしてそのユース・ケースを最初から再開します。
- 必須フィールドが欠落している: 以下のフィールド (E メール、パスワード、確認パスワード、名、姓) の 1 つでも指定されていない場合、システムはエラー・メッセージを表示します。そしてそのユース・ケースを最初から再開します。
- パスワードが無効である: パスワードが確認パスワードと一致しない場合、システムは警告を表示します。

結果として、それぞれの例外フローごとに、エラー・ページまたはエラー・メッセージを作成する必要が生じます。

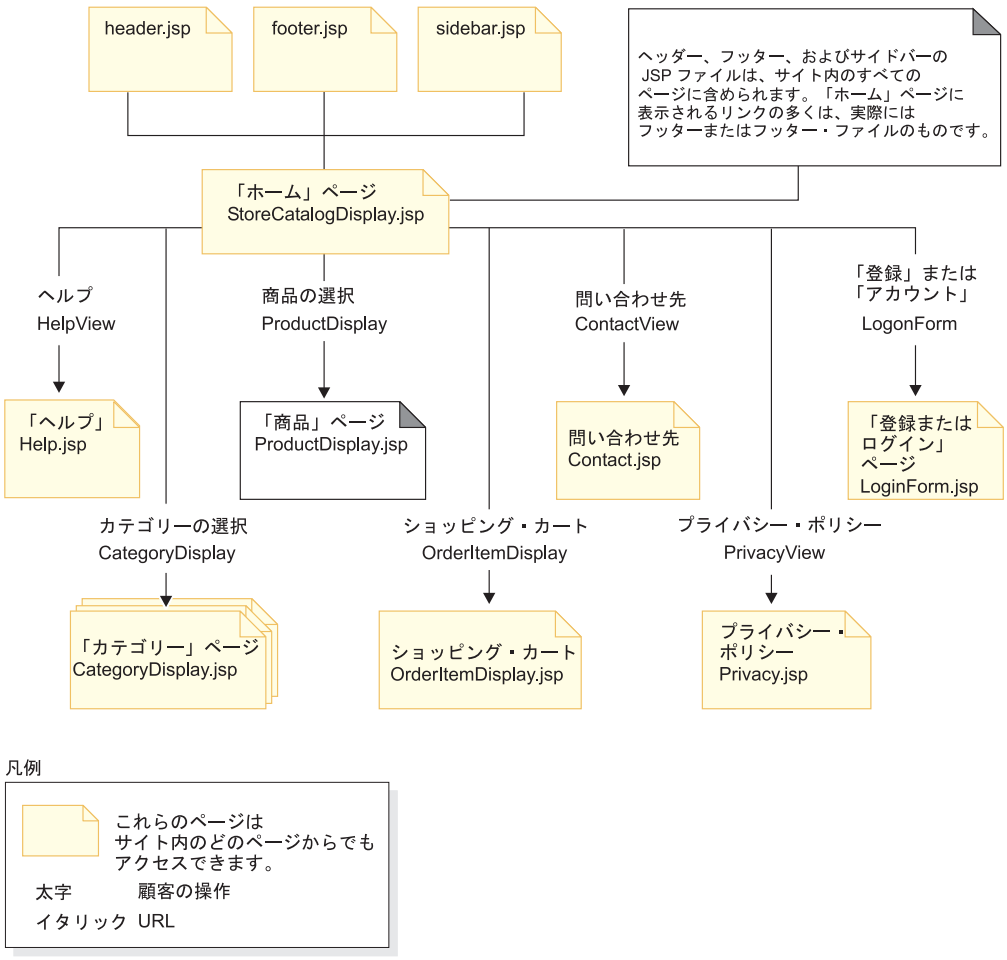
コマンドとビューの URL のリストの開発

InFashion のショッピング・フロー・ダイアグラムで示したように、チェックアウトや登録などのビジネス・プロセスは、いくつかのページを必要とする場合があります。これらのページを単にページの集合にとどめず、組み合わせて作業ビジネス・プロセスまたはフローとするためには、ページにコマンドまたはビューを組み込む必要があります。

必要な URL のリストの開発

ストアを作成するために必要なページのリストを開発したのとまったく同様、ストアのビジネス・プロセスをインプリメントするために必要なコマンドとビューの URL のリストを開発することも必要です。ストアのショッピング・フロー・ダイアグラムとデフォルトのコマンドとビューのリストを使用して、各アクションを完了するために必要な URL を識別します。

サンプル・ストアで使用されているコマンドとビューの URL を理解しておくことも、ストアで必要な URL を判別する助けになります。以下の図は、InFashion のショッピング・フロー・ダイアグラムにあるいくつかのアクションの URL を示しています。詳しくは、WebSphere Commerce のオンライン・ヘルプにある『サンプル・ストアの情報』を参照してください。



JSP ファイル名とビューの関連付け

WebSphere Commerce Server は、要求に対する応答としてビューを構成するために、ビュー・コマンドを使用します。 WebSphere Commerce Server には、以下のビュー・コマンドが用意されています。

- `HttpForwardViewCommandImpl`: このビュー・コマンドは、ビュー要求を JSP ファイルに転送します。
- `HttpRedirectViewCommandImpl`: このビュー・コマンドは、ビュー要求を別の URL にリダイレクトします。
- `HttpDirectViewCommandImpl`: このタイプのビュー・コマンドは、応答ビューを直接クライアントに送信します。 このコマンドは JSP ファイルを呼び出しません。直接ビューでは、ビュー・コマンドによってではなくコントローラー・コマンドによって、出力応答が作成されるようになります。

JSP ファイルを直接レンダリングするには、 `HttpForwardViewCommandImpl` ビュー・コマンドを使用します。たとえば、InFashion で使用される URL を示すダイアグラムの場合、「ヘルプ」ページ (`Help.jsp`) を表示するには、HelpView をビューに登録し、 `Help.jsp` および `HttpForwardViewCommandImpl` コマンドと関連付けます。これを次の例に示します。

```
<viewreg
viewname="HelpView"
devicefmt_id="-1"
storeent_id="@storeent_id_1"
interfacename="com.ibm.commerce.command.ForwardViewCommand"
classname="com.ibm.commerce.command.HttpForwardViewCommandImpl"
properties="docname=help.jsp"
internal="0"
https="0"
/>
```

インターフェースおよびインプリメンテーション・クラスに完全修飾クラス名が使用されていることに注意してください。

注: この例では、ビューを呼び出す URL は制限されていません。つまり、誰でも URL に直接アクセスすることができます。この手法を使用する場合は、 JSP ファイルが公用データのみをレンダリングするようにしてください。

表示コマンドから戻されるビューをレンダリングするには、 HttpForwardViewCommandImpl ビュー・コマンドを使用します。表示コマンドはデータベースからデータを読み取りますが、そのデータに対して変更を加えることはありません。たとえば、InFashion で使用される URL を示すダイアグラムの場合、OrderItemDisplay コマンドは OrderItemDisplayViewShiptoAssoc ビューを戻します。このビューがビュー・レジストリーに登録されたときに、 OrderItemDisplay.jsp と HttpForwardViewCommandImpl がそのビューに関連付けられました。これを次の例に示します。

```
<viewreg
viewname="OrderItemDisplayViewShiptoAssoc"
devicefmt_id="-1"
storeent_id="@storeent_id_1"
interfacename="com.ibm.commerce.command.ForwardViewCommand"
classname="com.ibm.commerce.command.HttpForwardViewCommandImpl"
properties="docname=OrderItemDisplay.jsp"
internal="0"
https="0"
/>
```

注: この例では、ビューを呼び出す URL は制限されています。つまり、指定されたアクセス権を持つ人だけが URL を直接呼び出すことができます。このビューへのアクセスは、そのコマンドの URL へのアクセス権を持つ人によって制御されます。

使用するあらゆる表示コマンド (たとえば OrderItemDisplay) に関連したあらゆるビューについて、 JSP ファイル名を関連付ける必要があります。ビューに JSP ファイル名を関連付けることについては、 53 ページの『WebSphere Commerce でのコマンド、ビュー、および URL の登録』を参照してください。

注: ProductDisplay および CategoryDisplay で示されるのは、ビュー・レジストリーではなくカタログ・データ中の関連した JSP ファイル名です。

データベースを変更するコマンドから戻されるビューをレンダリングするには、 HttpRedirectViewCommandImpl ビュー・コマンドを使用します。リダイレクト・ビューを使用するには、 URL において &URL= パラメーターを使ってビュー名を指定してください。たとえば、InFashion サンプル・ストアの AddressForm に住所情報を追加し、**「送信」**をクリックすると、AddressAdd コマンドが呼び出されます。

AddressAdd コマンドを呼び出すために使用する URL は、 &URL= パラメーターを使用して、AddressBookForm をビューとして指定します。結果として AddressBookForm ビューへのリダイレクトが生じます。 AddressBookForm ビューがビュー・レジストリーに登録されたときに、 AddressBookForm.jsp と HttpForwardViewCommandImpl がそのビューに関連付けられました。

すべての非表示コマンドについては、 URL=parameter テクニックを使用してください。非表示コマンドとは、データベース内のデータに変更を加えるコマンドです。

第 3 部 ストア・データの概要

第 4 章 ストア・データ

この章は、ストアを構成する WebSphere Commerce Server のストア・データのアーキテクチャーおよびデータ資産の概要を示します。 WebSphere Commerce Server の情報モデルについてもこの章で説明します。

ストア・データとは?

ストア・データとは WebSphere Commerce Server のデータベースにロードされる情報で、これによってストアは機能するようになります。ストアが正しく動作するためには、顧客のすべてのアクティビティーをサポートするデータがストアに配置されていなければなりません。たとえば、顧客が買い物できるようにするには、販売商品のカタログ (カタログ・データ)、オーダーの処理に関係するデータ (税および配送データ)、および要求を実行するための在庫 (在庫およびフルフィルメント・データ) をストアに含めなければなりません。

ストア・データの情報モデル

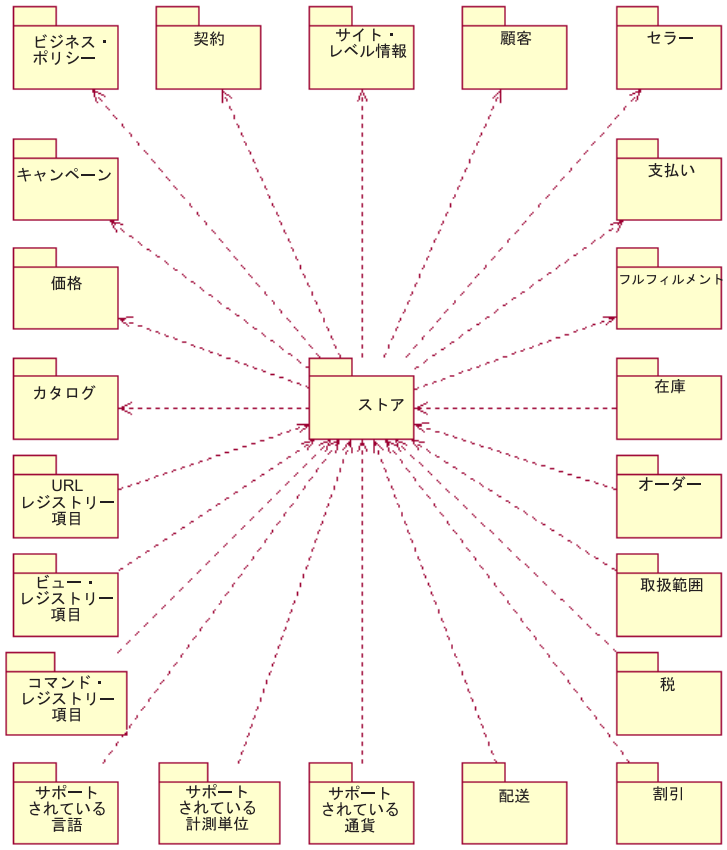
本書では、WebSphere Commerce Server のストア・データの構造を示すために情報モデルが使用されます。 WebSphere Commerce Server の情報モデルは、WebSphere Commerce Server のデータおよびオブジェクト・モデルに含まれる情報を高水準で抽象化したものです。情報モデルは、データおよびオブジェクト・モデルの最も重要な機能を強調していますが、スキーマおよびオブジェクトのインプリメンテーションに固有の、より低いレベルの詳細は含まれていません。

たとえば、データまたはオブジェクト・モデル内のテーブルやオブジェクトにエンティティー・リレーションシップのデータ (外部鍵の対など) が含まれている場合、それらのテーブルやオブジェクトがエンティティーとして情報モデルに表示されることはありません。その代わりに、情報モデルの中でそのようなエンティティー・リレーションシップは、エンティティー同士の関連を表す線で表現されます。また、情報モデルでは*詳細* 拡張が示されません (これはインプリメンテーションの結果として別個のテーブルに保管される、エンティティーの追加のデータ属性です。たとえば商品の説明は、商品のエンティティーとは別個に保管される拡張です)。関係オブジェクトおよび詳細拡張の詳細については、 WebSphere Commerce オンライン・ヘルプのオブジェクト・モデルを参照してください。

WebSphere Commerce オブジェクトおよびデータ・モデルの詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

ストア・データ資産

以下の図は、WebSphere Commerce ストアのデータ資産を示しています。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce 情報モデルの一部です。情報モデルの詳細については、 29 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則については、 311 ページの『付録 A. UML の凡例』を参照してください。

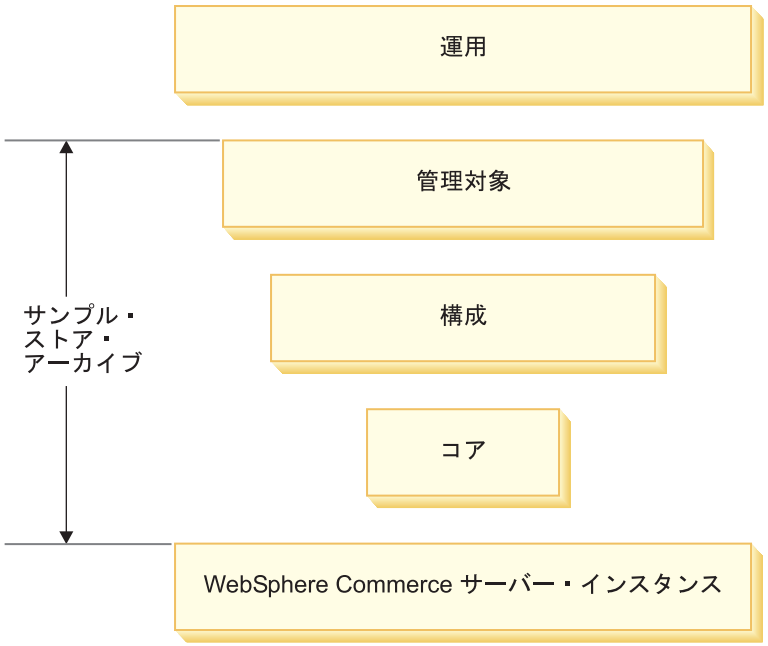
図の中の矢印の 2 つの方向に注意してください。通貨からストアのように、ストアの方向を指している矢印がいくつかあります。この場合、通貨資産がこの特定のストア専用のものであり、ストアの一部であることを示しています。これらはそのストアがサポートする通貨です。そのストアが削除されると、ストアによってサポートされる通貨のリストも削除されます。

図の中のカタログのようにストアから資産に矢印が向いている場合、その資産は他のストアと共有することができます。 1 つのカタログを複数のストアで共有することができます。しかし、ストアが削除されても、カタログはそのまま残ります。

上記の図で示されているそれぞれのデータ資産については、 39 ページの『第 4 部 ストア・データの開発』の各章で詳しく説明します。

ストア・データ・アーキテクチャー

WebSphere Commerce ストア内のデータは、以下の図に示されているアーキテクチャーに従います。 30 ページの『ストア・データ資産』にある図で示されているストア・データ資産はそれぞれ、以下に示すストア・データのタイプの 1 つ以上に属するものとして分類することができます。



WebSphere Commerce サーバー・インスタンス

基本レベルのデータは、WebSphere Commerce サーバー・インスタンスに含まれています。インスタンスが作成されると、 XML 形式でロードされるブートストラップ・ファイルによりデータベースに情報が取り込まれます。ブートストラップ・ファイルは、以下のデータ・タイプを作成します。

- 計算方法タイプ、デバイス・タイプ (ブラウザー、E メール、I モードなど)、メッセージ・タイプおよび役割
- デフォルト管理 ID (WCSADMIN)
- デフォルトのコマンド、ビュー、および URL
- デフォルトのビジネス・ポリシー
- インスタンスによってサポートされる言語および通貨
- デフォルトの組織 (ストアの所有者として使用可能)
- デフォルトのサイト組織
- デフォルトのストア・グループ

この情報は、そのインスタンス内に存在するすべてのストアで適用可能です。また、この情報は 30 ページの『ストア・データ資産』の図で、サイト・レベル情報として示されています。

ブートストラップ・ファイルおよびこのファイルがデータを読み込むデータベース・テーブルの詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

コア・データ

ストア・データの次のレベルは、コア・データです。コア・データは、ストアの最小データを構成します。これには次のものが含まれます。

- STOREENT テーブル内のストア ID。これはデータベースにストアを作成します。
- デフォルトの契約
- 契約データベース・テーブル内のストア ID
- 契約データベース・テーブル内のストアを所有する組織のメンバー ID
- STORE テーブル内のストア・ディレクトリー。ストア・ディレクトリーは、ストアの Web 資産があるディレクトリーです。
- STADDRESS テーブル中のストアのアドレスのためのニックネームまたは識別名。ニックネームはストアごとに固有です。

ストア・サービスを使用してストアを作成する場合であれば、この情報は新しいストア・アーカイブにより作成されます。ストア・サービスを使用すると、ストア所有者の役割を果たすデフォルトの組織を選択することができます。あるいは、管理コンソールを使用して、所有者の役割を果たす別の組織を作成することもできます。ストアの作成にストア・サービスを使用しなかった場合、ローダー・パッケージを使用してその情報をデータベースにロードするか、またはデータベースを直接編集する必要があります。

30 ページの『ストア・データ資産』の図にあるストア・データは、コア・データです。

構成データ

構成データは、コマース・サーバーのランタイムを制御します。共通のサーバー・ランタイムは、コマース・アプリケーションがデプロイ、実行されるフレームワークを提供します。このフレームワークは、プログラミング・モデル、プロセス・モデル、例外処理、トランザクション制御、データ・アクセス、および永続モデルで構成されます。共通のサーバー・ランタイムは WebSphere Application Server によって提供されるランタイム・サービスを使用して、 WebSphere Commerce サーバー・アプリケーションをサポートします。構成データは、ストア・ページを表示するためにストアが使用するコマンド、ビュー、および JSP ファイルを決定します。

30 ページの『ストア・データ資産』の図に示されている以下のデータ資産は、構成データとして分類されます。

- コマンド・レジストリー項目
- ビュー・レジストリー項目
- URL レジストリー項目

管理対象データ

管理対象データは、セラーが作成するデータです。セラーのサイトの顧客にとって、これは読み取り専用データです。セラーはこのデータの状態を完全に制御できるため、管理対象データはコンテンツ・マネージメント・システムを介して管理することもできます。

30 ページの『ストア・データ資産』の図に示されている以下のデータ資産は、管理対象データとして分類されます。

- キャンペーン
- ビジネス・ポリシー
- 契約
- 配送センター
- 管轄区域
- 税
- 割引
- 配送
- 通貨
- 計測単位
- 言語
- カタログ
- 価格
- 顧客
- セラー
- 支払

運用データ

運用データとは、サイトの顧客がサイトとの対話の結果として (直接的または間接的に) 作成または変更するデータのことです。たとえば、顧客のオーダーは運用データと見なされます。また、在庫レベルも運用データと見なされます。ストアの運用に応じて上下するからです。顧客も運用データと見なされます。また、セラーにより作成されたデータも運用データと見なすことができます。

セラーは運用データに対して加える変更を完全には制御できないため、コンテンツ・マネージメント・システムを使用してこのデータを管理しても意味がありません。

30 ページの『ストア・データ資産』の図に示されている以下のデータ資産は、運用データとして分類されます。

- オーダー
- 在庫
- フルフィルメント
- 顧客

注: 中には、運用データと管理対象データのどちらに属するかを区別しにくいインスタンスもあります。たとえば、あるストアでは顧客および契約データが管理対象データと見なされ、別のストアでは同じタイプのデータが運用データと見なされる場合があります。最初のストアでは、顧客の集団が限定されているため、顧客データとそれに関連する契約を管理しているのかもしれません (つまり、顧客はオンラインで登録できません)。一方、2 番目のストアでは、顧客はオンラインで登録して、オンラインで契約情報を作成できます。

2 番目の例は、カタログ・データです。単一セラー・サイトでは、カタログは管理対象データと見なされます。マーケットプレイスのサイトでは、カタログ・データは運用データと見なされます。

サイトによっては、同じタイプのレコードでも、管理対象データと見なされる場合と、運用データと見なされる場合とがあります。たとえば、デフォルト契約は管理対象データと見なされる一方、オンラインで結ばれる特定の契約は運用データと見なされることがあります。

ストア・データ・アーキテクチャーおよびサンプル・ストア

WebSphere Commerce に付属のサンプル・ストアには、ストア・データ・アーキテクチャーのストア・データ・タイプがほとんど含まれています。たとえば、WebSphere Commerce Server インスタンスがあらかじめ存在していなければ、サンプル・ストアを使用してストアを作成したり、あるいはサンプル・ストアを発行したりすることはできません。その後、ストア・サービスのツールを使って、サンプル・ストアに基づくストアを作成するときに、コア・データが作成されます。サンプル・ストアには、必要となるすべての構成データすべてや、機能するストアに必要な管理対象データのほとんどが含まれています。特定のサンプル・ストアに基づいてストアを作成する場合、 WebSphere Commerce Accelerator のツールを使ってデータのセットアップのいくつかを完了するように指示される場合があります。

データ作成用のツール

WebSphere Commerce には、ストア・データを作成したり操作したりするためのツールがいくつか用意されています。それらのツールは、以下のとおりです。

WebSphere Commerce ローダー・パッケージ

ローダー・パッケージは、基本的に言って、データを準備して WebSphere Commerce データベースにロードするユーティリティーで構成されます。詳細については、 225 ページの『第 7 部 ストアの発行』を参照してください。

ストア・サービス

ストア・サービスは、データベース内のアクティブなデータではなく、ストア・アーカイブ形式の事前発行済みデータを編集します。また、ストア・サービスによって、全ストア・データ資産をデータベースに発行することもできます。詳細については、 225 ページの『第 7 部 ストアの発行』を参照してください。

管理コンソール

管理コンソールは、管理操作および構成タスクを実行して、サイトまたはストアを制御する手段になります。また、管理コンソールを使用して、新しい組織や新しいユーザーを作成したり、ユーザーに役割 (ストア開発者、ストア管理者、サイト管理者など) を割り当てたりすることもできます。また、管理コンソールを使用して、ストアで使用可能な通知タイプおよびメッセージ・タイプを識別することもできます。

WebSphere Commerce Accelerator

WebSphere Commerce Accelerator は、オンライン・ツールのワークベンチで、これを使用してさまざまなストア資産を作成および保守することができます。大部分のストア・データは、 WebSphere Commerce Accelerator にあるツールを使用して作成および管理できます。しかし、WebSphere Commerce Accelerator を使用してデータを作成する前に、ストア・サービスの発行ツールかまたはローダー・パッケージを使用して、特定のデータをデータベースにロードしておかなければならない場合もあります。詳しくは、『ツールとストア・データの要約表』を参照してください。

組織管理コンソール

組織管理コンソールを使用すると、サイトまたはストアにアクセスする組織を管理することができます。組織管理コンソールを使用すると、バイヤーの管理者が組織内のバイヤーを管理することも可能です。

ツールとストア・データの要約表

以下の図は、各データ・タイプを作成するために使用できるツールを示しています。

データ作成用のツール	コア・データ	構成データ	管理対象データ	運用データ
WebSphere Commerce ローダー・パッケージ	コア・データを XML ファイルの形式でロードするには、ローダー・パッケージを使用します。詳細については、 47 ページの『XML ファイルによるストア・データ資産の作成』を参照してください。	構成データを XML ファイルの形式でロードするには、ローダー・パッケージを使用します。詳細については、 54 ページの『コマンド、ビュー、および URL の登録用 XML ファイルの作成』を参照してください。	管理対象データを XML ファイルの形式でロードするには、ローダー・パッケージを使用します。詳しくは、管理対象データ資産に関する対応する章を参照してください。	一般に、ローダー・パッケージで運用データをロードすることはできません。

データ作成用のツール	コア・データ	構成データ	管理対象データ	運用データ
ストア・サービス (ストア・アーカイブ形式の事前発行済みデータの場合のみ)	ストア・サービスを使用して新しいストア・アーカイブを作成すると、コア・データが自動的に作成されます。ストア・サービスの使い方の詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。	該当しない。	ストア・サービスを使用すると、以下の管理対象資産の一部分を作成および編集できます。 • 管轄区域 • 税 • 配送 • 通貨 • 言語 ストア・サービスを使用して編集または作成できるデータベース資産の詳細については、WebSphere Commerce オンライン・ヘルプの『ストア・データベース資産の変更』を参照してください。	該当しない。
管理コンソール	ストア所有者の役割を果たす組織を作成するには、管理コンソールを使用します。	該当しない。	該当しない。	該当しない。

データ作成用のツール	コア・データ	構成データ	管理対象データ	運用データ
WebSphere Commerce Accelerator	該当しない。	該当しない。	以下のデータを作成または編集するには、WebSphere Commerce Accelerator を使用します。 • キャンペーン • 契約 (データベースにあらかじめデフォルト契約が存在していないと、WebSphere Commerce Accelerator のビジネス関係管理ツールを使用して追加の契約を作成したり、既存の契約を変更したりすることができません。データベースにデフォルト契約を作成するには、ローダー・パッケージまたはストア・サービスを使用してください。)	顧客はストアに登録するとき、またはストアで買い物をするときに、運用データを作成します。ただし、場合によっては、WebSphere Commerce Accelerator を使用して顧客のオーダーを発行したり、返品を作成したりすることもできます。 また、WebSphere Commerce Accelerator を使用して在庫を管理することもできます。

データ作成用のツール	コア・データ	構成データ	管理対象データ	運用データ
WebSphere Commerce Accelerator (続き)	該当しない。	該当しない。	• フルフィルメント • 割引 • カタログ (データベースにあらかじめマスター・カタログが存在していないと、WebSphere Commerce Accelerator の商品管理ツールを使用してナビゲーション・カタログを作成したり、商品情報を追加または変更したりすることができません。データベースにマスター・カタログを作成するには、ローダー・パッケージまたはストア・サービスを使用してください。) • 価格	該当しない。
組織管理コンソール	該当しない。	該当しない。	該当しない。	顧客およびバイヤーは、ストアに入るときに作成されます。ただし、組織管理コンソールを使用して、バイヤーを承認したり、あるいは新規バイヤーを作成したりすることもできます。

第 4 部 ストア・データの開発

ここに含まれる章では、各ストア・データ資産についてさらに詳しく説明します。ここで説明されているストア・データ資産は、 WebSphere Commerce ストア・データのアーキテクチャー構造にしたがって編成されています。

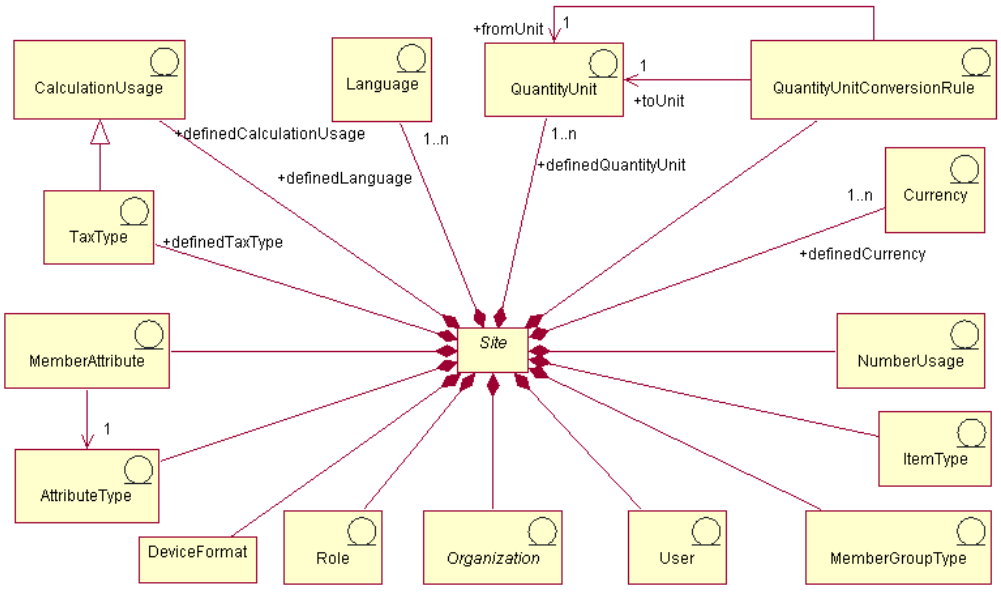
- WebSphere Commerce Server インスタンス
  - サイト
- コア・データ
  - ストア
- 構成データ
  - コマンド・レジストリー
  - ビュー・レジストリー
  - URL レジストリー
- 管理対象データ
  - 共有資産
    - カタログ
    - 価格
    - 契約 (ビジネス・ポリシーを含む)
    - フルフィルメント
    - キャンペーン
    - 支払
  - ストア資産だけに当てはまるもの
    - サポートされている言語
    - サポートされる通貨
    - サポートされる計測単位
    - 管轄区域
    - 配送
    - 課税
    - 割引
- 運用データ
  - 在庫
  - オーダー
  - 顧客

第 5 章 サイト資産

WebSphere Commerce Server インスタンスにはそれぞれ、関連情報の独自のデータベースがあります。インスタンスはブートストラップ・ファイルによって作成されます。ブートストラップ・ファイルは、スキーマが作成された後、データベース・テーブルに情報を取り込みます。データがロードされたら、該当するデータベース・テーブルで事前にロードされた情報を見ることができます。多くのデータベース・テーブルには、ストア・レベルの情報またはストア・グループ・レベルの情報が入っています。これは、ストアまたはストアのグループに固有の情報です。この情報は通常、ストア管理者によって管理されます。しかし、いくつかのテーブルには、インスタンス内のすべてのストアで使用可能な WebSphere Commerce サイト・レベルの機能を表す情報が入っています。これらのテーブルは、WebSphere Commerce サイト管理者によって管理される場合があります。この章では、それらの機能について説明します。ブートストラップ・ファイルの詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。ストア固有の資産情報に関する詳細については、45 ページの『第 6 章 ストア資産』を参照してください。

WebSphere Commerce のサイト資産について

以下の図は、サイトに含まれるデータのタイプ、およびそれらのデータのタイプとサイトとの関係を示しています。



この図で使用されている規則については、311 ページの『付録 A. UML の凡例』を参照してください。この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce 情報モデルの一部です。情報モデルの詳細については、29 ページの『ストア・データの情報モデル』を参照してください。

言語

サイトでは、 LANGUAGE テーブルの中に多くの *言語* を定義し、 LANGUAGEDS テーブルの中でそれらを記述することができます。各ストアは一般に、 STORELANG テーブルに行を追加することによって、これらの言語のサブセットをサポートします。事前に定義されている言語は、ドイツ語、中国語 (繁体字)、中国語 (簡体字)、日本語、韓国語、イタリア語、フランス語、スペイン語、ブラジル・ポルトガル語、および英語です。

メンバー属性

メンバー属性 は MBRATTR テーブルに保管されます。メンバー属性は、組織またはユーザーに応じた値を保管できる、定義済みの属性のセットを表します。その種の属性名の例として、JobFunction、ProcurementCard、SpendingLimit、ReferredBy、および CountryOfOperation などがあります。特定の組織またはユーザーの属性値は MBRATTRVAL テーブルに保管できます。これらの値は各ストアまたはストア・グループごとに異なる場合があります。

属性タイプ

属性タイプ は ATTRTYPE テーブルに保管され、属性値を表すために使用できる定義済みデータ・タイプを表します。データ・タイプの例として、INTEGER、STRING、および FLOAT があります。

メンバー・グループ・タイプ

メンバー・グループ・タイプ は MBRGRPTYPE テーブルに保管され、定義済みのメンバー・グループの使用法を表します。 MBRGRPUSG テーブルに行を追加することによって、メンバー・グループに使用法が割り当てられます。メンバー・グループの使用法の例として、AccessGroup (アクセス・コントロール・ポリシーに関して使用)、および UserGroup (顧客グループなど一般用) などがあります。

ユーザー

ユーザー は、認証されたユーザーを表します。ユーザーは一般に、購買組織に代わってオーダーを発行または承認する顧客、販売組織のオーダーを処理したり、あるいはストア・レベルの資産を保守したりする販売エージェント、または WebSphere Commerce Server インスタンスを保守するサイト管理者を表します。各ユーザーは、1 つのサイトと関連付けられ、USERS テーブルの中で定義されます。

組織

組織 という資産は、組織、および組織内にある組織単位を表します。組織は一般に、購買または販売に責任を持つビジネス・エンティティーを表します。企業間取引の購買組織内の顧客が発行するオーダーは、購買組織の代わりに顧客が発行したものとして記録されます。ストア、カタログ、および配送センターは、販売の特定の局面に責任を持つ組織によって所有されます。組織は ORGENTITY テーブルに定義されています。

役割

役割 は、組織内でユーザーに割り当てることができる、定義済みの役割のセットを表します。たとえば、ユーザーに販売組織内の顧客サービス担当者の役割を割り当てたり、あるいは購買組織内のバイヤー承認者の役割を割り当てたりすることもできます。デフォルトの役割の名前と説明は、ROLE テーブルに読み込まれます。特定の役割の詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

数量単位変換

各サイトには、*数量の変換規則* があります。それらの変換規則は、異なる計測単位を変換するために使用される乗算または除算の演算を表します。これは、QTYCONVERT テーブルに読み込まれます。

数量単位

数量単位 は、サイトの計測単位のセットを表します。それらは、QTYUNIT テーブルの中で定義され、QTYUNITDSC テーブルの中に記述します。各ストアは、計測単位の値を丸めて表示用に形式設定する方法を、使用目的に応じて指定できます。これは、QTYFORMAT テーブルに行を追加することによって行えます。

税タイプ

税タイプ は、税を計算する計算方法を表します。消費税と配送税は、税を計算する 2 つの異なる計算方法です。税タイプは TAXTYPE テーブルに定義されています。

計算の使用法

計算の使用法 は、 OrderPrepare コマンドで実行できる、さまざまな種類の計算を表します。計算の使用法は、割引、配送、消費税、配送税、および e クーポンに関して定義されます。計算の使用法は、CALUSAGE テーブルの中で定義されます。

通貨

各サイトでは、 SETCURR テーブルの中でいくつかの*通貨* を定義し、SETCURRDSC テーブルの中でそれらを記述します。各ストアは、CURLIST テーブルに行を追加することによって (サポートされる通貨に対してそれぞれ 1 行)、それらの通貨のサブセットをサポートします。

数値の使用法

数値の使用法 は、数値を使用する方法を表します。ストアは、表示する数値の使用方法に応じて、それぞれの数値に異なる丸めと形式設定のルールを指定することができます。たとえば、あるストアは単価については単価の使用法を指定して小数第 4 位で丸め、他の通貨の額についてはデフォルトの使用法を指定して小数第 2 位で丸めるかもしれません。数値の使用法は、NUMBRUSG テーブルの中で定義され、NUMBRUSGDS テーブルの中で記述されます。

アイテム・タイプ

アイテム・タイプ は、様々な基本アイテムを表します。 WebSphere Commerce には、ダイナミック・キットと通常アイテムの 2 種類の基本アイテムがあります。ア

イテム・タイプは、ITEMTYPE テーブルで事前定義されています。基本アイテムの詳細については、177 ページの『第 21 章 在庫資産』を参照してください。

デバイス形式

デバイス形式 は DEVICEFMT テーブルに保管され、サイトが使用する多くのデバイス形式 (ブラウザー、I モード、E メール、MQXML、および MQNC など) を表します。これらすべてのデバイス・タイプによって、ユーザーは様々なメディアを介してサイトと対話することができます。

注: 言語、通貨、数量単位、および数量単位変換規則など、サイト資産の一部については、サイト管理者は該当するテーブルに行を追加することによって、サイト・レベルの機能を拡張することができます。 その他のサイト資産については、それらが表すサイト・レベルの機能を拡張するために、関係するカスタマイズも必要になる可能性があります。たとえば、サイト管理者が、カスタマイズされた通貨記号と一緒に小計を表示するよう、数値の新しい使用法を追加するとします。その場合、小計を表示するプログラムにカスタマイズを加えて、小計の金額を表示用に形式設定する際、新しい小計の数値使用法を指定する必要があるかもしれません。

WebSphere Commerce のサイト資産の作成

サイト資産は、WebSphere Commerce Server のインスタンスを作成するときに作成されます。 WebSphere Commerce Server のインスタンスの作成の詳細については、*IBM WebSphere Commerce インストール・ガイド* の第 5 章『インスタンスの作成または変更』を参照してください。

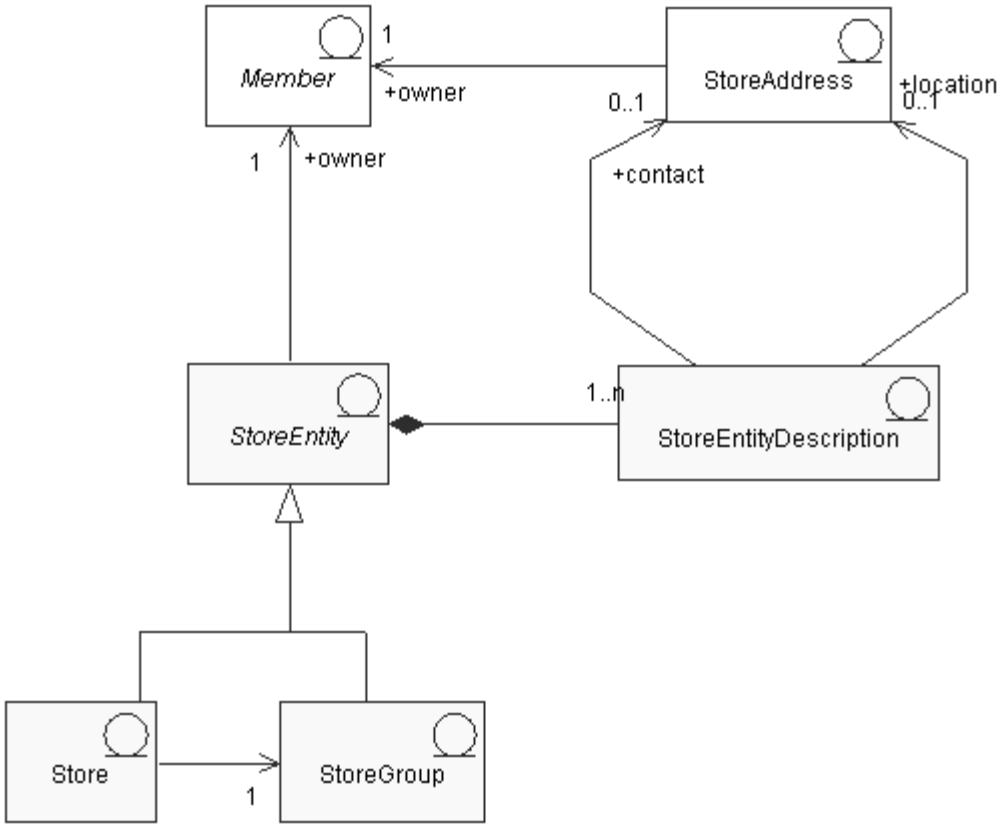
第 6 章 ストア資産

WebSphere Commerce でストアを作成するには、まず以下のものをデータベースに作成する必要があります。

- ストア
- ストアが属するグループ
- ストアまたはストア・グループの二者を表す、抽象ストア・エンティティー・オブジェクト。

WebSphere Commerce のストア資産について

以下の図は、WebSphere Commerce Server 内のストア資産を示しています。



この図と、ストア・データの説明の中の他のすべての図は、 WebSphere Commerce Server の情報モデルの一部です。情報モデルの詳細については、 29 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則の詳細については、 311 ページの『付録 A. UML の凡例』を参照してください。

ストア・エンティティー

ストア・エンティティー は、ストアまたはストア・グループのいずれかを表す抽象スーパークラスです。

ストア・エンティティーには 1 人の所有者 (メンバー) がいます。メンバーの詳細については、187 ページの『第 23 章 顧客資産とセラー資産』を参照してください。

ストア・エンティティーの説明

ストア・エンティティーの説明 は、ストア・エンティティーについて記述します。ストア・エンティティーには、説明が含まれることがあります。ストアが複数の言語をサポートしている場合ストア・エンティティーの説明が複数の言語のことがあります。説明には、ストア・エンティティーの連絡先住所が 1 つと、ストア・エンティティーのロケーションの住所が 1 つ含まれる場合があります。

ストア

ストア はストア・エンティティーです。ストアは、1 つのストア・グループに属していなければなりません。

ストア・グループ

ストア・グループ はストアの集合です。ストア・グループはストア・エンティティーです。ストア・グループは、共通情報 (ストア・グループ・レベルで保管でき、ストア・グループ内のすべてのストアで共有できるもの) のコンテナーとして機能します。たとえば、同じストア・グループ内のストアは、課税カテゴリー、サポートされる言語、サポートされる通貨、計算コード、および配送管轄区域などの情報を共有できます。

現在のところ、WebSphere Commerce Server に存在し、サイト管理レベルで保守できるストア・グループは 1 つだけです。

WebSphere Commerce Server のストア資産の構造の詳細については、WebSphere Commerce のオンライン・ヘルプの『ストアのオブジェクト・モデル』と『データ・モデル』を参照してください。

WebSphere Commerce でのストア資産の作成

WebSphere Commerce のストア・サービス・ツールを使用すると、以下のストア資産を作成または編集することができます。

- 連絡先資産内のストア ID とメンバー ID
- STOREENT テーブル内のストア ID
- STORE テーブル内のストア・ディレクトリー
- STADDRESS 内の住所のニックネーム
- ストアの説明
- ストアのアドレス

注: ストア・サービス・ツールは、事前に読み込み済みの XML ファイルを、ストア・アーカイブ形式で処理します。

結果として、以下の 2 つの方法でストア資産を作成できます。

- WebSphere Commerce に付属のサンプル・ストアの 1 つまたは既存のストア・アーカイブの、既存のストア資産を編集する。
- XML ファイル形式でストア資産を作成する。XML ファイルは、ストア・アーカイブの一部として発行することもできますし、ローダー・パッケージを使用してロードすることもできます。

既存のストア・アーカイブのストア資産を編集する方法の詳細については、WebSphere Commerce のオンライン・ヘルプを参照してください。 XML ファイル形式でストア資産を作成することに関する情報は、 『XML ファイルによるストア・データ資産の作成』を参照してください。

XML ファイルによるストア・データ資産の作成

ストア資産を XML ファイルの形式で作成します。XML ファイルは、ローダー・パッケージを使用して、データベースにロード可能です。多文化ストアを作成する場合は、ストアでサポートされているロケールごとに別々の XML ファイルを作成することもできます。説明情報はすべてロケール固有のファイルに指定されるので、これを翻訳するのは容易です。ローダー・パッケージについて詳しくは、 225 ページの『第 7 部 ストアの発行』を参照してください。

サンプル・ストア (以下のタスクの多くの例がサンプル・ストアから取られています) では、翻訳の不要な情報はすべて 1 つの `store.xml` ファイルに指定されており、翻訳の必要な情報は、そのストアがサポートするロケールごとに別の `store.xml` ファイルに指定されています。ロケール固有のファイルには、すべての説明情報が含まれています。

ストア資産を作成するには、以下のようにします。

1. サンプル・ストアのストア資産を作成するために使用される XML ファイルを確認します。サンプル・ストアのすべてのファイルは、対応するストア・アーカイブ・ファイルの中にあります。

   ストア・アーカイブ・ファイルは、以下に示すディレクトリーにあります。

   - NT `drive:¥WebSphere¥CommerceServer¥samplestores`
   - 2000 `drive:¥Program Files¥WebSphere¥CommerceServer¥samplestores`
   - AIX `/usr/WebSphere/CommerceServer/samplestores`
   - Solaris `/opt/WebSphere/CommerceServer/samplestores`
   - Linux `/opt/WebSphere/CommerceServer/samplestores`
   - 400 `/qibm/proddata/WebCommerce/samplestores`

   注: WebSphere Commerce オンライン・ヘルプには、サンプル・ストアに含まれる各データ資産についての情報が記載されています。

   各サンプル・ストアには、`store.xml` ファイルが 2 つ組み込まれています。これにはストア情報が含まれています。ストア・アーカイブの `store.xml` ファイルを表示するには、ZIP プログラムを使用してストア・アーカイブを解凍しま

す。 store.xml ファイルは、data ディレクトリーにあります。言語ごとの store.xml は、 data ディレクトリーのロケールごとのサブディレクトリーにあります。

2. 313 ページの『付録 B. データの作成』の情報を確認します。
3. サンプル・ストア・アーカイブの store.xml ファイルの 1 つをコピーするか、新しいファイルを作成することにより、store.xml ファイルを作成します。詳しくは、store.xml に対応する DTD ファイルを参照してください。 DTD ファイルは以下のディレクトリーにあります。
   - NT drive:¥WebSphere¥CommerceServer¥xml¥sar
   - 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥xml¥sar
   - AIX /usr/WebSphere/CommerceServer/xml/sar
   - Solaris /opt/WebSphere/CommerceServer/xml/sar
   - Linux /opt/WebSphere/CommerceServer/xml/sar
   - 400 /qibm/proddata/WebCommerce/xml/sar
4. ストア・エンティティーを作成します。
   a. 次の例を参考にして、XML ファイルの STOREENT テーブルにストア・エンティティーを定義します。

```
<storeent
 storeent_id="@storeent_id_1"
 member_id="&MEMBER_ID"
 type="S"
 identifier="ToolTech"
 setccurr="USD"
/>
```

   ここで、
   - storeent_id は、生成されるユニーク鍵です。
   - member_id は、ストア・エンティティーの所有者です。
   - type は、ストア・エンティティーの種類で、 G = StoreGroup、S = Store です。
   - identifier は、所有者と共にストア・エンティティーを固有に識別するストリングです。
   - setccurr は、ストア・エンティティーのデフォルト通貨です。言い換えると、希望する通貨を持たない顧客によって使用される通貨です。ストアに関してこれが NULL の場合、デフォルト通貨はストア・グループから取得されます。
5. ストアの住所を作成します。
   a. 次の例を参考にして、 XML ファイルの STADDRESS テーブルにストアの連絡先 (複数の場合もあり) を作成します。多文化カタログを作成する場合は、ロケール固有の XML ファイルにこの情報を組み込む必要があります。

```
<staddress
staddress_id="@staddress_id_en_US_1"
member_id="&MEMBER_ID"
nickname="storeaddress_English"
address1="12xx Martindale Avenue"
address2="Suite 9xx"
```

```
businesstitle="ToolTech"
city="Toolsville"
state="Ontario"
zipcode="Lxx 1xx"
country="Canada"
phone1="1-800-555-1234"
fax1="1-800-555-4321"
email1="info@tooltech.xxx"
/>
```

ここで、

- staddress_id は、生成されるユニーク鍵です。
- member_id は、ストア・エンティティーの所有者です。

6. ストア・エンティティーの説明を作成します。
   a. 次の例を参考にして、 XML ファイルの STOREENTDS テーブルにストア・エンティティーの説明を作成します。多文化カタログを作成する場合は、ロケール固有の XML ファイルにこの情報を組み込む必要があります。

```
<storeentds
description="Commerce Models Store entity"
language_id="&en_US"
displayname="ToolTech"
storeent_id="@storeent_id_1"
staddress_id_cont="@staddress_id_en_US_1"
staddress_id_loc="@staddress_id_en_US_1"
```

ここで、

- description は、顧客に対して表示するのに適した、ストア・エンティティーの詳細説明です。
- language_id は、ストアでショッピングする顧客に対して表示される情報のためのデフォルト言語です。
- displayname は、顧客に対して表示するのに適した、ストア・エンティティーの簡略説明です。
- storeent_id は、ストア・エンティティーです。
- staddress_id_cont は、StoreEntity の連絡先住所です。
- staddress_id_loc は、StoreEntity の物理的場所です。

7. データベースにストアを作成します。
   a. 次の例を参考にして、 XML ファイルの STORE テーブルにストアを定義します。

```
<store
store_id="@storeent_id_1"
directory="ToolTech"
ffmcenter_id="@ffmcenter_id_1"
language_id="&en_US"
storegrp_id="-1"
allocationgoodfor="43200"
bopmpadfactor="0"
defaultbooffset="2592000"
ffmcselectionflags="0"
maxbooffset="7776000"
rejectedordexpiry="259200"
rtnffmctr_id="@ffmcenter_id_1"
pricerefflags="0"
storetype="B2B"
/>
```

ここで、

- store_id は、生成されるユニーク鍵です。
- directory は、ストア固有の Web 資産があるディレクトリーです。ファイル・システムにおけるこうした資産の実際の場所は、この列の値に WebSphere Commerce 構成ファイルの幾つかの構成パラメーター、StoresDocRoot、StoresWebPath、および StoresPropertiesPath の値を加えたものに基づきます。たとえば、 StoresDocRoot が D:¥WebSphere¥wcs¥stores、StoresWebPath が web、StorePropertiesPath が properties、およびこの列の値が mystore の場合、JSP ファイルはディレクトリー D:¥WebSphere¥wcs¥stores¥web¥mystore に位置し、プロパティー・ファイルは D:¥WebSphere¥wcs¥stores¥properties¥mystore に位置します。
- ffmcenter_id は、ストアのデフォルトの配送センターです。
- language_id は、ストアでショッピングする顧客に対して表示される情報のためのデフォルト言語です。
- storegrp_id は、そのストアが関連付けられているストア・グループです。この数値は、STOREGRP テーブルに生成されます。
- allocationgoodfor は、割り振りが行われてから長時間経過した場合に、ATP 在庫割り振りを取り消すために ReleaseExpiredAllocations スケジューラー・ジョブを使用できることを意味します。
- bopmpadfactor は、このストアが異なる配送センターごとに (税金または配送料のような) オーダー金額を計算する場合には、配送センターがバック・オーダー済みのアイテムに最終的に割り振られる際に、以前に送信済みのオーダーのオーダー金額を変更できることを意味します。必要であるなら、 Payment Manager に提供されるオーダー金額によるパーセントを表すこの埋め込み係数を増加させることができます。たとえば、5 を指定すると 5 % 増加できます。
- defaultbooffset は、バック・オーダーされた OrderItem の販売予定時期が判別できない場合、将来この秒数に設定されることを意味します。
- maxbooffset は、バック・オーダーされた OrderItem の販売予定時期が通常では将来この秒数を超える場合に、将来この秒数に設定されることを意味します。
- rejectedordexpiry は、支払状況が拒否状況にある時間がこの秒数を超えると、オーダーがキャンセルされることを示します。
- rtnffmctr_id は、商品をストアに返品するためのデフォルトの配送センターです。
- pricerefflags には、GetContractUnitPrices タスク・コマンドのデフォルト・インプリメンテーションによって価格が更新される際に、どの TradingAgreements およびオファーが検索されるかを制御するビット・フラグが入ります。
  - 1 = usePreviousOnly。OrderItems によって参照されるものを使用します。それらが使用できなくなると、障害が発生します。
  - 2 = usePreviousOrSearchAgain。usePreviousOnly と同じですが、それらが使用できなくなった場合、障害が発生するのではなく、ORDIOFFER および ORDITRD テーブルに保管されているものを検索します。

– 4 = alwaysSearchAgain。常に ORDIOFFER および ORDITRD テーブルに保管されているものを検索します。

- storetype は、ユーザー・インターフェースが使用する以下のいずれかのストア・タイプを示します。 StoreType に応じて、ユーザー・インターフェースは適切な機能を提供することになります。 B2B = 企業間取引。 B2C = 企業対顧客の取引。

8. ストアのサポートされる言語を定義します。
   a. 次の例を参考にして、ストアのサポートされる言語を XML ファイルに定義して、情報を STORELANG テーブルに追加します。ストアがマルチリンガル・サポートされている場合には、この情報をロケール固有の XML ファイルに含める必要があります (ストアがサポートする言語ごとに 1 つずつ)。

```
<storelang
language_id="&en_US"
storeent_id="@storeent_id_1"
/>
```

   ここで、

   - language_id は、ストア・エンティティーによってサポートされる言語です。
   - storeent_id は、ストア・エンティティーです。

   b. 次の例を参考にして、言語に関する情報を STORELANGDS テーブルに追加します。ストアがマルチリンガル・サポートされている場合には、この情報をロケール固有の XML ファイルに含める必要があります (ストアがサポートする言語ごとに 1 つずつ)。

```
<storlangds
 description="United States"
 language_id="&en_US"
 storeent_id="@storeent_id_1"
  language_id_desc="&en_US"
/>
```

   ここで、

   - description は、選択リストの中で顧客に対して表示するのに適した、言語の簡略説明です。
   - language_id は、説明の言語です。
   - storeent_id は、言語をサポートするストア・エンティティーです。
   - language_id_desc は、説明される言語です。

@ と **&** の使用法の詳細については、 313 ページの『付録 B. データの作成』を参照してください。

第 7 章 コマンド、表示、および URL レジストリー・データ

WebSphere Commerce Server インスタンスを作成すると、WebSphere Commerce に付属するデフォルトのコマンド、ビュー、および URL が、WebSphere Commerce Server データベースの対応するテーブル (CMDREG、VIEWREG、および URLREG) に登録されます。これらのコマンド、ビュー、および URL は、インスタンス内に存在するすべてのストアで使用することができます。

WebSphere Commerce は、デフォルト・ビューを表示するデフォルト JSP ファイルも提供します。これらの JSP ファイルは、VIEWREG テーブルのビューと関連しています。

新しいコマンド、ビュー、または URL を作成する場合、または既存のものをカスタマイズする場合は、対応するデータベース・テーブル (CMDREG、VIEWREG、および URLREG) にそれらを登録しないとストアで使用可能になりません。自分のストアで使用するために新しい JSP ファイルを作成する場合は、それらのファイルを VIEWREG テーブルの中の対応するビューと関連付ける必要があります。

注: 新しい JSP ファイルを作成するものの、それにビューと関連したデフォルトの JSP ファイルと同じ名前を付ける場合は、新しい JSP ファイルを VIEWREG テーブルに登録する必要はありません。

コマンド、ビュー、または URL に関する詳細は、 *IBM WebSphere Commerce プログラマーズ・ガイド* を参照してください。*プログラマーズ・ガイド* には、コマンド、ビュー、URL、および JSP ファイルを登録する方法と、いつ登録するべきかに関する情報も含まれています。

WebSphere Commerce Server のコマンドおよびビュー資産の構造の詳細については、 WebSphere Commerce オンライン・ヘルプでコマンドおよびビュー・データ・モデルを参照してください。

WebSphere Commerce でのコマンド、ビュー、および URL の登録

ストアのために複数の新しいコマンド、ビュー、URL、または JSP ファイルを作成あるいはカスタマイズしたなら、それを登録するために XML ファイルを使用することもできますし (XML ファイルは、後でローダー・パッケージを使用してデータベースにロードできます)、それをストア・アーカイブの一部として登録することもできます。ストア・アーカイブは、ストア・サービスを使って発行できます。ローダー・パッケージについて詳しくは、 225 ページの『第 7 部 ストアの発行』を参照してください。

注: 新規またはカスタマイズ済みコマンドをロードするための XML ファイルを作成する前に、*プログラマーズ・ガイド* に載せられているコマンドの動作に関する詳細情報を参照してください。

コマンド、ビュー、および URL の登録用 XML ファイルの作成

ストア用の新しいコマンド、ビュー、および JSP ファイルを登録するための XML ファイルを作成するには、次の手順に従います。

1. サンプル・ストア用のコマンド、ビュー、JSP ファイルを登録するのに使用される XML ファイルを確認します。各サンプル・ストアには、command.xml ファイルが入っており、このファイルには登録情報が入っています。ストア・アーカイブ・ファイルは以下のディレクトリーにあります。

   - NT drive:¥WebSphere¥CommerceServer¥samplestores
   - 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥samplestores
   - AIX /usr/WebSphere/CommerceServer/samplestores
   - Solaris /opt/WebSphere/CommerceServer/samplestores
   - Linux /opt/WebSphere/CommerceServer/samplestores
   - 400 /qibm/proddata/WebCommerce/samplestores

   注: WebSphere Commerce オンライン・ヘルプには、サンプル・ストアに含まれる各データ資産についての情報が記載されています。

   ストア・アーカイブの内容を表示するには、解凍プログラムを使います。command.xml ファイルは、データ・ディレクトリーにあります。
2. 313 ページの『付録 B. データの作成』の情報を確認します。
3. サンプル・ストア・アーカイブの command.xml ファイルの 1 つをコピーするか、新しいファイルを作成することによって、command.xml ファイルを作成します。詳しくは、command.xml に対応する DTD ファイルを参照してください。DTD ファイルは以下のディレクトリーにあります。

   - NT drive:¥WebSphere¥CommerceServer¥xml¥sar
   - 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥xml¥sar
   - AIX /usr/WebSphere/CommerceServer/xml/sar
   - Solaris /opt/WebSphere/CommerceServer/xml/sar
   - Linux /opt/WebSphere/CommerceServer/xml/sar
   - 400 /qibm/proddata/WebCommerce/xml/sar
4. コントローラー・コマンドは、URLREG テーブルと CMDREG テーブルに登録する必要があります。 URLREG テーブルに新規またはカスタマイズ済みのコントローラー・コマンドを登録するには、次の例を指針として、新しいカスタマイズ済みのコントローラー・コマンドごとに、XML ファイルにエントリーを作成します。

```
<urlreg
url="MyProductDisplay"
storeent_id="@storeent_id_1″
interfacename="com.mystore.commerce.catalog.commands.ProductDisplayCmd"
https="0"
description="Product display command for my store"
```

```
authenticated="0"
internal="0" />
```

ここで、

- urlreg は、この情報を読み込むデータベース・テーブル (URLREG) の名前です。
- url は URI 名です。
- storeent_id はストア・エンティティー ID です。@ 記号を使用するのは、内部別名解決法と呼ばれます。内部別名解決法を使用する場合は、 XML 文書内で 1 次鍵 (ID) の代わりに別名が用いられます。これで別名は、そのエレメントを参照するために、XML ファイル内の他の場所で使用できます。したがって、XML ファイルを構築するのに必要な固有索引を知っている必要はありません。発行中、ID リゾルバーは @ 記号を固有な値に置き換えます。詳細については、 313 ページの『付録 B. データの作成』を参照してください。
- interfacename はコントローラー・コマンド・インターフェース名です。
- https はこの URL 要求に必要なセキュア HTTP です。セキュア HTTP が必要な場合は 1、必要ない場合は 0 を使用します。
- authenticated は、この URL 要求にユーザー・ログオンが必要かどうかを示します。認証が必要な場合は 1、必要ない場合は 0 を使用します。
- internal は、コマンドが WebSphere Commerce にとって内部的なものかどうかを示します。内部 URL は、WebSphere Commerce のツールで使用されます。内部的なものである場合は 1、外部的なものである場合は 0 を使用します。自分で作成する URL は外部的なものでなければなりません。

5. CMDREG テーブルに新規のコントローラー・コマンド、または新規のタスク・コマンドを登録するには、次のタスク・コマンドの例 (ToolTech サンプル・ストア command.xml ファイルからの例) を参考にして、新規の、またはカスタマイズ済みのコントローラーまたはタスク・コマンドごとに、XML ファイルにエントリーを作成します。

```
< cmdreg
storeent_id="@storeent_id_1"
interfacename="com.ibm.commerce.payment.commands.DoPaymentCmd"
classname="com.ibm.commerce.payment.commands.DoPaymentMPFCmdImpl"/>
```

ここで、

- cmdreg は、この情報を読み込むデータベース・テーブル (CMDREG) の名前です。
- storeent_id はストア・エンティティー ID です。@ 記号を使用するのは、内部別名解決法と呼ばれます。内部別名解決法を使用する場合は、 XML 文書内で 1 次鍵 (ID) の代わりに別名が用いられます。これで別名は、そのエレメントを参照するために、XML ファイル内の他の場所で使用できます。したがって、XML ファイルを構築するのに必要な固有索引を知っている必要はありません。発行中、ID リゾルバーは @ 記号を固有な値に置き換えます。詳細については、 313 ページの『付録 B. データの作成』を参照してください。
- interfacename はコマンド・インターフェース名です。

- classname はコマンド・インプリメンテーション・クラス名です。通常、この名前は、インターフェース名の末尾に Impl を付加したものとなっています。

6. 新しいビューを登録する場合、または新しい JSP ファイルをビューに関連付ける場合は、次の例 (ToolTech サンプル・ストア command.xml ファイルからの例) を参考にして、VIEWREG テーブルにエントリーを作成します。

```
<viewreg
viewname="OrderOptionsView"
devicefmt_id="-1"
storeent_id="@storeent_id_1"
interfacename="com.ibm.commerce.command.ForwardViewCommand"
classname="com.ibm.commerce.command.HttpForwardViewCommandImpl"
properties="docname=Shipping.jsp"
internal="0"
https="0"/>
```

ここで、

- viewreg は、この情報を読み込むデータベース・テーブル (VIEWREG) の名前です。
- viewname はビューの名前です。
- devicefmt_id は、のビューが使用されるデバイスのタイプ (ブラウザーなど) です。
- storeent_id はストア・エンティティー ID です。@ 記号を使用するのは、内部別名解決法と呼ばれます。内部別名解決法を使用する場合は、 XML 文書内で 1 次鍵 (ID) の代わりに別名が用いられます。これで別名は、そのエレメントを参照するために、XML ファイル内の他の場所で使用できます。したがって、XML ファイルを構築するのに必要な固有索引を知っている必要はありません。発行中、ID リゾルバーは @ 記号を固有な値に置き換えます。詳細については、 313 ページの『付録 B. データの作成』を参照してください。
- interfacename はビュー・コマンド・インターフェース名です。デフォルト・オプションは ForwardView、DirectView、および RedirectView です。
- classname はビュー・インプリメンテーション・クラス名です。通常、この名前は、インターフェース名の末尾に Impl を付加したものとなっています。
- properties はコマンドへの入力プロパティーとして設定される、デフォルトの名前と値の対です。同じページが常に表示される場合は、このプロパティーに docname=Shipping.jsp などと JSP ファイル名を設定します。
- internal は、ビューが WebSphere Commerce にとって内部的なものかどうかを示します。内部ビューは、WebSphere Commerce のツールによって使用されます。内部的なものである場合は 1、外部的なものである場合は 0 を使用します。自分で作成するビューは外部的なものでなければなりません。
- https はこの URL 要求に必要なセキュア HTTP です。セキュア HTTP が必要な場合は 1、必要ない場合は 0 を使用します。

@ と **&** の使用法の詳細については、 313 ページの『付録 B. データの作成』を参照してください。

第 8 章 カタログ資産

従来のカタログと同様、オンライン・カタログも、販売する商品やサービスから構成されています。オンライン・カタログのサイズや構造は、購入用の商品取引のタイプや金額によって、ストアごとにかなり異なることもありますが、カタログに必要なものは次のとおりです。

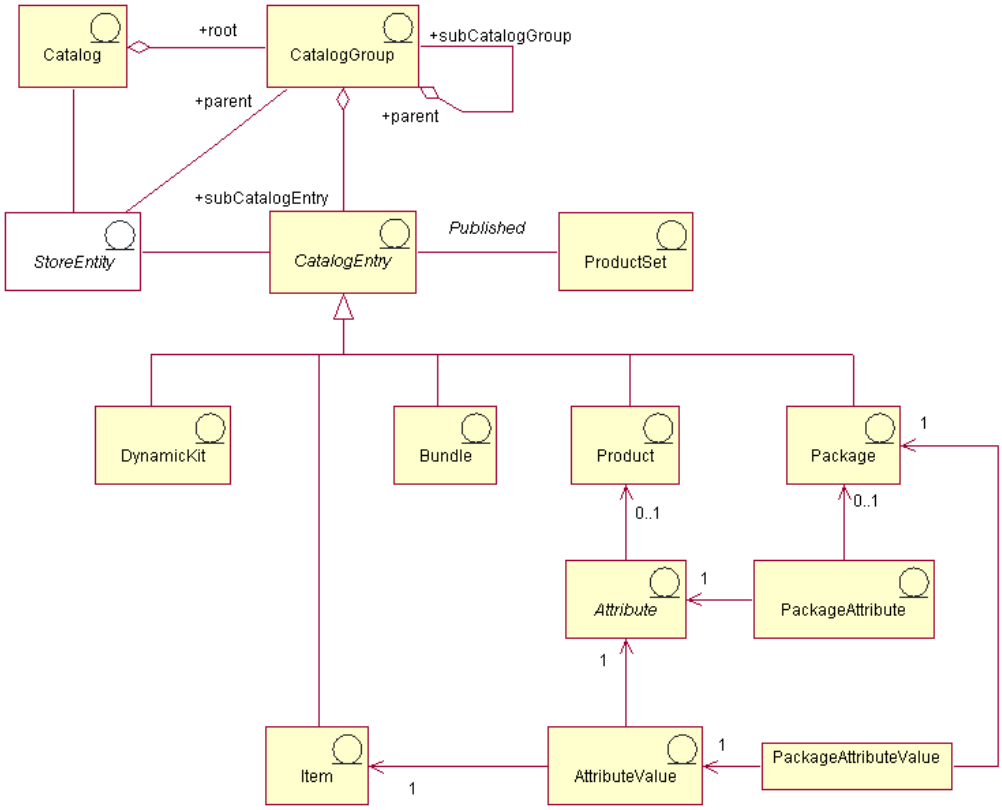
- 販売するもの。次のものが含まれます。
  - 販売価格。必ずと言ってよいほどオンライン・カタログに掲載されます。
  - 商品データ。商品取引の詳細やイメージなど。
  - カテゴリー。大部分のカタログ (ただしすべてではない) が、顧客のナビゲーションの便宜を図って商品取引をカテゴリーに分類しているのと同様です。
- 商品の表示方法。カタログ表示ページは、ページが顧客にどのように表示されるかの概要を決定し、さまざまなカタログ・ページのルック・アンド・フィールを統一の取れたものとします。カタログの構造は、扱う商品取引によって異なります。

WebSphere Commerce のカタログについて

WebSphere Commerce では、ストアのオンライン・カタログにいくつかの要件が課されています。 WebSphere Commerce システムの全ストアに、マスター・カタログがなければなりません。マスター・カタログは、ストアの商品取引を管理する中心です。これは、すべての商品、アイテム、関係、およびストアで販売されるものすべての標準価格を含む 1 つのカタログです。

マスター・カタログは複数のストアで共有することができ、また必要な数のストアを定義できます。カタログ管理用のマスター・カタログを作成することに加えて、表示の目的で 1 つ以上のナビゲーション・カタログを作成することもできます。ナビゲーション・カタログにはマスター・カタログと同じエントリーを含めることができますが、カスタマーに表示する目的で、ナビゲーション・カタログはマスター・カタログよりずっと柔軟な構造になっています。必要に応じていくつでもナビゲーション・カタログを作成することができます。しかし、オンラインの商品取引を管理するためにマスター・カタログを使用するので、マスター・カタログをナビゲーション・カタログとして使用して、メインテナンスのオーバーヘッドを最小限に抑えるようお勧めします。

WebSphere Commerce ストアで使用するために新しいマスター・カタログを作成する場合、または ToolTech などの WebSphere Commerce サンプル・ストアで使用可能な既存のマスター・カタログを使用する場合は、自分のカタログをこれらの要件に合うように変更する必要があります。次の図は、WebSphere Commerce におけるカタログの基本構造の概観を示しています。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce 情報モデルの一部です。情報モデルの詳細については、29 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則については、311 ページの『付録 A. UML の凡例』を参照してください。

カタログ

カタログ は情報モデルの出発点です。カタログには、オンライン・カタログ用のすべての階層情報およびナビゲーション情報が入っています。カタログは、オンライン・ストアで表示および購入可能な、カタログ・グループとカタログ・エントリーの集合です。

WebSphere Commerce では、カタログは、データベースにおいて、*カタログ・エンティティー* で表されています。カタログ・エンティティーは、固有のカタログ ID とカタログの説明 (カタログ名など) から構成されています。各カタログは個別かつ固有のエンティティーであるため、簡単に 1 つ以上のストアと関連付けることができます。 WebSphere Commerce システムのストアはどれも、最低 1 つのカタログ・エンティティーと関連付けなければなりません。マスター・カタログは、オンライン・ストアで販売されているすべてのアイテムを含む、特殊なカタログです。

カタログ・グループ

カタログ・グループ とは、ナビゲーション上の目的およびカタログの区分化の目的で作成される、カタログ・エントリーの一般的なグループ分けのことです。 1 つのカタログ・グループは 1 つのカタログに所属し、複数のカタログ・グループまたは

カタログ・エントリーを含むことができます。カタログ・グループは複数のカタログと関連付けることができます。カタログ・グループはカテゴリー とも呼ばれます。

フラット・カタログは、商品がカテゴリーごとにグループ分けされず、代わりに商品のリストが表示されるカタログです。 WebSphere Commerce ではフラット・カタログを作成することも可能ですが、構造上およびナビゲーション上の理由から、カタログ・グループを作成することをお勧めします。

カタログ・グループを作成するときには、まずカタログを階層、つまり逆さまにした木の形に整理する必要があります。ツリーは、一般カタログ・グループ (ルート・カテゴリーと呼ばれる) から始まり、特定のサブカテゴリーへと徐々に分岐していきます。これは、それ以上分割できなくなるまで続きます。商品しか入っていない最低レベルのカタログ・グループは、リーフと呼ばれます。カタログ・グループは、そのすぐ下のカテゴリーに対しては親、 1 つ上のカテゴリーに対しては子になります。たとえば、メンズ・ファッションは紳士服のカテゴリーのグループであり、パンツやシャツといったカタログ・グループは商品のグループです。

カタログ・エントリー

各カタログ・グループには*カタログ・エントリー* が入っています。カタログ・エントリーとは、オンライン・カタログでオーダー可能な商品のことです。エントリーには通常、部品番号、説明、1 つ以上のオファー価格、イメージ、およびその他詳細情報があります。カタログ・エントリーは、商品、アイテム、パッケージ、バンドル、またはダイナミック・キットのいずれかになります。必要なら、5 つの既存のモデルのうちいずれにもあてはまらない、新しいカタログ・エントリー・タイプを作成することもできます。カタログ・エントリーの各タイプの詳細については、以下で説明します。

商品

商品 は、カタログ・エントリーの 1 タイプです。商品は、同じ属性を表すアイテムまたは SKU のグループのテンプレートとなります。たとえば、シャツはカタログに記載されている商品です。シャツに属性と属性値を追加すると、それぞれのバリエーションが 1 つのアイテムとなります。たとえば、S サイズの黒のシャツなどです。

アイテム

アイテム は、特定の名前、部品番号、および価格を持つ具体的な商品取引の単位です。たとえば、S サイズの黒のシャツはアイテムですが、シャツは商品です。特定の商品に関連するすべてのアイテムは同じ属性セットを示し、それらの属性値によって区別されます。

注: WebSphere Commerce アクセラレーターのユーザーにとって、アイテムと *SKU* は同義語です。 WebSphere Commerce アクセラレーターの商品管理ツールを使用する場合、オーダー可能なアイテムを SKU と呼びます。 WebSphere Commerce データベース・スキーマでは、この特定のタイプのカタログ・エントリーをアイテムと呼びます。

パッケージ

パッケージ は、カタログ・エントリーの不可分のコレクションです。たとえば、コンピューター・パッケージには、別売りできない特定の中央演算処理装置、モニター、およびハード・ディスクが含まれます。商品と同様、パッケージには定義されている属性があり、パッケージは完全解体パッケージのコンテナーにあたります。完全解体パッケージは、SKU に相当します。パッケージはそれ自身の価格を持った実際にオーダー可能な SKU であり、ショッピング・カートに追加できます。パッケージは、ナビゲーション中もショッピング・カート内に置かれた後も分解、変更することはできません。

バンドル

バンドル は、カタログ・エントリーのコレクションで、これにより顧客は 1 度に複数のアイテムを購入することができます。たとえば、コンピューターのバンドルは中央演算処理装置、モニター、ハード・ディスク、および CD-ROM ドライブなどで構成されています。バンドルにできるのは、アイテムのグループ、あるいは商品、アイテム、および完全解体パッケージの組み合わせです。アイテムだけを含むバンドルを選択する場合、バンドルは個別にオーダーできる SKU に分解され、これは個別にショッピング・カートに追加されます。ただし、商品を含むバンドルを選択する場合、これらの商品を SKU 分解によってアイテムに解体してからショッピング・カートに追加する必要があります。いずれの場合でも、いったんバンドルが分解されてそのコンポーネント・アイテムがショッピング・カートに追加されると、各アイテムを変更または除去できます。

ダイナミック・キット

ダイナミック・キット は、顧客が動的に構成できるカタログ・エントリーのタイプです。この商品の構成 (またはグループ分け) は顧客の要件に基づき、1 単位で販売されます。ダイナミック・キットのコンポーネントは、一連の定義済み規則とユーザーの対話を介した外部商品コンフィギュレーターで構成されます。オーダーにダイナミック・キットを追加することは、パッケージを追加することに似ています。パッケージと同じように、ダイナミック・キットの個々のコンポーネントは変更できず、構成は、全体として行う必要があります。しかし、ダイナミック・キット・コンポーネントは、外部商品コンフィギュレーターを使用して再構成を行うことによって変更することができます。

商品セット

商品セット は、発行済みの CatalogEntry オブジェクトと関連付けられます。商品セットは、カタログを論理サブセットに区分化するためのメカニズムを提供します。この区分化によって、様々なユーザーにカタログの異なる部分を示すことができます。契約を作成して、契約の参加者だけが、事前定義された商品セットに当たる商品を購入できるように指定できます。 WebSphere Commerce は、マスター・カタログから商品セットを作成し、それらを利用して契約で権利フィルターを行うためのツールを提供します。

属性

属性 は、カラーやサイズなどの、オンライン・ストアの商品のプロパティーです。属性は商品に属しています。属性と属性値との可能な組み合わせが、それぞれ 1 つのアイテムを定義します。

属性値

属性値 は、属性のプロパティーです。たとえば、特定のカラー (青または黄) やサイズ (S、M、L) などです。属性値をアイテムに割り当てる前に、それを定義しておく必要があります。属性と属性値との可能な組み合わせが、それぞれ 1 つのアイテムを定義します。

パッケージの属性

パッケージの属性 は、パッケージに含まれている商品の属性から継承されます。パッケージには 0 個以上の属性を指定できます。 1 つのパッケージ属性は、1 つの属性を参照します。

パッケージの属性値

パッケージの属性値 は、パッケージの属性に割り当てられた値です。パッケージの属性は、パッケージに含まれている商品の属性値から継承されます。 1 つのパッケージ属性値は、1 つの属性値を参照します。

WebSphere Commerce でのカタログ資産の構造に関する詳細は、WebSphere Commerce オンライン・ヘルプの『カタログ・データ・モデル』を参照してください。

WebSphere Commerce でのカタログ資産の作成

ストア用にカタログ資産を作成するには、複数の WebSphere Commerce データベース・テーブルに情報を追加することによって、マスター・カタログを作成する必要があります。カタログは XML ファイルを使って作成できます。XML ファイルは、ローダー・パッケージを使ってデータベースにロードされます。多文化カタログを作成する場合、ストアがサポートする各ロケールごとに個別の XML ファイルが必要です。それぞれの XML ファイルでは、カタログ、カタログ・グループ、およびカタログ・エントリーについての翻訳可能な情報 (説明など) が追加されます。

マスター・カタログの作成

複数のカテゴリー・レベルを含むマスター・カタログを作成する場合は、次のタスクを完了します。

パート 1: カタログ作成の準備

1. カタログ情報と、それに対応する WebSphere Commerce のオブジェクト・モデルとデータ・モデルを確認します。カタログ情報は WebSphere Commerce サーバーのコンポーネントの 1 つであり、オーダー可能な商品取引のために、オンライン・カタログのナビゲーション、区分化、カテゴリー化、および関連を提供します。

2. WebSphere Commerce ローダー・パッケージ情報を確認します。ローダー・パッケージは、 WebSphere Commerce データを準備してロードするユーティリティーで構成されます。ローダー・パッケージを使用して、データベースに大量のデータをロードしたり、データを更新したりすることができます。ローダー・パッケージについて詳しくは、 225 ページの『第 7 部 ストアの発行』を参照してください。
3. 313 ページの『付録 B. データの作成』の情報を確認します。
4. カタログ所有者とするため、管理コンソールで組織を作成します。詳しくは、WebSphere Commerce オンライン・ヘルプのトピック『組織の作成』を参照してください。
5. 既存の XML エントリー、および ToolTech サンプル・ストアの `catalog.xml` ファイルを参考にして、マスター・カタログ用の新しい XML ファイルを作成します。多文化カタログを作成する場合、ストアがサポートする各ロケールごとに個別の `catalog.xml` ファイルを作成します。説明情報はすべてロケール固有のファイルに指定されるので、これを翻訳するのは容易です。この例では、翻訳の不要なすべての情報のために 1 つの `catalog.xml` ファイルが使用され、さらにストアがサポートするロケールごとに第 2 の `catalog.xml` が使用されて翻訳の必要な情報が入れられます。または、ToolTech サンプル・ストアから既存の XML ファイルを使用して、必要に応じて情報を変更することもできます。ToolTech の `catalog.xml` ファイルは、ストア・アーカイブ・ファイルにあります。 `catalog.xml` ファイルを表示するには、ZIP プログラムを使用して、ストア・アーカイブを解凍します。 `catalog.xml` ファイルは以下の data ディレクトリーにあります。

   - NT *drive*:¥WebSphere¥CommerceServer¥samplestores
   - 2000 *drive*:¥Program Files¥WebSphere¥CommerceServer¥samplestores
   - AIX /usr/WebSphere/CommerceServer/samplestores
   - Solaris Linux /opt/WebSphere/CommerceServer/samplestores
   - 400 /qibm/proddata/WebCommerce/samplestores

   注: WebSphere Commerce オンライン・ヘルプには、サンプル・ストアに含まれる各データ資産についての情報が記載されています。

   `catalog.dtd` ファイルは以下のディレクトリーにあります。

   - NT *drive*:¥WebSphere¥CommerceServer¥xml¥sar
   - 2000 *drive*:¥Program Files¥WebSphere¥CommerceServer¥xml¥sar
   - AIX /usr/WebSphere/CommerceServer/xml/sar
   - Solaris Linux /opt/WebSphere/CommerceServer/xml/sar
   - 400 /qibm/proddata/WebCommerce/xml/sar

パート 2: カタログ・エンティティーの作成

1. ToolTech サンプル・ストアの次の例を参考にして、CATALOG および CATALOGDSC テーブルに情報を追加することによって、カタログ・エンティティーを作成します。カタログ・エンティティーはデータベースの中のカタログを表します。

```
<catalog
catalog_id="@catalog_id_1"
member_id="&MEMBER_ID;"
identifier="ToolTech"
description="ToolTech Catalog"
tpclevel="0"
/>
```

   ここで、

   - catalog_id は、内部参照番号です。
   - member_id は、カタログの所有者を識別する内部参照番号です。
   - identifer は、カタログの外部名です。
   - description は、カタログの説明です。

2. ToolTech サンプル・ストアの次の例を参考にして、後で翻訳することを意図したロケール固有の XML ファイルにカタログの説明を追加します。

```
<catalogdsc
catalog_id="@catalog_id_1"
language_id="&en_US;"
name="Store master catalog"
/>
```

   ここで、

   - catalog_id は、この言語特定情報をカタログに関連付ける内部参照番号です。
   - language_id は、言語の ID です。
   - name は、言語に依存するカタログの名前です。

パート 3: カタログ・グループの作成

1. ToolTech サンプル・ストアの次の例を参考にして、CATGROUP および CATGRPDESC テーブルに情報を追加することによって、カタログ・グループを作成します。カタログ・グループ (カテゴリーとも呼ばれる) は、他のカタログ・グループまたは製品をグループにしたものです。カタログの各カタログ・グループごとにこのタスクを完了します。

```
<catgroup
catgroup_id="@catgroup_id_1"
member_id="&MEMBER_ID;"
identifier="Woodworking"
markfordelete="0"
/>
```

   ここで、

   - catgroup_id は、カタログ・グループの内部参照番号です。
   - member_id は、カタログの所有者を識別する内部参照番号です。
   - identifer は、カタログの外部名です。

