IBM® WebSphere® Commerce

Catalog Manager ユーザーズ・ガイド

バージョン 5.4

IBM® WebSphere® Commerce

Catalog Manager ユーザーズ・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、 127 ページの『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、バージョン 5.4 (プログラム 5724-A18)

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 Catalog Manager User'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

始めに

本書 *IBM WebSphere Commerce Catalog Manager* ユーザーズ・ガイド は、WebSphere Commerce Catalog Manager について説明しています。特に本書には、以下のトピックに関する詳細があります。

- Catalog Manager のツールおよびユーティリティーを使用したデータのトランスフォームおよびロード
- Catalog Manager の Web エディターを使用したカタログ・データの処理
- Catalog Manager コマンド

本書の表記規則

本書では、以下のような規則を使用しています。

太文字	フィールド名、ボタン名、またはメニュー選択などの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) を示します。
モノスペース (Monospace)	表示されているとおりに入力するテキストの例や、ファイル名およびディレクトリー・パスを示します。
イタリック	強調のため、またユーザーが独自の値に置き換えることができる変数を表すために使用されます。
	作業を完了するために役立つ追加情報を示します。
Windows	WebSphere Commerce for Windows® に固有な情報を示します。
NT	WebSphere Commerce for Windows NT® に固有な情報を示します。
2000	WebSphere Commerce for Windows 2000® に固有な情報を示します。
AIX	WebSphere Commerce for AIX® に固有な情報を示します。
Solaris	WebSphere Commerce for Solaris™ オペレーティング環境ソフトウェアに固有な情報を示します。
Linux	WebSphere Commerce for Linux に固有な情報を示します。
400	WebSphere Commerce for IBM @server™ iSeries™ (以前の名称は AS/400®) に固有な情報を示します。
DB2	DB2® Universal Database に固有な情報を示します。
Oracle	Oracle® データベースに固有な情報を示します。
Business	WebSphere Commerce Business Edition に固有な情報を示します。

Professional　WebSphere Commerce Professional Edition に固有な情報を示します。

本書の対象読者

本書は、WebSphere Commerce Catalog Manager の使用法について理解する必要のあるサイトの開発者、管理者、および出資者や、この機能について理解する必要のある他のビジネス・ユーザーを対象としています。

特に、WebSphere Commerce Catalog Manager の種々の機能の使用法について理解する必要のある WebSphere Commerce のストア開発者およびストア開発者は、本書をお読みになる必要があります。

プログラムの拡張を実行する、WebSphere Commerce ストア開発者およびサイト管理者には、さらに以下の分野の知識がなければなりません。

- データベース・テクノロジー
- Enterprise JavaBeans™ のコンポーネント・アーキテクチャー
- Hypertext Markup Language (HTML)
- Java™
- JavaServer Pages™ テクノロジー
- VisualAge® for Java™、エンタープライズ版、バージョン 3.5 以上
- Extensible Markup Language (XML)

カテゴリー・マネージャーおよびプロダクト・マネージャーは、カタログ、製品、およびアイテムの要件をストア・デベロッパーに伝える方法を理解するために、本書をお読みになる必要があります。また、Web エディターを使用してストアのカタログを更新する方法も理解していなければなりません。

詳細情報の入手先

このユーザーズ・ガイドは、 WebSphere Commerce の Web サイトから Adobe Portable Document Format (PDF) 形式で入手できます。この資料の最新版および WebSphere Commerce に関連したその他の情報については、以下の場所にある WebSphere Commerce のテクニカル・ライブラリーの Web サイトを調べてください。

- 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

第 1 部 Catalog Manager の概要

WebSphere Commerce Catalog Manager は、特定のカタログ管理問題を解決するために必要な順序でつなぎ合わせることのできる様々な機能を備えた汎用ツールキットを提供しています。これには、WebSphere Commerce スキーマを対象としたカスタマイズを処理するのに十分な柔軟性があります。

Catalog Manager には、複数のソースからの情報を、統合された WebSphere Commerce システムに集約し、様々なデータをすべて情報管理の標準手段として XML ファイルを使用する標準のカタログおよび製品定義フォーマットに再マップする能力があります。

Catalog Manager は、以下を行うための手段を提供しています。

- ASCII および XML ファイル形式の複数の入力ソースのデータを、WebSphere Commerce にインポートする
- データを ASCII から XML フォーマットにトランスフォームし、再び元に戻す
- ある XML フォーマットのデータを別の XML フォーマットに再マップする
- 複数の入力ストリームのデータを 1 つの集約データベースに集約する
- Web ブラウザー・インターフェースを介してデータを作成 / 編集 / 削除する

Catalog Manager には、以下が含まれています。

- Catalog Manager ローダー・パッケージ

  このパッケージは主に、データを作成して WebSphere Commerce データベース内にロード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。大量のデータのロード、および WebSphere Commerce データベース内のデータの更新を行うには、このローダー・パッケージを使用することができます。

  ローダー・パッケージを使用すれば、さらに以下を行うことができます。

  – データベースからデータを XML 文書として抽出する
  – XML データを代替の XML フォーマットにトランスフォームする
  – データを、文字区切り可変長フォーマットと XML データ・フォーマットの間でトランスフォームする
- Catalog Manager 管理ツール

  さらに、Catalog Manager にはその機能の管理に役立つユーザー・インターフェースを備えた以下の 2 つのツールが含まれています。

  – テキスト・トランスフォーメーション・ツール
  – XSL エディター
- Catalog Manager Web エディター

  Web エディターを使用すると、Web ブラウザーを使用してカタログ・データを作成、削除、および変更することができます。

第 2 部 データのトランスフォーム、ロード、および抽出

第 1 章 概要

Catalog Manager ローダー・パッケージは主に、データを作成して WebSphere Commerce データベース内にロードするためのユーティリティーで構成されています。大量のデータのロード、および WebSphere Commerce データベース内のデータの更新を行うには、このローダー・パッケージを使用することができます。ローダー・パッケージを使用すれば、さらに以下を行うことができます。

- データベースからデータを XML 文書として抽出する
- XML データを代替の XML フォーマットにトランスフォームする
- データを、文字区切り可変長フォーマットと XML データ・フォーマットの間でトランスフォームする

Catalog Manager には、さらに以下の 2 つのツールと、その機能の管理を支援するユーザー・インターフェースが含まれています。

- テキスト・トランスフォーメーション・ツール
- XSL エディター

ユーティリティー

WebSphere Commerce ローダー・パッケージには、データを作成して WebSphere Commerce データベース内にロードするためのユーティリティーが含まれています。これらのユーティリティーは、量にかかわりなくデータのロードや、WebSphere Commerce データベース内のデータの更新を行うために使用することができます。

ロード・プロセスは、データを WebSphere Commerce データベース内に移動するために必要な以下のステップで成り立っています。

1. DTD ジェネレーターを使用した DTD の生成
2. ID リゾルバーを使用した入力ファイル内の ID の解決
3. ローダーを使用したデータのロード

さらに、ローダー・パッケージには、データベースから XML 文書としてデータを抽出したり、 XML データを代替の XML フォーマットにトランスフォームしたりするためのユーティリティーが含まれています。

ローダー・パッケージは、以下のユーティリティーで構成されています。

- **テキスト・トランスフォーマー**

  テキスト・トランスフォーマーは、データを、文字区切り可変長フォーマットと XML データ・フォーマットの間でトランスフォームします。

  詳しくは、 11 ページの『第 2 章 テキストのトランスフォーム』を参照してください。

- **XML トランスフォーマー**

  XML トランスフォーマーは、 XML 文書内のデータを、他のユーザーやシステムが必要に応じて使用するために、変更、集約して、代替の XML フォーマットに再マップします。

詳しくは、 19 ページの『第 3 章 XML データのトランスフォーム』を参照してください。

- **DTD ジェネレーター**

  DTD は、XML データ文書内で使用できる構造化エレメントとマークアップ定義を指定します。たとえば、DTD は文書内で使用されるエレメントをリストし、各エレメントが取ることのできる属性を指定します。

  DTD ジェネレーターは、データベース・スキーマに基づいてローダーに適合する DTD を作成するユーティリティーです。この DTD には、ローダーがデータをインポートするテーブルと列についての記述があります。 DTD ジェネレーターは、Catalog Manager Web エディターと共に使用することのできるスキーマと詳細な XML 文書を作成することもできます。

  DTD ジェネレーターは、データが準拠している必要のあるターゲットのデータベースに基づいて DTD を生成します。この DTD は、ロード・プロセスで使用されます。 DTD ジェネレーターは、1 回だけ実行する必要があります。

  詳しくは、 25 ページの『第 4 章 DTD およびスキーマの生成』を参照してください。

- **ID リゾルバー**

  ID リゾルバーは、ID の必要な XML エレメントの ID を生成するユーティリティーです。 XML の内容にすでに ID がある場合は、ID リゾルバーを稼働する必要はありません。

  ID リゾルバーは、関連する ID で XML エレメントのセットを更新します。ローダー・パッケージの XML ファイルは、ターゲットのデータベース・スキーマに直接マップするため、このステップは不可欠です。そのため、これらは ID を持っていなければなりません。

  ID リゾルバーには、エラーがある場合に例外文書を生成するエラー・レポーターが含まれています。

  詳しくは、 31 ページの『第 5 章 ID の解決』を参照してください。

- **ローダー**

  ローダーは、有効な正しいフォームの XML を入力データとして使用してデータベースにデータをロードします。 XML 文書のエレメントはデータベース内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。ローダーは、システムにデータをロードするための最も一般的な手段です。

  ローダーを使用すると、テーブルに対して列レベルの更新を行うことができます。また、データベースからデータを削除することもできます。

  以下の例は、ローダーに対する、正しいフォームの有効な XML 入力データの抜粋です。

```
<ADDRBOOK
  ADDRBOOK_ID="11801"
  MEMBER_ID="100"
  DISPLAYNAME="Friends"
  DESCRIPTION="All my friends"
    TYPE="P"
/>
```

  上記の例で、ADDRBOOK はテーブル名であり、更新される列は ADDRBOOK エレメントの属性によって示されています。

ローダー・ユーティリティーには、以下の機能が含まれます。

– **エラー・レポーター**

ローダーには、エラーがある場合に例外文書を生成するエラー・レポーターが含まれています。

– **商品アドバイザーの検索スペースの同期**

商品アドバイザーの検索スペースは、 WebSphere Commerce カタログから情報を抽出し、その情報を検索に適したフォームで表示することによって作成されます。カタログが更新されると、この検索スペースとカタログは非同期になります。検索スペースのカタログとの同期の遅れを防ぐには、ローダーの商品アドバイザー検索スペースの同期機能を使用可能にします。

商品アドバイザー検索スペースの同期が使用可能になっている場合、ローダーは、スケジューラー・ジョブ・テーブル SCHCONFIG とスケジューラー・ジョブ・インスタンス・テーブル SCHSTATUS にコマンド情報を追加して、以下の表に示された適切な検索スペース・コマンドをスケジュールします。

テーブル	モード	検索スペース・コマンド	状況
CATENTRY	更新	UpdateSearchSpaces	発行フラグの更新
	削除	RemoveProductFromAllSearchSpaces	商品の削除
CATENTDESC	追加	UpdateSearchSpaces	説明/言語の追加
	更新	UpdateSearchSpaces	発行フラグの更新
			説明データの更新
	削除	UpdateSearchSpaces	説明/言語の除去
LISTPRICE	追加	UpdateSearchSpaces	表示価格の追加
	更新	UpdateSearchSpaces	表示価格の更新
	削除	UpdateSearchSpaces	表示価格の削除
ATTRVALUE	追加	UpdateSearchSpaces	ユーザー定義値の追加
	更新	UpdateSearchSpaces	ユーザー定義値の更新
	削除	UpdateSearchSpaces	ユーザー定義値の削除
CATENTATTR	追加	UpdateSearchSpaces	属性の追加
	更新	UpdateSearchSpaces	属性の更新
	削除	UpdateSearchSpaces	属性の削除
CATGPENREL	追加	AddProductsToSearchSpace	カテゴリーへの商品の追加
	削除	RemoveProductsFromSearchSpace	カテゴリーからの商品の除去

詳しくは、 47 ページの『第 6 章 データのロード』を参照してください。

- **エクストラクター**

エクストラクターは、データベースに対する照会を使用して、選択されたデータのサブセットをデータベースから XML 文書へ抽出します。データベースから抽出するデータは、抽出フィルター XML 文書を使用して指定します。

エクストラクターの機能は、ローダーの機能の逆です。 WebSphere Commerce データベースからデータの選択サブセットを XML ファイル形式で抽出するには、エクストラクターを使用します。ユーザーは、たとえば、次の休日に関連した商品に関連するデータを抽出したり、統合データベースからデータを抽出して他のシステムで使用したりすることができます。

詳しくは、 67 ページの『第 7 章 データの抽出』を参照してください。

ローダー・パッケージには、ロガー機能も含まれています。ローダー・パッケージ内のそれぞれのユーティリティーは、プログラム・トレース情報を提供するだけでなく、成功、失敗、およびエラーを示すメッセージを作成します。

管理ツール

Catalog Manager には、その機能の管理に役立つ以下のツールが含まれています。

テキスト・トランスフォーメーション・ツール

テキスト・トランスフォーメーション・ツールは、管理者が、文字区切り可変長フォーマットと XML データ・フォーマットの間のデータのトランスフォーム処理に必要な情報を準備するのを助けます。

XSL エディター

XML トランスフォーマーは、 Extensible Stylesheet Language (XSL) を使用して、 XML ファイルを他の XML ファイルにトランスフォームするためのルールを定義します。 XSL エディターのマッピング機能は、ソース文書型定義 (DTD) 内のエレメントをターゲット DTD 内のエレメントに関連付けるために用いられるビジュアル・インターフェースを管理者に提供します。

第 2 章 テキストのトランスフォーム

Catalog Manager には、他のツール (スプレッドシート・プログラムなど) による ASCII ファイル出力を、 WebSphere Commerce データベースに入れることのできる XML データ・フォーマットに変換する機能があります。

テキスト・トランスフォーメーション・ツールは、管理者が、文字区切り可変長フォーマットと XML データ・フォーマットの間のデータのトランスフォーム処理に必要な情報を準備するのを助けます。以下のビューが提供されています。

1. 「Text Schema Edit (テキスト・スキーマの編集)」ビューでは、トランスフォーメーションに使用される 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)」ビューでは、トランスフォーメーション・プロセスの実行に使用され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)」ビューでは、トランスフォーメーション・プロセスを開始することができます。

テキスト・トランスフォーメーション・ツールの立ち上げ

テキスト・トランスフォーメーション・ツールを立ち上げるには、以下の WebSphere Commerce ディレクトリーにある適切なスクリプトまたはコマンドを使用します。

- NT *drive*:¥WebSphere¥CommerceServer¥bin¥TextTrans.cmd
- 2000 *drive*:¥Program Files¥WebSphere¥CommerceServer¥bin¥ TextTrans.cmd
- AIX /usr/WebSphere/CommerceServer/bin/texttrans.sh
- Solaris Linux /opt/WebSphere/CommerceServer/bin/texttrans.sh

400 iSeries 環境では、まず管理者は、テキスト・トランスフォーメーション・ツールを実行するために必要なファイルを Windows NT または Windows 2000 マシンにコピーする必要があります。以下に、それを行う方法の例を示します。

1. Windows マシン上に新しいディレクトリー (たとえば *drive*:¥TextTrans) を作成します。
2. *drive*:¥TextTrans の下に、以下のような新しいサブディレクトリーを作成します。

   ```
   ¥bin
   ¥lib¥loader
   ¥wcsadmin
   ```

3. 以下の図表に従って、iSeries マシンからそれぞれのディレクトリーにファイルをコピーします。

コピー元		コピー先
/QIBM/ProdData/WebCommerce		drive:¥TextTrans
/bin		¥bin
/TextTrans.cmd	-------->	¥TextTrans.cmd
/lib/loader		¥lib¥loader
/wcmxmlp.jar	-------->	¥wcmxmlp.jar
/wcmxslt.jar	-------->	¥wcmxslt.jar
/wcsadmin		¥wcsadmin
/acsxml.ico	-------->	¥acsxml.ico
/swing.jar	-------->	¥swing.jar
/TextTransformerUI.zip	-------->	¥TextTransformerUI.zip
/TextTransform.cnt	-------->	¥TextTransform.cnt
/TextTransform.hlp	-------->	¥TextTransform.hlp
/TextTransform.reg	-------->	¥TextTransform.reg

4. 以下のテキストを挿入して、Windows マシン上の TextTrans.cmd ファイルを変更します。

   ```
   set WCS_HOME=drive:¥TextTrans
   ```

   これを、以下のテキストの前に挿入します。

   ```
   set lib=%WCS_HOME%¥lib¥loader
   ```

5. Windows マシン上で TextTrans.cmd を実行して、テキスト・トランスフォーメーション・ツールを立ち上げます。

「Text Schema Edit (テキスト・スキーマの編集)」ビューの使用

以下は、文字区切り可変長フォーマットと XML フォーマットの間でデータをトランスフォームするために、 XML スキーマ・ファイルを作成および変更するための手順です。

スキーマ・ファイルの作成

新しいスキーマ・ファイルを作成するには、以下のようにします。

1. テキスト・トランスフォーメーション・ツールを立ち上げます。
2. **「ファイル > New (新規)」**を選択するか、ツールバー上の「新規」アイコンをクリックします。
3. パスを選択し、作成する XML スキーマ・ファイルの名前を入力します。

   注: デフォルトのファイル名は ″Default.xml″ です。
4. **「CSV Format (CSV フォーマット)」**を選択します。

   注: 「WebSphere Commerce Suite Format (WebSphere Commerce Suite フォーマット)」は選択しないでください。このオプションは、WebSphere Commerce の初期のバージョンでのみ使用されます。
5. 終了したら、**「OK」**をクリックします。

この後は、以下に示すステップに従って新しいエレメントと属性を作成できます。

スキーマ・ファイルのオープン

スキーマ・ファイルをオープンするには、以下のようにします。

1. テキスト・トランスフォーメーション・ツールを立ち上げます。
2. **「ファイル > オープン」**を選択するか、ツールバーの「オープン」アイコンをクリックします。
3. オープンするスキーマ・ファイルを選択します。
4. 終了したら、**「OK」**をクリックします。

この後は、以下に示すステップに従ってエレメントと属性を変更できます。

スキーマ・ファイルの保管

スキーマ・ファイルを保管するには、以下のようにします。

1. テキスト・トランスフォーメーション・ツールを立ち上げます。
2. スキーマに加えた変更を保管するには、**「ファイル > 保管」**を選択するか、ツールバーの「保管」アイコンをクリックします。
3. スキーマのコピーを新しい名前で保管するには、以下のようにします。
   a. **「ファイル > 別名保管」**をクリックします。
   b. パスを選択し、作成する XML ファイルの名前を入力します。
   c. 終了したら、**「OK」**をクリックします。

スキーマ・ファイルの編集

スキーマ・ファイルを編集するには、上記のとおりにスキーマ・ファイルをオープンしてから、以下のようにします。

エレメントの追加

エレメントを追加するには、以下のようにします。

1. 「Element List (エレメント・リスト)」フィールドに新しいエレメントの名前を入力します。
2. 「行の追加」アイコンをクリックします。

エレメントの除去

エレメントを除去するには、以下のようにします。

1. エレメント名を選択します。
2. 「行の除去」アイコンをクリックします。

エレメントの置換

エレメントを置換するには、以下のようにします。

1. 「Element List (エレメント・リスト)」フィールドに新しいエレメントの名前を入力します。
2. 置換するエレメントの名前を選択します。
3. 「行の置換」アイコンをクリックします。

エレメントを 1 行上に移動する

エレメントを 1 行上に移動するには、以下のようにします。

1. 上に移動するエレメントの名前を選択します。
2. 「行を上に移動」アイコンをクリックします。

エレメントを 1 行下に移動する

エレメントを 1 行下に移動するには、以下のようにします。

1. 下に移動するエレメントの名前を選択します。
2. 「行を下に移動」アイコンをクリックします。

属性の追加

属性を追加するには、以下のようにします。

1. 「属性リスト」フィールドに新しい属性の名前を入力します。
2. 「行の追加」アイコンをクリックします。

属性の除去

属性を除去するには、以下のようにします。

1. 属性名を選択します。
2. 「行の除去」アイコンをクリックします。

属性の置換

属性を置換するには、以下のようにします。

1. 「属性リスト」フィールドに新しい属性の名前を入力します。

2. 置換する属性の名前を選択します。
3. 「行の置換」アイコンをクリックします。

属性を 1 行上に移動する

属性を 1 行上に移動するには、以下のようにします。

1. 上に移動する属性の名前を選択します。
2. 「行を上に移動」アイコンをクリックします。

属性を 1 行下に移動する

属性を 1 行下に移動するには、以下のようにします。

1. 下に移動する属性の名前を選択します。
2. 「行を下に移動」アイコンをクリックします。

スキーマ・ファイル構造の変更

「Text Schema Edit (テキスト・スキーマの編集)」ビューの下方のペインにある「File Structure (ファイル構造)」ビューは、文字区切り可変長ファイルのレイアウトを説明します。以下は、予期されるファイル構造内で必要なフィールドです。

Field Separator: (フィールド区切り文字:)
属性値を区切る区切り文字を指定します。デフォルトはコンマ (「,」) です。

Record Separator: (レコード区切り文字:)
データ・レコードを区切る区切り文字を指定します。デフォルトは「

」(¥r¥n へのエンティティー参照と同じ) です。

String Delimiter: (ストリング区切り文字:)
ストリングの開始および終了境界を示す区切り文字を指定します。デフォルトは単一引用符 (「'」) です。