- markfordelete は、カタログ・グループが削除の対象としてマークされているかどうかを示します。
  - 0 = No
  - 1 = Yes

2. ToolTech サンプル・ストアの次の例を参考にして、後で翻訳することを意図したロケール固有の XML ファイルにカタログ・グループの説明を追加します。カタログの各カタログ・グループごとにこのタスクを完了します。

```
<catgrpdesc
language_id="&en_US;"
catgroup_id="@catgroup_id_1"
name="Woodworking"
shortdescription="Woodworking"
longdescription="Woodworking"
published="1"
/>
```

ここで、

- language_id は、言語の ID です。
- catgroup_id は、カタログ・グループの内部参照番号です。
- name は、言語に依存するカタログの名前です。
- shortdescription は、カタログ・グループの要旨です。
- longdescription は、カタログ・グループの詳細記述です。
- published は、このカタログ・グループを、 language_id によって示された言語で表示するかどうかを指示します。
  - 0 = No
  - 1 = Yes

注: カタログ・グループとその説明を作成するたびに、新しいカタログ・グループを表すよう、catgroup_id が変わります。たとえば、catgroup_id="@catgroup_id_2"、catgroup_id="@catgroup_id_3"、および catgroup_id="@catgroup_id_4"、などです。

3. カタログ・グループを作成したら、CATTOGRP テーブルに情報を追加することによって、最上位のカタログ・グループをカタログに割り当てます。このカタログ・グループは、そのすぐ下のカタログ・グループの親になります。カタログの中の各最上位カタログ・グループごとにこのタスクを完了します。 ToolTech サンプル・ストアの次の例を参考にしてください。

```
<cattogrp
catalog_id="@catalog_id_1"
catgroup_id="@catgroup_id_1"
/>
```

ここで、

- catalog_id は、カタログの参照番号です。
- catgroup_id は、カタログ・グループの参照番号です。

注: 最上位のカタログ・グループをカタログに割り当てるたびに、新しいカタログ・グループの関連を表すよう、catgroup_id が変更されます。たとえば、catgroup_id="@catgroup_id_2"、catgroup_id="@catgroup_id_3"、および catgroup_id="@catgroup_id_4"、などです。

4. カタログ・グループの親と子の構造が決定されたら、CATGRPREL テーブルに情報を追加することによって、カタログ・グループ間の関係を作成します。カタログの中の、親子のカタログ・グループ構造のそれぞれについて、このタスクを完了します。 ToolTech サンプル・ストアの次の例を参考にしてください。

```
<catgrprel
catgroup_id_parent="@catgroup_id_1"
catgroup_id_child="@catgroup_id_11"
catalog_id="@catalog_id_1"
sequence="0"
/>
```

ここで、

- catgroup_id_parent は、この関係のソース・カタログ・グループです。
- catgroup_id_child は、この関係のターゲット・カタログ・グループです。
- catalog_id は、カタログの参照番号です。
- sequence は、カタログ・グループの内容の表示順序を決定する番号です。

注: 各カタログ・グループの関係ごとに、新しい関係を表すよう、catgroup_id_child と sequence が変更されます。たとえば、後に続く関係は、 catgroup_id_child="@catgroup_id_12" と sequence="1"、catgroup_id_child="@catgroup_id_13" と sequence="2" (以下同様) などと表示されるかもしれません。カタログの中で誘導型構造を使用していない場合は、CATGRPREL 関係を除去できます。

パート 4: 在庫情報の作成

1. ToolTech サンプル・ストアの次の例を参考にして、BASEITEM、BASEITEMDSC、ITEMSPC、ITEMVERSN、VERSIONSPC、DISTARRANG および STOREITEM テーブルに情報を追加することによって、在庫情報を作成します。最初に、BASEITEM テーブルに情報を追加することにより、基本アイテムを作成します。基本アイテムとは、名前と説明を共有する、商品の一般的なファミリーを表します。カタログの中の在庫アイテムのグループごとにこのタスクを完了します。

```
<baseitem
baseitem_id="@baseitem_id_102"
member_id="&MEMBER_ID;"
markfordelete="0"
partnumber="tooltech_sku_102"
itemtype_id="ITEM"
quantitymeasure="C62"
quantitymultiple="1.0"
/>
```

ここで、

- baseitem_id は、生成されるユニーク鍵です。
- member_id は基本アイテムの所有者です。
- markfordelete は、基本アイテムが削除の対象としてマークされているかどうかを示します。
  - 0 = No
  - 1 = Yes
- partnumber は、基本アイテムを所有者ごとに固有に識別します。

- itemtype_id は、基本アイテムのタイプです。
  - ITEM = アイテム、パッケージ、またはバンドル
  - DNKT = ダイナミック・キット
- quantitymeasure は、数量の倍数の計測単位です。
- quantitymultiple は、整数単位で計測される、基本アイテムの量です。quantitymeasure と共に、各整数単位がどれだけの量を表すかを示します。

注: カタログの中に作成するあらゆる商品に対して、基本アイテムを作成する必要があります。基本アイテムを作成するたびに、baseitem_id と partnumber の番号が変わって、新しい基本アイテムが作成されます。たとえば、ある新しい基本アイテムのエントリーは baseitem_id="@baseitem_id_147" と partnumber="tooltech_sku_147" になり、次の基本アイテムのエントリーは baseitem_id="@baseitem_id_192" と partnumber="tooltech_sku_192" になるかもしれません。

2. ToolTech サンプル・ストアの次の例を参考にして、指定されたアイテムについての情報をデータベースに追加します。指定されたアイテムはすべての属性に値が指定されているアイテムであり、カタログ内のアイテム、パッケージ、バンドル、またはダイナミック・キットを表します。カタログの中の指定されたアイテムごとに、このタスクを完了します。

```
<itemspc
itemspc_id="@itemspc_id_106"
baseitem_id="@baseitem_id_102"
markfordelete="0"
partnumber="T0000106"
member_id="&MEMBER_ID;"
discontinued="N"
/>
```

ここで、

- itemspc_id は、生成されるユニーク鍵です。
- baseitem_id は、商品基本アイテムです。
- markfordelete は、指定されたアイテムが削除の対象としてマークされているかどうかを示します。
  - 0 = No
  - 1 = Yes
- partnumber は、指定されたアイテムを所有者ごとに固有に識別します。
- member_id は、指定されたアイテムの所有者です。
- discontinued は、指定されたアイテムが製造中止になったかどうかを示します。
  - Y = 製造中止されました。十分な在庫がある場合はオーダー可能ですが、バックオーダーはできません。
  - N = アクティブです。在庫切れの場合はバックオーダーできます。

注: カタログの中に作成するアイテムごとに、指定されたアイテムを作成する必要があります。指定されたアイテムを定義するたびに、itemspc_id="@itemspc_id_107"、baseitem_id="@baseitem_id_102"、partnumber="T0000107" の各番号が変わり、指定された新しいアイテムが作成されます。たとえば、指定された新しいアイテムのエントリーは

itemspc_id="@itemspc_id_108"、baseitem_id="@baseitem_id_102"、および partnumber="T0000108" になり、指定された次のアイテムのエントリーは itemspc_id、baseitem_id、および partnumber などとなるかもしれません (以下同様)。

3. ToolTech サンプル・ストアの次の例を参考にして、あるアイテム・バージョンと基本アイテムとの関係に関する次の情報をデータベースに追加します。カタログの中のこのような関係ごとに、このタスクを完了します。

```
<itemversn
itemversn_id="@itemversn_id_102"
baseitem_id="@baseitem_id_102"
expirationdate="2010-01-01 00:00:00.000000"
versionname="version"
/>
```

ここで、

- itemversn_id は、アイテム・バージョンを識別するために生成された参照番号です。
- baseitem_id は、基本アイテムです。
- expirationdate は、アイテム・バージョンの有効期限です。
- versionname は、基本アイテムのアイテム・バージョンを固有に識別します。

注: アイテム・バージョンと基本アイテムとの間の関係を作成するたびに、itemversn_id と baseitem の番号が変わって、新しい関係が作成されます。 baseitem_id は、既存の基本アイテムに一致します。たとえば、新しい関係のエントリーは itemversn_id="@itemversn_id_107" と baseitem_id="@baseitem_id_107" になり、次の関係のエントリーは itemversn_id="@itemversn_id_108" と baseitem_id="@baseitem_id_108" などとなるかもしれません (以下同様)。

4. ToolTech サンプル・ストアの次の例を参考にして、商品バージョンと指定されたアイテムの関係に関する次の情報をデータベースに追加します。カタログの中のこのような関係ごとに、このタスクを完了します。

```
<versionspc
versionspc_id="@versionspc_id_106"
itemspc_id="@itemspc_id_106"
itemversn_id="@itemversn_id_102"
/>
```

ここで、

- versionspc_id は、生成される固有 ID です。
- itemspc_id は、カタログ・エントリーと関連のある指定されたアイテムです。
- itemversn_id は、アイテムのバージョンを示します。

注: 商品バージョンと指定済みアイテムの関係を作成するたびに、versionspc_id と itemspc_id の番号が変わって、新しい基本アイテムが作成されます。 itemspc_id は、指定された既存のアイテムに一致します。たとえば、新しい関係のエントリーは versionspc_id="@versionspc_id_107" と itemspc_id="@itemspc_id_107" になり、次の関係のエントリーは versionspc_id="@versionspc_id_108" と itemspc_id="@itemspc_id_108" などとなるかもしれません (以下同様)。

5. ToolTech サンプル・ストアの次の例を参考にして、配送手配をデータベースに追加します。分散配置によって、ストアは独自の在庫を販売できるようになります。カタログの中のこのような配送手配ごとに、このタスクを完了します。

```
<distarrang
distarrang_id="@distarrang_id_102"
wholesalestore_id="@storeent_id_1"
merchantstore_id="@storeent_id_1"
baseitem_id="@baseitem_id_102"
pickingmethod="F"
startdate="2000-12-25 00:00:00.000000"
enddate="2010-01-01 00:00:00.000000"
/>
```

ここで、

- distarrang_id は、配送手配の参照番号です。
- wholesalestore_id は、マーチャント・ストアで販売可能な在庫を所有する卸売ストアです。この卸売ストアは merchantstore_id と同じでなければなりません。
- merchantstore_id は、卸売ストアの在庫から販売できるマーチャント・ストアです。このマーチャント・ストアは wholesalestore_id と同じでなければなりません。
- baseitem_id は、配送手配の対象となる商品です。
- pickingmethod は、この配置により RECEIPT テーブルから在庫がピッキングされる順序を決定します。
  - F = FIFO (先入れ先出し) - 古いものから順に在庫から取り出します。
  - L = LIFO (後入れ先出し) - 新しいものから順に在庫から取り出します。
- startdate は、この配送手配の開始が有効になる時刻です。
- enddate は、この配送手配の停止が有効になる時刻です。

注: 配送手配を作成するたびに、 distarrang_id と baseitem の番号が変わって、新しい配送手配が作成されます。たとえば、第 2 の配送手配には distarrang_id="@distarrang_id_147" と baseitem_id="@baseitem_id_147" が含まれ、第 3 の配送手配には distarrang_id="@distarrang_id_192" と baseitem_id="@baseitem_id_192" が含まれる、というようになります。

6. ToolTech サンプル・ストアの次の例を参考にして、特定のストアが特定の基本アイテムのうち指定されたアイテムの在庫をデータベースに割り振る方法に影響する属性を追加します。カタログの中のこのような基本アイテムごとに、このタスクを完了します。

```
<storeitem
baseitem_id="@baseitem_id_102"
storeent_id="@storeent_id_1"
trackinventory="Y"
forcebackorder="N"
releaseseparately="N"
returnnotdesired="N"
backorderable="Y"
creditable="Y"
minqtyforsplit="0"
/>
```

ここで、

- baseitem_id は、基本アイテムです。
- storeent_id は、ストアまたはストア・グループです。
- trackinventory は、 RECEIPT テーブルの中で在庫をトラッキングするかどうかを指定します。
  - N = 在庫をトラッキングせず、 RECEIPT テーブルにはエントリーがありません。
  - Y = RECEIPT テーブルの中で在庫をトラッキングします。
- forcebackorder は、基本アイテムに対して指定されたアイテムの割り振りを一時的に中断します。
  - N = 在庫を割り振ることができます (通常の動作)。
  - Y = 十分な在庫があっても在庫の割り振りができません。
- releaseseparately は、基本アイテムに対して指定されたオーダー・アイテムをリリースする方法を指定します。
  - N = オーダー・アイテムを他のオーダー・アイテムと共にリリースします。
  - Y = オーダー・アイテムは別個にリリースする必要があります (専用の梱包で)。
- returnnotdesired は、顧客が返品したい場合、あるいは返品可能な場合であっても、アイテムの返品が望ましくないということを指定します (腐りやすい食品などの場合)。
  - N = アイテム返品に関する顧客の要望に基づいて、クレジット評価を要求しますが、返品は予期されません。
  - Y = 返品が予期されているかのようなクレジット評価を要求します。
- backorderable は、基本アイテムに対して指定されたアイテムのバックオーダーができないことを指定します。
  - N = アイテムのバックオーダーはできません。
  - Y = アイテムのバックオーダーが可能です。
- creditable は、このアイテムに関してマーチャントがオーバーライドなしでクレジットを発行するかどうかを指定します。
  - N = 現金販売。
  - Y = クレジット可能。
- minqtyforsplit は、新しいオーダー・アイテムの残りの未割り振り数量が、最小数量として指定された数量より少ない場合、在庫の割り振りにおいてオーダー・アイテムを自動的には分割しないことを指定します。

注: ストア・アイテムの在庫割り振り規則を定義するたびに、 baseitem_id 番号が、新しい基本アイテムを表す番号に変わります。たとえば、新しい割り振りには baseitem_id="@baseitem_id_147" が含まれていて、第 3 のものには baseitem_id="@baseitem_id_192" が含まれている、などです。

7. ToolTech サンプル・ストアの次の例を参考にして、後で翻訳することを意図したロケール固有の XML ファイルに基本アイテムの説明を追加します。カタログの中のこのような基本アイテムの説明ごとに、このタスクを完了します。

```
<baseitmdsc
baseitem_id="@baseitem_id_102"
language_id="&en_US;"
shortdescription="Circular Saw"
longdescription="Light on weight but not in quality. The Circular Saw
weighs a maximum of 10.9lbs., with a choice of a 12 or 14 amp motor,
and speeds of up to 600 rpms! Low friction 220V aluminum alloy shoe
will ensure the job gets done on time."
/>
```

ここで、

- baseitem_id は、生成されるユニーク鍵です。
- language_id は、この情報の言語です。
- shortdescription は、基本アイテムの要旨です。
- longdescription は、基本アイテムの詳細記述です。

パート 5: カタログ・エントリーの作成

1. ToolTech サンプル・ストアの次の例を参考にして、CATENTRY および CATENTDESC テーブルに情報を追加することにより、カタログ・エントリーを作成します。各タイプのカタログ・エントリー (商品、アイテム、パッケージ、バンドル、およびダイナミック・キット) は、カタログの中で販売対象となる、オーダー可能な商品取引を表します。各商品カタログ・エントリーごとに 1 回ずつ基本アイテムを定義する必要があります。カタログの中の各商品カタログ・エントリーごとにこのタスクを完了します。

```
<catentry
catentry_id="@product_id_102"
baseitem_id="@baseitem_id_102"
member_id="&MEMBER_ID"
catenttype_id="ProductBean"
partnumber="T0000102"
mfpartnumber="Sprain-Tools-102"
mfname="Sprain Tools"
markfordelete="0"
buyable="1"
/>
```

ここで、

- catentry_id は、商品カタログ・エントリーの内部参照番号です。
- baseitem_id は、カタログ・エントリーと関連のある基本アイテムです。
- member_id は、カタログ・エントリーを識別する参照番号です。
- catenttype_id は、カタログ・エントリーのタイプを識別します。
  - ItemBean = アイテムを表示する。
  - ProductBean = 商品を表示する。
  - PackageBean = パッケージを表示する。
  - BundleBean = バンドルを表示する。
  - DynamicKitBean = ダイナミック・キットを表示する。
- partnumber は、カタログ・エントリーの部品番号を識別する参照番号です。
- mfpartnumber は、カタログ・エントリーを識別するためにメーカーが使用する部品番号です。
- mfname は、カタログ・エントリーのメーカーの名前です。

- markfordelete は、カタログ・エントリーが削除の対象としてマークされているかどうかを示します。
  - 0 = No
  - 1 = Yes
- buyable は、カタログ・エントリーを個別に購入できるかどうかを示します。
  - 0 = No
  - 1 = Yes

  注: 基本アイテムをカタログ・エントリーに追加するたびに、新しいカタログ・エントリーを表すよう、catentry_id と baseitem_id のシーケンスが変わります。 catenttype_id は、カタログ・エントリーのタイプに応じて変わります。マスター・カタログの構造に由来する制限のため、 1 つのカタログ・エントリーが複数のカテゴリーに属することは不可能です。カタログ・エントリーを複数のカテゴリーに入れるには、ナビゲーション・カタログを使用する必要があります。
- ToolTech サンプル・ストアの次の例を参考にして、カタログ・エントリーごとに指定されたアイテムを定義します。カタログの中の各カタログ・エントリーごとにこのタスクを完了します。

```
<catentry
catentry_id="@catentry_id_106"
itemspc_id="@itemspc_id_106"
member_id="&MEMBER_ID"
catenttype_id="ItemBean"
partnumber="T0000106"
mfpartnumber="Sprain-Tools-106"
mfname="Sprain Tools"
markfordelete="0"
buyable="1"
/>
```

  ここで、
  - catentry_id は、カタログ・エントリーの内部参照番号です。
  - itemspc_id は、カタログ・エントリーが属する指定されたアイテムです。
  - member_id は、カタログ・エントリーを識別する参照番号です。
  - cattentype_id は、カタログ・エントリーのタイプを識別します。
    - ItemBean = アイテムを表示する。
    - ProductBean = 商品を表示する。
    - PackageBean = パッケージを表示する。
    - BundleBean = バンドルを表示する。
    - DynamicKitBean = ダイナミック・キットを表示する。
  - partnumber は、カタログ・エントリーの部品番号を識別する参照番号です。
  - mfpartnumber は、カタログ・エントリーを識別するためにメーカーが使用する部品番号です。
  - mfname は、カタログ・エントリーのメーカーの名前です。
  - markfordelete は、カタログ・エントリーが削除の対象としてマークされているかどうかを示します。

- 0 = No
- 1 = Yes

– buyable は、カタログ・エントリーを個別に購入できるかどうかを示します。
- 0 = No
- 1 = Yes

注: 指定されたアイテムをカタログ・エントリーに追加するたびに、catentry_id と itemspc_id のシーケンスが新しいカタログ・エントリーを表すように変わります。 catenttype_id は、カタログ・エントリーのタイプに応じて変わります。マスター・カタログの構造に由来する制限のため、 1 つのカタログ・エントリーが複数のカテゴリーに属することは不可能です。カタログ・エントリーを複数のカテゴリーに入れるには、ナビゲーション・カタログを使用する必要があります。

- ToolTech サンプル・ストアの次の例を参考にして、ロケール固有の XML ファイルに説明を追加します。カタログの中の各カタログ・エントリーの説明ごとに、このタスクを完了します。

```
<catentdesc
catentry_id="@product_id_102"
language_id="&en_US"
name="Circular"
shortdescription="Circular Saw"
longdescription="Light on weight but not in quality. The Circular Saw
weighs a maximum of 10.9lbs., with a choice of a 12 or 14 amp motor,
and speeds of up to 600 rpms! Low friction 220V aluminum alloy shoe
will ensure the job gets done on time."
thumbnail="images/circular_saw_sm.gif"
fullimage="images/circular_saw.gif"
available="1"
published="1"
/>
```

ここで、

– catentry_id は、この言語特定情報が関連するカタログ・エントリーを示す内部参照番号です。
– language_id は、言語の ID です。
– name は、言語に依存するカタログ・エントリーの名前です。
– shortdescription は、カタログ・エントリーの要旨です。
– longdescription は、カタログ・エントリーの詳細記述です。
– thumbnail は、サムネール・イメージのパスです。
– fullimage は、完全なイメージのパスです。
– available は、カタログ・エントリーの可用性に対する時間の長さを指示します。
– published は、このカタログ・エントリーを、language_id によって示された言語で表示するかどうかを指示します。
- 0 = 表示する。
- 1 = 表示しない。

パート 6: 属性と属性値の作成

1. ToolTech サンプル・ストアの次の例を参考にして、後で翻訳することを意図したロケール固有の XML ファイルの ATTRIBUTE および ATTRVALUE テーブルに情報を追加することにより、商品の属性と属性値を作成します。カタログの中の各商品ごとに属性の特定のセットがあります。たとえば、シャツやパンツのサイズおよびカラーなどです。アイテムは、属性値によって定義されます。たとえば、シャツは商品ですが、M サイズの黒のシャツはアイテムです。カタログの中の属性ごとにこのタスクを完了します。

   ```
   <attribute
   attribute_id="@attribute_id_103"
   language_id="&en_US"
   attrtype_id="STRING"
   name="Amps"
   sequence="0"
   description="Amps"
   catentry_id="@product_id_102"
   description2="Amps"
   />
   ```

   ここで、

   - attribute_id は、属性の内部参照番号です。
   - language_id は、この属性値が関係する言語です。
   - attrtype_id は、対応する属性値のタイプです。
   - name は、属性の名前です。
   - sequence は、指定の商品の属性の表示順序を決定する順序番号です。
   - description は、属性の説明です。
   - catentry_id は、この属性が属する商品の参照番号です。
   - description2 は、属性の追加説明です。

   注: catentry_id によって定義される商品に属性を追加するたびに、新しい属性を表すよう、attribute_id のシーケンスが変わります。

2. ToolTech サンプル・ストアの次の例を参考にして、属性値を追加します。カタログの中の属性値ごとにこのタスクを完了します。

   ```
   <attrvalue
   attrvalue_id="@attrvalue_id_114"
   language_id="&en_US"
   attribute_id="@attribute_id_103"
   name="12.0amps"
   attrtype_id="STRING"
   stringvalue="12.0amps"
   sequence="0"
   catentry_id="@catentry_id_106"
   />
   ```

   ここで、

   - attrvalue_id は、属性値の内部参照番号です。
   - language_id は、この属性値が関係する言語です。
   - attribute_id は、値と関連した属性の内部参照番号です。
   - name は、属性値の名前です。
   - attrtype_id は、属性値のタイプです。

- stringvalue は、属性値です。
- sequence は、指定の属性の属性値の表示順序を決定する順序番号です。
- catentry_id は、この属性値が記述するアイテム ID です。

注: 属性値を属性に追加するたびに、異なる値を表すよう、attrvalue_id のシーケンスが変わります。 attribute_id シーケンスも変わって、異なる属性を表します。新しい属性値が追加されるたびに、sequence は大きくなります。たとえば、後に続く属性値は sequence="1"、sequence="2"、および sequence="3" (以下同様) などになるかもしれません。

パート 7: 商品とアイテムの間の関係の作成

1. カタログに商品とアイテムを作成したら、CATENTREL テーブルに情報を追加することによって、商品とアイテムの間の関係を定義します。 ToolTech サンプル・ストアの次の例を参考にします。カタログの中の商品とアイテムの関係ごとにこのタスクを完了します。

```
<catentrel
catentry_id_parent="@product_id_147"
catreltype_id="PRODUCT_ITEM"
catentry_id_child="@catentry_id_152"
sequence="2"
quantity="1"
/>
```

ここで、

- catentry_id_parent は、この関係のソース・カタログ・エントリー (商品) の参照番号です。
- catreltype_id は、関係のタイプ (PRODUCT_ITEM) です。
- catentry_id_child は、この関係のターゲット・カタログ・エントリー (アイテム) の参照番号です。
- sequence は、表示順序を判別するために使用されるシーケンス番号です。
- quantity は、関係と関連している可能性がある数量です。

注: 商品とアイテムの間の関係を追加するたびに、 catentry_id_parent と catentry_id_child の番号が変わり、 catreltype_id に基づいてさまざまな関係が作成されます。新しい関係はそれぞれ、sequence 番号が異なります。たとえば、sequence="2" がある場合、次の関係の番号は sequence="3" になり、さらに sequence="4" (以下同様) と続きます。

パート 8: パッケージとバンドルの作成

1. 商品とアイテムを作成したら、CATENTRY、CATENTDESC、および CATENTREL の各テーブルに情報を追加することによって、パッケージとバンドルを作成します。 ToolTech サンプル・ストアの次の例を参考にして、まず CATENTRY テーブルに情報を追加して、パッケージまたはバンドルを作成します。カタログの中の各パッケージおよびバンドルごとに、このタスクを完了します。

```
<catentry
catentry_id="@package_id_102"
member_id="&MEMBER_ID"
catenttype_id="PackageBean"
partnumber="sku-@package_id_102"
```

```
mfpartnumber="sku-@package_id_102"
mfname="ToolTech"
markfordelete="0"
buyable="1"
/>
```

ここで、

- catentry_id は、カタログ・エントリーの参照番号です。
- member_id は、カタログ・エントリーの所有者を識別する参照番号です。
- catenttype_id は、カタログ・エントリーのタイプを識別します。
  - PackageBean = パッケージを表示する。
  - BundleBean = バンドルを表示する。
- partnumber は、カタログ・エントリーの部品番号を識別する参照番号です。
- mfpartnumber は、カタログ・エントリーを識別するためにメーカーが使用する部品番号です。
- mfname は、カタログ・エントリーのメーカーの名前です。
- markfordelete は、カタログ・エントリーが削除の対象としてマークされているかどうかを示します。
  - 0 = No
  - 1 = Yes
- buyable は、カタログ・エントリーが個別に購入できるかどうかを示します。
  - 0 = No
  - 1 = Yes

  注: パッケージまたはバンドルを作成するたびに、 catentry_id、partnumber、および mfpartnumber の番号が、異なるパッケージまたはバンドルを作成するように変わります。たとえば、新しいパッケージを作成する場合、エントリーをパッケージとして識別するため、catentry_id="@package_id_103"、 partnumber="sku-@package_id_103"、および mfpartnumber="sku-@package_id_103" (catenttype_id="PackageBean" を含む) を使用できます。新しいバンドルを作成するためには、エントリーをバンドルとして識別するために、catentry_id="@package_id_110"、 partnumber="sku-@package_id_110"、および mfpartnumber="sku-@package_id_110" (catenttype_id="BundleBean" を含む) を使用する、というようになります。
- ToolTech サンプル・ストアの次の例を参考にして、後で翻訳することを意図したロケール固有の XML ファイルの CATENTDESC テーブルに情報を追加することにより、パッケージまたはバンドルの説明を追加します。カタログの中の各パッケージおよびバンドルの説明ごとに、このタスクを完了します。

```
<catentdesc
catentry_id="@catentry_id_102"
language_id="-1"
name="computer"
shortdescription="Computer"
longdescription="A combination of a central processing unit, monitor,
 hard drive, and color printer. An ideal starter system."
thumbnail="images/package_system_sm.gif"
```

```
fullimage="images/package_system.gif"
available="1"
published="1"
/>
```

ここで、

- catentry_id は、この言語特定情報が関連するカタログ・エントリーを示す内部参照番号です。
- language_id は、言語の ID です。
- name は、言語に依存するカタログ・エントリーの名前です。
- shortdescription は、カタログ・エントリーの要旨です。
- longdescription は、カタログ・エントリーの詳細記述です。
- thumbnail は、カタログ・エントリーのサムネール・イメージのパスです。
- fullimage は、カタログ・エントリーの完全なイメージのパスです。
- available は、カタログ・エントリーの可用性に対する時間の長さを指示します。
- published は、カタログ・エントリーを、language_id によって示された言語で表示するかどうかを指示します。
  - 0 = カタログ・エントリーを表示しない。
  - 1 = カタログ・エントリーを表示する。

- ToolTech サンプル・ストアの次の例を参考にして、CATENTREL テーブルに情報を追加し、パッケージまたはバンドルとそのコンポーネントとの間の関係を作成します。カタログの中のパッケージまたはバンドルのコンポーネントの関係ごとにこのタスクを完了します。

```
<catentrel
catentry_id_parent="@catentry_id_102"
catreltype_id="PACKAGE_COMPONENT"
catentry_id_child="@catentry_id_97"
sequence="1.0"
quantity="1.0"
/>
```

ここで、

- catentry_id_parent は、この関係のソース・カタログ・エントリー (パッケージまたはバンドル) の参照番号です。
- catreltype_id は、この関係のタイプです。
  - PACKAGE_COMPONENT は、パッケージとそのコンポーネントの間の関係を表します。
  - BUNDLE_COMPONENT は、バンドルとそのコンポーネントの間の関係を表します。
- catentry_id_child は、この関係のターゲット・カタログ・エントリー (コンポーネント) の参照番号です。
- sequence は、表示順序を判別するために使用されるシーケンス番号です。
- quantity は、関係と関連している可能性がある数量です。

注: パッケージとバンドル間の関係を作成するたびに、既存のカタログ・エントリーに合わせて catentry_id_parent と catentry_id_child の番号が変わります。新しい関係はそれぞれ、sequence 番号が異なります。たとえば、sequence="1.0" から始まる場合、次の関係は sequence="2.0" になり、さらに sequence="3.0" (以下同様) と続きます。

パート 9: カタログ・グループとカタログ・エントリーの間の関係の作成

1. カタログの中にカタログ・グループとカタログ・エントリーを作成したら、CATGPENREL テーブルに情報を追加することによって、カタログ・グループとカタログ・エントリーとの間の関係を定義します。 ToolTech サンプル・ストアの次の例を参考にします。カタログの中のカタログ・グループとカタログ・エントリーとの関係ごとにこのタスクを完了します。

```
<catgpenrel
catgroup_id="@catgroup_id_11"
catalog_id="@catalog_id_1"
catentry_id="@product_id_102"
sequence="0"
/>
```

ここで、

- catgroup_id は、この関係のソース・カタログ・グループです。
- catalog_id は、この関係が含まれているカタログです。
- catentry_id は、この関係のターゲット・カタログ・グループです。
- sequence は、カタログ・グループの内容の表示順序を決定するシーケンス番号です。

注: カタログ・グループとカタログ・エントリーと間の関係を作成するたびに、catgroup_id と catentry_id の番号が変わり、さまざまなカタログ・グループとカタログ・エントリーの新しい関係が形成されます。新しい関係はそれぞれ、sequence 番号が異なります。たとえば、sequence="0" から始まる場合、次の関係は sequence="1" になり、さらに sequence="2" (以下同様) と続きます。

パート 10: 取引管理アソシエーションの作成

1. ToolTech サンプル・ストアの次の例を参考にして、MASSOCECE テーブルに情報を追加し、カタログ・エントリー同士の取引管理アソシエーションを作成します。カタログの中の取引管理アソシエーションごとにこのタスクを完了します。

```
<massoccece
massoccece_id="@relationship_id_100"
massoctype_id="X-SELL"
catentry_id_from="@product_id_1"
catentry_id_to="@product_id_15"
massoc_id="REQUIRES"
quantity="2.0"
rank="1.00000"
/>
```

ここで、

- massoccece_id は、このエントリーの参照番号です。
- massoctype_id は、アソシエーション・タイプの ID です。

  - X-SELL = 関連商品販売。
  - UPSELL = 上位商品販売。
  - ACCESSORY = アクセサリー。
- catentry_id_from は、アソシエーションのソースであるカタログ・エントリーです。
- catentry_id_to は、アソシエーションのターゲットであるカタログ・エントリーです。
- massoc_id は、セマンティック指定子の ID です。
  - REQUIRED
  - OPTIONAL
  - COMES WITH
- quantity は、このアソシエーションに関連した数量です。
- rank は、表示順序に使用されるシーケンス番号です。

注: 取引管理アソシエーションを追加するたびに、新しい関係を表すよう、massoccece_id の番号が変わります。 catentry_id_from と catentry_id_to の番号が変わり、アソシエーションの新しい商品取引内容が作成されます。

パート 11: ストアへのカタログの関連付け

1. ToolTech サンプル・ストアの既存の storecatalog.xml を参考にして、カタログ、そのカタログ・グループ、およびカタログ・エントリーを、データベースの中のストアに割り当てることによって、カタログをストアに関連付けます。また、表示ページをカタログ・グループおよびカタログ・エントリーに割り当てることも必要です。この情報を、STORECAT、STORECENT、STORECGRP、DISPCGPREL、および DISPENTREL の各テーブルに追加します。多文化カタログを作成する場合、ストアがサポートする各ロケールごとに別個のストア / カタログ関係 XML ファイルを作成します。

   ```
   <storecat
   catalog_id="@catalog_id_1"
   storeent_id="@storeent_id_1"
   mastercatalog="1"
   />
   ```

   ここで、

   - catalog_id は、カタログの参照番号です。
   - storeent_id は、データベースの中のストア・エンティティーの参照番号です。
   - mastercatalog は、ストアのマスター・カタログを指定します。値 1 は、このカタログをマスター・カタログとして指定することを示します。
2. ToolTech サンプル・ストアの次の例を参考にして、ストア / カタログ関係にカタログ・エントリーを追加します。カタログの中の各カタログ・エントリーごとにこのタスクを完了します。

   ```
   <storecent
   storeent_id="@storeent_id_1"
   catentry_id="@product_id_102"
   />
   ```

ここで、

- storeent_id は、データベースの中のストア・エンティティーの参照番号です。
- catentry_id は、カタログ・エントリーの参照番号です。

注: catentry_id をストア・エンティティーに追加するたびに、参照番号が既存のカタログ・エントリーに合わせて変化します。

3. ToolTech サンプル・ストアの次の例を参考にして、ストア・エンティティーにカタログ・グループを追加します。カタログの各カタログ・グループごとにこのタスクを完了します。

```
<storecgrp
storeent_id="@storeent_id_1"
catgroup_id="@catgroup_id_1"
/>
```

ここで、

- storeent_id は、データベースの中のストア・エンティティーの参照番号です。
- catgroup_id は、カタログ・グループの参照番号です。

注: catgroup_id をストア・エンティティーに追加するたびに、参照番号が既存のカタログ・グループに合わせて変化します。

.

パート 12: カタログへの税の関連付け

1. 税を、特定のストアのカタログの商品およびサービスに関連付けます。この情報を CATENCALCD テーブルに追加することによって、税額計算コードをカタログ・エントリーと関連付けることが必要です。詳細については、 160 ページの『WebSphere Commerce での税資産の作成』を参照してください。

パート 13: カタログへの配送方法の関連付け

1. 配送方法をカタログの商品およびサービスに関連付けるには、配送計算コードをカタログ・エントリーと関連付けることが必要です。この情報を CATENCALCD テーブルに追加してください。詳細については、 141 ページの『WebSphere Commerce での配送資産の作成』を参照してください。

パート 14: カタログへの配送センターの関連付け

1. 製品を顧客に配送するために、カタログを配送センターに関連付けます。配送センターは、ストアの商品の在庫および配送を管理します。この情報を FFMCENTER テーブルに追加してください。詳細については、 116 ページの『WebSphere Commerce での配送資産の作成』を参照してください。

パート 15: カタログ・エントリーの価格の作成

1. カタログ・エントリーの価格設定を行います。価格設定は、カタログ・エントリーの価格範囲、およびその価格を使用するために満たす必要のある基準を表します。機能的なカタログを作成するには、オファリング情報をデータベースに追加することが必要です。この情報を、TRADEPOSCN、TDPSCNCNTR、MGPTRDPSCN、OFFER、および OFFERPRICE テーブルに追加してください。

詳細については、 96 ページの『WebSphere Commerce での価格設定資産の作成』を参照してください。あるいは、WebSphere Commerce アクセラレーターの商品管理ツールを使用して、カタログ・エントリーの価格設定の作成または更新を行えます。

パート 16: XML ファイルのロード

1. データを作成したら、ローダー・パッケージを使用するか、ストア・サービスの発行機能を使って、XML ファイルをデータベースにロードします。ローダー・パッケージについて詳しくは、 225 ページの『第 7 部 ストアの発行』を参照してください。

   注: WebSphere Commerce アクセラレーターの商品管理ツールを使用することによっても、マスター・カタログのカタログ資産を作成できます。商品管理ツールについては、WebSphere Commerce のオンライン・ヘルプを参照してください。

ストア・カタログ資産の表示

カタログ、カタログ・グループ、およびカタログ・エントリーをストアに関連付けたなら、データベース中にこれらの関係を作成することによって、カタログ・エントリーとカタログ・グループを表示する JSP テンプレートを割り当てます。これらの関係を XML ファイルの形式で作成します。XML ファイルは、ローダー・パッケージを使用して、データベースにロードできます。

ToolTech サンプルの `storecatalog.xml` ファイルがストア・アーカイブ・ファイルにあります。 `storecatalog.xml` ファイルを表示するには、ZIP プログラムを使用して、ストア・アーカイブを解凍します。 `storecatalog.xml` ファイルは以下の data ディレクトリーにあります。

- NT *drive*:¥WebSphere¥CommerceServer¥samplestores
- 2000 *drive*:¥Program Files¥WebSphere¥CommerceServer¥samplestores
- AIX /usr/WebSphere/CommerceServer/samplestores
- Solaris Linux /opt/WebSphere/Commerce/samplestores
- 400 /qibm/proddata/WebCommerce/samplestores

注: WebSphere Commerce オンライン・ヘルプには、サンプル・ストアに含まれる各データ資産についての情報が記載されています。

`store-catalog.dtd` ファイルは、以下のディレクトリーにあります。

- NT *drive*:¥WebSphere¥CommerceServer¥xml¥sar
- 2000 *drive*:¥Program Files¥WebSphere¥CommerceServer¥xml¥sar
- AIX /usr/WebSphere/CommerceServer/xml/sar
- Solaris Linux /opt/WebSphere/CommerceServer/xml/sar
- 400 /qibm/proddata/WebCommerce/xml/sar

ストア - カタログ関係を作成する前に、ストア・データ資産を作成してあることを確認してください。以下のタスクを実行します。それぞれのタスクを実行すると、storecatalog.xml ファイルにエントリーが作成されます。

1. ストアの中のカタログ・グループ (カテゴリー) を表示するには、JSP テンプレートをカタログ・グループに割り当てることが必要です。カタログ・グループに特定の表示ページのテンプレートを割り当てることもできますし、すべてのカタログ・グループを表示するためのデフォルト・テンプレートを割り当てることもできます。 ToolTech サンプル・ストアの次の例を参考にして、DISPCGPREL テーブルに情報を追加し、カタログ・グループ・テンプレートを割り当てます。カタログ・グループに割り当てる各テンプレートごとにこのタスクを完了します。

   ```
   <dispcgprel
   catgroup_id="@catgroup_id_1"
   devicefmt_id="-1"
   dispcgprel_id="@dispcgprel_id_1"
   mbrgrp_id="0"
   pagename="/ToolTech/CategoryDisplay.jsp"
   storeent_id="@storeent_id_1"
   rank="0"/>
   ```

   ここで、

   - catgroup_id は、このページ名が表示されるカタログ・グループの参照番号です。値 0 は、このページ名がすべてのカタログ・グループに使用されることを示します。
   - devicefmt_id は、ページが表示されるデバイス・タイプの参照番号です。値 -1 は、このテンプレート・ページが HTTP ブラウザーに使用されることを示します。
   - dispcgprel_id は、このエントリーの参照番号です。
   - mbrgrp_id は、このテンプレート・ページが表示されるメンバー・グループの参照番号です。値 0 は、このテンプレート・ページがすべてのメンバー・グループに使用されることを示します。
   - pagename は、表示テンプレート・ページの名前およびパスです。
   - rank は、複数のページが選択基準を満たすときに、関係を切るために使用されるシーケンス番号です。

   注: JSP テンプレートをカタログ・グループに関連付けるたびに、 catentry_id のシーケンスが既存のカタログ・エントリーに合わせて変化します。

2. ストアのカタログ・エントリー (商品、アイテム、パッケージ、バンドル、およびダイナミック・キット) を表示するには、 JSP テンプレートをカタログ・エントリーに割り当てることが必要です。すべてのカタログ・エントリーを表示するためのデフォルト・テンプレートを割り当てることもできますし、カタログ・エントリーのタイプごとにデフォルトを割り当てることもできます。たとえば、製品、アイテム、あるいは特定のカタログ・エントリーに、それぞれ別個のテンプレートを割り当てるという方法です。 ToolTech サンプル・ストアの次の例を参考にして、DISPENTREL テーブルに情報を追加し、テンプレートを割り当てます。カタログ・エントリーに割り当てる各テンプレートごとにこのタスクを完了します。

```
<dispentrel
auctionstate="0"
catentry_id="0"
catenttype_id="ProductBean"
devicefmt_id="-1"
dispentrel_id="@dispentrel_id_1"
mbrgrp="0"
pagename="/ToolTech/ProductDisplay.jsp"
storeent_id="@storeent_id_1"
rank="0"/>
```

ここで、

- auctionstate は、このテンプレート・ページが、オークション対象のカタログ・エントリーを表示することを示します。
  - 0 = オークション・テンプレートではない。
  - 1 = オークション・テンプレート。
- catentry_id は、このページ名が表示されるカタログ・エントリーの参照番号です。値 0 は、このページ名がすべてのカタログ・エントリーに使用されることを示します。
- catenttype_id は、このページを使用して表示されるカタログ・エントリーのタイプです。
  - ProductBean = 商品を表示する。
  - ItemBean = アイテムを表示する。
  - PackageBean = パッケージを表示する。
  - BundleBean = バンドルを表示する。
  - DynamicKitBean = ダイナミック・キットを表示する。
- devicefmt_id は、ページが表示されるデバイス・タイプの参照番号です。値 -1 は、このテンプレート・ページが HTTP ブラウザーに使用されることを示します。
- dispentrel_id は、カタログ・エントリーの参照番号です。
- mbrgrp は、このテンプレート・ページが表示されるメンバー・グループの参照番号です。値 0 は、このテンプレート・ページがすべてのメンバー・グループに使用されることを示します。
- pagename は、表示テンプレート・ページの名前およびパスです。
- storeent_id は、このページが表示されるストアの参照番号です。
- rank は、複数のページが選択基準を満たすときに、関係を切るために使用されるシーケンス番号です。

注: JSP テンプレートをカタログ・エントリーに関連付けるたびに、catentry_id のシーケンスが既存のカタログ・エントリーに合わせて変化します。

ナビゲーション・カタログの作成

WebSphere Commerce ストアでは、マスター・カタログとナビゲーション・カタログという 2 種類のカタログを使用できます。オンライン・カタログはナビゲーション・カタログと見なされます。マスター・カタログが構造上の制限を満たさない場合にのみ、WebSphere Commerce のカタログはナビゲーション・カタログと呼ばれ

ます。そのような場合、それらのカタログは、より柔軟な表示構造を使用することにより、ストアのマスター・カタログから区別されます。このとき、2 つの主な制限があります。

- マスター・カタログは、適切なツリーでなければならない。つまり、循環が存在していてはならず、親カテゴリー A にサブカテゴリー B があるとき、B および B のいかなるサブカテゴリーも A の親カテゴリーになることはできません。
- 1 つの商品が複数のカテゴリーに属することはできません。

以下のタスクでは、サンプル・ストア・カタログ NewFashion を使用し、カテゴリー循環を作成して商品とその SKU を 2 番目のカテゴリーの下に含めることにより、マスター・カタログの構造をナビゲーション・カタログに変更する方法について説明します。従来のナビゲーション・カタログは、カテゴリー関係テーブル (各カタログのサブカテゴリー関係が含まれる CATGRPREL と、各カタログのカテゴリーと商品の関係が含まれる CATGPENREL) に情報を追加することによって作成されます。例では NewFashion を使用していますが、独自のマスター・カタログでも、カタログ情報、構造、および設計に合わせて適切な調整を行うことにより、以下の基本的なステップを行うことができます。

カテゴリー循環の作成

NewFashion カタログには、「メンズ」、「レディース」、「新着」、および「ホーム・ページ販売」という 4 つの最上位カテゴリーが含まれています。この例では、「新着」カテゴリーを「メンズ」カテゴリーの中にコピーし、「ホーム・ページ販売」を「レディース」の中にコピーしてその名前を「新着」にする方法を示します。

カテゴリー循環を使用して NewFashion サンプル・ストア・マスター・カタログをナビゲーション・カタログに変更するには、以下のようにします。

1. NewFashion ストア・アーカイブを発行して NewFashion サンプル・ストアを作成する。 NewFashion は米国英語版と他の 9 つの各国語版が WebSphere Commerce に付属しています。 `NewFashion_en_US_`*`locale`*`.sar` ファイルの 1 つを発行用に選択してください。
2. `catalog.xml` ファイルをエディターで開く。このファイルは以下の WebSphere Commerce ディレクトリーにあります。
   - NT *drive*`:¥WebSphere¥CommerceServer¥samplestores¥NewFashion¥`*`locale`*`¥data`
   - 2000 *drive*`:¥Program Files¥WebSphere¥CommerceServer¥samplestores¥NewFashion¥`*`locale`*`¥data`
   - AIX `/usr/WebSphere/CommerceServer/samplestores/NewFashion/`*`locale`*`/data`
   - Solaris Linux `/opt/WebSphere/Commerce/samplestores/NewFashion/`*`locale`*`/data`
   - 400 `/qibm/proddata/WebCommerce/samplestores/NewFashion/`*`locale`*`/data`

3. catalog.xml ファイルで CATGRPREL データ・セクションを見付ける。**「メンズ・ファッション」** と**「新着」** の間に新しい最上位カテゴリー関係を作成します。この時点では、どちらのカテゴリーも最上位にあります。ナビゲーション関係用に新しいセクションを作成し、元の構造を保持しながら、**「新着」**を**「メンズ・ファッション」**のサブカテゴリーとして入れます。 CATGRPREL セクションの下に、以下の部分を追加します。

```
<catgrprel
catgroup_id_parent="@catgroup_id_11"
catgroup_id_child="@catgroup_id_21"
catalog_id="@catalog_id_1"
sequence="7"
/>
```

ここで、

- catgroup_id_parent は、NewFashion サンプル・ストアによって定義された、親カテゴリーのカタログ・グループ内部参照番号です。この例では、@catgroup_id_11 は**「メンズ・ファッション」**カテゴリーです。
- catgroup_id_child は、NewFashion サンプル・ストアによって定義された、子カテゴリーのカタログ・グループ内部参照番号です。この例では、@catgroup_id_21 は**「新着」**カテゴリーです。
- catalog_id は、NewFashion サンプル・ストアによって定義された、カタログの内部参照番号です。
- sequence は、NewFashion サンプル・ストアによって定義された、カタログ・グループの内容の表示順序を決定する番号です。この例では、**「メンズ・ファッション」**の最初の 6 つのサブカテゴリーが表示された後で、 **「新着」**カテゴリーが最後に表示されます。

4. 上記のステップを繰り返し、今回は、**「レディース・ファッション」**と**「ホーム・ページ販売」**の間に新しい最上位カテゴリー関係を作成します。この時点では、どちらのカテゴリーも最上位にあります。ナビゲーション関係用に新しいセクションを作成し、元の構造を保持しながら、**「ホーム・ページ販売」**を**「レディース・ファッション」**のサブカテゴリーとして入れます。 CATGRPREL セクションの下に、以下の部分を追加します。

```
<catgrprel
catgroup_id_parent="@catgroup_id_20"
catgroup_id_child="@catgroup_id_22"
catalog_id="@catalog_id_1"
sequence="9"
/>
```

ここで、

- catgroup_id_parent は、NewFashion サンプル・ストアによって定義された、親カテゴリーのカタログ・グループ内部参照番号です。この例では、@catgroup_id_20 は**「レディース・ファッション」**カテゴリーです。
- catgroup_id_child は、NewFashion サンプル・ストアによって定義された、子カテゴリーのカタログ・グループ内部参照番号です。この例では、@catgroup_id_22 は**「ホーム・ページ販売」**カテゴリーです。
- catalog_id は、NewFashion サンプル・ストアによって定義された、カタログの内部参照番号です。

- sequence は、NewFashion サンプル・ストアによって定義された、カタログ・グループの内容の表示順序を決定する番号です。この例では、**「ホーム・ページ販売」**カテゴリーが最後に表示されます。

5. これで**「ホーム・ページ販売」**は**「レディース・ファッション」** のサブカテゴリーになったので、カテゴリー名が正しくありません。**「メンズ・ファッション」**の新しいサブカテゴリーに合わせて、カテゴリー名を **「新着」**に変更してください。
6. catalog.xml ファイルを保管する。
7. 変更内容を表示するには、以下のようにする。変更された NewFashion ストア・アーカイブを発行するか、281 ページの『データ・グループのロード』の指示に従い、ローダー・パッケージを使用して catalog.xml ファイルをロードします。

2 番目のカテゴリーへの商品の追加

この例では、元の構造を保持しながら、1 つのカテゴリーから別のカテゴリーに商品をコピーする方法を示します。**「ホーム・ページ販売」**カテゴリーに含まれる**「サマー・ナイトガウン」** 商品は、最上位カテゴリー**「レディース・ファッション」**の**「スリープ・ウェア」** サブカテゴリーにも所属することができます。以下の説明では、**「サマー・ナイトガウン」**商品とその SKU を**「スリープ・ウェア」** カテゴリーにコピーする方法を示します。

商品を 2 番目のカテゴリーに追加して NewFashion サンプル・ストア・マスター・カタログをナビゲーション・カタログに変更するには、以下のようにします。

1. NewFashion ストア・アーカイブを発行して NewFashion サンプル・ストアを作成する。 NewFashion は米国英語版と他の 9 つの各国語版が WebSphere Commerce に付属しています。 NewFashion_en_US_*locale*.sar ファイルの 1 つを発行用に選択してください。
2. catalog.xml ファイルをエディターで開く。このファイルは以下の WebSphere Commerce ディレクトリーにあります。
   - NT *drive*:¥WebSphere¥CommerceServer¥samplestores¥NewFashion¥*locale*¥data
   - 2000 *drive*:¥Program Files¥WebSphere¥CommerceServer¥samplestores¥NewFashion¥*locale*¥data
   - AIX /usr/WebSphere/CommerceServer/samplestores/NewFashion/*locale*/data
   - Solaris Linux /opt/WebSphere/Commerce/samplestores/NewFashion/*locale*/data
   - 400 /qibm/proddata/WebCommerce/samplestores/NewFashion/*locale*/data
3. catalog.xml ファイルで CATGPENREL データ・セクションを見付ける。**「サマー・ナイトガウン」**の新しい商品エントリーを作成します。これはもともと、**「ホーム・ページ販売」**カテゴリーの下にある商品です。 CATGPENREL セクションの下に、以下の部分を追加してこの商品を含めます。

```
<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@product_id_2692"
sequence="2"
/>
```

ここで、

- catgroup_id は、NewFashion サンプル・ストアによって定義された、カタログ・グループ内部参照番号です。この例では、@catgroup_id_18 は**「スリープ・ウェア」**カテゴリーです。
- catalog_id は、NewFashion サンプル・ストアによって定義された、カタログの内部参照番号です。
- catentry_id は、NewFashion サンプル・ストアによって定義された、カタログ・エントリー内部参照番号です。この例では、@catentry_id_2692 は**「サマー・ナイトガウン」**カテゴリーです。
- sequence は、NewFashion サンプル・ストアによって定義された、カタログ・グループの内容の表示順序を決定する番号です。この例では、**「サマー・ナイトガウン」**商品が最後に表示されます。

4. **「サマー・ナイトガウン」**商品エントリーを追加したら、 NewFashion サンプル・ストアで定義されているように、商品の SKU エントリーを CATGPENREL セクションの下に追加する。現在、**「サマー・ナイトガウン」**商品には、10 個の定義済み SKU が含まれています。 CATGPENREL セクションの下に、以下の部分を追加してこれらの SKU を含めます。

```
<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@catentry_id_2695"
sequence="2"
/>

<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@catentry_id_2696"
sequence="2"
/>

<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@catentry_id_2697"
sequence="2"
/>

<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@catentry_id_2698"
sequence="2"
/>

<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@catentry_id_2699"
sequence="2"
/>
```

```
<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@catentry_id_2700"
sequence="2"
/>

<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@catentry_id_2701"
sequence="2"
/>

<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@catentry_id_2702"
sequence="2"
/>

<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@catentry_id_2703"
sequence="2"
/>

<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@catentry_id_2704"
sequence="2"
/>
```

ここで、

- catgroup_id は、NewFashion サンプル・ストアによって定義された、カタログ・グループ内部参照番号です。この例では、@catgroup_id_18 は**「スリープ・ウェア」**カテゴリーです。
- catalog_id は、NewFashion サンプル・ストアによって定義された、カタログの内部参照番号です。
- catentry_id は、NewFashion サンプル・ストアによって定義された、カタログ・エントリー内部参照番号です。この例では、@catentry_id_2695 ～ @catentry_id_2704 は、**「サマー・ナイトガウン」**商品用に定義されている 10 個の SKU を表しています。
- sequence は、NewFashion サンプル・ストアによって定義された、カタログ・グループの内容の表示順序を決定する番号です。この例では、**「サマー・ナイトガウン」**の SKU が最後に表示されます。

5. catalog.xml ファイルを保管する。
6. 変更内容を表示するには、以下のようにする。変更された NewFashion ストア・アーカイブを発行するか、281 ページの『データ・グループのロード』の指示に従い、ローダー・パッケージを使用して catalog.xml ファイルをロードします。

WebSphere Commerce でのカタログ資産の管理

時間とともに、マスター・カタログの資産情報を更新する必要が生じます。カタログの管理は継続的なプロセスであり、継続的に商品取引の追加や除去を行ったり、カテゴリーやカタログ・グループの作成や関連づけを行ったり、説明や価格などの商品情報を更新したりしなければなりません。

商品管理ツール

WebSphere Commerce アクセラレーターの商品管理ツールを使用すれば、さまざまなウィザードやノートブックを使用してストアのマスター・カタログにある商品を管理できます。以下のように、カタログの内容を更新したり、新しいカタログ・データを作成したりできます。

- ウィザードやノートブックを使用して、商品や商品詳細情報を作成、更新、および削除する。商品は、SKU、つまり最終的に顧客に販売される個々のアイテムのテンプレートとして機能します。商品詳細情報には、商品コード (商品を一意的に識別する)、商品の名前と説明、取引管理オプション (商品を顧客に表示したり、商品が特別な販売促進品であることを示したりする) 、商品イメージ、税と配送の仕様、商品に割り当てられる割引、および製造元情報が含まれます。
- 購入用の SKU (またはアイテム) を生成、更新、および削除する。 SKU は販売用の商品取引におけるそれぞれのオーダー可能アイテムを表します。特定の商品に関連するすべての SKU は同じ属性セットを示し、それらの属性値によって区別されます。 SKU への追加や変更には、商品と同じ情報が含まれます。ただしこれは、オーダー可能かどうかに基づきます。
- カテゴリー (またはカタログ・グループ) を作成、更新、および削除する。カテゴリーは、ストアが提供する商品やサービスを編成するために使用される同様のプロパティーを持つオブジェクトのグループです。カテゴリーやカテゴリー詳細 (たとえば、カテゴリー・コード、名前、および親カテゴリーやイメージなどの説明) を作成、変更、および削除することにより、マスター・カタログのカテゴリー階層を管理できます。
- 親カテゴリーを選択したり、商品と SKU を 1 つのカテゴリーから別のカテゴリーに移動したりすることにより、商品と SKU をカテゴリーに関連付ける。
- 商品の属性と属性値を作成する。属性と属性値との可能な組み合わせが、それぞれ新しい SKU となります。属性値を SKU に割り当てる前に、それを定義しておく必要があります。属性とその値を作成したら、名前、説明、(テキスト、整数または 10 進数)、および属性と属性値が表示される順序などの情報を作成または更新できます。
- カタログの価格設定の作成、更新、削除、および商品への関連付けを行う。商品や SKU の価格を 1 つ以上の通貨で定義し、それとともに、その価格を使用するために満たさなければならない条件 (たとえば、1 つだけ買ったときとまとめ買いしたときの価格設定) のセットを定義できます。

各タスクの詳細については、オンライン・ヘルプの『Product Management (商品管理)』セクションを参照できます。

注:

1. 商品管理ツールは、小さい変更を行う場合にのみ使用をお勧めします。カタログの大きな更新 (季節ごとの商品取引の追加や削除、または在庫一掃セールの準備など) では、ローダー・パッケージを使用してください。
2. カタログ・データの変更内容は、キャッシングを使用不可にするか、現在キャッシングされている JSP ページを削除しない限り、ストアで表示できません。詳しくは、WebSphere Commerce オンライン・ヘルプで CacheDelete コマンドを参照してください。 CacheDelete コマンドを実行すると、動的ページ・キャッシュのリモート・クリーンアップが開始され、ファイル・システムへの直接アクセスなしにキャッシュを管理できるようになります。このコマンドを使用する前に、「自動ページ無効」が使用可能になっていることを確かめてください。このコマンドを使用するにはサイト管理者かストア開発者としてログインしなければならないことに注意してください。

Catalog Manager

ローダー・パッケージか Web Editor (Catalog Manager で使用可能な 2 つのユーティリティー) を使用してカタログを保守することもできます。既存の商品情報をデータベースに大量にインポートする場合は、ローダー・パッケージが理想的です。これは、WebSphere Commerce でカタログ情報の作成や管理を行うための主なツールです。このパッケージは主に、データを準備したり WebSphere Commerce データベースにロードしたりするためのコマンド・ユーティリティーで構成されています。また、ローダー・パッケージでは、データベースからデータを XML 文書として抽出したり、XML データを代替 XML 形式に変換したり、文字区切り変数形式と XML データ形式の間でデータを変換したりすることもできます。

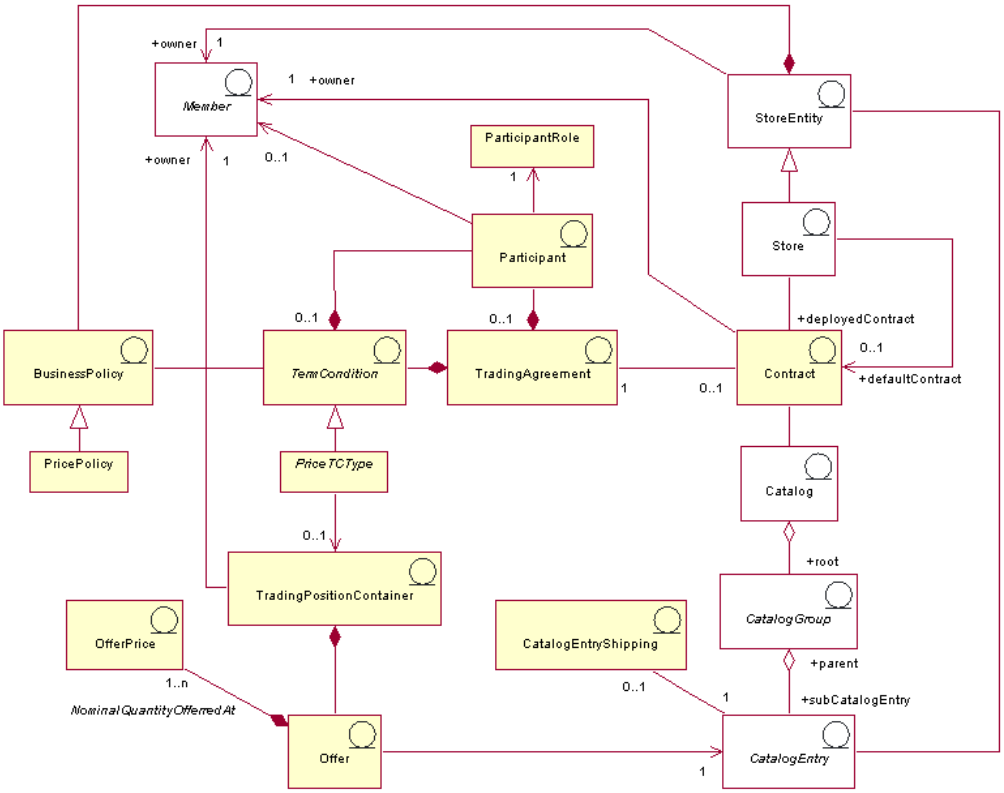
Web Editor では Web ブラウザーのインターフェースが使用されており、カタログ・データの作成、編集、または削除を行えます。 Web Editor の中心をなすのは、情報の表示や更新を行うためのデータ入力フォームです。最も単純なケースでは、これらのフォームは WebSphere Commerce データベース内のテーブルに対応しています。管理者は、提供されているデフォルト・フォームを使用するか、使用可能なフォームをカスタマイズすることができます。詳しくは、*IBM WebSphere Commerce Version 5.4 Catalog Manager* ユーザーズ・ガイド を参照してください。

第 9 章 価格設定資産

価格設定は、カタログ・エントリーの価格、およびその価格を使用するために満たす必要のある基準を表します。機能カタログを作成するには、価格情報をデータベースに追加することが必要です。価格情報を、ローダー・パッケージを使用して、データベースにロード可能な XML ファイルの形式で作成できます。または、価格設定データの量が少ない場合は、 WebSphere Commerce Accelerator の商品管理ツールを使用することもできます。

WebSphere Commerce の価格設定について

以下の図は、WebSphere Commerce Serverにおける価格設定資産を示しています。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、 29 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則については、 311 ページの『付録 A. UML の凡例』を参照してください。

オファー

オファー、つまり価格設定は、同じ商品またはアイテムについて、顧客または組織によって異なる価格を提供するものです。オファーは、カタログ・エントリーの価

格と、その価格で購入するために顧客が満たす必要のある基準 (購入数量など) を表します。たとえば、商品取引またはサービスは、子供、学生、大人、および高齢者によって価格が異なることがよくあります。 WebSphere Commerce では、これは取引位置とも呼ばれており、取引位置コンテナーの一部です。

オファー価格

オファー価格 は、取引条件または契約によってストアがカタログ・エントリーをオファーする価格です。 1 つのオファーでは、複数の通貨で定義された 1 つ以上のオファー価格を指定できます。

取引位置コンテナー

オファーは、メンバーが所有する、*取引位置コンテナー* の一部です。取引位置コンテナーには、取引位置があります。これは、すべての顧客が使用できるようにするか、あるいは取引条件や契約、そして契約の条項によって特定のグループ内の顧客だけが使用できるようにすることができます。契約において、取引位置コンテナーは、価格ビジネス・オブジェクトの 1 つであり、それは複数の価格ビジネス・ポリシーで参照したり、 1 つのストアで、または 1 つのストア・グループ中のすべてのストアで共用したりできます。取引位置コンテナーは、*価格リスト* とも呼ばれます。

使用条件

使用条件 は、取引条件の振る舞いとプロパティーを定義します。ストアの操作のいくつかの局面はビジネス・ポリシーによって定義されているので、使用条件の多くはビジネス・ポリシーを参照します。

価格設定条件の種類

価格設定条件 は、契約に基づいて購入できる商品と顧客がその商品に支払う価額を定義します。契約には、以下の価格設定条件のうち、少なくとも 1 つが必要です。

- *カスタマイズ済み価格リスト* では、販売用の商品リストとその価格の両方が契約内で販売用にカスタマイズされるように指定します。アイテムは、ストア・カタログの特定のセクションに限定されることはなく、ストア・カタログのどこにあるものでも可能です。
- *調整条件付き総合カタログ* は、ストア・カタログの中で利用可能なすべての商品と、ストア・カタログでの定義にしたがって基本価格から何 % 割り引くかの情報とを提供します。調整が指定されていないなら、アイテムは基本価格で販売されます。
- *調整条件付き価格リスト* は、価格リストの中で利用可能なすべての商品と、ストア・カタログでの定義にしたがって基本価格から何 % 割り引くかの情報とを提供します。調整が指定されていないなら、アイテムは基本価格で販売されます。
- *選択的調整条件付き価格リスト* は、調整が価格リスト全体には適用されないという点以外は、調整条件付き価格リストと同じです。調整は、価格リストのサブセットに対してなされます。価格リストのそのサブセットは、商品セットビジネス・ポリシーか、またはカスタマイズ済み商品セットのいずれかです。

取引条件

取引条件 は、契約、RFQ、ビジネス・アカウント、またはオークションです。取引条件は、セラーとバイヤーの間でネゴシエーションされた合意事項であり、それによりバイヤーは、特定のアイテムを、契約の中で指定されている条件およびビジネス・ポリシーで購入できます。たとえば、それにより顧客は、価格設定条件にしたがって、指定された期間に指定された価格でストアから商品を購入することができます。 WebSphere Commerce において、すべての顧客は、契約に従ってストアでの買い物をしなければなりません。ストアでは 1 つ以上の契約をデプロイすることができ、そのうちの 1 つをデフォルトの契約として指定できます。デフォルト契約には、一連のストア・デフォルト・ポリシーに関連する一連の条件が含まれます。取引条件には、それぞれ役割の異なる 0 個以上の参加者が含まれます。

参加者

参加者 は、取引条件または条件の一部となることができます。参加者は、それ自身メンバー・グループや組織などであり得る 1 メンバーです。契約においてバイヤーの役割の参加者が指定されている場合、バイヤーがその契約にしたがって買い物をするためには、バイヤーはバイヤー参加者のメンバーでなければなりません。契約の条件にも、0 個以上の参加者を含めることができます。

参加者の役割

ある参加者の*参加者の役割* は、以下のうちのいずれか 1 つです。

- クリエーター
- セラー
- バイヤー
- サプライヤー
- 承認者
- アカウント・ホルダー
- バイヤー連絡先
- セラー連絡先
- 代理人
- 管理者

契約

商品のオファー価格を含む*契約*。 WebSphere Commerce では、すべての顧客は契約の下でショッピングを行わなければなりません。契約により、顧客は、指定された価格で、指定された期間に渡り、契約に定められている使用条件とビジネス・ポリシーに基づいて、ストアから商品を購入できます。ストアは 0 個以上の契約を所有し、最低 1 つのデフォルト契約を所有しています。

ビジネス・ポリシー

ビジネス・ポリシー とは、ストアまたはストアのグループが従う一連のルールであり、ビジネス・プロセス、業界慣例、およびストアやストア・グループのオファリングの範囲や特性を定義するものです。ビジネス・ポリシーの規則をインプリメントする 1 つ以上のビジネス・ポリシー・コマンド、規則が適用されるビジネス・オ

ブジェクトへの参照、およびビジネス・ポリシー・コマンドの操作を構成する一連のプロパティーの組み合わせでビジネス・ポリシーは施行されます。

価格設定ポリシー

価格設定ポリシー には、価格リストへの参照が含まれており、価格リストに対するビジネス・ポリシーの適用方法を定義する複数のビジネス・ポリシー・コマンドに関連付けることができます。ポリシーは、ストアに対して定義したり、ストア・グループに対して定義したりできます。ストア・グループに関してポリシーを登録すると、そのポリシーはそのグループに含まれるすべてのストアで使用されます。

カタログ・エントリー配送

商品の配送用の包装方法についての情報を含む、*カタログ・エントリー配送* 情報。各カタログ・エントリーには、さまざまな配送情報を定義できます。たとえば、包装時の商品の高さ、重量、長さなどです。

他の価格設定資産

以下の資産が価格設定と関連付けられています。

- *member* は取引位置コンテナーです。取引位置コンテナーの所有者は 1 人だけです。
- WebSphere Commerce Server データベース内のストアを表すストア・エンティティー。
- *カタログ* には、契約の中で参照されるカタログ・エントリーが含まれます。カタログには、オンライン・カタログ用のすべての階層情報およびナビゲーション情報が入っています。カタログは、オンライン・ストアで表示および購入可能な、カタログ・グループとカタログ・エントリーの集合です。
- *カタログ・グループ*、つまりカテゴリーとは、ナビゲーション上の目的およびカタログの区分化の目的で作成される、カタログ・エントリーの一般的なグループ分けのことです。 1 つのカタログ・グループは 1 つのカタログに所属し、複数のカタログ・グループまたはカタログ・エントリーを含むことができます。カタログ・グループは複数のカタログと関連付けることができます。
- *カタログ・エントリー* とは、オンライン・カタログでオーダー可能な商品のことです。カタログ・エントリーはカタログ・グループに所属します。オファーは常に 1 つのカタログ・エントリーに関連付けられます。

WebSphere Commerce Server での価格設定資産の構造の詳細については、WebSphere Commerce オンライン・ヘルプの『価格設定のオブジェクト・モデル』と『データ・モデル』を参照してください。

WebSphere Commerce での価格設定資産の作成

価格設定資産を作成するためには、以下の 2 つの選択肢があります。

- WebSphere Commerce Accelerator の商品管理ツールを使用して価格設定を作成する。 WebSphere Commerce Accelerator のツールを使用する方法は、極めて小さなカタログの価格設定を作成する場合に最適です。

- WebSphere Commerce ローダー・パッケージでロード可能な XML ファイルで価格設定を作成するか、ストア・サービスを介して発行できるストア・アーカイブの一部として価格設定を作成する。この方法は、大量のデータを作成するのに最適です。

WebSphere Commerce Accelerator の商品管理ツールを使用して価格設定を作成する場合の詳細については、 WebSphere Commerce オンライン・ヘルプを参照してください。 XML ファイルで価格設定を作成する場合の詳細については、 『XML ファイルによる価格資産の作成』を参照してください。

XML ファイルによる価格資産の作成

価格設定資産を XML ファイルの形式で作成します。XML ファイルは、ローダー・パッケージを使用して、データベースにロード可能です。ローダー・パッケージについて詳しくは、 225 ページの『第 7 部 ストアの発行』を参照してください。

1. サンプル・ストアの価格設定資産を作成するために使用される XML ファイルを確認します。サンプル・ストアのすべてのファイルは、対応するストア・アーカイブ・ファイルの中にあります。

   ストア・アーカイブ・ファイルは以下のディレクトリーにあります。

   - NT `drive:¥WebSphere¥CommerceServer¥samplestores`
   - 2000 `drive:¥Program Files¥WebSphere¥CommerceServer¥samplestores`
   - AIX `/usr/WebSphere/CommerceServer/samplestores`
   - Solaris `/opt/WebSphere/CommerceServer/samplestores`
   - Linux `/opt/WebSphere/CommerceServer/samplestores`
   - 400 `/qibm/proddata/WebCommerce/samplestores`

     注: WebSphere Commerce オンライン・ヘルプには、サンプル・ストアに含まれる各データ資産についての情報が記載されています。

   各サンプル・ストアには、`offering.xml` ファイルが 2 つ組み込まれています。これには価格設定情報が含まれています。ストア・アーカイブの `offering.xml` ファイルを表示するには、 ZIP プログラムを使用してこれを解凍します。`offering.xml` ファイルは data ディレクトリーにあります。言語ごとの `offering.xml` は、 data ディレクトリーのロケールごとのサブディレクトリーにあります。
2. 313 ページの『付録 B. データの作成』の情報を確認します。
3. サンプル・ストア・アーカイブの `offering.xml` ファイルの 1 つをコピーするか、または新しいファイルを作成することによって、`offering.xml` ファイルを作成します。詳しくは、`offering.xml` に対応する DTD ファイルを参照してください。 DTD ファイルは以下のディレクトリーにあります。

   - NT `drive:¥WebSphere¥CommerceServer¥xml¥sar`
   - 2000 `drive:¥Program Files¥WebSphere¥CommerceServer¥xml¥sar`
   - AIX `/usr/WebSphere/CommerceServer/samplestores`

- Solaris /opt/WebSphere/CommerceServer/xml/sar
- Linux /opt/WebSphere/CommerceServer/xml/sar
- 400 /qibm/proddata/WebCommerce/xml/sar

4. 取引位置コンテナーを作成します。ストアの品物に価格を付けるには、まず取引位置コンテナーを作成しなければなりません。取引位置コンテナーを作成するためには、情報を TRADEPOSCN テーブルに追加します。
   a. 次の例を参考にして、XML ファイルの TRADEPOSCN テーブルに取引位置コンテナーを作成します。

```
<tradeposcn
tradeposcn_id="@tradeposcn_id_101"
member_id="&MEMBER_ID"
markfordelete="0"
name="ToolTech"
precedence="0"

/>
```

   ここで、
   - tradeposcn_id は、生成されるユニーク鍵です。
   - member_id は取引位置コンテナーの所有者です。
   - markfordelete は次のとおりです。
     - 0 = TradingPositionContainer は使用可能。
     - 1 = TradingPositionContainer は削除対象としてマークされており (DBClean ユーティリティーを参照)、使用できない。
   - name は、特定の所有者ごとに固有の、取引位置コンテナーの簡略名です。
   - precedence は、特定の時点に複数の取引位置コンテナーが適格であった場合に、 PRECEDENCE の一番高いコンテナーが使用されることを示します。

5. CATGRPTPC テーブルに情報を追加することによって、マスター・カタログを取引位置コンテナーに関連付けます。マスター・カタログを取引位置コンテナーに関連付ける場合、マスター・カタログのすべてのカタログ・エントリーに標準価格がなければなりません。マスター・カタログの作成の詳細については、 82 ページの『ストア・カタログ資産の表示』を参照してください。
   a. 次の例を参考にして、CATGRPTPC テーブルに情報を追加することにより、マスター・カタログを取引位置コンテナーに関連付けます。

```
<catgrptpc
catalog_id="@catalog_id_1"
tradeposcn_id="@tradeposcn_id_101"
/>
```

   ここで、
   - catalog_id は、マスター・カタログです。
   - tradeposcn_id は、取引位置コンテナーです。

6. OFFER および OFFERPRICE テーブルに情報を追加することによって、カタログ・エントリーにオファーおよびオファー価格を作成します。
   a. 次の例を参考にして、OFFER テーブルに情報を追加することにより、カタログ・エントリーにオファーを作成します。価格を作成する前に、カタログ・

エントリーを作成しておかなければならないことに注意してください。カタログ・エントリーの作成の詳細については、 82 ページの『ストア・カタログ資産の表示』を参照してください。

```
offer
offer_id="@offer_id_138"
startdate="2000-06-19 00:00:00.000000"
catentry_id="@product_id_102"
precedence="0"
published="1"
identifier="1"
flags="1"
tradeposcn_id="@tradeposcn_id_101"
/>
```

ここで、

- offer_id は、生成されるユニーク鍵です。
- startdate は、このオファーが有効である時刻範囲の開始日です。
- catentry_id は、販売用にオファーされるカタログ・エントリーです。
- precedence は、特定の時点で複数のオファーが有効な場合に、PRECEDENCE の一番高いオファーが使用されることを示します。
- published は次のとおりです。
  - 0 = 発行されない (一時的に使用不可)
  - 1 = 発行する
  - 2 = 削除のマーク (発行されない)
- identifier は、指定されたカタログ・エントリーと取引位置コンテナーと一緒に、このオファーを固有に識別する番号です。
- flags は次のとおりです。
  - 1 = shiptoAddressRequired - 1 の場合には、 OrderPrepare は OrderItem がこのオファーを参照していて配送先住所がなければ、エラーを返します。
- tradeposcn_id は、このオファーが含まれている取引位置コンテナーです。

b. 次の例を参考にして、OFFERPRICE テーブルに情報を追加することにより、カタログ・エントリーにオファー価格を作成します。オファー価格は、カタログ・エントリーが販売用にオファーされる際の実際の価格です。価格を作成する前に、カタログ・エントリーを作成しておかなければならないことに注意してください。カタログ・エントリーの作成の詳細については、 82 ページの『ストア・カタログ資産の表示』を参照してください。

```
<offerprice
offer_id="@offer_id_138"
currency="USD"
price="590.00"
/>
```

ここで、

- offer_id は、この価格と関連したオファーです。
- currency は、価格がオファーされる際の通貨です。
- price は、オファーで参照される商品の名目数量 (CATENTSHIP.NOMINALQUANTITY を参照) の価格です。

注: ストアに複数の通貨を表示するには、 OFFERPRICE テーブルの中で通貨ごとに別個の XML エントリーを作成してください。たとえば、カナダ・ドルの通貨を表示するには、新しい XML エントリーの中で `currency="CAD"` を使用します。 `price` の値が変更され、カナダ・ドルでの価格が反映されます。 あるいは、変換を使用することにより、顧客が選択する通貨に基づき、さまざまに異なるレートで表示することもできます。詳細については、 129 ページの『XML ファイルを使用した通貨資産の作成』を参照してください。

c. カタログのすべてのカタログ・エントリーに関して、ステップ a と b を繰り返します。

@ と **&** の使用法の詳細については、 313 ページの『付録 B. データの作成』を参照してください。

第 10 章 契約資産

WebSphere Commerce では、すべてのストアの顧客は契約の下でショッピングを行わなければなりません。契約により、顧客は、指定された価格で、指定された期間に渡り、指定された条件で、ストアから商品を購入できます。ストアのカタログをブラウズするときに顧客が表示できるのは、ストアと結んだ契約範囲内の商品だけです。

ストアと契約していない顧客 (ゲスト・ショッパーなど) がストアでショッピングできるようにしたり、顧客の契約の範囲外の商品を購入できるようにしたりするには、ストアにデフォルト契約 が必要です。

重要

WebSphere Commerce Professional Edition はストアのデフォルト契約しかサポートしません。

ストアのデフォルト契約以外の契約をサポートするのは WebSphere Commerce Business Edition だけです。

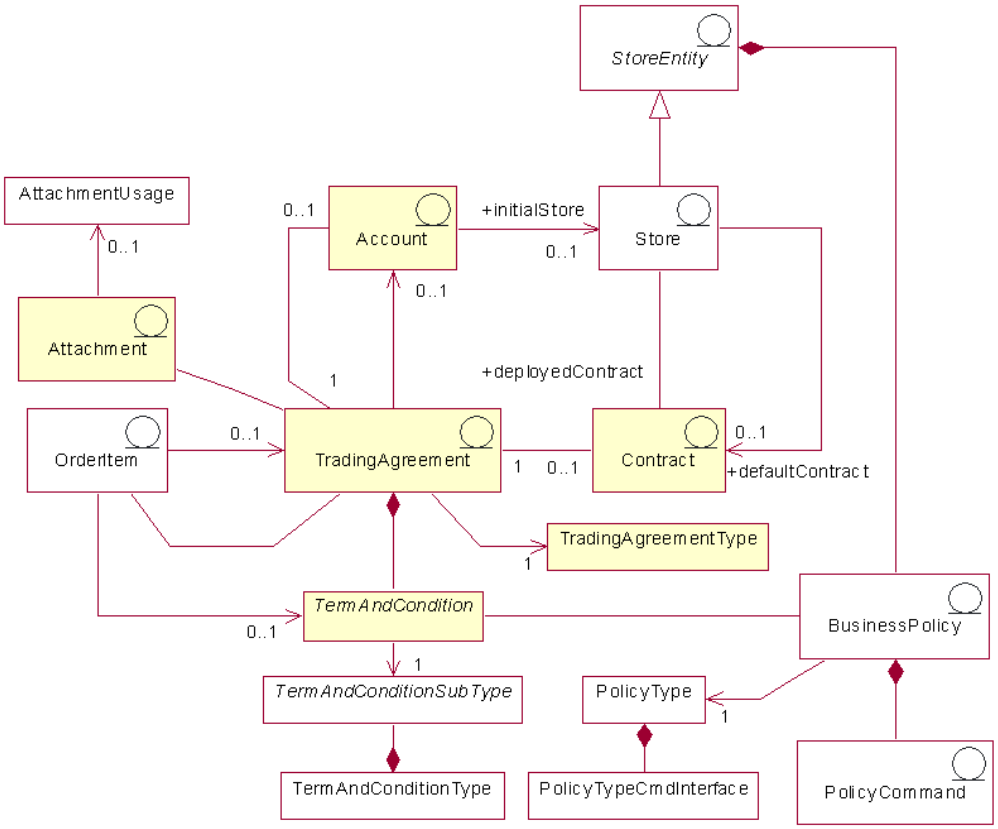
すべての顧客がストアでショッピングできるようにするには、 WebSphere Commerce で作成されたストアに次のものを含めることが必要です。

- ビジネス・ポリシー
- デフォルト契約

ビジネス・ポリシーはデフォルト契約によって参照され、すべての顧客がストアでショッピングすることを許可します。

WebSphere Commerce の契約について

次の図は、WebSphere Commerce における契約の構造を示しています。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルについての詳細は、 29 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則についての詳細は、 311 ページの『付録 A. UML の凡例』を参照してください。

アカウント (ビジネス・アカウント)

ビジネス・アカウントは、バイヤー組織とセラー組織との間の関係を表します。ビジネス・アカウントは、さまざまな取引条件をまとめたり、バイヤーとセラーの間の関係に関連した使用条件を指定したりするために使用できます。たとえば、送り状のカスタマイズ、注文の検査、またはセラーに対するバイヤーの貸付限度額などです。

契約は、バイヤーとセラー間の合意事項を表すので、ビジネス・アカウントと関連付けられます。この例外は、ビジネス・アカウントと関連付けられないストア・デフォルト契約です。ビジネス・アカウントは、多数の契約をアカウントに関連付けることができます。

ビジネス・アカウントは、取引条件の 1 タイプです。取引条件についての詳細は、103 ページの『取引条件』を参照してください。

重要: ビジネス・アカウントは、WebSphere Commerce Business Edition でのみサポートされます。

契約

ストアに関連付けられるアクティブな契約のタイプが 2 つあります。デプロイ契約とデフォルト契約です。デプロイ契約は、特定のバイヤー組織または個々のバイヤーと結ばれるもので、ストアを作成した後に、WebSphere Commerce Accelerator を使用して作成できます。デプロイ契約は、1 つのビジネス・アカウントと関連付けられます。デフォルト契約は、ストアと他の契約を結んでいないバイヤーの場合のストアのデフォルトの振る舞いを定義するものです。デフォルト契約を作成するには XML ファイルを使用する必要があり、1 つのストアに作成できるデフォルト契約は 1 つだけです。契約の詳細については、オンライン情報を参照してください。デフォルト契約資産の作成に関する詳細は、105 ページの『WebSphere Commerce でのデフォルト契約資産の作成』を参照してください。

契約は、取引条件の 1 タイプです。取引条件の詳細については、『取引条件』を参照してください。

取引条件

WebSphere Commerce には、バイヤーとセラーの間の相互作用を制御する取引メカニズムが多数用意されています。以下の取引メカニズムが WebSphere Commerce の異なるエディションでサポートされています。

- オークション (Business Edition と Professional Edition の両方でサポートされます)
- ビジネス・アカウント (Business Edition でのみサポートされます)
- 契約 (この章の前述の制約事項を参照)
- 見積依頼 (RFQ) (Business Edition でのみサポートされます)

これらの取引メカニズムすべてに共通する特性があります。たとえば、どの取引メカニズムにも参加者がいますし、取引メカニズムの振る舞いを制御する規則もあります。取引メカニズムの振る舞いを制御する規則のことを、WebSphere Commerce では*使用条件* と呼びます。

取引条件は、取引メカニズムのインスタンスを表し、取引メカニズムのそのインスタンスのプロパティーを記録します。 WebSphere Commerce の契約、ビジネス・アカウント、および RFQ は、それぞれ取引条件に示されます。 WebSphere Commerce では、すべてのオークションを制御する単一の取引条件があります。

取引条件を構成するのは、TRADING テーブルに保管されるプロファイル、PARTICIPNT テーブルに保管される参加者、TERMCOND テーブルに保管される使用条件、および ATTACHMENT テーブルに汎用リソース ID (URI) として保管されるオプションのアタッチメントです。取引条件は複数のアタッチメントを持つことができるので、アタッチメントは TRDATTACH テーブルを介して取引条件に関連付けられます。アタッチメントは RFQ ではサポートされていないことに注意してください。

一般の取引条件に加えて、各タイプの取引条件に特有の追加情報が、それぞれ独自のテーブルに保管されます。CONTRACT には契約固有の情報、 RFQ には RFQ 固有の情報、ACCOUNT にはビジネス・アカウント固有の情報が保管されます。

使用条件

使用条件は、取引条件の振る舞いとプロパティーを定義します。

契約の場合、使用条件は、バイヤー組織のために契約がどのようにインプリメントされるかを定義します。契約では、契約で販売されるもの、販売されるアイテムの価格、オーダーの支払方法、アイテム返品の処理方法、オーダーの承認方法、およびオーダーの配送元が定義されます。

ストアの操作のいくつかの局面はビジネス・ポリシーによって定義されているので、使用条件の多くはビジネス・ポリシーを参照します。使用条件は、自分が参照するビジネス・ポリシーに対してパラメーターを提供します。ビジネス・ポリシーにパラメーターを提供することによって、各契約ごとにビジネス・ポリシーの振る舞いを変更することができます。

ビジネス・ポリシー

ビジネス・ポリシーとは、ストアまたはストアのグループが従う一連のルールのことです。ビジネス・ポリシーは、ビジネス・プロセス、経営施策、およびストアまたはストアのグループのオファリングの有効範囲と特性を定義します。これは、ストアまたはストアのグループ内で許容され、サポートされる施策の中心的なソースおよび参照テンプレートです。

WebSphere Commerce では、ビジネス・ポリシーの規則をインプリメントする 1 つ以上のビジネス・ポリシー・コマンド、規則が適用されるビジネス・オブジェクトへの参照、およびビジネス・ポリシー・コマンドの操作を構成する一連のプロパティーの組み合わせでビジネス・ポリシーは施行されます。使用条件は、自分が参照するビジネス・ポリシーに対してパラメーターを提供する場合があります。これにより、ビジネス・ポリシーの振る舞いを、ビジネス・ポリシーを参照する使用条件に応じて変更することができます。

アタッチメント

アタッチメントは、取引条件の他の要素が扱っていない、取引条件に関する付加的な情報を提供します。一例としては、RFQ の要件に関する付加的な情報と、RFQ に関するあらゆる一般的な所見とを記したファイルが挙げられます。 1 つの取引条件に複数のアタッチメントがある場合もあります。アタッチメントは WebSphere Commerce の外側に保管され、取引条件はアタッチメントの Universal Resource Identifiers (URI) を保管します。以下に URI の例を挙げます。

- http://www.mycompany.com/information/document1.txt
- file:///home/joeuser/mydocs/document1
- ftp://ftp.mycompany.com/information/attachment.txt

すべてのアタッチメントには、アタッチメントの目的を示す、アタッチメント使用法を割り当てることができます。アタッチメント使用法は、アタッチメントのオプションのプロパティーです。

オーダー・アイテム

オーダー・アイテムとは、オーダーに含まれる商品のことです。 1 つのオーダーに含まれる各オーダー・アイテムを、それぞれ異なる取引条件の下で購入することができます。バイヤーは、ストアの設計に応じて、ショッピング・フローを開始するときか、またはアイテムをオーダーに追加するときのいずれかの時点で、ショッピングを行う際に使用する契約取引条件を選択できます。異なる契約取引条件の下でアイテムを購入する場合、以下の規則が適用されます。

- オーダー内のすべてのアイテムの契約取引条件は、少なくとも 1 つの支払いメソッドを共有していなければなりません。アイテムの契約が支払いメソッドを共有していないと、バイヤーはそのアイテムをオーダーに追加できません。オーダーの支払いでは、オーダー内のすべてのアイテムで共有される支払いメソッドだけが使用できます。
- オーダー内のすべてのアイテムは、同じビジネス・アカウントまたはストア・デフォルト契約に属する契約取引条件からのものでなければなりません。

WebSphere Commerce の契約資産の構造の詳細については、 WebSphere Commerce のオンライン・ヘルプの『契約データ・モデル』を参照してください。

WebSphere Commerce でのデフォルト契約資産の作成

デフォルト契約は、ストアのデフォルトの振る舞いを定義します。すべての契約と同様に、購入可能な商品、価格、支払いメソッド、配送方法、および他のストアの振る舞いを設定することができます。

WebSphere Commerce サンプル・ストアに付属しているストアのデフォルト契約には、以下の事柄を指定する使用条件が含まれています。

- 顧客はストアのマスター・カタログにある購入可のすべての商品を、マスター・カタログの標準価格設定で購入できる (割引や値上げはなし)。
- 配送料がセラー (ストア) に支払われる。
- 顧客は一定の日数以内であれば手数料なしで、購入したものを返品できる。
- 顧客は元の購入で使用したのと同じ支払いメソッドで、リファンドを受け取ることができる。

また、ストアのデフォルト契約の最も一般的なバージョンでは、バイヤーが使用できる支払いメソッドと配送方法を制限する使用条件が省略されています。これらの条件が省略されているため、バイヤーは、購入代金の支払いに、ストアがサポートするデフォルトの支払いメソッドをどれでも使用でき、ストアで用意している配送方法をどれでも使用できます。

デフォルト契約のプロパティーは、その契約の使用条件の中で定義されます。一部の使用条件は、ビジネス・ポリシーを参照します。ビジネス・ポリシーと使用条件の詳細については、オンライン情報を参照してください。

デフォルト契約資産を作成するには、以下のようにします。

1. 使用条件、契約、デフォルト契約、およびビジネス・ポリシーに関する、オンライン情報を確認します。

2. wcs.bootstrap.xml ファイルで定義されるビジネス・ポリシーを確認します。wcs.bootstrap.xml ファイルの詳細については、オンライン情報を参照してください。
3. サンプル・ストアのデフォルト契約資産を作成するために使用されるファイルを確認します。サンプル・ストアのファイルはすべて、対応するアーカイブ・ファイルの中にあります。各サンプル・ストアには、businesspolicy.xml および contract.xml が含まれ、これらには追加のビジネス・ポリシー情報とデフォルト契約情報が含まれます。ストア・アーカイブ・ファイルは以下のディレクトリーにあります。
   - AIX /usr/WebSphere/CommerceServer/samplestores
   - 400 /qibm/proddata/WebCommerce/samplestores
   - Linux /opt/WebSphere/CommerceServer/samplestores
   - Solaris /opt/WebSphere/CommerceServer/samplestores
   - 2000 *drive*:¥Program Files¥WebSphere¥CommerceServer¥samplestores
   - NT *drive*:¥WebSphere¥CommerceServer¥samplestores

   注:

   a. WebSphere Commerce オンライン・ヘルプには、サンプル・ストアに含まれる各データ資産についての情報が記載されています。
   b. ストア・アーカイブの businesspolicy.xml および contract.xml ファイルを表示するには、 ZIP プログラムを使用してこれを解凍します。ファイルは、data ディレクトリーにあります。
   c. WebSphere Commerce Business Edition に付属している ToolTech サンプル・ストアの契約資産ファイルには、ストアのデフォルト契約以外の契約の情報も含まれています。
4. 313 ページの『付録 B. データの作成』の情報を確認します。
5. サンプル・ストア・アーカイブの businesspolicy.xml ファイルの 1 つをコピーするか、新しいファイルを作成することにより、businesspolicy.xml ファイルを作成します。新しいファイルの作成方法の説明は、107 ページの『ビジネス・ポリシー XML ファイルの作成』にあります。説明されているビジネス・ポリシーと異なるビジネス・ポリシーを作成する場合は、 businesspolicy.xml に対応する DTD ファイルを参照してください。 DTD ファイルは以下のディレクトリーにあります。
   - AIX /usr/WebSphere/CommerceServer/xml/sar
   - 400 /qibm/proddata/WebCommerce/xml/sar
   - Linux /opt/WebSphere/CommercServer/xml/sar
   - Solaris /opt/WebSphere/CommercServer/xml/sar
   - 2000 *drive*:¥Program Files¥WebSphere¥CommerceServer¥xml¥sar
   - NT *drive*:¥WebSphere¥CommerceServer¥xml¥sar
6. ローダー・パッケージを使用して businesspolicy.xml ファイルをロードします。ローダー・パッケージについて詳しくは、 225 ページの『第 7 部 ストア

の発行』を参照してください。多文化ストアを作成する場合は、ストアでサポートされているロケールごとに別々の XML ファイルを作成することもできます。説明情報はすべてロケール固有のファイルに指定されるので、これを翻訳するのは容易です。

7. サンプル・ストア・アーカイブの contract.xml ファイルの 1 つをコピーするか、新しいファイルを作成することにより、contract.xml ファイルを作成します。新しいファイルの作成方法の説明は、109 ページの『デフォルト契約 XML ファイルの作成』にあります。より複雑なデフォルト契約を作成する場合は、contract.xml ファイルの構造を定義する B2BTrading.dtd ファイルを確認してください。 B2BTrading.dtd は以下のディレクトリーにあります。
   - AIX /usr/WebSphere/CommerceServer/xml/trading
   - 400 /qibm/proddata/WebCommerce/xml/trading
   - Linux /opt/WebSphere/CommerceServer/xml/trading
   - Solaris /opt/WebSphere/CommerceServer/xml/trading
   - 2000 *drive*:¥Program Files¥WebSphere¥CommerceServer¥xml¥trading
   - NT *drive*:¥WebSphere¥CommerceServer¥xml¥trading
8. ContractImportApprovedVersion コマンドを使用して契約を発行します。詳細は、289 ページの『第 30 章 アカウント、契約、および商品セットの発行』を参照してください。 ContractImportApprovedVersion コマンドについての情報は、オンライン情報の中にもあります。

WebSphere Commerce Business Edition ユーザーは、WebSphere Commerce Accelerator を使用して特定の顧客用の契約を定義できます。特定の顧客用の契約の作成についての詳細は、オンライン情報を参照してください。

ビジネス・ポリシー XML ファイルの作成

ストアのデフォルト契約の使用条件が参照できるよう、WebSphere Commerce には多数のビジネス・ポリシーが備わっていますが、自分で定義しなければならないビジネス・ポリシーもあります。ストアのデフォルト契約条件が参照する、返品の課金、返品の承認、および価格設定に関するビジネス・ポリシーは、自分で定義しなければなりません。これらのビジネス・ポリシー用のコマンドが用意されているので、これらを変更せずに使用できます。独自のビジネス・ポリシーを作成したい場合は、 *IBM® WebSphere® Commerce プログラマーズ・ガイド* を参照してください。

自分のストア用にビジネス・ポリシーを作成するには、ビジネス・ポリシーを作成して、そのビジネス・ポリシーに 1 つ以上のコマンドを関連付ける必要があります。ビジネス・ポリシーを作成するには、POLICY テーブルに情報を追加します。コマンドをビジネス・ポリシーに関係付けるには、POLICYCMD テーブルに情報を追加します。

ビジネス・ポリシーを作成して、そのポリシーにコマンドを関連付けるには、次のようにします。

1. POLICY テーブルに情報を追加することによって、ビジネス・ポリシー XML ファイルにビジネス・ポリシーを作成します。次の例を参考にしてください。

```
<policy
policy_id="@policy_id_10"
policyname="MasterCatalogPriceList"
policytype_id="Price"
storeent_id="@storeent_id_1"
properties="name=InFashion&amp;member_id=&MEMBER_ID"
/>
```

ここで、

- policy_id は、ビジネス・ポリシーに対する固有な数値 ID です。
- policyname は、このビジネス・ポリシーに対する固有名です。
- policytype_id は、定義するポリシーのタイプです。有効な policytype_id は以下のとおりです。
  - InvoiceFormat
  - Payment
  - Price
  - ProductSet
  - ReturnApproval
  - ReturnCharge
  - ReturnPayment
  - ShippingCharge
  - ShippingPayment
- storeent_id は、ストアまたはストア・グループです。
- properties は、ビジネス・ポリシー・コマンドに送信される名前と値の対のリストです。

2. POLICYCMD テーブルに情報を追加することによって、ビジネス・ポリシー XML ファイルでコマンドをビジネス・ポリシーに関連付けます。次の例を参考にしてください。

```
<policycmd
policy_id="@policy_id_10"
businesscmdclass=
  "com.ibm.com.commerce.price.commands.CalculateContractPricesCmdImpl"
/>
```

ここで、

- policy_id は、コマンドに関連付けられているビジネス・ポリシーの数値 ID です。
- businesscmdclass は、ビジネス・ポリシーをインプリメントしている Java クラスの名前です。

businesscmdclass 属性が改行されているのは、表示上の理由に過ぎません。

@ と **&** の使用法の詳細については、 313 ページの『付録 B. データの作成』を参照してください。

デフォルト契約 XML ファイルの作成

デフォルト契約を作成するには、契約、契約所有者、契約の説明、契約の参加者、および契約の使用条件を定義することが必要です。契約情報は、CONTRACT、PARTICIPNT、TRADING、および TERMCOND の 4 つのテーブルに保管されます。

デフォルト契約は、STOREDEF データベース・テーブルを使用してストアと関連付けられます。 WebSphere Commerce Business Edition ユーザーの場合、デフォルト契約以外の契約は STORECNTR データベース・テーブルを使用してストアと関連付けられます。

デフォルト契約を作成するには、以下のようにします。

1. XML ファイルにデフォルト契約を定義します。デフォルト契約は、次のように XML ファイルの先頭で定義されます。

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <!DOCTYPE Trading SYSTEM "B2BTrading.dtd">
   <Trading>
   <Contract state="Active" origin="Manual"
     name="&STORE_IDENTIFIER; Default Contract" majorVersionNumber="1"
     minorVersionNumber="0" contractUsage="Default">
   ```

   Contract エレメントが改行されているのは、表示上の理由に過ぎません。
2. 契約所有者を定義します。次の例を参考にしてください。

   ```
   <ContractOwner>
     <Member>
       <Organization distinguishName="&MEMBER_IDENTIFIER;" />
     </Member>
   </ContractOwner>
   ```

   distinguishName は、契約を所有しているユーザーの LDAP 識別名形式の名前です。たとえば、uid=erickoeck,ou=People,dc=ibm,dc=com などです。
3. 契約 XML ファイルに契約の説明を定義します。次の例を参考にしてください。

   ```
   <ContractDescription title="This is a store default contract." languageId="-1">
   </ContractDescription>
   ```

   ここで、

   - title は、契約のテキスト記述です。
   - languageId はタイトルの言語です。 languageId には、以下の値が事前定義されています。
     - -1 (英語)
     - -2 (フランス語)
     - -3 (ドイツ語)
     - -4 (イタリア語)
     - -5 (スペイン語)
     - -6 (ブラジル・ポルトガル語)
     - -7 (中国語 (簡体字))
     - -8 (中国語 (繁体字))
     - -9 (韓国語)

– -10 (日本語)

ご使用のサイトの言語資産を更新することによって、languageId に追加の値を定義することもできます。言語資産についての詳細は、125 ページの『第 14 章 言語資産』を参照してください。

4. 契約 XML ファイルに契約の参加者を定義します。次の例を参考にしてください。

```
<Participant role="Buyer">
</Participant>
<Participant role="Seller">
  <Member>
    <Organization distinguishName="&MEMBER_IDENTIFIER;"/>
  </Member>
</Participant>
```

distinguishName は、この契約のセラーであるユーザーの LDAP 識別名形式の名前で、たとえば、uid=erickoeck,ou=People,dc=ibm,dc=com などです。多くの場合、これは契約の所有者と同じになります。

注: バイヤー参加者役割にメンバーは指定されません。契約はバイヤー役割を持つすべてのユーザーに使用可能であるためです。

5. 契約 XML ファイルに使用条件を定義します。 XML エレメントおよび属性は、使用条件のタイプによって異なります。各タイプの条件に使用する XML エレメントと属性について学ぶには、B2BTrading.dtd ファイルを使用します。使用条件を定義する際は、一般に以下の属性が使用されます。

policyName
使用条件が参照するビジネス・ポリシーの名前。この名前は POLICY.POLICYNAME に保管されます。

policyType
使用条件が参照するビジネス・ポリシーのタイプ。有効な値は、以下のとおりです。

- Price
- ProductSet
- InvoiceFormat
- Payment
- ReturnApproval
- ReturnCharge
- ReturnPayment
- ShippingCharge
- ShippingMode

storeIdentity
使用条件のストアまたはストア・グループ。

distinguishName
ストアまたはストア・グループを所有しているユーザーの名前。名前は LDAP 識別名形式でなければなりません。たとえば、uid=wcsadmin,o=Root Organization などです。

以下に、サンプル使用条件と、この条件が定義する内容の説明とを列挙します。

- すべてのバイヤーは、ストアのマスター・カタログのすべてのアイテムを、マスター・カタログに設定された価格で購入できます。

```
<TermCondition>
  <PriceTC>
    <PriceTCMasterCatalogWithOptionalAdjustment>
    </PriceTCMasterCatalogWithOptionalAdjustment>
  </PriceTC>
</TermCondition>
```

- バイヤーは配送料をセラーに支払います。

```
<TermCondition>
  <ShippingTC>
    <ShippingTCShippingCharge>
      <PolicyReference policyName="StandardShippingChargeBySeller"
        policyType="ShippingCharge" storeIdentity="&STORE_IDENTIFIER;">
        <Member>
          <User distinguishName="&MEMBER_IDENTIFIER;">
        </Member>
      </PolicyReference>
    </ShippingTCShippingCharge>
  </ShippingTC>
</TermCondition>
```

  PolicyReference エレメントが改行されているのは、表示上の理由に過ぎません。

- バイヤーは、返品の課金なしで商品を返品することができます。商品は、ApprovalByDays ビジネス・ポリシーで定義された日数以内に返品しなければなりません。

```
<TermCondition>
  <ReturnTC>
    <ReturnTCReturnCharge>
      <ReturnChargePolicyReference>
        <PolicyReference policyName="NoCharges"
         policyType="ReturnCharge"
         storeIdentity="&STORE_IDENTIFIER;">
          <Member>
            <Organization distinguishName="&MEMBER_IDENTIFIER;">
          </Member>
        </PolicyReference>
      </ReturnChargePolicyReference>
      <ReturnApprovalPolicyReference>
        <PolicyReference policyName="ApprovalByDays"
         policyType="ReturnApproval"
         storeIdentity="&STORE_IDENTIFIER;">
          <Member>
            <Organization distinguishName="&MEMBER_IDENTIFIER;">
          </Member>
        </PolicyReference>
      </ReturnApprovalPolicyReference>
    </ReturnTCReturnCharge>
  </ReturnTC>
</TermCondition>
```

  PolicyReference エレメントが改行されているのは、表示上の理由に過ぎません。

WebSphere Commerce Business Edition ユーザーへの注意:
ストアのデフォルト契約からこれらの使用条件が省略してあるということは、デ

フォルトでは、ストアが返品を受け付けないことを示します。ただし、他の契約では、返品の使用条件を定義することにより、バイヤーが返品を行えるようにすることができます。

WebSphere Commerce Professional Edition ユーザーへの注意:
ストアのデフォルト契約からこれらの使用条件が省略してあるということは、ストアが返品を受け付けないことを示します。

- リファンドは、オーダーの完了時にバイヤーが使用したのと同じ支払いメソッドで支払われます。

```
<TermCondition>
  <ReturnTC>
    <ReturnTCRefundPaymentMethod>
      <PolicyReference policyName="UseOriginalPayment"
       policyType="ReturnPayment" storeIdentity="&STORE_IDENTIFIER;">
        <Member>
          <User distinguishName="&MEMBER_IDENTIFIER;">
        </Member>
      </PolicyReference>
    </ReturnTCRefundPaymentMethod>
  </ReturnTC>
</TermCondition>
```

PolicyReference エレメントが改行されているのは、表示上の理由に過ぎません。

@ と **&** の使用法の詳細については、 313 ページの『付録 B. データの作成』を参照してください。

第 11 章 フルフィルメント資産

配送センターは、ストアにより、在庫保管庫、および配送受取センターの両方として使用されます。 1 つのストアに対して、1 つ以上の配送センターが関連していることがあります。配送センターは、ストアの商品の在庫と配送を管理します。フルフィルメントには、ピッキング、パッキング、配送が含まれます。ピッキングとは、配送センターからの 1 つ以上のリリースの中から商品を選択することで、パッキングとはそれらの商品を配送コンテナーに入れることで、配送とはそれらを顧客に送ることです。

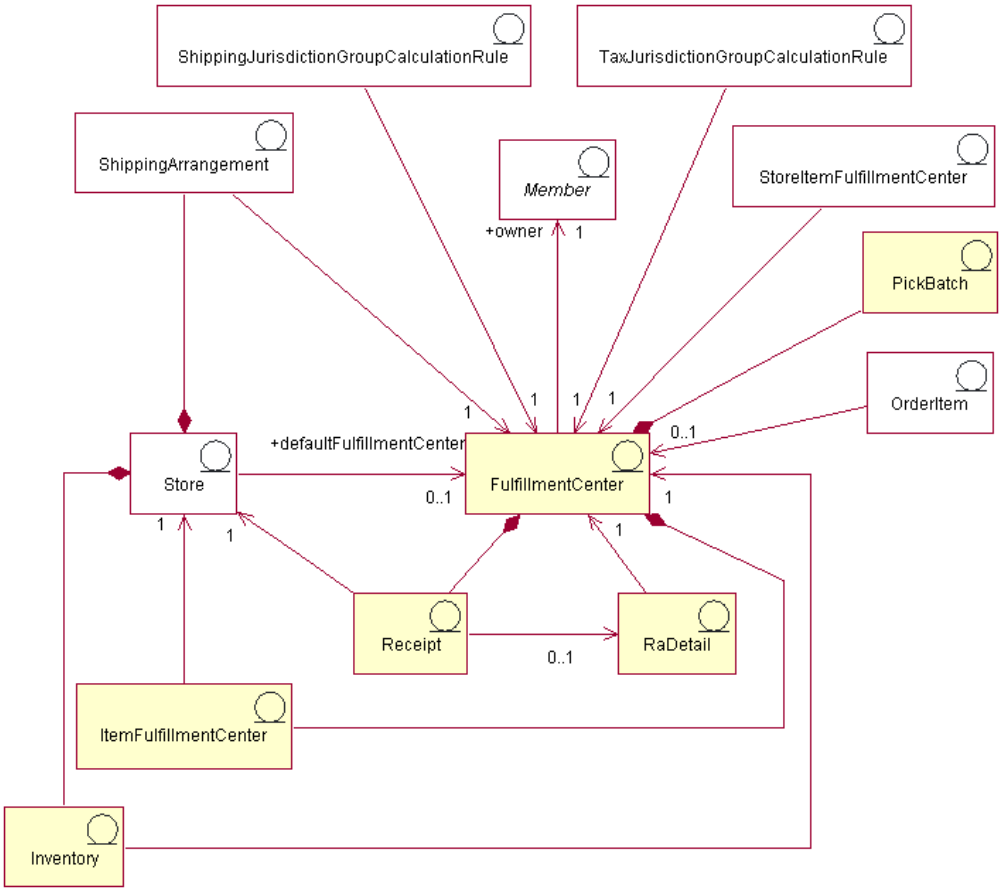
商品のフルフィルメントは、「商品」ウィザードと「商品」ノートブックで構成します。商品の構成には、在庫を追跡記録するオプション、バック・オーダーを許可するオプション、バック・オーダーを強制するオプション、商品を別々にリリースするオプション、および商品を返品不可にするオプションが用意されています。

一般に、配送センターには作業する人が同時に何人かいて、それぞれ 1 つ以上の異なるタスクの実行を担当します。 WebSphere Commerce アクセラレーターは最も一般的なタスクを役割に分け、これらの役割がユーザーに割り当てられます。 WebSphere Commerce アクセラレーターで、フルフィルメントに関する 1 つ以上の役割を割り当てた場合, ログイン時に 1 つの配送センターを選択する必要があります。

注: フルフィルメント、配送センター、および役割の詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

WebSphere Commerce のフルフィルメント資産について

フルフィルメント資産を理解するには、フルフィルメントとストアの間の関係を理解することが必要です。これは、情報モデルを使って説明できます。次に、ストアおよび他の資産と在庫との関係を説明します。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、29 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則の詳細については、311 ページの『付録 A. UML の凡例』を参照してください。

配送センター

上の図では、*配送センター*がフルフィルメントのプロセスの中心にあります。配送センターには、MEMBER テーブルで定義された所有者がいます。各ストアは配送センターと関連しており、デフォルト配送センターが存在します。 1 つの配送センターを複数のストアと関連付けることができます。図に示されているように、ストアと配送センターの間の相互作用がいくつかあります。ストア資産の詳細については、45 ページの『WebSphere Commerce のストア資産について』を参照してください。

受け取り

配送センターは、アイテムの在庫を毎日、毎週、または毎月受け取ります。アイテムの在庫を受け取ると、RECEIPT テーブルに*受け取り*が作成されます。このテーブルは、受け取った数量と、在庫を所有しているストアの情報を記録します。オーダーが処理されると、現在の使用可能な在庫レベルを反映するよう RECEIPT テーブルも更新されます。受け取りの作成についての詳細は、 181 ページの『WebSphere Commerce での在庫資産の作成』を参照してください。

RaDetail

RaDetail は、予測在庫の記録におけるアイテムについての詳細情報です。この情報を使用して、いつ配送センターに在庫が到着するかを見積もったり、バック・オーダーされたアイテムの配送予定日を顧客に知らせたりすることができます。

在庫

ストアには、配送センターと関連した*在庫* があります。在庫には、物理的に報告することのできる、配送センター内にあるすべてのものが含まれます。在庫には 1 つのストアと 1 つの配送センターが関連付けられます。ストアが配送センターに所有している在庫についての情報も記録されます。たとえば予約済みの数量、バック・オーダー中の金額、およびバック・オーダーに割り当てられる金額などです。この情報は、ITEMFFMCTR テーブルに保管されます。在庫および在庫資産の詳細については、 177 ページの『第 21 章 在庫資産』を参照してください。

配送調整

ストアと配送センターとの間の最後の関係には、*配送調整*が関係します。配送調整は、配送センターが、配送方式を使用して、ストアの代わりに商品を配送できることを示します。各ストアは配送センターと配送調整を行い、各配送センターはストアと配送調整を行います。配送調整は、SHPARRANGE テーブルでセットアップされます。配送調整の作成についての詳細は、 151 ページの『配送フルフィルメント資産の作成』を参照してください。

他のフルフィルメント資産

配送センターとの関係の中には、ストアに直接関係しないものもあります。ピッキング・バッチは、1 つの配送センターと関連する関係です。ピッキング・バッチは、処理するオーダー・リリースを配送センターで 1 つの単位としてグループ化し、ピッキング・スリップとパック・スリップを作成します。アイテムがピッキングされ、パックされると、オーダー・リリースを配送できるようになり、配送を確認することができます。ピッキング・バッチ情報は、PICKBATCH テーブルに保管されます。オーダー・アイテムも、1 つの配送センターと関連する関係です。アイテムは、属性によって定義される商品の特定インスタンスです。オーダー内の各アイテムに関する情報は、ORDERITEM テーブルに保管されています。オーダー資産の詳細については、 183 ページの『第 22 章 オーダー資産』を参照してください。

他のエンティティーと同様、配送センターにはそのアクションのいくつかを制御する規則があります。各配送センターには、税および配送料に関する規則があります。これらの規則は、それぞれ TAXJCRULE および SHPJCRULE テーブルで定義されます。税および配送資産の詳細については、 139 ページの『第 18 章 配送資

産』、および 157 ページの『WebSphere Commerce の税資産について』を参照してください。

WebSphere Commerce Server の配送資産の構造に関する詳細は、WebSphere Commerce オンライン・ヘルプの『配送データ・モデル』を参照してください。

WebSphere Commerce での配送資産の作成

品物を提供する 1 つ以上の配送センターを定義しておかないと、ストアは顧客に品物を配送できません。この情報を XML ファイルの形式で作成します。XML ファイルは、ローダー・パッケージを使用して、データベースにロードできます。ローダー・パッケージについて詳しくは、 225 ページの『第 7 部 ストアの発行』を参照してください。

XML ファイルを使用してストアのフルフィルメント資産を作成するには、以下のようにします。

1. サンプル・ストアのフルフィルメント資産を作成するために使用される XML ファイルを確認します。サンプル・ストアのすべてのファイルは、対応するストア・アーカイブ・ファイルの中にあります。

   ストア・アーカイブ・ファイルは以下のディレクトリーにあります。

   - NT `drive:¥WebSphere¥CommerceServer¥samplestores`
   - 2000 `drive:¥Program Files¥WebSphere¥CommerceServer¥samplestores`
   - AIX `/usr/WebSphere/CommerceServer/samplestores`
   - Solaris `/opt/WebSphere/CommerceServer/samplestores`
   - Linux `/opt/WebSphere/CommerceServer/samplestores`
   - 400 `/qibm/proddata/WebCommerce/samplestores`

   注: WebSphere Commerce オンライン・ヘルプには、サンプル・ストアに含まれる各データ資産についての情報が記載されています。

   個々のサンプル・ストアには 1 つの `fulfillment.xml` ファイルが組み込まれており、このファイルにはフルフィルメント情報が組み込まれています。ストア・アーカイブの `fulfillment.xml` ファイルを表示するには、 ZIP プログラムを使ってこれを解凍します。 `fulfillment.xml` ファイルは data ディレクトリーにあります。

2. 313 ページの『付録 B. データの作成』の情報を確認します。
3. サンプル・ストア・アーカイブの `fulfillment.xml` ファイルの 1 つをコピーするか、新しいファイルを作成することにより、`fulfillment.xml` ファイルを作成します。詳しくは、`fulfillment.xml` に対応する DTD ファイルを参照してください。 DTD ファイルは以下のディレクトリーにあります。

   - NT `drive:¥WebSphere¥CommerceServer¥xml¥sar`
   - 2000 `drive:¥Program Files¥WebSphere¥CommerceServer¥xml¥sar`
   - AIX `/usr/WebSphere/CommerceServer/xml/sar`

- Solaris /opt/WebSphere/CommerceServer/xml/sar
- Linux /opt/WebSphere/CommerceServer/xml/sar
- 400 /qibm/proddata/WebCommerce/xml/sar

4. ストアがサポートする 1 つ以上の配送センターを定義します。
   a. 次の例を参考にして、XML ファイルの FFMCENTER テーブルに配送センターを定義します。

```
<ffmcenter
   ffmcenter_id="@ffmcenter_id_1"
   member_id="&MEMBER_ID"
   name="ToolTech Home"
  defaultshipoffset="0"
  markfordelete="0"
   />
```

   ここで、
   - ffmcenter_id は、生成されるユニーク鍵です。
   - member_id は、配送センターの所有者です。
   - name は、所有者と共にこの配送センターを固有に識別するストリングです。
   - defaultshipoffset は、この配送センターからアイテムを配送するのにかかる秒数の見積もりです。この値は、STORITMFFC テーブルでオーバーライドできます。
   - markfordelete は、配送センターが削除されるべきかどうかを次のように示します。 0 = 削除しない。 1 = 使用されなくなったら削除する。詳しくは、WebSphere Commerce オンライン・ヘルプの『データベース・クリーンアップ情報』を参照してください。

   b. 次の例を参考にして、XML ファイルの FFMCENTDS テーブルに配送センターを記述します。多文化ストアを作成する場合は、ロケール固有の XML ファイルにこの情報を組み込む必要があります。

```
<ffmcentds
   ffmcenter_id="@ffmcenter_id_1"
   description="The fulfillment center that supplies products to ToolTech."
   language_id="&en_US"
  displayname="ToolTech Fulfillment"
   staddress_id="@staddress_id_en_US_1"
   />
```

   ここで、
   - ffmcenter_id は、生成されるユニーク鍵です。
   - description は、顧客に対して表示するのに適した配送センターの説明です。
   - language_id は、この情報の表示に使用される言語です。
   - displayname は、顧客に対して表示するのに適した配送センターの名前です。
   - staddress_id は、配送センターの物理的場所です。

   c. ストアがサポートするすべての配送センターについて、ステップ a と b を繰り返します。

@ と **&** の使用法の詳細については、 313 ページの『付録 B. データの作成』を参照してください。

ストア・フルフィルメント資産の作成

ストアに品物を提供する 1 つ以上の配送センターを定義した後、配送センターを、各商品に関連付けることが必要です。つまり、どの配送センターがどの商品を提供するかを識別しなければなりません。この関係を作成するには、INVENTORY テーブルに情報を追加します。この情報を XML ファイルの形式で作成します。XML ファイルは、ローダー・パッケージを使用して、データベースにロードできます。ローダー・パッケージについて詳しくは、 225 ページの『第 7 部 ストアの発行』を参照してください。

注:

1. ストアを配送センターに関連付ける前に、ストア資産を作成することが必要です。ストア資産の作成の詳細については、 47 ページの『XML ファイルによるストア・データ資産の作成』を参照してください。また、ストア・フルフィルメント資産を作成する前に、カタログ資産を作成する必要もあります。詳細は、 82 ページの『ストア・カタログ資産の表示』を参照してください。
2. 非 ATP フルフィルメントをインプリメントしている場合にだけ、ストア・フルフィルメント資産を作成してください。 INVENTORY テーブルは、ATP 機能が組み込まれているストアでは使用されません。

XML ファイルを使用してストア・フルフィルメントの関係を作成するには、以下のようにします。

1. サンプル・ストアのストア・フルフィルメント資産を作成するために使用される XML ファイルを確認します。サンプル・ストアのすべてのファイルは、対応するストア・アーカイブ・ファイルの中にあります。

   ストア・アーカイブ・ファイルは以下のディレクトリーにあります。

   - NT `drive:¥WebSphere¥CommerceServer¥samplestores`
   - 2000 `drive:¥Program Files¥WebSphere¥CommerceServer¥samplestores`
   - AIX `/usr/WebSphere/CommerceServer/samplestores`
   - Solaris `/opt/WebSphere/CommerceServer/samplestores`
   - Linux `/opt/WebSphere/CommerceServer/samplestores`
   - 400 `/qibm/proddata/WebCommerce/samplestores`

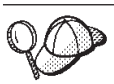
     注: WebSphere Commerce オンライン・ヘルプには、サンプル・ストアに含まれる各データ資産についての情報が記載されています。

   個々のサンプル・ストアには 1 つの `storefulfill.xml` ファイルが組み込まれており、このファイルにはストア・フルフィルメント情報が組み込まれています。ストア・アーカイブの `storefulfill.xml` ファイルを表示するには、 ZIP プログラムを使ってこれを解凍します。 `storefulfill.xml` ファイルは data ディレクトリーにあります。

2. 313 ページの『付録 B. データの作成』の情報を確認します。
3. サンプル・ストア・アーカイブの storefulfill.xml ファイルの 1 つをコピーするか、または新しいファイルを作成することにより、 storefulfill.xml ファイルを作成します。詳しくは、storefulfill.xml に対応する DTD ファイルを参照してください。 DTD ファイルは以下のディレクトリーにあります。
   - NT drive:¥WebSphere¥CommerceServer¥xml¥sar
   - 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥xml¥sar
   - AIX /usr/WebSphere/CommerceServer/xml/sar
   - Solaris /opt/WebSphere/CommerceServer/xml/sar
   - Linux /opt/WebSphere/CommerceServer/xml/sar
   - 400 /qibm/proddata/WebCommerce/xml/sar
4. 次の例を参考にして、XML ファイルの INVENTORY テーブルに情報を追加することにより、ストアと配送センターの関係を作成します。

```
<inventory
catentry_id="@catentry_id_1470"
quantity="100"
ffmcenter_id="@ffmcenter_id_1"
store_id="@storeent_id_1"
quantitymeasure="C62"
inventoryflags="0"
/>
```

   ここで、
   - catentry_id は、この配送センターが提供するカタログ・エントリーです。
   - quantity は、この配送センターから使用できる数量です (QUANTITYMEASURE に示された単位による)。
   - ffmcenter_id は、在庫を提供する配送センターです。
   - store_id は、在庫が供給されるストアです。
   - quantitymeasure は、QUANTITY の計測単位です。
   - inventoryflags は、QUANTITY の使用法を示す以下のビット・フラグです。
     - 1 = noUpdate. デフォルトの UpdateInventory タスク・コマンドは QUANTITY を更新しません。
     - 2 = noCheck. デフォルトの CheckInventory および UpdateInventory タスク・コマンドは QUANTITY をチェックしません。
5. ストアの中の各カタログ・エントリーごとにステップ 3 を繰り返します。



@ と **&** の使用法の詳細については、 313 ページの『付録 B. データの作成』を参照してください。

第 12 章 キャンペーン資産

キャンペーンは、組織的にマーケティング活動を展開するのに役立ちます。多くの場合、キャンペーンは、マーケティング・マネージャーまたは取引管理マネージャーのいずれかによって作成されます。それには、しばしば特定の目標が関係しています。たとえば、「学校へ戻ろう」というキャンペーンであれば、キャンペーン中の子供服の販売を促進するという目標が関係しています。

WebSphere Commerce のキャンペーンについて

WebSphere Commerce の中でのキャンペーンには、任意の数のキャンペーン・イニシアチブが含まれており、それによって 1 つの条件が定義されます。キャンペーン・イニシアチブは、定義された条件が真になった場合に、顧客を対象としたコンテンツを生成します。その結果として、キャンペーンはさまざまなイニシアチブを編成するためのハイレベルなマーケティング・エレメントということになります。

キャンペーン・イニシアチブは、一連のイニシアチブを含むキャンペーンに関連付けられます。その関係を示す例として、オフィス・サプライ品を扱うあるストアで、「学校へ戻ろう」キャンペーンを繰り広げるとします。その場合、イニシアチブは、登録されている顧客のうち職業欄が学生となっている顧客を対象として、ペンの割引を宣伝したりレポート用紙の購入を勧めたりするなどの低レベルのアクションを担当します。

キャンペーン・イニシアチブは、以下の 3 種類の動的コンテンツを表示できます。

- お勧め商品提示商法
- 協調フィルタリング・ベースの商品提示法
- 顧客キャッチ広告

お勧め商品提示商法のコンテンツは、特定の顧客を対象として、その顧客のプロファイルやその他の振る舞いに基づいたルール・ベースの商品推奨を提供することを意図して設計されたものです。そのようなコンテンツを表示するイニシアチブは、抱き合わせ販売や高額商品販売の機会を作り出すことを意図したものです。

協調フィルタリング・ベースの商品提示法も商品推奨を作成することを意図したものですが、使用する推奨アルゴリズムが少し異なっています。この場合、事前に定義されたルールではなく顧客の振る舞い全体に基づいて対象商品が決められます。

顧客キャッチ広告は、お勧め商品提示商法と同じ基準に基づき、特定の顧客を対象として宣伝コンテンツを提供します。しかし、顧客キャッチ広告は、オンライン・ストアのさまざまなアクティビティーに関する顧客の認知度を高めたり、特別なオファーを強調したり、ブランドの認知度を高めたりすることを意図したものです。

イニシアチブは、サイトのどのページにも組み込むことができます。サイトの設計時に、 e-マーケティング・スポットと呼ばれる特別なプレースホルダーがサイト上に配置されます。顧客に対して表示される時点でそれらのプレースホルダーは、対

象となる特定のコンテンツに置き換えられます。その位置は、 e-マーケティング・スポットの中の希望する位置にイニシアチブを表示するようスケジューリングすることによって割り当てます。

キャンペーン・イニシアチブには、表示するタイミングと対象を決定する条件が含まれます。その条件は、イニシアチブが作成された時点で定義され、イニシアチブの存続期間中にイニシアチブの可視性や表示内容を調整するために変更することも可能です。

キャンペーン・イニシアチブでは、その使用に関する統計が生成されます。それらの統計情報は、マーチャント、マーケティング・マネージャー、および取引管理マネージャーにより、 WebSphere Commerce アクセラレーターを使用して表示できます。統計は、イニシアチブがインプリメントされている各 e-マーケティング・スポットごとに、そのイニシアチブの閲覧率を示します。これらの統計は、イニシアチブの効果や、これが表示される様々な場所における比較成功率についてのフィードバックを示します。

WebSphere Commerce でのキャンペーン資産の作成

多くの場合、キャンペーンとキャンペーン・イニシアチブは、マーケティング・マネージャーによって、または WebSphere Commerce Accelerator において「キャンペーン」ウィザードや「キャンペーン・イニシアチブ」ウィザードを使って取引管理マネージャーによって作成されます。詳しくは、WebSphere Commerce オンライン・ヘルプを参照してください。

第 13 章 支払資産

WebSphere Commerce では、IBM Payment Manager がサポートされています。ストアのための支払資産を作成するには、ストアで Payment Manager を使用するかどうかを指定し、使用する場合には、ストアの受け付ける決済カセットとブランドの種類を指定します。

その情報は、次のようにして指定します。

- ストア発行時にストア・サービスを使ってロードされる支払データを、 XML ファイルの形式 (paymentinfo.xml) で作成する。これにより、発行するストアについて指定されているマーチャントおよびブランドのタイプに対応するよう、Payment Manager が構成されます。詳細は、『XML ファイルを使用した支払資産の作成』を参照してください。

  注: paymentinfo.xml は、 WebSphere Commerce Server データベースの中のテーブルにデータを入れるわけではありません。 Payment Manager を構成します。 paymentinfo.xml が適用されるのは、支払いメソッドとしてオフライン・クレジット・カードを使用している場合だけです。他の支払方法を構成する場合は、次の項目を参照してください。
- 管理コンソールまたは Payment Manager のユーザー・インターフェースを使うことによって、 Payment Manager をストアに合わせてセットアップする。管理コンソールを使用する場合には、 Payment Manager のメニューにメニュー項目が表示されます。 Payment Manager のユーザー・インターフェースを使用する場合には、ナビゲーション・フレームの中の「管理」の下にメニュー項目が表示されます。詳しくは、WebSphere Commerce のオンライン・ヘルプのトピック 『*Setting up Payment Manager for your store*』を参照してください。

XML ファイルを使用した支払資産の作成

XML ファイルを使用してストアの支払資産を作成するには、以下のようにします。

1. サンプル・ストアの支払資産を作成するために使用される XML ファイルを確認します。サンプル・ストアのすべてのファイルは、対応するストア・アーカイブ・ファイルの中にあります。

   ストア・アーカイブ・ファイルは以下のディレクトリーにあります。

   - NT `drive:¥WebSphere¥CommerceServer¥samplestores`
   - 2000 `drive:¥Program Files¥WebSphere¥CommerceServer¥samplestores`
   - AIX `/usr/WebSphere/CommerceServer/samplestores`
   - Solaris `/opt/WebSphere/CommerceServer/samplestores`
   - Linux `/opt/WebSphere/CommerceServer/samplestores`
   - 400 `/qibm/proddata/WebCommerce/samplestores`

注: WebSphere Commerce オンライン・ヘルプには、サンプル・ストアに含まれる各データ資産についての情報が記載されています。

各サンプル・ストアには 1 つの paymentinfo.xml ファイルが組み込まれています。これには支払いに関する情報が入っています。ストア・アーカイブの paymentinfo.xml ファイルを表示するには、 ZIP プログラムを使ってこれを解凍します。 paymentinfo.xml ファイルは、data ディレクトリーにあります。

2. サンプル・ストア・アーカイブの paymentinfo.xml ファイルの 1 つをコピーするか、新しいファイルを作成することにより、paymentinfo.xml ファイルを作成します。詳しくは、paymentinfo.xml に対応する DTD ファイルを参照してください。 DTD ファイルは以下のディレクトリーにあります。
   - NT drive:¥WebSphere¥CommerceServer¥xml¥sar
   - 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥xml¥sar
   - AIX /usr/WebSphere/CommerceServer/xml/sar
   - Solaris /opt/WebSphere/CommerceServer/xml/sar
   - Linux /opt/WebSphere/CommerceServer/xml/sar
   - 400 /qibm/proddata/WebCommerce/xml/sar
3. Payment Manager を使用可能または使用不可にします。
   a. 次の例を参考にして、XML ファイルの中で Payment Manager を使用可能または使用不可にし、ストアの受け付ける決済カセット、通貨、およびブランドの種類を指定します。

```
<paymentinfo>
<PaymentManager enable="yes"/>
<Cassette type="OfflineCard">
<Account currency="USD">
<Brand type="MasterCard"/>
<Brand type="VISA"/>
<Brand type="American Express"/>
<Account/>
<Account currency="EUR">
<Brand type="MasterCard"/>
<Brand type="VISA"/>
<Brand type="American Express"/>
</Account>
</Cassette>
</paymentinfo>
```

ここで、

- enable には、Payment Manager を使用可能にするか、それとも使用不可にするかを指定します。
- Cassette type は、サポートされるカセットの種類です。
- Account currency は、ストアでサポートする通貨です。 OfflineCard カセット・タイプを使用している場合には、アカウント通貨が必要です。通貨は、ISO 4217 規格に従って 3 文字のコードで指定します。たとえば、米国ドルの場合は ″USD″ です。
- Brand type は、指定されたアカウントと通貨でサポートされるクレジット・カードの種類です。

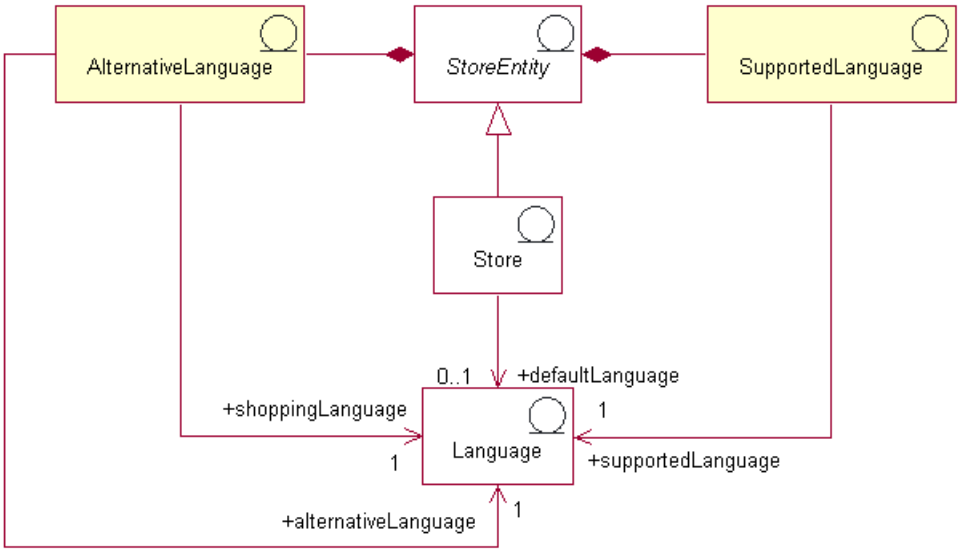
第 14 章 言語資産

WebSphere Commerce では、サイトに多数の使用できる言語を定義することができます。 LANGUAGE テーブルは、サポートされる 10 言語を定義します。すなわち、ドイツ語、中国語 (繁体字)、中国語 (簡体字)、日本語、韓国語、イタリア語、フランス語、スペイン語、ブラジル・ポルトガル語、英語です。サイトでは、さまざまな文化や地域の顧客に対する情報の表示方法を調整するために、さらに付加的な言語や既存の言語の方言を定義できます。

WebSphere Commerce の言語資産について

言語資産を理解するには、言語とストアの間の関係を理解することが必要です。これは、以下の情報モデルを使って説明できます。次に、ストアおよび他の資産と言語との関係、および関連を説明します。

以下の図は、言語資産情報モデルを示しています。



WebSphere Commerce の言語には 4 つの種別があります。デフォルト言語、サポートされる言語、代替言語、およびショッピング言語です。これらの種別のそれぞれは、ストアで異なる役割を果たします。すべての言語は、LANGUAGE テーブルに保管されます。

デフォルト言語

*デフォルト言語*は、各ストアに関連付けられます。これは、ストアが主な言語として使用するよう選択した言語であり、ショッピング言語を明示的に選択しない顧客に対して表示される言語になります。ストアのデフォルト言語は、暗黙のうちにストアによってサポートされます。つまりストアでは、デフォルト言語 (または LANGPAIR テーブルで定義されている場合には、その代替言語のいずれか) による

情報の表示が常に可能でなければなりません。サポートされている言語または代替言語のいずれかで情報が利用できない場合、情報はデフォルト言語で表示されます。

サポートされる言語

STORELANG テーブルは、各ストアでサポートする言語を指定します。ストアでは、*サポート言語*、または LANGPAIR テーブルで定義されている場合には、その代替言語のいずれかで情報を表示することが可能でなければなりません。またストアは、そのストア・グループでサポートされているすべての言語をサポートする必要があります。

代替言語

サポートされる言語のいずれかで情報を表示できない場合、ストアは、*代替言語* による情報の表示を試みます (使用可能な場合)。ストアでは、代替言語を 1 つ 1 つを試す順序を指定できます。ストアの代替言語には、そのストア・グループの代替言語が含まれます。代替言語は、情報の一部が 1 つの言語でしか利用できないが、それとは異なる関連言語で買い物をする顧客にも表示する必要があるという場合に便利です。たとえば、情報をサポートされているすべての言語に翻訳する作業がまだ完了していない場合、あるいは同じ言語の 2 つの類似した方言がサポートされていて、情報が同一の場合などです。

WebSphere Commerce Server の言語資産の構造の詳細については、WebSphere Commerce オンライン・ヘルプで『言語オブジェクト・モデル』および『データ・モデル』を参照してください。

WebSphere Commerce での言語資産の作成

次の方法のうち 1 つを使って、ストアがサポートする言語を定義できます。

- ストア・サービスのツールを使用する
- XML ファイルを使用する。XML ファイルは、ローダー・パッケージでロードすることも、ストア・サービスの発行ツールでロードすることもできます。

注: ストア・サービス・ツールは、事前に読み込み済みの XML ファイルを、ストア・アーカイブ形式で処理します。

ストアがサポートする言語をストア・サービスを使って定義する方法の詳細については、 WebSphere Commerce オンライン・ヘルプを参照してください。ストアがサポートする言語を XML ファイルに定義する方法の詳細については、 47 ページの『XML ファイルによるストア・データ資産の作成』を参照してください。

第 15 章 通貨資産

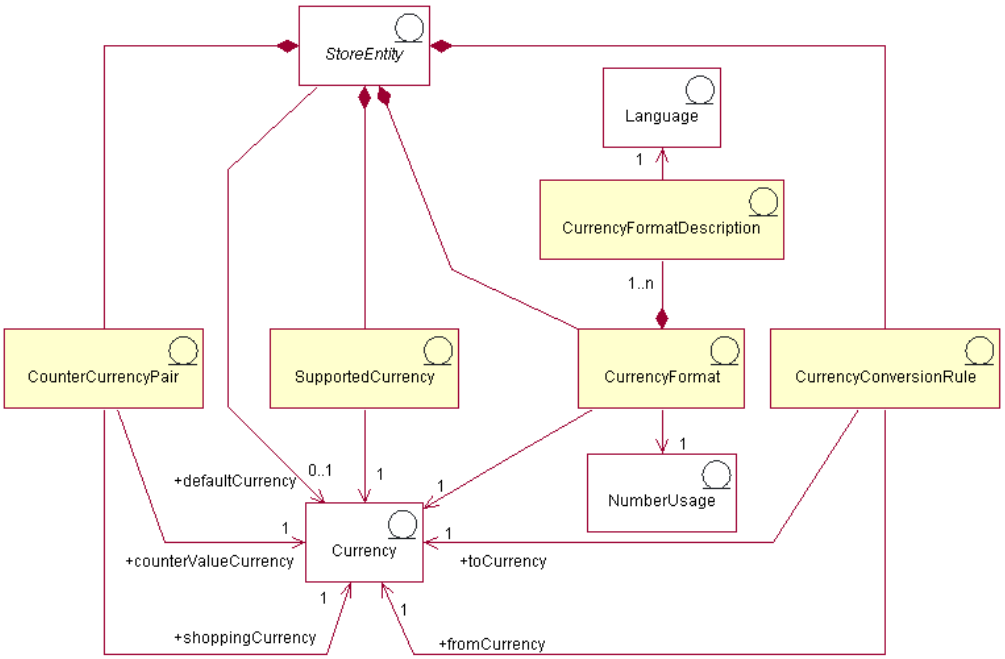
サイトでは、1 種類の通貨で価格を表示することもできますが、ユーロ用に用意された指示に従って複数の通貨を使用することも可能です。複数のストアを擁するサイトでは、ストアによって異なる通貨を使用したり、ストア・グループに通貨を割り当てたりすることができます。作成しようとしているサイトの特性によっては、使用したい通貨とその表示法を指定することができます。

WebSphere Commerce では、顧客がショッピング通貨を選択できるようにすることができます。ショッピング通貨は、顧客が特定のストアで商品に支払う通貨です。ストア・ページ上のすべての金額はこの通貨で表示されます。顧客がショッピング通貨を変更すると、ショッピング・カートに追加したアイテムの価格とオーダー合計価格は、自動的に新しいショッピング通貨に換算され、再計算されて表示されます。

顧客は、ユーロを含む多くの通貨でショッピングできます。ユーロは 1999 年 1 月 1 日に正規の通貨になり、現在は金融市場で使用されています。ユーロ通貨とすべての参加国の通貨の換算率 (為替レート) は固定されています。ヨーロッパ連合の将来の統一通貨の紙幣とコインは、2002 年 1 月 1 日に使用可能になります。 6 か月後に、現在の各国の通貨は永久に流通できなくなります。 1999 年から 2001 年の移行期間は、マーチャントは各国の通貨とユーロの両方を扱わなければなりません。

WebSphere Commerce の通貨資産について

以下の図は、WebSphere Commerce Server における通貨構造を示しています。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、 29 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則についての詳細は、 311 ページの『付録 A. UML の凡例』を参照してください。

上の図では、通貨が情報モデルの中心にあります。各ストア、またはストアのグループには、デフォルト通貨があります。

通貨形式

ストア・エンティティーには、多数の*通貨形式設定* のルールを設定できます。ストアに特定通貨の形式設定ルールがない場合は、そのストア・グループの形式設定ルールが使用されます。通貨形式は、CURFORMAT テーブルでセットアップされます。

数値の使用法

形式設定された各通貨ルールには、*数値の使用法* が 1 つ関連付けられます。数量や金額などの数値は、それに関連する使用法に応じて、さまざまに丸めたり形式設定したりできます。ストアは、表示する数値の使用方法に応じて、それぞれの数値に異なる丸めと形式設定のルールを指定することができます。たとえば、あるストアは単価については単価の使用法を指定して小数第 4 位で丸め、他の通貨の額についてはデフォルトの使用法を指定して小数第 2 位で丸めるかもしれません。数値の使用法は、NUMBRUSG テーブルの中に保存されます。

通貨形式の説明

通貨形式のルールには、多数の*通貨形式記述* を設定できます。通貨形式記述は、ある数量を (表示目的で) 特定言語の特定の数量単位で形式設定する方法を説明します。各説明は、LANGUAGE テーブルで言語と関連付けられます。言語資産の詳細については、 125 ページの『第 14 章 言語資産』を参照してください。通貨形式記述は、CURFMTDESC テーブル保存されます。

サポートされる通貨

ストア・エンティティーには、多数のサポートされる*通貨* を設定できます。サポートされる通貨とは、支払いを受け取る通貨です。

通貨変換ルール

すべての通貨には、他の通貨との間で変換を制御するルールがあります。各*通貨変換ルール* は、価格 (特定通貨のデータベースに保管されている) を変換し、サポートされているショッピング通貨による顧客への請求額を算出するために使用できます。

カウンター通貨

カウンター通貨 は、サポートされる通貨に伴って表示される通貨金額です。これは購入では使用できませんが、情報表示の目的で使用されます。顧客は、ユーロでショッピングすることにした場合、欧州通貨統合の通貨と他の通貨で金額をストアに

表示することができます。ショッピング通貨の金額は、そのショッピング通貨に対応するすべてのカウンター値通貨に変換されます。カウンター通貨は、オランダ・ギルダーとユーロのように、サポートされる通貨と対にされます。カウンター通貨の対は、CURCVLIST に保管されます。

WebSphere Commerce Server の通貨資産の構造の詳細については、WebSphere Commerce オンライン・ヘルプの『通貨データ・モデル』を参照してください。

WebSphere Commerce での通貨資産の作成

WebSphere Commerce のストア・サービス・ツールを使用すると、サポートされる通貨をストアに追加したり、ストアのデフォルト通貨を選択したりできます。ストア・サービス・ツールを使用して編集できる資産の詳細については、 WebSphere Commerce オンライン・ヘルプのトピック『ストア・データベース資産の変更』 を参照してください。

注: ストア・サービス・ツールは、事前に読み込み済みの XML ファイルを、ストア・アーカイブ形式で処理します。

また、サポートされる通貨とデフォルト通貨を XML ファイルを使用してストアに追加することもできます。そのファイルは、ローダー・パッケージを使用してデータベースにロードできます。この方法では、通貨換算やカウンター値通貨など、別の種類の通貨資産の作成も可能です。

既存のストア・アーカイブの通貨資産を編集することに関する情報は、 WebSphere Commerce のオンライン・ヘルプを参照してください。新しい通貨資産を XML ファイルの形式で作成することに関する情報は、 『XML ファイルを使用した通貨資産の作成』を参照してください。

XML ファイルを使用した通貨資産の作成

ストアの通貨資産は、XML ファイルの形式で作成します。その XML ファイルは、ローダー・パッケージを使用して、データベースにロードできます。ローダー・パッケージについて詳しくは、 225 ページの『第 7 部 ストアの発行』を参照してください。

XML ファイルを使用してストアの通貨資産を作成するには、以下のようにします。

1. サンプル・ストアの通貨資産を作成するために使用される XML ファイルを確認します。サンプル・ストアのすべてのファイルは、対応するストア・アーカイブ・ファイルの中にあります。

   ストア・アーカイブ・ファイルは以下のディレクトリーにあります。

   - NT `drive:¥WebSphere¥CommerceServer¥samplestores`
   - 2000 `drive:¥Program Files¥WebSphere¥CommerceServer¥samplestores`
   - AIX `/usr/WebSphere/CommerceServer/samplestores`
   - Solaris `/opt/WebSphere/CommerceServer/samplestores`
   - Linux `/opt/WebSphere/CommerceServer/samplestores`

- 400 /qibm/proddata/WebCommerce/samplestores

  注: WebSphere Commerce オンライン・ヘルプには、サンプル・ストアに含まれる各データ資産についての情報が記載されています。

個々のサンプル・ストアには 1 つの currency.xml ファイルが組み込まれており、このファイルには通貨情報が組み込まれています。ストア・アーカイブの currency.xml ファイルを表示するには、 ZIP プログラムを使用してこれを解凍します。 currency.xml ファイルは data ディレクトリーにあります。

2. 313 ページの『付録 B. データの作成』の情報を確認します。
3. サンプル・ストア・アーカイブの currency.xml ファイルの 1 つをコピーするか、新しいファイルを作成することにより、currency.xml ファイルを作成します。詳しくは、currency.xml に対応する DTD ファイルを参照してください。DTD ファイルは以下のディレクトリーにあります。
   - NT drive:¥WebSphere¥CommerceServer¥xml¥sar
   - 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥xml¥sar
   - AIX /usr/WebSphere/CommerceServer/xml/sar
   - Solaris /opt/WebSphere/CommercServer/xml/sar
   - Linux /opt/WebSphere/CommercServer/xml/sar
   - 400 /qibm/proddata/WebCommerce/xml/sar
4. ストアでサポートされる通貨を定義します。
   a. 次の例を参考にして、 XML ファイルの CURLIST テーブルでストアがサポートする通貨を定義します。

      ```
      <curlist currstr="USD" storeent_id="@storeent_id_1" />
      ```

      ここで、
      - currstr は、サポートされる通貨を表す 3 文字の ISO 4217 通貨コードです。このコードは、 SETCURR テーブルの SETCCURR 列に出現するものでなければなりません。ストアはサポートされているすべての通貨での支払いを受諾できなければなりません。
      - storeent_id は、ストア・エンティティーです。

   b. ストアがサポートする各通貨ごとに繰り返します。

ストアのデフォルト通貨は、STOREENT テーブルで定義されます。詳細については、 47 ページの『XML ファイルによるストア・データ資産の作成』を参照してください。

5. (オプション) ストアの価格がどんな通貨で表示されるかは、価格の設定方法によって異なります。ストアで使用するあらゆる通貨に対して価格を定義したり、デフォルト通貨だけについて価格を定義したりできます。価格の設定については、96 ページの『WebSphere Commerce での価格設定資産の作成』を参照してください。

価格の設定においてデフォルト通貨だけについて価格を定義した場合に、ストアにおいてサポートされている他の通貨で価格を表示したいなら、換算率をストアに追加する必要があります。その換算率を使用することにより、デフォルト通貨からサポートされる通貨に変換します。

a. 換算前の通貨 (たとえば米国ドル USD) と、 1 つ以上の換算後の通貨 (たとえば日本円 JPY) とを決定します。各通貨の ISO 通貨コードについては、ISO 4217 の国際通貨についての情報を参照してください。

b. 次の例を参考にして、CURCONVERT テーブルに情報を追加します。

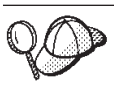
```
<curconvert
storeent_id="@storeent_id_1"
fromcurr="USD"
tocurr="JPY"
factor="105.10"
multiplyordivide="M"
bidirectional="Y"
updatable="Y"
curconvert_id="@curconvert_id_1" />
```

ここで、

- storeent_id は、ストア・エンティティーです。
- fromcurr は、換算前の通貨です。通常、FROMCURR 通貨の金額は、価格、割引、配送料など、販売対象商品に関連した金額を決定するために使用されるルールやその他の情報の一部です。
- tocurr は、換算後の通貨です。通常、TOCURR は顧客が支払う通貨です。多くの場合、この通貨の金額は、単価、配送料、税額など、オーダー・アイテムの一部です。
- factor は、換算係数です。
- multiplyordivide は、FROMCURR から TOCURR への換算に関して、次のように指定します。
  - M = FACTOR を乗算する。
  - D = FACTOR で除算する。

  双方向ルールの場合、逆の操作を使用して TOCURR から FROMCURR に換算されます。
- bidirectional は、ルールが双方向か単一方向かを指定します。
  - Y = 双方向
  - N = 単一方向
- updatable は、通貨換算ルールを管理するユーザー・インターフェースで使用することを意図したフラグです。有効な値は、以下のとおりです。
  - N = 換算率は変更不能 - 決して変更できません。
  - Y = 換算率は変更可能です。
- curconvert_id は、生成されるユニーク鍵です。

c. 価格を表示するすべての通貨について、ステップ a と b を繰り返します。



価格設定情報の中で、サポートされているすべての通貨について価格を定義した場合であっても、ストアでサポートされる通貨のための通貨換算率を定義できます。

6. (オプション) ショッピング通貨とカウンター通貨の両方に表示価格を含めたいなら (たとえばオランダのギルダーとユーロの両方による表示価格の場合)、CURCVLIST テーブルに情報を追加する必要があります。
   a. 次の例を参考にして、CURCVLIST テーブルに情報を追加します。

```
<curcvlist
storeent_id="@storeent_id_1"
currstr="NLG"
countervaluecurr="EUR"
displayseq="1" />
```

   ここで、

   - storeent_id は、ストア・エンティティーです。
   - currstr は、通貨を表す three 文字の ISO 4217 通貨コードです。このコードは、 SETCURR テーブルの SETCCURR 列に出現するものでなければなりません。通常、FROMCURR 通貨の金額は、価格、割引、配送料なと、販売対象商品に関連した金額を決定するために使用されるルールやその他の情報の一部です。
   - countervaluecurr は、カウンター値通貨を表す three 文字の ISO 4217 通貨コードです。このコードは、 SETCURR テーブルの SETCCURR 列に出現するものでなければなりません。
   - displayseq は、カウンター値通貨の表示順序を指定する数値です。カウンター値通貨は、 CURCVLIST テーブルの DISPLAYSEQ 列で指定されるカウンター値表示順序に従い、昇順で表示されます。

@ と **&** の使用法の詳細については、 313 ページの『付録 B. データの作成』を参照してください。

通貨に関するその他のタスク

通貨や通貨に関する以下のタスク、

- 現在のところ WebSphere Commerce でサポートされていない新しい通貨を追加する
- 既存の通貨形式を変更する
- 新しい通貨や特定のストアのために新しい形式設定ルールを追加する

WebSphere Commerce オンライン・ヘルプを参照してください。

第 16 章 計測単位資産

商品の販売、在庫の追跡は、さまざまな数量単位 (キログラム、インチ、リットルなど) で行うことができます。商品は、これらの単位の最小数量や、それに特定数量を掛けた数でオーダーすることができます。

コントローラー・コマンドは、UOM (計測単位) を使用して数量単位を指定します。 UOM パラメーターが指定されていない場合、顧客が指定した数量に、CATENTSHIP データベース・テーブル内のカタログ・エントリーの名目数量が掛けられます。結果は要求された数量として認識されます。

要求数量は、カタログ・エントリーの、一番近い倍数に切り上げられます。たとえば、倍数が 2 kg で要求数量が 4.1 kg の場合、切り上げの結果は 6 kg になります。切り上げ済み数量は在庫の検査時に使用されます。在庫には、独自の数量単位があります。在庫の数量単位とカタログ・エントリーの数量単位が異なる場合、 2 つの単位間での変換が必要です。

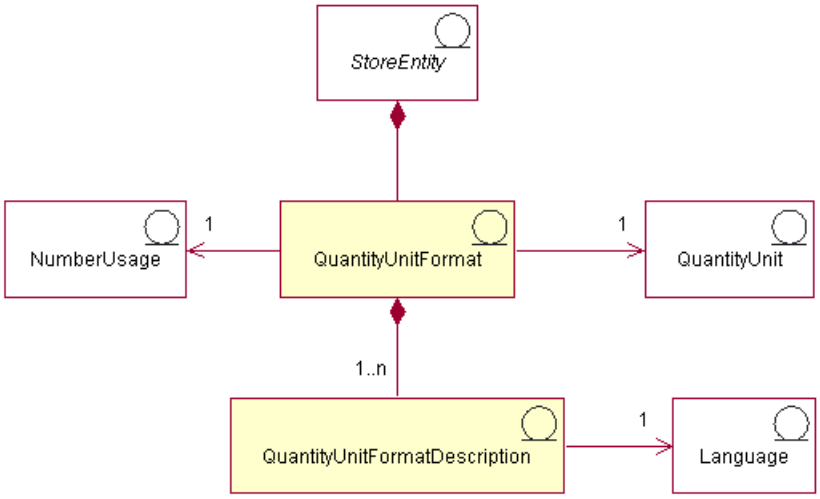
予定可能 (ATP) 在庫が使用可能な場合 (STORE の ALLOCATIONGOODFOR 列を参照)、在庫の数量単位は BASEITEM テーブルの QUANTITYMEASURE 列で定義されます。それ以外の場合は、INVENTORY テーブルの QUANTITYMEASURE 列で定義されます。

丸めの数量をカタログ・エントリーの名目上の数量で割ったものは、正規化された数量として認識されます。正規化された数量は、実行するコマンドに応じて、オーダー・アイテムまたは興味のあるアイテムに保管されます。たとえば、切り上げ済み数量が 6 kg で名目数量が 2 kg の場合、正規化数量は 3 になります。

カタログ・エントリーに対するオファーを検索する際、要求された数量はどのオファーが最適価格を提示するかに作用し、その結果どのオファーを使用するかが決まります。たとえば、切り上げ済み数量が 6 kg で 2 つのオファーがあるとします。一方は、名目数量 2 kg、最小数量 10 kg に対して価格を 4 ドルとするもの、もう一方は、名目数量 2 kg、最小数量 2 kg に対して価格を 4.5 ドルとするものです。この場合、使用できるのは、後者のオファーだけです。

WebSphere Commerce の計測単位について

次の図は、WebSphere Commerce Server における計測単位の構造を示しています。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、 29 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則についての詳細は、 311 ページの『付録 A. UML の凡例』を参照してください。

数量単位と数量単位形式

数量単位 は、ストアで使用される計測単位です。たとえばキログラム、ポンド、メートル、インチ、リットルなどです。数量単位形式は、この数量単位がストアで形式設定される方法、たとえば数量単位の表示時に使用される小数部の桁数などです。

各*数量単位形式* はただ 1 つのストア・エンティティーの一部ですが、各ストア・エンティティーは複数の数量単位形式を持つことがあります。

1 つの数量単位と数値の使用法に対して 1 つの数量単位形式が可能であり、ストアがサポートする言語の数に応じて、1 つ以上の数量単位形式の記述が可能です。

数量単位形式の説明

数量単位形式記述 は、ある数量を (表示目的で) 特定言語の特定の数量単位で形式設定する方法を記述します。

数値の使用法

数値の使用法 では、アプリケーションにおいて数値を使用する方法を定義します。たとえば、WebSphere Commerce コードで数値の使用法を使用することによって、数値 (通貨または数量) のフォーマットまたは四捨五入方法を選択できます。これらのコード (NUMBRUSG テーブルで定義される) によって、 CURFORMAT、CURFMTDESC、QTYFORMAT、および QTYFMTDESC にある、そのタイプの数値の使用法に対して指定された規則に従って、数値をフォーマットすることができます。これによりストアは、様々な状態の要件を満たすように、様々な方法で数値をフォーマットすることができます。

WebSphere Commerce Server の計測単位資産の構造の詳細については、WebSphere Commerce オンライン・ヘルプの『数量単位データ・モデル』を参照してください。

WebSphere Commerce での計測単位の作成

計測単位は、インスタンスが作成されるときに、 WebSphere Commerce Server データベースに事前に読み込まれます。詳細については、41 ページの『第 5 章 サイト資産』を参照してください。

また、ストアで使用する新しい計測単位を WebSphere Commerce に定義したり、ストアで使用しないことにした計測単位を削除したりすることもできます。

ストアで使用する新しい計測単位を定義するには、次のテーブルに情報を追加します。

- QTYUNIT
- QTUNITDSC
- QTYFORMAT
- QTYFMTDESC
- QTYUNITMAP
- QTYCONVERT

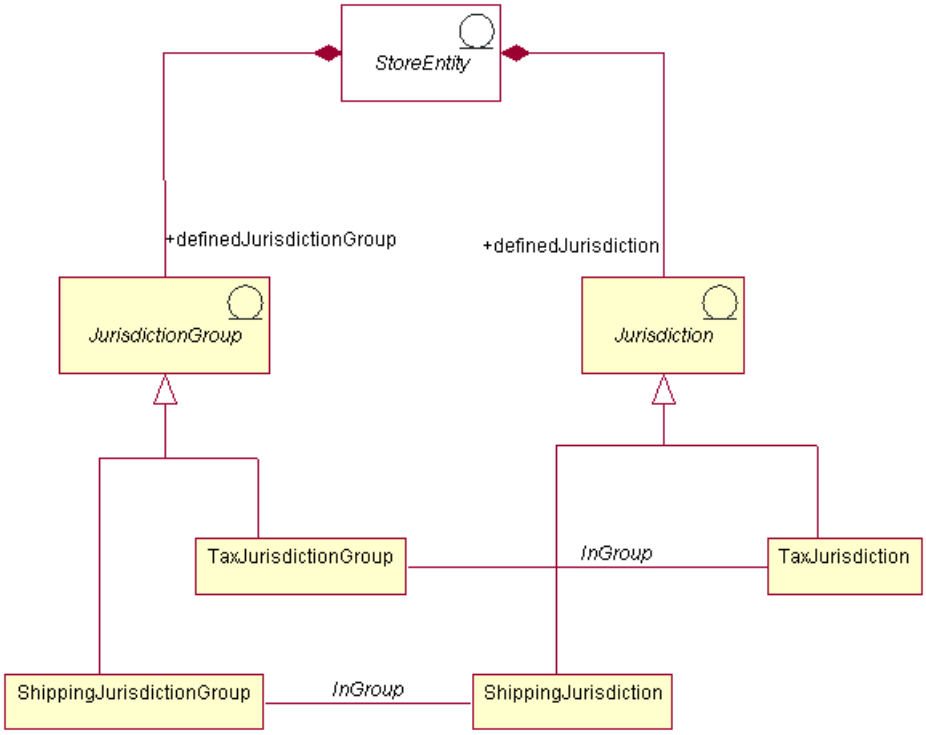
第 17 章 管轄区域資産

管轄区域 とは、商品を販売する国、都道府県、または郵便番号の範囲を表す地理的領域のことです。複数の管轄区域をグループにまとめて、*管轄区域グループ* にすることもできます。

管轄区域グループは、オーダーについての配送料と課税額の計算で使用されます。つまり、管轄区域グループは、配送料と税額計算のために使用するルールを限定するために使用できます。それら限定された計算ルールは、その計算ルールに対応する管轄区域グループ内のいずれかの管轄区域に含まれる住所にアイテムを出荷する場合にのみ、アイテムに適用されます。それで、配送料と税額の計算方法は、オーダーに含まれるさまざまなアイテムの配送先住所に応じて異なる場合があります。

WebSphere Commerce の管轄区域資産について

以下の図は、WebSphere Commerce Server に管轄区域および管轄区域グループが組み込まれている方法を示しています。



この図と、ストア・データの説明の中の他のすべての図は、 WebSphere Commerce Server の情報モデルの一部です。情報モデルの詳細については、 29 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則の詳細については、 311 ページの『付録 A. UML の凡例』を参照してください。

WebSphere Commerce の場合、管轄区域または管轄区域グループはストアの一部であり、作成対象となるストアまたはストア・グループ専用になります。たとえば、ストアに 3 つの管轄区域を作成してからそのストアを削除すると、管轄区域も削除されます。これらの管轄区域は、他の既存のストアや、今後作成するかもしれないストアで使用することはできません。

しかし、あるストア・グループに関して管轄区域を作成した場合、そのグループ内のストアが削除されても、それらの管轄区域は削除されません。そのストア・グループ内で新規に作成されるストアでは、それらの管轄区域が使用可能です。

WebSphere Commerce は、配送管轄区域と課税管轄区域という、2 つのタイプの管轄区域をサポートしています。複数の配送管轄区域をグループにまとめることにより、配送料の計算ルールを限定するための配送管轄区域グループにすることもできます。同じように、複数の課税管轄区域をグループにまとめることにより、税額の計算ルールを限定するための課税管轄区域グループにすることができます。

WebSphere Commerce Server の管轄区域資産の構造の詳細については、WebSphere Commerce オンライン・ヘルプの『管轄区域データ・モデル』を参照してください。

WebSphere Commerce での管轄区域資産の作成

課税額と配送料を適用するためには、ストアに管轄区域資産を作成する必要があります。管轄区域の作成の詳細については、160 ページの『WebSphere Commerce での税資産の作成』、または 141 ページの『WebSphere Commerce での配送資産の作成』を参照してください。

ストアに管轄区域を作成したら、ストア・サービスの「税」および「配送」ノートブックを使用して、管轄区域を編集したり、新しい管轄区域を作成したりできます。

注: ストア・サービスは、作成するすべての管轄区域について自動的に管轄区域グループを作成します。ストア・サービスは、ストアの管轄区域を作成しますが、ストア・グループに対する管轄区域は作成しません。

注: ストア・サービス・ツールは、ストア・アーカイブの形式で事前に読み込まれたデータに対してしか機能しません。

第 18 章 配送資産

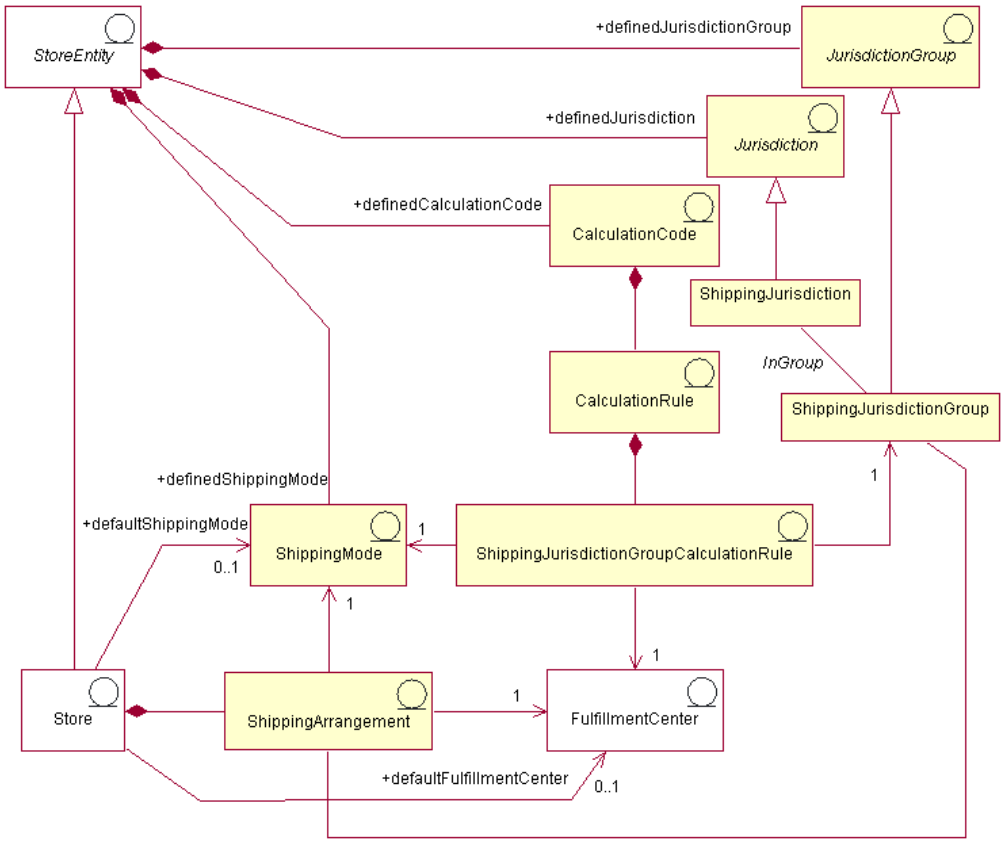
配送とは、ストアが顧客へ商品を物理的に配達する方法のことです。多くの場合、商品は配送センター (ストアの商品の保管に責任のある別個の代理店) から配送されます。

配送サービスの提供と課金を行うには、WebSphere Commerce を使用して作成するストアに、以下のものを組み込まなければなりません。

- 少なくとも 1 つの配送モード
- 少なくとも 1 つの計算コード
- 管轄区域および管轄区域グループ

WebSphere Commerce の配送資産について

以下の図は、WebSphere Commerce Server 内での配送の構造を示しています。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、 29 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則についての詳細は、 311 ページの『付録 A. UML の凡例』を参照してください。

配送モード

配送モード とは、商品を配送する方法のことです。詳しく説明すると、配送モードとは、運送会社 (配送センターから顧客への配送サービスを提供する会社) とその運送会社が提供する配送サービスの組み合わせのことです。たとえば、ABC 運輸の翌日配送サービスや、 ABC 運輸の速達は、配送モードの一例です。

配送モードは、ストア・エンティティーの一部です。ストア・エンティティーを削除すると、そのストア・エンティティー内で定義されている配送モードも削除されます。ストアのデフォルト配送モードは必須ではありませんが、設定することをお勧めします。

配送調整

配送調整 は、ストアと配送センターの間の調整で、配送センターは指定された配送モードを使用して特定のストアに商品を配送するように指定します。配送調整には、特定の制限を課すことができます。それには、配送調整の有効期間や配送管轄区域が含まれます。

配送モードに配送調整が関連付けられている場合、それはその配送モードだけに適用されます。それ以外の場合、配送調整は可能なすべての配送モードに適用されます。配送調整はストアの一部なので、ストアを削除すると共に削除されます。

計算コード

計算コード は配送料の計算に使用されます。つまり、配送料の計算コードはオーダー・アイテムの配送料が計算される方法を示します。オーダー・アイテムの配送料を計算するためには、カタログ・エントリーまたはカタログ・エントリー・グループのいずれかに対して、配送料計算コードを割り当てる必要があります。

計算コードは、ストア・エンティティーの一部です。 1 つの計算コードは 1 つのストア・エンティティーだけと関連付けることができますが、 1 つのストア・エンティティーに複数の計算コードがある場合があります。ストア・エンティティーを削除すると、そのストア・エンティティーに関連した計算コードも削除されます。

計算ルール

各計算コードには*計算ルール* のセットがあります。あるオーダー・アイテムの配送料は、配送モード、配送センター、および配送管轄区域に応じて異なる場合があります。 ShippingJurisdictionGroupCalculationRules は、各オーダー・アイテムで使用する計算ルールを決定するために、配送計算ルールと、管轄区域、配送センター、および配送モードとを関連付ける関係オブジェクトです。

計算ルール、または ShippingJurisdictionGroupCalculationRules によって参照されるその他のオブジェクトのいずれかを削除すると、 ShippingJurisdictionGroupCalculation ルールも削除されます。

管轄区域および管轄区域グループ

管轄区域 とは、商品を販売する国、都道府県、または郵便番号の範囲を表す地理的領域のことです。管轄区域をグループ化すると、*管轄区域グループ* になります。

WebSphere Commerce は、配送管轄区域と課税管轄区域という、2 つのタイプの管轄区域をサポートしています。どちらの管轄区域もそれぞれ対応するグループの一部になります。たとえば、配送管轄区域は配送管轄区域グループの一部になり、課税管轄区域は課税管轄区域グループの一部になります。

管轄区域グループは、計算ルールに関連付けられます。計算ルールは、計算の一部に管轄区域グループを使用して、配送料の金額を決定します。

管轄区域と管轄区域グループは、ストア・エンティティーの一部です。ストア・エンティティーを削除すると、そのストア・エンティティーに関連した管轄区域と管轄区域グループも削除されます。

1 つの配送先住所が、複数の配送管轄区域に関係する場合もあります。たとえば、東京内の配送先住所は、「東京、日本」、「日本」、および「世界」の各配送管轄区域に適用されます。 1 つの配送先住所が複数の配送管轄区域に適用される場合、複数の配送計算ルールが適用できることになります。そのような場合、対応する ShippingJurisdictionGroupCalculationRules の優先順位を使用することによって、使用されるルールが決定されます。

WebSphere Commerce Server の配送資産の構造に関する詳細は、WebSphere Commerce オンライン・ヘルプの『税オブジェクトとデータ・モデル』を参照してください。

WebSphere Commerce での配送資産の作成

WebSphere Commerce のストア・サービス・ツールを使用すると、ストア・アーカイブ内の特定の配送資産 (配送モードや管轄区域など) の作成と編集ができます (すべての配送資産の作成と編集ができるわけではありません)。ストア・サービス・ツールを使用して編集できる資産の詳細については、 WebSphere Commerce オンライン・ヘルプのトピック『ストア・データベース資産の変更』 を参照してください。

注: ストア・サービス・ツールは、事前に読み込み済みの XML ファイルを、ストア・アーカイブ形式で処理します。

また、配送資産を XML ファイルの形式で作成することもできます。ローダー・パッケージを使用してこれをデータベースにロードできます。したがって、配送資産を作成する選択肢には以下の 2 つがあります。

- WebSphere Commerce に付属のサンプル・ストアの 1 つまたは既存のストア・アーカイブの、既存の配送資産を編集する。
- 新しい配送資産を XML ファイルの形式で作成する。

既存のストア・アーカイブの配送資産を編集することに関する情報は、 WebSphere Commerce のオンライン・ヘルプを参照してください。新しい配送資産を XML ファイルの形式で作成することに関する情報は、 『XML ファイルを使用した配送資産の作成』を参照してください。

XML ファイルを使用した配送資産の作成

ローダー・パッケージを使用してデータベースにロードできる XML ファイルの形式で配送資産を作成します。ローダー・パッケージについて詳しくは、 225 ページの『第 7 部 ストアの発行』を参照してください。多文化ストアを作成する場合

は、ストアでサポートされているロケールごとに別々の XML ファイルを作成することもできます。説明情報はすべてロケール固有のファイルに指定されるので、これを翻訳するのは容易です。

サンプル・ストア (以下のタスクの多くの例がサンプル・ストアから取られています) では、翻訳の不要な情報はすべて 1 つの shipping.xml ファイルに指定されており、翻訳の必要な情報は、そのストアがサポートするロケールごとに別の shipping.xml ファイルに指定されています。ロケール固有のファイルにはすべての説明情報が含まれているので、これを翻訳するのは容易です。

XML ファイルを使用してストアの配送資産を作成するには、以下のようにします。

1. サンプル・ストアの配送資産を作成するために使用される XML ファイルを確認します。サンプル・ストアのすべてのファイルは、対応するストア・アーカイブ・ファイルの中にあります。個々のサンプル・ストアには複数の shipping.xml ファイルが組み込まれており、これらのファイルには配送情報が含まれています。ストア・アーカイブ・ファイルは以下のディレクトリーにあります。
   - NT drive:¥WebSphere¥CommerceServer¥samplestores
   - 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥samplestores
   - AIX /usr/WebSphere/CommerceServer/samplestores
   - Solaris /opt/WebSphere/CommerceServer/samplestores
   - Linux /opt/WebSphere/CommerceServer/samplestores
   - 400 /qibm/proddata/WebCommerce/samplestores

   注: WebSphere Commerce オンライン・ヘルプには、サンプル・ストアに含まれる各データ資産についての情報が記載されています。

   ストア・アーカイブの shipping.xml ファイルを表示するには、 ZIP プログラムを使用してこれを解凍します。 shipping.xml ファイルは data ディレクトリーにあります。言語ごとの shipping.xml は、 data ディレクトリーのロケールごとのサブディレクトリーにあります。
2. 313 ページの『付録 B. データの作成』の情報を確認します。
3. サンプル・ストア・アーカイブの shipping.xml ファイルの 1 つをコピーするか、新しいファイルを作成することにより、shipping.xml ファイルを作成します。詳しくは、shipping.xml に対応する DTD ファイルを参照してください。DTD ファイルは以下のディレクトリーにあります。
   - NT drive:¥WebSphere¥CommerceServer¥xml¥sar
   - 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥xml¥sar
   - AIX /usr/WebSphere/CommerceServer/xml/sar
   - Solaris /opt/WebSphere/CommercServer/xml/sar
   - Linux /opt/WebSphere/CommercServer/xml/sar
   - 400 /qibm/proddata/WebCommerce/xml/sar

4. 商品やサービスの配送先の管轄区域と管轄区域グループを定義します。管轄区域はすべて管轄区域グループに属していなければなりません。
   a. 以下の例を参考にして、XML ファイルの JURSTGROUP テーブルに管轄区域グループを定義します。

      ```
      <jurstgroup
      jurstgroup_id="@jurstgroup_id_1"
      description="Jurisdiction Group1 for Shipping"
      subclass="1"
      storeent_id="@storeent_id_1"
      code="World"/>
      ```

      ここで、
      - jurstgroup_id は、生成されるユニーク鍵です。
      - description は、管轄区域グループを管理するユーザー・インターフェースでの表示に適した、管轄区域グループの簡略説明です。
      - subclass は、以下の管轄区域グループのサブクラスです。
        - 1 = ShippingJurisdictionGroup
        - 2 = TaxJurisdictionGroup
      - storeent_id は、この管轄区域グループに関連したストア・エンティティーです。
      - code は、そのストア・エンティティーおよびサブクラスとともに、この管轄区域グループを固有に識別するコードです。

   b. 次の例を参考にして、 XML ファイルの JURST テーブルに管轄区域を定義します。

      ```
      < jurst
      jurst_id="@jurst_id_1"
      storeent_id="@storeent_id_1"
      code="World"
      subclass="1"/>
      ```

      ここで、
      - jurst_id は、生成されるユニーク鍵です。
      - storeent_id は、この管轄区域グループに関連したストア・エンティティーです。
      - code は、そのストア・エンティティーおよびサブクラスとともに、この管轄区域グループを固有に識別するコードです。
      - subclass は、以下の管轄区域のサブクラスです。
        - 1 = ShippingJurisdiction
        - 2 = TaxJurisdiction

   c. 以下の例を参考にして、JURSTGRPREL テーブルに情報を追加し、ステップ b で作成した管轄区域とステップ a で定義した管轄区域グループを関連付けます。

      ```
      <jurstgprel
      ```