Header Included: (ヘッダーの組み込み:)
テキスト・データ・ファイル内にヘッダー行がある場合は「true」、テキスト・データ・ファイル内にヘッダー行がない場合は「false」として指定されるブール値です。この例ではヘッダーが XML タグ名として使用されるので、ヘッダー行がある場合、それは XML タグ名のルールに準拠していなければなりません。デフォルトは「false」です。

Number of header lines: (ヘッダー行の数:)
テキスト・データ・ファイル内に存在するヘッダー行の行数を指定します。デフォルトはゼロ (「0」) です。

データを XML から文字区切り可変長フォーマットにトランスフォームするためのスキーマ・ファイルの準備

「Text Schema Edit (テキスト・スキーマの編集)」ビューを使って作成された XML スキーマ・ファイルを使用して、データを XML フォーマットから文字区切り可変長フォーマットにトランスフォームする場合、トランスフォーメーションを処理する前に、テキスト・エディターを使用してスキーマ・ファイル内のデータ・タイプ仕様を「CSV フォーマット」から「XML フォーマット」に変更する必要があります。

「Transformation Command Edit (トランスフォーメーション・コマンドの編集)」ビューの使用

新しいコマンド・ファイルの作成、既存のコマンド・ファイルのオープン、またはコマンド・ファイルへの変更の保管は、「Transformation Command Edit (トランスフォーメーション・コマンドの編集)」ビューを使用して行えます。デフォルトのコマンド・ファイル名は ″Manifest.txt″ です。

ユーザーは、新しいコマンドの作成、コマンドの除去、編集済み情報によるコマンドの置換、またはコマンドの順序の変更を行うことができます。

注: コマンド・ファイルは、コマンド・テーブルが更新されるたびに自動的に保管されます。

コマンドの作成

コマンドを作成するには、以下のようにします。

1. ソース・ファイルを指定します。これは、文字区切り可変長フォーマット・ファイル (.csv 拡張子が付いたもの) か、 XML フォーマット・ファイル (.xml 拡張子が付いたもの) のいずれかです。
2. トランスフォーメーションで使用される XML スキーマ・ファイルを指定します。
3. トランスフォーメーション・プロセス中に作成または変更される出力ファイルの名前 (つまり、新しいデータが格納される場所) を指定します。これは、XML フォーマット・ファイル (.xml 拡張子が付いたもの) か、文字区切り可変長フォーマット・ファイル (.csv 拡張子が付いたもの) のいずれかです。
4. コマンド・モードを指定します。

   出力ファイルを作成する場合は**「作成」**を、出力データを既存のデータ・ファイルに付加する場合は**「Append (付加)」**を選択します。
5. 「行の追加」アイコンをクリックします。

コマンドの除去

コマンドを除去するには、以下のようにします。

1. コマンドを選択します。
2. 「行の除去」アイコンをクリックします。

コマンドの編集または置換

コマンドを編集または置換するには、以下のようにします。

1. 「コマンドの編集」アイコンをクリックします。

   適切な入力フィールドに行データが入っています。
2. 適切な入力フィールド内のテキストを変更します。
3. 「行の置換」アイコンをクリックして行を更新します。

コマンドを 1 行上に移動する

コマンドを 1 行上に移動するには、以下のようにします。

1. コマンドを選択します。
2. 「行を上に移動」アイコンをクリックします。

注: これにより、トランスフォーメーション・プロセスの順序が変わります。

コマンドを 1 行下に移動する

コマンドを 1 行下に移動するには、以下のようにします。

1. コマンドを選択します。
2. 「行を下に移動」アイコンをクリックします。

注: これにより、トランスフォーメーション・プロセスの順序が変わります。

コマンドのクリア

コマンドを消去するには、以下のようにします。

1. コマンドを選択します。
2. 「入力フィールドのクリア」アイコンをクリックします。

   これで、ソース・ファイル、スキーマ・ファイル、および出力ファイルのフィールド内のテキストは消去されます。

「Transformation Process (トランスフォーメーション・プロセス)」ビューの使用

テキスト・トランスフォーメーション・プロセスを立ち上げるには、以下のようにします。

1. 「ファイル」フィールドに、パラメーター・ファイルの名前を入力またはブラウズ表示します。
2. **「処理」**をクリックします。

「処理」ボタンの下の出力域に、トランスフォーメーション・プロセスの状況を示す情報が表示されます。この出力情報は、テキスト域の下部にある「保管」ボタンをクリックすることによって保管できます。また、「クリア」ボタンをクリックすれば、すべての状況情報を消去できます。

第 3 章 XML データのトランスフォーム

Extensible Stylesheet Language (XSL) は、以下のものを提供します。

1. XML 文書のフォーマットを指定するための言語
2. XML ファイルを、規則的に構造化された別のファイルにトランスフォームする方法について説明する言語

   XML ファイルを、異なる XML スキーマまたは DTD に準拠する別の XML ファイルにトランスフォームするには、 XSL のトランスフォーメーション機能を使用することができます。

XML ファイルを代替の XML フォーマットにトランスフォームするには、トランスフォーム XSL ルール・ファイルを使用してトランスフォーメーション・ルールを指定する必要があります。

以下は、MemberSubsystemExtracted.xml 内のデータを、 MemberSubsystem.xsl をトランスフォーム XSL ルール・ファイルとして使用し、日本語を各国語として用いてトランスフォームする例です。

- NT 2000 AIX Solaris Linux

```
java com.ibm.wca.XMLTransformer.XMLTransformer -infile MemberSubsystemExtracted.xml
-transform MemberSubsystem.xsl -outfile TransMbrStr.xml -param 'language="-10"'
```

- 400

```
QWEBCOMM/TRNWCSXML INFILE(MemberSubsystemExtracted.xml)
TRANSFORM(MemberSubsystem.xsl) INSTROOT(/QIBM/UserData/WebCommerce/instances/my_inst)
OUTFILE(TransMbrStr.xml) PARAM('language=-10')
```

XML トランスフォーマーは、 XSL を使用して、XML ファイルを別の XML ファイルにトランスフォームするためのルールを定義します。 XSL エディターのマッピング機能は、ソース DTD 内のエレメントをターゲット DTD 内のエレメントに関連付けるためのビジュアル・インターフェースを提供します。 2 つの DTD については、最初の (ソース) DTD に準拠する XML ファイルを 2 番目の (ターゲット) DTD に準拠する XML ファイルにトランスフォームする方法を決定する XSL ルールを作成することができます。

XSL エディターの立ち上げ

XSL エディターを立ち上げるには、以下の WebSphere Commerce ディレクトリーにある適切なスクリプトまたはコマンドを使用します。

- NT *drive*:¥WebSphere¥CommerceServer¥bin¥XSLEditor.cmd
- 2000 *drive*:¥Program Files¥WebSphere¥CommerceServer¥bin¥ XSLEditor.cmd
- AIX /usr/WebSphere/CommerceServer/bin/xsleditor.sh
- Solaris Linux /opt/WebSphere/CommerceServer/bin/xsleditor.sh

400 iSeries 環境では、まず管理者は、 XSL エディターを実行するために必要なファイルを Windows NT または Windows 2000 マシンにコピーする必要があります。以下に、それを行う方法の例を示します。

1. Windows マシン上に新しいディレクトリー (たとえば *drive*:¥XMLTrans) を作成します。
2. *drive*:¥TextTrans の下に、以下のような新しいサブディレクトリーを作成します。

   ```
   ¥bin
   ¥lib¥loader
   ¥wcsadmin
   ```

3. 以下の図表に従って、iSeries マシンからそれぞれのディレクトリーにファイルをコピーします。

   ```
   コピー元                                  コピー先
   /QIBM/ProdData/WebCommerce              drive:¥XMLTrans
     /bin                                    ¥bin
       /XSLEditor.cmd          --------->      ¥XSLEditor.cmd
     /lib/loader                             ¥lib¥loader
       /wcmxmlp.jar            --------->      ¥wcmxmlp.jar
       /wcmxslt.jar            --------->      ¥wcmxslt.jar
     /wcsadmin                               ¥wcsadmin
       /config.dtd             --------->      ¥acsxml.ico
       /swing.jar              --------->      ¥swing.jar
       /XML_transform.ico      --------->      ¥XML_transform.ico
       /XMLTransformerUI.zip   --------->      ¥XMLTransformerUI.zip
       /XMLTransformWP.xml     --------->      ¥XMLTransformWP.xml
       /XMLTransformWP.dtd     --------->      ¥XMLTransformWP.dtd
       /XMLRuleConfig.xml      --------->      ¥XMLRuleConfig.xml
       /XMLTransform.cnt       --------->      ¥XMLTransform.cnt
       /XMLTransform.hlp       --------->      ¥XMLTransform.hlp
       /XMLTransform.reg       --------->      ¥XMLTransform.reg
   ```

4. 以下のテキストを挿入して、Windows マシン上の XSLEditor.cmd ファイルを変更します。

   ```
   set WCS_HOME=drive:¥XMLTrans
   ```

   これを、以下のテキストの前に挿入します。

   ```
   set lib=%WCS_HOME%¥lib¥loader
   ```

5. Windows マシン上で XSLEditor.cmd を実行して、XML トランスフォーメーション・ツールを立ち上げます。

マッピング・ルール構築域の処理

XSL マッピングを立ち上げると、「Mapping Rule Building Area (マッピング・ルール構築域)」ウィンドウが表示されます。このウィンドウは、マッピング・ルール構築域を管理するために使用します。

マッピング・ルール構築域の作成

マッピング・ルール構築域を作成するには、以下のようにします。

1. XSL エディターを立ち上げます。
2. ドロップダウン・メニューから**「New (新規)」**を選択します。
3. 「名前」フィールドに、新しいマッピング・ルール構築域の名前を入力します。
4. 「説明」フィールドに、新しいマッピング・ルール構築域の簡略説明を入力します。
5. 「Source Schema (ソース・スキーマ)」フィールドに、既存ファイルの名前を入力するか、ソース・スキーマとして使用する既存ファイルにナビゲートします。
6. 「Target Schema (ターゲット・スキーマ)」フィールドに、既存ファイルの名前を入力するか、ターゲット・スキーマとして使用する既存ファイルにナビゲートします。
7. 「XSL Rule File (XSL ルール・ファイル)」フィールドに、作成する新しいルール・ファイルの名前を入力します。

   ここには、完全パスを入力できます。パスを指定しないと、ファイルは現行作業ディレクトリーに作成されます。
8. **「オープン」**をクリックして、新しいマッピング・ルール構築域を作成し、オープンします。

マッピング・ルール構築域の変更

マッピング・ルール構築域を変更するには、以下のようにします。

1. XSL エディターを立ち上げます。
2. ドロップダウン・メニューから、変更するマッピング・ルール構築域を選択します。
3. **「オープン」**をクリックして、マッピング・ルール構築域をオープンします。
4. 変更したいフィールドを更新します。
5. **「保管」**をクリックして、変更を保管します。