```
jurst_id="@jurst_id_1"
jurstgroup_id="@jurstgroup_id_1"
subclass="1"/>
```

ここで、

- jurst_id は管轄区域です。
- jurstgroup_id は管轄区域グループです。
- subclass は、管轄区域のサブクラスと管轄区域グループのサブクラスです。これらは一致していなければなりません。
  - 1 = ShippingJurisdiction[Group]
  - 2 = TaxJurisdiction[Group]

d. ストアでサポートされているすべての管轄区域と管轄区域グループについて、ステップ a ～ c を繰り返します。

5. ストアで使用される配送モードを定義します。

a. 次の例を参考にして、XML ファイルの SHIPMODE テーブルに配送モードを定義します。

```
<shipmode
shipmode_id="@shipmode_id_1"
field1
storeent_id="@storeent_id_1"
code="Ground 1 week"
carrier="XYZ Carrier"/>
```

ここで、

- shipmode_id は、生成されるユニーク鍵です。
- field1 は、カスタマイズの可能なフィールドです。
- storeent_id は、この配送モードに関連したストア・エンティティーです。
- code は、ストア・エンティティーごとに固有の、マーチャントが割り当てたコードです。
- carrier は、運送会社の名前または ID です。

b. 次の例を参考にして、配送モードに関する情報を SHPMODEDSC テーブルに追加します。多文化のストアを作成する場合は、ロケール固有の XML ファイルにこの情報を組み込む必要があります。

```
< shpmodedsc
description="International mail"
field1="USD$5.00 per order plus USD$1.00 for each item"
field2="5 business days"
shipmode_id="@shipmode_id_1"
language_id="&en_US;" />
```

ここで、

- description は、顧客が選択するために表示するのに適した ShippingMode の簡略説明です。

- field1 および field2 は、カスタマイズの可能なフィールドです。
- shipmode_id は、生成されるユニーク鍵です。
- language_id は、使用される言語です。

c. ストアの中のすべての配送モードについて、ステップ a と b を繰り返します。

6. ストアで使用される計算コードを定義します。

a. 以下の例を参考にして、XML ファイルの CALCODE テーブルに計算コードを定義します。

```
< calcode
calcode_id="@calcode_id_1"
code="shipping Code 1- per/order"
calusage_id="-2"
storeent_id="@storeent_id_1"
groupby="0″
published="1"
sequence="+0.00E+000"
calmethod_id="-23"
calmethod_id_app="-24"
calmethod_id_qfy="-22"
flags="0" />
```

ここで、

- calcode_id は、生成されるユニーク鍵です。
- code は、この CalculationCode を固有に識別する文字ストリングであり、特定の CalculationUsage および StoreEntity が与えられています。
- calusage_id は、この CalculationCode がどんな種類の計算に使用されるかを示します。たとえば、CalculationCode は次の通貨金額の 1 つを計算するために使用することができます。
  - 割引額 (-1)
  - 配送料 (-2)
  - 消費税 (-3)
  - 配送税 (-4)
  - クーポン (-5)
- storeent_id は、この計算コードに関連したストア・エンティティーです。
- groupby は、計算時に OrderItems をグループ化する方法を CalculationCodeCombineMethod に指示するビット・フラグです。 0 = グループ化しない。適用可能なすべての OrderItems を 1 つのグループに入れる。詳細は、 WebSphere Commerce オンライン・ヘルプの『*CALCODE テーブル: 詳細*』を参照してください。
- published は、計算コードが発行されるかどうかを指定します。
  - 0 = 発行されない (一時的に使用不可)

– 1 = 発行される
– 2 = 削除のマーク (発行されない)

- sequence は、CalculationCodes が計算され、低位から高位の順で適用されるよう定義します。 2 つの計算コードのシーケンス番号が同じなら、calcode_id の小さい計算コードから順に計算されます。
- calmethod_id は、この CalculationCode の通貨金額を計算する方法を定義する CalculationCodeCalculateMethod です。 calmethod_id=″-23″ は、配送用の CalculationCodeCalculateMethod です。 WebSphere Commerce に付属している配送計算メソッドはこの 1 つだけです。
- calmethod_id_app は、関連する OrderItems の計算済みの金額を保管する CalculationCodeApplyMethod です。 calmethod_id_app=″-24″ は、配送用の CalculationCodeApplyMethod です。 WebSphere Commerce に付属している配送適用メソッドはこの 1 つだけです。
- calmethod_id_qfy は、この CalculationCode と関連した OrderItems を定義する CalculationCodeQualifyMethod です。 calmethod_id_qfy=″-22″ は、配送用の CalculationCodeQualifyMethod です。 WebSphere Commerce に付属している配送限定メソッドはこの 1 つだけです。
- flags は、この CalculationCode の CalculationCodeQualifyMethod が呼び出されるかどうかを指定します。
  – 0 = 制限なし。メソッドは呼び出されません。
  – 1 = 制限あり。メソッドは呼び出されます。

b. 以下の例を参考にして、 XML ファイルの CALCODEDSC テーブルに計算コードの説明情報を追加します。多文化ストアを作成する場合は、ロケール固有の XML ファイルにこの情報を組み込む必要があります。

```
<calcodedsc
calcode_id="@calcode_id_3"
description="5.00USD per order"
language_id="&en_US"
longdescription= "This shipping calculation code charges 5.00USD per order."
/>
```

ここで、

- calcode_id は、この情報が適用される計算コードです。
- description は、計算コードの簡略説明です。
- language_id は、この情報が適用される言語です。
- longdescription は、計算コードの詳細記述です。

c. ストアの中で使用される計算コードごとに、ステップ a と b を繰り返します。

7. ストアの計算ルールを定義します。

a. 次の例を参考にして、XML ファイルの CALRULE テーブルで計算ルールをセットアップします。

```
<calrule
calrule_id="@calrule_id_1"
calcode_id="@calcode_id_1"
startdate="1900-01-01 00:00:00.000000"
```

```
enddate="2100-01-01 00:00:00.000000"
sequence="+1.00000000000000E+000"
combination="2"
calmethod_id="-27"
calmethod_id_qfy="-26"
flags="1"
identifier="1" />
```
ここで、

- calrule_id は、生成される固有の ID です。
- calcode_id は、この計算ルールを含む計算コードです。
- startdate は、この計算ルールが有効になる時刻です。
- enddate は、この計算ルールの停止が有効になる時刻です。
- sequence は、この計算ルールが処理される順序です。同じ計算コードの計算ルールは低位値から高位値の順序で処理されます。
- combination は、デフォルトの CalculationRuleCombineMethod インプリメンテーションによって実行される、特殊な処理を示す以下のビット・フラグを示します。詳細は、WebSphere Commerce オンライン・ヘルプの『CALRULE テーブル』を参照してください。
- calmethod_id は、一連の OrderItems の通貨結果を計算する CalculationRuleCalculateMethod です。
- calmethod_id_qfy は、CalculationRuleCalculateMethod に送信する OrderItems のセットを決定する CalculationRuleQualifyMethod です。
- flags は、この計算ルールを他の計算ルールと結合する方法を決定するために CalculationRuleCombineMethod によって使用されます。詳細は、『CALRULE テーブル: 詳細』を参照してください。
- identifier は、この計算ルールとその計算コードを組み合わせて識別します。

詳細は、WebSphere Commerce オンライン・ヘルプの『CALRULE テーブル』を参照してください。

b. ストアの中で使用される計算ルールごとに、ステップ a を繰り返します。個々の計算コードに複数の計算ルールがある場合もあるので注意してください。たとえば、calcode_id=″@calcode_id_1″ を、複数の calrule_ids と関連付けることができます。

8. ストアの計算スケールを定義します。

計算スケールは、計算に適用される範囲のセットです。たとえば、配送料金の場合、それぞれが料金に対応する重量範囲のセットがある場合があります。つまり、重量が 0〜5 kg の商品の配送料は $10.00、重量が 5〜10 kg の商品の配送料は $15.00、などです。これらの範囲がスケールを構成します。

a. 以下の例を参考にして、XML ファイルの CALSCALE テーブルで計算スケールをセットアップします。

```
<calscale
calscale_id="@calscale_id_1"
code="Scale Code 1 per order USD"
```

```
storeent_id="@storeent_id_1"
calusage_id="-2"
setccurr="USD"
calmethod_id="-28"/>
```

ここで、

- calscale_id は、生成される固有の ID です。
- code は、この計算スケールを固有に識別する文字ストリングであり、特定の計算方法およびストア・エンティティーが与えられています。
- storeent_id は、この計算スケールを含むストア・エンティティーです。
- calusage_id は、この CalculationScale がどんな種類の計算に使用されるかを示します。たとえば、CalculationScale は次の通貨金額の 1 つを計算するために使用することができます。
  - 割引額 (-1)
  - 配送料 (-2)
  - 消費税 (-3)
  - 配送税 (-4)
  - クーポン (-5)
- setccurr が指定されると、これはこの計算スケールの計算範囲オブジェクトの範囲開始値の通貨を示します。 CalculationScaleLookupMethod はこの通貨で「ルックアップ番号」を戻すことになります。
- calmethod_id は、一連のオーダー・アイテムが指定された CalculationScaleLookupMethod です。通貨金額を計算するために計算スケールで使用できるルックアップ値、基本通貨値、結果乗数、および正確な重量のセットを決定します。使用する CalculationScaleLookupMethod を決定するには、以下のようにします。
  - WebSphere Commerce オンライン・ヘルプの『CALMETHOD テーブル』を参照します。 SUBCLASS 列の説明を参照してください。『CALMETHOD テーブル: 詳細』のリンクをクリックします。このテーブルには、使用できる計算メソッドのタイプがリストされています。MonetaryCalculationScaleLookupMethod メソッドは 9 です。
  - ブートストラップ・ファイル wcs.bootstrap_xx_XX.xml を開きます (xx_XX はロケールのコード)。ブートストラップ・ファイルは以下のディレクトリーに置かれています。
    - NT drive:¥WebSphere¥CommerceServer¥schema¥xml
    - 2000 drive:¥Program Files¥WebSphere¥CommerceServer ¥schema¥xml
    - AIX /usr/WebSphere/CommerceServer/schema/xml
    - Solaris /opt/WebSphere/CommerceServer/schema/xml
    - Linux /opt/WebSphere/CommerceServer/schema/xml
    - 400 /qibm/proddata/WebCommerce/schema/xml

– 使用できる計算メソッドをリストしているセクション (CALMETHOD) を見つけます。
– calusage_ID が税 (消費税の場合は -3、配送税の場合は -4) の計算メソッドを見つけます。
– サブクラスが 7 の計算メソッドを見つけます。これはいくつか存在します。必要に応じて 1 つ選択してください。

詳細は、WebSphere Commerce オンライン・ヘルプの『CALSCALE テーブル』を参照してください。

b. ストアで使用される計算スケールごとに、ステップ a を繰り返します。たとえば、配送に関し、InFashion は、オーダー基準単価のスケールとアイテム基準単価のスケールを作成します。

9. 計算スケールの計算範囲を定義します。

a. 次の例を参考にして、XML ファイルの CALRANGE テーブルで計算範囲をセットアップします。

```
<calrange
calrange_id="@calrange_id_1"
calscale_id="@calscale_id_1"
calmethod_id="-33"
rangestart="0.00000"
cumulative="0"/>
```

ここで、

- calrange_id は、生成される固有の ID です。
- calscale_id は、この計算範囲を含む計算スケールです。
- calmethod_id は、 CalculationRangeLookupResult からの通貨金額を決定する CalculationRangeMethod です。たとえば、FixedAmountCalculationRangeCmd、 PerUnitAmountCalculationRangeCmd、または PercentageCalculationRangeCmd。 CalculationRangeMethod を決定するには、以下のようにします。
  – WebSphere Commerce オンライン・ヘルプの『CALMETHOD テーブル』を参照します。 SUBCLASS 列の説明を参照してください。『CALMETHOD テーブル: 詳細』のリンクをクリックします。このテーブルには、使用できる計算メソッドのタイプがリストされています。CalculationRangeMethod は 10 です。
  – ブートストラップ・ファイル wcs.bootstrap_xx_XX.xml を開きます (xx_XX はロケールのコード)。ブートストラップ・ファイルは以下のディレクトリーに置かれています。
    - NT drive:¥WebSphere¥CommerceServer¥schema¥xml
    - 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥schema¥xml
    - AIX /usr/WebSphere/CommerceServer/schema/xml
    - Solaris /opt/WebSphere/CommerceServer/schema/xml

- Linux /opt/WebSphere/CommerceServer/schema/xml
- 400 /qibm/proddata/WebCommerce/schema/xml

– 使用できる計算メソッドをリストしているセクション (CALMETHOD) を見つけます。
– calusage_ID が税 (消費税の場合は -3、配送税の場合は -4) の計算メソッドを見つけます。
– サブクラスが 9 の計算メソッドを見つけます。これはいくつか存在します。必要に応じて 1 つ選択してください。

• cumulative は、以下の有効値です。
– 0 = 最高の RANGESTART 値に一致する CalculationRange だけが使用される。
– 1 = 一致する CalculationRanges はすべて使用される。計算された通貨金額が合計されて、最終結果が得られます。

詳細は、WebSphere Commerce オンライン・ヘルプの『CALRANGE テーブル』を参照してください。

b. ストアの中で使用される計算スケールに関連した計算範囲ごとに、ステップ a を繰り返します。

10. 計算スケールの計算ルックアップ値を定義します。計算ルックアップ値は、計算スケールに関連した値です。たとえば、計算スケールに、以下のような配送の重量範囲と関連価格が組み込まれているとします。

• 0 ～ 5 kg の料金は 10 ドル
• 5 ～ 10 kg の料金は 15 ドル

この場合、ルックアップ値は 10 ドルと 15 ドルです。

a. 以下の例を参考にして、 XML ファイルの CALRLOOKUP テーブルで計算ルックアップ値をセットアップします。 多文化ストアを作成する場合は、ロケール固有の XML ファイル、つまりストアでサポートされているロケール当たり 1 つのファイルにこの情報を組み込む必要があります。たとえば、ストアから米国と日本の顧客に配送する場合は、 1 つの XML ファイルに US ドルのルックアップ値を追加し、もう 1 つの XML ファイルに日本円のルックアップ値を追加する必要があります。

```
<calrlookup
calrlookup_id="@calrlookup_id_1"
setccurr="USD"
calrange_id="@calrange_id_1"
value="5.00"/>
```

ここで、

• calrlookup_id は、生成される固有の ID です。
• calrange_id は、計算範囲ルックアップ結果を含む計算範囲です。
• value は、計算範囲ルックアップ結果の値です。この値は、計算範囲の計算範囲メソッドによって使用され、通貨結果が決定されます。

詳細は、WebSphere Commerce オンライン・ヘルプの『CALRLOOKUP テーブル』を参照してください。

b. ストアで使用される計算スケールに関連したルックアップ値ごとに、ステップ a を繰り返します。

11. 計算ルールと計算スケールを関連付けます。

a. 以下の例を参考にして、XML ファイルの CRULESCALE テーブルで計算スケールと計算ルールを関連付けます。

```
< crulescale
calrule_id="@calrule_id_1"
calscale_id="@calscale_id_1" />
```

ここで、

- `calrule_id` は計算ルールです。
- `calscale_id` は計算スケールです。

b. 関連付ける計算スケールとルールごとに、ステップ a を繰り返します。

@ と **&** の使用法の詳細については、 313 ページの『付録 B. データの作成』を参照してください。

配送フルフィルメント資産の作成

ストアで配送資産が正しく機能するためには、ストアで使用される配送管轄区域グループと計算ルール、および配送センターと配送モードを関連付けなければなりません。

配送資産を配送センターに関連付けるには、その前にフルフィルメント資産を作成しなければなりません。フルフィルメント資産の作成の詳細については、 116 ページの『WebSphere Commerce での配送資産の作成』を参照してください。

フルフィルメント資産を作成し終えたら、 SHPJCRULE テーブルと SHPARRANGE テーブルに情報を追加して、配送資産を関連付けます。 以下のようにします。

1. サンプル・ストアの配送フルフィルメント資産を作成するために使用される XML ファイルを確認します。サンプル・ストアのすべてのファイルは、対応するストア・アーカイブ・ファイルの中にあります。個々のサンプル・ストアには 1 つの `shipfulfill.xml` ファイルが組み込まれており、このファイルには配送フルフィルメント情報が組み込まれています。ストア・アーカイブの `shipfulfill.xml` ファイルを表示するには、 ZIP プログラムを使用してこれを解凍します。 `shipfulfill.xml` ファイルは、data ディレクトリーにあります。

   ストア・アーカイブ・ファイルは以下のディレクトリーにあります。

   - NT `drive:¥WebSphere¥CommerceServer¥samplestores`
   - 2000 `drive:¥Program Files¥WebSphere¥CommerceServer¥samplestores`
   - AIX `/usr/WebSphere/CommerceServer/samplestores`
   - Solaris `/opt/WebSphere/CommerceServer/samplestores`
   - Linux `/opt/WebSphere/CommerceServer/samplestores`

- 400 /qibm/proddata/WebCommerce/samplestores

  注: WebSphere Commerce オンライン・ヘルプには、サンプル・ストアに含まれる各データ資産についての情報が記載されています。

2. 313 ページの『付録 B. データの作成』の情報を確認します。
3. サンプル・ストア・アーカイブの shipfulfill.xml ファイルの 1 つをコピーするか、または新しいファイルを作成することにより、shipfulfill.xml ファイルを作成します。詳しくは、shipfulfill.xml に対応する DTD ファイルを参照してください。 DTD ファイルは以下のディレクトリーにあります。
   - NT drive:¥WebSphere¥CommerceServer¥xml¥sar
   - 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥xml¥sar
   - AIX /usr/WebSphere/CommerceServer/xml/sar
   - Solaris /opt/WebSphere/CommerceServer/xml/sar
   - Linux /opt/WebSphere/CommerceServer/xml/sar
   - 400 /qibm/proddata/WebCommerce/xml/sar
4. SHPJCRULE テーブルに情報を追加して、計算ルールと配送管轄区域グループを関連付けます。次の例を参考にしてください。多文化ストアを作成する場合は、ストアでサポートされているロケールごとに 1 つずつ XML ファイルを作成する必要もあります。

   ```
   <shpjcrule
   calrule_id="@calrule_id_1"
   ffmcenter_id="@ffmcenter_id_1"
   jurstgroup_id="@jurstgroup_id_1"
   precedence="0"
   shipmode_id="@shipmode_id_1"
   shpjcrule_id="@shpjcrule_id_1"
   ```

   ここで、
   - calrule_id は、使用される計算ルールです。
   - ffmcenter_id は、配送センターです。これが NULL の場合、この関連はすべての配送センターに適用されます。
   - jurstgroup_id は配送管轄区域グループです。これが NULL の場合、この関連はすべての配送管轄区域グループに適用されます。
   - precedence は、配送先住所が、同一の配送センターと配送モードのために指定された複数の配送管轄区域グループに属する場合に、SHPJCRULE.PRECEDENCE 値が最大の計算ルールにのみ限定されることを示します。
   - shipmode_id は配送モードです。
   - shpjcrule_id は、生成される固有の ID です。
5. ストアの管轄区域グループ、配送センター、およびルールの関連ごとに、ステップ 3 を繰り返します。
6. SHPARRANGE テーブルに情報を追加して、配送モードと配送センターをストアに関連付けます。次の例を参考にしてください。

```
<shparrange
 shparrange_id="@shparrange_id_2"
 store_id="@storeent_id_1"
 ffmcenter_id="@ffmcenter_id_1"
 shipmode_id= "@shipmode_id_2"
 startdate="1970-06-22 23:00:00.000000"
 enddate= "2008-06-22 23:00:00.000000"
 precedence= "0"
 flags="0"
/>
```

ここで、

- shparrange_id は、生成される固有の ID です。
- store_id は、ストアです。
- ffmcenter_id は、配送センターです。
- shipmode_id は配送モードです。 NULL は、配送モードに関係なくこの配送調整を使用できることを示します。
- startdate は、この配送調整の開始が有効になる時刻です。
- enddate は、この配送調整の停止が有効になる時刻です。
- precedence は、特定の時点に複数の配送調整が (同じストアおよび配送モードに対して) 有効であった場合には、 PRECEDENCE が一番高い配送調整が使用されることを示します。
- flags は、以下のビット・フラグを含みます。
  - 1 = 制限あり - この配送調整は、この配送調整と (テーブル SHPARJURGP を介して) 関連付けられた配送管轄区域グループのいずれかと一致する住所を持つオーダー・アイテムにのみ適用されます。

7. ストアで使用されるすべての配送モードについて、ステップ 5 を繰り返します。

@ と **&** の使用法の詳細については、 313 ページの『付録 B. データの作成』を参照してください。

ストア - カタログ - 配送資産の作成

配送モードとストアを関連付けるには、ストアに組み込まれている契約ごとに、ストアの計算コードとカタログ・エントリーを関連付けなければなりません。

ストア - カタログ - 配送資産を作成するには、その前にストア資産とカタログ資産を作成しなければなりません。ストア資産の作成の詳細については、 47 ページの『XML ファイルによるストア・データ資産の作成』を参照してください。カタログ資産の作成の詳細については、 82 ページの『ストア・カタログ資産の表示』を参照してください。

ストア - カタログ - 配送資産を作成するには、以下のようにします。

1. サンプル・ストアの配送フルフィルメント資産を作成するために使用される XML ファイルを確認します。サンプル・ストアのすべてのファイルは、対応するストア・アーカイブ・ファイルの中にあります。

   ストア・アーカイブ・ファイルは以下のディレクトリーにあります。

- NT drive:¥WebSphere¥CommerceServer¥samplestores
- 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥samplestores
- AIX /usr/WebSphere/CommerceServer/samplestores
- Solaris /opt/WebSphere/CommerceServer/samplestores
- Linux /opt/WebSphere/CommerceServer/samplestores
- 400 /qibm/proddata/WebCommerce/samplestores

注: WebSphere Commerce オンライン・ヘルプには、サンプル・ストアに含まれる各データ資産についての情報が記載されています。

個々のサンプル・ストアには 1 つの store-catalog-shipping.xml ファイルが組み込まれており、このファイルには配送フルフィルメント情報が組み込まれています。 ストア・アーカイブの store-catalog-shipping.xml ファイルを表示するには、 ZIP プログラムを使用してこれを解凍します。 store-catalog-shipping.xml ファイルは、data ディレクトリーにあります。

2. 313 ページの『付録 B. データの作成』の情報を確認します。
3. サンプル・ストア・アーカイブの store-catalog-shipping.xml ファイルの 1 つをコピーするか、新しいファイルを作成することにより、store-catalog-shipping.xml ファイルを作成します。詳しくは、store-catalog-shipping.xml に対応する DTD ファイルを参照してください。DTD ファイルは以下のディレクトリーにあります。
   - NT drive:¥WebSphere¥CommerceServer¥xml¥sar
   - 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥xml¥sar
   - AIX /usr/WebSphere/CommerceServer/xml/sar
   - Solaris /opt/WebSphere/CommerceServer/xml/sar
   - Linux /opt/WebSphere/CommerceServer/xml/sar
   - 400 /qibm/proddata/WebCommerce/xml/sar
4. CATENCALCD テーブルに情報を追加して、ストア - カタログ - 配送の関係を作成します。次の例を参考にしてください。

```
<catencalcd
 calcode_id="@calcode_id_1"
 catencalcd_id="@catencalcd_id_1"
   store_id="@storeent_id_1"
/>
```

ここで、

- calcode_id は計算コードです。
- catencalcd_id は、生成される固有の ID です。
- store_id は、ストアです。

@ と **&** の使用法の詳細については、 313 ページの『付録 B. データの作成』を参照してください。

デフォルト配送モードの作成

ストアのデフォルト配送モードを設定するには、情報を STOREDEF テーブルに追加しなければなりません。情報を STOREDEF テーブルに追加するには、以下のようにします。

1. サンプル・ストアのデフォルト資産を作成するために使用される XML ファイルを確認します。サンプル・ストアのすべてのファイルは、対応するストア・アーカイブ・ファイルの中にあります。

   ストア・アーカイブ・ファイルは以下のディレクトリーにあります。

   - NT `drive:¥WebSphere¥CommerceServer¥samplestores`
   - 2000 `drive:¥Program Files¥WebSphere¥CommerceServer¥samplestores`
   - AIX `/usr/WebSphere/CommerceServer/samplestores`
   - Solaris `/opt/WebSphere/CommerceServer/samplestores`
   - Linux `/opt/WebSphere/CommerceServer/samplestores`
   - 400 `/qibm/proddata/WebCommerce/samplestores`

     注: WebSphere Commerce オンライン・ヘルプには、サンプル・ストアに含まれる各データ資産についての情報が記載されています。

   個々のサンプル・ストアには 1 つの `store-defaults.xml` ファイルが含まれており、このファイルにはデフォルトの配送情報が組み込まれています。ストア・アーカイブの `store-defaults.xml` ファイルを表示するには、 ZIP プログラムを使用してこれを解凍します。 `store-defaults.xml` ファイルは、data ディレクトリーにあります。

2. 313 ページの『付録 B. データの作成』の情報を確認します。
3. サンプル・ストア・アーカイブの `store-defaults.xml` ファイルの 1 つをコピーするか、新しいファイルを作成することにより、`store-defaults.xml` ファイルを作成します。詳しくは、`store-defaults.xml` に対応する DTD ファイルを参照してください。 DTD ファイルは以下のディレクトリーにあります。

   - NT `drive:¥WebSphere¥CommerceServer¥xml¥sar`
   - 2000 `drive:¥Program Files¥WebSphere¥CommerceServer¥xml¥sar`
   - AIX `/usr/WebSphere/CommerceServer/xml/sar`
   - Solaris `/opt/WebSphere/CommerceServer/xml/sar`
   - Linux `/opt/WebSphere/CommerceServer/xml/sar`
   - 400 `/qibm/proddata/WebCommerce/xml/sar`
4. 次の例を参考にして、XML ファイルの STOREDEF テーブルに情報を追加して、ストアのデフォルト配送モードを指定します。

```
<storedef
   store_id="@storeent_id_1"
   shipmode_id="@shipmode_id_1"
/>
```

   ここで、

- `store_id` は、ストアです。
- `shipmode_id` は、ストアのデフォルト配送モードです。

@ と **&** の使用法の詳細については、 313 ページの『付録 B. データの作成』を参照してください。

第 19 章 税資産

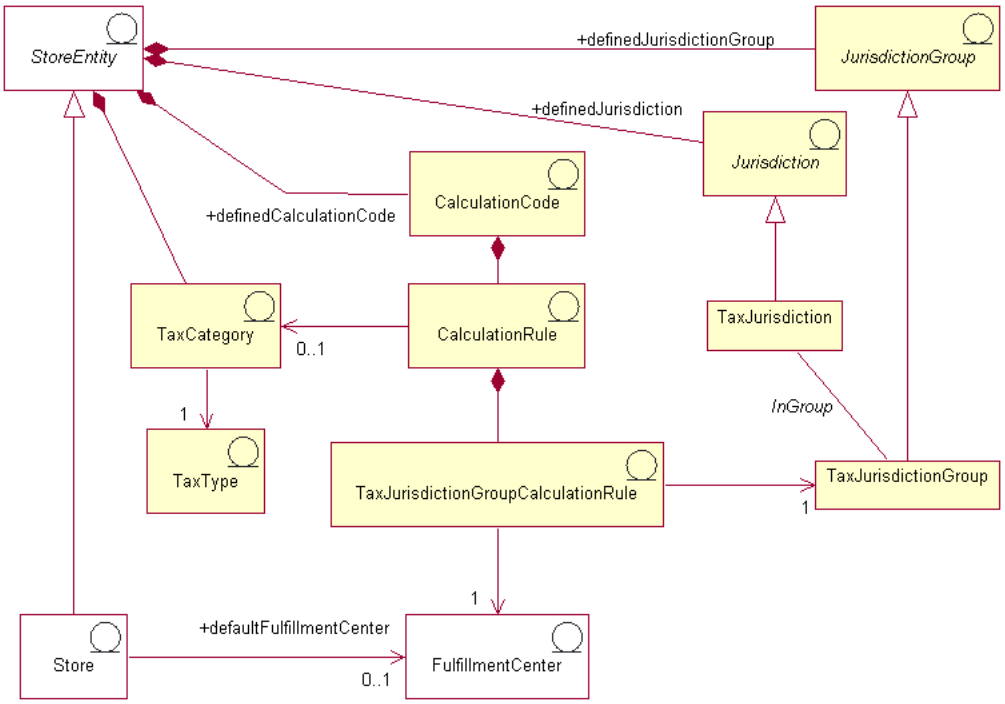
ストアに備えられている商品やサービスの税を課金したり徴収したりするには、WebSphere Commerce を使用して作成したストアに以下のものを組み込まなければなりません。

- 課税カテゴリー
- 計算コード
- 管轄区域および管轄区域グループ

課税カテゴリー、計算コード、および管轄区域と管轄区域グループの組み合わせにより、ストアの課税額が作成されます。

WebSphere Commerce の税資産について

以下の図は、WebSphere Commerce Server の課税構造を示しています。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、29 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則については、311 ページの『付録 A. UML の凡例』を参照してください。

課税カテゴリー

課税カテゴリー は、ストアが徴収する必要がある各種の税 (国税、都道府県税、市町村税など) に対応します。

1 つの課税カテゴリーは、1 つのストア・エンティティーの一部になります。一方、1 つのストア・エンティティーに複数の課税カテゴリーがある場合があります。ストア・エンティティーを削除すると、そのストア・エンティティーに関連した課税カテゴリーも削除されます。

税タイプ

ストアは普通、2 タイプの税を徴収します。それは、売上税または消費税と、配送税です。課税カテゴリーごとに 1 つの*税タイプ* があります。個々の課税カテゴリーの税タイプは 1 つだけですが (たとえば、税カテゴリーが国税の場合は税タイプは消費税)、数種類の課税カテゴリーが 1 つの税タイプに属する場合があります (たとえば、消費税の税タイプは、国税、都道府県税、市町村税のカテゴリーにあてはまります)。

計算コード

計算コード は課税額の計算に使用されます。つまり、税額計算コードはオーダー・アイテムの税が計算される方法を示します。オーダー・アイテムの税額を計算するためには、カタログ・エントリーまたはカタログ・エントリー・グループのいずれかに対して、消費税および配送税計算コードを割り当てる必要があります。特定のカタログ・エントリーまたはカタログ・エントリーのグループに対して、各税タイプの税額計算コードを 1 つだけ適用できます。通常は、売上税または消費税は正価に課され、配送税は配送料に課されます。

計算コードは、ストア・エンティティーの一部です。 1 つの計算コードは 1 つのストア・エンティティーだけと関連付けることができますが、 1 つのストア・エンティティーに複数の計算コードがある場合があります。ストア・エンティティーを削除すると、そのストア・エンティティーに関連した計算コードも削除されます。

計算ルール

各計算コードには、少なくとも 1 つの*計算ルール* があります。これは、各課税カテゴリーの計算方法を定義し、計算実行の条件を指定するものです。個々の税額計算ルールは、1 つの課税カテゴリー、1 つの管轄区域グループ、および 1 つの配送センターに関連付けられます。それらの組み合わせにより、計算ルールが使用される条件が定義されます。たとえば、特定の課税カテゴリーにおける金額の計算においては、オーダーで指定される配送先住所や配送センターに応じて異なるルールが選択されることがあります。

各計算ルールは、ちょうど 1 つの計算コードに属します。

1 つの税額計算コードが複数の計算ルールを持つ場合があります。ストアに関連付けられている課税カテゴリー、課税管轄区域グループ、および配送センターの組み合わせごとに 1 つの計算ルールです。消費税および配送税の各計算ルールは、複数の TaxJurisdictionGroupCalculationRules (TaxRule) に関連付けることができます。たとえば下の表の中で、計算ルール 10001 は、管轄区域グループ 1234 と 1235 の両方に適用されます。

TAXJCRULE_ID	CALRULE_ID	FFMCENTER_ID	JURSTGROUP_ID	PRECEDENCE
10001	10001	NULL	1234	0
10002	10001	NULL	1235	0

各 TaxRule により、それぞれの計算ルールが適用される条件が定義されます。たとえば、ストアの出荷先となる管轄区域グループごとに計算ルールを定義できます。次に示す例の場合、計算ルール 10001 は管轄区域グループ 1234 と 1235 の両方に適用されます。

以下の例の税額計算コードでは、課税管轄区域がアルバータの場合は、都道府県税の消費税カテゴリーの計算ルール A が使用され、課税管轄区域がブリティッシュコロンビアの場合はルール C が使用されます。

課税管轄区域	国税の消費税	都道府県税の消費税
カナダのアルバータ	計算ルール B (Y%)	計算ルール A (X%)
カナダのブリティッシュコロンビア	計算ルール B (Y%)	計算ルール C (Z%)

配送先住所が複数の課税管轄区域グループと一致する場合、関連付けられた TAXJCRULE.PRECEDENCE 列の値が最も大きい計算ルールが使用されます。

TaxJurisdictionGroupCalculationRules (TaxRule) と計算ルールとの関連付けにより、どんな場合に計算ルールが適用されるかが決定されます。消費税または配送税の計算ルールは、 TaxRule で指定される条件のいずれか 1 つが満たされている場合に適用されます。以下の例の場合、計算ルール 10001 は、管轄区域グループ 1001 に配送する場合、または配送センター 1001 から配送する場合、または管轄区域グループ 1001 に配送する場合に適用されます。

CALRULE_ID	FFMCENTER_ID	JURSTGROUP_ID
10001	NULL	1001
10001	1001	1001

各 TaxJurisdictionGroupCalculationRule は、多くても 1 つの管轄区域グループに関連付けられます。計算ルールは、それ自体が直接管轄区域グループに関連付けられるわけではありません。

管轄区域および管轄区域グループ

管轄区域 とは、商品を販売する国、都道府県、または郵便番号の範囲を表す地理的領域のことです。管轄区域をグループ化すると、*管轄区域グループ* になります。

WebSphere Commerce は、配送管轄区域と課税管轄区域という、2 つのタイプの管轄区域をサポートしています。どちらの管轄区域もそれぞれ対応するグループの一部になります。たとえば、配送管轄区域は配送管轄区域グループの一部になり、課税管轄区域は課税管轄区域グループの一部になります。

管轄区域と管轄区域グループにより、課税額の計算に使用される計算ルールが決められます。

管轄区域と管轄区域グループは、ストア・エンティティーの一部です。個々の管轄区域や管轄区域グループは、1 つのストア・エンティティーの一部になります。しかしながら、1 つのストア・エンティティーに複数の管轄区域や管轄区域グループがある場合があります。ストア・エンティティーを削除すると、そのストア・エンティティーに関連した管轄区域と管轄区域グループも削除されます。

WebSphere Commerce Server の税資産の構造に関する詳細は、 WebSphere Commerce オンライン・ヘルプで『オブジェクト・モデルとデータ・モデル』を参照してください。

WebSphere Commerce での税資産の作成

WebSphere Commerce のストア・サービス・ツールを使用すると、すべての税資産ではなく特定の税資産 (課税カテゴリーや管轄区域など) を、ストア・アーカイブに作成して編集できます。ストア・サービス・ツールを使用して編集できる資産の詳細については、 WebSphere Commerce オンライン・ヘルプのトピック『*ストア・データベース資産の変更*』 を参照してください。

注: ストア・サービス・ツールは、事前に読み込み済みの XML ファイルを、ストア・アーカイブ形式で処理します。

また、税資産を XML ファイルの形式で作成することもできます。ローダー・パッケージを使用してこのファイルをデータベースにロードできます。したがって、配送資産を作成する選択肢には以下の 2 つがあります。

- WebSphere Commerce に備えられているサンプル・ストアの 1 つ、または既存のストア・アーカイブの、既存の税資産を編集する。
- 新しい税資産を XML ファイルの形式で作成する。このファイルは、ストア・アーカイブの一部として発行するか、またはローダー・パッケージを使用してロードできます。

既存のストア・アーカイブの税資産を編集することに関する情報は、 WebSphere Commerce のオンライン・ヘルプを参照してください。新しい税資産を XML ファイルの形式で作成することに関する情報は、 『XML ファイルを使用した税資産の作成』を参照してください。

XML ファイルを使用した税資産の作成

税資産を XML ファイルの形式で作成します。XML ファイルは、ローダー・パッケージを使用して、データベースにロード可能です。ローダー・パッケージについて詳しくは、 225 ページの『第 7 部 ストアの発行』を参照してください。多文化ストアを作成する場合は、ストアでサポートされているロケールごとに別々の XML ファイルを作成することもできます。説明情報はすべてロケール固有のファイルに指定されるので、これを翻訳するのは容易です。

サンプル・ストア (以下のタスクの多くの例がサンプル・ストアから取られています) では、翻訳の不要な情報はすべて 1 つの `tax.xml` ファイルに指定されており、翻訳の必要な情報は、そのストアがサポートするロケールごとに別の `tax.xml` ファイルに指定されています。ロケール固有のファイルには、すべての説明情報が含まれています。

XML ファイルを使用してストアの税資産を作成するには、以下のようにします。

1. サンプル・ストアの税資産を作成するために使用される XML ファイルを確認します。サンプル・ストアのすべてのファイルは、対応するストア・アーカイブ・ファイルの中にあります。

   ストア・アーカイブ・ファイルは以下のディレクトリーにあります。

   - NT `drive:¥WebSphere¥CommerceServer¥samplestores`
   - 2000 `drive:¥Program Files¥WebSphere¥CommerceServer¥samplestores`
   - AIX `/usr/WebSphere/CommerceServer/samplestores`
   - Solaris `/opt/WebSphere/CommerceServer/samplestores`
   - Linux `/opt/WebSphere/CommerceServer/samplestores`
   - 400 `/qibm/proddata/WebCommerce/samplestores`

     注: WebSphere Commerce オンライン・ヘルプには、サンプル・ストアに含まれる各データ資産についての情報が記載されています。

   各サンプル・ストアには、tax.xml ファイルが 2 つ組み込まれています。これには税に関する情報が含まれています。ストア・アーカイブの tax.xml ファイルを表示するには、 ZIP プログラムを使用してこれを解凍します。 tax.xml ファイルは、data ディレクトリーにあります。言語ごとの tax.xml は、 data ディレクトリーのロケールごとのサブディレクトリーにあります。
2. 313 ページの『付録 B. データの作成』の情報を確認します。
3. サンプル・ストア・アーカイブの tax.xml ファイルの 1 つをコピーするか、または新しいファイルを作成することにより、tax.xml ファイルを作成します。詳しくは、tax.xml に対応する DTD ファイルを参照してください。 DTD ファイルは以下のディレクトリーにあります。

   - NT `drive:¥WebSphere¥CommerceServer¥xml¥sar`
   - 2000 `drive:¥Program Files¥WebSphere¥CommerceServer¥xml¥sar`
   - AIX `/usr/WebSphere/CommerceServer/xml/sar`
   - Solaris `/opt/WebSphere/CommerceServer/xml/sar`
   - Linux `/opt/WebSphere/CommerceServer/xml/sar`
   - 400 `/qibm/proddata/WebCommerce/xml/sar`
4. 商品やサービスの配送先の管轄区域と管轄区域グループを定義します。課税管轄区域を、それが適用される課税カテゴリーの計算ルールに従って、課税管轄区域グループに割り当ててください。

   a. 以下の例を参考にして、XML ファイルの JURSTGROUP テーブルに管轄区域グループを定義します。

   ```
   <jurstgroup
   jurstgroup_id="@jurstgroup_id_2"
   description="Tax Jurstiction Group 1"
   subclass="2"
   storeent_id="@storeent_id_1"
   ```

```
code="World"/>
```

ここで、

- jurstgroup_id は、生成されるユニーク鍵です。
- description は、管轄区域グループを管理するユーザー・インターフェースでの表示に適した、管轄区域グループの簡略説明です。
- subclass は、以下の管轄区域グループのサブクラスです。
  - 1 = ShippingJurisdictionGroup
  - 2 = TaxJurisdictionGroup
- storeent_id は、この管轄区域グループに関連したストア・エンティティーです。
- code は、そのストア・エンティティーおよびサブクラスとともに、この管轄区域グループを固有に識別するコードです。

b. 以下の例を参考にして、XML ファイルの JURST テーブルに管轄区域を定義します。

```
<jurst
jurst_id="@jurst_id_2"
storeent_id="@storeent_id_1"
code="World"
subclass="2"/>
```

ここで、

- jurst_id は、生成されるユニーク鍵です。
- storeent_id は、この管轄区域グループに関連したストア・エンティティーです。
- code は、そのストア・エンティティーおよびサブクラスとともに、この管轄区域グループを固有に識別するコードです。
- subclass は、以下の管轄区域のサブクラスです。
  - 1 = ShippingJurisdiction
  - 2 = TaxJurisdiction

c. 以下の例を参考にして、JURSTGRPREL テーブルに情報を追加し、ステップ b で作成した管轄区域とステップ a で定義した管轄区域グループを関連付けます。

```
<jurstgprel
jurst_id="@jurst_id_2"
jurstgroup_id="@jurstgroup_id_1"
subclass="2"/>
```

ここで、

- jurst_id は管轄区域です。
- jurstgroup_id は管轄区域グループです。
- subclass は、管轄区域のサブクラスと管轄区域グループのサブクラスです。これらは一致していなければなりません。
  - 1 = ShippingJurisdiction[Group]

– 2 = TaxJurisdiction[Group]

d. ストアでサポートされているすべての管轄区域と管轄区域グループについて、ステップ a ～ c を繰り返します。

5. ストアで使用される課税カテゴリーを定義します。

a. 以下の例を参考にして、XML ファイルの TAXCGRY テーブルに課税カテゴリーを定義します。

```
<taxcgry
taxcgry_id="@taxcgry_id_1"
taxtype_id="-3"
storeent_id="@storeent_id_1"
name="Sales Tax"
displayseq="0"
 displayusage="0"/>
```

ここで、

- taxcgry_id は、生成されるユニーク鍵です。
- taxtype_id="-3" は、この課税カテゴリーの税タイプです。 WebSphere Commerce では、以下の 2 つの税タイプをサポートしています。
  - 消費税または売上税 (-3)
  - 配送税 (-4)
- storeent_id は、この課税カテゴリーに関連したストア・エンティティーです。
- name は、課税カテゴリーの名前です。名前とストア・エンティティーとで、この課税カテゴリーを固有に識別します。
- displayseq は、オーダーで表示される場合の税額の順序を、低額のものから順に指定します。
- displayusage は、この課税カテゴリーを PriceDataBean との関連で以下のように指定します。
  - 0 = 計算されない
  - 1 = 計算される

  PriceDataBean を使用して、商品価格とともに表示する必要がある税金額を得ることができます。

b. ストアの中で使用される課税カテゴリーごとに、ステップ a を繰り返します。

c. 以下の例を参考にして、 XML ファイルの TAXCGRYDS テーブルに課税カテゴリーの説明情報を追加します。多文化ストアを作成する場合は、ロケール固有の XML ファイルにこの情報を組み込む必要があります。

```
<taxcgryds
 taxcgry_id="@taxcgry_id_1"
 description="Sales Tax"
 language_id="&en_US"/>
```

ここで、

- taxcgry_id は、課税カテゴリーです。
- description は、顧客に対して表示するのに適した、課税カテゴリーの簡略説明です。
- language_id は、この情報の表示に使用される言語です。

d. ストアの中で使用される課税カテゴリーごとに、ステップ c を繰り返します。

6. ストアで使用される計算コードを定義します。

a. 以下の例を参考にして、XML ファイルの CALCODE テーブルに計算コードを定義します。

```
<calcode
calcode_id="@calcode_id_3"
 code="Tax Code 1"
calusage_id="-3"
storeent_id="@storeent_id_1"
groupby="0"
published="1"
sequence="0"
calmethod_id="-43"
calmethod_id_app="-44"
calmethod_id_qfy="-42"
displaylevel="0"
flags="0"
precedence="0"
/>
```

ここで、

- calcode_id は、生成されるユニーク鍵です。
- code は、この計算コードを固有に識別する文字ストリングであり、特定の計算方法およびストア・エンティティーが与えられています。
- calusage_id は、この計算コードがどのような種類の計算に使用されるかを示します。たとえば、計算コードは次の通貨金額の 1 つを計算するために使用することができます。
  - 割引額 (-1)
  - 配送料 (-2)
  - 消費税 (-3)
  - 配送税 (-4)
  - クーポン (-5)
- storeent_id は、この計算コードに関連したストア・エンティティーです。
- groupby は、計算時にオーダー・アイテムをグループ化する方法を計算コード結合メソッドに指示するビット・フラグです。 0 を指定した場合、グループ化されません (適用可能なすべてのオーダー・アイテムが 1 つのグループに属することになります)。詳細は、 WebSphere Commerce オンライン・ヘルプの『*CALCODE テーブル: 詳細*』を参照してください。
- published は、計算コードが発行されるかどうかを指定します。
  - 0 = 発行されない (一時的に使用不可)
  - 1 = 発行する
  - 2 = 削除のマーク (発行されない)
- sequence は、計算コードが計算される順序です。計算コード、低位から高位の順で計算されて適用されます。 2 つの計算コードのシーケンス番号が同じなら、 calcode_id の小さい計算コードから順に計算されます。

- calmethod_id は、この計算コードの税金額を計算する方法を定義する計算コード計算メソッドです。使用する計算コード計算メソッドを決定するには、以下のようにします。
  - WebSphere Commerce オンライン・ヘルプの『CALMETHOD テーブル』を参照します。 SUBCLASS 列の説明を参照してください。『CALMETHOD テーブル: 詳細』のリンクをクリックします。このテーブルには、使用できる CALMETHOD のタイプがリストされています。計算コード計算メソッド・タイプは 3 です。
  - ブートストラップ・ファイル wcs.bootstrap_xx_XX.xml を開きます (xx_XX はロケールのコード)。ブートストラップ・ファイルは以下のディレクトリーに置かれています。
    - NT drive:¥WebSphere¥CommerceServer¥schema¥xml
    - 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥schema¥xml
    - AIX /usr/WebSphere/CommerceServer/schema/xml
    - Solaris /opt/WebSphere/CommerceServer/schema/xml
    - Linux /opt/WebSphere/CommerceServer/schema/xml
    - 400 /qibm/proddata/WebCommerce/schema/xml
  - 使用できる計算メソッドをリストしているセクション (CALMETHOD) を見つけます。
  - calusage_ID が税 (消費税の場合は -3、配送税の場合は —4) の計算メソッドを見つけます。
  - サブクラスが 3 の計算メソッドを見つけます。この計算メソッドは —43 です。
- calmethod_id_app は、関連する OrderItems の計算済みの金額を保管する CalculationCodeApplyMethod です。 calmethod_id に記述されているメソッドを使用して、使用される計算コード適用メソッドを判別します。
  - calmethod_id_app="-44" は、消費税の場合の CalculationCodeApplyMethod です。
- calmethod_id_qfy は、この計算コードと関連したオーダー・アイテムを定義する CalculationCodeQualifyMethod です。 calmethod_id に記述されているメソッドを使用して、使用される計算コード限定メソッドを判別します。
  - calmethod_id_qfy="-42" は、消費税の場合の CalculationCodeQualifyMethod です。
- display level は、この計算コードが計算する金額を、以下の値で表示するかどうかを決定します。
  - 0 = オーダー・アイテム
  - 1 = オーダー
  - 2 = 商品
  - 3 = アイテム
  - 4 = 契約

- flags は、この計算コードの CalculationCodeQualifyMethod が呼び出されるかどうかを指定します。
  - 0 = 制限なし。メソッドは呼び出されません。
  - 1 = 制限あり。メソッドは呼び出されます。

b. 以下の例を参考にして、XML ファイルの CALCODEDSC テーブルに計算コードの説明情報を追加します。多文化ストアを作成する場合は、ロケール固有の XML ファイルにこの情報を組み込む必要があります。

```
<calcodedsc
calcode_id="@calcode_id_3"
description="Vitamins
language_id="&en_US"
longdescription= "In Ontario vitamins are taxed federally, but
not provincially."
/>
```

ここで、

- calcode_id は、この情報が適用される計算コードです。
- description は、計算コードの簡略説明です。
- language_id は、この情報が適用される言語です。
- longdescription は、計算コードの詳細記述です。

c. ストアの中で使用される計算コードごとに、ステップ a と b を繰り返します。

7. ストアの計算ルールを定義します。

a. 以下の例を参考にして、XML ファイルの CALRULE テーブルで計算ルールをセットアップします。

```
<calrule
calrule_id="@calrule_id_10"
calcode_id="@calcode_id_3"
startdate="1900-01-01 00:00:00.000000"
taxcgry_id="@taxcgry_id_1"
enddate="2100-01-01 00:00:00.000000"
 flags="1"
identifier="1"
combination="2"
calmethod_id="-47"
calmethod_id_qfy="-46"
/>
```

ここで、

- calrule_id は、生成される固有の ID です。
- calcode_id は、この計算ルールを含む計算コードです。
- startdate は、この計算ルールが有効になる時刻です。
- taxcgry_id は、この計算ルールが有効な課税カテゴリーです。
- enddate は、この計算ルールの停止が有効になる時刻です。
- combination は、この計算ルールを他の計算ルールと結合する方法を決定するために CalculationRuleCombineMethod によって使用されます。詳細は、『CALRULE テーブル: 詳細』を参照してください。
- identifier は、この計算ルールとその計算コードを組み合わせて識別します。

- flags は、デフォルトの CalculationRuleCombineMethod インプリメンテーションによって実行される、特殊な処理を示す以下のビット・フラグを示します。詳細は、WebSphere Commerce オンライン・ヘルプの『CALRULE テーブル』を参照してください。
- calmethod_id は、一連のオーダー・アイテムの通貨結果を計算する CalculationRuleCalculateMethod です。使用する計算ルール計算メソッドを決定するには、以下のようにします。
  - WebSphere Commerce オンライン・ヘルプの『CALMETHOD テーブル』を参照します。 SUBCLASS 列の説明を参照してください。『CALMETHOD テーブル: 詳細』のリンクをクリックします。このテーブルには、使用できる CALMETHOD のタイプがリストされています。計算ルール計算メソッドは 7 です。
  - ブートストラップ・ファイル wcs.bootstrap_xx_XX.xml を開きます (xx_XX はロケールのコード)。ブートストラップ・ファイルは以下のディレクトリーに置かれています。
    - NT drive:¥WebSphere¥CommerceServer¥schema¥xml
    - 2000 drive:¥Program Files¥WebSphere¥CommerceServer ¥schema¥xml
    - AIX /usr/WebSphere/CommerceServer/schema/xml
    - Solaris /opt/WebSphere/CommerceServer/schema/xml
    - Linux /opt/WebSphere/CommerceServer/schema/xml
    - 400 /qibm/proddata/WebCommerce/schema/xml
  - 使用できる計算メソッドをリストしているセクション (CALMETHOD) を見つけます。
  - calusage_ID が税 (消費税の場合は -3、配送税の場合は -4) の計算メソッドを見つけます。
  - サブクラスが 7 の計算メソッドを見つけます。この計算メソッドは -47 です。
- calmethod_id_qfy は、 CalculationRuleCalculateMethod に送信する OrderItems のセットを決定する CalculationRuleQualifyMethod です。calmethod_id に記述されているメソッドを使用して、使用される計算ルール限定メソッドを判別します。

b. ストアの中で使用される計算ルールごとに、ステップ a を繰り返します。個々の計算コードについて、適用可能な各課税カテゴリーごとに 1 つずつ、複数の計算ルールがある場合もあるので注意してください。たとえば、calcode_id="@calcode_id_1" を、複数の calrule_ids と関連付けることができます。

8. ストアの計算スケールを定義します。

   計算スケールは、計算に適用される範囲のセットです。これらの範囲がスケールを構成します。

   a. 以下の例を参考にして、XML ファイルの CALSCALE テーブルで計算スケールをセットアップします。

```
<calscale
calscale_id="@calscale_id_19"
code="Sales Tax 1"
 storeent_id="@storeent_id_1"
calusage_id="-3"
setccurr="USD"
calmethod_id="-53"
/>
```

ここで、

- calscale_id は、生成される固有の ID です。
- code は、この計算スケールを固有に識別する文字ストリングであり、特定の計算方法およびストア・エンティティーが与えられています。
- storeent_id は、この計算スケールを含むストア・エンティティーです。
- calusage_id は、この CalculationScale がどのような種類の計算に使用されるかを示します。 たとえば、CalculationScale は次の通貨金額の 1 つを計算するために使用することができます。
  - 割引額 (-1)
  - 配送料 (-2)
  - 消費税 (-3)
  - 配送税 (-4)
  - クーポン (-5)
- setccurr が指定されると、これはこの計算スケールの計算範囲オブジェクトの範囲開始値の通貨を示します。 CalculationScaleLookupMethod はこの通貨で「ルックアップ番号」を戻すことになります。この場合、それは指定されていません。 CalculationScaleLookupMethod は、オーダーの通貨によりルックアップ番号を戻します。スケール範囲の開始値が 0 でない場合以外は、通貨を指定する必要はありません。
- calmethod_id は、一連のオーダー・アイテムが指定された CalculationScaleLookupMethod です。通貨金額を計算するために計算スケールで使用できるルックアップ番号、基本通貨値、結果乗数、および正確な重量のセットを決定します。使用する CalculationScaleLookupMethod を決定するには、以下のようにします。
  - WebSphere Commerce オンライン・ヘルプの『CALMETHOD テーブル』を参照します。 SUBCLASS 列の説明を参照してください。『CALMETHOD テーブル: 詳細』のリンクをクリックします。このテーブルには、使用できる CALMETHOD のタイプがリストされています。 MonetaryCalculationScaleLookupMethod メソッドは 9 です。
  - ブートストラップ・ファイル wcs.bootstrap_xx_XX.xml を開きます (xx_XX はロケールのコード)。ブートストラップ・ファイルは以下のディレクトリーに置かれています。
    - NT drive:¥WebSphere¥CommerceServer¥schema¥xml
    - 2000 drive:¥Program Files¥WebSphere¥CommerceServer ¥schema¥xml
    - AIX /usr/WebSphere/CommerceServer/schema/xml
    - Solaris /opt/WebSphere/CommerceServer/schema/xml

- Linux /opt/WebSphere/CommerceServer/schema/xml
- 400 /qibm/proddata/WebCommerce/schema/xml

– 使用できる計算メソッドをリストしているセクション (CALMETHOD) を見つけます。
– calusage_ID が税 (消費税の場合は -3、配送税の場合は -4) の計算メソッドを見つけます。
– サブクラスが 9 の計算メソッドを見つけます。サブクラス 9 の計算メソッドは複数あります。必要に合ったメソッドを選出します。

詳細は、WebSphere Commerce オンライン・ヘルプの『CALSCALE テーブル』を参照してください。

b. ストアで使用される計算スケールごとに、ステップ a を繰り返します。
c. 以下の例を参考にして、 XML ファイルの CALSCALDS テーブルに計算スケールの説明情報を追加します。多文化ストアを作成する場合は、ロケール固有の XML ファイルにこの情報を組み込む必要があります。

```
<calscaleds
calscale_id="@calscale_id_19"
description="Sales Tax 5% "
language_id="&en_US"
/>
```

ここで、

- calscale_id は、この記述の適用対象となる計算スケールです。
- description は、計算の実行方法を顧客に対して表示するための、計算スケールの簡略説明です。たとえば、「1 kg 当たり 0.1 ドル、最低料金 5 ドル」や、「5 個以上お買い上げの場合 10 % オフ」などです。
- language_id は、この情報の表示に使用される言語です。

d. ストアで使用される計算スケールごとに、ステップ c を繰り返します。

9. 計算スケールの計算範囲を定義します。
   a. 以下の例を参考にして、XML ファイルの CALRANGE テーブルで計算範囲をセットアップします。

```
<calrange
calrange_id="@calrange_id_37"
calscale_id="@calscale_id_19"
calmethod_id="-59"
rangestart="0.00000"
cumulative="0"
/>
```

ここで、

- calrange_id は、生成される固有の ID です。
- calscale_id は、この計算範囲を含む計算スケールです。
- calmethod_id は、 CalculationRangeLookupResult からの通貨金額を決定する CalculationRangeMethod です。たとえば、FixedAmountCalculationRangeCmd、 PerUnitAmountCalculationRangeCmd、または PercentageCalculationRangeCmd。 CalculationRangeMethod を決定するには、以下のようにします。

– WebSphere Commerce オンライン・ヘルプの『CALMETHOD テーブル』を参照します。 SUBCLASS 列の説明を参照してください。『CALMETHOD テーブル: 詳細』のリンクをクリックします。このテーブルには、使用できる CALMETHOD のタイプがリストされています。 CalculationRangeMethod は 10 です。

– ブートストラップ・ファイル wcs.bootstrap_xx_XX.xml を開きます (xx_XX はロケールのコード)。ブートストラップ・ファイルは以下のディレクトリーに置かれています。

- NT drive:¥WebSphere¥CommerceServer¥schema¥xml
- 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥schema¥xml
- AIX /usr/WebSphere/CommerceServer/schema/xml
- Solaris /opt/WebSphere/CommerceServer/schema/xml
- Linux /opt/WebSphere/CommerceServer/schema/xml
- 400 /qibm/proddata/WebCommerce/schema/xml

– 使用できる計算メソッドをリストしているセクション (CALMETHOD) を見つけます。

– calusage_ID が税 (消費税の場合は -3、配送税の場合は -4) の計算メソッドを見つけます。

– サブクラスが 10 の計算メソッドを見つけます。サブクラス 10 にはいくつかの計算メソッドがあります。必要に応じて 1 つ選択してください。

- rangestart は、ルックアップ番号がこの RANGESTART 以上の場合、または RANGESTART が NULL の場合に、この行がルックアップ番号にマッチングする、というように使用されます。
- cumulative は、以下のとおりです。
  – 0 = 最高の RANGESTART 値に一致する CalculationRange だけが使用される。
  – 1 = 一致する CalculationRanges はすべて使用される。計算された通貨金額が合計されて、最終結果が得られます。

詳細は、WebSphere Commerce オンライン・ヘルプの『CALRANGE テーブル』を参照してください。

b. ストアの中で使用される計算スケールに関連した計算範囲ごとに、ステップ a を繰り返します。上記の例の場合、すべての金額の税率は同じなので、範囲は 1 つだけです。

10. 計算スケールの計算ルックアップ値を定義します。計算ルックアップ値は、計算スケールに関連した値です。たとえば、計算スケールに、レストランで消費される肉に対するオンタリオ州の消費税に関して、以下のような範囲と対応する税率が含まれているとします。

- $0.00 ～ $3.99 の場合、税率 0.00%
- $4.00 以上の場合、税率 8.00%

この場合、ルックアップ値は 0.00 と 8.00 です。

a. 以下の例を参考にして、XML ファイルの CALRLOOKUP テーブルで計算ルックアップをセットアップします。

```
<calrlookup
 calrlookup_id="@calrlookup_id_37"
 calrange_id="@calrange_id_37"
 value="5.00"
/>
```

ここで、

- calrlookup_id は、生成される固有の ID です。
- calrange_id は、計算範囲ルックアップ結果を含む計算範囲です。
- value は、計算範囲ルックアップ結果の値です。この値は、計算範囲の計算範囲メソッドによって使用され、通貨結果が決定されます。この例の場合、税率は 5.00% です。

詳細は、WebSphere Commerce オンライン・ヘルプの『CALRLOOKUP テーブル』を参照してください。

b. ストアの中で使用される計算スケールに関連したルックアップ値ごとに、ステップ a と b を繰り返します。この例の場合、CALRLOOKUP.SETCCURR が NULL なので CALRLOOKUP 値は 1 つだけであり、すべての金額について税率が同じなので CALRANGE は 1 つだけです。

11. 計算ルールと計算スケールを関連付けます。

a. 以下の例を参考にして、XML ファイルの CRULESCALE テーブルで計算スケールと計算ルールを関連付けます。

```
<crulescale
 calrule_id="@calrule_id_10"
 calscale_id="@calscale_id_19"
/>
```

ここで、

- calrule_id は計算ルールです。
- calscale_id は計算スケールです。

b. 関連付ける計算スケールとルールごとに、ステップ a を繰り返します。上記の例の場合、各計算ルールの計算スケールは 1 つだけです。

注: 税率が購入金額に応じて異なる場合、範囲開始値が 0 以外のスケールを作成することが必要になります。その場合、サポートされている通貨のうち換算率を設定していないものについて (CURCONVERT テーブルを参照)、 CALSCALE.SETCCURR を該当する通貨に設定することにより、サポートされている通貨ごとに計算スケールを作成し、さらにそれらすべてをその特定の課税カテゴリーの計算ルールに関連付ける必要があります。たとえば、オンタリオ州の消費税は、$4.00 の場合には課せられません。米国ドルでの肉の販売をサポートするストアの場合は、米国ドルからカナダ・ドルへの換算を設定するか、または該当する範囲開始値 (おそらく $6.00 USD) を使用して別個の税額計算を作成し、オーダーの通貨に応じて、該当する計算スケールだけが使用されることになります。

@ と **&** の使用法の詳細については、 313 ページの『付録 B. データの作成』を参照してください。

税フルフィルメント資産の作成

ストアで税資産が正しく機能するためには、ストアの中の課税管轄区域グループとストアで使用される配送センターを関連付けてから、その両方に計算ルールを関連付けなければなりません。

税資産を配送センターに関連付けるには、その前にフルフィルメント資産を作成しなければなりません。フルフィルメント資産の作成の詳細については、 116 ページの『WebSphere Commerce での配送資産の作成』を参照してください。

フルフィルメント資産を作成し終えたら、TAXJCRULE テーブルに情報を追加して、税資産を関連付けます。以下のようにします。

1. サンプル・ストアの税フルフィルメント資産を作成するために使用される XML ファイルを確認します。サンプル・ストアのすべてのファイルは、対応するストア・アーカイブ・ファイルの中にあります。

   ストア・アーカイブ・ファイルは以下のディレクトリーにあります。

   - NT `drive:¥WebSphere¥CommerceServer¥samplestores`
   - 2000 `drive:¥Program Files¥WebSphere¥CommerceServer¥samplestores`
   - AIX `/usr/WebSphere/CommerceServer/samplestores`
   - Solaris `/opt/WebSphere/CommerceServer/samplestores`
   - Linux `/opt/WebSphere/CommerceServer/samplestores`
   - 400 `/qibm/proddata/WebCommerce/samplestores`

     注: WebSphere Commerce オンライン・ヘルプには、サンプル・ストアに含まれる各データ資産についての情報が記載されています。

   各サンプル・ストアには 1 つの `taxfulfill.xml` ファイルが組み込まれています。これには税に関する情報が入っています。ストア・アーカイブの `taxfulfill.xml` ファイルを表示するには、ZIP プログラムを使ってこれを解凍します。 `taxfulfill.xml` ファイルは、data ディレクトリーにあります。
2. 313 ページの『付録 B. データの作成』の情報を確認します。
3. サンプル・ストア・アーカイブの `taxfulfill.xml` ファイルの 1 つをコピーするか、新しいファイルを作成することにより、`taxfulfill.xml` ファイルを作成します。詳しくは、`taxfulfill.xml` に対応する DTD ファイルを参照してください。 DTD ファイルは以下のディレクトリーにあります。

   - NT `drive:¥WebSphere¥CommerceServer¥xml¥sar`
   - 2000 `drive:¥Program Files¥WebSphere¥CommerceServer¥xml¥sar`
   - AIX `/usr/WebSphere/CommerceServer/xml/sar`
   - Solaris `/opt/WebSphere/CommerceServer/xml/sar`

- Linux /opt/WebSphere/CommerceServer/xml/sar
- 400 /qibm/proddata/WebCommerce/xml/sar

4. 以下の例を参考にして、XML ファイルの TAXJCRULE テーブルに情報を追加します。

```
<taxjcrule
taxjcrule_id="@taxjcrule_id_1"
calrule_id="@calrule_id_10"
ffmcenter_id="@ffmcenter_id_1"
jurstgroup_id="@jurstgroup_id_2"
precedence="0"
/>
```

ここで、

- taxjcrule_id は、生成される固有の ID です。
- calrule_id は、使用される計算ルールです。
- ffmcenter_id は、配送センターです。これが NULL の場合、この関連はすべての配送センターに適用されます。
- jurstgroup_id は課税管轄区域グループです。これが NULL の場合、この関連はすべての課税管轄区域グループに適用されます。
- precedence は、配送先住所が、同一の配送センターと配送モードのために指定された複数の課税管轄区域グループに属する場合に、TAXJCRULE.PRECEDENCE 値が最大の計算ルールにのみ限定されることを示します。

5. ストアの管轄区域グループ、配送センター、およびルールの関連ごとに、ステップ 3 を繰り返します。

@ と **&** の使用法の詳細については、 313 ページの『付録 B. データの作成』を参照してください。

ストア - カタログ - 税資産の作成

ストアの税を商品やサービスと関連付けるには、ストアに組み込まれている契約ごとに、ストアの計算コードとカタログ・エントリーを関連付けなければなりません。

ストア - カタログ - 税資産を作成するには、その前にストア資産とカタログ資産を作成しなければなりません。ストア資産の作成の詳細については、 47 ページの『XML ファイルによるストア・データ資産の作成』を参照してください。カタログ資産の作成の詳細については、 82 ページの『ストア・カタログ資産の表示』を参照してください。

ストア - カタログ - 税資産を作成するには、以下のようにします。

1. サンプル・ストアのストア - カタログ - 税資産を作成するために使用される XML ファイルを確認します。サンプル・ストアのすべてのファイルは、対応するストア・アーカイブ・ファイルの中にあります。

   ストア・アーカイブ・ファイルは以下のディレクトリーにあります。

   - NT drive:¥WebSphere¥CommerceServer¥samplestores

- 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥samplestores
- AIX /usr/WebSphere/CommerceServer/samplestores
- Solaris /opt/WebSphere/CommerceServer/samplestores
- Linux /opt/WebSphere/CommerceServer/samplestores
- 400 /qibm/proddata/WebCommerce/samplestores

  個々のサンプル・ストアには 1 つの store-catalog-tax.xml ファイルが組み込まれており、このファイルには配送フルフィルメント情報が組み込まれています。ストア・アーカイブの store-catalog-tax.xml ファイルを表示するには、 ZIP プログラムを使用してこれを解凍します。 store-catalog-tax.xml ファイルは、data ディレクトリーにあります。

  注: WebSphere Commerce オンライン・ヘルプには、サンプル・ストアに含まれる各データ資産についての情報が記載されています。

2. 313 ページの『付録 B. データの作成』の情報を確認します。
3. サンプル・ストア・アーカイブの store-catalog-tax.xml ファイルの 1 つをコピーするか、新しいファイルを作成することにより、store-catalog-tax.xml ファイルを作成します。詳しくは、store-catalog-tax.xml に対応する DTD ファイルを参照してください。 DTD ファイルは以下のディレクトリーにあります。
   - NT drive:¥WebSphere¥CommerceServer¥xml¥sar
   - 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥xml¥sar
   - AIX /usr/WebSphere/CommerceServer/xml/sar
   - Solaris /opt/WebSphere/CommerceServer/xml/sar
   - Linux /opt/WebSphere/CommerceServer/xml/sar
   - 400 /qibm/proddata/WebCommerce/xml/sar
4. CATENCALCD テーブルに情報を追加して、ストア - カタログ - 税の関係を作成します。次の例を参考にしてください。

```
<catencalcd
   calcode_id="@calcode_id_3"
   catencalcd_id="@catencalcd_id_3"
   store_id="@storeent_id_1"
/>
```

   ここで、

   - calcode_id は計算コードです。
   - catencalcd_id は、生成される固有の ID です。
   - store_id は、ストアです。

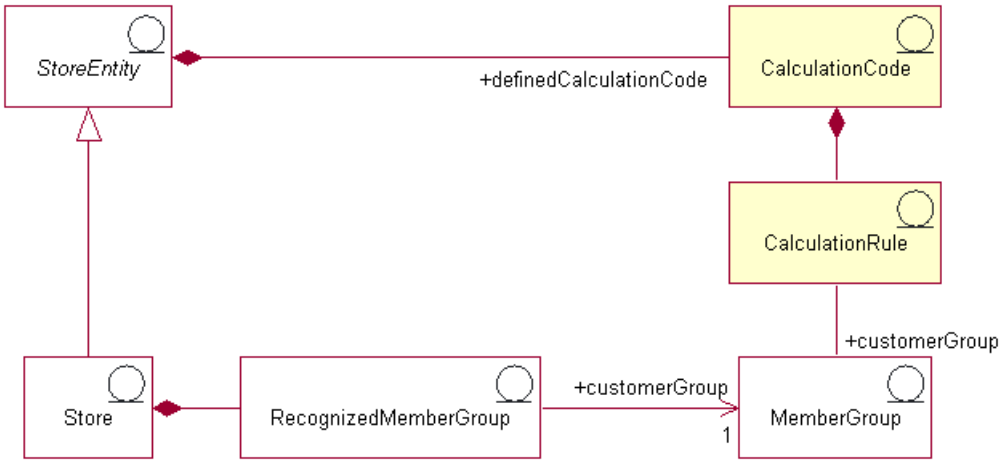
@ と **&** の使用法の詳細については、 313 ページの『付録 B. データの作成』を参照してください。

第 20 章 割引資産

割引は、顧客の購買を促進するきっかけになる可能性があります。パーセンテージ割引 (10% 割引など) または固定額の割引 (15 ドルの値引きなど) を提示することができます。割引は、特定の商品または買い物全体のどちらに対しても適用することができます。たとえば、高齢者に対して 20% の割引を実施したり、多数の赤い野球キャップが在庫にあれば、期間限定で赤いキャップを 25% 引きで販売することができます。

WebSphere Commerce の割引について

以下の図は、WebSphere Commerce Server の割引構造を示しています。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、29 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則については、311 ページの『付録 A. UML の凡例』を参照してください。

計算コード

割引 は割引*計算コード* で表され、それを使用して計算されます。割引計算コードは、オーダー・アイテムの割引が計算される方法を示します。

計算コードは、ストア・エンティティーに属するものです。 1 つのストア・エンティティー内に複数の計算コードを定義できます。ストア・エンティティーを削除すると、そのストア・エンティティーに関連して定義されている計算コードも削除されます。

各割引計算コードには、割引の有効な期間を定義する開始日と終了日があります。また、割引計算コード有効なメンバー・グループを定義する 1 つ以上のメンバー・グループに関連付けることもできます。

割引計算コードは、1 つ以上のカタログ・エントリーおよびカタログ・グループに、付加することができます。カタログ・グループに計算コードを付加すると、カタログ・グループのすべてのカタログ・エントリーに直接付加したのと同じ効果があります。しかし、カタログ・グループ A にカタログ・グループ B が含まれている場合、カタログ・グループ A の割引計算コードは、カタログ・グループ B 内の製品やアイテムには関連付けられません。

カタログ・エントリーやカタログ・グループに複数の割引が関連付けられる場合があります。 1 つのオーダーに複数の割引コードが適用可能な場合、割引計算は、計算コード順序属性の昇順に実行されます。

注: 割引の順序は、割引後の金額をさらに割り引くことをインプリメントするように定義してください。

以下の方法の 1 つにより、オーダー・アイテムは計算のためにグループ化されます。

- 取引条件別
- 商品別
- オファー別
- 配送先住所別

詳しくは、WebSphere Commerce オンライン・ヘルプを参照してください。

計算ルール

各計算コードには計算ルールのセットがあります。これは、計算が実行される条件を定義します。個々の割引計算ルールは 1 つ以上のメンバー・グループに関連付けられ、そのメンバー・グループに対して割引が有効になります。メンバー・グループに対して一度に複数の割引があてはまる場合があります。

注: 適格なメンバー・グループが計算コード・レベルで定義されている場合、それを計算ルールのレベルで再度定義する必要はありません。

WebSphere Commerce での割引資産の作成

WebSphere Commerce を使用して作成したストアに割引を作成する主な方法は、WebSphere Commerce Accelerator の「割引」ウィザードを使用することです。WebSphere Commerce Accelerator を使用して割引を作成することの詳細については、 WebSphere Commerce のオンライン・ヘルプを参照してください。

XML ファイルを使用して割引を作成し、その後でローダー・パッケージによってロードするか、ストア・サービスによって発行することもできます。しかし、この方法で作成された割引や、以前のバージョンからのマイグレーション時にインポートされた割引は、機能的には問題ないものの、WebSphere Commerce Accelerator には正しく表示されない場合があります。

第 21 章 在庫資産

在庫には、配送センター内で物理的に存在可能なものがすべて含まれます。満たすべき在庫タイプの特定の定義 (たとえば、アイテム、商品、SKU、バンドル、パッケージなど) はありますが、これらはすべて在庫と見なされます。商品のフルフィルメントは、「商品」ウィザードと「商品」ノートブックで構成します。これには、在庫を追跡記録するオプション、バック・オーダーを許可するオプション、バック・オーダーを強制するオプション、別々にリリースするオプション、および商品を返品不可にするオプションが含まれています。 WebSphere Commerce アクセラレーターは、受け取ることのできる在庫の、 2 つの主要タイプを区別します。

- 予測在庫レコードと関連付けられた予測在庫
- 特別在庫、または予測として記録されていない在庫

予測在庫は取引先から受け取り、一般に、購入オーダーと一緒に支払われます。WebSphere Commerce アクセラレーターは予測在庫レコードを使用して予測在庫をトラッキングし、またこれを使用することによって、外部 ID (通常は外部システムからの購入オーダー番号) を記録することができます。これにより簡単に、到着したものと到着していないもの、およびオーダーした在庫を把握していくことができます。予測在庫詳細とは、予測在庫レコードの商品に関する詳細のことです。たとえば、商品を見込んでいる配送センター、予測受け取り日付、予測数量、コメントなどです。

在庫がいったん受け取られると、それに対応する予測在庫レコードは削除できません。また、予測在庫の詳細情報も、それに含まれる在庫の一部でも受け取られると、変更や削除ができなくなります。

配送センターで使用可能な在庫に対してオーダーが発行された場合、オーダー・システムは在庫をそれらのオーダーに割り振ります。在庫をオーダーに割り当てると、この在庫はオーダー・システムで使用できなくなります。オーダーがキャンセルされると、在庫はまた使用できるようになります。購入可ではない在庫がオーダーされた場合、バックオーダーを作成できます。バック・オーダーを実施するために使用できる予測在庫がある場合、予測在庫はバック・オーダーに対して割り振られ、顧客には予測配送日付が知らされます。

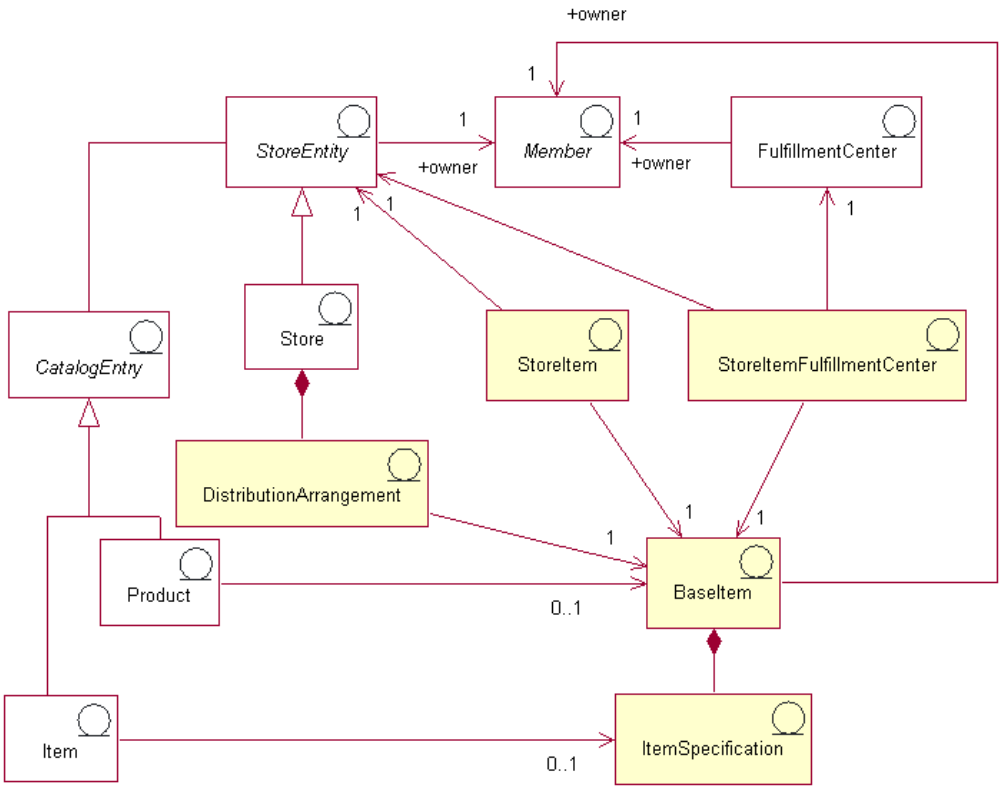
対応する予測在庫レコードがない在庫が配送センターに届くと、特別在庫受領書が作成されます。これは、予期せずに在庫が到着したために生じたのかもしれません。あるいは、予測在庫レコードを使用して在庫受け取りを記録しないようにマーチャントまたはセラーが選択した可能性もあります。

注: 在庫受け取りが予測在庫受け取りか特別在庫受け取りかに関係なく、商品が受け取られるためにはそれが WebSphere Commerce に存在しなければなりません。

WebSphere Commerce の在庫資産について

在庫資産を理解するには、在庫とストアの間の関係を理解することが必要です。これは、情報モデルを使って説明できます。次に、ストアおよび他の資産と在庫との関係、および関連を説明します。以下の図は、ATP 在庫の関係と非 ATP 在庫の間の関係と関連の全体像を示したものです。非 ATP 在庫とは、以前のバージョンの商品を在庫にしたときの扱い方のことです。ストアが ATP フィーチャーを使用しないことにした場合は、引き続きこれを使用できます。個々の図とその関連については、後で説明します。 ATP 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。

ATP 在庫



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、 29 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則の詳細については、 311 ページの『付録 A. UML の凡例』を参照してください。

基本アイテム

基本アイテム とは、名前と説明を共有する、商品の一般的なファミリーです。基本アイテムはもっぱらフルフィルメントのために使用されます。ストアごとに固有なものではありません。カタログの商品を表す各カタログ・エントリーには、フルフィルメントを目的として、対応する基本アイテムがあります。基本アイテムは BASEITEM テーブルに定義されています。

アイテム仕様

アイテム仕様 とは、すべての属性の値が定義されている基本アイテムのことです。カタログのアイテムを表す各カタログ・エントリーには、フルフィルメントを目的として、対応するアイテム仕様があります。

カタログ・エントリー

商品やアイテムは*カタログ・エントリー* です。カタログ・エントリーは、ストア・エンティティーと関連付けられます。結果として、商品やアイテムなどのカタログ・エントリーがストアに表示されます。

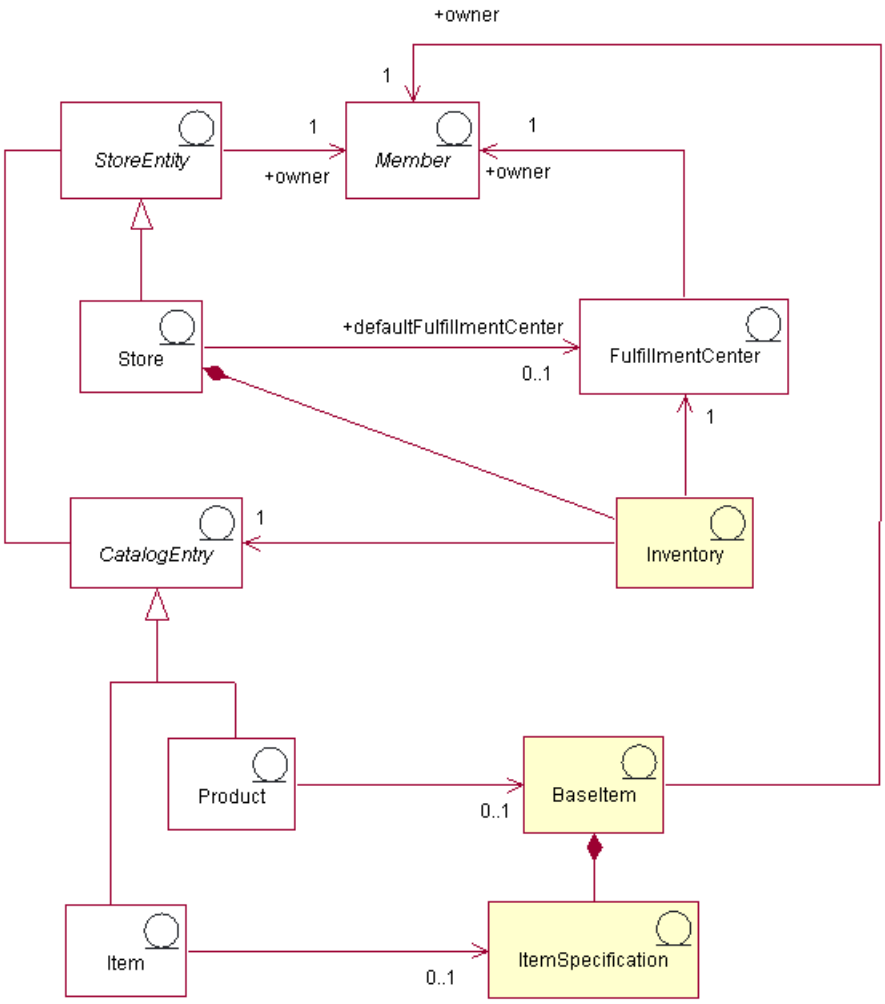
配送手配

配送手配 は基本アイテムと関連付けられ、これによりストアは独自の在庫を販売できるようになります。配送手配は、DISTARRANG テーブルに保管されます。

ストア・アイテム

ストア・アイテム は、特定のストアやストア・グループが、特定の基本アイテムの指定アイテムの在庫を割り振る方法を制御する属性を表します。これには、バック・オーダーや在庫調査を行えるようにするかどうかなどが含まれます。STORITMFFC テーブルには、オーダー・アイテムがフルフィルメントのためにリリースされた時点から、これが顧客に配送されるまでにかかる見積秒数が定義されます。このテーブルにデータが読み込まれるのは、ストアがストア・アイテムの FFMCENTER デフォルト配送オフセットにオーバーライドを定義することを望む場合だけです。

非 ATP 在庫



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、 29 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則の詳細については、 311 ページの『付録 A. UML の凡例』を参照してください。

非 ATP 在庫の図でも、基本アイテムが中心になっています。基本アイテムと商品、アイテム、およびカタログ・エントリーとの関係は、一般の在庫の図と同じです。この図でも、基本アイテムはメンバーによって所有されています。そのメンバーが基本アイテムを定義すると、これをストアで販売できるようになります。しかしながら、この図には、配送手配、ストア・アイテムの関連、ストア・アイテム配送センターがありません。

配送センター

在庫には 1 つの*配送センター* と 1 つのストアが関連付けられます。ストアは 1 つのデフォルト配送センターを指定できます。基本アイテムと同様に、配送センターもメンバーが所有し、その所有権をストアと共用します。フルフィルメント資産の詳細については、 113 ページの『第 11 章 フルフィルメント資産』を参照して

ください。

WebSphere Commerce Server の在庫資産の構造の詳細については、WebSphere Commerce オンライン・ヘルプで『在庫オブジェクト・モデル』と『データ・モデル』を参照してください。

WebSphere Commerce での在庫資産の作成

在庫は運用データなので、毎日、顧客がストアから商品を購入したりアイテムを返品したりするたびに変更されます。商品を販売したり、配送センターがサプライヤーから新規在庫を受け取ったりするたびに、在庫レベルは変動します。 WebSphere Commerce Accelerator によって、以下の関連タスクを実行することができます。

- 予測在庫の記録
- 取引先からの予測在庫と特別在庫の受け取り
- 在庫調整
- 返品記録の保守
- 返品理由の保守
- 顧客からの返品在庫の受け取り
- 返品在庫の処分の管理

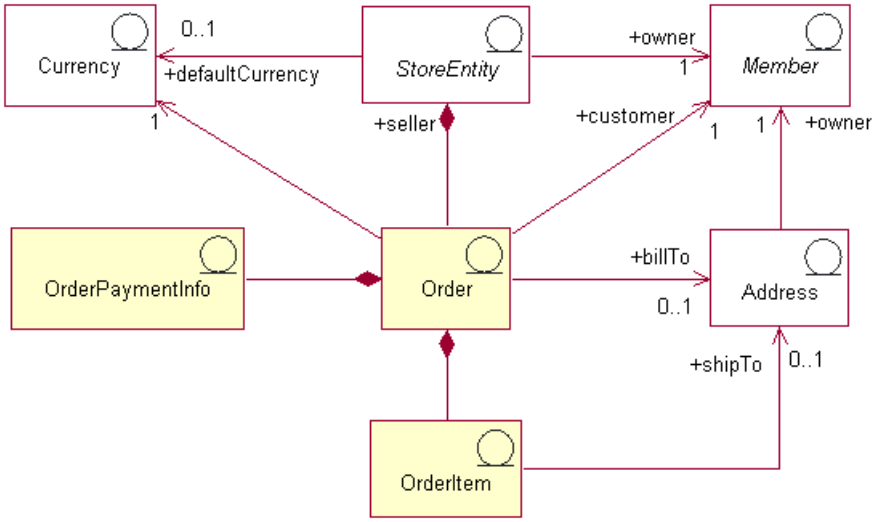
WebSphere Commerce Accelerator を使用して在庫を管理することの詳細については、WebSphere Commerce のオンライン・ヘルプを参照してください。

第 22 章 オーダー資産

WebSphere Commerce システムのオーダー資産は、ショッピング・カート、オーダー管理、およびオーダー・プロセッシング機能を備えています。オーダー・プロセッシング機能には、クイック・オーダーや即時購入、スケジュール・オーダー、複数保留オーダー、再オーダー、オーダーの分割、およびバック・オーダーが含まれます。また、価格設定、税、支払い、およびフルフィルメントなどの関連サービスも、オーダー資産の一部です。

WebSphere Commerce のオーダー資産について

以下の図は、WebSphere Commerce Server のオーダー資産を示しています。図の後に、個々の資産の説明があります。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、29 ページの『ストア・データの情報モデル』を参照してください。

オーダーおよびオーダー・アイテム

WebSphere Commerce システムの場合、顧客やショッパーにとっては、オーダー は選択した商品のリストで (たとえば、オーダーに 2 冊の本と CD を含めることができます)、そのリスト上にある各商品がオーダー・アイテム です (たとえば、本や CD はそれぞれ、同じオーダーのオーダー・アイテムです)。顧客がストアでオーダーを発行する際には、ストアから送り状を送付する先となる、請求先住所を指定しなければなりません。各オーダーごとに 1 つの通貨 ID が関連付けられます。ストアの側から見ると、オーダーはオーダー・アイテムのリストです。これはストアのデータの一部です。

通貨

ストアでは、1 種類の*通貨* で価格を表示することもできますし、複数の通貨を使用することもできます。個々のストアで*デフォルト通貨* も定義しなければなりません。また、顧客が*ショッピング通貨* を選択できるようにすることもできます。ショッピング通貨がストアのデフォルト通貨と同じである場合、これは STOREENT テーブルですでにサポートされています。ショッピング通貨がストアのデフォルト通貨でない場合、CURLIST テーブルにその通貨を追加しなければなりません。顧客はショッピング通貨を使用して、ストアでオーダーを発行します。

支払い情報

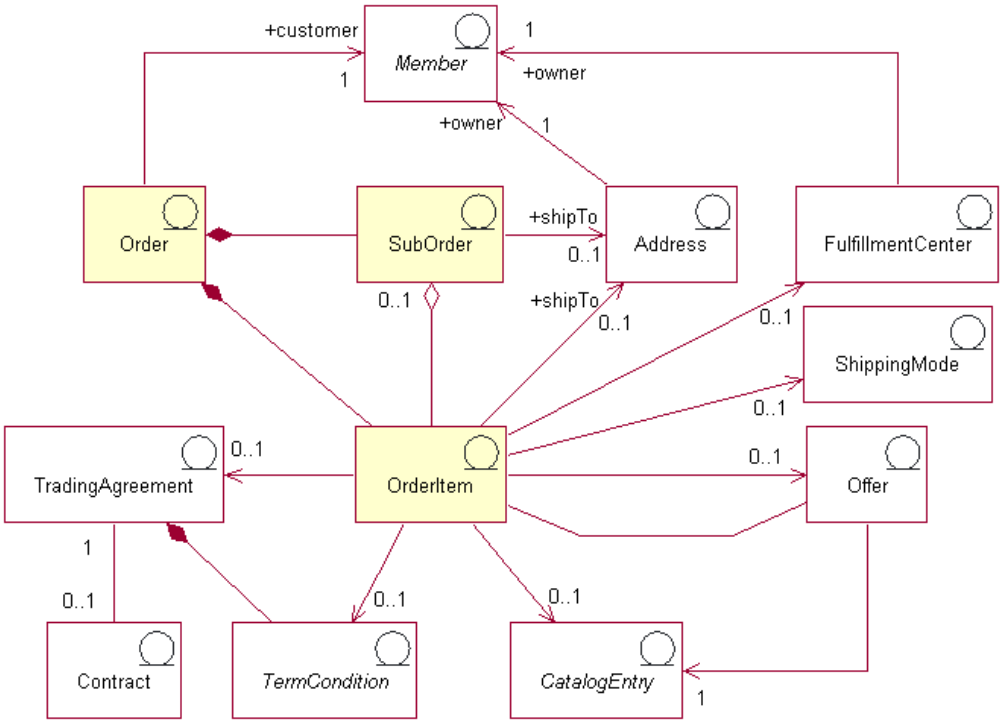
いったん顧客が希望のショッピング通貨を選択したら、支払いはすべてその通貨で処理されます。ストアの支払いのサポートとポリシーに応じて、顧客はオンライン決済 (この場合、顧客はインターネットを介してストアのサイトに支払い情報を提供する) かオフライン決済 (この場合、顧客はインターネット・チャネルを介さずに、電話や FAX などで支払い情報を提供する) のどちらかを使用して、支払いと購入を行えます。オンライン決済とオフライン決済のどちらの方式であるかに関係なく、顧客はオーダーを発行する際に、以下を含む*支払い情報* を指定しなければなりません。

- 支払いメソッド: 顧客がオーダーする際の支払いメソッド。ストアの Payment Manager で構成されている決済カセットに応じて、オフライン決済、CyberCash (オンライン決済)、 SET Secure Electronic Transaction™ および Merchant Initiated Authorization (MIA) (顧客がオンライン・ウォレットを使用する必要のないオンライン決済)、またはカスタム支払いメソッドを受諾するようにストアをセットアップできます。
- カードに関する情報 (クレジット・カード支払いの場合): 顧客がオーダー時の支払いに使用するクレジット・カードの社名、番号、および有効期限日付。通常クレジット・カード情報は、ストアでオンライン決済がサポートされている場合に必要になります。
- 購入オーダー番号: ストアでオーダーする際に顧客が提示する購入オーダー番号。購入オーダー番号は、ストアと顧客との間の契約に記された条件によって規定されているように、その顧客がそのストアからオーダーする権限のある顧客であることを認証します。

オーダー・アイテム

オーダー・アイテム とは、オーダーに含まれる個々の商品やアイテムのことです。1 つのオーダーには 1 つ以上のオーダー・アイテムがなければなりません。各オーダー・アイテムは、顧客が購入に選択したものを表します。さらに、各オーダー・アイテムは、取引条件 (通常は契約)、配送モード、配送センター、および価格オファーを参照します。割引、配送料および税総額は、各オーダー・アイテムとともに保管されなければなりません。

以下の図は、WebSphere Commerce のオーダー・アイテム資産を示しています。図の後に、個々の資産の説明があります。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、 29 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則についての詳細は、 311 ページの『付録 A. UML の凡例』を参照してください。

サブオーダー

オーダー・アイテムは、サブオーダー に分けられます。サブオーダーとは、特定の住所に配送されるオーダーの一部のことです。たとえば、顧客は、ショッピング・カート内の複数の商品に、それぞれ異なる配送先住所を指示することができます。個々の配送先住所と、それに関連付けられた商品が、サブオーダーを構成します。サブオーダー内のオーダー・アイテムの配送先住所は同じであり、それらのオーダー・アイテムの額の小計を表示するのに使用できます。

OrderItem オブジェクトの数量属性は、単位のない数値であり、 CatalogEntry オブジェクトに関連する CatalogEntryShippingInformation オブジェクトの名目数量属性を掛けて、その OrderItem によって表示される実際の量にすることができます。CatalogEntryShippingInformation オブジェクトは、数量の単位を指定します。

オーダーは通常、単一のストアに関連付けられますが、ストア、またはストア・グループのいずれかも関連付けることができる特別なオーダー・タイプが、オーダー・プロファイルです。オーダー・プロファイルは、オブジェクト・モデル内で、状況「Q」のオーダーとして表されます。オーダー・プロファイルは、顧客に関するデフォルトの情報 (支払い情報、配送先住所、配送モード、請求先住所など) を保持します。

その他のオーダー・アイテム資産

オーダー・アイテムを、以下のオブジェクトのうちの 1 つに関連付けることができます。あるいは、どれにも関連付けないことも可能です。

- そのオーダー・アイテムを含むオーダーを発行した顧客の配送先住所。ストアの配送センターで配送先住所を使用してオーダー・アイテムを適切に配送できるように、顧客はオーダー・プロセス中に配送先住所を指定しなければなりません。
- 顧客のオーダーに含める必要のあるオーダー・アイテムの配送や受け取りを行ったり、オーダー・アイテムの在庫を保管したりする配送センター。
- オーダー・アイテムの配送モード。これは、運送会社 (配送センターから顧客への配送サービスを提供する会社) とその運送会社が提供する配送サービスの組み合わせです。たとえば、ABC 運輸の翌日配送サービスや、 ABC 運輸の速達は、配送モードの一例です。
- オーダー・アイテムに関連付けられている価格のオファー。さまざまな価格リスト (または「取引位置コンテナー」) にさまざまなオファーを組み込むと、ストアで、同じ商品または SKU について、顧客によって異なる価格を提供できます。たとえば、旅行代理店で、大人料金、高齢者料金、子供料金、および学生料金の 4 種類の価格リストに飛行機のチケットのオファーを組み込むことができます。
- オーダー・アイテムのカタログ・エントリー。つまり、カタログからアイテムをオーダーすると個々のオーダー・アイテムになります。
- アイテムをオーダーする際の使用条件を定義した取引条件。これは通常は契約ですが、オーダーが処理用に送信されるまでの間は、見積依頼 (RFQ) の場合もあり、これはネゴシエーションを表します。

WebSphere Commerce Server のオーダー資産の構造に関する詳細は、WebSphere Commerce オンライン・ヘルプで『オーダー・オブジェクト・モデル』と『データ・モデル』を参照してください。

WebSphere Commerce でのオーダー資産の作成

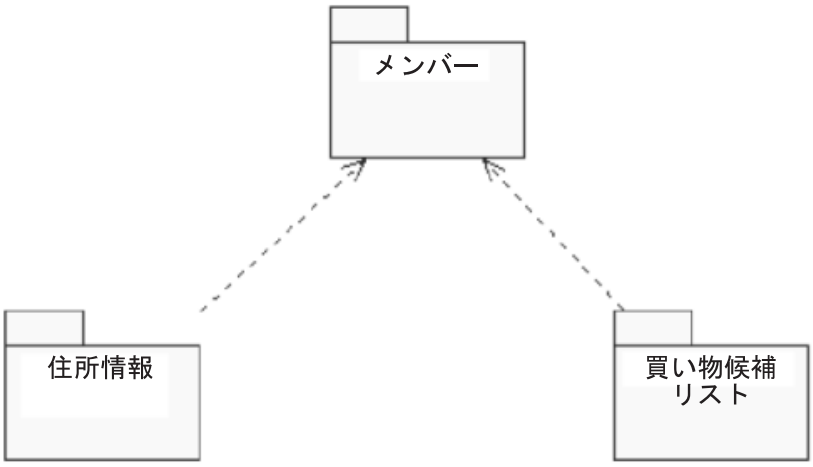
顧客はストアからオーダーを発行したり、ストアの顧客サービス担当者にこのタスクの完了を要求したり (WebSphere Commerce Accelerator を使用する) できます。B2C 顧客に代わってオーダーを作成するには、 WebSphere Commerce のオンライン・ヘルプのトピック『登録済み顧客のオーダーの作成』および『未登録顧客のオーダーの作成』を参照してください。 B2B 顧客に代わってオーダーを作成するには、ヘルプ・トピック『ビジネス・ユーザーのオーダーの作成』を参照してください。

第 23 章 顧客資産とセラー資産

ストアには複数の役割を組み込むことができますが、少なくとも顧客とセラーは必要です。

WebSphere Commerce の顧客資産について

顧客 とは、ストアで買い物し、必要に応じて買い物候補リストを作成し、オーダーを発行し、ストアやセラーから購入する人のことです。以下の図は、顧客がストアからオーダーを発行するのに必要な資産を示しています。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、 29 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則については、 311 ページの『付録 A. UML の凡例』を参照してください。

上の図で示されているように、WebSphere Commerce システムにはメンバーが含まれています。個々のユーザー・メンバーや組織エンティティー・メンバーに役割を割り当てることができます。

注: WebSphere Commerce では、メンバー は組織エンティティー、ユーザー、またはメンバー・グループのいずれかです。詳細は、 193 ページの『メンバー』を参照してください。

この場合、ユーザー・メンバーは顧客です。顧客は、住所情報を提示しなければならず、買い物候補アイテムのリストを作成できます。上記の図では、メンバー (顧客) と、そのメンバーに関連付けられた顧客資産との間の相互の関係が示されてい

ます。顧客は住所を所有して提示しなければならず、買い物候補リストを作成してストアで買い物できます。住所と買い物候補リストは顧客に従属しています。

住所情報

顧客は、ストアから購入する際に、連絡先住所、請求先住所、および配送先住所の 3 種類の住所情報を提示しなければなりません。これらの住所タイプについて以下に説明します。個々の住所は固有にすることもできますし、同じにすることもできます。

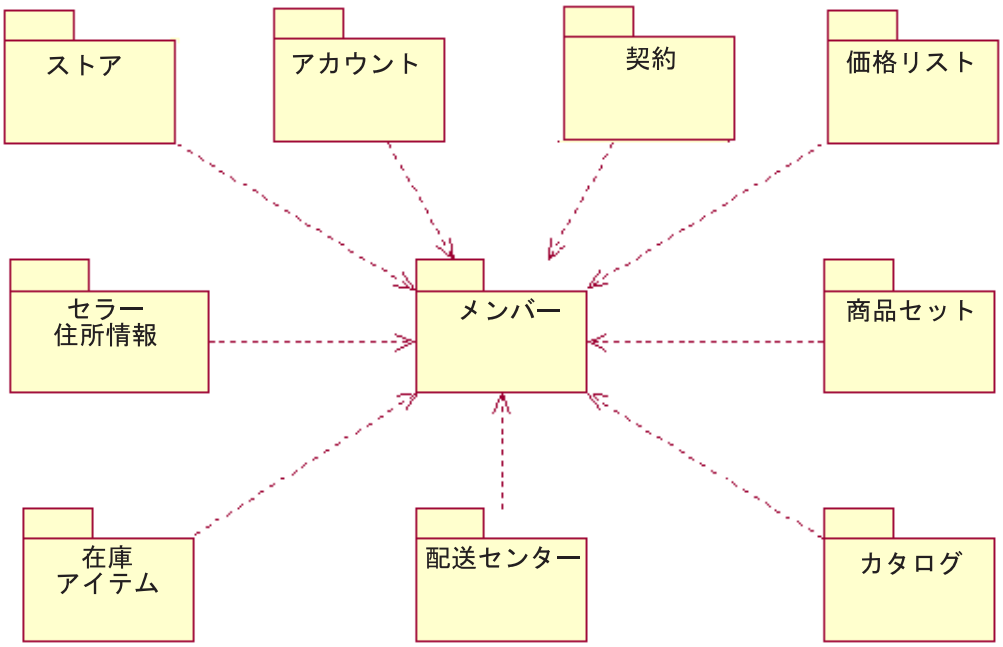
- 連絡先住所は、さまざまな目的で顧客に通知するのに使用されます。たとえば、オーダーの状況や変更内容、および近日中に行われるストアの行事 (販売促進活動やストアのメインテナンスなど) が含まれます。顧客の連絡先住所には、番地の名前と番号、市区町村、都道府県、郵便番号、国、 E メール・アドレス、電話番号、および FAX 番号が含まれます。通常、連絡先住所は、勤務先住所などの、最も容易に顧客と連絡が取れる場所にします。
- 請求先住所は、購入に対する勘定書または送り状を送るために使用されます。請求先住所には、番地の名前と番号、市区町村、都道府県、郵便番号、国、電話番号、および E メール・アドレスが含まれます。請求先住所は、連絡先住所や配送先住所と同じにすることも、違うものにすることも可能です。
- 配送先住所は、購入商品の配送に使用されます。配送先住所には、番地の名前と番号、市区町村、都道府県、郵便番号、国、電話番号、および E メール・アドレスが含まれます。配送先住所は、連絡先住所や請求先住所と同じでも違っていてもかまいません。

買い物候補リスト

ストアで*買い物候補リスト* をサポートできます。つまり、顧客が、買い物候補リストに、将来オーダーしたいと思う商品を追加します。買い物候補リストはショッピング・カートではありません。買い物候補リスト (ウィッシュ・リスト) には、複数のストアからのアイテムを含めることができますが、価格、配送先住所、配送モード、在庫納期情報、あるいは割引、配送料、税などの計算額は含まれません。

WebSphere Commerce のセラー資産について

セラー は、ストア・セールスの追跡に加え、全体的なストア目標と管理を監視します。セラーは、商品とサービスを顧客に販売します。セラーの役割はマーチャントと同じで、すべての WebSphere Commerce Accelerator 機能に対するアクセス権があります。以下の図は、セラーがストアを保守し、顧客に対する販売を行うのに必要な資産を示しています。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、 29 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則についての詳細は、 311 ページの『付録 A. UML の凡例』を参照してください。

上の図で示されているように、WebSphere Commerce システムにはメンバーが含まれています。各メンバーには、ストアの顧客サービス担当者やウェアハウスのストア受取人などの役割が割り当てられます。セラーの役割が顧客に対する販売を行うには、以下の資産を保守できます。

- ストア
- アカウント (オプション)
- 契約 (最低限 WebSphere Commerce のデフォルト契約)
- 商品セット
- 価格リスト
- カタログ
- 配送センター
- 在庫アイテム

上記の図では、メンバー (セラー) と、セラー資産との間の関係が示されています。つまり、セラーは上にリストされている資産を持ってストアを保守することができ、一方で資産はセラーによってデプロイされる必要があります。

ストア

WebSphere Commerce のオンライン・ストア は、 HTML および JavaServer Pages のファイルに加え、課税、配送、支払い、カタログ、およびその他のデータベース資産を持つ集合で構成されます。これらの集合はストア・アーカイブに入れられま

す。またストアには保管データもあります。このデータは WebSphere Commerce データベースに読み込まれる情報で、ストアはこのデータを使用して機能します。

WebSphere Commerce ストアの詳細については、 45 ページの『第 6 章 ストア資産』および 39 ページの『第 4 部 ストア・データの開発』を参照してください。

アカウント

ストアでビジネス・アカウント をセットアップし、顧客がこれらのアカウントを使用してストアから購入できるようにすることができます。アカウントには、以下の情報があります。

- アカウント名。多くの場合、これは顧客に関連する組織の名前です。この組織はストアと契約を交わしており、そのストアで買い物をする顧客との取引条件を明記しています。たとえば IBM という組織が、ABC オフィス・サプライ社と契約を結ぶ場合などです。
- 担当者名。これは、アカウントを担当するセラーの組織内の担当組織の名前です。
- アカウントに属する契約の数。

WebSphere Commerce アカウントの詳細については、 102 ページの『アカウント (ビジネス・アカウント)』および WebSphere Commerce のオンライン・ヘルプを参照してください。

契約

通常 WebSphere Commerce では、すべての顧客は*契約* の下で買い物しなければなりません。顧客とセラーとの間のアカウントごとに 1 つ以上の契約 (未登録の顧客か顧客がストアで買い物する場合、または顧客が他の契約の範囲外の商品を購入できるようにしたい場合は、少なくとも*デフォルト契約*) と関連付けなければなりません。契約により、顧客は、指定された価格で、指定された期間に渡り、契約に定められている使用条件とビジネス・ポリシーに基づいて、ストアから商品を購入できます。セラーが契約をデプロイして、顧客がストアから購買できるようにします。

セラーが使用できる WebSphere Commerce の契約とデフォルト契約の詳細については、 103 ページの『契約』を参照してください。

商品セット

商品セット には、セラーがオンライン・カタログを論理サブセットにカテゴリー化して、さまざまな顧客がさまざまなカタログ・ビューの利点を利用できるようにするための機構が備えられています。さらに、セラーは顧客用の契約を作成し、顧客が事前定義済みの商品セットの下の商品しか購入できないことを決定できます。

WebSphere Commerce の商品セットの詳細については、 62 ページの『商品セット』を参照してください。

価格リスト

価格リスト は、セラーが顧客にオファーしたり提示したりする価格と関連付けられます。セラーは、同じ商品について、顧客によって異なる価格をリストできます。

WebSphere Commerce では、価格のオファーは*取引位置* とも呼ばれ、カタログ・エントリーの価格と、その価格での購入資格を得るために顧客が満たす必要のある基準を表します。

WebSphere Commerce では、Offer オブジェクトは、メンバーが所有する、TradingPositionContainer の一部です。 TradingPositionContainer には、TradingPosition が含まれており、それらをすべての顧客に、あるいはストアによって顧客グループとして認識された特定のグループのショッパーにのみ利用可能にすることができます。 TradingPositionContainer が価格リストと呼ばれることもあります。価格リストには 2 種類あります。 1 つは標準価格リストで、ストア・カタログ内の商品の基本価格が含まれます。もう 1 つはカスタム価格リストで、商品とそれらのカスタマイズ済みの価格のリストを指定します。

WebSphere Commerce の価格リストの詳細については、 93 ページの『第 9 章 価格設定資産』を参照してください。

カタログ

WebSphere Commerce のストアでは、 1 つ以上のオンライン・*カタログ* を使用して、セラーが販売をオファーする商品やサービスを陳列します。通常、販売するアイテムのオンライン・カタログには価格、イメージ、および説明が含まれます。オンライン・カタログではまた、容易にナビゲーションできるよう、商品が明確なカテゴリーに分けて表示されている場合があります。

WebSphere Commerce システムのストアごとに、*マスター・カタログ* がなければなりません。マスター・カタログは、カタログの管理に使用されます。マスター・カタログを中心として、セラーの商品取引が管理されます。これは、すべての商品、アイテム、関係、およびストアで販売されるものすべての標準価格を含む 1 つのカタログです。セラーに複数のストアがある場合は、それらのストア間でマスター・カタログを共用できます。

WebSphere Commerce の商品セットの詳細については、 59 ページの『第 8 章 カタログ資産』を参照してください。

配送センター

配送センター は、ストアにより、在庫保管庫、および配送受取センターの両方として使用されます。 1 人のセラーには、1 つ以上の配送センターがあります。

WebSphere Commerce サーバーの側から見ると、 FulfillmentCenter オブジェクトは、Store オブジェクトから独立したものです。それは商品の在庫と配送を管理します。オーダーの発送において、配送センターは顧客が指定する ShippingMode オブジェクトに従います。 ShippingMode オブジェクトは、オーダーを実行するために、運送会社と配送方法を指示します。配送センター内で、ShippingArrangement オブジェクトは、ある特定の ShippingMode を使用して商品を配送するように、 Store オブジェクトが FulfillmentCenter オブジェクトと手配済みであることを示します。

WebSphere Commerce 配送センターの詳細については、 113 ページの『第 11 章 フルフィルメント資産』および 139 ページの『第 18 章 配送資産』を参照してください。

在庫アイテム

在庫アイテム には、物理的に報告することのできる配送センター内にあるすべてのものが含まれます。 WebSphere Commerce システムでは満たすべき特定の在庫タイプ (たとえば、アイテム、商品、SKU、バンドル、パッケージなど) が定義されていますが、これらはすべて在庫と見なされます。 WebSphere Commerce Accelerator の商品管理ツールを使用して、商品が配送されるよう構成されます。

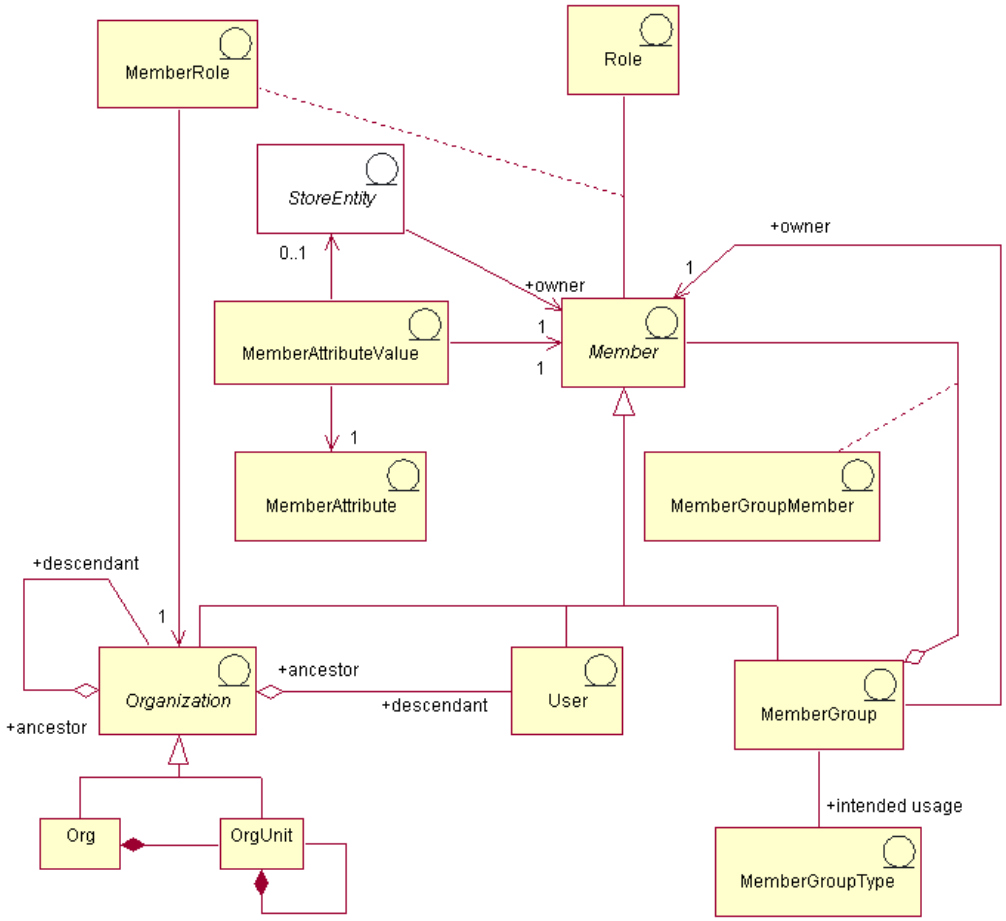
WebSphere Commerce 在庫アイテムの詳細については、 WebSphere Commerce オンライン・ヘルプおよび 177 ページの『第 21 章 在庫資産』を参照してください。

WebSphere Commerce のメンバー資産について

WebSphere Commerce メンバー資産には、WebSphere Commerce システムの参加者に関するデータが組み込まれます。メンバーになれるのは、ユーザー、ユーザーのグループ、または組織エンティティーです。管理者 (サイト管理者など) は、ユーザーや組織エンティティーのメンバーに役割を割り当てます。メンバーに役割が割り当てられると、アクセス・コントロール・コンポーネントは、メンバーがアクティビティーに参加するのを許可します。たとえば、組織にはバイヤーまたはセラーのいずれか、あるいは両方の役割を割り当てることが可能です。ユーザーへは複数の役割を割り当てることができます。管理者は、様々なビジネス目的に応じて分類されるユーザーのグループを作成することができます。 WebSphere Commerce の管理コンソールを使用して、組織、ユーザー、役割、およびメンバーのグループを作成して処理してください。

メンバー資産に関するビジネス・ロジックは、メンバー登録およびプロファイル管理サービスを提供します。メンバー資産と密接に関係する別のサービスには、アクセス・コントロール、認証、およびセッション管理が含まれます。これらのトピックに関する詳細情報は、 WebSphere Commerce のオンライン・ヘルプを参照してください。

以下の図は、WebSphere Commerce のメンバー資産を示しています。図の後に、個々の資産の説明があります。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、 29 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則については、 311 ページの『付録 A. UML の凡例』を参照してください。

メンバー

WebSphere Commerce のメンバー は、以下のいずれかになります。

- *組織エンティティー*。組織エンティティーとは、「IBM」などの組織や「電子商取引部門」などの組織内の組織単位のことです。
- ユーザー (登録済みであることも、登録されていないこともあります)。登録済みユーザーには、固有 ID とパスワードがあり、登録上の目的により、プロファイル・データの入力が必要です。登録済みユーザーは、そのプロファイル・タイプに従って分類されます。タイプ「B」はビジネス・ユーザー (または B2B 顧客) を、タイプ「C」は小売ユーザー (または B2C 顧客) を指します。登録済みユーザーと登録されていないユーザーの詳細については、 WebSphere Commerce オンライン・ヘルプの『メンバー』を参照してください。

- メンバー・グループ。これは、さまざまなビジネス上の理由でカテゴリー化されたユーザーのグループです。グループ分けはアクセス・コントロールの目的、承認の目的、およびマーケティングの目的 (割引や価格の計算や商品の表示など) で使用することができます。

メンバーは、個々のストア・エンティティー (つまり、ストアまたはストア・グループ) を所有します。

メンバー属性

WebSphere Commerce のメンバーには*属性* の集合があり、個々の属性には関連した*値* があります。メンバーの基本ユーザー・プロファイルは、登録情報、個人情報、住所情報、購入ヒストリー、およびその他の各種属性をまとめたものです。

ビジネス・ユーザー・プロファイルには、基本ユーザー・プロファイルに加え、従業員番号、仕事の肩書き、仕事の内容などのような雇用情報が含まれます。登録の際に、ビジネス・ユーザーは、所属する企業組織を識別する必要があります。組織エンティティーのプロファイルは、組織名や業種などのような追加情報を含みます。

アクセス・コントロール・ルールにより、プロファイル管理を実行するためのユーザー権限が決まります。メンバー・プロファイルには、様々な個別属性およびビジネス関連属性 (たとえば、支払い情報、住所、優先言語および優先通貨、パーベイシブ・コンピューティング・デバイス) を入れることができます。属性をストアに依存させることができます。これらの属性は、メンバー・グループではサポートされません。

役割

各ユーザー・メンバーは、組織で 1 つ以上の役割を果たすことができます。サイト管理者は、各ユーザー・メンバーに役割を割り当てます。たとえば、IBM 社のメンバーであるジョン・スミス氏の役割は顧客サービス担当者で、ジョン氏は IBM 社の顧客にかかわる仕事をし、顧客の登録情報、オーダー、または返品に関する問い合わせや関心事に答えます。またジョン氏には顧客サービス・スーパーバイザーの役割もあり、前述の仕事をすべて担当することに加えて、他の顧客サービス担当者全員の承認と監視の許可権限もあります。

WebSphere Commerce システムには、以下のデフォルトの役割タイプの設定があります。

- サイトおよびコンテンツ開発役割
- 技術操作役割
- マーケティング管理役割
- 商品管理役割
- ビジネス関係管理役割
- 物流管理およびオペレーション管理役割
- 組織管理役割

上記の個々の役割に関する詳細情報は、WebSphere Commerce のオンライン・ヘルプのトピック『役割』を参照してください。サイト管理者は、組織エンティティーでサイト管理者が作成する新しい役割のほかに、上記の役割も割り当てることができます。つまり、組織エンティティーに属しているユーザーが、その組織エンティティーに割り当てられている役割を果たすことができます。

ユーザーに役割が割り当てられると、組織エンティティーに役割の有効範囲が設定されます。ユーザーに役割が割り当てられている場合、そのユーザーが、必ずしも自分の属している組織エンティティーの役割を実行するとは限りません。つまり、管理者はその割り当てを実行する際に、ユーザーが役割を実行する組織エンティティーを選択できます。管理者がルート組織を選択した場合、ユーザーはすべての組織エンティティーに関するその役割を果たします。

WebSphere Commerce のメンバー資産の構造に関する詳細は、 WebSphere Commerce オンライン・ヘルプで『メンバー・オブジェクト・モデル』と『データ・モデル』を参照してください。

WebSphere Commerce でのメンバー資産の作成

セラー (ストア所有者として活動する組織) を作成して、そのセラーに関する情報を保守するには、管理コンソールを使用してください。詳しくは、WebSphere Commerce オンライン・ヘルプのトピック『組織の作成』を参照してください。

ストア開発者は顧客を作成しません。顧客がストアに登録する際に、WebSphere Commerce システムに登録情報が徴収されて保守されます。

第 5 部 ストアへのアクセス・コントロールの追加

第 24 章 ストアでのアクセス・コントロール

WebSphere Commerce では、顧客であれ管理者であれ、特定のユーザーが実行できるタスクをアクセス・コントロールによって決定できます。この章では、ストアにアクセス・コントロールを追加する方法、たとえば、顧客に表示するページの制限やストア内で実行できるタスクの制限について取り上げます。

WebSphere Commerce でのアクセス・コントロール・モデルや、サイト・レベルでのアクセス・コントロールの適用については、 *IBM WebSphere Commerce アクセス・コントロール・ガイド* を参照してください。カスタマイズしたコードにアクセス・コントロールをインプリメントすることの詳細は、 *IBM WebSphere Commerce プログラマーズ・ガイド* を参照してください。これらのガイドは、以下の URL で利用できます。

Business

`http://www.ibm.com/software/webservers/commerce/wc_be/lit-tech-general.html`

Professional

`http://www.ibm.com/software/webservers/commerce/wc_pe/lit-tech-general.html`

WebSphere Commerce のアクセス・コントロールについて

WebSphere Commerce のアクセス・コントロール・モデルについては、 *IBM WebSphere Commerce アクセス・コントロール・ガイド* で詳しく説明されています。しかし、本書でも、アクセス・コントロールがストア開発にどのように影響するかについて、簡単に要約されています。

WebSphere Commerce のアクセス・コントロールは、以下の要素で構成されています。すなわち、ユーザー、アクション、リソース、および関係です。

- ユーザーは、システムを使用する人のことです。アクセス・コントロールを行うには、ユーザーを関係のあるアクセス・グループに分類しなければなりません。たとえばストアでは、ユーザーを以下のアクセス・グループにグループ化できます。すなわち、登録済み顧客、ゲスト顧客、顧客サービス担当者のような管理者グループです。
- アクションは、ユーザーがリソースに対して実行できる活動です。アクセス・コントロールのためには、アクションも関係のあるアクション・グループに分類しなければなりません。たとえば、ストアで使用される一般的なアクションの 1 つに表示があります。表示は、ストア・ページを顧客に表示するときに行われます。ストアで使用する表示は、特定のアクション・グループに割り当てなければなりません。
- リソースは、保護されているエンティティーです。たとえば、アクションが表示の場合、そのアクションのリソースは、クラス `com.ibm.commerce.command.ViewCommand` になります。アクセス・コントロールのために、リソースはリソース・グループにグループ化されます。

- 関係は、ユーザーとリソースの間の関係です。アクセス・コントロール・ポリシーでは、ユーザーとリソースの間の関係を満たさなければならない場合があります。

アクセス・コントロール・ポリシー

アクセス・コントロール・ポリシーは、アクセス・グループのユーザーがリソースに関する特定の関係を満たしている限り、そのアクセス・グループが WebSphere Commerce のリソースに対して特定のアクションを実行することを許可します。

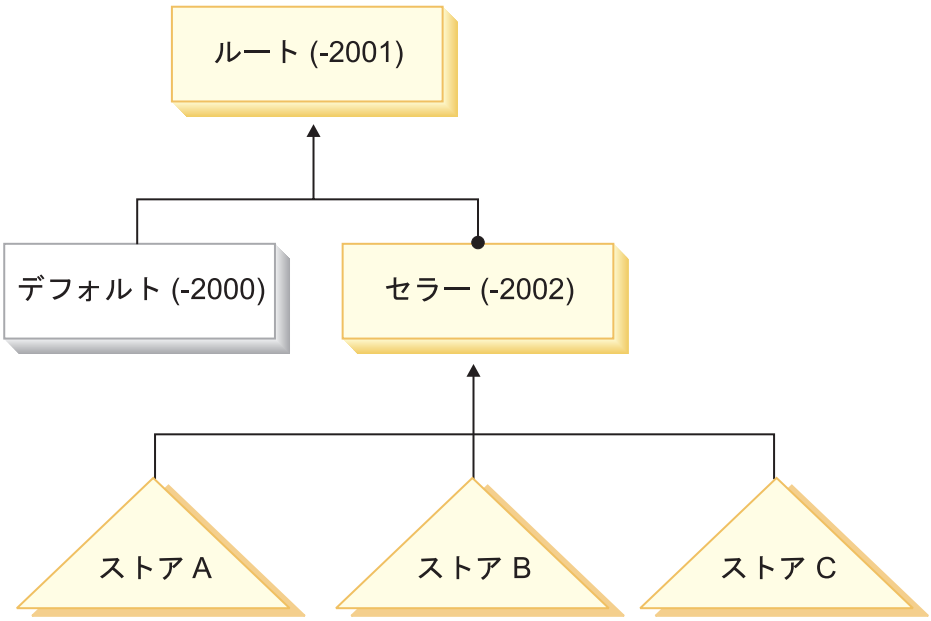
WebSphere Commerce には、インスタンス作成時にロードされる、 200 以上のデフォルト・アクセス・コントロール・ポリシーが用意されています。これらのポリシーは、オーダーの作成と処理や、 Business 見積依頼と Business 契約などの商取引を含む、広い範囲の一般的なビジネス活動を網羅しています。デフォルト・ポリシーについては、 *IBM WebSphere Commerce アクセス・コントロール・ガイド* で説明されています。

各アクセス・コントロール・ポリシーは、特定の組織エンティティーによって所有されます。アクセス・コントロール・ポリシーは、アクセス・コントロール・ポリシー所有者に所有されたリソースにだけ適用できます。

注: アクセス・コントロールの面から言うと、リソースの所有権には特別な意味があります。すべてのリソースは、`com.ibm.commerce.security.Protectable` インターフェースをインプリメントする必要があります。このインターフェースのメソッドの 1 つは `getOwner()` で、これはリソースの所有者のメンバー ID を戻します。たとえば、Order エンティティー Bean は、リモート・インターフェースが Protectable インターフェースを拡張することで保護されるリソースです。 Order で `getOwner()` をインプリメントすると、それは特定の Order リソースが、 オーダーの発生したストアの所有者を戻すようなものとなります。リソースがコマンドであるポリシー (たとえば `com.ibm.commerce.command.ViewCommand`) なら、`getOwner()` のデフォルトのインプリメントは、現在コマンドのコンテキストにあるストアの所有者を戻すことになります。コマンド・コンテキストにストアがなければ、ルートの組織が所有者として使用されます。詳しくは、*IBM WebSphere Commerce プログラマーズ・ガイド* を参照してください。

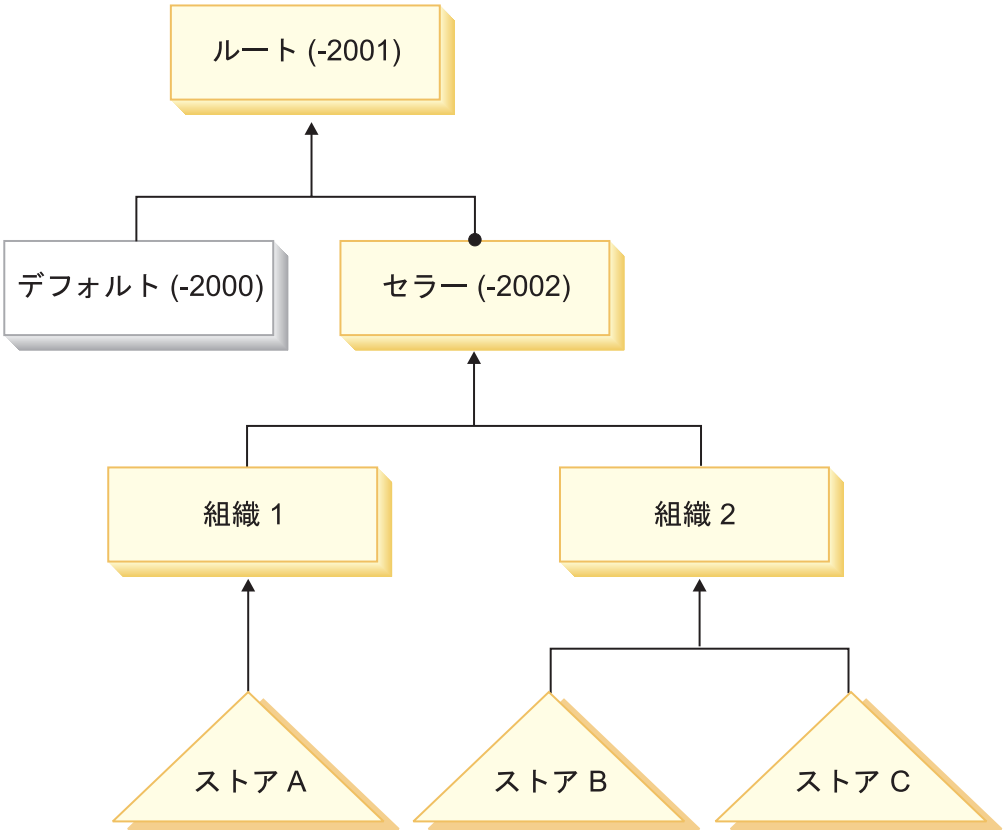
以下の図を考えてください。

Professional



WebSphere Commerce Professional Edition では、ルートの組織は最上位レベルの組織です。これは、他のすべての組織エンティティーを所有します。その結果、ルートの組織によって所有されるアクセス・コントロール・ポリシーは、サイト内のすべてのリソースに適用されます。アクセス・コントロール・ポリシーがセラー組織によって所有される場合には、そのアクセス・コントロール・ポリシーは、セラーによって所有されるリソースだけに適用されます。

Business



WebSphere Commerce Business Edition では、ルートの組織は最上位レベルの組織です。これは、他のすべての組織エンティティーを所有します。その結果、ルートの組織によって所有されるアクセス・コントロール・ポリシーは、サイト内のすべてのリソースに適用されます。アクセス・コントロール・ポリシーがセラー組織によって所有される場合、そのアクセス・コントロール・ポリシーは、セラーによって所有されるリソースにのみ適用されます。すなわち、ストア A、B、および C を含む組織 1 と 2 に所有されているリソースに適用されます。アクセス・コントロール・ポリシーが組織 1 によって所有される場合、ストア A を含むリソースにのみ適用されます。アクセス・コントロール・ポリシーが組織 2 によって所有される場合、ストア B および C のリソースに適用されます。

Business アクセス・コントロール・ポリシーは組織エンティティーによって所有されるため、サイトで複数のストアを作成していて、個々のストアに異なるアクセス・コントロール・ポリシーを適用する予定なのであれば、それぞれのストアに対して個別の組織を作成する必要があります。

注: WebSphere Commerce で提供されるほとんどのデフォルト・アクセス・コントロール・ポリシーは、ルートの組織によって所有され、サイト内のすべてのリソースに適用されます。ルートの組織によって所有されるアクセス・コントロール・ポリシーは、サイト・レベル・ポリシーとして言及されます。他の組織エンティティーによって所有されるポリシーは、組織レベル・ポリシーと呼ばれます。

ストアでのアクセス・コントロール

WebSphere Commerce で作成されるストアはすべて、ルートの組織が所有するデフォルトのアクセス・コントロール・ポリシーに従います。ストアのリソースも、そのストアを所有する組織に所有されるすべてのアクセス・コントロール・ポリシーと、その組織の下部組織に所有されるすべてのアクセス・コントロール・ポリシーに従います。

デフォルトでは、ルートの組織がほとんどのアクセス・コントロール・ポリシーを所有しています。しかし、WebSphere Commerce に付属するいずれかのサンプル・ストアをベースにしてストアを作成する場合、そのストアを所有する組織によって所有される新しいアクセス・コントロール・ポリシーを作成します。サンプル・ストアでのアクセス・コントロール・ポリシーの詳細は、 『サンプル・ストアでのアクセス・コントロール』 を参照してください。

独自のストアを作成する場合には、サンプルをベースとしているかどうかに関係なく、新しいアクセス・コントロール・ポリシーを作成したり、既存のポリシーを変更することができます。これは、該当組織によって所有されるストアにのみ適用されます。たとえば、ストア・ページを表示するときの新しいビューを作成する場合、アクセス・コントロール・ポリシーをそれらのビューに割り当てる必要があります。

組織レベルでのアクセス・コントロール・データは、高水準のアクセス・コントロール・ポリシー・ファイルで定義されます。これらのファイルでは、任意のポリシーで使用できるアクション、アクション・グループ、リソース、リソース・グループ、および関係を定義しています。さらに、特定の組織に固有なポリシーも定義しています。 WebSphere Commerce で提供されているサンプル・ストアには、このような高水準アクセス・コントロール・ポリシー・ファイルが含まれています。次の節では、サンプル・ストアがこのようなアクセス・コントロール・ポリシー・ファイルを使用して組織情報を定義する仕組みを説明します。

サンプル・ストアでのアクセス・コントロール

すべてのストアが、高水準アクセス・コントロール・ポリシー・ファイルを持っています。これらのファイルは、それらのストア向けに特別に作成されたアクセス・コントロール・ポリシーを定義します。これらのポリシーは、ストアを所有する組織に所有されます。 NewFashion および Business ToolTech サンプル・ストア用に定義されたアクセス・コントロール・ポリシーについて、これから説明します。

Business ToolTech 用に作成されたアクセス・コントロール・ポリシーは、以下のとおりです。

- AllUsersForToolTechExecuteToolTechAllUsersViews
- RegisteredApprovedUsersForToolTechExecuteToolTech RegisteredApprovedUsersViews

NewFashion 用に作成されたアクセス・コントロール・ポリシーは、次のとおりです。

- AllUsersExecuteNewFashionAllUsersViews

これらのアクセス・コントロール・ポリシーにより、サンプル・ストア、またはサンプルをベースにしたストアで、どのユーザーにどのビューを表示するかが決まります。

これらのポリシーのアクセス・コントロール情報は、*samplestorename*AccessPolicies.xml と *samplestorename*AccessPolicies_*locale*.xml という、 2 つの高水準アクセス・コントロール・ポリシー・ファイルで定義されます。これらのファイルは、サンプル・ストアで使用される可能なアクション、アクション・グループ、リソース、およびポリシー定義を定義します。これらのファイルは、以下に示すディレクトリーにあります。

- NT drive:¥WebSphere¥CommerceServer¥samples¥stores¥*samplestorename*
- 2000 drive:¥ProgramFiles¥WebSphere¥CommerceServer¥samples ¥stores¥*samplestorename*
- AIX /usr/WebSphere/CommerceServer/samples/stores/*samplestorename*
- Solaris /opt/WebSphere/CommerceServer/samples/stores/*samplestorename*
- Linux /opt/WebSphere/CommerceServer/samples/stores/*samplestorename*
- 400 /QIBM/ProdData/WebCommerce/samples/stores/*samplestorename*

  ここで *samplestorename* は、ストアのベースとして使用したサンプル・ストア・アーカイブの名前 (たとえば、NewFashion) です。

サンプル・ストアをストア・アーカイブとしてパッケージする前に、これら 2 つの高水準アクセス・コントロール・ポリシー・ファイルが変換されます。これにより、ローダー・パッケージでの使用に適した 2 つの XML ファイルが生成されます。続いて、これらの変換された XML ファイルがストア・アーカイブにパッケージされ、残りのストア・アーカイブと共に発行されます。

サンプル・ストアのアクセス・コントロール・ポリシー・ファイルについて: ストア・レベルでアクセス・コントロールを追加する仕組みを理解するために、高水準サンプル・ストアのアクセス・コントロール・ポリシー・ファイルに精通してください。以下の例は、Business ToolTechAccessPolicies.xml ファイルから取られたものです。

アクションの定義: Business ToolTechAccessPolicies.xml ファイルの最初のセクションでは、ストアでの新しいアクションを定義します。これらは、既存のアクセス・コントロール・ポリシーでは網羅されていません。このケースでは、アクションはストアで使用されるすべてのビューになります。 URL から直接に呼び出せるビューまたは (ビューに転送することで立ち上げることとは対照的に) 別のコマンドからのリダイレクトで立ち上げられるビューを使用してストアにページを表示するには、ビューをアクションとして定義しなければなりません。次の例を考えてください。

```
<!-- [Start of Action definitions] -->
<!-- [this is the dictionary of possible actions -->
<Action Name="GenericApplicationError"
CommandName="GenericApplicationError">
</Action>

<Action Name="GenericSystemError"
```

```
CommandName="GenericSystemError">
</Action>

<Action Name="OrderOptionsView"
CommandName="OrderOptionsView">
</Action>

 <!--[End of Action definitions] -->
```

ここで、

- Action Name は、XML ファイルでは、このアクションを指すときに使用されるラベルです。これらの例では、ラベルはビュー名と同じです。
- CommandName は、VIEWREG テーブルの VIEWNAME 列に保管されたビューの名前です。 CommandName は、ACACTION テーブルの Action 列に保管されます。

注: アクションがすでにデフォルト・ポリシーか WebSphere Commerce 内の別のポリシーで定義されている場合、アクションを使用するポリシーごとにアクションを再定義する必要はありません。

リソース・カテゴリーの定義: ファイルの 2 番目のセクションでは、リソース・カテゴリーを定義します。リソース・カテゴリーとは、リソースのクラスを指します。 ToolTech 用に識別されたリソース・カテゴリーは、クラス com.ibm.commerce.command.ViewCommand および com.ibm.commerce.tools.command.ToolsForwardViewCommand です。

```
<!-- [Start of Resource Category definitions] -->
<!-- the dictionary of Protectable resources -->
<ResourceCategory Name="com.ibm.commerce.command.ViewCommandResourceCategory"
ResourceBeanClass="com.ibm.commerce.command.ViewCommand">
</ResourceCategory>
<ResourceCategory Name="com.ibm.commerce.tools.command.
ToolsForwardViewCommandResourceCategory"
ResourceBeanClass="com.ibm.commerce.tools.command.ToolsForwardViewCommand">
</ResourceCategory>
<!-- [End of Resource Category definitions] -->
```

ここで、

- ResourceCategory Name は、XML ファイルでは、このリソース・カテゴリーを指すときに使用するラベルです。
- ResourceBeanClass はクラスの名前です。

注: リソース・カテゴリーがすでにデフォルト・ポリシーか WebSphere Commerce 内の別のポリシーで定義されている場合、リソース・カテゴリーを使用するポリシーごとにリソース・カテゴリーを再定義する必要はありません。上記の例で定義されているリソース・カテゴリーは、すでにデフォルト・ポリシーで定義されていますが、説明の目的で、ここでもう一度定義されます。

アクション・グループの定義: 3 番目のセクションでは、アクション・グループを定義します。このアクション・グループは、ファイルの最初のセクションで定義したアクションをグループ化したものです。 ToolTech の例では、新しいユーザー・ビューはすべて、グループ ToolTechAllUserViews (これは、すべてのユーザーがそれらのビューにアクセスできるようにするポリシーで使用される) にグループ化さ

れるか、グループ ToolTechRegisteredApprovedUserviews (これは、登録ユーザーだけがそれらのビューにアクセスできるようにするポリシーで使用される) にグループ化されます。

注: また、WebSphere Commerce 内のどこかで定義されるアクションを、使用しているアクション・グループに追加することも可能です。これらのアクションが WebSphere Commerce のどこかで定義されていれば、 204 ページの『アクションの定義』 で説明した Action リストで定義する必要はありません。

```
<!-- [Start of Action Group definitions] -->
<!-- Dictionary of grouped actions usable in policies -->
<!-- cross-component view-related action groups -->
<ActionGroup Name="ToolTechAllUsersViews"
OwnerID="RootOrganization">

<ActionGroupAction Name="UserRegistrationForm"/>
<ActionGroupAction Name="UserRegistrationErrorView"/>
<ActionGroupAction Name="GenericApplicationError"/>
<ActionGroupAction Name="GenericSystemError"/>
<ActionGroupAction Name="LogonForm"/>
</ActionGroup>

<!-- [End of Action Group definitions] -->
```

ここで、

- ActionGroup Name は、アクション・グループの名前です。
- OwnerID は、アクション・グループの所有者です。ルートの組織は、アクション・グループの所有者でなければなりません。
- ActionGroupAction Name は、このグループに属するアクションの名前です。ActionGroupAction Name は、 204 ページの『アクションの定義』 の Action Name エレメントで定義した名前と一致していなければなりません。

リソース・グループの定義: 次のセクションでは、リソース・グループを定義します。リソース・グループとは、関連した一群のリソースを指します。

```
<!-- [Start of Resource Group definitions] -->
<!-- Dictionary of grouped resources usable in policies -->

<!-- Grouped resources permitting view execution,
 by any command extending com.ibm.commerce.command.ViewCommand
(with or without the Tools Framework) -->
<ResourceGroup Name="ViewCommandResourceGroup"  OwnerID="RootOrganization">

<ResourceGroupResource Name="com.ibm.commerce.command.ViewCommandResourceCategory"/>
</ResourceGroup>

</ResourceGroup> !-- [End of Resource Group definitions] -->
```

ここで、

- ResourceGroup Name は、リソース・グループの名前です。
- OwnerID は、リソース・グループの所有者ですルートの組織は、リソース・グループを所有していなければなりません。
- ResourceGroupResource Name は、グループに含まれるリソース・カテゴリーの名前です。

ポリシーの定義: 最後のセクションでは、ストアで使用する新しいポリシーを定義します。

```
!-- [Start of Policy definitions] -->
!-- AllUsers for ToolTech can execute ToolTechAllUsersViews -->


<Policy Name="AllUsersForToolTechExecuteToolTechAllUsersViews"
OwnerID="MEMBER_ID"
UserGroup="AllUsers"
UserGroupOwner="RootOrganization"
ActionGroupName="ToolTechAllUsersViews"
ResourceGroupName="ViewCommandResourceGroup">
    </Policy>
!-- RegisteredApprovedUsers for ToolTech can execute
ToolTechRegisteredApprovedUsersViews -->

<Policy Name="RegisteredApprovedUsersFor
ToolTechExecuteToolTechRegisteredApprovedUsersViews"
OwnerID="MEMBER_ID"
UserGroup="RegisteredApprovedUsers"
UserGroupOwner="RootOrganization"
ActionGroupName="ToolTechRegisteredApprovedUsersViews"
ResourceGroupName="ViewCommandResourceGroup">
    </Policy>

!-- [End of of Policy definitions] -->
```

ここで、

- Policy Name は、定義されるポリシーの名前です。
- OwnerId は、ポリシーの所有者です。このケースでは、ポリシーの所有者は、ストアを所有する組織です。
- UserGroup は、ポリシーを適用する対象のユーザーのグループ (アクセス・グループ) です。
- UserGroupOwner は、アクセス・グループの所有者です。この例では、アクセス・グループの所有者は、ポリシーの所有者とは異なります。ポリシーの所有者と UserGroupOwner が同じであれば、このエレメントは省略できます。
- ActionGroupName は、ポリシーを適用する対象のアクションのグループです。
- ResourceGroupName は、ポリシーを適用する対象のリソースのグループです。

ストアへのアクセス・コントロールの追加

ストア開発の観点からすると、必要なアクセス・コントロールのうち最も一般的なのは、ストアのために作成する新しいビューとコマンドのアクセス・コントロールです。しかし、他のタイプのアクセス・コントロールをストアに追加することができます。ビュー、コマンド、および他の機能用のアクセス・コントロールについての詳細は、 *IBM WebSphere Commerce アクセス・コントロール・ガイド* を参照してください。本書で概説されている次のステップに進む前に、必ず *IBM WebSphere Commerce アクセス・コントロール・ガイド* を確認してください。

新しいアクセス・コントロール機能を、サンプル・ストアをベースにしたストアに追加する場合、既存の高水準アクセス・コントロール・ポリシー XML ファイルを編集します。詳細な指示は、211 ページの『ストア・アーカイブのアクセス・コントロール・ファイルの編集』 を参照してください。アクセス・コントロールを、サンプル・ストアをベースにしていないストアに追加する場合、新しい高水準アクセ

ス・コントロール・ポリシー XML ファイルを作成してから変換する必要があります。詳細な指示は、『ストアでのアクセス・コントロールの作成』 を参照してください。

ストアでのアクセス・コントロールの作成

アクセス・コントロール資産はストアの他の資産とは異なり、2 つの高水準アクセス・コントロール XML ファイルを作成してから、それらを変換します。結果として生成される XML ファイルは、ローダー・パッケージを使用してデータベースにロードするか、ストア・アーカイブ内で使用することができます。

アクセス・コントロール資産を作成するには、以下のようにします。

1. サンプル・ストアのストア資産を作成するために使用される高水準 XML ファイル (*samplestorename*AccessPolicies.xml、および *samplestorename*AccessPolicies_*locale*.xml) を確認します。これらのファイルは、以下に示すディレクトリーにあります。
   - NT `drive:¥WebSphere¥CommerceServer¥samples¥stores¥samplestorename`
   - 2000 `drive:¥ProgramFiles¥WebSphere¥CommerceServer¥samples¥stores¥samplestorename`
   - AIX `/usr/WebSphere/CommerceServer/samples/stores/samplestorename`
   - Solaris `/opt/WebSphere/CommerceServer/samples/stores/samplestorename`
   - Linux `/opt/WebSphere/CommerceServer/samples/stores/samplestorename`
   - 400 `/QIBM/ProdData/WebCommerce/samples/stores/samplestorename`

   ここで *samplestorename* は、ストアのベースとして使用したサンプル・ストア・アーカイブの名前 (たとえば、NewFashion) です。
2. 313 ページの『付録 B. データの作成』の情報を確認します。
3. *samplestorename*AccessPolicies.xml ファイルのいずれかをコピーするか、新しいファイルを作成することにより、*storename*AccessPolicies.xml ファイルを作成します。詳しくは、*samplestorename*AccessPolicies.xml に対応する DTD ファイルを参照してください。 DTD ファイルは以下のディレクトリーにあります。
   - NT `drive:¥WebSphere¥CommerceServer¥xml¥policies¥dtd`
   - 2000 `drive:¥Program Files¥WebSphere¥CommerceServer¥xml¥policies¥dtd`
   - AIX `/usr/WebSphere/CommerceServer/xml/policies/dtd`
   - Solaris `/opt/WebSphere/CommerceServer/xml/policies/dtd`
   - Linux `/opt/WebSphere/CommerceServer/xml/policies/dtd`
   - 400 `/QIBM/ProdData/WebCommerce/xml/policies/dtd`
4. ストアがサポートしているロケールごとに、個別の高水準 XML ファイルを作成します。説明情報および表示名情報はすべてロケール固有のファイルに指定されるので、これを翻訳するのは容易です。

*samplestorename*AccessPolicies_locale.xml ファイルのいずれかをコピーするか、新しいファイルを作成することにより、ストアの言語ごとに *storename*AccessPolicies_*locale*.xml ファイルを作成する。

注: ストアがサポートするのが 1 つの言語だけの場合にも、ロケール固有のファイルを作成する必要があります。

5. 適切なアクセス・コントロール情報をファイルに追加します。詳細は、203 ページの『サンプル・ストアでのアクセス・コントロール』 および *IBM WebSphere Commerce アクセス・コントロール・ガイド* を参照してください。
6. *storename*AccessPolicies.xml および *storename*AccessPolicies_locale.xml を、次のディレクトリーにコピーします。
   - NT drive:¥WebSphere¥CommerceServer¥xml¥policies¥xml
   - 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥xml¥policies¥xml
   - AIX /usr/WebSphere/CommerceServer/xml/policies/xml
   - Solaris /opt/WebSphere/CommerceServer/xml/policies/xml
   - Linux /opt/WebSphere/CommerceServer/xml/policies/xml
   - 400 任意のユーザー data ディレクトリーにコピーします。既存の XML ファイルで、DTD への絶対パスを指定します。アクセス・コントロール DTD ファイルは、/QIBM/ProdData/WebCommerce/xml/policies/dtd ディレクトリーにあります。
7. xmltransform コマンドを実行して、*storename*AccessPolicies.xml を変換します。
   a. コマンド・プロンプトで、ディレクトリーを次のように変更します。
      - NT drive:¥WebSphere¥CommerceServer¥bin
      - 2000 drive:¥ProgramFiles¥WebSphere¥CommerceServer¥bin
      - AIX /usr/WebSphere/CommerceServer/bin
      - Solaris /opt/WebSphere/CommerceServer/bin
      - Linux /opt/WebSphere/CommerceServer/bin
   b. 次のように入力します。 xmltransform -infile ..¥xml¥policies¥xml¥samplestorenameAccessPolicies.xml -transform ..¥xml¥policies¥xsl¥accesscontrol.xsl -outfile ..¥xml¥policies¥xml¥samplestorenameAccessPoliciesOut.xml

      400 TRNWCSXML INFILE (input file) TRANSFORM('/QIBM/ProdData/WebCommerce/xml/policies/xsl/accesscontrol.xsl') INSTROOT(instance_root) OUTFILE(output_file)
   c. 次のログ・ファイルを調べ、変換が正常に完了したことを確認します。
      - NT drive:¥WebSphere¥CommerceServer¥bin¥xmltransform.db2.log
      - 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥bin¥xmltransform.db2.log

- AIX /usr/WebSphere/CommerceServer/bin/xmltransform.db2.log
- Solaris /opt/WebSphere/CommerceServer/bin/xmltransform.db2.log
- Linux /opt/WebSphere/CommerceServer/bin/xmltransform.db2.log
- 400 /QIBM/UserData/WebCommerce/instances/instancename /logs/TRNWCSXML.tx

  変換が成功したら、次のメッセージが表示されます。 "<DATE> <TIME> java.lang.Class main XMLTransformer Transform Successful"

8. xmltransform コマンドを実行して、*storename*AccessPolicies_locale.xml を変換します。
   a. コマンド・プロンプトで、ディレクトリーを次のように変更します。
      - NT drive:¥WebSphere¥CommerceServer¥bin
      - 2000 drive:¥ProgramFiles¥WebSphere¥CommerceServer¥bin
      - AIX /usr/WebSphere/CommerceServer/bin
      - Solaris /opt/WebSphere/CommerceServer/bin
      - Linux /opt/WebSphere/CommerceServer/bin
   b. 次のように入力します。 xmltransform -infile ..¥xml¥policies¥xml¥*storename*AccessPolicies_locale.xml -transform ..¥xml¥policies¥xsl¥accesscontrolnls.xsl -outfile ..¥xml¥policies¥xml¥*storename*AccessPoliciesOut_locale.xml
   c. 400 TRNWCSXML INFILE(input file) TRANSFORM('/QIBM/ProdData/WebCommerce/xml/policies/xsl/accesscontrolnls.xsl') INSTROOT(instance_root) OUTFILE(output_file)
9. 生成される XML ファイルに、以下の変更を加えます。
   a. *storename*AccessPolicesOut.xml で、開始タグと終了タグを以下のものに置き換えます。

      ```
      <?xml version="1.0"?>
      <!DOCTYPE accesscontrol-asset SYSTEM "accesscontrol.dtd">
      <accesscontrol-asset>
      </accesscontrol-asset>
      ```

   b. *storename*AccessPolicesOut_locale.xml で、開始タグと終了タグを以下のものに置き換えます。

      ```
      <?xml version="1.0" encoding="correct language code for the file"?>
      <!DOCTYPE accesscontrol-asset SYSTEM "../accesscontrol.dtd">
      <accesscontrol-asset>
      </accesscontrol-asset>
      ```

   c. *storename*AccessPolicesOut_locale.xml で、 @locale を &locale に置き換えます。たとえば、LANGUAGE_ID=″@en_US″ を LANGUAGE_ID=″&en_US;″ に変更します。
   d. *storename*AccessPolicesOut_locale.xml で、 ″acpoldesc″ テーブルへの参照を探し出します。 ACPOLICY_ID 値の最後にある @ を除去します。たとえば、"@AllUsersExecuteInFashionAllUsersViews@" を "@AllUsersExecuteInFashionAllUsersViews" に変更します。

e. *storename*AccessPolicesOut.xml で、 MEMBER_ID="MEMBER_ID" を MEMBER_ID="&MEMBER_ID;" に置き換えます。

f. *storename*AccessPolicesOut.xml で、 "acpolicy" テーブルへの参照を探し出します。 ACPOLICY_ID 値の最後にある "@MEMBER_ID" を除去します。たとえば、"@AllUsersExecuteInFashionAllUsersViews@MEMBER_ID" を "@AllUsersExecuteInFashionAllUsersViews" に変更します。

10. この時点で、以下の 2 つのオプションがあります。
    - ローダー・パッケージを使用してアクセス・コントロール・データをロードする。詳細については、 225 ページの『第 7 部 ストアの発行』を参照してください。
    - アクセス・コントロール・データをストア・アーカイブに追加する。ストア・アーカイブの作成の詳細については、 215 ページの『第 6 部 ストアのパッケージ化』を参照してください。

@ と **&** の使用法の詳細については、 313 ページの『付録 B. データの作成』を参照してください。

ストア・アーカイブのアクセス・コントロール・ファイルの編集

WebSphere Commerce には、サンプル・ストアごとに 2 つの変換済みアクセス・コントロール XML ファイルが備えられています。一方はすべての言語に適用され (*samplestorename*AccessPolicies.xml)、もう一方はロケール固有情報を含みます (*samplestorename*AccessPolicies_locale.xml)。それらのファイルのそれぞれを変換する必要があります。そうすると、2 つの XML ファイルが生成されます。一方はすべての言語に適用され (*samplestorename*AccessPoliciesOut.xml)、もう一方はロケール固有情報を含みます (*samplestorename*AccessPoliciesOut_locale.xml)。

ストア・アーカイブのアクセス・コントロール・データベース資産を編集するには、以下のようにします。

1. ストアのベースとして使用したサンプル・ストアの事前変換済みアクセス・コントロール XML ファイルを探し出します。これらのファイルは、*samplestorename*AccessPolicies.xml と *samplestorename*AccessPolicies_locale.xml と呼ばれます。これらのファイルは、デフォルトでは、次のディレクトリーにあります。
    - NT drive:¥WebSphere¥CommerceServer¥samples¥stores¥*samplestorename*
    - 2000 drive:¥ProgramFiles¥WebSphere¥CommerceServer¥samples¥stores¥*samplestorename*
    - AIX /usr/WebSphere/CommerceServer/samples/stores/*samplestorename*
    - Solaris /opt/WebSphere/CommerceServer/samples/stores/*samplestorename*
    - Linux /opt/WebSphere/CommerceServer/samples/stores/*samplestorename*
    - 400 /QIBM/ProdData/WebCommerce/samples/stores/*samplestorename*

    ここで *samplestorename* は、ストアのベースとして使用したサンプル・ストア・アーカイブの名前 (たとえば、NewFashion) です。

注: 対応する DTD ファイルを変更すると、不必要なポリシーが生成される場合があります。

2. ファイルに対して必要な変更を加えます。既存のファイルについての詳細は、204 ページの『サンプル・ストアのアクセス・コントロール・ポリシー・ファイルについて』 を参照してください。 WebSphere Commerce で使用できるアクセス・コントロールの方式の詳細は、 *IBM WebSphere Commerce アクセス・コントロール・ガイド* を参照してください。
3. *samplestorename*AccessPolicies.xml および *samplestorename*AccessPolicies_locale.xml を、次のディレクトリーにコピーします。
   - NT drive:¥WebSphere¥CommerceServer¥xml¥policies¥xml
   - 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥xml¥policies¥xml
   - AIX /usr/WebSphere/CommerceServer/xml/policies/xml
   - Solaris /opt/WebSphere/CommerceServer/xml/policies/xml
   - Linux /opt/WebSphere/CommerceServer/xml/policies/xml
   - 400 任意のユーザー data ディレクトリーにコピーします。既存の XML ファイルで、DTD への絶対パスを指定します。アクセス・コントロール DTD ファイルは、/QIBM/ProdData/WebCommerce/xml/policies/dtd ディレクトリーにあります。
4. xmltransform コマンドを実行して、*samplestorename*AccessPolicies.xml を変換します。
   a. コマンド・プロンプトで、ディレクトリーを次のように変更します。
      - NT drive:¥WebSphere¥CommerceServer¥bin
      - 2000 drive:¥ProgramFiles¥WebSphere¥CommerceServer¥bin
      - AIX /usr/WebSphere/CommerceServer/bin
      - Solaris /opt/WebSphere/CommerceServer/bin
      - Linux /opt/WebSphere/CommerceServer/bin
   b. 次のように入力します。
      ```
      xmltransform -infile
      ..¥xml¥policies¥xml¥samplestorenameAccessPolicies -transform
      ..¥xml¥policies¥xsl¥accesscontrol.xsl -outfile
      ..¥xml¥policies¥xml¥samplestorenameAccessPoliciesOut.xml
      ```
      400
      ```
      TRANSFORM('/QIBM/ProdData/WebCommerce/xml/policies
      /xsl/accesscontrol.xsl')
      INSTROOT(instance_root) OUTFILE(output_file)
      ```
   c. 次のログ・ファイルを調べ、変換が正常に完了したことを確認します。
      - NT drive:¥WebSphere¥CommerceServer¥bin¥xmltransform.db2.log
      - 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥bin¥xmltransform.db2.log

- AIX /usr/WebSphere/CommerceServer/bin/xmltransform.db2.log
- Solaris /opt/WebSphere/CommerceServer/bin/xmltransform.db2.log
- Linux /opt/WebSphere/CommerceServer/bin/xmltransform.db2.log
- 400 /QIBM/UserData/WebCommerce/instances/instancename/logs/TRNWCSXML.tx
  変換が成功したら、次のメッセージが表示されます。 "<DATE> <TIME> java.lang.Class main XMLTransformer Transform Successful"

5. xmltransform コマンドを実行して、*samplestorename*AccessPolicies_locale.xml を変換します。
   a. コマンド・プロンプトで、ディレクトリーを次のように変更します。
      - NT drive:¥WebSphere¥CommerceServer¥bin
      - 2000 drive:¥ProgramFiles¥WebSphere¥CommerceServer¥bin
      - AIX /usr/WebSphere/CommerceServer/bin
      - Solaris /opt/WebSphere/CommerceServer/bin
      - Linux /opt/WebSphere/CommerceServer/bin
   b. 次のように入力します。 xmltransform -infile ..¥xml¥policies¥xml¥samplestorenameAccessPolicies_locale.xml -transform ..¥xml¥policies¥xsl¥accesscontrolnls.xsl -outfile ..¥xml¥policies¥xml¥samplestorenameAccessPoliciesOut_locale.xml
   c. 400 TRNWCSXML INFILE(input file) TRANSFORM('/QIBM/ProdData/WebCommerce/xml/policies/xsl/accesscontrolnls.xsl') INSTROOT(instance_root) OUTFILE(output_file)
6. 生成される XML ファイルに、以下の変更を加えます。
   a. *samplestorename*AccessPolicesOut.xml で、開始タグと終了タグを以下のものに置き換えます。