マッピング・ルール構築域の削除

マッピング・ルール構築域を削除するには、以下のようにします。

1. XSL エディターを立ち上げます。
2. ドロップダウン・メニューから、削除するマッピング・ルール構築域を選択します。
3. 「除去」ボタンをクリックして、エントリーを除去します。

   注: マッピング・ルール構築域を除去しても、物理ファイルはディスクから削除されません。

XSL エディターの使用

XSL エディターを使用してマッピング・ルール構築域をオープンすると、 XSL エディターのメイン・ウィンドウに、そのマッピング・ルール構築域が表示されます。

XSL エディター・メイン・ウィンドウの左側のペインに、「Source Schema (ソース・スキーマ)」と表示されたソース DTD の階層図が表示されます。右側のペインには、「Target Schema (ターゲット・スキーマ)」と表示されたターゲット DTD の階層図が表示されます。

マッピング・ルールの作成

エレメントからエレメントへのマッピング

ソース階層からエレメントを選択してドラッグし、それをターゲット階層内のエレメントの上でドロップします。これで、XSL ルールが生成され、ウィンドウの下の方にある「Mapping Rule (マッピング・ルール)」ビューに表示されます。

以下は、生成された XSL ルールの例です。

```
<xsl:template match="merchant">
  <xsl:element name="MERCHANT">
  </xsl:element>
</xsl:template>
```

注: 必要であっても存在していない先祖関係があれば、自動的に生成されます。

属性から属性へのマッピング

ソース階層から属性を選択してドラッグし、それをターゲット階層内の属性の上でドロップします。これで、XSL ルールが生成され、ウィンドウの下の方にある「Mapping Rule (マッピング・ルール)」ビューに表示されます。

以下は、生成された XSL ルールの例です。

```
<xsl:attribute name="MEADDR1">
  <xsl:apply-templates select="@mecmail1"/>
</xsl:attribute>
```

注: 必要であっても存在していない先祖関係があれば、自動的に生成されます。

カスタム・マッピング式の作成

カスタム・マッピング式を作成するには、まずターゲット階層からエレメントまたは属性を選択します。それから、右マウス・ボタン・クリックして「Create Custom Expression (カスタム式の作成)」メニューを選択します。「Create Custom Expression (カスタム式の作成)」ウィンドウが表示されます。このウィンドウの 2 つのドロップダウン・メニューには使用可能なテンプレートとルール式のリストが入っています。カスタム式は、以下のようにして完成させます。

1. カスタム式を追加するテンプレートを選択します。
2. 作成するルール式を選択します (たとえば、**Constant Expression (定数式)**)。
3. テーブルにリストされている各パラメーターの「値」フィールドに値を入力し、**Enter** を押してその値をコミットします。

4. **「OK」**をクリックして、作成ステップを完了します。あるいは、ルールを作成しないでキャンセルする場合は**「キャンセル」**をクリックします。

生成される XSL ルールは、ルール構成ファイル (XSLRuleConfig.xml) で定義されているカスタム式に基づきます。ユーザーは必要に応じてルール構成ファイルを変更し、新しいルールを追加することができます。ルールを「Rule Expressions (ルール式)」リスト内で使用できるようにするには、そのルールの「可視性」属性を「true」に設定します。

マッピング・ルールの削除

マッピング・ルールを削除するには、以下のようにします。

1. 「Mapping Rule (マッピング・ルール)」テーブルからルールを選択します。
2. 右マウス・ボタン・クリックして、**「削除」**を選択します。

   このルールとそのすべての子孫が削除されます。

   注: 更新されたマッピング・ルールおよび生成された XSL ルールは、自動的に保持されます。

XML トランスフォーメーションの処理

XML トランスフォーメーションを処理するには、以下のようにします。

1. **「ツール > Transform (トランスフォーム)」**を選択して、「Process Transform (トランスフォームの処理)」ウィンドウを立ち上げます。
2. 以下のように、必須フィールドを完成させます。
   a. 「Input XML File (入力 XML ファイル)」フィールドに、ソース XML データ・ファイルのパスと名前を入力するかまたはブラウザー表示します。
   b. 「XSL Rule File (XSL ルール・ファイル)」フィールドに、トランスフォーメーションに使用するマッピング・ルール・ファイルのパスと名前を入力するかまたはブラウザー表示します。

      マッピング・ルール構築域がオープンしている場合、このフィールドには、そのマッピング・ルール構築域で現在オープンしているマッピング・ルール・ファイルのパスがあらかじめ入力されています。
   c. 「Output XML File (出力 XML ファイル)」フィールドに、トランスフォーメーション・プロセス中に作成される新しい XML データ・ファイルのパスと名前を入力するかまたはブラウズ表示します。
3. **「開始」**をクリックして、XML トランスフォーメーション・プロセスを開始します。あるいは、トランスフォーメーションの処理をせずにウィンドウを終了する場合は、**「クローズ」**をクリックします。

「Mapping Rule (マッピング・ルール)」テーブルのカスタマイズ

「Mapping Rule (マッピング・ルール)」テーブルをカスタマイズするには、以下のようにします。

1. テーブルの中の列を非表示にするには、テーブル内のセルを右マウス・ボタン・クリックして、**「Hide column (列を非表示にする)」**を選択します。
2. テーブル内の非表示にされた列を表示するには、以下のようにします。
   a. テーブル内のセルを右マウス・ボタン・クリックします。

b. **「Show columns (列を表示する)」**を選択して、非表示にされている列のリストを立ち上げます。

c. このリストから列を選択します。

注: 複数の列を選択するには、 **Shift** を押しながら列名をクリックします。

d. 選択した列を表示するには**「OK」**を、操作をキャンセルするには**「キャンセル」**をクリックします。

3. テーブルの中の非表示にされているすべての列を表示するには、テーブル内のセルを右マウス・ボタン・クリックして、**「Show all columns (すべての列を表示する)」**を選択します。

   すべての列がデフォルトの順序で表示されます。

完全な XSL ルール / 値式の表示

「Value Expression (値式)」または「XSL Rule (XSL ルール)」列でセルをクリックすると、選択されている行についての完全なルール内容が含まれたウィンドウが表示されます。

第 4 章 DTD およびスキーマの生成

DTD ジェネレーターは、ローダー・パッケージと共に使用する DTD とスキーマを作成することができます。 DTD ジェネレーターはデータベース・テーブル名の含まれた入力ファイルを使用し、ユーザーが DTD Generate コマンドを呼び出す方法に応じて、 DTD のみか、DTD とスキーマ (データベースについて説明する詳細な XML ファイルを含む) を作成します。

DTD ジェネレーターのセットアップ

ローダー DTD は、WebSphere Commerce データベース・スキーマに直接マップします。各テーブルがエレメントで、各列が属性です。

例: ローダー DTD のデータベース・スキーマへのマッピング

CATENTRY テーブルの DDL ステートメント

```
CREATE TABLE
   "CATENTRY" (
   "CATENTRY_ID"   BIGINT       NOT NULL ,
   "MEMBER_ID"     BIGINT       NOT NULL ,
   "CATENTTYPE_ID" CHAR(16)     NOT NULL ,
   "MARKFORDELETE" INTEGER      NOT NULL ,
   "PARTNUMBER"    VARCHAR(64)  NOT NULL ,
   "MFPARTNUMBER"  VARCHAR(64)           ,
   "MFNAME"        VARCHAR(64)           ,
   "URL"           VARCHAR(254)          ,
   "FIELD1"        INTEGER               ,
   "FIELD2"        INTEGER               ,
   "FIELD3"        DECIMAL(20,5)         ,
   "FIELD4"        VARCHAR(254)          ,
   "FIELD5"        VARCHAR(254)          ,
   "LASTUPDATE"    TIMESTAMP             ,
   "OID"           VARCHAR(64)           ,
   "ONSPECIAL"     INTEGER               ,
   "ONAUCTION"     INTEGER               ,
   "BUYABLE"       INTEGER               ,
   "BASEITEM_ID"   INTEGER               ,
   "CLASSIFGRP_ID" INTEGER               ,
   "ITEMSPC_ID"    INTEGER               ,
   "STATE"         INTEGER
);
```

DTD

```
<!ELEMENT CATENTRY EMPTY>
<!ATTLIST CATENTRY
    CATENTRY_ID      CDATA  #REQUIRED
    MEMBER_ID        CDATA  #REQUIRED
    CATENTTYPE_ID    CDATA  #REQUIRED
    MARKFORDELETE    CDATA  #REQUIRED
    PARTNUMBER       CDATA  #REQUIRED
    MFPARTNUMBER     CDATA  #IMPLIED
    MFNAME           CDATA  #IMPLIED
    URL              CDATA  #IMPLIED
    FIELD1           CDATA  #IMPLIED
    FIELD2           CDATA  #IMPLIED
    FIELD3           CDATA  #IMPLIED
    FIELD4           CDATA  #IMPLIED
    FIELD5           CDATA  #IMPLIED
    LASTUPDATE       CDATA  #IMPLIED
    OID              CDATA  #IMPLIED
    ONSPECIAL        CDATA  #IMPLIED
    ONAUCTION        CDATA  #IMPLIED
    BUYABLE          CDATA  #IMPLIED
    BASEITEM_ID      CDATA  #IMPLIED
    CLASSIFGRP_ID    CDATA  #IMPLIED
    ITEMSPC_ID       CDATA  #IMPLIED
    STATE            CDATA  "1"
>
```

DTD ジェネレーターが機能する方法は、以下の手順で設定できます。

1. 新しい DTD ジェネレーターのカスタマイザー・プロパティー・ファイルを作成します。
   - NT 2000 Solaris Linux

     DB2ConnectionCustomizer.properties が DTDGenerator.zip アーカイブ内にあります。このファイルを抽出し、拡張子は .properties のままでファイルの名前を変更してから、そのファイルをクラスパス内にあるディレクトリー内

に入れます。**重要:** 既存の DB2ConnectionCustomizer.properties ファイルは除去または変更しないでください。

- 400

  ISeries_GENWCSDTD_Customizer.properties が /QIBM/ProdData/WebCommerce/properties ディレクトリーにあります。このファイルを */instroot*/xml ディレクトリーにコピーし、拡張子は .properties のままで新規ファイルの名前を変更してから、新規ファイルに必要な変更を加えます。**重要:** 元の ISeries_GENWCSDTD_Customizer.properties ファイルは除去または変更しないでください。

2. 新しいファイルのデータベース・ドライバー値を変更します。たとえば、以下のようにします。

```
DBVendorName = DB2
DBDriverName = COM.ibm.db2.jdbc.app.DB2Driver
DBURL = jdbc:db2:
```

   ここで、

   - DBVendorName は、データベースのタイプを選択するために使用されます。

     オプションは、以下のとおりです。

     – DB2 Universal Database for iSeries (DB2/iSeries)
     – その他のオペレーティング・システム用の DB2 (DB2)
     – Oracle データベース (Oracle)
   - DBDriverName は、JDBC ドライバーを選択するために使用されます。

     オプションは、以下のとおりです。

     – DB2 Universal Database for iSeries (com.ibm.db2.jdbc.app.DB2Driver)
     – その他のオペレーティング・システム用の DB2 (COM.ibm.db2.jdbc.app.DB2Driver)
     – Oracle データベース (oracle.jdbc.driver.OracleDriver)
   - DBURL は、データベースにアクセスするための URL を指定するために使用されます。