```
<?xml version="1.0"?>
<!DOCTYPE accesscontrol-asset SYSTEM "accesscontrol.dtd">
<accesscontrol-asset>
</accesscontrol-asset>
```

   b. *samplestorename*AccessPolicesOut_locale.xml で、開始タグと終了タグを以下のものに置き換えます。

```
<?xml version="1.0" encoding="correct language code for the file"?>
<!DOCTYPE accesscontrol-asset SYSTEM "../accesscontrol.dtd">
<accesscontrol-asset>
</accesscontrol-asset>
```

   c. *samplestorename*AccessPolicesOut_locale.xml で、 @locale を &locale に置き換えます。たとえば、LANGUAGE_ID=″@en_US″ を LANGUAGE_ID=″&en_US;″ に変更します。
   d. *samplestorename*AccessPolicesOut_locale.xml で、 ″acpoldesc″ テーブルへの参照を探し出します。 ACPOLICY_ID 値の最後にある @ を除去します。たとえば、"@AllUsersExecuteInFashionAllUsersViews@" を "@AllUsersExecuteInFashionAllUsersViews" に変更します。

e. *samplestorename*AccessPolicesOut.xml で、 MEMBER_ID=″MEMBER_ID″ を MEMBER_ID=″&MEMBER_ID;″ に置き換えます。

f. *samplestorename*AccessPolicesOut.xml で、 ″acpolicy″ テーブルへの参照を探し出します。 ACPOLICY_ID 値の最後にある ″@MEMBER_ID″ を除去します。たとえば、"@AllUsersExecuteInFashionAllUsersViews@MEMBER_ID" を "@AllUsersExecuteInFashionAllUsersViews" に変更します。

7. ストアのストア・アーカイブ・ファイルを探し出します (たとえば、mystore.sar)。ストア・アーカイブ・ファイルは、デフォルトでは、次のディレクトリーにあります。

- NT drive:¥WebSphere¥CommerceServer¥instances¥instancename¥sar
- 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥instances¥instancename ¥sar
- AIX /usr/WebSphere/CommerceServer/instances/instancename/sar
- Solaris /opt/WebSphere/CommerceServer/instances/instancename/sar
- Linux /opt/WebSphere/CommerceServer/instances/instancename/sar
- 400 /QIBM/UserData/WebCommerce/instances/instancename/sar

8. *samplestorename*AccessPolicesOut.xml および *samplestorename*AccessPoliciesOut_locale.xml を、 accesscontrol.xml と名前変更します。

注: ロケール固有の accesscontrol.xml ファイルは、デフォルトでは、データ/ロケール・ディレクトリー (たとえば、data/en_US) にあります。

9. ZIP プログラムを使用して、ストア・アーカイブ・ファイルをオープンします。

10. ストア・アーカイブ・ファイルにある既存の accesscontrol.xml とロケール固有の accesscontrol.xml を、ステップ 8 で名前変更したファイルに置き換えます。

11. ストア・アーカイブ・ファイルを保管します。

第 6 部 ストアのパッケージ化

第 25 章 ストアのパッケージ化

ストアの使用目的が、これをサンプルとして他人に送付すること、別のサーバーまたは別のプラットフォームにデプロイすること、あるいは他のストアを作成する基盤として使用することであるなら、これをストア・アーカイブの形でパッケージ化できます。

WebSphere Commerce で提供されているサンプル・ストアは、ストア・アーカイブとしてパッケージ化されています。サンプル・ストア・アーカイブに含まれるファイルは、次のように分類されます。

- Web 資産: HTML ファイル、JSP ファイル、イメージ、グラフィックス、および組み込みファイルなど、ストア・ページの作成に使用するファイルです。 Web 資産はストア・アーカイブの圧縮ファイル (webapp.zip) として、グループ化されます。
- プロパティー・リソース・バンドル (オプション): ストア・ページのテキストが入っています。ストアが複数の言語をサポートする場合、リソース・バンドルには複数のバンドル (言語ごとに 1 つ) があります。プロパティー・リソース・バンドルは、ストア・アーカイブ内で圧縮ファイル (properties.zip) として分類されます。
- ストア・データ資産: データベースにロードされるデータ。ストア・データ資産には、キャンペーン、カタログ、通貨、フルフィルメント情報、価格設定、配送、ストア、税情報などのデータが含まれます。
- 支払資産: IBM Payment Manager の構成情報。
- ディスクリプター: XML ファイル sarinfo.xml。 Web 資産圧縮アーカイブ・ファイル、リソース・バンドル、およびストア・データベース資産 XML ファイルの名前を含むストア・アーカイブについて記述します。 sarinfo.xml ファイルには、組み込みファイルと整合性検査ファイルの名前、および発行処理中に必要なアーカイブ・ファイルについての情報が含まれます。ストア・アーカイブで必須のファイルは sarinfo.xml だけです。

ストア・アーカイブの作成

ストアをストア・アーカイブとしてパッケージ化するには、次のようにします。

1. WebSphere Commerce に含まれているサンプル・ストア・アーカイブの構造や内容を調べます。

   ストア・アーカイブ・ファイルは、以下に示すディレクトリーにあります。

   - NT drive:¥WebSphere¥CommerceServer¥samplestores
   - 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥samplestores
   - AIX /usr/WeSphere/CommerceServer/samplestores
   - Solaris /opt/WebSphere/CommerceServer/samplestores
   - Linux /opt/WebSphere/CommerceServer/samplestores

- 400 /qibm/proddata/WebCommerce/samplestores

ストア・アーカイブを表示するには、解凍プログラムを使います。

2. ストアの WebSphere Commerce Server に、一時ディレクトリーを作成します。たとえば、*mystore* とします。
3. Web 資産 (JSP ファイル、HTML、イメージ) をまとめて、一時ディレクトリー内の webapp というディレクトリーの中に入れます。 ZIP プログラムを使用して、 webapp フォルダーの圧縮アーカイブ・ファイルを作成します。
4. (オプション) プロパティー・リソース・バンドルをまとめて、一時ディレクトリー内の properties というディレクトリー内に入れます。ストアが複数の言語をサポートする場合、リソース・バンドルには複数のバンドル (言語ごとに 1 つ) があります。 ZIP プログラムを使用して、 properties フォルダーの圧縮アーカイブ・ファイルを作成します。
5. ストア・データ資産をまとめて、一時ディレクトリー内の data というディレクトリーに入れます。ストアで複数の言語をサポートする場合は、ロケール名を使用することにより、言語特有の情報を入れるためのサブディレクトリーを作成します。たとえば、en_US のようにします。
6. (オプション) 既存のストア・アーカイブにある sarrule.xml ファイルを、data ディレクトリーにコピーします。 sarrule.xml ファイルは、サンプル・ストア・アーカイブ内の data ディレクトリーに含まれています。この sarrule.xml を使用して、ストア・サービスでの発行の際に一貫性のチェックを行うことができます。 sarrule ファイルについては、 323 ページの『付録 D. sarrule.xml』を参照してください。
7. 一時ディレクトリー内に SAR-INF というディレクトリーを作成します。
8. ストア・アーカイブのための sarinfo.xml ファイルを作成します。 XML の仕様については、以下に示すディレクトリーに含まれているアーカイブ・ディスクリプター sarinfo.dtd を参照してください。
   - NT drive:¥WebSphere¥CommerceServer¥xml¥sar
   - 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥xml¥sar
   - AIX /usr/WeSphere/CommerceServer/xml/sar
   - Solaris /opt/WebSphere/CommercServer/xml/sar
   - Linux /opt/WebSphere/CommercServer/xml/sar
   - 400 /qibm/proddata/WebCommerce/xml/sar

   a. 317 ページの『付録 C. sarinfo.xml』にあるサンプルなどの情報を参考にして、 sarinfo.xml ファイルを作成します。データ資産の中には他のデータ資産より前に発行しなければならないものがあるため、データ資産の発行順は重要です。そのため、sarinfo.xml ファイルで指定される資産の順序は、サンプル・ストアの sarinfo.xml ファイルで指定されている資産の順序と同じでなければなりません。

      注: ストア・アーカイブに sarrule.xml ファイルを組み込む (ステップ 6 を参照) 場合は、sarrule.xml に関する情報を sarinfo.xml ファイルに

組み込む必要があります。逆に、sarrule.xml ファイルを組み込まないのであれば、sarinfo.xml ファイル内にこのファイルに関する記述が含まれないようにしてください。

b. sarinfo.xml を、ステップ 6 で作成した SAR-INF ディレクトリーに保存します。

9. Web 資産、プロパティー・リソース・バンドル、ストア・データ資産、および sar-inf ディレクトリーで構成される ZIP ファイルを作成します。その ZIP ファイルの名前は *storearchivename.sar* とします。
10. ストア・アーカイブを使用してストア・アーカイブを編集または発行する場合には、 *storearchivename.sar* を以下に示すディレクトリーに保存します。
    - NT drive:¥WebSphere¥CommerceServer¥instances¥*instancename*¥sar
    - 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥instances¥*instancename*¥sar
    - AIX /usr/WebSphere/CommerceServer/instances/*instancename*/sar
    - Solaris /opt/WebSphere/CommerceServer/instances/*instancename*/sar
    - Linux /opt/WebSphere/CommerceServer/instances/*instancename*/sar
    - 400 /QIBM/UserData/WebCommerce/instances/*instancename*/sar
11. これで、ストア・サービス内のストア・アーカイブのリストの中に、作成したストア・アーカイブが表示されるようになります。

サンプル・ストア・アーカイブの作成

ストアをストア・アーカイブとしてパッケージ化したなら、ストア・サービスにおいてそれをサンプル・ストアとして使用できます。サンプル・ストア・アーカイブは、新規ストア作成の基礎としてコピーして使用するためのストア・アーカイブです。ストア・アーカイブをサンプル・ストア・アーカイブとしてパッケージ化するには、次のようにします。

1. ストア・アーカイブ・ファイルを、以下に示すディレクトリーに保存します。
    - NT drive:¥WebSphere¥CommerceServer¥samplestores¥*storeachivename*
    - 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥samplestores¥*storeachivename*
    - AIX /usr/WeSphere/CommerceServer/samplestores/*storeachivename*
    - Solaris /opt/WebSphere/CommerceServer/samplestores/*storeachivename*
    - Linux /opt/WebSphere/CommerceServer/samplestores/*storeachivename*
    - 400 /qibm/proddata/WebCommerce/samplestores/*storeachivename*
2. (オプション) プレビュー・ページを作成します。ストア・ページのプレビューをストア・サービスに表示するためには、まずプレビュー・ページを作成する必要があります。以下のようにします。

    a. (オプション) ストア・サービスで、**「新規」**を選択します。「ストア・アーカイブの作成」ページが表示されます。**「Samples (サンプル)」**リストに示

されているサンプル・ストアの中から 1 つ選択し、**「プレビュー」**をクリックします。表示されるページは、プレビュー・ページと呼ばれます。それらのページは、事前に定義された買い物の流れのサンプルを示すものであり、サンプル・ストアのプレビューとして使用される HTML ファイルです。

b. プレビュー・ページに表示する買い物の流れを決定します。

c. (オプション) 発行済みストアの中にサンプル・データを作成します。たとえば、アイテムをショッピング・カートに追加し、いくつかの配送先住所と請求先住所を作成します。このストアからプレビュー・ページを作成し、データを入れることによってページがリアルなものになります。

d. Internet Explorer を使用して、ストアを表示します。ページごとに「ファイル」→「名前を付けて保存」を選択することにより、 HTML を保存します。さらに、スタイルシート (.css) とイメージも保存する必要があります。それらのファイルは、以下に示すディレクトリーに保存します。

- *stylesheet*.css
  - NT drive:¥WebSphere¥AppServer¥installedApps¥WC_Enterprise_App_*instancename*.ear¥wcstools.war¥tools¥devtools¥*storearchivename*¥preview
  - 2000 drive:¥Program Files¥WebSphere¥AppServer¥installedApps¥WC_Enterprise_App_*instancename*.ear¥wcstools.war¥tools¥devtools¥*storearchivename*¥preview
  - AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/wcstools.war/tools/devtools/*storearchivename*/preview
  - Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename.ear*/wcstools.war/tools/devtools/*storearchivename*/preview
  - Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename.ear*/wcstools.war/tools/devtools/*storearchivename*/preview
  - 400 /Qibm/UserData/WebAsAdv4/WAS*instancename*/installedApps/WC_Enterprise_App_*instancename*.ear/wcstools.war/tools/devtools/*storearchivename*/preview
- HTML
  - NT drive:¥WebSphere¥AppServer¥installedApps¥WC_Enterprise_App_*instancename*.ear¥wcstools.war¥tools¥devtools¥*storearchivename*¥preview¥*locale*
  - 2000 drive:¥Program Files¥WebSphere¥AppServer¥installedApps¥WC_Enterprise_App_*instancename*.ear¥wcstools.war¥tools¥devtools¥*storearchivename*¥preview¥*locale*
  - AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/wcstools.war/tools/devtools/*storearchivename*/preview/*locale*

- Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename.ear*/wcstools.war/tools/devtools/*storearchivename*/preview/*locale*
- Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename.ear*/wcstools.war/tools/devtools/*storearchivename*/preview/*locale*
- 400 /Qibm/UserData/WebAsAdv4/WAS*instancename*/installedApps/WC_Enterprise_App_*instancename*.ear/wcstools.war/tools/devtools/*storearchivename*/preview/*locale*

- ロケールに依存しないイメージ
  - NT drive:¥WebSphere¥AppServer¥installedApps¥WC_Enterprise_App_*instancename*.ear¥wcstools.war¥tools¥devtools¥*storearchivename*¥preview¥images
  - 2000 drive:¥Program Files¥WebSphere¥AppServer¥installedApps¥WC_Enterprise_App_*instancename*.ear¥wcstools.war¥tools¥devtools¥*storearchivename*¥preview¥images
  - AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/wcstools.war/tools/devtools/*storearchivename*/preview/images
  - Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename.ear*/wcstools.war/tools/devtools/*storearchivename*/preview/images
  - Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename.ear*/wcstools.war/tools/devtools/*storearchivename*/preview/images
  - 400 /Qibm/UserData/WebAsAdv4/WAS*instancename*/installedApps/WC_Enterprise_App_*instancename*.ear/wcstools.war/tools/devtools/*storearchivename*/preview/images
- ロケールに依存するイメージ
  - NT drive:¥WebSphere¥AppServer¥installedApps¥WC_Enterprise_App_*instancename*.ear¥wcstools.war¥tools¥devtools¥*storearchivename*¥preview¥*locale*¥images
  - 2000 drive:¥Program Files¥WebSphere¥AppServer¥installedApps¥WC_Enterprise_App_*instancename*.ear¥wcstools.war¥tools¥devtools¥*storearchivename*¥preview¥*locale*¥images
  - AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/wcstools.war/tools/devtools/*storearchivename*/preview/*locale*/images

– Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename.ear*/wcstools.war/tools/devtools/*storearchivename*/preview/*locale*/images

– Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename.ear*/wcstools.war/tools/devtools/*storearchivename*/preview/*locale*/images

– 400 /Qibm/UserData/WebAsAdv4/WAS*instancename*/installedApps/WC_Enterprise_App_*instancename*.ear/wcstools.war/tools/devtools*storearchivename*/preview/*locale*/images

e. イメージと css ファイルの位置が変更されているため、 HTML ページに含まれるイメージと css ファイルへの参照を変更する必要があります。参照を変更したなら、 HTML ページをブラウザーで開いた場合にイメージが正常に表示されることを確認してください。

f. HTML ページに含まれるリンクを、コマンドではなく、HTML ファイルを参照するリンクに変更します。

3. ストア・アーカイブを要約する HTML ファイルを作成します。この情報は、ストア・サービスの「ストア・アーカイブの作成」ページの**「サンプルの説明」**に表示されます。

a. 次の例を参考にして、新しいファイルを作成します。

```
<doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=shift-jis">
</head>
<body>
Describe store here
</body>
</html>
```

b. このファイルを Feature_*locale*.html として保存します (locale は使用する言語に対応する略語)。たとえば、en_US のようにします。このファイルを、以下に示すディレクトリーに保存します。

- NT drive:¥WebSphere¥CommerceServer¥samplestores¥*storeachivename*
- 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥samplestores¥*storeachivename*
- AIX /usr/WeSphere/CommerceServer/samplestores/*storeachivename*
- Solaris /opt/WebSphere/CommerceServer/samplestores/*storeachivename*
- Linux /opt/WebSphere/CommerceServer/samplestores/*storeachivename*
- 400 /qibm/proddata/WebCommerce/samplestores/*storeachivename*

4. ストア・アーカイブを sarregistry.xml ファイルに追加します。その sarregistry.xml ファイルによって、ストア・サービスの「ストア・アーカイブの作成」ページに含まれる**「サンプル」**リストに表示されるストア・アーカイブが決定されます。この sarregistry.xml は、各ストア・アーカイブに関連付け

るプレビュー・ページとフィーチャー・ファイルも決定します。
sarregistry.xml は、以下に示すディレクトリーにあります。

- NT drive:¥WebSphere¥CommerceServer¥xml¥tools¥devtools
- 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥xml¥tools¥devtools
- AIX /usr/WebSphere/CommerceServer/xml/tools/devtools
- Solaris /opt/WebSphere/CommerceServer/xml/tools/devtools
- Linux /opt/WebSphere/CommerceServer/xml/tools/devtools
- 400 /QIBM/ProdData/WebCommerce/xml/tools/devtools

a. 以下の例を参考にして、新しいストア・アーカイブを sarregistry.xml に追加します。

```
<SampleSAR fileName="infashion_en_US_es_ES.sar" relativePath="InFashion">
        <html locale="es_ES" featureFile="InFashion/Feature_es_ES.html"
        sampleSite="RetailModel/preview/es_ES/index.html"/>
        <html locale="en_US" featureFile="InFashion/Feature_en_US.html"
        sampleSite="RetailModel/preview/en_US/index.html"/>
</SampleSAR>
```

この例は、英語とスペイン語の InFashion ストア・アーカイブのためのプレビュー・ページを定義するものです。太字で示した行で、英語とスペイン語のプレビュー・ページのためのホーム・ページが定義されています。

5. これで、ストア・サービスの「ストア・アーカイブの作成」ページの**「サンプル」**リストに、作成したストア・アーカイブが表示されるようになります。

第 7 部 ストアの発行

機能するストアを作成するには、ストアフロント Web 資産を WebSphere Commerce Server に発行し、ストア・データを WebSphere Commerce データベースに発行する必要があります。

ここに含まれる章では、WebSphere Commerce に備わっている発行オプションについて説明します。

- 227 ページの『第 26 章 ストア全体の発行』 - この章では、ストアがストア・アーカイブのフォームになっている場合に、ストア・サービスかコマンド行の発行ユーティリティーを使用してストア全体 (ストアフロントとストア・データ資産) を発行する方法について説明します。
- 239 ページの『第 27 章 ストアフロント資産とストア構成ファイルの発行』 - この章では、ストアフロント資産とストア構成ファイルの発行について説明します。
- 243 ページの『第 28 章 ストア・データのロードの概要』 - この章では、ローダー・パッケージと他の WebSphere Catalog Manager コンポーネントを使用してストア・データ資産をデータベースに発行する方法について説明します。
- 279 ページの『第 29 章 WebSphere Commerce データ・グループのロード』 - この章では、ローダー・パッケージや他の WebSphere Catalog Manager コンポーネントを使用してストア・データ資産のグループやストア・データ全体をデータベースに発行する方法について説明します。
- 289 ページの『第 30 章 アカウント、契約、および商品セットの発行』 - この章では、アカウント、契約、および商品セット資産の発行について説明します。

第 26 章 ストア全体の発行

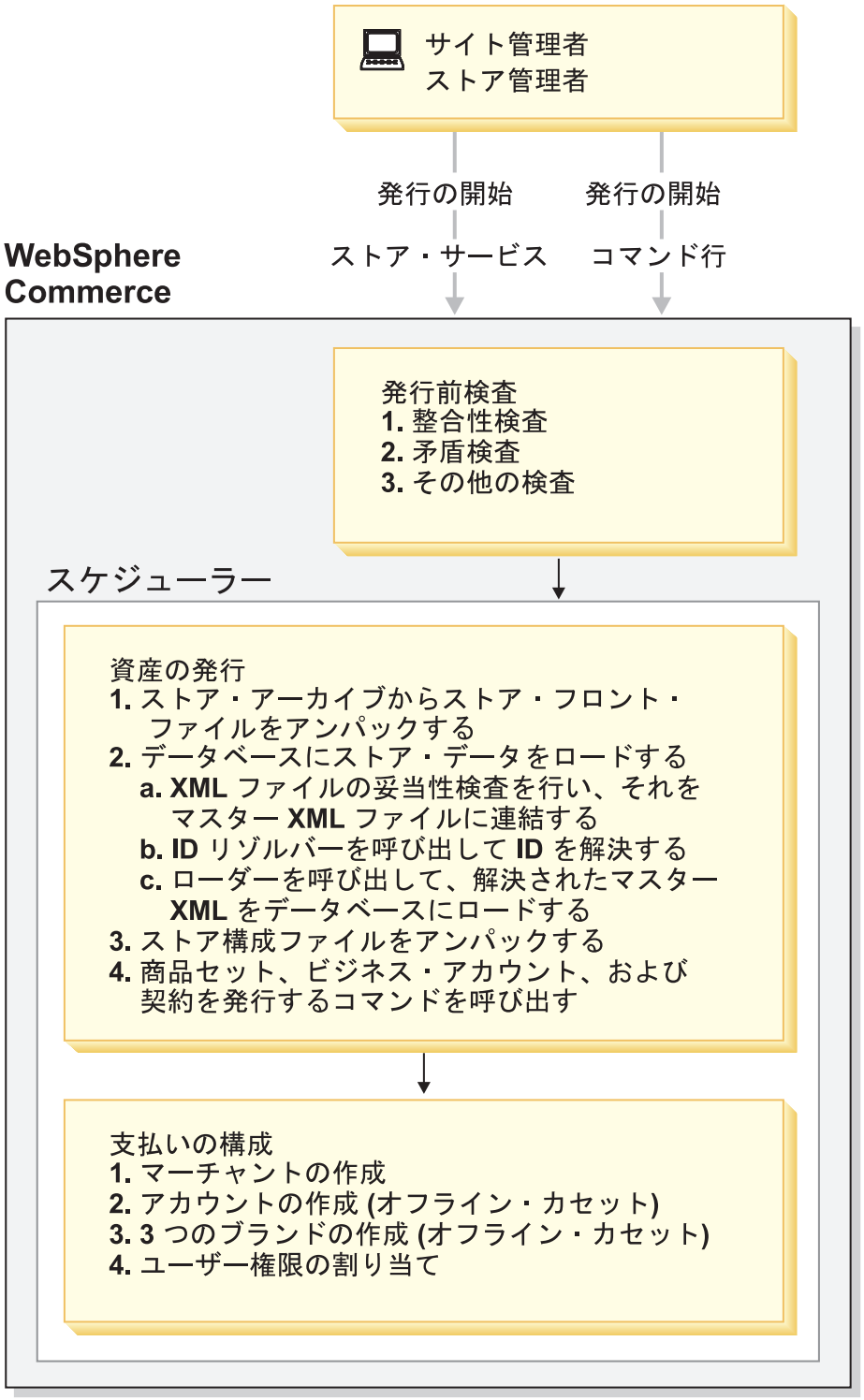
機能するストアを作成するには、ストアフロント Web 資産を WebSphere Commerce Server に発行し、ストア・データを WebSphere Commerce データベースに発行する必要があります。この章では、ストアがストア・アーカイブのフォームになっている場合に、ストア・サービスかコマンド行の発行ユーティリティーを使用してストア全体 (ストアフロントとストア・データ資産) を発行する方法について説明します。

注: ストアをストア・アーカイブとしてパッケージしないほうが好ましい場合は、個々の資産を 1 つずつ発行することができます。詳細については、239 ページの『第 27 章 ストアフロント資産とストア構成ファイルの発行』、 243 ページの『第 28 章 ストア・データのロードの概要』、 279 ページの『第 29 章 WebSphere Commerce データ・グループのロード』、および 289 ページの『第 30 章 アカウント、契約、および商品セットの発行』を参照してください。

WebSphere Commerce での発行について

ストア・サービスやコマンド行で使用できる発行オプションを使うと、ストア全体 (ストアフロントとストアのデータ資産) を一度にまとめて発行できます。このオプションを使用するためには、ストア資産がストア・アーカイブのフォームでパッケージされていなければなりません。ストアをストア・アーカイブとしてパッケージする処理については、215 ページの『第 6 部 ストアのパッケージ化』を参照してください。

次の図は、発行プロセスの手順を簡単にまとめたものです。



発行の開始

ストアを発行するためには、サイト管理者権限またはストア管理者権限 (すべてのストアに対する) が必要です。サイト管理者やストア管理者は、次のいずれかの方法を使用して、発行のプロセスを開始することができます。

- ストア・サービス
- コマンド行

どちらの方法で発行を行う場合も、発行したいストア・アーカイブの選択が必要です。また、どちらの方法を使用する場合でも、ストア・アーカイブをどの程度発行するかを選択できます。たとえば、次の任意の組み合わせを選択して発行できます。

- ストアのデータベース資産 (オンライン・カタログ・データあり / なし)
- JSP ファイル、HTML ファイル、およびイメージなどの Web 資産
- プロパティー・ファイル (ストア用のテキスト)

初回の発行では、ストアの機能する様子が見られるように、ストア・アーカイブ全体 (上のすべて) を発行することが勧められています。しかし、2 回目以降の発行では、ストア・アーカイブ全体をまた発行しなおすのではなく、データベース資産、Web 資産、またはプロパティー・リソース・バンドルのいずれか 1 つだけを更新したい場合があるかもしれません。

ストア・アーカイブの初回の発行で、オンライン・カタログ・データの発行を選択しなかった場合は、カタログ内のデータを使用する一部のデータ資源 (契約、アカウント、および商品セット) が適切に発行されない可能性があります。また、初回発行時にカタログ・データと Web 資産の発行が選択されなかった場合は、ストア ID、カタログ ID、および言語 ID が作成されないため、ストア・サービスからストアを起動することができません。

オンライン・カタログ・データは、ストア・アーカイブ内の以下の XML ファイルで構成されます。

- `catalog.xml`
- `offering.xml`
- `store-catalog.xml`
- `store-catalog-shipping.xml`
- `store-catalog-tax.xml`
- `storefulfill.xml`

この情報を後日発行する場合は、ストア・サービスを使用するか、ローダー・パッケージからこれを発行できます。上にリストされているカタログ・グループ・データのようなデータのグループのロードについては、 281 ページの『データ・グループのロード』を参照してください。

ストア・アーカイブやコマンド行を使用してストア・アーカイブを発行する方法についての詳しい説明は、 WebSphere Commerce オンライン・ヘルプを参照してください。

注: ストア・サービスやコマンド行を使用して発行のプロセスを開始した後は、操作は一切必要ありません。先の図やこの章でリストされているその他のステップは、すべて、WebSphere Commerce システムによって実行されます。

発行前検査

サイト管理者やストア管理者が発行を開始すると、WebSphere Commerce は、実際の発行プロセスを開始する前にいくつかの検査を実行します。実行されるのは、次のような検査です。

- 整合性検査
- 矛盾検査
- その他の検査

整合性検査

整合性検査の情報は、ストア・アーカイブの sarrule.xml ファイルに含まれています。発行前の検査では、sarrule.xml ファイルに示されている規則を使用して、XML ファイルの情報とストア・アーカイブの Web 資産の整合性が確認されます。たとえば、command.xml ファイルが特定の JSP ファイルを参照する場合は、ストア・アーカイブの webapp.zip にその JSP ファイルがあるかどうかを確認する検査が行われます。なお、整合性検査がエラーを検出すると、エラーがログに書き込まれますが、発行は通常通り続行されます。ログ・ファイルに関する詳細は、237 ページの『ログ・ファイルの発行』を参照してください。

sarrule.xml ファイルのサンプルは、323 ページの『付録 D. sarrule.xml』を参照してください。

矛盾検査

発行前検査では、ストアとカタログの間の矛盾も WebSphere Commerce によって検査されます。とりわけ、以下を調べるためのデータベースの検査が実行されます。

- 同一の識別名を持つストアの存在: ストアがすでに存在する場合は、そのストアを上書きするか、発行をキャンセルするかを尋ねるメッセージが表示されます。
- 同一のカタログ ID を持つカタログの存在: カタログがすでに存在する場合は、そのカタログを上書きするか、発行をキャンセルするかを尋ねるメッセージが表示されます。なお、既存のカタログが別のストアに属しているときは、そのカタログがどのストアに属しているかが示され、そのまま発行を続行してカタログを上書きするか、発行をキャンセルするかを尋ねるメッセージが表示されます。

その他の検査

WebSphere Commerce の発行前検査では、スケジューラーが使用可能になっているかどうか、キャッシュ・トリガーとキャッシュが使用不可になっているかどうか、および要約テーブルが使用不可になっているかどうかの検査も行われます。これらの設定が間違っている場合は、使用可能/使用不可が適切に設定されていない機能を知らせる警告メッセージが表示されます。

スケジューラー: この後のセクションでさらに詳しく扱いますが、発行のジョブは、スケジューラーによって実行されます。したがって、スケジューラーが使用不可になっていると、発行のプロセスは実行できません。

キャッシュおよびキャッシュ・トリガー: 発行の際にキャッシュを残しておくと、発行でデータベースが更新されたときにキャッシュ・トリガーが起動されてしまう可能性があります。キャッシュ・トリガーで不要なデータベース・アクティビティーが生まれれば、データベース・トランザクションのログがオーバーフローし、発

行のパフォーマンスに影響を与えかねません。キャッシュ・トリガーを使用不可にする操作については、WebSphere Commerce オンライン・ヘルプを参照してください。

要約テーブル: DB2 要約テーブルを使用可能なままにしておくと、発行の途中で要約テーブルが更新されてしまい、結果として、データベース・トランザクションのログがオーバーフローして発行のパフォーマンスに影響を与える可能性があります。要約テーブルを使用不可にする操作については、WebSphere Commerce オンライン・ヘルプを参照してください。

資産の発行

発行プロセスの資産発行のフェーズは、スケジュール・ジョブとしてスケジューラーによって実行されます。スケジューラーが発行のジョブを実行すると、WebSphere Commerce は以下のアクションを完了させます。

- ストア・アーカイブからストアフロント・ファイルをアンパックする
- ストア・アーカイブ内の XML ファイルからデータベースにストア・データをロードする
- ストア構成ファイルをアンパックする
- 商品セット、アカウント、および契約を発行するコマンドを呼び出す

ストア・アーカイブからストアフロント・ファイルをアンパックする

資産発行のフェーズで最初に実行されるアクションは、 Web 資産をストア・アーカイブから WebSphere Commerce Server にアンパックする作業です。ストア・アーカイブからファイルをアンパックする際、WebSphere Commerce は以下の作業を行います。

Web 資産をアンパックし、それを WebSphere Commerce Server 上の以下のロケーションにコピーする:

- JSP ファイル、HTML ファイル、イメージおよびグラフィックスは、ストア Web アプリケーション文書のルートの下にある、以下のストア・ディレクトリー *(storedir)* に発行されます。
  - NT `drive:¥WebSphere¥AppServer¥installedApps¥WC_Enterprise_App_`*instancename*`.ear¥wcstores.war¥`*storedir*
  - 2000 `drive:¥Program Files¥WebSphere¥AppServer¥installedApps¥WC_Enterprise_App_`*instancename*`.ear¥wcstores.war¥`*storedir*
  - AIX `/usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_`*instancename*`.ear/wcstores.war/`*storedir*
  - Solaris `/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_`*instancename*`.ear/wcstores.war/`*storedir*
  - Linux `/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_`*instancename*`.ear/wcstores.war/`*storedir*
  - 400 `/QIBM/UserData/WebASAdv4/WASinstancename/installedApps/WC_Enterprise_App_`*instancename*`.ear/wcstores.war/`*storedir*

- リソース・バンドルとプロパティー・ファイルは、アプリケーションがプロパティーを参照する以下のパスに発行されます。
  - NT drive:¥WebSphere¥AppServer¥installedApps¥WC_Enterprise_App_*instancename*.ear¥wcstores.war¥WEB-INF¥classes¥*storedir*
  - 2000 drive:¥Program Files¥WebSphere¥AppServer¥installedApps¥WC_Enterprise_App_*instancename*.ear¥wcstores.war¥WEB-INF¥classes¥*storedir*
  - AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/wcstores.war/WEB-INF/classes/*storedir*
  - Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/wcstores.war/WEB-INF/classes/*storedir*
  - Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/wcstores.war/WEB-INF/classes/*storedir*
  - 400 /QIBM/UserData/WebASAdv4/WASinstancename/installedApps/WC_Enterprise_App_*instancename*.ear/wcstores.war/WEB-INF/classes/*storedir*

ストア・アーカイブ内の XML ファイルからデータベースにストア・データをロードする

ストア・アーカイブからデータベースに XML ファイルのストア・データをロードする際、 WebSphere Commerce は以下を行います。

注: データベースにロードされるのは、db-load タイプの XML ファイルだけです。ファイル・タイプは、sarinfo.xml ファイルで指定されます。 sarinfo.xml ファイルに関する詳細は、 317 ページの『付録 C. sarinfo.xml』を参照してください。

ストア・アーカイブ内の XML ファイルの妥当性検査を行い、それをマスター XML ファイルに連結する: WebSphere Commerce は、相当する DTD ファイルを使用して XML ファイルの妥当性検査を実行します。 DTD ファイルは以下のディレクトリーにあります。

- NT drive:¥WebSphere¥CommerceServer¥xml¥sar
- 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥xml¥sar
- AIX /usr/WebSphere/CommerceServer/xml/sar
- Solaris /opt/WebSphere/CommerceServer/xml/sar
- Linux /opt/WebSphere/CommerceServer/xml/sar
- 400 /qibm/proddata/WebCommerce/xml/sar

XML ファイルが無効な場合、WebSphere Commerce はエラー・ログにエラーを書き込みます。エラー・ログ内のエラーが、WebSphere Commerce 構成ファイル instance_name.xml (デフォルトでは MaxErrorsInSarXML=1) の DevTools の部分で指定されたエラーの最大数を超えると、発行は失敗します。 WebSphere Commerce 構成ファイル instance_name.xml は、以下のディレクトリーにあります。

- NT drive:¥WebSphere¥CommerceServer¥instances¥*instancename*¥xml¥*instance_name*.xml

- 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥instances ¥*instancename*¥xml¥*instance_name*.xml
- AIX /usr/WebSphere/CommerceServer/instances/*instancename* /xml/*instance_name*.xml
- Solaris /opt/WebSphere/CommerceServer/instances/*instancename* /xml/*instance_name*.xml
- Linux /opt/WebSphere/CommerceServer/instances/*instancename* /xml/*instance_name*.xml
- 400 /QIBM/UserData/WebCommerce/instances/*instancename* /xml/*instance_name*.xml

XML ファイルの妥当性が確認された場合は、これらのファイルが *storename*master.xml という 1 つのファイルに連結されます。 *storename*master.xml ファイルは、以下のディレクトリーにあります。

- NT drive:¥WebSphere¥CommerceServer¥temp¥*instancename*¥tools¥devtools
- 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥temp¥*instancename*¥tools¥devtools
- AIX /usr/WebSphere/CommerceServer/temp/*instancename*/tools/devtools
- Solaris /opt/WebSphere/CommerceServer/temp/*instancename*/tools/devtools
- Linux /opt/WebSphere/CommerceServer/temp/*instancename*/tools/devtools
- 400 /QIBM/UserData/WebCommerce/instances/*instancename* /temp/tools/devtools

ID リゾルバーを呼び出して ID を解決する: ローダー・パッケージ・ユーティリティーの 1 つ、ID リゾルバーは、ストア・アーカイブ XML ファイル内の XML エレメントに ID を生成します。たとえば、サンプル・ストア XML ファイルで使用されている @ 別名を固有の値に置き換えます。サンプル・ストアで使用されている内部別名解決については、 313 ページの『付録 B. データの作成』を参照してください。 ID リゾルバーとその他のローダー・パッケージ・コンポーネントについては、 243 ページの『第 28 章 ストア・データのロードの概要』を参照してください。

ストア・サービスやコマンド行の発行ユーティリティーで ID リゾルバーを呼び出すときは、どの ID リゾルバー・メソッドを使用するかを指定する必要があります。 ID リゾルバーには、ID リゾルバーへの入力を入力を処理するメソッドがいくつかあります。具体的にあげると、まず、オリジナルのデータに ID がある場合にデータを扱うメソッド (更新メソッド) と、オリジナルのデータに ID がない場合のメソッド (ロード・メソッド) があります。また、一部の ID があって、一部の ID がない場合には、混合メソッドが使用されます。ストア・サービスでは、混合メソッドの使用が勧められています。ストア・サービスやコマンド行の発行メソッドで呼び出すメソッドは、 WebSphere Commerce 構成ファイル instance_name.xml で指定できます。 ID リゾルバーのメソッドに関する詳細は、 246 ページの『ID Resolve コマンド』を参照してください。

ストア・サービスやコマンド行の発行ユーティリティーでは、 ID リゾルバーで使用するカスタマイザー・ファイルも指定する必要があります。 WebSphere Commerce 構成ファイル instance_name.xml でカスタマイザー・ファイルが指定されない場合、発行コードは、デフォルト・カスタマイザー・ファイル DBConnectionCustomizer または OracleConnectionCustomizer のいずれかを使用します。カスタマイザー・ファイルは、以下のディレクトリーにあります。

Oracle

- NT drive:¥WebSphere¥AppServer¥installedApps¥WC_Enterprise_App_*instancename*.ear¥properties
- 2000 drive:ProgramFiles¥WebSphere¥AppServer¥installedApps¥WC_Enterprise_App_*instancename*.ear¥properties
- AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/properties
- Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/properties
- Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/properties
- 400 /QIBM/UserData/WebASAdv4/WASinstancename/installedApps/WC_Enterprise_App_*instancename*.ear/properties

DB2

- NT drive:¥WebSphere¥AppServer¥installedApps¥WC_Enterprise_App_*instancename*.ear¥lib¥loader¥idresgen.zip
- 2000 drive:ProgramFiles¥WebSphere¥AppServer¥installedApps¥WC_Enterprise_App_*instancename*.ear¥lib¥loader¥idresgen.zip
- AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/lib/loader/idresgen.zip
- Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/lib/loader/idresgen.zip
- Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/lib/loader/idresgen.zip
- 400 /QIBM/UserData/WebASAdv4/WASinstancename/installedApps/WC_Enterprise_App_*instancename*.ear/lib/loader/idresgen.zip

注: デフォルトでは、WebSphere Commerce 構成ファイル instance_name.xml には、このパラメーターの属性や値が指定されていません。独自のカスタマイザー・ファイルを指定したい場合は、 instance_name.xml ファイルの DevTools セクションに以下を追加する必要があります。

- IDResolverCustomizerFile="*myIDResolverCustomizerFile*"

ID リゾルバーは、*storename*master.xml とそれに対応する DTD ファイル wcs.dtd を使用します。 ID が解決されると、ID リゾルバーは、固有の ID が含まれた *storenametime_stamp* master.xml ファイルを作成します。なお、ID 解決プロセスの途中でエラーが発生した場合、ローダー・パッケージは、 error.xml ファイル、*storename*master.error.xml を作成します。

*storenametime_stamp*master.xml ファイル、および *storename*master.error.xml は、以下のディレクトリーにあります。

- NT drive:¥WebSphere¥CommerceServer¥temp¥*instancename*¥tools¥devtools
- 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥temp¥*instancename*¥tools¥devtools
- AIX /usr/WebSphere/CommerceServer/temp/*instancename*/tools/devtools
- Solaris /opt/WebSphere/CommerceServer/temp/*instancename*/tools/devtools
- Linux /opt/WebSphere/CommerceServer/temp/*instancename*/tools/devtools
- 400 /QIBM/UserData/WebCommerce/instances/*instancename*/temp/tools/devtools

ローダー・パッケージを呼び出して、解決されたマスター XML ファイルをデータベースにロードする: ローダー・パッケージは、解決された *storenametime_stamp*master.xml をデータベースにロードします。ロード・プロセスの途中でエラーが発生した場合は、ローダー・パッケージによって error.xml ファイル *storenametime_stamp* master.error.xml が作成されます。

ローダー・パッケージについて詳しくは、 243 ページの『第 28 章 ストア・データのロードの概要』を参照してください。

ストア・サービスやコマンド行の発行ユーティリティーでローダー・パッケージを呼び出すときは、どのローダー・メソッドを使用するかを指定する必要があります。ストア・サービスでは、以下のローダー・メソッドが使用されます。

- SQL インポート
- インポート
- ロード

ストア・サービスやコマンド行の発行メソッドで呼び出すメソッドは、 WebSphere Commerce 構成ファイル instance_name.xml で指定できます。

- SQL インポート: このメソッドでは、Java Database Connectivity (JDBC) を使用してデータの挿入と更新を行います。これは、操作の点で最も柔軟性の高いメソッドですが、大量のデータを少ないテーブルにインポートするときには、最も処理に時間がかかるメソッドでもあります。このメソッドでは、セル・レベルの更新が可能です。 SQL インポートの使用は推奨されています。

- インポート: これは、DB2 にもともと組み込まれているインポート機能を使用するメソッドで、平均的な処理速度と柔軟性でセル・レベルの更新を行うことができます。
- ロード: これは、RDBMS にもともと組み込まれている機能 (DB2 Load または SQLLoad) を使用するメソッドで、大量のデータを少ないテーブルにロードするときには、最も処理が速いメソッドです。ただし、ステージング・サーバーへの発行を行うときには、このロード・メソッドは使用できません。

ロード・コマンドのメソッドについては、254 ページの『Load コマンド』を参照してください。

ストア・サービスやコマンド行の発行ユーティリティーでは、ローダーで使用するカスタマイザー・ファイルも指定する必要があります。 WebSphere Commerce 構成ファイル `instance_name.xml` でカスタマイザー・ファイルが指定されない場合、発行コードは、デフォルト・カスタマイザー・ファイル MassLoadCustomizer を使用します。

注: デフォルトでは、WebSphere Commerce 構成ファイル `instance_name.xml` には、このパラメーターの属性や値が指定されていません。独自のカスタマイザー・ファイルを指定したい場合は、 `instance_name.xml` ファイルの DevTools セクションに以下を追加する必要があります。

- `LoaderCustomizerFile="myLoaderCustomizerFile"`

ストア構成ファイルのアンパック

Business サンプル・ストア・アーカイブ ToolTech および NewFashion のアーカイブには、以下のファイルも含まれています。

- `tools_properties.zip`
- `tools_xml.zip`
- `runtime_xml.zip`

これらのファイルは、`sarinfo.xml` ファイルに登録され、ストアを構成するためにストア・サービスで使用されます。発行プロセスでは、これらのファイルがアンパックされ、 WebSphere Commerce 構成ファイル `instance_name.xml` で指定されたディレクトリーにコピーされます。

注: これらのファイルは、変更したり、削除したり、他のストアにコピーしたりしないでください。

商品セット、アカウント、および契約を発行するコマンドを呼び出す

一部のストア・データ資産 (契約、アカウント、および商品セット) は、ローダー・パッケージではロードできません。そのため、発行プロセスでは、これらの資産を WebSphere Commerce Server に発行するための相当するコマンドも呼び出されます。次のようなコマンドが呼び出されます。

- ProductSetPublish— 商品セット・データを商品セット・テーブルに発行します。
- AccountImport — ストア・アーカイブ内の `businessaccount.xml` ファイルからアカウントを作成します。
- ContractImportApprovedVersion — ストア・アーカイブ内の `contract.xml` ファイルから契約をインポートします。

エラー処理

発行プロセスの資産発行のフェーズでエラーが発生した場合は、発行ログ (『ログ・ファイルの発行』を参照) か、ストア・サービスの「発行の要約」ページで、エラー・メッセージを確認できます。

支払いの構成

発行プロセスの最後のステップは、支払いの構成です。 WebSphere Commerce では、IBM Payment Manager がサポートされています。支払い処理のメソッドとして Payment Manager を使用する計画がある場合は、 123 ページの『第 13 章 支払資産』で説明されている方法に従って支払い XML ファイルを作成する必要があります。発行されるストア・アーカイブに支払い XML ファイルが組み込まれている場合、 WebSphere Commerce は、発行のプロセスで、以下の支払いの構成を完了させます。

- マーチャントの作成
- アカウントの作成 (オフライン・カセット専用)
- 3 つのブランドの作成 (オフライン・カセット専用)
- ユーザー権限の割り当て

エラー処理

発行プロセスの支払い構成のフェーズでエラーが発生した場合は、発行ログ (『ログ・ファイルの発行』を参照) か、ストア・サービスの「発行の要約」ページで、エラー・メッセージを確認できます。

ログ・ファイルの発行

発行プロセスの資産発行のフェーズで発生したエラーはすべて、以下のログおよびトレース・ファイルに書き込まれます。

- `messages.txt`: 発行プロセスのローダー・パッケージ部分で生成されたエラー・メッセージが含まれています。発行が失敗した場合は、まず最初にこのログを調べてください。これらのエラー・メッセージに示される行番号や列番号は、一時 `master.xml` ファイル、つまり *storename*`master.xml` または *storenametime_stamp*`master.xml` を参照するものです。
- `trace.txt`: 発行プロセスのローダー・パッケージ部分および ID リゾルバー部分に関するトレース情報が含まれています。 trace.txt は、デフォルトではオフになっています。
- `ecmsg_`*hostname_timestamp*`.log`: WebSphere Commerce Server から戻されるすべての実行エラー・メッセージが記録されているログです。
- `wcs.log file`: WebSphere Application Server で稼働するすべてのアプリケーション (発行ユーティリティーを含む) から通常のコンソールに送られた出力が含まれています。

これらのログ・ファイルは、以下のディレクトリーにあります。

- NT `drive:¥WebSphere¥CommerceServer¥instances¥`*instancename*`¥logs`
- 2000 `drive:¥Program Files¥WebSphere¥CommerceServer¥`*instancename*`¥logs`
- AIX `/usr/WebSphere/CommerceServer/instances/`*instancename*`/logs`

- Solaris /opt/WebSphere/CommerceServer/instances/*instancename*/logs
- Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/wcstores.war/WEB-INF/classes/*storedir*
- 400 /QIBM/UserData/WebCommerce/instances/*instancename*/logs

trace.txt ファイルおよび messages.txt ログ・ファイルの構成 (つまり、ログ・レベルその他のオプションの調整) を行うには、以下のファイルを編集してください。

- NT drive:¥WebSphere¥CommerceServer¥xml¥loader¥WCALoggerConfig.xml
- 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥xml¥loader¥WCALoggerConfig.xml
- AIX /usr/WebSphere/CommerceServer/xml/loader/WCALoggerConfig.xml
- Solaris /opt/WebSphere/CommerceServer/xml/loader/WCALoggerConfig.xml
- Linux /opt/WebSphere/CommerceServer/xml/loader/WCALoggerConfig.xml
- 400 /QIBM/UserData/WebCommerce/instances/instancename/xml/WCALoggerConfig.xml

第 27 章 ストアフロント資産とストア構成ファイルの発行

ストアフロント資産、HTML および JSP ファイル、プロパティー・ファイルとリソース・バンドル、およびストア・ページを形成しているイメージやグラフィックスの発行は、ストアの機能的な面を作成するプロセスに含まれます。ストアフロントの発行は、 227 ページの『第 26 章 ストア全体の発行』で説明されているストア全体の発行オプションの一環として、ストア・サービスまたはコマンド行から行うこともできますし、単純に、WebSphere Commerce Server 上の指定されたロケーションに資産をコピーすることによっても行えます。

サンプル・ストア Business ToolTech および NewFashion に含まれている JSP ファイルを発行する場合は、そのストア・アーカイブに属しているストア構成ファイルも発行する必要があります。どちらのストアにも、次のストア構成ファイルが含まれています。

- `tools_properties.zip`
- `tools_xml.zip`
- `runtime_xml.zip`

ストア・サービスやコマンド行を使用してストア全体を発行する場合 (すべての発行オプションを選択) は、ストア構成ファイルも発行されますが、WebSphere Commerce Server にストアフロント資産をコピーしてストアを発行する場合は、ストア構成ファイルも WebSphere Commerce Server にコピーする必要があります。

ストア・サービスやコマンド行を使用した、ストアフロント資産およびストア構成ファイルの発行

ストアフロント資産とストア構成ファイルの発行は、ストア・サービスを使用して、あるいはコマンド行から発行ユーティリティーを使用して行うことができます。

注: ストア・サービスや発行ユーティリティーを使用してストア構成ファイルを発行する場合は、 Web 資産やデータ資産を含むすべてのストア資産を発行する必要があります。ストア構成ファイルだけを発行することはできません。

ストア・サービスやコマンド行を使用してストアフロント資産とストア構成ファイルを発行するためには、ストアフロント資産とストア構成ファイルがストア・アーカイブ・フォーマットにパッケージされていなければなりません。ストアフロント資産をストア・アーカイブとしてパッケージする処理については、 215 ページの『第 6 部 ストアのパッケージ化』を参照してください。

ストア・サービスやコマンド行の発行ユーティリティーでは、ストア・アーカイブ内のすべてのタイプの資産を発行する (ストアフロント資産、ストア・データ資産、およびリソース・バンドルを含む) か、特定のタイプの資産だけを発行するかを選択できます。ストア・サービスやコマンド行を使用した資産発行の詳細な手順については、 WebSphere Commerce オンライン・ヘルプを参照してください。

WebSphere Commerce Server へのコピーによるストアフロント資産およびストア構成ファイルの発行

資産をストア・アーカイブにパッケージしないほうが好ましい場合は、ストアフロント資産を直接 WebSphere Commerce Server にコピーする方法でも、資産を発行できます。 Web 資産 (HTML、JSP ファイル、イメージおよびグラフィックス) は、Web アプリケーション文書のルートに発行してください。リソース・バンドルやプロパティー・ファイルは、アプリケーションがプロパティーを参照するパスに発行してください。

ストアフロント資産とストア構成ファイルを WebSphere Commerce Server にコピーするには、次のようにします。

1. JSP ファイル、HTML、組み込みファイル、イメージ、およびグラフィックスを、以下のストア・ディレクトリー (*storedir*)、ストア Web アプリケーション文書のルートにコピーしてください。
   - NT drive:¥WebSphere¥AppServer¥installedApps¥WC_Enterprise_App_*instancename*.ear¥wcstores.war¥*storedir*
   - 2000 drive:¥Program Files¥WebSphere¥AppServer¥installedApps¥WC_Enterprise_App_*instancename*.ear¥wcstores.war¥*storedir*
   - AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/wcstores.war/*storedir*
   - Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/wcstores.war/*storedir*
   - Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/wcstores.war/*storedir*
   - 400 /QIBM/UserData/WebASAdv4/WASinstancename/installedApps/WC_Enterprise_App_*instancename*.ear/wcstores.war/*storedir*
2. リソース・バンドルとプロパティー・ファイルを、アプリケーションがプロパティーを参照する以下のパスにコピーしてください。
   - NT drive:¥WebSphere¥AppServer¥installedApps¥WC_Enterprise_App_*instancename*.ear¥wcstores.war¥WEB-INF¥classes¥*storedir*
   - 2000 drive:¥Program Files¥WebSphere¥AppServer¥installedApps¥WC_Enterprise_App_*instancename*.ear¥wcstores.war¥WEB-INF¥classes¥*storedir*
   - AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/wcstores.war/WEB-INF/classes/*storedir*
   - Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/wcstores.war/WEB-INF/classes/*storedir*
   - Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/wcstores.war/WEB-INF/classes/*storedir*
   - 400 /QIBM/UserData/WebASAdv4/WASinstancename/installedApps/WC_Enterprise_App_*instancename*.ear/wcstores.war/WEB-INF/classes/*storedir*

3. ストア構成ファイルを、 WebSphere Commerce 構成ファイル instance_name.xml で定義されているロケーションにコピーしてください。このファイルは、以下のディレクトリーにあります。

   - NT drive:¥WebSphere¥CommerceServer¥instances¥*instancename*¥xml¥*instance_name*.xml
   - 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥instances¥*instancename*¥xml¥*instance_name*.xml
   - AIX /usr/WebSphere/CommerceServer/instances/*instancename*/xml/*instance_name*.xml
   - Solaris /opt/WebSphere/CommerceServer/instances/*instancename*/xml/*instance_name*.xml
   - Linux /opt/WebSphere/CommerceServer/instances/*instancename*/xml/*instance_name*.xml
   - 400 /QIBM/UserData/WebCommerce/instances/*instancename*/xml/*instance_name*.xml

   デフォルトでは、ストア構成ファイルは以下のロケーションに置かれます。

   - runtime_xml.zip は、StoresXMLPath にコピーされます。デフォルトでは、このパスは次のようになっています。
     - NT drive:¥WebSphere¥CommerceServer¥instances¥*instancename*¥xml¥*instance_name*.xml¥wcstores.war¥xml
     - 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥instances¥*instancename*¥xml¥*instance_name*.xml¥wcstores.war¥xml
     - AIX /usr/WebSphere/CommerceServer/instances/*instancename*/xml/*instance_name*.xml/wcstores.war/xml
     - Solaris /opt/WebSphere/CommerceServer/instances/*instancename*/xml/*instance_name*.xml/wcstores.war/xml
     - Linux /opt/WebSphere/CommerceServer/instances/*instancename*/xml/*instance_name*.xml/wcstores.war/xml
     - 400 /QIBM/UserData/WebCommerce/instances/*instancename*/xml/*instance_name*.xml/wcstores.war/xml
   - tools_properties.zip は、ToolsStoresPropertiesPath にコピーされます。デフォルトでは、このパスは次のようになっています。
     - NT drive:¥WebSphere¥CommerceServer¥instances¥*instancename*¥xml¥*instance_name*.xml¥properties¥tools¥stores
     - 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥instances¥*instancename*¥xml¥*instance_name*.xml¥properties¥tools¥stores

- AIX /usr/WebSphere/CommerceServer/instances/*instancename*/xml/*instance_name*.xml/properties/tools/stores
- Solaris /opt/WebSphere/CommerceServer/instances/*instancename*/xml/*instance_name*.xml/properties/tools/stores
- Linux /opt/WebSphere/CommerceServer/instances/*instancename*/xml/*instance_name*.xml/properties/tools/stores
- 400 /QIBM/UserData/WebCommerce/instances/*instancename*/xml/*instance_name*.xml/properties/tools/stores

- tools_xml.zip は、ToolsStoresXMLPath にコピーされます。デフォルトでは、このパスは次のようになっています。
  - NT drive:¥WebSphere¥CommerceServer¥instances¥*instancename*¥xml¥*instance_name*.xml¥xml¥tools¥stores
  - 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥instances¥*instancename*¥xml¥*instance_name*.xml¥xml¥tools¥stores
  - AIX /usr/WebSphere/CommerceServer/instances/*instancename*/xml/*instance_name*.xml/xml/tools/stores
  - Solaris /opt/WebSphere/CommerceServer/instances/*instancename*/xml/*instance_name*.xml/xml/tools/stores
  - Linux /opt/WebSphere/CommerceServer/instances/*instancename*/xml/*instance_name*.xml/xml/tools/stores
  - 400 /QIBM/UserData/WebCommerce/instances/*instancename*/xml/*instance_name*.xml/xml/tools/stores

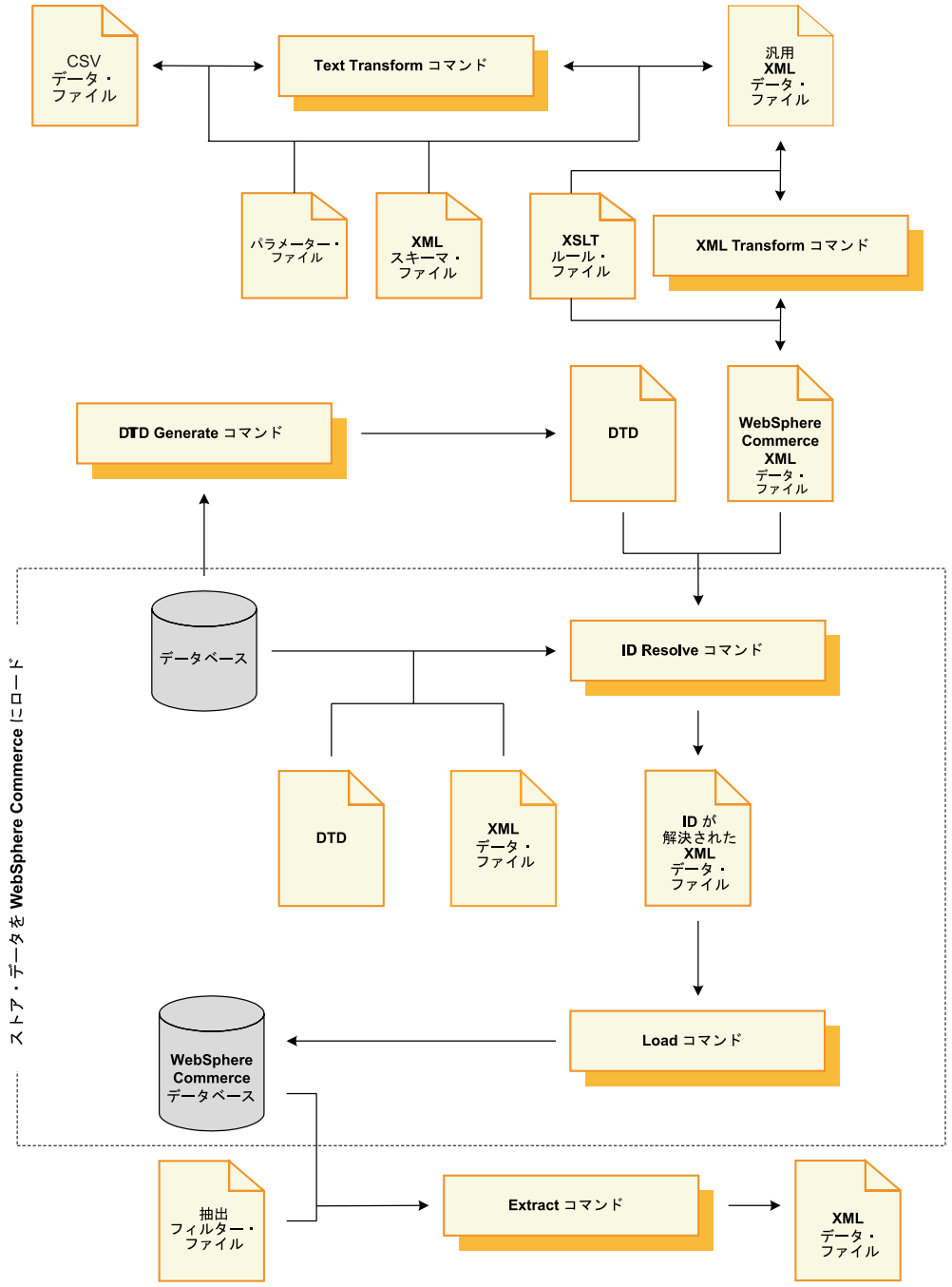
第 28 章 ストア・データのロードの概要

ストア・データを作成したなら、それをストア・アーカイブとしてパッケージし、ストア・サービスを使用して発行できます。あるいは、WebSphere Commerce Catalog Manager を使用して、 WebSphere Commerce Server データベースに直接ロードすることもできます。 WebSphere Commerce データ・グループへのロード・プロセスについては、 279 ページの 『第 29 章 WebSphere Commerce データ・グループのロード』および 281 ページの 『データ・グループのロード』を参照してください。

Catalog Manager には、データの準備とストアへのロードに使用できる、 6 つのコマンド・ユーティリティー (ここでは集合的に「ローダー・パッケージ」と呼びます) と、 3 つの関連する管理用ツールがあります。これらのコマンドやツールでは、情報の管理に Extensible Markup Language (XML) データ・ファイルを使用しています。

WebSphere Commerce でのデータ・ロードについて

次の図は、ローダー・パッケージ・コマンドを使用して実行できる、データの準備、ロード、および抽出のプロセスを示しています。



点線で囲まれた部分に、2 つのプロセスが示されていることに注目してください。これが、ストア・データを WebSphere Commerce Server データベースにロードする際に最も一般的に使用されるプロセス、つまり、ID の解決とデータのロードです。この章では、これらのプロセスに焦点を合わせます。

WebSphere Commerce Server データベースにロードするデータの準備については、39 ページの 『第 4 部 ストア・データの開発』を参照してください。

WebSphere Commerce Server データベースにデータをロードする際には、次の 2 つのローダー・パッケージ・コマンド・ユーティリティーが一般的に使用されます。

- **ID Resolve コマンド**

  WebSphere Commerce Server データベースにロードされる XML データは、ターゲット・データベース・スキーマに直接マップされるため、すべての XML 要素には固有のキーまたは ID が必要です。 ID リゾルバーは、これらの XML 要素に固有の ID を生成します。

  このコマンドの使用については、 246 ページの 『ID Resolve コマンド』および 271 ページの 『ID 解決の例』を参照してください。

- **Load コマンド**

  ローダーは、データベースにデータをロードするための入力として、妥当かつ整形式の XML ファイルを使用します。 XML 文書の要素はデータベースのテーブル名にマップされ、要素属性は列にマップされます。

  このコマンドの使用については、254 ページの 『Load コマンド』および 277 ページの 『データ・ロードの例』を参照してください。

この章では、主にこれらのコマンドに焦点を合わせます。

データの管理には、以下のローダー・パッケージ・コマンド・ユーティリティーも使用できます。

- **DTD Generate コマンド**

  DTD ジェネレーターは、 XML データをロードするターゲット・データベースのテーブルと列を説明する、文書型定義 (DTD) を生成します。加えて、DTD ジェネレーターでは、データベースの XML スキーマも生成できます。

- **Text Transform コマンド**

  テキスト変換ツールは、文字区切り変数フォーマットと XML データ・フォーマットの間でのデータ変換を行います。

- **XML Transform コマンド**

  XML 変換プログラムは、XML 文書内のデータを代替の XML フォーマットに変換します。変換のためのマッピング・ルールの定義には、Extensible Stylesheet Language (XSL) が使用されます。

- **Extract コマンド**

  抽出プログラムは、データベースに対するクエリーを用い、フィルター・ファイルを使用しながら、選択されたデータのサブセットをデータベースから XML 文書に抽出します。

この章は、これらのコマンドを主要な論点としていません。これらのコマンドに関する詳細は、最新版の *IBM WebSphere Commerce 5.4 Catalog Manager ユーザーズ・ガイド* を参照してください。

WebSphere Commerce Catalog Manager には、データ管理機能の管理を支援するための以下のツールも組み込まれています。

- **テキスト変換ツール**

  テキスト変換ツールは、文字区切り変数フォーマットと XML データ・フォーマットの間でデータ変換を処理するために必要な情報を処理および作成するのに役立ちます。

- **XSL エディター**

  XSL エディターは、XML 変換プログラムで使用できる、 XSL ファイルを編集するための視覚的なインターフェースを提供します。 XML フォーマットへのデータ変換を行うためのマッピング規則を定義するときは、XSL エディターを使用して、ソース DTD の要素からターゲット DTD の要素へのアソシエーションを確立します。

- **Web エディター**

  Web エディターを使用することにより、 Web ブラウザーを通してデータベース内のデータを作成、変更、または削除できるようになります。

この章は、これらのツールを主要な論点としていません。これらのツールに関する詳細は、最新版の *IBM WebSphere Commerce 5.4 Catalog Manager ユーザーズ・ガイド* を参照してください。

ストア・データをロードするためのローダー・パッケージ・コマンド

ID Resolve コマンド

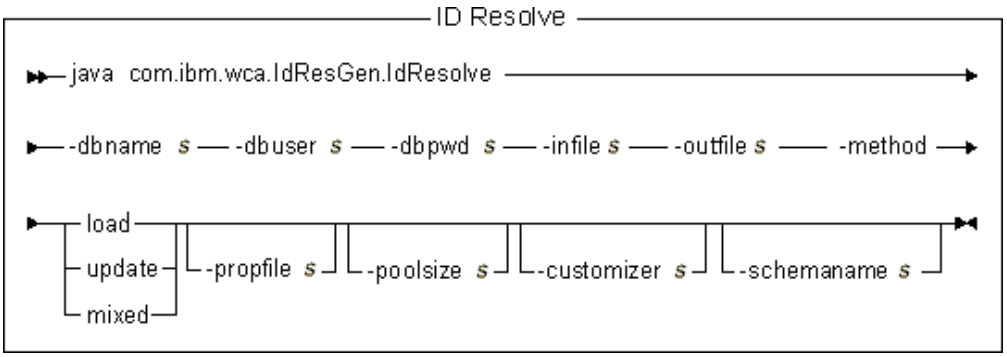
このコマンドは、XML データ要素をデータベースにロードするために必要な、XML データ要素の ID を生成します。必要な ID がソースの XML データにすでに含まれている場合は、 ID リゾルバーを実行する必要はありません。

WebSphere Commerce データベース・スキーマは、テーブル間の様々な関係を表すのに用いられる、テーブル内の 1 次鍵や外部キーを定義します。この理由から、WebSphere Commerce XML スキーマの要素には、固有のキーか ID が含まれていなければなりません。 WebSphere Commerce Server データベースでは、 ID を解決する必要のあるテーブルが KEYS および SUBKEYS テーブルで定義されます。これらのテーブルは、Websphere Commerce では「1 次テーブル」と呼ばれています。

注: KEYS または SUBKEYS にないテーブルの ID を解決することが必要になった場合には、 ID リゾルバーを実行する前に、そのテーブルを SUBKEYS テーブルに追加してください。

WebSphere Commerce XML スキーマは、データベース間やデータベース・インスタンス間で移植可能なものとするよう意図されているため、 ID は通常内部別名で表されます。データを任意の WebSphere Commerce Server データベースにロードできるようにするためには、これらの別名を有効なキーに解決する必要があります。詳細は、313 ページの 『付録 B. データの作成』を参照してください。

Windows AIX Solaris Linux



注: 上の図は、主にコマンド・パラメーターを参照するための図です。このコマンドに備えられており、 270 ページの 『ローダー・パッケージ・コマンドおよびスクリプトの使用』でリストされているコマンド・ファイルまたはスクリプトは、実際の Java コマンドのラッパーとして機能し、同じパラメーターを受け入れます。それゆえ、Java コマンドを直接呼び出すのではなく、コマンド・ファイルまたはスクリプトを使用することをお勧めします。

パラメーター値:

-dbname
ターゲット・データベースの名前

-dbuser
データベースに接続しているユーザーの名前

-dbpwd
データベースに接続しているユーザーのパスワード

-infile テーブル・レコード・ディスクリプターを含む XML 文書の名前

-outfile
生成される出力 XML ファイルの名前 (このファイルをローダーへの入力として使用できます)

-method
入力ファイルの処理で使用するメソッド

- ロード・メソッドは、ファイル内の***すべての*** レコードがデータベースに***存在しない*** 場合に、入力ファイルを処理するために使用します。
- 更新メソッドは、ファイル内の***すべての*** レコードがデータベースに***存在する*** 場合に、入力ファイルを処理するために使用します。
- 混合メソッドは、ファイル内の***一部のレコードだけ*** がデータベース内に***存在している*** 場合に、入力ファイルを処理するために使用します。

デフォルトの方式はロードです。

-propfile
「名前=値」の形式で Java プロパティーを指定したテキスト・ファイル。このファイルは、外部鍵 ID 検索のための索引列名、および主テーブル (CATEGORY および PRODUCT) クエリーの選択述部を定義するために使用されます。このファイルの中で、 ID を含まない固有索引が定義されてい

るテーブルについては、エントリーを省略できます。このパラメーターはオプションです。デフォルトのファイルは IdResolveKeys.properties です。このファイルの拡張子は .properties ですが、値を指定するときは、拡張子は使用しないでください。

-poolsize

予約する ID の数。このパラメーターはオプションです。デフォルトの数は 50 です。

-customizer

使用するカスタマイザー・プロパティー・ファイルの名前。このパラメーターはオプションです。DB2ConnectionCustomizer.properties は、デフォルトのファイルです。次の例のいずれかに示されているようにして、カスタマイザー・プロパティー・ファイルを指定できます。

```
-customizer d:¥WebSphere¥CommerceServer¥prop¥idres.properties
```

```
-customizer d:¥WebSphere¥CommerceServer¥prop¥idres
```

現行ディレクトリーにこのファイルがある場合は、次の例に示すようにして同じファイルを指定できます。

```
-customizer idres.properties
```

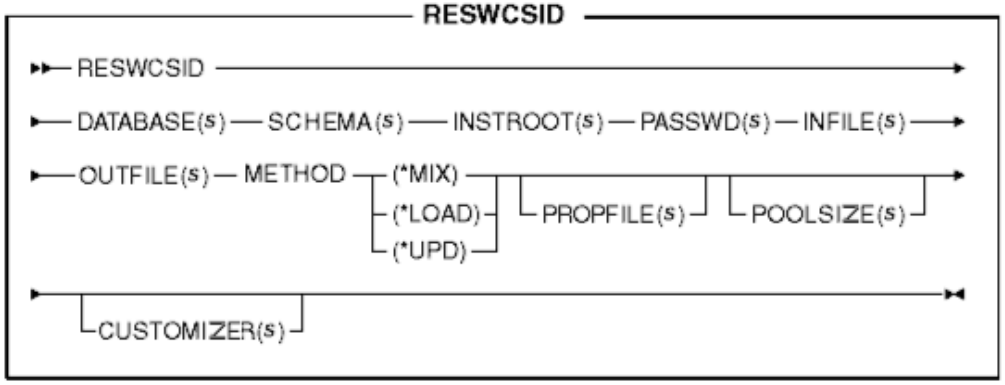
クラスパス・システム環境変数で指定されたディレクトリーにこのファイルがある場合は、次の例に示すようにして同じファイルを指定できます。

```
-customizer idres
```

-schemaname

ターゲット・データベース・スキーマの名前。このパラメーターはオプションです。デフォルトでは、データベース・ユーザーの名前が使用されます。

400



パラメーター値:

DATABASE

ターゲット・データベースの名前 (リレーショナル・データベース・ディレクトリーで表示されるもの)。

SCHEMA

ターゲット・データベース・スキーマの名前。これは、インスタンス名と同じです。

INSTROOT

WebSphere Commerce インスタンスのルート・パスの絶対パス名 (/QIBM/UserData/WebCommerce/instances/*instance_name* など)。

PASSWD

WebSphere Commerce インスタンスのパスワード

INFILE

テーブル・レコード・ディスクリプターを含むに XML 文書の名前

OUTFILE

生成される出力 XML ファイルの名前 (このファイルをローダーへの入力として使用できます)

METHOD

入力ファイルの処理で使用するメソッド

- ロード・メソッド (*LOAD) は、ファイル内の***すべての*** レコードがデータベースに***存在しない*** 場合に、入力ファイルを処理するために使用します。
- 更新メソッド (*UPD) は、ファイル内の***すべての*** レコードがデータベースに***存在する*** 場合に、入力ファイルを処理するために使用します。
- 混合メソッド (*MIX) は、ファイル内の***一部のレコードだけ*** がデータベース内に***存在している*** 場合に、入力ファイルを処理するために使用します。

PROPFILE

「名前=値」の形式で Java プロパティーを指定したテキスト・ファイル。このファイルは、外部鍵 ID 検索のための索引列名、および主テーブル (CATEGORY および PRODUCT) クエリーの選択述部を定義するために使用されます。このファイルの中で、 ID を含まない固有索引が定義されているテーブルについては、エントリーを省略できます。このパラメーターはオプションです。デフォルトのファイルは IdResolveKeys.properties です。このファイルの拡張子は .properties ですが、値を指定するときは、拡張子は使用しないでください。

POOLSIZE

予約する ID の数。このパラメーターはオプションです。デフォルトの数は 50 です。

CUSTOMIZER

使用するカスタマイザー・プロパティー・ファイルの名前。このパラメーターはオプションです。デフォルトのファイルは ISeries_RESWCSID_Customizer.properties です。次の例のいずれかに示されているようにして、カスタマイザー・プロパティー・ファイルを指定できます。

```
CUSTOMIZER(/wc/prop/idres.properties)
CUSTOMIZER(/wc/prop/idres)
```

現行ディレクトリーにこのファイルがある場合は、次の例に示すようにして同じファイルを指定できます。

```
CUSTOMIZER(idres.properties)
```

クラスパス・システム環境変数で指定されたディレクトリーにこのファイルがある場合は、次の例に示すようにして同じファイルを指定できます。

```
CUSTOMIZER(idres)
```

解決のテクニック:

ID リゾルバーは、プロパティー・ファイルが使用されているかどうかに応じ、以下の 2 つか 3 つのテクニックを組み合わせて ID を解決します。

- **内部別名解決**

  内部別名 ID 解決法では、ソース XML 文書のユニーク鍵 (ID) の代わりに別名が使用されます。これで別名は、その要素を参照するために、XML ファイル内の他の場所で使用できます。

  内部別名は、XML ファイルを通じて一貫して使用される必要があります。たとえば、住所録 ID ADDRBOOK_ID の別名が @addrbook_1 だとすると、そのファイル中のその ID に対するすべての外部鍵参照では、 @addrbook_1 を使用しなければなりません。

  別名は、特定の XML ファイルにおける一時的なものであることに注意してください。別名は保存されません。また、同じ別名を再び設定しない限り、別名は別の XML ファイルでは使用できません。ただし、ストア・サービスの発行においては、すべてのデータを通じて別名解決が実行されるように XML ファイルが連結されます。

- **固有索引解決**

  ID リゾルバーは、データベース・スキーマを分析して、要件を満たす固有索引があるかどうかも判別できます。 ID リゾルバーが固有索引を探すのは、分析されるテーブルのエントリーがプロパティー・ファイル内にない場合や、プロパティー・ファイルそのものがない場合だけです。これらの条件が整うと、固有索引のチェックが実行されます。固有索引は、それが存在しており、テーブルの 1 次鍵を含んでいなければ、有効と見なされます。

- **プロパティー・ファイル指定**

  ID リゾルバーでは、1 次行の ID が必要なテーブルの検索にどの基本エントリーの列を使用するかを記述するために、代替の Java プロパティー・ファイルを使用します。

WebSphere Commerce に組み込まれているサンプル・ストア・アーカイブでは、内部別名 ID 解決法が使用されています。この方法を使用すると、ストア・アーカイブの移植が可能になります。固有索引 ID 解決やプロパティー・ファイル指定のテクニックを使用した場合でも移植は可能ですが、これらの索引はデータベース上でユーザーに公開されているため、ユーザーによって変更される可能性があります。ユーザーが索引値を変更すると、その値は基本テーブルと外部テーブルの両方で変更しなければなりません。その点で、内部別名 ID 解決テクニックの場合、別名の値にアクセスできるのは作成者だけです。ストア・サービスやローダー・パッケージを使用して発行が行われる際、ID リゾルバーは別名を固有な値に置き換えます。データがロードされると、別名はユーザーからは認識されません。詳細は、313 ページの 『付録 B. データの作成』を参照してください。

ID リゾルバーは以下のプロセスを使用します。

1. 入力 XML データに、すでにハード・コード ID (たとえば、″12345″) がある基本テーブルの要素が含まれている場合、ID リゾルバーは、その要素に新しい ID を作成しません。
2. 入力 XML データに、ID のない基本テーブルのエントリーが含まれている場合、 ID リゾルバーは、データベースを調べてこの要素の行がすでに存在するかどうかを確認します。

   データベースでの要素の検索には、ユニーク鍵を作成するために使用される、要素内の他の列が必要です。これらの他の列は、プロパティー・ファイルで指定できます。あるいは、ID リゾルバーに使用する列を決定させることも可能です。

   - プロパティー・ファイルが使用されていて、そのプロパティー・ファイル内に分析するテーブルのエントリーがある場合、 ID リゾルバーは、プロパティー・ファイルで指定された列を使用してユニーク鍵を作成します。
   - 使用されているプロパティー・ファイルがないか、分析されるテーブルのエントリーがプロパティー・ファイルにない場合、ID リゾルバーは、固有索引解決法を使用します。

     固有索引解決法では、ID を見つける手段として、テーブル上の指定された固有索引のうち、任意のものが使用されます。たとえば、CATALOG テーブルでは MEMBER_ID と IDENTIFIER が固有索引になるため、これを CATALOGDSC の外部鍵 CATALOG_ID の解決点として用いることができます。

   同じユニーク鍵を持つ行が存在する場合、その要素はすでにデータベース内に存在しているものと判断されます。そのような行が存在しなければ、要素は新しいデータと見なされます。
3. 要素がデータベースに行としてすでに存在する場合は、その ID が取り出され、後で使用できるように保存されます。要素がデータベースにない場合は、ID リゾルバーにより、 KEYS テーブルか SUBKEYS テーブルの使用可能な値を使用して新しい ID が生成されます。
4. XML 文書で要素に内部別名 (たとえば、″@store_id_1″ など) が指定されている場合は、同じ内部別名を使用して後で ID を検索できるよう、その別名と ID が関連付けられます。
5. 後で XML 文書内の要素が基本テーブルの要素を参照する必要が生じた場合は、内部別名 (基本テーブルの要素が間違っていた場合 (たとえば、″@store_id_1″)) か検索列の値 (要素が間違っていなかった場合 (たとえば、″@WC2001@100″)) を使用して、この参照が行われます。どちらの場合でも、指定された値を使用して実際の ID が検索され、検出された ID で値が置き換えられます。
6. 出力 XML 文書が生成される時点では、すべての基本テーブルの要素が実際の ID を持つようになり、それらの基本テーブル要素を参照するすべての要素は、上で言及した内部別名や検索列の値ではなく、実際の ID を使用して参照を行うようになります。これが、XML 文書が完全に解決された状態です。

処理に関する推奨事項:

ID Resolve コマンドを使用すると、入力ファイルを処理するためのロード、更新、または混合メソッドを選択できます。

ロード方式の選択:

ID リゾルバーのロード方式は、データベースにロードされるすべての新規レコードの新規 ID を生成するために使用されます。

注: ID リゾルバーのロード方式を指定する場合、入力ファイル内のレコードが事前にデータベース内に存在するべきではありません。 ID リゾルバーとともにロード方式を使用し、ソース XML ファイル内のレコードがすでにターゲット・データベースに存在していると、ローダーはデータのロード時にエラーを生成します。 ID リゾルバーは ID 解決時に XML ファイル内のレコードに新規の基本キーを割り当てます。ただし、データベースにデータをロードする際、エラーが生成されます。ローダーは重複するレコードを処理する時点でも停止しませんが、エラーを報告し、重複するレコードはデータベースにロードしません。

以下の例を使用すると、データベースに対して新しいデータ要素の ID が生成されます。

- Windows

```
idresgen -dbname db -dbuser user -dbpwd pwd -infile input.xml
-outfile output.xml -method load -customizer customizer -schemaname mall
```

- AIX Solaris Linux

```
./idresgen.sh -dbname db -dbuser user -dbpwd pwd -infile input.xml
-outfile output.xml -method load -customizer customizer -schemaname mall
```

- 400

```
QWEBCOMM/RESWCSID DATABASE(DATABASE_NAME) SCHEMA(MALL)
INSTROOT(/QIBM/UserData/WebCommerce/instances/mser)
PASSWD(mypassword) INFILE(input.xml) OUTFILE(output.xml)
METHOD(*LOAD)
```

更新方式の選択:
ID リゾルバーの更新方式を指定する場合、入力ファイル内のレコードは事前にデータベース内に存在している必要があります。 ID リゾルバーは、データベース内の ID を探し出します。データベースにレコードが存在しない場合、 ID リゾルバーはそのレコードの ID を解決することができず、エラーが発生したと指摘します。以下の例を使用すると、データベースにすでに存在するデータ要素の ID が探し出されます。

- Windows

```
idresgen -dbname db -dbuser user -dbpwd pwd -infile input.xml
-outfile output.xml -method update -customizer customizer -schemaname mall
```

- AIX Solaris Linux

```
./idresgen.sh -dbname db -dbuser user -dbpwd pwd -infile input.xml
-outfile output.xml -method update -customizer customizer -schemaname mall
```

- 400

```
QWEBCOMM/RESWCSID DATABASE(DATABASE_NAME) SCHEMA(MALL)
INSTROOT(/QIBM/UserData/WebCommerce/instances/mser) PASSWD(mypassword)
INFILE(input.xml) OUTFILE(output.xml) METHOD(*UPD)
```

混合方式の選択:
入力データ・ファイルに、すでにデータベースに存在するレコードと新規レコードが含まれる場合、混合方式を使って ID リゾルバーを実行しなければなりません。この方式を使うと、ID リゾルバーはレコードがデータベースに存在しない場合にのみ、レコードの新規 ID を作成します。新規 ID が作成されない場合に、データベースから既存の ID を取得します。以下の例を使用すると、新規データの ID が生成され、データベースにすでに存在するデータ要素の ID が探し出されます。

- Windows

```
idresgen -dbname db -dbuser user -dbpwd pwd -infile input.xml
-outfile output.xml -method mixed -customizer customizer -schemaname mall
```

- AIX Solaris Linux

```
./idresgen.sh -dbname db -dbuser user -dbpwd pwd -infile input.xml
-outfile output.xml -method mixed -customizer customizer -schemaname mall
```

- 400

```
QWEBCOMM/RESWCSID DATABASE(DATABASE_NAME) SCHEMA(MALL)
INSTROOT(/QIBM/UserData/WebCommerce/instances/mser) PASSWD(mypassword)
INFILE(input.xml) OUTFILE(output.xml) METHOD(*MIX)
```

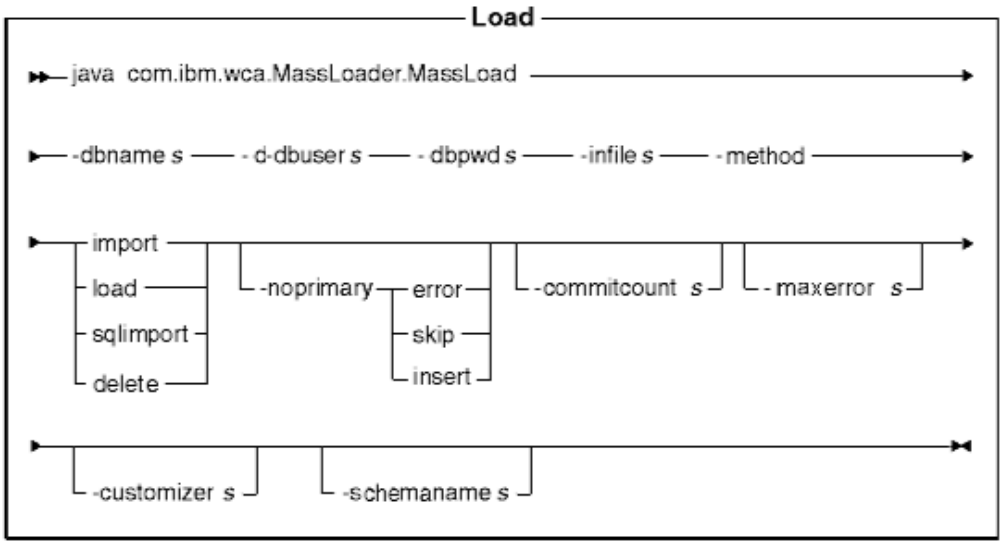
注: 混合方式は、ストア・サービスで推奨される方式です。

このコマンドを実行するために使用するファイルの設定とカスタマイズの詳細ついては、 *IBM WebSphere Commerce 5.4 Catalog Manager ユーザーズ・ガイド* の最新バージョンを参照してください。

Load コマンド

このコマンドは、XML 入力ファイルをターゲット・データベースにロードします。

Windows AIX Solaris Linux



注: 上の図は、主にコマンド・パラメーターを参照するための図です。このコマンドに備えられており、 270 ページの 『ローダー・パッケージ・コマンドおよびスクリプトの使用』でリストされているコマンド・ファイルまたはスクリプトは、実際の Java コマンドのラッパーとして機能し、同じパラメーターを受け入れます。それゆえ、Java コマンドを直接呼び出すのではなく、コマンド・ファイルまたはスクリプトを使用することをお勧めします。

パラメーター値:

-dbname
ターゲット・データベースの名前

-dbuser
データベースに接続しているユーザーの名前

-dbpwd
データベースに接続しているユーザーのパスワード

-infile 入力 XML ファイルの名前

-method
データをデータベースに挿入する際にローダーが使用する操作の方式

- ロード (load) 方式では、データベース・ベンダーのネイティブ・ローダーが使用されます。 Oracle データベースの場合、ロード方式はローカルとリモートの両方で使用できますが、 DB2 データベースの場合、ロード方式はローカルでしか使用できません。
- リモート DB2 データベースにデータをロードするには、インポート・オプションを使用します。インポート (import) 方式の場合インポート (import) または更新 (update) オプションがデータベース・ベンダーで使用できる場合には、それが使用されます。インポートまたは更新のオプシ

ョンが使用できない場合、 JDBC を使用する SQL ステートメントを使用してデータベースが更新されます。

- SQL インポート (sqlimport) 方式は、ローカル・データベースでもリモート・データベースでも使用できます。
- delete メソッドでは、データベースからデータが削除されます。

-noprimary

入力ファイルの中でレコードの 1 次鍵が欠落している場合に、ローダーが実行するアクション

- error オプションを指定した場合、欠落している 1 次鍵をエラーとして報告して終了します。
- skip オプションの場合、入力ファイルの中に 1 次鍵のないレコードがあればスキップされます。
- insert オプションの場合は、データの挿入または削除が試行されます。

このパラメーターはオプションです。デフォルトのアクションは error です。

-commitcount

SQL update 動作方式を使用している場合、ここに指定する数のレコードが処理されたなら、データベースのコミットが発生します。このパラメーターはオプションです。デフォルトの数は 1 です。

-maxerror

SQL update 動作方式において、ここに指定する数のエラーが発生すると、ローダーは終了します。このパラメーターはオプションです。

-customizer

使用するカスタマイザー・プロパティー・ファイルの名前。このパラメーターはオプションです。 `MassLoadCustomizer.properties` がデフォルトのファイルです。カスタマイザー・プロパティー・ファイルを次の例のように指定できます。

```
-customizer d:¥WebSphere¥CommerceServer¥prop¥ml.properties
```

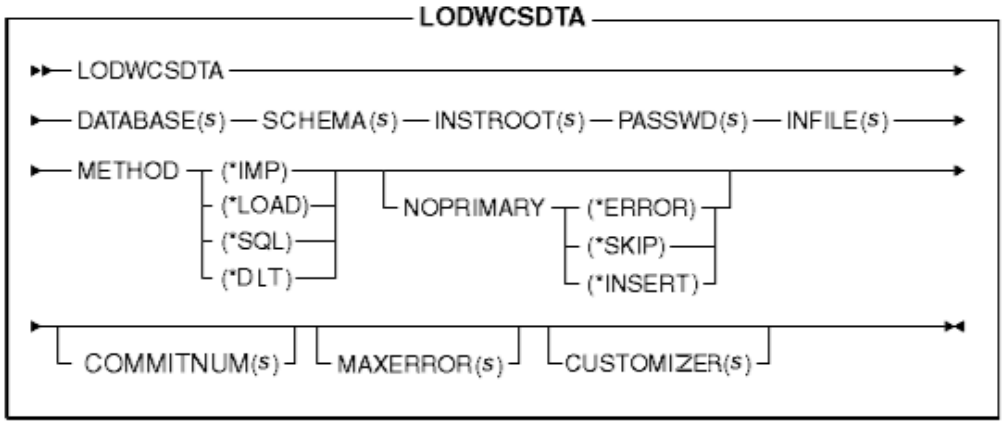
クラスパスのシステム環境変数で指定されたディレクトリーにこのファイルが存在する場合、同じファイルを次の例のように指定できます。

```
-customizer ml
```

-schemaname

ターゲット・データベース・スキーマの名前。このパラメーターはオプションです。

400



パラメーター値:

DATABASE

ターゲット・データベースの名前 (リレーショナル・データベース・ディレクトリーで表示されるもの)。

SCHEMA

ターゲット・データベース・スキーマの名前。これは、インスタンス名と同じです。

INSTROOT

WebSphere Commerce インスタンスのルート・パスの絶対パス名 (`/QIBM/UserData/WebCommerce/instances/`*instance_name* など)

PASSWD

WebSphere Commerce インスタンスのパスワード

INFILE

入力 XML ファイルの名前

METHOD

データをデータベースに挿入する際にローダーが使用する操作の方式

- ロード (*LOAD) 方式では、データベース・ベンダーのネイティブ・ローダーが使用されます。 Oracle データベースの場合、ロード方式 (*LOAD) はローカルとリモートの両方で使用できますが、 DB2 データベースの場合、ロード方式 (*LOAD) はローカルでしか使用できません。
- リモート DB2 データベースにデータをロードするには、インポート (*IMP) オプションを使用します。インポート (*IMP) 方式の場合インポート (import) または更新 (update) オプションがデータベース・ベンダーで使用できる場合には、それが使用されます。インポートまたは更新のオプションが使用できない場合、 JDBC を使用する SQL ステートメントを使用してデータベースが更新されます。
- SQL インポート (*SQL) 方式は、ローカル・データベースでもリモート・データベースでも使用できます。
- 削除 (*DLT) メソッドでは、データベースからデータが削除されます。

NOPRIMARY

入力ファイルの中でレコードの 1 次鍵が欠落している場合に、ローダーが実行するアクション

- エラー (*ERROR) オプションを指定した場合、欠落している 1 次鍵をエラーとして報告して終了します。
- スキップ (*SKIP) オプションの場合、入力ファイルの中に 1 次鍵のないレコードがあればスキップされます。
- 挿入 (*INSERT) オプションの場合は、データの処理 (挿入または削除) が試行されます。

このパラメーターはオプションです。デフォルトのアクションは error です。

COMMITNUM

SQL update 動作方式を使用している場合、ここに指定する数のレコードが処理されたなら、データベースのコミットが発生します。このパラメーターはオプションです。デフォルトの数は 1 です。

MAXERROR

SQL update 動作方式において、ここに指定する数のエラーが発生すると、ローダーは終了します。このパラメーターはオプションです。

CUSTOMIZER

使用するカスタマイザー・プロパティー・ファイルの名前。このパラメーターはオプションです。デフォルトのファイルは `ISeries_LODWCSDTA_Customizer.properties` です。カスタマイザー・プロパティー・ファイルを次の例のように指定できます。

```
CUSTOMIZER(/wc/prop/ml.properties)
```

クラスパスのシステム環境変数で指定されたディレクトリーにこのファイルが存在する場合、同じファイルを次の例のように指定できます。

```
CUSTOMIZER(ml)
```

処理に関する推奨事項:

データをロードする前に、 3 つの処理方式のうちのどの方式を使用すると最も良い結果が得られるか判別する必要があります。

ロード方式の選択:

次のいずれかの状況におけるロード方式を検討してください。

- ソース・データがクリーンであり、データベースにデータが入っていない

  注: クリーン・データとは、データのロード先のテーブルの制約を違反していないデータのことです。
- ソース・データがクリーンであり、データベースにデータが入っていない
- ソース・データがクリーンであり、ロード中のデータがデータベースに入っていない
- ソース・データがクリーンであり、 1 つ以上のターゲット・テーブルに 1 次鍵が含まれておらず、ロード中のデータがデータベースに入っていない
- データベースがローカル DB2 データベースである

- ロード中、他のユーザーまたはアプリケーションからデータベースにアクセスされない

400 ロード方式を使用すると、データはデータベースにロードされます。データがすでに存在する場合、重複鍵エラーの結果としてコマンドは失敗し、重複エラーのメッセージが表示されます。

ロード方式を使用する際、以下の制限があります。

- ロード方式はビット・データ・フィールドでデータの挿入または更新ができない。
- DB2 ロード方式を使用すると、データベースへの新規レコードの挿入だけが行われる。既存のレコードの更新は行われない。

インポート方式の選択:

Windows AIX Solaris Linux DB2 でインポート方式を使用する場合も、データはデータベースにロードされます。データがすでに存在する場合、削除はされないものの、新しい値で更新されます。次のいずれかの状況におけるこの方式を検討してください。

- データベース管理システムが DB2 である
- データがクリーンかどうか分からない
- 同種データの多数セットを列レベルで更新する必要がある
- データのインポート先の全テーブルに 1 次鍵がある

400 iSeries でインポート方式を使用する場合も、データはデータベースにロードされます。データがすでに存在する場合、削除はされないものの、新しい値で更新されます。次のいずれかの状況におけるこの方式を検討してください。

- データがクリーンかどうか分からない
- データがすでにデータベース内に存在する
- データのインポート先の全テーブルに 1 次鍵がある

インポート方式を使用する際、以下の制限があります。

- インポート方式を使用するためには、データベース管理システムが DB2 でなければならない。
- インポート方式はビット・データ・フィールドでデータの挿入または更新ができない。
- インポート方式の場合、ローダーは 1 次鍵が定義されているテーブルだけを挿入または更新を行う。この方式は、1 次鍵を持たないテーブルにデータを挿入したり、更新したりできません。入力レコードの列に 1 次の値しかない場合、そのレコードは拒否されます。

SQL インポート方式の選択:

SQL インポート方式の場合、データを更新したりデータベースにデータを挿入したりするために、 JDBC または SQL ステートメントが使用されます。データが存在しない場合には挿入され、データが存在する場合、それらのデータは更新されます。次のいずれかの状況におけるこの方式を検討してください。

- 既存のデータを更新しており、列レベルの更新を必要とする
- データの中にクリーンではないものがある
- データベースがローカルにない

注: 商品アドバイザー検索スペースの同期を使用する場合には、データのロードに SQL インポート方式を使用する必要があります。

削除方式の選択:

削除方式は、入力 XML 文書にあるデータをデータベースから削除するために使用します。要素には、1 次鍵の値かまたはテーブルの固有索引が含まれていなければなりません。「カスケード削除」を使用可能にして、従属データを別のテーブルに持つデータを削除すると、その従属データも削除されます。

その他の考慮事項:

- **SQL インポート方式とロード方式の比較**

  SQL インポート方式は、データの整合性を検査し (外部参照を含む)、既存のデータの更新を可能にしますが、ロード方式はこれを行いません。

- **インポート方式と SQL インポート方式の比較**

  インポート方式と SQL インポート方式のどちらも同じ機能を実行します。一般に、インポート方式の方が早いですが、一時ファイル用のディスク・スペースを必要とします。

  インポート方式は、1 次鍵を定義したテーブルだけを挿入または更新しますが、SQL インポート方式では、テーブルに 1 次鍵を定義する必要はありません。

- **使用されるデータベース製品に基づく方式の比較**

  インポート方式とロード方式は、 DB2 用に最適化された固有のユーティリティーを使用しますが、 SQL インポート方式は、 JDBC 呼び出し (これは多数のデータベース製品で一般的) を使用します。

パフォーマンスに関する考慮事項:

ローダーを使用して大量の文書をデータベースにロードする際は、以下の項目を考慮してください。

- **Java 仮想マシン (JVM) のヒープ・サイズ**

  JVM ヒープに割り当てられているデフォルトの最大メモリー量は 64 MB です。これを増やさない場合、 JVM はロード・プロセス中にメモリー不足になってしまう可能性があります。 Java ヒープに割り当てられている最大メモリー量は、Java コマンドの JVM -mx オプションを使用して変更することができます。

- **トレースのロギング**

  トレース・ロガーは大量の XML 文書のロードするときに、 JVM ヒープを使い果たしてしまう可能性があります。トレース情報は大抵、実行が失敗した場合にそれをデバッグするために使用されます。ロード・プロセスのトレースが不要な場合は、トレースをオフにしてください。トレースをオフにすると、パフォーマンスは著しく向上します。ロギング構成 XML 文書を変更することで、トレースをオフにします。ロギング構成 XML 文書の変更の詳細については、 *IBM WebSphere Commerce 5.4 Catalog Manager ユーザーズ・ガイド* の最新バージョンを参照してください。

- **コミット数**

  ローダーが SQL インポート・モードを操作するときのデフォルトのコミット数は 1 です。したがって、デフォルトではデータベースを更新するたび、またはデータベースに挿入するたびにトランザクションがコミットされます。大量文書を扱う場合のローダーのパフォーマンスを向上させるには、コミット数を増やす必

要があります。この値は ″100″ にすることが勧められています。ただし、サーバー上の物理メモリー量や DBMS トランザクションのログ・サイズなどによっては、この値だと大きい場合があります。

ローダーのコミット数は Load コマンドの -commitcount *count* オプションを使用して変更します (*count* は、トランザクションをコミットする前に実行されるステートメントの数です)。

- **ロギング構成**

  まれに、以下の状況のいずれかのために、データ・ロード中の進行が低速になることがあります。

  – ローダーを呼び出したユーザーに、ディレクトリーに対する書き込み許可がない、あるいはロギング構成文書で指定されたファイルの更新許可がない。
  – ファイルのロケーションとしてロギング構成文書で指定されたディレクトリーが存在しない。
  – ファイルのロケーションとしてロギング構成文書で指定されたドライブに十分のスペースがない。

  これらの問題のいくつかを訂正するときに、ファイルのロケーションを変更するためにロギング構成文書 (デフォルトでは `WCALoggerConfig.xml`) を変更するが必要がある場合があります。ロギング構成 XML 文書の変更の詳細については、*IBM WebSphere Commerce 5.4 Catalog Manager ユーザーズ・ガイド* の最新バージョンを参照してください。

このコマンドを実行するために使用するファイルの設定とカスタマイズの詳細ついては、 *IBM WebSphere Commerce 5.4 Catalog Manager ユーザーズ・ガイド* の最新バージョンを参照してください。

データの変換および抽出に関するローダー・パッケージ・コマンド

DTD Generate コマンド

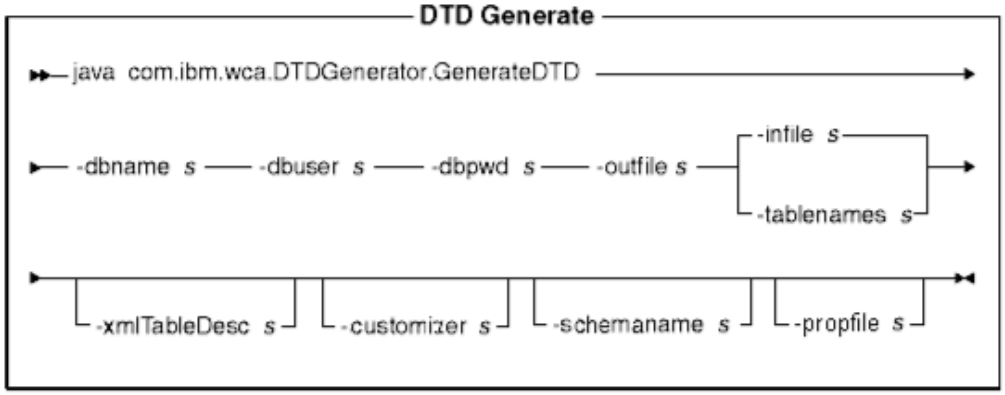
このコマンドは、ローダー・パッケージで使用する DTD を作成します。その DTD は、ロード・プロセスを通じて使用されることになります。コマンドの呼び出し方によって、 DTD ジェネレーターが DTD だけを生成する場合と、 DTD とともに XML スキーマを生成する場合があります。

DTD ジェネレーターは、 WebSphere Commerce データベース・スキーマに基づいて DTD を作成することができます。サンプル・ストア・アーカイブに含まれている DTD を使用し、データベース・スキーマを変更しない場合、 DTD ジェネレーターを使用して DTD を生成する必要はありません。提供されている DTD は以下のディレクトリーにあります。

- NT *drive*:¥WebSphere¥CommerceServer¥xml¥sar
- 2000 *drive*:¥Program Files¥WebSphere¥CommerceServer¥xml¥sar
- AIX /usr/WebSphere/CommerceServer/xml/sar
- Solaris Linux /opt/WebSphere/CommerceServer/xml/sar
- 400 /QIBM/ProdData/WebCommerce/xml/sar

提供されている DTD を使用することをお勧めします。しかし、データベース・スキーマをカスタマイズする場合は、提供されている DTD を変更内容と一致するように編集するか、または DTD を新しく作成するかしなければなりません。

Windows AIX Solaris Linux



注: 上の図は、主にコマンド・パラメーターを参照するための図です。このコマンドに備えられており、 270 ページの 『ローダー・パッケージ・コマンドおよびスクリプトの使用』でリストされているコマンド・ファイルまたはスクリプトは、実際の Java コマンドのラッパーとして機能し、同じパラメーターを受け入れます。したがって、Java コマンドを直接呼び出すのではなく、コマンド・ファイルまたはスクリプトを使用することをお勧めします。

パラメーター値:

-dbname

ターゲット・データベースの名前

-dbuser
データベースに接続しているユーザーの名前

-dbpwd
データベースに接続しているユーザーのパスワード

-outfile
出力 DTD ファイルの名前 (.dtd 拡張子を伴う場合がある)

-infile 各行にデータベース・テーブル名を含む入力ファイルの名前

-tablenames
コンマで区切られ、二重引用符で囲まれるテーブルの名前

-xmltabledesc
作成される XML スキーマ・ファイルのファイル・パス。このパラメーターはオプションです。

-customizer
使用するカスタマイザー・プロパティー・ファイルの名前。このパラメーターはオプションです。 DB2ConnectionCustomizer.properties がデフォルトのファイルです。次の例のようにカスタマイザー・プロパティー・ファイルを指定できます。

```
-customizer d:¥WebSphere¥CommerceServer¥prop¥dtdgen.properties
```

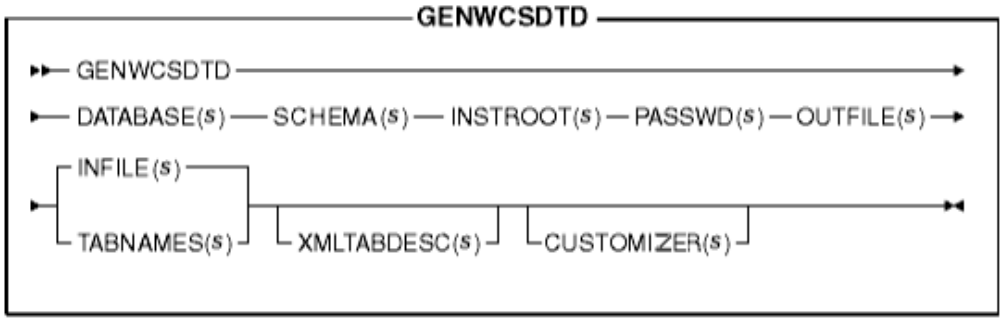
クラスパスのシステム環境変数で指定されたディレクトリーにこのファイルが存在する場合、同じファイルを次の例のように指定できます。

```
-customizer dtdgen
```

-schemaname
ターゲット・データベース・スキーマの名前。このパラメーターはオプションです。これは、デフォルトのデータベース・ユーザー名になります。

-propfile
Web エディターについてのヘルプ・テキスト、デフォルト値、およびフィールド説明情報を保管できるプロパティー・ファイルは、説明を構成します。このパラメーターはオプションです。

400



パラメーター値:

DATABASE
ターゲット・データベースの名前 (リレーショナル・データベース・ディレクトリーで表示されるもの)。

SCHEMA
ターゲット・データベース・スキーマの名前。これは、インスタンス名と同じです。

INSTROOT
WebSphere Commerce インスタンスのルート・パスの絶対パス名 (`/QIBM/UserData/WebCommerce/instances/`*instance_name* など)

PASSWD
WebSphere Commerce インスタンスのパスワード

OUTFILE
出力 DTD ファイルの名前 (`.dtd` 拡張子を伴う場合がある)

INFILE
各行にデータベース・テーブル名を含む入力ファイルの名前

TABNAMES
コンマで区切られ、二重引用符で囲まれるテーブルの名前

XMLTABDESC
作成される XML スキーマ・ファイルのファイル・パス。このパラメーターはオプションです。

CUSTOMIZER
使用するカスタマイザー・プロパティー・ファイルの名前。このパラメーターはオプションです。デフォルトのファイルは `ISeries_GENWCSDTD_Customizer.properties` です。次の例のようにカスタマイザー・プロパティー・ファイルを指定できます。

```
CUSTOMIZER(/wc/prop/dtdgen.properties)
```

クラスパスのシステム環境変数で指定されたディレクトリーにこのファイルが存在する場合、同じファイルを次の例のように指定できます。

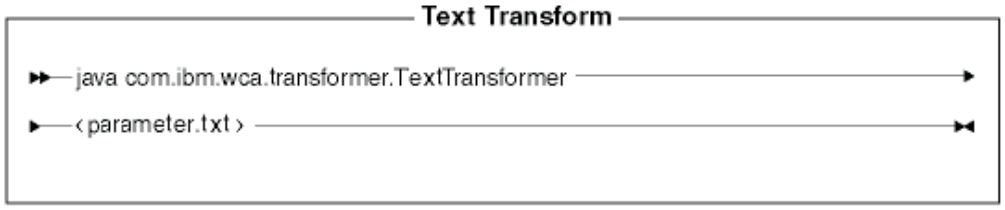
```
CUSTOMIZER(dtdgen)
```

このコマンドを実行するために使用するファイルの設定とカスタマイズの詳細ついては、 *IBM WebSphere Commerce 5.4 Catalog Manager ユーザーズ・ガイド* の最新バージョンを参照してください。

Text Transform コマンド

このコマンドは、文字区切り可変長形式のデータと XML 形式のデータの間で変換を行います。

Windows AIX Solaris Linux



注: 上の図は、主にコマンド・パラメーターを参照するための図です。このコマンドに備えられており、 270 ページの 『ローダー・パッケージ・コマンドおよびスクリプトの使用』でリストされているコマンド・ファイルまたはスクリプトは、実際の Java コマンドのラッパーとして機能し、同じパラメーターを受け入れます。したがって、Java コマンドを直接呼び出すのではなく、コマンド・ファイルまたはスクリプトを使用することをお勧めします。

パラメーター値:

以下の値がパラメーター・ファイル (*parameter.txt*) の中で指定され、コンマによって区切られています。

input file
変換されるファイルの名前

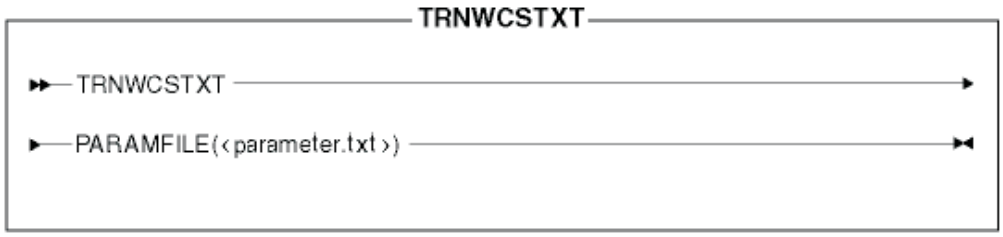
schema file
変換で使用される XML スキーマ・ファイルの名前

output file
変換データが保管される出力ファイルの名前

transformation method
データを出力ファイルに追加する際に使用される方式。新しいファイルを作成する場合は **Create** を、出力データを既存のデータ・ファイルに不可する場合は **Append** を指定してください。

このファイルは、「マニフェスト」ファイルまたは「コマンド」ファイルとも呼ばれます。これには、4 つのパラメーターごとの複数行が含まれることもあります。

400



パラメーター値:

以下の値がパラメーター・ファイル (*parameter.txt*) の中で指定され、コンマによって区切られています。

input file
変換されるファイルの名前

schema file
変換で使用される XML スキーマ・ファイルの名前

output file
変換データが保管される出力ファイルの名前

transformation method
データを出力ファイルに追加する際に使用される方式。新しいファイルを作成する場合は **Create** を、出力データを既存のデータ・ファイルに不可する場合は **Append** を指定してください。

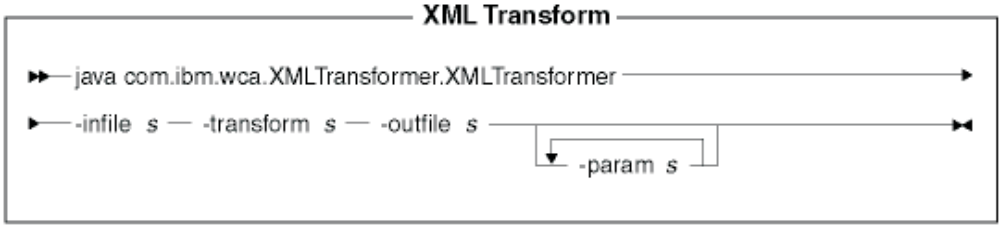
注: このファイルは、「マニフェスト」ファイルまたは「コマンド」ファイルとも呼ばれます。

このコマンドの詳細については、 *IBM WebSphere Commerce 5.4 Catalog Manager ユーザーズ・ガイド* の最新バージョンを参照してください。

XML Transform コマンド

このコマンドは、XML ファイルを別の XML 形式に変換します。

Windows AIX Solaris Linux



注: 上の図は、主にコマンド・パラメーターを参照するための図です。このコマンドに備えられており、 270 ページの 『ローダー・パッケージ・コマンドおよびスクリプトの使用』でリストされているコマンド・ファイルまたはスクリプトは、実際の Java コマンドのラッパーとして機能し、同じパラメーターを受け入れます。したがって、Java コマンドを直接呼び出すのではなく、コマンド・ファイルまたはスクリプトを使用することをお勧めします。

パラメーター値:

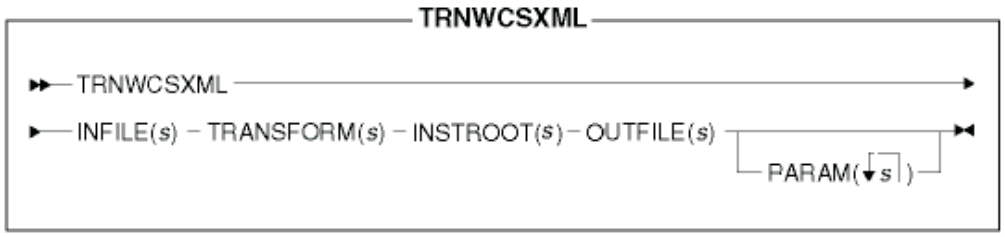
-infile 変換されるファイルの名前

-transform
変換 XSL マッピング・ルール・ファイル

-outfile
変換データが保管される出力 XML ファイルの名前

-param
XSL マッピング・ルール・ファイルに渡されるパラメーター。***このパラメーターはオプションです。*** 複数の ″name=value″ の対に渡すために、このパラメーターは何回でも指定できます。

400



パラメーター値:

INFILE
変換されるファイルの名前

TRANSFORM
変換 XSL マッピング・ルール・ファイル

INSTROOT
WebSphere Commerce インスタンスのルート・パスの絶対パス名 (/QIBM/UserData/WebCommerce/instances/*instance_name* など)

OUTFILE
変換データが保管される出力 XML ファイルの名前

PARAM
XSL マッピング・ルール・ファイルに渡されるパラメーター。***このパラメーターはオプションです。*** 複数の "name=value" の対に渡すために、このストリングに複数の値を含めることができます。

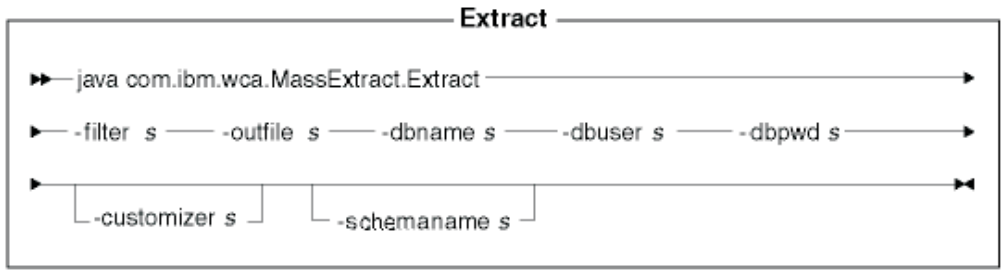
このコマンドの詳細については、 *IBM WebSphere Commerce 5.4 Catalog Manager* ユーザーズ・ガイド の最新バージョンを参照してください。

Extract コマンド

このコマンドは、データベースから選択されたデータのサブセットを XML ファイルの形式で抽出します。

抽出プログラムを使ってデータベースからデータを抽出するには、抽出フィルター・ファイルを使って、データベースから抽出したいデータを指定しなければなりません。使用する抽出フィルターは、抽出したいデータの種類によって異なります。

Windows AIX Solaris Linux



注: 上の図は、主にコマンド・パラメーターを参照するための図です。このコマンドに備えられており、 270 ページの 『ローダー・パッケージ・コマンドおよびスクリプトの使用』でリストされているコマンド・ファイルまたはスクリプトは、実際の Java コマンドのラッパーとして機能し、同じパラメーターを受け入れます。したがって、Java コマンドを直接呼び出すのではなく、コマンド・ファイルまたはスクリプトを使用することをお勧めします。

パラメーター値:

-filter 抽出フィルター・ファイルの名前

-outfile
抽出データが保管される出力 XML ファイルの名前

-dbname
データ抽出元のデータベースの名前

-dbuser
データ抽出元のデータベースのデータベース・ユーザー名

-dbpwd
データ抽出元のデータベースのユーザー名に関連したパスワード

-customizer
使用するカスタマイザー・プロパティー・ファイルの名前。
`DB2ConnectionCustomizer.properties` がデフォルトのファイルです。カスタマイザー・プロパティー・ファイルを次の例のように指定できます。

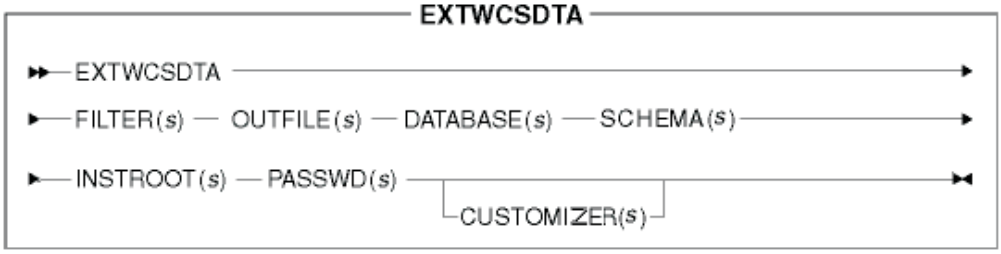
```
-customizer d:¥WebSphere¥CommerceServer¥prop¥extract.properties
```

クラスパスのシステム環境変数で指定されたディレクトリーにこのファイルが存在する場合、同じファイルを次の例のように指定できます。

```
-customizer extract
```

-schemaname
ターゲット・データベース・スキーマの名前。

400



パラメーター値:

FILTER
抽出フィルター・ファイルの名前

OUTFILE
抽出データが保管される出力 XML ファイルの名前

DATABASE
データ抽出元のデータベースの名前 (リレーショナル・データベース・ディレクトリーで表示されるもの)

SCHEMA
データ抽出元のデータベース・スキーマの名前。これは、インスタンス名と同じになります。

INSTROOT
WebSphere Commerce インスタンスのルート・パスの絶対パス名 (`/QIBM/UserData/WebCommerce/instances/`*instance_name* など)

PASSWD
WebSphere Commerce インスタンスのパスワード

CUSTOMIZER
使用するカスタマイザー・プロパティー・ファイルの名前。デフォルトのファイルは `ISeries_EXTWCSDTA_Customizer.properties` です。カスタマイザー・プロパティー・ファイルを次の例のように指定できます。

```
CUSTOMIZER(/wc/prop/extract.properties)
```

クラスパスのシステム環境変数で指定されたディレクトリーにこのファイルが存在する場合、同じファイルを次の例のように指定できます。

```
CUSTOMIZER(extract)
```

このコマンドの詳細については、 *IBM WebSphere Commerce 5.4 Catalog Manager ユーザーズ・ガイド* の最新バージョンを参照してください。

ローダー・パッケージ・コマンドに関連したツール

テキスト変換ツール

テキスト変換ツールは、文字区切り変数形式のデータを XML 形式のデータに変換したり、 XML 形式のデータを文字区切り変数形式のデータに変換したりする処理を行うのに役立ちます。このツールには、以下のビューが備えられています。

1. 「Text Schema Edit (テキスト・スキーマの編集)」ビュー。このビューで、変換に使用される XML スキーマを作成および変更することができます。
2. 「Transformation Command Edit (トランスフォーメーション・コマンドの編集)」ビュー。このビューで、変換プロセスを実行するために使用する実際のコマンドを作成および変更することができます。
3. 「Transformation Command Process (トランスフォーメーション・コマンド・プロセス)」ビュー。このビューで、変換プロセスを立ち上げることができます。

このツールの詳細については、 *IBM WebSphere Commerce 5.4 Catalog Manager ユーザーズ・ガイド* の最新バージョンを参照してください。

XSL エディター

XML 変換プログラムは、 XSL を使用して XML ファイルを別の XML ファイルに変換するためのルールを定義します。 XSL エディターにあるマッピング機能にはビジュアル・インターフェースが備えられており、これを使用してソース DTD 内の要素とターゲット DTD 内の要素との関連付けを行うことができます。 2 つの

DTD が備えられていますが、最初の (ソース) DTD に準拠する XML ファイルから 2 番目の (ターゲット) DTD に準拠するファイルへの変換方法を決定する XSL ルールを作成できます。

このツールの詳細については、 *IBM WebSphere Commerce 5.4 Catalog Manager ユーザーズ・ガイド* の最新バージョンを参照してください。

Web エディター

Web エディターを使用すると、 Web ブラウザーでのカタログ・データの作成、削除、および変更が行えます。情報を表示したり更新したりするためのデータ入力フォームは、 Web エディターの中心的なものです。最も単純な例を挙げると、このフォームは WebSphere Commerce Server データベースのテーブルに該当します。管理者は、提供されているデフォルトのフォームを使用することもできますし、使用可能フォームをカスタマイズすることもできます。

このツールの詳細については、 *IBM WebSphere Commerce 5.4 Catalog Manager ユーザーズ・ガイド* の最新バージョンを参照してください。

ストア・データのロード

このセクションでは、ローダー・パッケージ・コマンド・ユーティリティーを使って、 WebSphere Commerce Server データベースにデータをロードする方法の例を示します。

注:

1. このセクションの例は、 Windows NT 環境で実行されます。その他の環境におけるこれらのコマンドの実行の詳細については、 *IBM WebSphere Commerce 5.4 Catalog Manager ユーザーズ・ガイド* の最新バージョンを参照してください。
2. ローダー・パッケージ・コマンド・ユーティリティーは DB2、 DB2 for iSeries および Oracle データベースをサポートしていますが、以下の例では DB2 のコマンドとオプションだけを取り上げています。 DB2 以外のデータベースを使用している場合は、 *IBM WebSphere Commerce 5.4 Catalog Manager ユーザーズ・ガイド* の最新バージョンにある説明に従って、カスタマイザー・プロパティー・ファイルを確実に変更してください。

WebSphere Commerce データ・グループのロード・プロセスの詳細については、279 ページの 『第 29 章 WebSphere Commerce データ・グループのロード』および281 ページの 『データ・グループのロード』を参照してください。

ローダー・パッケージ・コマンドおよびスクリプトの使用

ローダー・パッケージ・コマンドを使用するには、以下の WebSphere Commerce ディレクトリーにあるスクリプトまたはコマンドを使用します。

- NT *drive*:¥WebSphere¥CommerceServer¥bin
- 2000 *drive*:¥Program Files¥WebSphere¥CommerceServer¥bin
- AIX /usr/WebSphere/CommerceServer/bin
- Solaris Linux /opt/WebSphere/CommerceServer/bin
- 400 QWEBCOMM 固有のライブラリー

スクリプトおよびコマンドは以下のとおりです。

- Windows

dtdgen.cmd	DTD Generate コマンド
idresgen.cmd	ID Resolve コマンド
massload.cmd	Load コマンド
txttransform.cmd	Text Transform コマンド
xmltransform.cmd	XML Transform コマンド
massextract.cmd	Extract コマンド

- AIX Solaris Linux

dtdgen.sh	DTD Generate シェル・スクリプト
idresgen.sh	ID Resolve シェル・スクリプト
massload.sh	Load シェル・スクリプト
txttransform.sh	Text Transform シェル・スクリプト
xmltransform.sh	XML Transform シェル・スクリプト
massextract.sh	Extract シェル・スクリプト

- 400

GENWCSDTD	DTD Generate コマンド
RESWCSID	ID Resolve コマンド
LODWCSDTA	Load コマンド
TRNWCSTXT	Text Transform コマンド
TRNWCSXML	XML Transform コマンド
EXTWCSDTA	Extract コマンド

ID 解決の例

このセクションで説明される ID 解決の例では、 ToolTech サンプル・ストアのストア資産ファイルが使用されています。

内部別名 ID 解決の使用

内部別名を使用するデータを WebSphere Commerce Server データベースにロードする前にその ID を解決するには、以下の例のように ID Resolve コマンドを実行します。

1. 270 ページの 『ローダー・パッケージ・コマンドおよびスクリプトの使用』でリストされている適切な ID Resolve コマンドまたはスクリプトを含むディレクトリーがパスに含まれていることを確認します。

   この例では、`idresgen.cmd` を使用します。
2. 作業ディレクトリーを作成します。

   この例では、 `c:¥WebSphere¥CommerceServer¥runtime¥test` というディレクトリーを作成します。
3. ID リゾルバーが検出できる場所に入力 XML ファイルと参照 DTD ファイルがあることを確認してください。

   この例では、以下のようにします。

   a. `c:¥WebSphere¥CommerceServer¥samlestore¥ToolTech¥en_US_fr_FR.sar` を `c:¥WebSphere¥CommerceServer¥runtime¥test` にコピーします。

   b. Windows コマンド・プロンプトで、以下のコマンドを入力します。

      ```
      cd c:¥WebSphere¥CommerceServer¥runtime¥test
      ```

   c. Windows コマンドで、以下のコマンドを入力します。

      ```
      jar -xvf en_US_fr_FR.sar
      ```

      これにより、 ToolTech サンプル・ストアの XML ファイルが `c:¥WebSphere¥CommerceServer¥runtime¥test¥data` にコピーされます。

   d. Windows コマンドで、以下のコマンドを入力します。

      ```
      cp c:¥WebSphere¥CommerceServer¥xml¥sar¥store.dtd
      c:¥WebSphere¥CommerceServer¥runtime¥test¥data
      ```

      これにより、 `store.dtd` ファイルが `c:¥WebSphere¥CommerceServer¥runtime¥test¥data` にコピーされます。

   e. Windows コマンドで、以下のコマンドを入力します。

      ```
      cp c:¥WebSphere¥CommerceServer¥xml¥sar¥DBLoadMacros.dtd
      c:¥WebSphere¥CommerceServer¥runtime¥test¥data
      ```

      これにより、 `DBLoadMacros.dtd` ファイルが `c:¥WebSphere¥CommerceServer¥runtime¥test¥data` にコピーされます。

f. Windows コマンドで、以下のコマンドを入力します。

```
cp c:¥WebSphere¥CommerceServer¥xml¥sar¥fulfillment.dtd
c:¥WebSphere¥CommerceServer¥runtime¥test¥data
```

これにより、 fulfillment.dtd ファイルが c:¥WebSphere¥CommerceServer¥runtime¥test¥data にコピーされます。

4. 適切な WebSphere Commerce Server データベースをロードします。

この例では、 ″mall″ という WebSphere Commerce Server データベース・インスタンスを使用します。 1 次鍵および外部キーは、このデータベースの KEYS テーブルと SUBKEYS テーブルから入手します。ですから、ID リゾルバーはデータベースがロードされないと ID を解決できません。

5. store.xml ファイルに、以下の要素があります。

```
<storeent
   STOREENT_ID="@storeent_id_1"
   MEMBER_ID="&MEMBER_ID;"
   TYPE="S"
   IDENTIFIER="ToolTech"
   SETCURR="USD"
/>
```

@storeent_id_1 仕様は、STOREENT_ID 属性の値の内部別名で、&MEMBER_ID; は、エンティティー・パラメーターです。 &MEMBER_ID; の値は、DBLoadMacros.dtd マクロ・ファイルで定義されます。 &MEMBER_ID; の値は、ローダーを使ってロードする前に置き換える必要があります。 ID リゾルバーが @storeent_id_1 を見付けると、 1 次テーブルのキャッシュの中に storeent が存在するか調べます。これは 1 次テーブルなので、storeeent は存在します。 ID リゾルバーはそのテーブルのカウンターを取り出し、それに 1 を加えて内部別名と置き換えます。 store.xml ファイル内にあるこの種の他のすべてのエントリーも、これと同じように処理されます。

6. Windows コマンドで、以下のコマンドを入力します。

```
idresgen -dbname mall -dbuser db2admin -dbpwd db2admin -infile store.xml
-outfile c:¥WebSphere¥CommerceServer¥runtime¥test¥data¥store1.xml -method load
```

store1.xml の中の最初の出力 XML フラグメントは、次のようになります。

```
<storeent
   STOREENT_ID="10001"
   MEMBER_ID="-2001"
   TYPE="S"
   IDENTIFIER="ToolTech"
   SETCCURR="USD"
/>
```

store1.xml の中の 2 番目の XML フラグメントは、次のようになります。

```
<store
   STORE_ID="10001"
   DIRECTORY="ToolTech"
   FFMCENTER_ID=""
   LANGUAGE_ID="-1"
   STOREGRP_ID="-1"
   ALLOCATIONGOODFOR="43200"
   BOPMPADFACTOR="0"
   DEFAULTBOOFFSET="2592000"
   FFMCSELECTIONFLAGS="0"
   MAXBOOFFSET="7776000"
   REJECTEDORDEXPIRY="259200"
```

```
   RTNFFMCTR_ID=""
   PRICEREFFLAGS="0"
   STORETYPE="B2B"
/>
```

FFMCENTER_ID 属性と RTNFFMCTR_ID 属性は解決されませんでした。内部別名 @ffmcenter_id_1 は解決されず、これは空の二重引用符 (″″) に置き換えられました。これは、エラーです。外部鍵制約を違反すると、ローダーを使用して適切にデータをロードすることができません。別名が FFMCENTER テーブルを参照する前にこのテーブルが処理されなかったため、 ID リゾルバーはこの内部別名を解決できませんでした。この問題を解決するには、以下の 2 つのうちのいずれかを行ってください。

- オプション 1:
  a. 以下のコマンドを入力して、 (FFMCENTER テーブルが定義されている) fulfillment.xml ファイルに対して ID リゾルバーを実行する。

     ```
     idresgen -dbname mall -dbuser db2admin -dbpwd db2admin
     -infile fulfillment.xml -outfile
     c:¥WebSphere¥CommerceServer¥runtime¥test¥data¥fulfillment1.xml
     -method load
     ```

     解決された fulfillment1.xml 出力の要素は、次のようになります。

     ```
     <fulfillment-asset>
     <ffmcenter
         FFMCENTER_ID="10001"
         MEMBER_ID="-2001"
        NAME="ToolTech Home"
        DEFAULTSHIPOFFSET="0"
         MARKFORDELETE="0"
     />
     </fulfillment-asset>
     ```

  b. 生成された出力ファイル (store1.xml) から FFMCENTER_ID 鍵を取得し、 c:¥WebSphere¥CommerceServer¥runtime¥test¥data にある store.xml の作業コピーの中のすべての @ffmcenter_id_1 を、ここで取得した鍵に置換してください。

  c. 以下のコマンドを入力します。

     ```
     idresgen -dbname mall -dbuser db2admin -dbpwd db2admin -infile store.xml
     -outfile c:¥WebSphere¥CommerceServer¥runtime¥test¥data¥store1.xml
     -method load
     ```

     完全に解決された store1.xml 出力ファイルの要素は、次のようになります。

     ```
     <store-asset>
     <storeent
         STOREENT_ID="10151"
         MEMBER_ID="-2001"
         TYPE="S"
         IDENTIFIER="ToolTech"
         SETCCURR="USD"
      />
     <store
         STORE_ID="10151"
         DIRECTORY="ToolTech"
         FFMCENTER_ID="10001"
         LANGUAGE_ID="-1"
         STOREGRP_ID="-1"
         ALLOCATIONGOODFOR="43200"
     ```

```
    BOPMPADFACTOR="0"
    DEFAULTBOOFFSET="2592000"
    FFMCSELECTIONFLAGS="0"
    MAXBOOFFSET="7776000"
    REJECTEDORDEXPIRY="259200"
    RTNFFMCTR_ID="10001"
    PRICEREFFLAGS="0"
    STORETYPE="B2B"
 />
<vendor
    VENDOR_ID="10001"
    STOREENT_ID="10151"
    VENDORNAME="Tooltech Vendor"
    MARKFORDELETE="0"
 />
<dispentrel
   AUCTIONSTATE="0"
    CATENTRY_ID="0"
    CATENTTYPE_ID="ProductBean"
    DEVICEFMT_ID="-1"
    DISPENTREL_ID="10001"
    MBRGRP_ID="0"
    PAGENAME="CatalogProductDisplay.jsp"
    STOREENT_ID="10151"
    RANK="0"
 />
<dispentrel
   AUCTIONSTATE="0"
    CATENTRY_ID="0"
    CATENTTYPE_ID="ItemBean"
    DEVICEFMT_ID="-1"
    DISPENTREL_ID="10002"
    MBRGRP_ID="0"
    PAGENAME="CatalogItemDisplay.jsp"
    STOREENT_ID="10151"
    RANK="0"
 />
<dispcgprel
    CATGROUP_ID="0"
    DEVICEFMT_ID="-1"
    DISPCGPREL_ID="10001"
    MBRGRP_ID="0"
    PAGENAME="CatalogCategories.jsp"
    STOREENT_ID="10151"
    RANK="0"
 />
<invadjcode
    ADJUSTCODE="PCNT"
    INVADJCODE_ID="10001"
    MARKFORDELETE="0"
    STOREENT_ID="10151"
 />
<invadjcode
    ADJUSTCODE="SPLG"
    INVADJCODE_ID="10002"
    MARKFORDELETE="0"
    STOREENT_ID="10151"
 />
<invadjcode
    ADJUSTCODE="DISC"
    INVADJCODE_ID="10003"
    MARKFORDELETE="0"
    STOREENT_ID="10151"
 />
<rtnreason
    REASONTYPE="C"
    RTNREASON_ID="10001"
```

```
    STOREENT_ID="10151"
    MARKFORDELETE="0"
    CODE="WPR"
 />
<rtnreason
    REASONTYPE="B"
    RTNREASON_ID="10002"
    STOREENT_ID="10151"
    MARKFORDELETE="0"
    CODE="DEF"
 />
<rtnreason
    REASONTYPE="M"
    RTNREASON_ID="10003"
    STOREENT_ID="10151"
    MARKFORDELETE="0"
    CODE="ERR"
 />
<rtnreason
    REASONTYPE="M"
    RTNREASON_ID="10004"
    STOREENT_ID="10151"
    MARKFORDELETE="0"
    CODE="WPS"
 />
</store-asset>
```

- オプション 2:
  a. fulfillment.xml と store.xml ファイルをマージして、以下に示す ffmcenter 要素がストア要素の前にくるようにします。

     ```
     <ffmcenter
        FFMCENTER_ID="@ffmcenter_id_1"
        MEMBER_ID="&MEMBER_ID;"
        NAME="ToolTech Home"
        DEFAULTBOOFFSET="0"
         MARKFORDELETE="0"
     />
     ```

  b. マージされたファイルに対して ID リゾルバーを実行します。

この例は、 WebSphere Commerce Server データベースにストア資産データを初めてロードするというシナリオに基づいているので、上記の例ではロード方式を使用しました。

ストア資産をロードした後に XML 文書内の特定の要素を変更する必要が生じる場合は、更新方式を使用してこれを行えます。更新方式では新規の ID が割り当てられないので、ロード方式と比べて実行時間がかかりません。更新方式では、データベース・クエリーを実行して ID を探し出し、 ID が検出されない場合にはエラーが報告されます。

入力 XML ファイルの中に、データベースにすでに存在する要素と存在しない要素が混在する場合には、混合方式を使用します。混合方式では、まずデータベースの検索が行われ、レコードが検出されない場合に ID が要素に割り当てられます。混在しているかどうか分からない場合は、混合方式を使用してください。混合方式と比べてロード方式と更新方式の方が高いパフォーマンスを示しますが、混合方式を使用して生成された解決 XML ファイルの方が、ロード時にエラーが発生する可能性は低くなります。

解決方式の選択の詳細については、 251 ページの 『処理に関する推奨事項』を参照してください。

ID リゾルバーで使用するプロパティー・ファイルの指定

-propfile パラメーターを使用すると、ID リゾルバーの解決方法を変更することができます。デフォルトのプロパティー・ファイルは IdResolveKeys.properties ですが、これを変更することもできますし、あるいは ID Resolve コマンドを呼び出す際に独自のファイルを指定することも可能です。

- Windows AIX Solaris Linux

  IdResolveKeys.properties は、以下のディレクトリーにあります。

  - NT *drive*:¥WebSphere¥CommerceServer¥properties
  - 2000 *drive*:¥Program Files¥WebSphere¥CommerceServer¥properties
  - AIX /usr/WebSphere/CommerceServer/properties
  - Solaris Linux /opt/WebSphere/CommerceServer/properties

  ID リゾルバーの実行時にこのファイルが現行ディレクトリーに存在しない場合、これをクラスパスの環境変数で定義されているディレクトリーに配置することができます。このファイルの絶対パスを指定することもできます。

- 400

  IdResolveKeys.properties を変更するには、/QIBM/ProdData/WebCommerce/properties ディレクトリーからこれをコピーし、/*instroot*/xml ディレクトリーに保管してから、この新規ファイルに対して必要な変更を加えます。

  注: 上記のディレクトリーは、 RESWCSID コマンドで使用するクラスパスにあります。

プロパティー・ファイルの指定は、内部別名の使用よりも優先されます。

以下は、store.xml ファイルの別のサンプル XML フラグメントです。

```
<store
   STORE_ID="@storeent_id_1"
   DIRECTORY="ToolTech"
   FFMCENTER_ID="@ffmcenter_id_1"
   LANGUAGE_ID="&en_US;"
   STOREGRP_ID="-1"
   ALLOCATIONGOODFOR="43200"
   BOPMPADFACTORr="0"
   DEFAULTBOOFFSET="2592000"
   FFMCSELECTIONFLAGS="0"
   MAXBOOFFSET="7776000"
   REJECTEDORDEXPIRY="259200"
   RTNFFMCTR_ID="@ffmcenter_id_1"
   PRICEREFFLAGS="0"
   STORETYPE="B2B"
/>
```

c:¥WebSphere¥CommerceServer¥runtime¥test¥data¥myPropFile として -propfile を指定して ID リゾルバーを実行し、指定したファイル myPropFile.properties に以下のエントリーが含まれる場合があります。

```
NAMEDELIMETER=@
SELECTDELIMETER=:
FFMCENTER=@FFMCENTER_ID@MEMBER_ID:FFMCENTER_ID MEMBER_ID
```

この場合、ストア要素が処理されるときに、 ID リゾルバーはデータベースを参照して FFMCENTER_ID および MEMBER_ID の where 文節を持つ FFMCENTER テーブルを探します。その後、この値に対して戻される索引を使用して、FFMCENTER_ID の ID を解決します。

このコマンドの使用については、 246 ページの 『ID Resolve コマンド』を参照してください。

データ・ロードの例

必要に応じて XML ファイルの ID を解決したら、 WebSphere Commerce Server データベースにデータをロードする準備が整ったことになります。

注: ソース XML ファイルの中に、アットマーク (@) またはアンパーサンド (&) 記号が先頭に付いているワードがないことを確認してください。これらの記号のいずれかの存在は、 XML ファイルをロードする準備が整っていないことを示すものとなります。

このセクションで説明するデータ・ロードの例では、 271 ページの 『ID 解決の例』で解決した `fulfillment1.xml` ファイルを使用します。

WebSphere Commerce Server データベースにデータをロードするには、以下の例に従って Load コマンドを実行します。

1. 270 ページの 『ローダー・パッケージ・コマンドおよびスクリプトの使用』でリストされている適切な Load コマンドまたはスクリプトを含むディレクトリーが、パスに含まれていることを確認します。

   この例では、`massload.cmd` を使用します。
2. 作業ディレクトリーを作成します。

   この例では、 271 ページの 『ID 解決の例』の例で作成した `c:¥WebSphere¥CommerceServer¥runtime¥test¥data` というディレクトリーを使用します。
3. ローダーが検出できる場所に入力 XML ファイルがあることを確認してください。

   この例では、 271 ページの 『ID 解決の例』で作成した `fulfillment1.xml` 出力ファイルが `c:¥WebSphere¥CommerceServer¥runtime¥test¥data` にあることを確認してください。
4. Windows コマンド・プロンプトで、以下のコマンドを入力します。

   ```
   cd c:¥WebSphere¥CommerceServer¥runtime¥test¥data
   ```
5. 必ず WebSphere Commerce Server データベースをバックアップしてください。
6. Windows コマンドで、以下のコマンドを入力します。

   ```
   massload -dbname mall -dbuser db2admin -dbpwd db2admin -infile
   c:¥WebSphere¥CommerceServer¥runtime¥test¥data¥fulfillment1.xml
   -method sqlimport -commitcount 50
   ```

ロードする要素の数は 50 未満ですが、この例では -commitcount に 50 という値を指定しています。これは、パフォーマンス上の理由です。デフォルトでは、コミッ

ト数は 1 になっています。このデフォルトを使用すると、各レコードごとのコミット操作がデータベースに書き込まれることになります。上記の例でこの数を 50 に設定することによって、ロードが成功した場合にのみデータベースの入出力が行われ、エラーが発生した場合にはデータベースに何も書き込まれないようにすることができます。ただし、大量のデータをロードする場合には、コミット数に要素の数より高い値を設定しないようお勧めします。これには次のような理由があります。

- コミット数の値を高くすると、多くのメモリーが消費される。
- こうすることで、少なくともいくつかのデータをデータベースに書き込むことができる。 -maxerror の値に応じて -commitcount の値を小さくすることによって、エラーの最大数を超過してツールが終了する前に、いくつかのデータをデータベースに書き込むことができます。 -maxerror のデフォルト値は 1 です。

-noprimary オプションはデフォルトの error になり、 1 次鍵が欠落しているときにツールがエラーを報告し、終了します。 -schemaname オプションは省略され、デフォルトのデータベース・ユーザー名になります。

これらの例では、ストア・サービスで使用される順序、また280 ページの 『データのロードの順序』で説明されている順序でストア資産をロードしていないため、271 ページの 『ID 解決の例』で解決される store2.xml ファイルは、いくつかのテーブルの保全性の制約を違反する場合があります。ロード方式の使用時に store2.xml をロードしようとする場合、制約違反によってデータベースは保留状態になります。しかしながら、Load コマンドを使用したこの例では、説明を簡単にするために、fulfillment.xml の解決済みバージョン (その外部鍵は、サンプル・ストアで定義されている MEMBER_ID の外部鍵のみ) に基づいています。この例では、271 ページの 『ID 解決の例』で出力された解決済み fulfillment1.xml ファイルをロードし、 sqlimport 方式を使用しています。ロード可能 XML ファイルの内容がクリーンかどうか分からない場合は、この例の中で示されている sqlimport 方式を使用して、データベースを変更することなく、またデータベースの保全性を危険にさらすことなく、データベース制約の違反が報告されるようにしてください。

このコマンドの使用については、 254 ページの 『Load コマンド』を参照してください。

第 29 章 WebSphere Commerce データ・グループのロード

Catalog Manager ローダー・パッケージを使用して WebSphere Commerce データ資産をロードできます。ストア・データ資産を発行する前に、資産をすべて作成してストア・アーカイブ・ファイルにパッケージしておくことを望まない場合は、このセクションで説明する一連の手順に従って、データを WebSphere Commerce データベースにロードできます。

この章では、まずストア・データ・グループについて、そしてグループの決定方法について説明します。次に、これらのデータ・グループを WebSphere Commerce データベースにロードするプロセスについて説明します。この節を読む前に、243 ページの『第 28 章 ストア・データのロードの概要』の情報を確認しておいてください。背景や、ローダー・パッケージを使用してデータをロードするときに知っておく必要のあることを理解できます。

データ・グループ

データ作成とロード・プロセスを単純化するため、ストア・データは複数のグループに分けられます。これらのデータ・グループは、論理的に関連したテーブル群で構成されます。データ・グループを編成する順序は、ロードを実行する上で重要です。オブジェクトが存在していなければ、オブジェクト同士の関連をロードすることはできないためです。

ストア・データ群全体をロードするには、280 ページの『データのロードの順序』に従う必要があります。ストア・データのグループを 1 つロードするには、この部分的なデータが論理的に完全であることを確認する必要があります。たとえば、ストア・アーカイブを発行するときに、カタログ・データ資産以外のすべてのデータ資産を発行するとします。カタログに従属するデータ (在庫、価格リスト、および一部の配送データと税データ) を含むカタログ・データは、発行しないでおきます。省略したデータを発行するには、それが論理的に完全なカタログ・データであることを確認する必要があります。すなわち、基本項目、カタログ項目、属性などがなければなりません。従属データも発行しなければなりません。この従属カタログ・データ群は、それ自体の間でも論理的に完全でなければなりません。つまり、SKU ごとに、適切な価格、在庫、配送、および税が定義されている必要があります。この場合、論理的に完全な、関連するカタログ・データを、集合的に*カタログ・データ・グループ* と言います。

このような理由に基づき、WebSphere Commerce データを別のデータ・グループに分けることができます。各データ・グループは、327 ページの『付録 E. データ・グループ』で説明されているとおり、WebSphere Commerce データベース・テーブルで構成されています。テーブル・リストは、WebSphere Commerce サンプル・ストアを基にしたものです。各データ・グループのテーブルのリストは完全なものではなく、一般的な指針に過ぎないことに注意してください。ストアに特有の必要に基いて、組み込んだり除外したりしなければならないテーブルがあるかもしれません。

データのロードの順序

データを正常にロードするために従うべき特定の順序があります。あるデータ・グループを WebSphere Commerce データベースにロードする前に、そのデータ・グループの外部従属関係を満たしておく必要があります。所定のデータ・グループをロードする前に、その外部従属関係として定義されたデータ・グループをロードしておかなければなりません。外部従属関係および関連したテーブルのリストについては、327 ページの『データ・グループの従属関係』を参照してください。データ・グループのロード順序については、以下の指針に従ってください。

1. ブートストラップ・データにのみ従属するデータ・グループ。
   a. **フルフィルメント**・データを最初にロードしなければなりません。続くデータ・グループはすべて、このグループに定義されたデータへの直接的または間接的な外部従属関係にあるからです。
   b. **組織**データ。とは言え、組織データに従属するデータ・グループはほかにありません。
   c. **アクセス・コントロール**・データは、ブートストラップ・データしか必要としません。このグループに定義されたデータとデータ従属関係にあるグループは、他にありません。したがって、このグループはいつロードしてもかまわないということです。
2. フルフィルメント・データにのみ従属するデータ・グループ。
   a. **ストア**・データ。
3. ストア・データ・グループに従属するデータ・グループ。以下のデータ・グループを任意の順序でロードできます。
   a. **キャンペーン**・データ。
   b. **コマンド**・データ。
   c. **通貨**データ。
   d. **ポリシー**・データ。
   e. **配送**データ。
   f. **税**データ。
4. 他のデータ・グループ
   a. **カタログ**・データ。
   b. **ストア・デフォルト**・データには、配送および契約データ・グループで定義されたデータへの外部従属関係があります。しかし、ストア・デフォルト・データをロードする前に、配送および契約データが存在していなければならないわけではありません。配送データも契約データも存在していない場合、このデータ・グループにデータを読み込む必要はありません。
   c. **契約**データ。契約データベース・テーブルは、直接にはロードされません。詳細は、331 ページの『契約データ・グループのロード』を参照してください。他のデータ・グループをロードした後で、契約データをロードすることをお勧めします。

データ・グループのロード

単一のデータ・グループまたはストア全体の XML データを WebSphere Commerce データベースにロードするには、以下のようにします。

1. 313 ページの『付録 B. データの作成』の情報を確認します。
2. 327 ページの『付録 E. データ・グループ』の情報を確認します。どの WebSphere Commerce ファイルとデータベース・テーブルが影響を受けるかを確認しておく必要があります。
3. ロード・プロセスを計画し、単一のデータ・グループをロードするか、それともストアのデータ群全体をロードするかを決定します。単一のデータ・グループをロードするにせよ、ストア全体をロードするにせよ、基本的なプロセスは変わりません。次のステップでは、それぞれのロード・プロセスに合わせて、以下のファイルを使用または作成します。
   a. 選択するデータ・グループに応じて 1 つ以上のデータ・ファイル。ストア全体をロードする場合、作成済みの資産ファイルがすべて必要です。たとえば、ストア・データ・グループ資産をロードする場合、`store.xml` ファイルと、ストアがサポートするロケール用に別の `store.xml` ファイルが必要です。ストア・データ群全体をロードする場合、*`data group name`*`.xml` と、ストアのデータ・グループごとに個別のロケール・ファイルが必要です。そうしたデータ・ファイルの例が WebSphere Commerce サンプル・ストアに付属しています。次のディレクトリー内にあります。
      - NT *`drive`*`:¥WebSphere¥CommerceServer¥samplestores¥NewFashion¥`*`locale`*`¥data`
      - 2000 *`drive`*`:¥Program Files¥WebSphere¥CommerceServer¥samplestores¥NewFashion¥`*`locale`*`¥data`
      - AIX `/usr/WebSphere/CommerceServer/samplestores/NewFashion/`*`locale`*`/data`
      - Solaris Linux `/opt/WebSphere/Commerce/samplestores/NewFashion/`*`locale`*`/data`
      - 400 `/qibm/proddata/WebCommerce/samplestores/NewFashion/`*`locale`*`/data`
   b. XML データ・ファイル群を正しい順序で 1 つに連結する、新しい XML ファイル。このファイルをマスター *XML ファイル* と呼びます。
   c. データ・グループの XML ファイルに必要なすべてのデータ・タイプを定義する、新しい DTD ファイル。このファイルをマスター *DTD ファイル* と呼びます。以下に示すファイルや、WebSphere Commerce に付属の `wcs.dtd` ファイルなどの、他の DTD ファイルを組み込まなければならない場合もあります。
   d. 外部従属関係を定義する 2 番目の DTD ファイル。
   e. すべての WebSphere Commerce テーブルの定義を含む 3 番目の DTD ファイル。 `wcs.dtd` ファイルは WebSphere Commerce にすでに存在しており、次のディレクトリーにあります。
      - NT *`drive`*`:¥WebSphere¥CommerceServer¥schema¥xml¥`

- 2000 *drive*:¥Program Files¥WebSphere¥CommerceServer¥schema¥xml¥
- AIX /usr/WebSphere/CommerceServer/schema/xml/
- Solaris Linux /opt/WebSphere/CommerceServer/schema/xml/
- 400 /QIBM/ProdData/WebCommerce/schema/xml/

f. 4 番目の DTD ファイル。このファイルを作成する必要があるのは、データ・ファイルに、wcs.dtd ファイルで定義されていないルート・エレメント (複数可) が含まれる場合だけです。そのようなケースでは、エレメントを定義する必要があります。続く例では、未定義エレメントを含むデータ・ファイルを使用しないことに注意してください。

4. 本書の前半のデータ資産について扱った章で説明されているとおりに、データ・ファイルを作成します。すでにデータ資産について扱った章を読み終えているのであれば、これらのデータ・ファイルはすでに存在しているかもしれないので、注意してください。データ・ファイルが連結されるときに競合が生じてしまう可能性があるため、データ・ファイルの先頭には DTD 宣言やページ・ディレクティブを含めないようにします。データ・ファイルの言語が複数に渡る場合、各ファイルの先頭は <?xml encoding = *locale specific encoding*> でなければなりません。たとえば、英語のデータ・ファイルでは <?xml encoding = "UTF-8"?> を指定し、フランス語のデータ・ファイルでは <?xml encoding = "ISO-8859-1"?> を指定します。簡単にするために、ルート・エレメントをまったく作成しないこともできます。ルート・エレメントを持たなければならない唯一のファイルは、マスター XML ファイルです。
5. マスター XML ファイルは、ロードするデータ・グループごと、またはストア・データ群全体ごとに作成します。それらのマスター・ファイルには、さまざまな XML ファイルを 1 つ以上のデータ・グループに組み込むための、ステートメントや外部参照エンティティーが含まれていることがあります。外部参照エンティティーを使用することにより、ファイルの内容を XML ファイルに組み込むことができます。 XML パーサーは、外部参照の代わりに、外部参照エンティティーによって参照されるファイルの内容を置き換えるかもしれません。

   a. 次の例は、単一のストア・データ・グループをロードする場合の例です。これを指針として使用し、この抽出部分を基にしてマスター・ファイルを作成することができます。

```
<?xml version="1.0"?>
<!DOCTYPE import SYSTEM "store-assets.dtd">
<import>
&store.xml;
/fr_FR/&store.xml;
/en_US/&store.xml;
</import>
```

   ここで、

   - import は、XML 文書のルート・エレメントを指します。ルート・エレメントは、WebSphere Commerce に付属する wcs.dtd ファイルですでに定義されています。ルート・エレメントには、WebSphere Commerce データベース内のすべてのテーブルの適切な定義が含まれています。しかし、WebSphere Commerce スキーマをカスタマイズした場合には、別のルート・エレメントを使用しなければならない場合があります。

- store-assets.dtd は、次のステップで作成するマスター DTD ファイルの名前を指します。
- &store.xml は、このデータ・グループ用にデータをロードするときの XML ファイルの名前を指します。この名前は、データ・グループごとにすでに作成されているデータ資産ファイルに合わせて変更されます。
- *path*_store.xml は、ストアがサポートする言語ごとに必要です。ストアが単一の言語であれば、1 つのファイルだけを参照することになります。ストアが 2 つ以上の言語をサポートしている場合は、各言語ごとにロケール固有のファイルが必要です。上記の抽出部分は、ストアが英語とフランス語をサポートしていることを前提としたものです。

b. 次の例は、ストア・データ群全体をロードする場合の例です。これを指針として使用し、この抽出部分を基にしてマスター・ファイルを作成することができます。

```
<?xml version="1.0"?>
<!DOCTYPE import SYSTEM "all-store-assets.dtd">
<import>
<!Fulfillment data group -->
&fulfillment.xml;

<!-- Store data group -->
&store.xml;
en_US/&store.xml;
fr_FR/&store.xml;

<!-- Tax data group -->
&tax.xml;
en_US/&tax.xml;
fr_FR/&tax.xml;
&taxfulfill.xml;

<!-- Shipping data group -->
&shipping.xml;
en_US/&shipping.xml;
fr_FR/&shipping.xml;
&shipfulfill.xml;

<!-- Catalog data group -->
&catalog.xml;
en_US/&catalog.xml;
fr_FR/&catalog.xml;
&storecatalog.xml;
&storefulfill.xml;
&offering.xml;
&store-catalog-tax.xml;
&store-catalog-shipping.xml;

<!-- Currency data group -->
&currency.xml;
en_US/&currency.xml;
fr_FR/&currency.xml;

<!-- Campaign data group -->
&campaign.xml;
en_US/&campaign.xml;
fr_FR/&campaign.xml;

<!-- Business policy data group -->
&businesspolicy.xml;
en_US/&businesspolicy.xml;
fr_FR/&businesspolicy.xml;
```

```
<!-- Access control data group -->
&accesscontrol.xml;
en_US/&accesscontrol.xml;
fr_FR/&accesscontrol.xml;

<!-- Other data groups -->
&command.xml;
&store-default.xml;
</import>
```

ここで、

- import は、XML 文書のルート・エレメントを指します。ルート・エレメントは、WebSphere Commerce に付属する wcs.dtd ファイルですでに定義されています。ルート・エレメントには、WebSphere Commerce データベース内のすべてのテーブルの適切な定義が含まれています。しかし、WebSphere Commerce スキーマをカスタマイズした場合には、別のルート・エレメントを使用しなければならない場合があります。
- all-store-assets.dtd は、次のステップで作成するマスター DTD ファイルの名前を指します。
- コメント化されたテキストは、ストアの別のデータ・グループを表しています。
- *data group name*.xml は、データをロードするもととなるメイン XML ファイルの名前を指しています。この名前は、データ・グループごとにすでに作成されているデータ資産ファイルに合わせて変更されます。
- *path_data group name*.xml は、ストアがサポートする言語ごとに必要です。ストアが単一の言語であれば、1 つのファイルだけを参照することになります。ストアが 2 つ以上の言語をサポートしている場合は、各言語ごとにロケール固有のファイルが必要です。上記の抽出部分は、ストアが英語とフランス語をサポートしていることを前提としたものです。

6. 上記の参照エンティティーや、データ・グループで必要な他の DTD ファイルを定義する、マスター DTD ファイルを作成します。
   a. 次の例は、単一のストア・データ・グループをロードする場合の例です。これを指針として使用し、あらゆるデータ・グループ用のマスター DTD ファイルを作成することができます。

```
<!ENTITY % wcs.dtd SYSTEM "drive:¥Program Files¥WebSphere¥CommerceServer¥
schema¥xml¥wcs.dtd">
%wcs.dtd;

<!ENTITY % NonStoreForeignKeys.dtd SYSTEM "NonStoreForeignKeys.dtd">
%NonStoreForeignKeys.dtd;
<!ENTITY store.xml SYSTEM "store.xml">
<!ENTITY en_US_store.xml SYSTEM "en_US/store.xml">
<!ENTITY fr_FR_store.xml SYSTEM "fr_FR/store.xml">
```

ここで、

- wcs.dtd は、ドメインの外側で定義されたデータを含む DTD ファイルを指しています。
- NonStoreForeignKeys.dtd は、ルート・エレメント以外のエレメントを定義する DTD ファイルを指しています。

注: パスが正しく指定されていることを確認してください。この例では、ファイルは、マスター DTD ファイルと同じディレクトリーにあります。

- store.xml、en_US_store.xml、および fr_FR_store.xml は、外部参照エンティティーです。
- store.xml は、ストア・データ・グループのデータ・ファイルです。
- *path*_store.xml は、ロケール固有のデータ・ファイル群です。ここでは、ストアが英語とフランス語をそれぞれサポートしていることを前提としています。ファイル名を、絶対パスか相対パスで修飾する必要があります。

b. 次の例は、ストア・データ群全体のためのものです。これを指針として使用し、この例を基にしてマスター DTD ファイルを作成することができます。

```
<!ENTITY % wcs.dtd SYSTEM "drive:¥Program Files¥WebSphere¥CommerceServer¥
schema¥xml¥wcs.dtd">
%wcs.dtd;

<!ENTITY % NonStoreForeignKeys.dtd SYSTEM "NonStoreForeignKeys.dtd">
%NonStoreForeignKeys.dtd;
<!ENTITY fulfillment.xml SYSTEM "data/fulfillment.xml">
<!ENTITY en_US_fulfillment.xml SYSTEM "data/en_US/fulfillment.xml">
<!ENTITY fr_FR_fulfillment.xml SYSTEM "data/fr_FR/fulfillment.xml">

<!ENTITY store.xml SYSTEM "data/store.xml">
<!ENTITY en_US_store.xml SYSTEM "data/en_US/store.xml">
<!ENTITY fr_FR_store.xml SYSTEM "data/fr_FR/store.xml">

<!ENTITY tax.xml SYSTEM "data/tax.xml">
<!ENTITY en_US_tax.xml SYSTEM "data/en_US/tax.xml">
<!ENTITY fr_FR_tax.xml SYSTEM "data/fr_FR/tax.xml">
<!ENTITY taxfulfill.xml SYSTEM "data/taxfulfill.xml">

<!ENTITY shipping.xml SYSTEM "data/shipping.xml">
<!ENTITY en_US_shipping.xml SYSTEM "data/en_US/shipping.xml">
<!ENTITY fr_FR_shipping.xml SYSTEM "data/fr_FR/shipping.xml">
<!ENTITY shipfulfill.xml SYSTEM "data/shipfulfill.xml">

<!ENTITY catalog.xml SYSTEM "data/catalog.xml">
<!ENTITY en_US_catalog.xml SYSTEM "data/en_US/catalog.xml">
<!ENTITY fr_FR_catalog.xml SYSTEM "data/fr_FR/catalog.xml">
<!ENTITY store-catalog.xml SYSTEM "data/store-catalog.xml">
<!ENTITY storefulfill.xml SYSTEM "data/storefulfill.xml">
<!ENTITY offering.xml SYSTEM "data/offering.xml">
<!ENTITY store-catalog-tax.xml SYSTEM "data/store-catalog-tax.xml">
<!ENTITY store-catalog-shipping.xml SYSTEM "data/store-catalog-shipping.xml">

<!ENTITY currency.xml SYSTEM "data/currency.xml">
<!ENTITY en_US_currency.xml SYSTEM "data/en_US/currency.xml">
<!ENTITY fr_FR_currency.xml SYSTEM "data/fr_FR/currency.xml">

<!ENTITY campaign.xml SYSTEM "data/campaign.xml">
<!ENTITY en_US_campaign.xml SYSTEM "data/en_US/campaign.xml">
<!ENTITY fr_FR_campaign.xml SYSTEM "data/fr_FR/campaign.xml">

<!ENTITY businesspolicy.xml SYSTEM "data/businesspolicy.xml">
<!ENTITY en_US_businesspolicy.xml SYSTEM "data/en_US/businesspolicy.xml">
<!ENTITY fr_FR_businesspolicy.xml SYSTEM "data/fr_FR/businesspolicy.xml">

<!ENTITY accesscontrol.xml SYSTEM "data/accesscontrol.xml">
<!ENTITY en_US_accesscontrol.xml SYSTEM "data/en_US/accesscontrol.xml">
<!ENTITY fr_FR_accesscontrol.xml SYSTEM "data/fr_FR/accesscontrol.xml">
```

```
<!ENTITY command.xml SYSTEM "data/command.xml">
<!ENTITY store-defaults.xml SYSTEM "data/store-defaults.xml">
```

ここで、

- wcs.dtd は、ドメインの外側で定義されたデータを含む DTD ファイルを指しています。
- NonStoreForeignKeys.dtd は、ルート・エレメント以外のエレメントを定義する DTD ファイルを指しています。

  注: パスが正しく指定されていることを確認してください。この例では、ファイルは、マスター DTD ファイルと同じディレクトリーにあります。

- store.xml、en_US_store.xml、および fr_FR_store.xml は、外部参照エンティティーです。
- *data group name*.xml は、データをロードするもととなるメイン XML ファイルの名前を指しています。この名前は、データ・グループごとにすでに作成されているデータ資産ファイルに合わせて変更されます。
- *path_data group name*.xml ファイル群は、ストアがサポートする言語ごとに必要です。ストアが単一の言語であれば、1 つのファイルだけを参照することになります。ストアが 2 つ以上の言語をサポートしている場合は、各言語ごとにロケール固有のファイルが必要です。上記の抽出部分は、ストアが英語とフランス語をサポートしていることを前提としたものです。

7. それぞれのデータ・グループは、外部従属関係を持つ場合があるため、自分のドメイン、つまり自分のデータのセットの外側で定義された情報を必要とします。このデータを DTD ファイルに指定できます。たとえば、ストア・データ・グループには、以下のような外部従属関係があります。