     オプションは、以下のとおりです。

     – DB2 Universal Database for iSeries (jdbc:db2://)
     – その他のオペレーティング・システム用の DB2 (jdbc:db2:)
     – Oracle データベース (jdbc:oracle:oci8:@)
3. 新しいファイルの名前を DTD Generate コマンドのカスタマイザー・パラメーターの値として指定します。

DTD の生成

TableNames.txt 入力ファイルには、以下のデータベース・テーブル名が各行に 1 つずつ含まれています。

```
MEMBER
ADDRBOOK
ADDRESS
```

以下は、DTD ジェネレーターの起動例です。

- NT 2000 AIX Solaris Linux

```
java com.ibm.wca.DTDGenerator.GenerateDTD -dbname MALL -dbuser db2inst1
-dbpwd db2ibm -outfile wc.dtd -infile TableNames.txt
```

- 400

```
QWEBCOMM/GENWCSDTD DATABASE(DATABASE_NAME) SCHEMA(MALL)
INSTROOT(/QIBM/UserData/WebCommerce/instances/mser)
PASSWD(mypassword) OUTFILE(wc.dtd) INFILE(TableNames.txt)
```

出力ファイル wc.dtd には、以下が含まれます。

```
<!ELEMENT MALL (( MEMBER | ADDRBOOK | ADDRESS)*)>
<!ELEMENT MEMBER EMPTY>
<!ATTLIST MEMBER
   MEMBER_ID            CDATA          #REQUIRED
   TYPE                 CDATA          #REQUIRED
   STATE                CDATA          #IMPLIED
>
<!ELEMENT ADDRBOOK EMPTY>
<!ATTLIST ADDRBOOK
   ADDRBOOK_ID          CDATA          #REQUIRED
   MEMBER_ID            CDATA          #REQUIRED
   TYPE                 CDATA          #IMPLIED
   DISPLAYNAME          CDATA          #REQUIRED
   DESCRIPTION          CDATA          #IMPLIED
>
<!ELEMENT ADDRESS EMPTY>
<!ATTLIST ADDRESS
   ADDRESS_ID           CDATA          #REQUIRED
   ADDRESSTYPE          CDATA          #IMPLIED
   MEMBER_ID            CDATA          #REQUIRED
   ADDRBOOK_ID          CDATA          #REQUIRED
   ORGUNITNAME          CDATA          #IMPLIED
   FIELD3               CDATA          #IMPLIED
   BILLINGCODE          CDATA          #IMPLIED
   BILLINGCODETYPE      CDATA          #IMPLIED
   STATUS               CDATA          #IMPLIED
   ORGNAME              CDATA          #IMPLIED
   ISPRIMARY            CDATA          #IMPLIED
   LASTNAME             CDATA          #IMPLIED
   PERSONTITLE          CDATA          #IMPLIED
   FIRSTNAME            CDATA          #IMPLIED
   MIDDLENAME           CDATA          #IMPLIED
   BUSINESSTITLE        CDATA          #IMPLIED
   PHONE1               CDATA          #IMPLIED
   FAX1                 CDATA          #IMPLIED
   PHONE2               CDATA          #IMPLIED
   ADDRESS1             CDATA          #IMPLIED
   FAX2                 CDATA          #IMPLIED
   NICKNAME             CDATA          #REQUIRED
   ADDRESS2             CDATA          #IMPLIED
   ADDRESS3             CDATA          #IMPLIED
   CITY                 CDATA          #IMPLIED
   STATE                CDATA          #IMPLIED
   COUNTRY              CDATA          #IMPLIED
   ZIPCODE              CDATA          #IMPLIED
   EMAIL1               CDATA          #IMPLIED
   EMAIL2               CDATA          #IMPLIED
   PHONE1TYPE           CDATA          #IMPLIED
   PHONE2TYPE           CDATA          #IMPLIED
   PUBLISHPHONE1        CDATA          #IMPLIED
   PUBLISHPHONE2        CDATA          #IMPLIED
   BESTCALLINGTIME      CDATA          #IMPLIED
   PACKAGESUPPRESSION   CDATA          #IMPLIED
   LASTCREATE           CDATA          #IMPLIED
   OFFICEADDRESS        CDATA          #IMPLIED
   SELFADDRESS          CDATA          "0"
   FIELD1               CDATA          #IMPLIED
   FIELD2               CDATA          #IMPLIED
   TAXGEOCODE           CDATA          #IMPLIED
   SHIPPINGGEOCODE      CDATA          #IMPLIED
>
```

スキーマおよび詳細な XML ファイルの生成

この例では、DTD ジェネレーターは以下のように起動されます。

- NT 2000

```
java com.ibm.wca.DTDGenerator.GenerateDTD -dbname SAMPLE
-dbuser johndoe -dbpwd password -xmlTableDesc c:¥sample¥sample.xml
-outfile tables.dtd -tablenames "employee,staff"
```

- AIX Solaris Linux

```
java com.ibm.wca.DTDGenerator.GenerateDTD -dbname SAMPLE
-dbuser johndoe -dbpwd password -xmlTableDesc usr/sample/sample.xml
-outfile tables.dtd -tablenames "employee,staff"
```

- 400

```
QWEBCOMM/GENWCSDTD DATABASE(MYDB) SCHEMA(SAMPLE)
INSTROOT(/QIBM/UserData/WebCommerce/instances/mser) PASSWD(mypassword)
OUTFILE(tables.dtd) TABNAMES('employee,staff') XMLTABDESC(/sample/sample.xml)
```

スキーマ・ファイルは sample ディレクトリーに作成され、そのファイル名は WCAWebForm.xsd です。 sample.xml 出力ファイルの内容は、以下のとおりです。

```
<?xml version="1.0" encoding="UTF-8"?>
  <formList xmlns="WCAWebForm.xsd" dbname="SAMPLE" dtdname="tables.dtd">
    <form name="EMPLOYEE">
      <uniqueIndex name="U2" columns="FIRSTNME,LASTNAME"/>
      <uniqueIndex name="U3" columns="MIDINIT,LASTNAME"/>
      <field name="EMPNO" type="string" maxlength="6" minOccurs='1'
          uniqueKey="true" showColumnInList="true" />
      <field name="FIRSTNME" type="string" maxlength="32" minOccurs='1'
          showColumnInList="true" />
      <field name="MIDINIT" type="string" maxlength="1" minOccurs='1'
          showColumnInList="true" />
      <field name="LASTNAME" type="string" maxlength="15" minOccurs='1'
          showColumnInList="true" />
      <field name="WORKDEPT" type="string" maxlength="3" showColumnInList="true" />
      <field name="PHONENO" type="string" maxlength="4"/>
      <field name="HIREDATE" type="date" maxlength="10"/>
      <field name="JOB" type="string" maxlength="8"/>
      <field name="EDLEVEL" type="integer" maxlength="5" minOccurs='1' />
      <field name="SEX" type="string" maxlength="1"/>
      <field name="BIRTHDATE" type="date" maxlength="10"/>
      <field name="SALARY" type="decimal" maxlength="9"/>
      <field name="BONUS" type="decimal" maxlength="9"/>
      <field name="COMM" type="decimal" maxlength="9"/>
    </form>
    <form name="STAFF">
      <field name="ID" type="integer" maxlength="5" minOccurs='1'
          uniqueKey="true" showColumnInList="true" />
      <field name="NAME" type="string" maxlength="9" showColumnInList="true" />
      <field name="DEPT" type="integer" maxlength="5" showColumnInList="true" />
      <field name="JOB" type="string" maxlength="5" showColumnInList="true" />
      <field name="YEARS" type="integer" maxlength="5" showColumnInList="true" />
      <field name="SALARY" type="decimal" maxlength="7"/>
      <field name="COMM" type="decimal" maxlength="7"/>
    </form>
  </formList>
```

第 5 章 ID の解決

ターゲット・データベースにロードされる XML データには、 ID を必要とする XML エレメント用の ID が含まれていなければなりません。 XML 文書内のカタログ・エンティティーの ID を生成または検索するには、 ID Resolve コマンドを呼び出します。

ID リゾルバーは、基本テーブルの ID のみ解決します。基本テーブルとは、KEYS または SUBKEYS テーブル内にリストされているテーブルです。 KEYS または SUBKEYS 内にないテーブルの ID を解決する必要がある場合は、 ID リゾルバーを実行する前に、そのテーブルを SUBKEYS テーブル内に追加します。

ID リゾルバーは、たとえば以下のような場合に使用できます。
- データに ID が必要なときに、新しい内容を XML フォーマットでロードする
- データベース内のオブジェクトにすでに ID がある場合に、内容を更新する

ID リゾルバーは実際の ID を提供します。そうでなければ、以下の技法を用いて ID を解決できます。
- 内部別名解決
- プロパティー・ファイル指定
- 固有索引解決

ID リゾルバーのセットアップ

ID リゾルバーがタイム・スタンプ、ストレージ、およびデータベース・ドライバーを処理する方法は、以下の手順で設定できます。

1. 新しい ID リゾルバーのカスタマイザー・プロパティー・ファイルを作成します。
   - NT 2000 AIX Solaris Linux

     `DB2ConnectionCustomizer.properties` が `IdResGen.zip` アーカイブ内にあります。このファイルを抽出し、拡張子は `.properties` のままでファイルの名前を変更してから、そのファイルをクラスパス内にあるディレクトリー内に入れます。**重要:** 既存の `DB2ConnectionCustomizer.properties` ファイルは除去または変更しないでください。
   - 400

     `ISeries_RESWCSID_Customizer.properties` が `/QIBM/ProdData/WebCommerce/properties` ディレクトリーにあります。このファイルを `/instroot/xml` ディレクトリーにコピーし、拡張子は `.properties` のままで新規ファイルの名前を変更してから、新規ファイルに必要な変更を加えます。**重要:** 元の `ISeries_RESWCSID_Customizer.properties` ファイルは除去または変更しないでください。
2. 新しいファイルで指定されているプロパティーの値を変更します。
3. 新しいファイルの名前を ID Resolve コマンドのカスタマイザー・パラメーターの値として指定します。

ID リゾルバーによるタイム・スタンプの処理方法の設定

ID リゾルバーのカスタマイザー・プロパティー・ファイルには、以下のデフォルトの入力タイム・スタンプ・マスクが含まれています。

```
InputTimeStampFormat.1 = yyyy-DD hh:mm:ss.SSSSSS
InputTimeStampFormat.2 = yyyy-MM-dd hh:mm:ss.SSSSSS
InputTimeStampFormat.3 = yyyy-DD-hh.mm.ss.SSSSSS
InputTimeStampFormat.4 = yyyy-MM-dd-HH.mm.ss.SSSSSS
InputTimeStampFormat.5 = yyyy-MM-dd-hh.mm.ss.SSSSSS
InputTimeStampFormat.6 = yyyy-MM-dd HH:mm:ss.SSSSSS
InputTimeStampFormat.7 = yyyy-DD HH:mm:ss.SSSSSS
```