```
  bootstrap.LANGUAGE.LANGUAGE_ID, bootstrap.MEMBER.MEMBER_ID,
bootstrap.SETCURR.SETCURR_ID, fulfillment.FFMCENTER.FFMCENTER_ID
```

データ・グループまたはストア・データ群全体をロードする場合には、WebSphere Commerce データベースからの以下のデータを定義する必要があります。このデータを使用するには、対応する XML エンティティー参照に従ってください。たとえば、ffmcenter_id エンティティーによって定義されたデータを使用する場合、 XML ファイルに &ffmcenter_id; と書き込みます。次の例は、ストア資産用の例です。これを指針として使用し、この抽出部分を基にして DTD ファイルを作成することができます。

```
<!ENTITY en_US "-1">
<!ENTITY fr_FR "-2">
<!ENTITY de_DE "-3">
<!ENTITY it_IT "-4">
<!ENTITY es_ES "-5">
<!ENTITY pt_BR "-6">
<!ENTITY zh_CN "-7">
<!ENTITY zh_TW "-8">
<!ENTITY ko_KR "-9">
<!ENTITY ja_JP "-10">
<!ENTITY MEMBER_ID "-2000">
<!ENTITY ffmcenter_id "10001">
```

ここで、

- MEMBER_ID は、ストアの所有者を識別する内部参照番号です。

- `ffmcenter` は、ストアの配送センターの参照番号です。ストアでは 2 つ以上の配送センターを使用できるため、 DTD ファイルでは 2 つ以上を定義できます。
- *`locale`* は、ロケールごとの WebSphere Commerce 参照番号です (国と言語または地域と言語で示されます)。この値は、LANGUAGE データベース・テーブルにあります。

注: 既存のストア・アーカイブをデータ・グループに分ける場合には、別名 `@ffmcenter_id` へのすべての参照を、対応するエンティティー参照 `&ffmcenter_id;` に置き換えるようにしてください。

8. すべてのデータ・ファイルを作成したら、246 ページの 『ID Resolve コマンド』で説明されているように、マスター XML ファイルに対して IDResolve コマンドを実行してデータを解決します。
9. 254 ページの 『Load コマンド』で説明されているように、解決済みのデータ・ファイルに対してローダー・コマンドを実行します。

第 30 章 アカウント、契約、および商品セットの発行

ローダー・パッケージでは一部のデータベース資産、(アカウント、契約、および商品セット) をロードできません。これらのデータベース資産は、 227 ページの『第 26 章 ストア全体の発行』で説明されているストア全体の発行オプションの一部として、ストア・サービスを使用するか、またはコマンド行から発行できます。あるいは該当するコマンドを使用して、アカウント、契約、または商品セットを発行することもできます。それらのコマンドは次のとおりです。

- ProductSetPublish— 商品セット・データを商品セット・テーブルに発行します。
- AccountImport — ストア・アーカイブ内の `businessaccount.xml` ファイルからアカウントを作成します。
- ContractImportApprovedVersion — ストア・アーカイブ内の `contract.xml` ファイルから契約を作成します。

ビジネス・アカウント資産は、WebSphere Commerce で提供されている一部のサンプル・ストア・アーカイブに、 XML ファイルの形式で含まれています。現在、商品セットはサンプル・ストア・アーカイブに含まれていません。しかし、ビジネス・アカウント資産用の XML ファイルを作成するよりも、用意されているツールを使用して、商品セットとビジネス・アカウント資産の両方を作成することをお勧めします。用意されているツールを使用して、これらの資産を作成することの詳細については、 WebSphere Commerce オンライン・ヘルプを参照してください。サンプル・ストア・アーカイブで提供されている該当する XML ファイルを発行することにした場合、あるいは自分で作成することにした場合のために、ビジネス・アカウントおよび商品セットを発行する方法の説明がこの後のセクションにあります。

注: カタログ資産は、アカウント、契約、および商品セットを発行する前に、発行する必要があります。ストア・サービスまたはコマンド行を使用してこれらの資産を発行する場合、カタログ・オプションを選択しているか、または発行されているカタログがすでにストアにあることを確認してください。該当するコマンドを使用してこれらの資産を発行する場合、カタログ資産をデータベースにすでにロードしていることを確認してください。

ストア・サービスまたはコマンド行を使用した、アカウント、契約、および商品セットの発行

ストア・サービス、またはコマンド行の発行ユーティリティーを使用して、アカウント、契約、および商品セットを発行できます。ストア・サービスまたはコマンド行を使用して、アカウント、契約、および商品セットを発行するには、資産がストア・アーカイブ形式でパッケージされていなければなりません。ストアフロント資産をストア・アーカイブとしてパッケージする処理については、 215 ページの『第 6 部 ストアのパッケージ化』を参照してください。

ストア・サービスやコマンド行の発行ユーティリティーでは、ストア・アーカイブ内のすべてのタイプの資産を発行する (ストアフロント資産、ストア・データ資産、およびリソース・バンドルを含む) か、特定のタイプの資産だけを発行するか

を選択できます。ストア・サービスやコマンド行を使用した資産発行の詳細な手順については、 WebSphere Commerce オンライン・ヘルプを参照してください。

コマンドを使用した、アカウント、契約、および商品セットの発行

資産をストア・アーカイブとしてパッケージしない場合、以下の該当するコマンドを使用して、アカウント、契約、および商品セットを発行できます。

- ProductSetPublish— 商品セット・データを商品セット・テーブルに発行します。
- AccountImport — ストア・アーカイブ内の `businessaccount.xml` ファイルからアカウントを作成します。
- ContractImportApprovedVersion — ストア・アーカイブ内の `contract.xml` ファイルから契約をインポートします。

アカウント資産の発行

アカウント資産を発行するには、以下のようにします。

1. 次のようにして、VIEWREG テーブルのアカウント表示ビューを登録します。
   - INSERT INTO viewreg (viewname, devicefmt_id, storeent_id, interfacename, classname, properties, internal, https), VALUES ('AccountDisplayView', -1, *store_id*, 'com.ibm.commerce.command.ForwardViewCommand', 'com.ibm.commerce.command.HttpForwardViewCommandImpl', 'docname=*the JSP file that displays after the account is imported*.jsp', 0, 0)
2. 管理コンソールを使用して、ビュー・レジストリーを最新表示するか、WebSphere Commerce インスタンスを再始動します。詳しくは、WebSphere Commerce オンライン・ヘルプを参照してください。
3. `businessaccount.xml` を以下のディレクトリーにコピーします。
   - NT `drive:¥WebSphere¥AppServer¥installedApps¥WC_Enterprise_App_instancename.ear¥xml¥trading`
   - 2000 `drive:¥Program Files¥WebSphere¥AppServer¥installedApps¥WC_Enterprise_App_instancename.ear¥xml¥trading`
   - AIX `/usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_instancename.ear/xml/trading`
   - Solaris `/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_instancename.ear/xml/trading`
   - Linux `/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_instancename.ear/xml/trading`
   - 400 `/QIBM/UserData/WebASAdv4/WASinstancename/installedApps/WC_Enterprise_App_instancename.ear/xml/trading`
4. `businessaccount.xml` をオープンして、次のように変更します。
   - すべての &STORE_IDENTIFIER; をストアのストア ID に置き換える。
   - すべての &MEMBER_IDENTIFIER; をストアのメンバー ID に置き換える。
5. ファイルを保管してクローズします。
6. ブラウザーで、以下のように入力します。

- https://*hostname*/webapp/wcs/stores/servlet/AccountImport?fileName=businessaccount.xml&store_id=store_id&flowid=1

契約資産の発行

契約資産を発行するには、以下のようにします。

1. contract.xml を以下のディレクトリーにコピーします。
   - NT drive:¥WebSphere¥AppServer¥installedApps¥WC_Enterprise_App_*instancename*.ear¥xml¥trading
   - 2000 drive:¥Program Files¥WebSphere¥AppServer¥installedApps¥WC_Enterprise_App_*instancename*.ear¥xml¥trading
   - AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/xml/trading
   - Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/xml/trading
   - Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*instancename*.ear/xml/trading
   - 400 /QIBM/UserData/WebASAdv4/WASinstancename/installedApps/WC_Enterprise_App_*instancename*.ear/xml/trading
2. contract.xml をオープンして、次のように変更します。
   - すべての &STORE_IDENTIFIER; をストアのストア ID に置き換える。
   - すべての &MEMBER_IDENTIFIER; をストアのメンバー ID に置き換える。
3. ファイルを保管してクローズします。
4. ブラウザーで、以下のように入力します。
   - https://*hostname*/webapp/wcs/tools/servlet/ContractImportApprovedVersion?fileName=contract.xml&targetStoreId=store_id&URL=ContractDisplay

商品セットの発行

商品セット資産を発行するには、以下のようにします。

- ブラウザーで、以下のように入力します。
  - https://hostname/webapp/wcs/stores/servlet/ProductSetPublish?productSetId=*&URL=*your re-direction URL*

第 8 部 ストアへの WebSphere Commerce フィーチャーの追加

第 31 章 ストアへのカスタマー・ケアの追加

WebSphere Commerce のカスタマー・ケア機能は、 Lotus Sametime サーバーを使用して、同期テキスト・インターフェースにより、リアルタイムの顧客サービス・サポートを提供します。ストア内でカスタマー・ケアが使用可能になっている場合、顧客はストアに入り、リンクをクリックして、顧客サービス担当者 (CSR) に接続します。そして、顧客はインターネットを介して CSR と通信を行うことができます。

注: この章では、ストア内でカスタマー・ケアを使用可能にする方法について説明します。しかし、ストア内でカスタマー・ケアを使用可能にする前に、最初に Sametime サーバーをインストールし、それが WebSphere Commerce で作動するように構成する必要があります。詳しくは、*WebSphere Commerce 追加ソフトウェア・ガイド* を参照してください。また、管理コンソールで CSR を登録し、 CSR を使用可能にして、カスタマー・ケアを使用できるようにする必要があります。カスタマー・ケアの総合的な概念および CSR がカスタマー・ケアを使用する方法、さらにこの作業の詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

注:

次のサンプル・ストアのいずれかに基づいてストアを作成する場合、ストア・サービスを使用して迅速かつ容易にストア内でカスタマー・ケアを使用可能にすることができます。Business ToolTech および NewFashion。ストア・サービスを使用してストアを発行した後、「ストア」ビューを選択してから、ストアを選択します。さらに**「構成」**を選択して、カスタマー・ケア機能を使用可能にします。詳しい説明については、 WebSphere Commerce オンライン・ヘルプを参照してください。

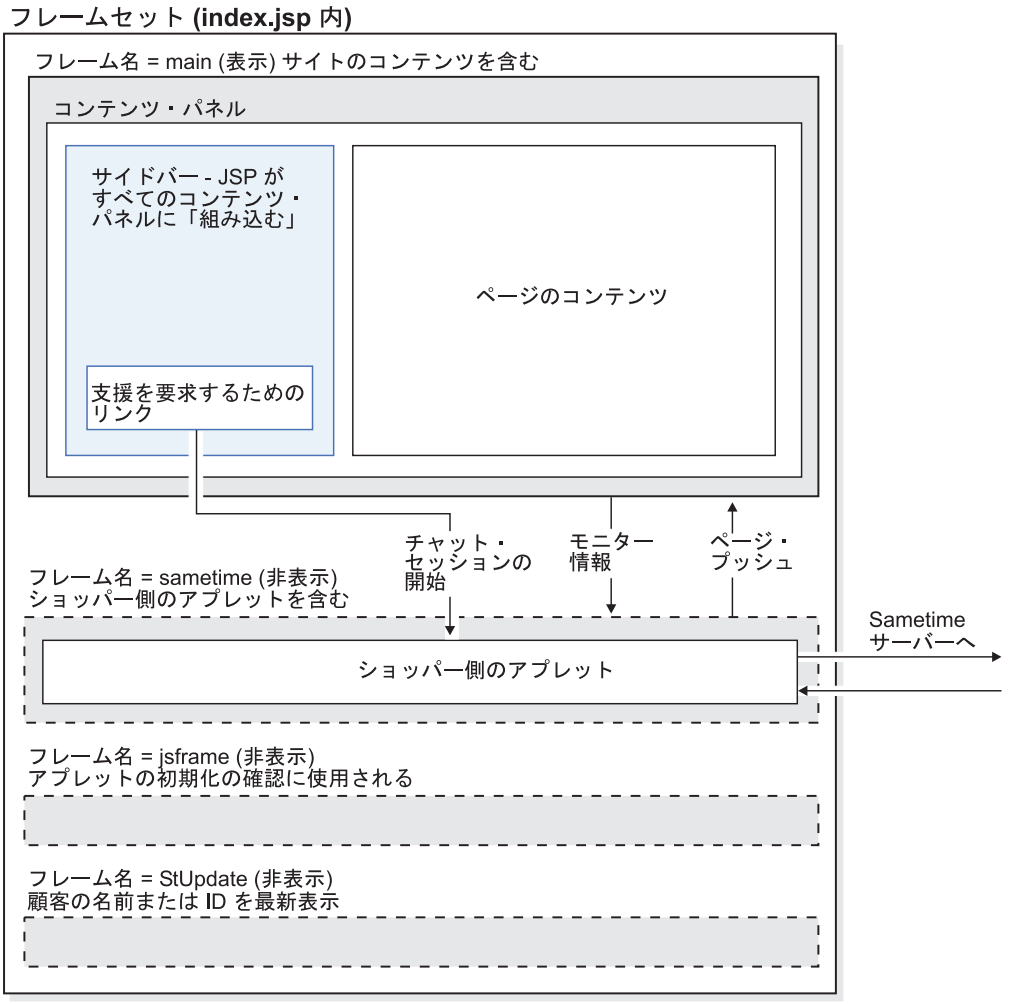
しかし、サンプル・ストアに基づいてストアを作成しない場合は、ストア内でカスタマー・ケアを使用可能にするために、いくらかの作業を行う必要があります。この章の残りの部分では、サンプル・ストアのいずれも使用せずにストア内でカスタマー・ケアを使用可能にするための必要な概念およびステップについて説明します。

注: サンプル・ストア Business ToolTech および NewFashion は、カスタマー・ケアをインプリメントする方法を説明し、カスタマー・ケアを使用可能にするためにストア内で使用できるコードを示します。この章では、2 つのストアの例を参照して、ストア内でカスタマー・ケアを使用可能にする方法を説明します。この章をお読みになる際には、必ずサンプル・ストアの最新版 (WebSphere Commerce 製品の Web サイトから入手可能) をご使用ください。

ストア内のカスタマー・ケアについて

カスタマー・ケアが使用可能になっているストア内で顧客がカスタマー・ケア・リンク (たとえば、**「Live Chat with Customer Assistant (顧客アシスタントとのライブ・チャット)」**) を選択する場合、チャット・ウィンドウを含むアプレットが立ち上がります。このアプレットは隠れたフレーム・セット内で実行されるので、サイトの外観に影響を与えることはありません。アプレットが立ち上がると、Lotus Sametime サーバーに接続します。

以下の図は、フレーム・セットの構成を示しています。



フレーム・セットには以下の 4 つのフレームがあります。

- メイン: ストアのコンテンツが入っているフレーム (ストア・ページを作成するファイルを含む)。コンテンツとしては、ページの本体、ヘッダーおよびフッター・ファイル、サイドバー・ファイルを作成するファイルがあります。このフレームのコンテンツはストアのビジターが見ることができます。メインフレームには Sametime フレームへの接続 (カスタマー・ケアおよびモニターの情報) が含まれることに注意してください。モニター情報については、 298 ページの『カスタマー・ケアを使用した顧客のモニター』で詳しく説明します。
- Sametime: カスタマー・ケア・アプレットが入っているフレーム。このフレームはストアのビジターには見えません。しかし、顧客がリンクをクリックしてアプ

レットを立ち上げると、カスタマー・ケア・ウィンドウが表示されます。また、このフレームは、ページ・プッシュ機能を使用して、情報をメインフレームにプッシュします。

- jsframe: アプレットが正しくロードされていることを確認するフレーム。このフレームのコンテンツは顧客には表示されません。
- StUpdate: 顧客の名前または ID を最新表示するフレーム。

フレーム・セットの使用

フレーム・セット内でカスタマー・ケア・アプレットを立ち上げると、アプレット・コードがストア・ページ内のコードから分離します。上の図が示すように、ストア・ページはフレーム・セットのメインフレームに含まれており、アプレット・コードは Sametime フレームに含まれています。アプレット・コードをストア・ページから分離することにより、フレーム・セットが最初に立ち上げられる際にアプレットが一度だけダウンロードされるため、ネットワーク・トラフィックが減少します。カスタマー・ケア・アプレットがフレーム・セットの一部になっていない場合、このアプレットはすべてのストア・ページに存在している必要があり、新規のストア・ページにアクセスするたびにダウンロードされることになります。

フレーム・セットを使用すると、Sametime サーバーとの接続を保守することもできます。アプレットがフレーム・セットではなく各ページの一部になっている場合、顧客が新規のページにアクセスするたびに新規の Sametime セッションが作成されます。カスタマー・ケア・アプレットは Sametime サーバーに匿名でログオンするため、顧客が新規のページにアクセスするたびに新規セッションを作成すると、ストアを用いた顧客の活動をトレースできません。フレーム・セットを使用すると、顧客の元の Sametime セッションが保守され、属性が変更されると顧客の活動が Sametime サーバーに返送されます。

フレーム・セットの使用について

カスタマー・ケアをストア内にインプリメントするにはフレーム・セットを使用することが勧められていますが、フレーム・セットの使用については以下の点を理解しておく必要があります。

- 単一のエントリー・ポイント: 顧客がフレームワーク内のストアをブラウズする場合、カスタマー・ケアの使用だけを行えます。同様に、CSR はフレーム・セットを介して顧客の動向のモニターだけを行えます。顧客がフレーム・セットを介してストアをブラウズしていることを確認するには、単一のエントリー・ポイント、たとえばストアのホーム・ページ (サンプル・ストアの場合は、 `index.jsp`) を介してサイトにアクセスする必要があります。顧客が別のページ (たとえば、カタログ・ページ) を介してストアにアクセスする場合、その顧客はフレーム・セット内にはいません。
- ブックマーク: フレーム・セットを使用する場合、顧客は個々のページではなく、サイトのメイン URL にのみブックマークを付けることができます。
- 最新表示: 顧客がフレーム・セット内におり、「最新表示」をクリックすると、フレーム・セットでコード化されているとおり (たとえば、`index.jsp`)、メインフレーム・アドレスに戻ります。
- ブラウザー・ウィンドウのサイズ変更: 顧客がフレーム・セット内にいるときにブラウザー・ウィンドウのサイズを変更すると、ブラウザーは自動的にエントリ

ー・アドレスを再ロードします。エントリー・アドレスが再ロードされる場合、Sametime サーバーへの接続が終了することがあります。この状況では、ブラウザーによって動作が異なります。

- セキュリティー: 顧客がフレーム・セットを介してサイトをブラウズする場合、フレーム・セット (ロケーション・バーの URL) と同様に個々のフレームは独自の接続を保守します。この接続は、非セキュア (http。デフォルトではポート 80) の場合と、セキュア (https。デフォルトではポート 443) の場合があります。顧客が非セキュア接続を介してストアをブラウズしている場合、フレーム・セット内のすべてのフレームは HTTP にあります。このシナリオでは、SSL については扱っていません。しかし、顧客がセキュア・ページ (たとえば、登録ページ) をブラウズする場合、フレーム・セット内の main フレームは HTTPS に切り替わりますが、それ以外のフレームは非セキュア (http) のままです。この状況では、顧客はカスタマー・ケア・アプレットを立ち上げることができません。ブラウザーにはアプレットを立ち上げる許可が与えられていません。なぜなら、アプレット (セキュア、ポート 443) はブラウザーのロケーション・バーの URL (HTTP、ポート 80) とは異なるサーバーから立ち上げられているように見えるからです。フレーム・セット全体が再びセキュアになるまで、つまりロケーション・バーの URL が HTTPS を指すまで、アプレットを立ち上げることはできません。フレーム・セット全体を再びセキュアにするには、以下のオプションがあります。
  - フレーム・セット全体をセキュア接続にリダイレクトする。しかし、これを行う場合、アプレットが新規の Sametime 接続に切り替わると、現在行われているチャットが終了します。
  - サイトに入るときに、フレーム・セット全体をセキュア接続にリダイレクトする。これにより、パフォーマンスを多少犠牲にすることになりますが、顧客のセッションに対するセキュリティーおよびブラウズのプライバシーは向上します。

  フレーム・セットをセキュア接続にリダイレクトする 1 つの方法は、以下のコードを index.html ファイルに追加することです。

  ```
  <html>
  <head>
  <META HTTP-EQUIV=Refresh CONTENT="0;URL=https://hostname/webapp/wcs/stores/
  servlet/NewFashion/index.jsp>
  </head>
  <html>
  </head>
  ```

カスタマー・ケアを使用した顧客のモニター

カスタマー・ケアを使用すると、以下のようにして、CSR と対応する顧客をモニターできます。

- 顧客の名前または ID の取得
- 顧客がブラウズするページの判別
- ショッピング・カート内のアイテム数の追跡

この情報を取得するために、カスタマイズしたコードがストア・ページに追加されます。以下のセクションでは、それぞれのモニター機能がサンプル・ストアにインプリメントされる方法について説明します。

顧客の名前または ID の取得

カスタマー・ケア・アプレットが立ち上げられ、CSR がログオンされると、 CSR はアプレットを使用する人物を名前またはショッパー ID によって識別することができます。サンプル・ストアには、カスタマー・ケア・アプレットで作動する特殊化されたコードが含まれており、これにより顧客の名前またはショッパー ID を判別します。このコードは、顧客がゲスト顧客、ショッピング・カートにアイテムを入れたゲスト顧客、または登録済み顧客のいずれかを判別し、名前または ID をその顧客に割り当てて、その名前をカスタマー・ケア・アプレットに戻します。そして、これらの名前が CSR に表示されます。たとえば、顧客がゲスト顧客であり、ショッピング・カートに何も入っていない場合、その顧客にはショッパー ID -1002 の生成済み ID が割り当てられます。顧客がゲスト顧客であり、ショッピング・カートにアイテムが入っている場合には、ショッパー ID が表示され、顧客が登録済み顧客の場合には姓名が表示されます。

サンプル・ストアは、StUpdate フレームを最新表示する以下のコードをストアのヘッダー・ファイルに追加することにより、顧客の名前または ID を取得します。このコードは `header.jsp` (NewFashion の場合) および `NavHeader.jsp` (Business ToolTech の場合) に含まれています。

注: 顧客がストア内の新規ページをブラウズするたびに、その顧客の名前または ID が最新表示されます。

```
<script language="javascript">
    if (typeof top.updateStInfo == 'function')
       top.updateStInfo();
</script>
```

前述のコードでは、StUpdate フレームで以下を最新表示します。

```
//set Customer Name for LiveHelp if user is registered.

if (userRegistrationDataBean.findUser()) {
if (userRegistrationDataBean.getLastName() != null & &
userRegistrationDataBean.getLastName().length() > 0) {
     if(cmdcontext != null) {
     Long uid = cmdcontext.getUserId();
     String customerName ="";
     if (locale.toString().equals("ja_JP")||locale.toString().equals("ko_KR")
||locale.toString().equals("zh_CN")||locale.toString().equals("zh_TW"))
                {
                customerName = "" + userRegistrationDataBean.getLastName() + " "
 + userRegistrationDataBean.getFirstName();
                }
                else {
          customerName = "" + userRegistrationDataBean.getFirstName() + " "
+ userRegistrationDataBean.getLastName();
          }
          }
    else {
        customerName=userRegistrationDataBean.getUserId();
         if (customerName.equals("-1002"))
          customerName="";
                customer_name=customer_name.trim();
      }
```

サンプル・ストアの「ログアウト」ページには、さらに多くのカスタム・コードが含まれています。このコードは顧客名を生成済み ID に設定し、ショッピング・カ

ート内のアイテム数をゼロにリセットします。 NewFashion の「ログアウト」ページは LoginForm.jsp です。 Business ToolTech では、Logoff.jsp です。カスタム・コードは以下のとおりです。

```
<HTML>
<HEAD>
<SCRIPT language="javascript">
 if (typeof parent.setCustomerName == 'function')
     parent.setCustomerName (parent.WCSGUESTID, '')
   if (typeof parent.setShoppingCartItems == 'function')
     parent.setShoppingCartItems(0);
</SCRIPT>
</HEAD>
</HTML>
```

顧客がブラウズするページの判別

カスタマー・ケアを使用すると、顧客が現在ストア内のどのページをブラウズしているかを CSR が判別することができます。サンプル・ストアでは、以下のコードをヘッダー・ファイル (NewFashion の場合は header.jsp および Business ToolTech の場合は NavHeader.jsp) に追加することにより、顧客が見ているページを判別します。

```
<%
//Determine Page Type for LiveHelp
String headerType = (String) request.getAttribute("liveHelpPageType");
if (headerType==null)
headerType = "";

%>

<script language="javascript">
<%
    String pname = request.getRequestURI();
    int indpn = pname.lastIndexOf('/');
    indpn = pname.lastIndexOf('/', indpn-1);
    if(indpn != -1)
        pname = pname.substring(indpn+1);

    //Determine if this is a personal page or not
    if (headerType.equals("personal") ) {
%>
if (typeof parent.setPageParams == 'function')
   parent.setPageParams('PERSONAL_URL', '<%=pname%>');
<% } else { %>
if (typeof parent.setPageParams == 'function')
parent.setPageParams(location.href, '<%=pname%>');
<% } %>
</script>
```

ページでヘッダー・ファイルを使用しない場合、そのページにはファイル StHeader1.jsp が含まれています。 StHeader1.jsp には、ストアのヘッダー・ファイルに追加されるものと同じコードが含まれています。

顧客のプライバシーを保守するために、CSR は特定のページにはアクセスできません。たとえば、CSR はキャンペーン・ページ、契約で決定した価格が含まれるページ、またはユーザー ID が含まれるページ (住所録ページなど) にはアクセスできません。これらのページには個人用というマークが付けられます。サンプル・ストアでは、以下のページに個人用というマークが付けられています。

- NewFashion

- AddressBookForm.jsp
- AllocationCheck.jsp
- edit_registration.jsp
- emptyshoppingcart.jsp
- interestItemDisplay.jsp
- myaccount.jsp
- orderItemDisplay.jsp
- OrderDisplayPending.jsp
- ResultList.jsp
- shoppingcart.jsp
- TrackOrderStatus.jsp

- Business ToolTech
  - Address.jsp
  - Addressbook.jsp
  - AddToExistReqList.jsp
  - AdvancedSearch.jsp
  - AllocationCheck.jsp
  - CatalogMainDisplay.jsp
  - CatalogItemDisplay.jsp
  - CatalogTopCategoriesDisplay.jsp
  - Confirmation.jsp
  - OrderDisplayPending.jsp
  - OrderItemDisplay.jsp
  - OrderDetail.jsp
  - QuickOrder.jsp
  - RequisitionListCreate.jsp
  - RequisitionListDetailDisplay.jsp
  - RequsitionListDisplay.jsp
  - RequsitionListUpdate.jsp
  - Result List.jsp
  - Shipping.jsp
  - shoppingcart.jsp
  - TrackOrderStatus.jsp
  - UserAccount.jsp
  - UserRegistrationUpdate.jsp

ページに個人用のマークを付ける (つまり、CSR が利用できないようにする) には、ヘッダーが組み込まれる直前に、サンプル・ストアは以下のコードをページに組み込みます。

```
<%
// Set header type needed for this JSP for Customer Care.  This must
// be set before Header.jsp
request.setAttribute("liveHelpPageType", "personal");
%>

<%
String incfile;
incfile = includeDir + "Header.jsp";
%>
<jsp:include="<%=incfile%>"flush="true"/>
```

注: CSR は個人用というマークが付けられたページのコンテンツを見ることはできませんが、そのページの URL は見ることができます。

ショッピング・カート内のアイテム数の追跡

カスタマー・ケアを使用すると、顧客がある時点でショッピング・カートに入れているアイテムの数を追跡することがでます。サンプル・ストアでは、以下の場所でショッピング・カート内のアイテム数を取得します。

- 「ショッピング・カート」ページ (NewFashion: `shoppingcart.jsp`, Business ToolTech:`ShoppingCart.jsp`)
- 「Empty shopping cart (空のショッピング・カート)」ページ (NewFashion: `emptyshopcart.jsp`, Business ToolTech:`EmptyOrder.jsp`)
- 「オーダーの確認」ページ (NewFashion: `confirmation.jsp`, Business ToolTech:`confirmation.jsp`)
- 「ログアウト」ページ (Business ToolTech:`Logoff.jsp`)

注: ショッピング・カートは個人用ページとして設計されていますが、 CSR はショッピング・カート内のアイテム数を追跡することができます。この方法を使用して CSR が見ることができるのは、ショッピング・カート内のアイテム数だけで、ショッピング・カートに何が入っているかはわかりません。しかし、CSR は**「View Shopping Cart (ショッピング・カートの表示)」**ボタンを使用して、ショッピング・カートの中身を見ることができます。詳しくは、WebSphere Commerce オンライン・ヘルプを参照してください。

サンプル・ストアでは、以下のコードを上記のページに追加することにより、ショッピング・カート内のアイテム数を判別します。

- 最初に int 変数が定義されます。

  ```
  int shoppingCartItems = 0;
  ```

- 次に、カートに追加するオーダー・アイテムがあるときには、次のコード行を使用して数量を shoppingCartItemes に追加します。

  ```
  shoppingCartItems+= orderItem.getQuantityInEJBType().intValue();
  ```

- さらに、ページの最後で以下のコードを追加して、顧客名をゲスト・ショッパー ID に設定し、顧客のショッピング・カート内のアイテム数を取得します。

  ```
  <script language="javascript">
  if (typeof parent.setShoppingCartItems == 'function')
  parent.setShoppingCartItems(<%=liveHelpShoppingCartItems%>);
  </script>
  ```

「Empty shopping cart (空のショッピング・カート)」ページと「オーダーの確認」ページで次のコードを使用して、カート内のアイテム数をゼロにリセットします。

```
<script language="javascript">
if (typeof parent.setShoppingCartItems == 'function')
parent.setShoppingCartItems(0);
</script
```

ストアへのカスタマー・ケアの追加

サンプルを使用しないストアにカスタマー・ケアを追加するには、以下のようにします。

パート 1: 前提条件をインストールする

カスタマー・ケアがストア内で作動するようにするには、以下を行う必要があります。

- Sametime サーバーをインストールする。詳しくは、*WebSphere Commerce 追加ソフトウェア・ガイド* を参照してください。
- WebSphere Commerce Sametime 統合パッケージをインストールする。詳しくは、*WebSphere Commerce 追加ソフトウェア・ガイド* を参照してください。
- WebSphere Commerce インスタンスを停止し、構成マネージャーで Sametime を使用可能にしてから、インスタンスを再始動する。詳しくは、*WebSphere Commerce 追加ソフトウェア・ガイド* を参照してください。
- CSR を作成し、管理コンソールを使用して CSR をカスタマー・ケアに登録する。詳しくは、WebSphere Commerce オンライン・ヘルプを参照してください。

パート 2: カスタマー・ケア統合ファイルをサンプル・ストアからコピーする

サンプル・ストア NewFashion と ToolTech には、カスタマー・ケアをストアに統合するために使用する以下のファイルが含まれています。

- `Sametime.js`: すべてのフレーム用に組み込まれている JavaScript 関数が入っています。このファイルの関数は、メインフレーム内のページにある親接頭部を付けて呼び出されます (たとえば、parent.setCustomerName)。
- `StBlank.jsp`: 空の JSP ファイル。
- `StFrame.jsp`: JavaScript 関数が入っており、ストアフロント用のカスタマー・ケア・アプレットを組み込みます。
- `StReadyJS.jsp`: アプレットが正しくロードされることを示します。
- `StHeader1.jsp`: パラメーターをアプレットに渡すヘッダー・ファイルで、このヘッダーを含むページが個人用ページかどうかを示します。
- `StUpdate.jsp`: 顧客の名前情報および ID を更新します。

Sametime 統合ファイルをサンプル・ストアからストアにコピーするには、以下のようにします。

1. NewFashion ストアまたは ToolTech ストア用のストア・アーカイブ・ファイルを見付けます。ストア・アーカイブ・ファイルは以下のディレクトリーにあります。

- NT drive:¥WebSphere¥CommerceServer¥samplestores
- 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥samplestores
- AIX /usr/WebSphere/CommerceServer/samplestores
- Solaris /opt/WebSphere/CommerceServer/samplestores
- Linux /opt/WebSphere/CommerceServer/samplestores
- 400 /qibm/proddata/WebCommerce/samplestores

2. ToolTech または NewFashion フォルダーをオープンしてから、ToolTech または NewFashion ストア・アーカイブを選択します。
3. WinZip または同様のツールを使用して、ストア・アーカイブ・ファイルをオープンします。
4. webapp.zip ファイルを見付けます。 WinZip または同様のツールを使用して、そのファイルをオープンします。
5. 以下のファイルを選択します。
   - Sametime.js
   - StBlank.jsp
   - StFrame.jsp
   - StReadyJS.jsp
   - StHeader1.jsp
   - StUpdate.jsp
6. ストアの Web 資産が入っているディレクトリーにファイルを抽出します。

   注: StHeader1.jsp は、サンプル・ストア・アーカイブの include ディレクトリーにあります。ストア・ページに組み込むファイル用の個別のディレクトリーがある場合、 StHeader1.jsp をそのディレクトリーに保管します。そうでない場合、他のストア・ファイルと同じディレクトリーに保管します。

パート 3: ストアへのフレーム・セットの追加

297 ページの『フレーム・セットの使用』の説明にあるとおり、カスタマー・ケア・アプレットはフレーム・セット内で実行します。このフレーム・セットはストア・ホーム・ページ、つまり、顧客がストアに入るために最も使用すると思われるページに追加する必要があります。フレーム・セットをストアに追加するには、以下のようします。

1. どのページがストアのエントリー・ポイントであるかを判別します。
2. ToolTech または NewFashion 用のストア・アーカイブ・ファイルをオープンします。ストア・アーカイブ・ファイルは以下のディレクトリーにあります。
   - NT drive:¥WebSphere¥CommerceServer¥samplestores
   - 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥samplestores
   - AIX /usr/WebSphere/CommerceServer/samplestores
   - Solaris /opt/WebSphere/CommerceServer/samplestores
   - Linux /opt/WebSphere/CommerceServer/samplestores

- 400 /qibm/proddata/WebCommerce/samplestores

3. WinZip または同様のツールを使用して、ストア・アーカイブ・ファイルをオープンします。
4. webapp.zip ファイルを見付けます。 WinZip または同様のツールを使用して、そのファイルをオープンします。
5. index.jsp ファイルをオープンします。
6. 以下のコードをコピーします。

```
<script src="<%="Sametime.js"%></script>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"DTD/xhtml1-transitional.dtd">
<html>
<head>
<title></title>
<script language="javascript">
var MainPageURL="";
if (MainPageURL=="")
MainPageURL="/webapp/wcs/stores/servlet/Logoff?storeId=<%=storeId%>
&langId=<%=langId%>&URL=LogonForm?storeId=<%=storeId%>&catalogId=<%=catalogId%>";
function loadFrame()
{
 main.document.location.href=MainPageURL;
 }
</script>
</head>
<%
String sBasePath="/webapp/wcs/stores/servlet/";
String sSametimeUrl="StFrame.jsp?storeId="+storeId;
String sBlankUrl="StBlank.jsp?storeId="+storeId;
String sUpdateUrl="StUpdate.jsp?storeId="+storeId;
%>
<FRAMESET border=0 frameBorder=0 ROWS="100%,1,1,1,1,1" onLoad="loadFrame();">
<FRAME NAME="main"
 SRC="javascript:top.loadFrame();" MARGINWIDTH=0 SCROLLING="Auto"
FRAMEBORDER="no" noresize>
<FRAME NAME="JSFrame"
 SRC="<%=sBlankUrl% >" MARGINWIDTH=0 SCROLLING="no"
FRAMEBORDER="no" noresize>
<FRAME NAME="sametime" SRC="<%=sSametimeUrl%>" MARGINWIDTH=0 SCROLLING="no"
FRAMEBORDER="no" noresize>
<FRAME NAME="StUpdate" SRC="<%=sUpdateUrl%>" MARGINWIDTH=0 SCROLLING="no"
FRAMEBORDER="no" noresize>
</FRAMESET>
</html>
```

注: 上記の例で、ToolTech サンプル・ストアの場合、メインフレームのソース (SRC) は別のページに対するコマンドです。ストアのセットアップ方法に応じて、 SRC は、コマンド、JSP ファイル、HTML ファイルのいずれかになります。

7. ステップ 6 でコピーしたコードを、ストアのエントリー・ポイントとして設計したページに貼り付けます。メインフレームのソース (SRC) に対して必要な変更を加えます。
8. ファイルを保管します。

パート 4: 顧客の名前または ID を取得するためのコードを追加する

カスタマー・ケアを使用する顧客のショッパー ID の名前を CSR に表示するには、以下のようにします。

1. 299 ページの『顧客の名前または ID の取得』の情報を確認します。
2. 顧客の名前またはショッパー ID の取得元を決定します。以下は、その例です。
   - ストアのエントリー・ポイント
   - 新規登録または更新登録
   - ログアウト
   - ヘッダー
3. 顧客の名前またはショッパー ID の取得元を決定した後、ストアのメイン・ヘッダー・ファイルをページに含めるかどうかを決定します。含めない場合、必要なコードをページ自体に追加するか、またはヘッダー・ファイルに含めるかを決定します。
4. NewFashion ストア・アーカイブ、または ToolTech ストア・アーカイブの `webapp.zip` ファイルから、以下のファイルのいずれかをオープンします。
   - NewFashion: `header.jsp`
   - ToolTech: `NavHeader.jsp`

   注: どちらのファイルも include ディレクトリーの `webapp.zip` ファイルにあります。
5. 以下のコードをコピーします。

```
<script language="javascript">
if (typeof top.updateStInfo == 'function')
 top.updateStInfo()
</script>
```

6. ステップ 5 でコピーしたコードを、ストアのヘッダー・ファイルに貼り付けるか、または該当するページに直接貼り付けます。

   注: 希望に応じて、上記のコードをストア・ヘッダー・ファイルにコピーするのではなく、ファイル `StHeader1.jsp` を該当するページに含めたり、またはコードをファイルに直接含めることができます。
7. ファイルを保管します。
8. (オプション) 「ログアウト」ページで顧客 ID をリセットし (たとえば、顧客の ID から -1002 にリセット)、ショッピング・カート内のアイテム数をゼロにリセットするには、以下のコードをログアウト・ファイルに追加します。

```
<HTML>
<HEAD>
<SCRIPT language="javascript">
 if (typeof parent.setCustomerName == 'function')
     parent.setCustomerName (parent.WCSGUESTID, '')
   if (typeof parent.setShoppingCartItems == 'function')
     parent.setShoppingCartItems(0);
</SCRIPT>
</HEAD>
</HTML>
```

パート 5: 顧客がブラウズするページを判別するためのコードを追加する

顧客がブラウズしているページを判別するには、以下のようにします。

1. StHeader1.jsp ファイルをストアのヘッダー・ファイルに組み込みます。たとえば、次のようにします。

   ```
   <%@ include file="StHeader1.jsp" %>
   ```

2. 以下のコードを、個人用のマークが付けられた (つまり、CSR がアクセスできない) ページに追加します。

   ```
   <%
   // Set header type needed for this JSP for Customer Care.  This must
   // be set before Header.jsp
   request.setAttribute("liveHelpPageType", "personal");
   %>

   <%
   String incfile;
   incfile = includeDir + "Header.jsp";
   %>
   <jsp:include="<%=incfile%>"flush="true"/>
   ```

3. 以下のコードを、ヘッダーを使用しないが個人用のマークが付けられているページに追加します。

   ```
   <%
   // Set header type needed for this JSP for Customer Care.  This must
   // be set before StHeader1.jsp
   request.setAttribute("liveHelpPageType", "personal");

   String incfile;
   incfile = includeDir + "StHeader1.jsp";
   %>
   <jsp:include page="<%=incfile%>" flush="true"/>
   ```

パート 6: ショッピング・カート内のアイテム数を追跡するためのコードを追加する

顧客がショッピング・カートに入れているアイテムを CSR が追跡できるようにするには、以下のようにします。

1. 302 ページの『ショッピング・カート内のアイテム数の追跡』の情報を確認します。
2. ショッピング・カート内のアイテムを追跡する場所を決定します。以下は、その例です。
   - 「ショッピング・カート」ページ
   - 「Empty shopping cart (空のショッピング・カート)」ページ
   - 「オーダーの確認」ページ
   - 「ログアウト」ページ
3. ショッピング・カート内のアイテムを追跡するページで、以下のようにします。
   a. int 変数を定義します。

      ```
      int liveHelpShoppingCartItems= 0;
      ```

   b. オーダー・アイテムがカートに追加されるときに shoppingCartItems に数量を追加するには、以下のコード行を追加します。

      ```
      liveHelpShoppingCartItems+= orderItem.getQuantityInEJBType().intValue();
      ```

c. ページの最後で以下のコードを追加して、ゲスト・ショッパー ID (-1002) を実際のショッパー ID に設定し、ショッピング・カート内のアイテム数を取得します。

```
<script language="javascript">
if (typeof parent.setCustomerName == 'function')
parent.setCustomerName(<%=cmdcontext.getUserId()%>, parent.CustomerName);
if (typeof parent.setShoppingCartItems == 'function')
parent.setShoppingCartItems(<%=liveHelpShoppingCartItems%>);
</script>
```

d. 「Empty shopping cart (空のショッピング・カート)」ページおよび「オーダーの確認」ページでデータを追跡する場合は、以下のコードをそれらのページに追加して、ショッピング・カートの値をゼロにリセットします。

```
<script language="javascript">
if (typeof parent.setShoppingCartItems == 'function')
parent.setShoppingCartItems(0);
</script
```

パート 7: カスタマー・ケアにリンクを追加する

顧客がストア内のカスタマー・ケアにアクセスできるようにするには、以下のようにします。

1. カスタマー・ケアへのリンクを置きたい場所を決定します。たとえば、リンクをナビゲーション・バーに置いて、顧客がいつでも利用できるようにしたり、ストア内の特定のページに置くことができます。
2. 以下のコードを、リンクを置くページにコピーします。

```
<a href="javascript:if((parent.sametime != null)) top.interact();">
<%=infashiontext.getString("LiveHelp")%></a>
```

パート 8: 顧客に表示するメッセージを変更する

CSR に初めて接続したときに顧客に表示されるメッセージ (たとえば、 ″Hello, how can I help you?″ や ″Our office hours are from 9 a.m. to 9 p.m. ″ は、Sametime サーバーのプロパティー・ファイルに保管されます。プロパティー・ファイルは 2 つのファイル・タイプ、Customer.properties と Agent.properties に分けられます。 Customer.properties ファイルには、顧客に表示されるメッセージが含まれており、 Agent.properties ファイルには、CSR に表示される情報が含まれています。どちらのファイルにも、WebSphere Commerce のインスタンスにインストールされたロケールごとに、対応するロケール固有のファイル (たとえば、Customer_de_DE.properties と Agent_de_DE.properties) があります。

これらのファイル内のメッセージを表示するには、以下のようにします。

1. Sametime サーバー上でプロパティー・ファイルを見付けます。デフォルトでは、プロパティー・ファイルは以下のディレクトリーにあります。
   - NT `drive:¥Sametime¥Data¥domino¥html¥wc¥properties`
2. 必要な変更を加えます。
3. ファイルをクローズして保管します。

第 9 部 付録

付録 A. UML の凡例

Unified Modeling Language (UML) は、ソフトウェア設計でそれぞれのエレメントを表示するための標準図形言語です。以下は最も一般的な UML エレメントの例です。仕様に関する詳細は、 http://www.rational.com および http://www.omg.org を参照してください。

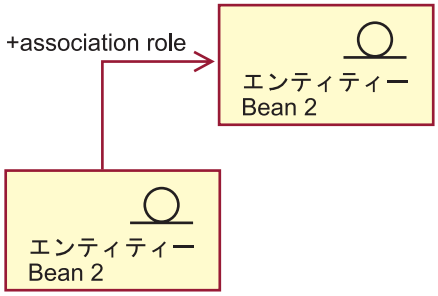
UML のダイアグラムは以下のアイテムで構成されています。

- ボックス: ボックスはオブジェクトのクラスを示します。クラス名がそのボックスの先頭に示されます。属性 (表示される場合) はクラス名の下に示されます。クラス名と属性は間の 1 行で区切られています。
- 線: 線は 2 つのクラスのオブジェクト間で可能な関連を示します。その線の終わりの 1 つにあるクラスのオブジェクトは、他のクラスのオブジェクトと「関連付ける」ことができます。
- 塗りつぶされたひし形: 線の終わりにある塗りつぶされたひし形は、値による埋め込みを表します。その線の反対側の終わりにあるクラスのオブジェクトは、そのひし形が接点をもつクラスの唯一のオブジェクトの部分です。
- 空のひし形: 線の終わりにある空のひし形は、参照による埋め込みを表します。その線のひし形の終わりにあるオブジェクトは、その線の反対側の終わりにあるクラスのグループ化オブジェクトのものと見なすことができます。
- 基数: これらは、関連線の終わりに示されて、基数の制限を示します。以下のテーブルには、基数の制限が要約されています。

基数	関連タイプ
1	1 のみ
0..1	0 または 1
0..n	0 以上
1..n	1 以上

  基数の制限が表示されない場合で、関連線の終わりに塗りつぶされたひし形が示されないかぎり、基数は 0..n と見なされます。この場合、基数は 1 でなければなりません。
- 正符号: 関連線の終わりに示される正符号は、その線の終わりのクラスのオブジェクトがその関連で 1 つの役割を果たすことを示します。正符号の後のテキストは、関連でのそのオブジェクトの役割を示します。
- 矢印: 関連線の終わりの矢印は、 2 つのオブジェクト間の関係の方向が矢印の方向であることを示します。関連線に矢印がないのは、オブジェクト間の関係の方向が通常双方向であることを示します。

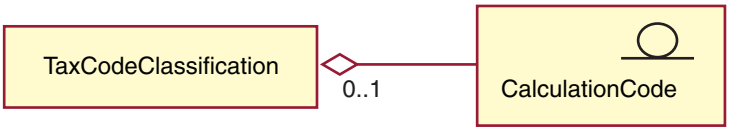
以下のダイアグラムも上記の概念を図示しています。



このダイアグラムは、 Enterprise Java Bean (EJB) を示すデコレーション・ステレオタイプ・シンボルで、 2 つのエンティティーを表示します。最初の bean から 2 番目の bean への単一方向関連です。正符号の後にテキストが続いていて、エンティティー bean 2 がこの関連でどんな役割を果たすかを記述しています。

このダイアグラムでは、StoreEntity は唯一の所有者を持っています。これは 1 つの Member です。 Member は 0 以上の StoreEntities を所有します。正符号は、その Member がその関連で 1 つの役割を果たすことを示します。この場合、この Member は StoreEntity の所有者です。矢印は、StoreEntity にその所有者を尋ねることによって StoreEntity の所有者を通常通り調べることができることを示し、その Member が所有するすべての StoreEntity を尋ねる必要はありません。

このダイアグラムでは、OrderItem は常に、唯一の Order の部分です。 Order には 0 個以上の OrderItem があります。



このダイアグラムは、 CalculationCode が 0 以上の TaxCodeClassifications によってグループ化され、 TaxCodeClassification は 0 以上の CalculationCodes をグループ化します。

付録 B. データの作成

XML ファイルの形式でストア・データを作成する前に、以下に示すことを実行してください。

- 作成する情報の順序を決定します。ストア・データに関する各章に載せられている情報は、データを作成する順序について示唆していますが、 XML ファイルの作成においては、親テーブルの情報が子テーブルの情報より前になければならない点に注意してください。
- ストアの使用方法を決定します。サンプル・ストア・アーカイブ・ファイル (.sar) は、新しいストア作成の基礎としてコピーして使用するためのストア・アーカイブであり、サンプル・ストア・アーカイブ・ファイルを作成する場合に作成するデータは、サンプル・ストアを作成しない場合のデータと少し異なります。詳細については、『サンプル・ストアのためのデータの作成』を参照してください。

サンプル・ストアのためのデータの作成

サンプル・ストア・アーカイブにあるデータは、ローダー・パッケージにとって妥当な、整形式の XML ファイルを使用しています。ストア・アーカイブ XML ファイルは移植可能であることが意図されており、データベースの特定のインスタンスに固有な、生成された 1 次鍵を含めるべきではありません。その代わりに、発行時に ID リゾルバーによって解決される内部用の別名が使用されます。これらの規則を使用すると、サンプル・ストア・アーカイブを何度もコピーして発行することができます。

複数のストアを生成するために使用するサンプル・ストア・アーカイブを作成するのではない場合、または移植可能で、別の WebSphere Commerce インスタンスに発行できるストア・アーカイブを作成するのではない場合、ストアのストア・データを XML ファイル形式で作成する際に、上記の規則を使用する必要はありません。

結果として、サンプル・ストア・アーカイブでは以下の規則が使用されます。

- `member_id="& MEMBER_ID;"` などの中の `&。 & XXX;` 規則は DTD マクロです (XML ではエンティティーといいます)。

  注: MEMBER_ID は、サンプル・ストア・アーカイブの作成時に DTD マクロとして定義する必要があります。

  WebSphere Commerce は以下のファイルに一連のマクロを定義します。

  - NT `drive:¥WebSphere¥CommerceServer¥xml¥sar¥DBLoadMacros.dtd`
  - 2000 `drive:¥Program Files¥WebSphere¥CommerceServer¥xml¥sar¥DBLoadMacros.dtd`
  - AIX `/usr/WebSphere/CommerceServer/xml/sar/DBLoadMacros.dtd`
  - Solaris `/opt/WebSphere/CommerceServer/xml/sar/DBLoadMacros.dtd`

– Linux /opt/WebSphere/CommerceServer/xml/sar/DBLoadMacros.dtd

– 400 /qibm/proddata/WebCommerce/xml/sar/DBLoadMacros.dtd

en_US および es_ES などのマクロは、適切な言語 ID に設定されます。たとえば、

```
<!ENTITY en_US "-1">
```

情報はストア・サービスのツールを使用して指定されます。たとえば、ユーザーは、ストア・サービスの「ストア・アーカイブの作成」ページで MEMBER_ID を選択します。 MEMBER_ID マクロは、ストアを所有するメンバーの ID に対するプレースホルダーです。ストア・アーカイブを作成する際に、ストアの所有者になるメンバーを選択します。 MEMBER_ID マクロはこのメンバーの ID に設定されます。たとえば、メンバー ID -2000 を選択すると、MEMBER_ID は次のように -2000 に設定されます。

```
<!ENTITY MEMBER_ID "-2000">
```

- `ffmcenter_id="@ffmcenter_id_1"` などの中の @。 @ 記号の使用は、内部別名解決法と呼ばれます。ローダー・パッケージ・ユーティリティーの 1 つである ID リゾルバーは、 ID を必要とする XML 要素のために、ID を生成します。 ID リゾルバーで使用されているテクニックの 1 つは、内部別名解決です。内部別名解決法を使用する場合は、 XML 文書内で 1 次鍵 (ID) の代わりに別名が用いられます。これで別名は、そのエレメントを参照するために、XML ファイル内の他の場所で使用できます。したがって、XML ファイルを構築するのに必要な固有索引を知っている必要はありません。ストア・サービスでの発行中、またはローダー・パッケージの使用中に、 ID リゾルバーは @ 記号を固有な値に置き換えます。 XML ファイルの以下の例を参照してください。

  – ID リゾルバー実行前

    ```
    <catalog
    catalog_id="@catalog_id_1"
    member_id="&MEMBER_ID;"
    identifer=InFashion"
    description="InFashion Catalog"/>
    ```

  – ID リゾルバー実行後

    ```
    <catalog
    catalog_id="10001"
    member_id="-2000"
    identifer=InFashion"
    description="InFashion Catalog"/>
    ```

  ここで、10001 は ID リゾルバーによって割り当てられる固有な ID で、 -2000 はストア・サービスにおいてユーザーによって選択されたメンバー ID です。その結果、XML ファイルはローダー・パッケージを使用してロードされます。 ID リゾルバーからファイルを実行すると、単一の XML ファイルのセットから、多くのストアを確実に作成できます。

ストア・サービスとサンプル・ストア

ストア・サービスの**「New (新規)」**および「Create Store Archive (ストア・アーカイブの作成)」のオプションは、前述の規約に従うものです。作成したストアを、ストア・サービスにより他のストアを作成するためのテンプレートとして使用したい場合は、データ資産の作成においてそれらの規約に従う必要があります。

しかし、サンプル・ストア・アーカイブを作成しない場合には、ストア・サービスに含まれるツールを使用することによって、それらの規約に従わないストア・アーカイブを編集したり発行したりできます。

付録 C. sarinfo.xml

ストア・アーカイブには、それぞれ 1 つの sarinfo.xml ファイルが含まれていなければなりません。このファイルはディスクリプターと呼ばれるもので、ストア・アーカイブ発行時に使用されるストア・アーカイブに関する情報が含まれています。その中には、ファイル資産としての ZIP ファイルやストア・データベース XML ファイルの名前や、その発行順序の情報が含まれます。ストア・アーカイブの中に複数の言語のファイルが含まれている場合、 sarinfo.xml ファイルの情報にはその情報も含まれ、それによって、各言語ファイルの発行順序が決定されます。

注: データ資産の中には他のデータ資産より前に発行しなければならないものがあるため、データ資産の発行順は重要です。そのため、sarinfo.xml ファイルで指定される資産の順序は、サンプル・ストアの sarinfo.xml ファイルで指定されている資産の順序と同じでなければなりません。サンプル・ストア・アーカイブは、以下に示すディレクトリーにあります。

- NT drive:¥WebSphere¥CommerceServer¥samplestores
- 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥samplestores
- AIX /usr/WebSphere/CommerceServer/samplestores
- Solaris /opt/WebSphere/CommerceServer/samplestores
- Linux /opt/WebSphere/CommerceServer/samplestores
- 400 /qibm/proddata/WebCommerce/samplestores

ストア・アーカイブの内容を表示するには、まず ZIP プログラムを使用してそれを解凍します。 sarinfo.xml は SAR-INF ディレクトリーにあります。

sarinfo.xml の例

以下に示す例は、ToolTech の sarinfo.xml ファイルです。その要素、属性、および属性値については、この後に示す情報を参照してください。ストア・アーカイブの XML 仕様については、以下に示すディレクトリーにある sarinfo.dtd を参照してください。

- NT drive:¥WebSphere¥CommerceServer¥xml¥sar
- 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥xml¥sar
- AIX /usr/WebSphere/CommerceServer/xml/sar
- Solaris /opt/WebSphere/CommerceServer/xml/sar
- Linux /opt/WebSphere/CommerceServer/xml/sar
- 400 /qibm/proddata/WebCommerce/xml/sar

```
<?xml version = "1.0"?>
<!DOCTYPE sarinfo SYSTEM "sarinfo.dtd">
<sarinfo complete-store="yes" multi-language="yes" version="1.0">
```

```
<store-info asset-name="store"/>

<file name="webapp.zip" type="zip">


<asset fragmented="no" name="webapp">
<file name="webapp.zip" type="zip">
<display-name>My Web App Display Name</display-name>
<description>My Web App</description>
</file>
</asset>

<asset fragmented="no" name="properties">
<file name="properties.zip" type="zip" />
</asset>

<asset fragmented="no" name="dbloadmacros">
<file name="data/DBLoadMacros.dtd" type="dtd"/>
</asset>

<asset fragmented="no" name="fulfillment">
<file name="data/fulfillment.dtd" type="dtd"/>
<file name="data/fulfillment.xml" priority="1" type="db-load"/>
<file name="data/en_US/fulfillment.xml" priority="31" type="db-load">
<locale>en_US</locale>
</file>
<file name="data/es_ES/fulfillment.xml" priority="31" type="db-load">
<locale>es_ES</locale>
</file>
</asset>

<asset fragmented="yes" name="store">
<file name="data/store.dtd" type="dtd"/>
<file name="data/store.xml" priority="2" type="db-load"/>
<file name="data/en_US/store.xml" priority="3" type="db-load">
<locale>en_US</locale>
</file>
<file name="data/es_ES/store.xml" priority="3" type="db-load">
<locale>es_ES</locale>
</file>
</asset>

<asset fragmented="yes" name="catalog">
<file name="data/catalog.dtd" type="dtd"/>
<file name="data/catalog.xml" priority="4" type="db-load"/>
<file name="data/en_US/catalog.xml" priority="5" type="db-load">
<locale>en_US</locale>
</file>
<file name="data/es_ES/catalog.xml" priority="5" type="db-load">
<locale>es_ES</locale>
</asset>

<asset fragmented="yes" name="tax">
<file name="data/tax.dtd" type="dtd"/>
<file name="data/tax.xml" priority="6" type="db-load"/>
<file name="data/en_US/tax.xml" priority="7" type="db-load">
<locale>en_US</locale>
</file>
<file name="data/es_ES/tax.xml" priority="7" type="db-load">
<locale>es_ES</locale>
</file>
</asset>

<asset fragmented="no" name="taxfulfill">
<file name="data/taxfulfill.dtd" type="dtd"/>
<file name="data/taxfulfill.xml" priority="8" type="db-load"/>
</asset>
```

```
<asset fragmented="yes" name="shipping">
<file name="data/shipping.dtd" type="dtd"/>
<file name="data/shipping.xml" priority="9" type="db-load"/>
file name="data/en_US/shipping.xml" priority="10" type="db-load">
<locale>en_US</locale>
/file>
file name="data/es_ES/shipping.xml" priority="10" type="db-load">
<locale>es_ES</locale>
<file>
</asset>

<asset fragmented="no" name="shippingfulfill">
<file name="data/shipfulfill.dtd" type="dtd"/>
<file name="data/shipfulfill.xml" priority="11" type="db-load"/>
</asset>

<asset fragmented="no" name="store-catalog">
<file name="data/store-catalog.dtd" type="dtd"/>
<file name="data/store-catalog.xml" priority="12" type="db-load"/>
</asset>

<asset fragmented="no" name="storefulfill">
<file name="data/storefulfill.dtd" type="dtd"/>
<file name="data/storefulfill.xml" priority="13" type="db-load"/>
</asset>

<asset fragmented="yes" name="offering">
<file name="data/offering.dtd" type="dtd"/>
<file name="data/offering.xml" priority="14" type="db-load"/>
</asset>

<asset fragmented="no" name="command">
<file name="data/command.dtd" type="dtd"/>
<file name="data/command.xml" priority="16" type="db-load"/>
</asset>

<asset fragmented="yes" name="currency">
<file name="data/currency.dtd" type="dtd"/>
<file name="data/currency.xml" priority="17" type="db-load"/>
<file name="data/en_US/currency.xml" priority="18" type="db-load">
<locale>en_US</locale>
</file>
<file name="data/es_ES/currency.xml" priority="18" type="db-load">
<locale>es_ES</locale>
</file>
</asset>

<asset fragmented="yes" name="campaign">
<file name="data/campaign.dtd" type="dtd"/>
<file name="data/campaign.xml" priority="20" type="db-load"/>

<file name="data/en_US/campaign.xml" priority="24" type="db-load">
<locale>en_US</locale>
</file>
<file name="data/es_ES/campaign.xml" priority="24" type="db-load">
<locale>es_ES</locale>
</file>
</asset>

<asset fragmented="no" name="store-catalog-tax">
<file name="data/store-catalog-tax.dtd" type="dtd"/>
<file name="data/store-catalog-tax.xml" priority="21" type="db-load"/>
</asset>

<asset fragmented="no" name="store-catalog-shipping">
<file name="data/store-catalog-shipping.dtd" type="dtd"/>
```

```
<file name="data/store-catalog-shipping.xml" priority="22" type="db-load"/>
</asset>

<asset fragmented="no" name="store-defaults">
<file name="data/store-defaults.dtd" type="dtd"/>
<file name="data/store-defaults.xml" priority="23" type="db-load"/>
</asset>

<asset fragmented="no" name="consistency_check">
<file name="data/sarrule.dtd" type="dtd"/>
<file name="data/sarrule.xml" priority="25" type="config"/>
</asset>

<asset fragmented="no" name="payment">
<file name="data/es_ES/paymentinfo.xml" type="config"/>
<file name="data/paymentinfo.dtd" type="dtd"/>
</asset>

<asset fragmented="yes" name="policy">
<file name="data/businesspolicy.dtd" type="dtd"/>
<file name="data/businesspolicy.xml" priority="26" type="db-load"/>
<file name="data/en_US/businesspolicy.xml" priority="27" type="db-load">
<locale>en_US</locale>
</file>
<file name="data/es_ES/businesspolicy.xml" priority="27" type="db-load">
<locale>es_ES</locale>
</file>
</asset>

<asset fragmented="no" name="organization">
<file name="data/organization.dtd" type="dtd"/>
<file name="data/organization.xml" priority="28" type="db-load"/>
</asset>


asset fragmented="no" name="businessaccount">
<file name="data/businessaccount.xml" type="xml"/>
/asset>
asset fragmented="yes" name="contract">
<file name="data/contract.xml" priority="1" type="xml"/>
<file name="data/en_US/contract.xml" priority="2" type="xml">
<locale>en_US</locale>
</file>
file name="data/es_ES/contract.xml" priority="2" type="xml">
<locale>es_ES</locale>
</file>
</asset>

<asset fragmented="yes" name="accesscontrol">
<file name="data/accesscontrol.dtd" type="dtd"/>
<file name="data/accesscontrol.xml" priority="29" type="db-load"/>
<file name="data/en_US/accesscontrol.xml" priority="30" type="db-load">
<locale>en_US</locale>
</file>
<file name="data/es_ES/accesscontrol.xml" priority="30" type="db-load">
<locale>es_ES</locale>
</file>
</asset>

<!-- next priority should be 32 -->

</sarinfo>
```

この場合、次のようになります。

sarinfo は、 sarinfo.xml ファイルに含まれる情報のすべてです。以下に示す表に、ストア・アーカイブに関する一般的な情報を含む属性を示します。

属性名	属性値
multi-language (必須)	このストア・アーカイブで多言語をサポートするかどうか。 • yes • no
complete-store (必須)	完全なストアのために必要な資産がストア・アーカイブに含まれているかどうか。 • yes • no
version	ストア・アーカイブのバージョン。例: 1.0、1.1
display-name	ストア・アーカイブの名前。
description	このストア・アーカイブに関する簡単な説明。
standard-schema (必須)	このストア・アーカイブが WebSphere Commerce の標準データベース・スキーマに従うものかどうか。 • yes • no
store-info asset-name (必須)	ストア・アーカイブのアンカーとして機能する資産。ストアのすべての情報は、その資産に属するファイルの中に含まれています。例: store
locale	ストア・アーカイブでサポートされるロケール。下記のロケール変数は、言語コードと国コードを下線 (_) で連結したものです。 • de_DE • en_US • es_ES • fr_FR • it_IT • ja_JP • ko_KR • pt_BR • zh_CN • Zh_TW

asset (必須) は、いくつかの関連するファイルの論理的な集まりです。たとえば、税 (tax) は、ストアの税に関するファイル全体からなるグループの名前です。

属性名	属性値
name (必須)	資産タイプの名前。たとえば、 • store • catalog • payment • tax
fragmented (必須)	資産情報を、その言語に従って複数のファイルに分割するかどうか。 • yes • no

`file`

属性名	属性値
name (必須)	ファイルの名前。
type (必須)	ファイルの形式の種類。たとえば、 • db-load - データベースにロードするファイル • dtd - 文書タイプ定義ファイル • zip - ファイル資産のための ZIP ファイル (webapp.zip など) • config - 構成ファイル
priority	ストア・アーカイブに含まれるファイルの発行順。 1,2,3,4 . . . **注:** 2 つのファイルの優先順位が同じなら、ロード順はあまり重要ではありません。ファイルを特定の順序でロードすることが必要な場合は、異なる優先順序を割り当ててください。
display-name	ファイルの名前。
description	参照用の説明。
locale	ロケール。下記のロケール変数は、言語コードと国コードを下線 (_) で連結したものです。 • de_DE • en_US • es_ES • fr_FR • it_IT • ja_JP • ko_KR • pt_BR • zh_CN • Zh_TW

付録 D. sarrule.xml

ストア・アーカイブごとに、1 つの sarrule.xml ファイルが含まれています。これにより、ストア・サービスを使用した発行において一貫性をチェックできます。発行時に発行ユーティリティーは、 sarrule.xml ファイルに指定されているルールを使用することにより、作成したストア・アーカイブの中に、 XML ファイルで指定されている Web 資産が含まれていることをチェックします。

sarrule.xml の例

以下に示す sarrule.xml ファイルは、ToolTech サンプル・ストアのものです。

ストア・アーカイブ・ファイルは、以下に示すディレクトリーにあります。

- NT drive:¥WebSphere¥CommerceServer¥samplestores
- 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥samplestores
- AIX /usr/WebSphere/CommerceServer/samplestores
- Solaris /opt/WebSphere/CommerceServer/samplestores
- Linux /opt/WebSphere/CommerceServer/samplestores
- 400 /qibm/proddata/WebCommerce/samplestores

ストア・アーカイブの sarrule.xml ファイルを表示するには、 ZIP プログラムを使ってこれを解凍します。 sarrule.xml ファイルは、データ・ディレクトリーにあります。

sarrule.dtd ファイルは以下のディレクトリーにあります。

- NT drive:¥WebSphere¥CommerceServer¥xml¥sar
- 2000 drive:¥Program Files¥WebSphere¥CommerceServer¥xml¥sar
- AIX /usr/WebSphere/CommerceServer/xml/sar
- Solaris /opt/WebSphere/CommerceServer/xml/sar
- Linux /opt/WebSphere/CommerceServer/xml/sar
- 400 /qibm/proddata/WebCommerce/xml/sar

注: ストア・アーカイブでは、サンプル・ストアに含まれている sarrule.xml ファイルを使用することをお勧めします。しかし、既存の sarrule.xml ファイルに新しいルールを追加することもできます。

```
<?xml version="1.0"?>
<!DOCTYPE SAR-rules SYSTEM "sarrule.dtd">
<SAR-rules>
<asset name = "command">
<check type="webasset registration">
<rule entry="viewreg" attribute="properties" type="java.lang.String"
        removeStoreDir="false" file="webapp.zip"/>
```

```
</check>
   </asset>
!--
   <asset name = "catalog">
<check type="webasset registration">
     <rule entry = "catalogdsc" attribute="thumbnail" type="java.lang.String"
 removeStoreDir="false" file="webapp.zip"/>
<rule entry = "catalogdsc" attribute="fullimage" type="java.lang.String"
         removeStoreDir="false" file="webapp.zip"/>
<rule entry = "catentdesc" attribute="thumbnail" type="java.lang.String"
         removeStoreDir="false" file="webapp.zip"/>
<rule entry = "catentdesc" attribute="fullimage" type="java.lang.String"
         removeStoreDir="false" file="webapp.zip"/>
<rule entry = "catgrpdesc" attribute="thumbnail" type="java.lang.String"
         removeStoreDir="false" file="webapp.zip"/>
<rule entry = "catgrpdesc" attribute="fullimage" type="java.lang.String"
         removeStoreDir="false" file="webapp.zip"/>
</check>
   </asset>
-->
   <asset name = "store">
<check type="webasset registration">
     <rule entry = "dispentrel" attribute="pagename" type="java.lang.String"
            removeStoreDir="false" file="webapp.zip"/>
     <rule entry = "dispcgprel" attribute="pagename" type="java.lang.String"
            removeStoreDir="false" file="webapp.zip"/>
</check>
   </asset>
</SAR-rules>
```

上記のうち、

```
<asset name = "command">
<check type="webasset registration">
<rule entry="viewreg" attribute="properties" type="java.lang.String"
         removeStoreDir="false" file="webapp.zip"/>
</check>
   </asset>
```

の部分で、ストア・サービスの発行ユーティリティーは、 command.xml ファイルの中に指定されている各 JSP ファイルが、ストア・アーカイブの Web 資産の中に存在していることをチェックします。

上記のうち、

```
   <asset name = "catalog">
<check type="webasset registration">
     <rule entry = "catalogdsc" attribute="thumbnail" type="java.lang.String"
         removeStoreDir="false" file="webapp.zip"/>
<rule entry = "catalogdsc" attribute="fullimage" type="java.lang.String"
         removeStoreDir="false" file="webapp.zip"/>
<rule entry = "catentdesc" attribute="thumbnail" type="java.lang.String"
         removeStoreDir="false" file="webapp.zip"/>
<rule entry = "catentdesc" attribute="fullimage" type="java.lang.String"
         removeStoreDir="false" file="webapp.zip"/>
<rule entry = "catgrpdesc" attribute="thumbnail" type="java.lang.String"
         removeStoreDir="false" file="webapp.zip"/>
<rule entry = "catgrpdesc" attribute="fullimage" type="java.lang.String"
         removeStoreDir="false" file="webapp.zip"/>
</check>
    </asset>
```

の部分で、ストア・サービスの発行ユーティリティーは、 catalog.xml ファイルの中に指定されている各カタログ資産が、ストア・アーカイブの Web 資産の中に存在していることをチェックします。

上記のうち、

```
<asset name = "store">
<check type="webasset registration">
     <rule entry = "dispentrel" attribute="pagename" type="java.lang.String"
            removeStoreDir="false" file="webapp.zip"/>
     <rule entry = "dispcgprel" attribute="pagename" type="java.lang.String"
            removeStoreDir="false" file="webapp.zip"/>
</check>
   </asset>
```

の部分で、ストア・サービスの発行ユーティリティーは、 store.xml ファイルの中に指定されている各ストア資産が、ストア・アーカイブの Web 資産の中に存在していることをチェックします。

発行時にメモリーが問題となるなら、発行の前に sarrule.xml の中のカタログ資産のルールをコメントにしてください。

付録 E. データ・グループ

すべてのストアは、作成とロードのために、グループに分けられます。これらのデータ・グループは論理的に関連のあるテーブルのセットです。オブジェクト間の関係をロードするにはオブジェクトが存在しなければならないため、データ・グループが編成される順序はデータのロードにとって重要です。

ストア用の XML 形式のデータをロードする際、選択したグループのデータだけをロードすることができます。これらのグループは、以前の章で作成したデータ資産 (たとえば、カタログ資産またはフルフィルメント資産) で構成されます。 281 ページの『データ・グループのロード』の説明に従ってデータ・グループをロードする前に、次の事柄を行います。

- ロードするデータ・グループを決定します。各グループには、データをロードする前に満たしていなければならない従属関係があります。
- 『データ・グループの従属関係』の情報を確認します。
- 選択したデータ・グループ用の XML ファイルを作成または更新したことを確認します。ストア・データに関する各章に載せられている情報は、データを作成する順序について示唆していますが、 XML ファイルの更新においては、親テーブルの情報が子テーブルの情報より前になければならない点に注意してください。

データ・グループの従属関係

データ資産の各グループは、情報を外部 WebSphere Commerce ファイルおよびデータベース・テーブルから取り出します。それぞれのデータ・グループはデータのロード元の特定の外部ファイルおよびデータベース・テーブルに従属しているため、ロード処理の前に以下の図表を考慮してください。

以下の点を覚えておいてください。

- **外部従属関係**列にリストされているファイルには、**関連テーブル**への外部鍵参照が含まれています。これらのテーブルに最初にデータを入力しなければならないことに注意してください。
- 名前の先頭が bootstrap のファイル名は、 WebSphere Commerce インスタンスの作成時にデータが入力されたことを示します。
- 各データ・グループ用のテーブルは、ロケール固有のテーブルを示すために分割されています。そうしたテーブルには、適用できるマルチリンガル情報 (商品説明など) がすべて含まれています。
- ブラケット内の XML ファイルにはテーブル・データが含まれています。
- アクセス・コントロールおよびフルフィルメント以外のすべてのデータ・グループには、ストア・データ・グループへの従属関係があります。
- ストア・デフォルト・データ・グループは、配送データ・グループおよび契約データ・グループに従属しています。しかし、ストア・デフォルト・データをロードする前に、配送および契約データが存在していなければならないわけではありません。

- XML ファイル名は、WebSphere Commerce サンプル・ストアが使用するファイルの名前です。データ・グループ表内の情報は、ストアのサイズ、機能、および必要に応じて変わることがあります。特定のテーブルが以下のリストにないとしても、データ・グループ資産を含むすべてのデータベース・テーブルを含めてください。

アクセス・コントロール・データ・グループ		
外部従属関係	**関連テーブル**	**関連テーブル (ロケール固有)**
bootstrap.LANGUAGE.LANGUAGE_ID, bootstrap.MEMBER.MEMBER_ID	**accesscontrol.xml** ACACTACTGP, ACACTGRP, ACACTION, ACPOLICY, ACRESCGRY, ACRESGPRES, ACRESGRP	**accesscontrol.xml** ACACGPDESC, ACACTDESC, ACPOLDESC, ACRSCGDES, ACRESGPDES
Business ビジネス契約データ・グループ		
外部従属関係	**関連テーブル**	
store.STOREENT.STOREENT_ID	契約データベース・テーブルは直接ロードされません。また、その他の WebSphere Commerce データ・グループとは異なるプロセスに従います。詳しくは、331 ページの 『契約データ・グループのロード』を参照してください。	
ビジネス・ポリシー・データ・グループ		
外部従属関係	**関連テーブル**	**関連テーブル (ロケール固有)**
bootstrap.LANGUAGE.LANGUAGE_ID, boostrap.MEMBER.MEMBER_ID, store.STOREENT.STOREENT_ID	**businesspolicy.xml** POLICY, POLICYCMD	**businesspolicy.xml** POLICYDESC
キャンペーン・データ・グループ		
外部従属関係	**関連テーブル**	**関連テーブル (ロケール固有)**
store.STOREENT.STOREENT_ID	**campaign.xml** CAMPAIGN, COLLATERAL, EMSPOT, STENCALUSG	**campaign.xml** COLLDESC
カタログ・データ・グループ		
外部従属関係	**関連資産およびテーブル**	**関連テーブル (ロケール固有)**

bootstrap.LANGUAGE.LANGUAGE_ID, bootstrap.MEMBER.MEMBER_ID, store.STOREENT.STOREENT_ID, shipping.CALCODE.CALCODE_ID, tax.CALCODE.CALCODE_ID	**catalog.xml** ATTRIBUTE, ATTRVALUE, BASEITEM, CATALOG, CATENTREL, CATENTRY, CATGPENREL, CATGROUP, CATGRPREL, CATTOGRP, ITEMSPC, ITEMVERSN, PKGATTR, PKGATTRVAL, RA, RADETAIL, STOREITEM, STORITMFFC, VERSIONSPC **offering.xml** CATGRPTPC, MGPTRDPSCN, OFFER, OFFERPRICE, TRADEPOSCN **storefulfill.xml** INVENTORY **store-catalog.xml** DISPCGPREL, DISPENTREL, STORECAT, STORECENT, STORECGRP **store-catalog-shipping.xml** CATENTCALD, CATENTSHIP **store-catalog-tax.xml** CATENTCALD	**catalog.xml** BASEITMDSC, CATALOGDSC, CATENTDESC, CATGRPDESC
コマンド・データ・グループ		
外部従属関係	**関連テーブル**	**関連テーブル (ロケール固有)**
store.STOREENT.STOREENT_ID	**command.xml** CMDREG, VIEWREG	なし
通貨データ・グループ		
外部従属関係	**関連テーブル**	**関連テーブル (ロケール固有)**

store.STOREENT.STOREENT_ID	**currency.xml** CURCONVERT, CURLIST	なし
フルフィルメント・データ・グループ		
外部従属関係	関連テーブル	関連テーブル (ロケール固有)
bootstrap.LANGUAGE.LANGUAGE_ID, boostrap.MEMBER.MEMBER_ID	**fulfillment.xml** FFMCENTER	なし
組織データ・グループ		
外部従属関係	関連テーブル	関連テーブル (ロケール固有)
bootstrap.LANGUAGE.LANGUAGE_ID, boostrap.MEMBER.MEMBER_ID	**organization.xml** ADDRBOOK, ADDRESS, MBRREL, MEMBER, ORGENTITY	なし
配送データ・グループ		
外部従属関係	関連テーブル	関連テーブル (ロケール固有)
bootstrap.LANGUAGE.LANGUAGE_ID, bootstrap.MEMBER.MEMBER_ID, fulfillment.FFMCENTER.FFMCENTER_ID, store.STOREENT.STOREENT_ID	**shipping.xml** CALCODE, CALRANGE, CALRLOOKUP, CALRULE, CALSCALE, CRULESCALE, JURST, JURSTGPREL, JURSTGROUP, SHIPMODE, STENCALUSG **shipfulfill.xml** SHPJCRULE, SHPARRANGE	**shipping.xml** SHPMODEDSC
ストア・データ・グループ		
外部従属関係	関連テーブル	関連テーブル (ロケール固有)
bootstrap.LANGUAGE.LANGUAGE_ID, bootstrap.MEMBER.MEMBER_ID, bootstrap.SETCURR.SETCURR_ID, fulfillment.FFMCENTER.FFMCENTER_ID	**store.xml** INVADJCODE, INVADJDESC, RTNREASON, STADDRESS, STORE, STOREENT, STORELANG, VENDOR	**store.xml** FFMCENTDS, INVADJDESC, RTNRSNDESC, STOREENTDS, STORLANGDS, VENDORDESC
ストア・デフォルト・データ・グループ **group**		
外部従属関係	関連テーブル	関連テーブル (ロケール固有)

contract.CONTRACT.CONTRACT_ID, store.STOREENT.STOREENT_ID, shipping.SHIPMODE.SHIPMODE_ID	**store-default.xml** STOREDEF	なし
税データ・グループ		
外部従属関係	**関連テーブル**	**関連テーブル (ロケール固有)**
bootstrap.LANGUAGE.LANGUAGE_ID, boostrap.MEMBER.MEMBER_ID, store.STOREENT.STOREENT_ID, fulfillment.FFMCENTER.FFMCENTER_ID, store.STOREENT.STOREENT_ID	**tax.xml** CALCODE, CALRANGE, CALRLOOKUP, CALRULE, CALSCALE, CRULESCALE, JURST, JURSTGROUP, JURSTGPREL, STENCALUSG, TAXCGRY, TAXCGRYDS, TAXJCRULE **taxfulfill.xml** TAXJCRULE	**tax.xml** CALCODEDSC, CALSCALEDS

契約データ・グループのロード

Business 契約データ・グループでは、その他の WebSphere Commerce データ・グループとは異なる指示が必要です。契約テーブルは直接ロードされないため、WebSphere Test Environment 内で以下を行う必要があります。

1. テキスト・エディターで、以下のディレクトリーにある contract.xml ファイルをオープンします。
   - NT *drive*:¥WebSphere¥CommerceDev¥xml¥trading
   - 2000 *drive*:¥Program Files¥WebSphere¥CommerceDev¥xml¥trading
   - AIX /usr/WebSphere/CommerceDev/xml/trading
   - Solaris Linux /opt/WebSphere/CommerceDev/xml/trading
   - 400 /QIBM/ProdData/WebCommerce/xml/trading
2. STOREENT テーブルで、すべての &STORE_IDENTIFIER; を ID 値に置き換えます。
3. すべての &MEMBER_IDENTIFIER; を o=Default Organization, 0=Root Organization に置き換えます。ファイルを保管してクローズします。
4. 管理コンソールのサイト・ページにログオンします。
5. ブラウザーで次のコマンドを発行して、契約をインポートします。
   http://localhost:8000/webapp/wcs/stores/servlet/ContractImportApprovedVersion?fileName=contract.xml&targetStoreId=*store_Id*&URL=ContractDisplay

付録 F. 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、プログラムまたはサービスの操作性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。

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。実施権の許諾については、下記の宛先に書面にてご照会ください。

〒106-0032 東京都港区六本木 3 丁目 2-31
IBM World Trade Asia Corporation
Intellectual Property Law & 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。

本書は定期的に見直され、必要な変更 (たとえば、技術的に不適切な表現や誤植など) 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。

表示されている IBM の価格は IBM が小売り価格として提示しているもので、現行価格であり、通知なしに変更されるものです。卸価格は、異なる場合があります。

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。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。 お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。したがって IBM は、これらのサンプル・プログラムについて信頼

性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。お客様は、IBM のアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、 IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物にも、次のように、著作権表示を入れていただく必要があります。

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

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。

AIX	IBM	WebSphere
AS/400	IBM Payment Manager	
DB2	iSeries	
DB2 Universal Database	OS/400	
eServer	VisualAge	

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。

SET、SET ロゴ、SET Secure Electronic Transaction および Secure Electronic Transaction は、SET Secure Electronic Transaction LLC の商標です。

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。

UNIX は、The Open Group がライセンスしている米国およびその他の国における登録商標です。

他の会社名、製品名およびサービス名などはそれぞれ各社の商標または登録商標です。

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