これらのタイム・スタンプ・マスクを変更したり、使用している ID リゾルバーのカスタマイザー・プロパティー・ファイルに必要な数のマスクを追加したりすることができます。入力タイム・スタンプを追加する場合は、現行シーケンスの次の番号を使用する必要があります。 (たとえば、上記のリストに追加する場合、次の入力タイム・スタンプ・マスクは `InputTimeStampFormat.8` になります。)

さらに、ID リゾルバーのカスタマイザー・プロパティー・ファイル内の以下のプロパティーの値を変更することによって、出力タイム・スタンプのフォーマット、マイクロ秒マスク、およびデータベース固有フォーマットがカスタマイズできます。

```
TargetTimeStampFormat = yyyy-MM-dd HH:mm:ss.SSSSSS
MicroSecondMask = SSSSSS
DatabaseSpecificFormat = YYYY-MM-DD HH24:MI:SS
```

ID リゾルバーによるストレージの処理方法の設定

以下は、永続ハッシュマップに関連したプロパティーにデフォルト値を指定している、 ID リゾルバーのカスタマイザー・プロパティー・ファイル内のセクションです。

```
////////////////////////////////////////////////////////////////////////
/// 0 = Normal hashmap with no backend storage
/// 1 = JDBM
////////////////////////////////////////////////////////////////////////

PersistentStorageType = 0

////////////////////////////////////////////////////////////////////////
/// If PersistentStorageType != 0, set MemoryStorageSize to the maximum size
/// of the hashmap in memory data and after that the hashmap will stream
/// the data to a persistent storage as specified
/// If -1, then it uses the normal hashmap with no backend storage
////////////////////////////////////////////////////////////////////////

MemoryStorageSize = 1
```

ID リゾルバーが永続ストレージを処理する方法は、 ID リゾルバーのカスタマイザー・プロパティー・ファイル内の `PersistentStorageType` の値を設定することによって指定できます。

- `PersistentStorageType = 0` を設定すると、 ID リゾルバーは、「通常の」方法 (この場合はメモリー内にシンボル・ハッシュマップが存在する) で操作します。
- `PersistentStorageType = 1` を設定すると、シンボルと鍵の保持に JDBM が使用されます。

メモリーに格納されるレコードの数は、 ID リゾルバーのカスタマイザー・プロパティー・ファイル内の MemoryStorageSize の値を設定することによって指定できます。

- MemoryStorageSize の値が「1」の場合は、メモリーにレコードが 1 つだけ保持されることを示します。
- MemoryStorageSize の値が「-1」の場合には特殊な意味があり、メモリーにすべてのレコードが保持されることを示します。

この場合、ID リゾルバーはその「通常の」動作に復帰します。

ID リゾルバーによるデータベース・ドライバーの処理方法の設定

ID リゾルバーのカスタマイザー・プロパティー・ファイル内の以下の行では、データベース・ドライバー用のデフォルト値を指定しています。

```
DBVendorName = DB2
DBDriverName = COM.ibm.db2.jdbc.app.DB2Driver
DBURL = jdbc:db2:
```

ここで、

- DBVendorName は、データベースのタイプを選択するために使用されます。
  オプションは、以下のとおりです。
  - DB2 Universal Database for iSeries (DB2/iSeries)
  - その他のオペレーティング・システム用の DB2 (DB2)
  - Oracle データベース (oracle)
- DBDriverName は、JDBC ドライバーを選択するために使用されます。
  オプションは、以下のとおりです。
  - DB2 Universal Database for iSeries (com.ibm.db2.jdbc.app.DB2Driver)
  - その他のオペレーティング・システム用の DB2 (COM.ibm.db2.jdbc.app.DB2Driver)
  - Oracle データベース (oracle.jdbc.driver.OracleDriver)
- DBURL は、データベースにアクセスするための URL を指定するために使用されます。
  オプションは、以下のとおりです。
  - DB2 Universal Database for iSeries (jdbc:db2://)
  - その他のオペレーティング・システム用の DB2 (jdbc:db2:)
  - Oracle データベース (jdbc:oracle:oci8:@)

データ処理方法の決定

ID Resolve コマンドにより、load、update、mixed のいずれかのメソッドを選択して入力ファイルを処理できます。

- ファイル内の***すべての*** レコードがデータベース内に***存在しない*** 場合は、 load メソッドを用いて入力ファイルを処理します。
- ファイル内の***すべての*** レコードがデータベース内に***存在する*** 場合は、 update メソッドを用いて入力ファイルを処理します。

- ファイル内のレコードの***一部だけ*** がデータベース内に***存在する*** 場合は、 mixed メソッドを用いて入力ファイルを処理します。

load メソッドの選択

ID リゾルバーの load メソッドは、データベースにロードされるレコード用に新しい ID を生成するために使用されます。このメソッドでは、レコードに対して新しい ID が作成されます。以下の例は、新しいデータ用に ID を生成するために使用します。

- NT 2000

```
idresgen -dbname db -dbuser user -dbpwd pwd -infile input.xml
-outfile output.xml -method load -customizer customizer -schemaname mall
```

load メソッドはデフォルトです。

- AIX Solaris Linux

```
./idresgen.sh -dbname db -dbuser user -dbpwd pwd -infile input.xml
-outfile output.xml -method load -customizer customizer -schemaname mall
```

load メソッドはデフォルトです。

- 400

```
QWEBCOMM/RESWCSID DATABASE(DATABASE_NAME) SCHEMA(MALL)
INSTROOT(/QIBM/UserData/WebCommerce/instances/mser)
PASSWD(mypassword) INFILE(input.xml) OUTFILE(output.xml)
METHOD(*LOAD)
```

update メソッドの選択

ID リゾルバーの update メソッドを指定する場合は、入力ファイル内のレコードがデータベース内にすでに存在していなければなりません。 ID リゾルバーはデータベース内の ID を見付けます。データベース内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 を解決できず、エラーが発生したことを示します。以下の例は、データベース内にすでに存在しているデータの ID を見付ける場合に使用します。

- NT 2000

```
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)
```

mixed メソッドの選択

入力データ・ファイル内に、データベース内にすでに存在するレコードに加えていくつかの新規のレコードが含まれている場合は、 mixed メソッドを用いて ID リゾルバーを実行する必要があります。このメソッドでは、ID リゾルバーは、レコード

がデータベース内に存在していない場合にだけ、そのレコード用に新しい ID を生成します。その他の場合は、データベースから既存の ID を取得します。以下の例は、新規データ用の ID を生成し、データベース内にすでに存在しているデータ用の ID を見付ける場合に使用します。

- NT 2000

```
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)
```

ID 解決技法の使用

ID リゾルバーは、Java プロパティー・ファイルを使用して、基本テーブルの ID が必要なテーブルの索引として、基本テーブルのどの列を使用すべきかを判別することができます。テーブルが KEYS または SUBKEYS テーブル内にリストされている場合、そのテーブルは基本テーブルです。

ID リゾルバーの内部別名解決を使用するには、 XML ファイル内の 1 次鍵属性 (ID) に別名を入れます。こうすると、XML ファイル全体でそのエレメントの参照に別名を使用できるようになります。この処理により、プログラムが XML ファイルの構築に必要な固有索引を判別する必要はなくなります。

また、ID リゾルバーはデータベース・スキーマを分析して、その要件を満たす固有索引があるかどうかを判別することができます。 ID リゾルバーは、分析されているテーブルのプロパティー・ファイル内にエントリーがない場合か、プロパティー・ファイルがない場合に限り固有索引を探します。これらの条件が当てはまる場合は、固有索引検査が実行されます。固有索引が存在し、テーブルの 1 次鍵を含んでいない場合、その固有索引は有効と見なされます。

ID リゾルバーを使用したプロパティー・ファイルの指定

ID リゾルバーを用いることにより、代替の Java プロパティー・ファイルを使用して、基本行の ID が必要なテーブルの索引として基本エントリーのどの列を使用すべきかを記述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、必要に応じて、ID Resolve コマンドを呼び出すときに、これを変更したり独自のファイルを指定したりすることができます。

- NT 2000 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を現行ディレクトリーに置かない場合は、クラスパス環境変数で定義されたディレクトリーに置くことができます。また、このファイルへの完全パスを指定することもできます。

- 400

  IdResolveKeys.properties を変更するには、これを /QIBM/ProdData/WebCommerce/properties ディレクトリーからコピーして /*instroot*/xml ディレクトリーに保管し、その後でこの新しいファイルに必要な変更を加えます。

  注: 上記のディレクトリーは、RESWCSID コマンドによって使用されるクラスパス内にあります。

プロパティー・ファイルを使用した ID の生成

以下の例では、ADDRBOOK および ADDRESS レコードの ID である ADDRBOOK_ID および ADDRESS_ID をそれぞれ解決する必要があります。MEMBER レコードの ID はすでに分かっています。各レコードには、WebSphere Commerce データベース用の有効な ID が必要です。さらに、ADDRESS レコード内の ADDRBOOK_ID は、その外部鍵制約を満たすために、基本テーブルの ID を必要とします。

```
<MEMBER
  MEMBER_ID="100"
  TYPE="U"
  STATE="1"
/>
<MEMBER
  MEMBER_ID="101"
  TYPE="U"
  STATE="1"
/>
<ADDRBOOK
  MEMBER_ID="100"
  DISPLAYNAME="Friends"                 Actual value of the DISPLAYNAME column
  DESCRIPTION="All my friends"
  TYPE="P"
/>
<ADDRESS
  ADDRBOOK_ID="@Friends"                Refers to the ADDRBOOK using the DISPLAYNAME
                                        value as a lookup
  MEMBER_ID="101"
  NICKNAME="Bob"
  ADDRESS1="15 Brave Developers St."
  CITY="Toronto"
  ZIPCODE="A0A0A0"
  COUNTRY="Canada"
  STATUS="P"
/>
```

外部鍵列の ID の解決時に関係行によって基本行内のどの列が使用されるのかを識別するプロパティー・ファイルが必要です。以下は、上記のファイルの構文解析が正しく行われるようにするための手順です。

IdResolveKeys.properties で、以下を指定します。

```
NAMEDELIMITER=@
SELECTDELIMITER=:

ADDRBOOK=@DISPLAYNAME:DISPLAYNAME
ADDRESS=@NICKNAME:NICKNAME
```

NAMEDELIMITER および SELECTDELIMITER はプロパティー・ファイル全体で使用される区切り文字を設定しており、これらが一貫して使用されます。

ADDRBOOK=@DISPLAYNAME:DISPLAYNAME は、住所録レコードが受け取られるときに、住所録行の ID が作成されることを示しています。 DISPLAYNAME フィールドは、入力レコードから抽出され、新しい ID との関係を形成するために使用されます。 DISPLAYNAME ストリングは、住所録行の DISPLAYNAME を突き合わせて、外部鍵に必要な ID を解決するために使用されます。

DISPLAYNAME が Friends である前述の入力例について、このレコードに作成される ID が 12951 であるとします。 DISPLAYNAME は 12951 のキー索引として使用されます。処理は、ADDRBOOK_ID が「@...」の形式 (これは、区切り文字の後に続くものが住所録 ID の検索に使用されることを示す) である次のレコード、ADDRESS に移ります。このストリングは DISPLAYNAME と一致し、 12951 が戻されて、ADDRBOOK_ID 属性内に入れられます。

```
<MEMBER
  MEMBER_ID="100"
  TYPE="U"
  STATE="1"
/>
<MEMBER
  MEMBER_ID="101"
  TYPE="U"
  STATE="1"
/>
<ADDRBOOK
  ADDRBOOK_ID="12951"                  Generated primary key
  MEMBER_ID="100"
  DISPLAYNAME="Friends"                Value of ADDRBOOK DISPLAYNAME unchanged
  DESCRIPTION="All my friends"
    TYPE="P"
/>
<ADDRESS
  ADDRESS_ID="13051"                   Generated primary key
  ADDRBOOK_ID="12951"                  ADDRESS refers to correct ADDRBOOK
  MEMBER_ID="101"
  NICKNAME="Bob"
  ADDRESS1="15 Brave Developers St."
  CITY="Toronto"
  ZIPCODE="A0A0A0"
  COUNTRY="Canada"
  STATUS="P"
/>
```

複合キーを含むプロパティー・ファイルの使用

3 つ以上の列からなるキーは複合キーです。複合キーの検索は、プロパティー・ファイル内で、後ろにフィールド名が付いた NAMEDELIMITER と SELECTDELIMITER の両方を指定することによって定義できます。 ADDRBOOK レコードの検索基準を表示名とメンバー ID の複合にするため、たとえば、プロパティー・ファイルに、以下のように指定します。

```
ADDRBOOK=@DISPLAYNAME@MEMBER_ID:DISPLAYNAME MEMBER_ID
```

次に、以下の XML 入力ファイル・フラグメントを指定します。

```
<ADDRBOOK
  MEMBER_ID="100"
  DISPLAYNAME="Friends"                 ADDRBOOK "Friends" of MEMBER 100
  DESCRIPTION="All my friends"
  TYPE="P"
/>
<ADDRBOOK
  MEMBER_ID="101"
  DISPLAYNAME="Friends"                 ADDRBOOK "Friends" of MEMBER 101
  DESCRIPTION="All my friends"
  TYPE="P"
/>
<ADDRESS
  ADDRBOOK_ID="@Friends@100"            Lookup the primary key for ADDRBOOK
                                        "Friends" of MEMBER 100
  MEMBER_ID="101"
  NICKNAME="Bob"
  ADDRESS1="15 Brave Developers St."
  CITY="Toronto"
  ZIPCODE="A0A0A0"
  COUNTRY="Canada"
  STATUS="P"
/>
```

これで、解決の後に以下が生成されます。

```
<MEMBER
  MEMBER_ID="100"
  TYPE="U"
  STATE="1"
/>
<MEMBER
  MEMBER_ID="101"
  TYPE="U"
  STATE="1"
/>
<ADDRBOOK
  ADDRBOOK_ID="12951"                   ADDRBOOK of interest
  MEMBER_ID="100"
  DISPLAYNAME="Friends"
  DESCRIPTION="All my friends"
  TYPE="P"
/>
<ADDRBOOK
  ADDRBOOK_ID="12952"
  MEMBER_ID="101"
  DISPLAYNAME="Friends"
  DESCRIPTION="All my friends"
  TYPE="P"
/>
<ADDRESS
  ADDRESS_ID="13051"
  ADDRBOOK_ID="12951"                   ADDRESS refers to correct ADDRBOOK
  MEMBER_ID="101"
  NICKNAME="Bob"
  ADDRESS1="15 Brave Developers St."
  CITY="Toronto"
  ZIPCODE="A0A0A0"
  COUNTRY="Canada"
  STATUS="P"
/>
```

カスケードされた 1 次鍵を含むプロパティー・ファイルの使用

基本テーブル STOREENT は 1 次鍵 STOREENT_ID を定義します。 STOREENT を参照する外部テーブル STORE は、基本テーブル STOREENT にとっては外部鍵となる 1 次鍵 STORE_ID を定義します。つまり、STORE_ID の値は、STOREENT_ID 値の 1 つでなければなりません。したがって、外部テーブル STORE の 1 次鍵である STORE_ID は、基本および外部の二重の役割を持ちます。

別のテーブル CONTRACT は STORE の外部テーブルで、 CONTRACT の外部鍵である STORE_ID は STORE 内の 1 次鍵 STORE_ID を参照するとします。したがって、STORE テーブルは、CONTRACT テーブルの基本テーブルです。

STORE テーブルの STORE_ID は、作成されるのではなく STOREENT_ID から参照されるので、 ID リゾルバーは STORE テーブルについて内部別名と ID 値の関連付けを行いません。 CONTRACT テーブルが STORE テーブルの STORE_ID を解決しようとすると、これは空の値を受け取ります。

このような特殊な条件があるので、内部別名の作成は、プロパティー・ファイル内にエントリーを作成することによって明示的に指定する必要があります。 `IdResolveKeys.properties` で、以下を指定します。

```
"STORE=@STORE_ID:STORE_ID"
```

これは、以下を実行することを ID リゾルバーに強制します。

- 外部参照として STORE_ID を解決するときに、内部別名と ID 値の関係を作成する
- CONTRACT テーブルの STORE_ID を解決するときに関係を使用する

プロパティー・ファイル内の `STORE=@STORE_ID:STORE_ID` エントリーと、以下の XML 入力ファイル・フラグメントを使用します。

```
<STOREENT
  IDENTIFIER="Out Fashions"
  MEMBER_ID="-2000"
  STOREENT_ID="@storeent_id_1"
  TYPE="G"
/>
<STORE
  STORE_ID="@storeent_id_1"
  STOREGRP_ID="1"
  STORELEVEL="store_level"
/>
<CONTRACT
  CONTRACT_ID="@contract_id_1"
  STATE="0"
  STORE_ID="@storeent_id_1"
/>
```

これで、解決の後に以下が生成されます。

```
<STOREENT
  IDENTIFIER="Out Fashions"
  MEMBER_ID="-2000"
  STOREENT_ID="10501"
  TYPE="G"
/>
```

```
<STORE
  STORE_ID="10501"
  STOREGRP_ID="1"
  STORELEVEL="store_level"
/>
<CONTRACT
  CONTRACT_ID="@contract_id_1"
  STATE="0"
  STORE_ID="10501"
/>
```

内部別名解決の使用

ID リゾルバーの内部別名解決を使用するには、 XML ファイル内の 1 次鍵属性 (ID) に別名を入れます。こうすると、XML ファイル全体でそのエレメントの参照に別名を使用できるようになります。この処理により、プログラムが XML ファイルの構築に必要な固有索引を判別する必要はなくなります。

内部別名はファイル全体で一貫して使用される必要があります。住所録 ID である ADDRBOOK_ID に @addrbook_1 という別名がある場合は、ファイル内のこの ID を参照するすべての外部鍵が @addrbook_1 を使用しなければなりません。別名は一時的なものであることに注意してください。これらは保管されません。別個の XML ファイルで別名を再び紹介せずにこれらを使用することはできません。

内部別名 ID 解決の使用例の一部

解決前:

```
<MEMBER
  MEMBER_ID="100"
  TYPE="U"
  STATE="1"
/>
<ADDRBOOK
  ADDRBOOK_ID="@addrbook_1"            Alias for ADDRBOOK
  MEMBER_ID="100"
  DISPLAYNAME="Friends"
  DESCRIPTION="All my friends"
  TYPE="P"
/>
<ADDRESS
  ADDRESS_ID="@address_1"              Alias for ADDRESS
  ADDRBOOK_ID="@addrbook_1"            Refers to the alias for ADDRBOOK
  MEMBER_ID="101"
  NICKNAME="Bob"
  ADDRESS1="1 Brave Developer St."
  CITY="Toronto"
  ZIPCODE="A3B0F4"
  COUNTRY="Canada"
  STATUS="P"
/>
```

解決後:

```
<MEMBER
  MEMBER_ID="100"
  TYPE="U"
  STATE="1"
/>
<ADDRBOOK
  ADDRBOOK_ID="11801"                Generated primary key
```

```
  MEMBER_ID="100"
  DISPLAYNAME="Friends"
  DESCRIPTION="All my friends"
  TYPE="P"
/>
<ADDRESS
  ADDRESS_ID="11901"                    Generated primary key
  ADDRBOOK_ID="11801"                   Refers to ADDRBOOK entry
  MEMBER_ID="100"
  NICKNAME="Bob"
  ADDRESS1="1 Brave Developer St."
  CITY="Toronto"
  ZIPCODE="A3B0F4"
  COUNTRY="Canada"
  STATUS="P"
/>
```

固有索引解決の使用

ID リゾルバーのデフォルト動作である固有索引解決は、分析されるテーブルのプロパティー・ファイル内にエントリーがない場合や、プロパティー・ファイルがない場合に使用されます。固有索引解決は、ID を見付ける手段として、テーブル上で指定された任意の固有な索引を使用します。たとえば、MEMBER_ID と IDENTIFIER は CATALOG テーブル上の固有索引で、 CATALOGDSC テーブルの 1 次鍵 CATALOG_ID に対する解決ポイントとして使用することができます。

データベースの内容を更新するには、データベース内の基本テーブルの固有キーを知っている必要があります。これは、データベースを参照して見付けることができます。たとえば、固有キーを検索する DB2 コマンドは、次のようになります。

```
db2 describe indexes for table schema.tablename show detail
```

固有索引解決の例の一部

解決前:

```
<MEMBER
  MEMBER_ID="100"
  TYPE="O"
  STATE="1"
/>

<CATALOG
  DESCRIPTION="Winter Catalog"
  IDENTIFIER="WC2001"
  MEMBER_ID="100"
  TPCLEVEL="2"
/>

<CATALOGDSC
  CATALOG_ID="@WC2001@100"              Refers back to catalog "WC2001" of member
                                        "100" (Note: The order is important.)
  FULLIMAGE="c:¥store¥img¥wc.gif"
  LANGUAGE_ID="-1"
  LONGDESCRIPTION="2001 Winter Catalog"
  SHORTDESCRIPTION="2001 Winter Catalog"
  NAME="InFashion 2001 Winter Catalog"
  THUMBNAIL="c:¥store¥img¥wc_th.gif"
/>
```

解決後:

```
<MEMBER
  MEMBER_ID="100"
  TYPE="O"
  STATE="1"
/>
<CATALOG
  CATALOG_ID="10351"                 Automatically generated primary key
  DESCRIPTION="Winter Catalog"
  IDENTIFIER="WC2001"
  MEMBER_ID="100"
  TPCLEVEL="2"
/>
<CATALOGDSC
  CATALOG_ID="10351"                 Refers to the correct catalog
  FULLIMAGE="c:¥store¥img¥wc.gif"
  LANGUAGE_ID="-1"
  LONGDESCRIPTION="2001 Winter Catalog"
  SHORTDESCRIPTION="2001 Winter Catalog"
  NAME="InFashion 2001 Winter Catalog"
  THUMBNAIL="c:¥store¥img¥wc_th.gif"
/>
```

MEMBER テーブルへのデータのロード

ID リゾルバーは、システムによって生成された ID を持つテーブルの解決を処理します。これには、KEYS または SUBKEYS テーブルに登録されているあらゆるテーブルおよび列が含まれます。この解決には、以下の 2 つの構成要素があります。

1. データベース内に基本テーブル (つまり、KEYS または SUBKEYS にリストされているテーブル) が存在するかどうかの判別。

   この解決は、固有索引解決かプロパティー・ファイル指定のいずれかを使用する、そのエレメントの XML データの内容に基づきます。

2. 基本テーブルの外部鍵があるかどうかの判別。

   これは、関連テーブルの外部鍵属性内の解決仕様を用いて行われます。

MEMBER テーブルは、 ORGENTITY、MBRGRP、および USER テーブルの「スーパー・クラス」として使用されます。これは、テーブルが MEMBER テーブルのサブタイプに対する外部鍵制約を持つ場合の参照保全の保守に役立つ「is-a」パターンを作成します。しかし、すべての MEMBER サブタイプは共通基本タイプを共用するので、 ID はサブタイプ間で固有でなければなりません。つまり、ORGENTITY_ID は、MBRGRP_ID と USER_ID のセット内で固有でなければなりません。そのために、KEYS テーブルは ORGENTITY、MBRGRP、および USER テーブルだけを参照し、これらの ID について相互に排他的な範囲を指定します。各サブタイプは 1 次鍵を持ちます。それぞれの 1 次鍵は、MEMBER テーブルの 1 次鍵に対する外部鍵でもあります。

MEMBER とそのサブタイプの間の制約のため、MEMBER とサブタイプが同期 ID を持てない場合があります。 ORGENTITY、MBRGRP、および USER テーブルをシステムにロードするため、 ID リゾルバーは、「is-a」パターンを認識し、これを適切に処理します。以下の ID リゾルバー用の XML 構文があります。

```
<ORGENTITY
    ORGENTITY_ID="@orgAlias"
    ORGENTITYNAME="Test Org"
    ORGENTITYTYPE="O">
    <ISA>
       <MEMBER
           TYPE="O"
           STATE="1"
       />
    </ISA>
</ORGENTITY>
```

これは、以下を生成します。

```
<MEMBER
    MEMBER_ID="12345"
    TYPE="O"
    STATE="1"
/>
<ORGENTITY
    ORGENTITY_ID="12345"          Synchronized with member element
    ORGENTITYNAME="Test Org"
    ORGENTITYTYPE="O"
/>
```

ID リゾルバーはこのようにして <isa> サブエレメントを処理し、同期 ID を作成します。

REFKEYS テーブルを使用した外部関係の作成

REFKEYS テーブルは、データベース内にまだ存在していない、テーブル間の外部関係を表すために作成されます。通常は、データベース・スキーマが、テーブルの列を他のテーブルにリンクする外部鍵宣言を作成することによって外部関係を記述します。データベース・スキーマが定義された外部関係を持たず、 ID が外部鍵として解決される必要がある場合は、以下のようにします。

1. 以下の DDL 例のようにして REFKEYS テーブルを作成します。

```
CREATE TABLE "REFKEYS" (
            "FKTABLE_NAME"    CHAR(18)   NOT NULL ,
            "FKCOLUMN_NAME"   CHAR(18)   NOT NULL ,
            "TABLENAME"       CHAR(18)   NOT NULL
);
```

   ここで、

   FKTABLE_NAME は外部 (または「子」) テーブル名です。
   FKCOLUMN_NAME は外部列名です。
   TABLENAME は基本 (または「親」) テーブル名です。

2. REFKEYS テーブル内に、必要な外部関係を記述するエントリーを作成します。

エラーのトラブルシューティング

ID の解決時にエラーが発生した場合は、以下の表を参照してください。

エラー	使用するメソッド	考えられる原因		考えられる対処法
Unresolved primary key (1 次鍵が未解決です)	All	テーブルが KEYS テーブルでも SUBKEYS テーブルでも指定されて**いない**ため、 ID リゾルバーはそのテーブル内の 1 次鍵 (ID) を解決**しません**。		ID リゾルバーを実行する前に、1 次鍵が解決されるテーブルの名前を SUBKEYS テーブルに追加してください。 加えて、データベース・スキーマに外部鍵の関係が存在していることを確認してください。
	更新	1 次鍵は、データベースを参照することによって解決されます。このデータベース参照は、プロパティー・ファイルのエントリーの情報か、特定のテーブルの固有索引を使用して行われます。優先順位はプロパティー・ファイルのエントリーの方が上位です。		入力ファイル内の固有索引情報に誤りがないことを確認してください。 1 次鍵を解決するために適切なデータベース参照が生成されるように、プロパティー・ファイルで適当なエントリーを作成または変更することもできます。
Unresolved foreign key (外部鍵が未解決です)	All	外部参照先のテーブルが KEYS テーブルでも SUBKEYS テーブルでも指定されて**いない**ため、 ID リゾルバーはそのテーブル内の外部鍵を解決**しません**。		ID リゾルバーを実行する前に、参照先のテーブルの名前を SUBKEYS テーブルに追加してください。 加えて、データベース・スキーマに外部鍵の関係が存在していることを確認してください。
		外部鍵は、内部別名を使用するか、データベースを参照することによって解決されます。データベース参照は、内部別名が外部鍵を解決できないときにのみ行われます。	内部別名は、1 次鍵を使用することによって、またプロパティー・ファイルのエントリーを使用することによって生成されます。	ソースおよびターゲットとして使用される入力ファイル内の内部別名の値に誤りがないことを確認してください。
			このデータベース参照は、プロパティー・ファイルのエントリーの情報か、特定のテーブルの固有索引を使用して行われます。優先順位はプロパティー・ファイルのエントリーの方が上位です。	入力ファイル内の固有索引情報に誤りがないことを確認してください。 1 次鍵を解決するために適切なデータベース参照が生成されるように、プロパティー・ファイルで適当なエントリーを作成または変更することもできます。

第 6 章 データのロード

データをロードする場合は、事前に以下を行う必要があります。

1. ローダーと共に使用する DTD とスキーマを作成します (データを初めてロードする場合)。

   注: ストア・アーカイブ用のデータをロードしており、ストア・アーカイブに提供されている DTD を使用して XML ファイルを作成済みである場合は、このステップは不要です。

2. ID を解決します (必要な場合)。

データは関連した DTD を持つ XML フォーマットでなければなりません。データをロードするには、Load コマンドを呼び出します。

ローダーのセットアップ

ローダー・パッケージを使用すると、以下を行うことによってローダーが機能する方法をセットアップできます。

- 入力ファイル内のエレメントを無視する
- 列に NULL を挿入する
- タイム・スタンプと日付データをロードする
- 現行タイム・スタンプをロードする
- イベント・キューを管理する
- 別のデータベース・ソフトウェアとオペレーティング・システムを使用して実行する
- コンポーネントを置換する
- 商品アドバイザー検索スペースの同期を使用する

これらのローダーの機能は、以下の手順でカスタマイズできます。

1. 新しいローダーのカスタマイザー・プロパティー・ファイルを作成します。
   - NT 2000 AIX Solaris Linux

     `MassLoadCustomizer.properties` が `MassLoader.zip` アーカイブにあります。このファイルを抽出し、拡張子は `.properties` のままでファイルの名前を変更してから、そのファイルをクラスパス内にあるディレクトリー内に入れます。**重要:** 既存の `MassLoadCustomizer.properties` ファイルは除去または変更しないでください。

   - 400

     `ISeries_LODWCSDTA_Customizer.properties` が `/QIBM/ProdData/WebCommerce/properties` ディレクトリーにあります。このファイルを `/instroot/xml` ディレクトリーにコピーし、拡張子は `.properties`

のままで新規ファイルの名前を変更してから、新規ファイルに必要な変更を加えます。**重要:** 元の ISeries_LODWCSDTA_Customizer.properties ファイルは除去または変更しないでください。

2. 新しいローダーのカスタマイザー・プロパティー・ファイルで指定されているプロパティーの値を変更します。
3. 新しいファイルの名前を Load コマンドのカスタマイザー・パラメーターの値として指定します。

入力ファイル内のエレメントを無視する

入力ファイルにターゲット・データベースにマップしないエレメントが含まれている場合は、ローダーがローダーのカスタマイザー・プロパティー・ファイルにあるそれらのエレメントを無視するように設定することができます。 IgnoreElements を使用して無視するエレメントを指定し、それらのエレメントをセミコロン (;) で区切ります。たとえば、import、literals、および ProductRepository エレメントを無視するには、ローダーのカスタマイザー・プロパティー・ファイルに以下を指定します。

```
IgnoreElements = import;literals;ProductRepository
```

列に NULL を挿入する

ローダーが列に NULL を挿入できるようにするには、ローダーのカスタマイザー・プロパティー・ファイル内の EnableNULLCheck プロパティーを「true」に設定します。たとえば、以下のようにします。

```
EnableNULLCheck = true
```

パフォーマンス上の理由で、この機能はデフォルトでは使用不可になっています。

NULLStringLiteral プロパティーを使用して、データ内のヌル値のストリング表記を決定します。たとえば、ヌル値の表記にストリング「-」が使用されるようにローダーを設定するには、ローダーのカスタマイザー・プロパティー・ファイルに以下のプロパティーと値を指定します。

```
NULLStringLiteral = -
```

デフォルトで、このプロパティーの値は「NULL」(括弧は付かない) です。

タイム・スタンプと日付データをロードする

ローダーは、タイム・スタンプおよび日付データ・タイプと共にデータを列へロードすることができます。文書内のタイム・スタンプと日付データのデータ・フォーマットは、カスタマイズ可能なパターンによって決まります。ユーザーは、既存のパターンを編集するか、既存のパターンのリストにさらにパターンを追加することができます。

タイム・スタンプと日付のデータは、使用可能なパターン (マスク) と突き合わせて検査されます。そのデータと一致する最初のパターンが、そのデータをデータベースにロードする前にターゲットのタイム・スタンプ・フォーマットに変換するために使用されます。

カスタマイズ可能な出力タイム・スタンプ・パターンには、 TimeStampFormat.JDBC および TimeStampFormat.Load の 2 つがあります。

1. TimeStampFormat.JDBC は、ローダーが JDBC 接続を使用して操作を実行する場合に使用されます。

   ローダーの SQL import および delete メソッドは、データベースの更新に JDBC 接続を使用します。

2. TimeStampFormat.Load はローダーがネイティブ・ユーティリティーを使用する場合に使用されます。

   ローダーの import および load メソッドはネイティブ・ユーティリティーを使用します。

タイム・スタンプのフォーマットは、ローダーのカスタマイザー・プロパティー・ファイル内のマスクを変更または追加することによってカスタマイズできます。

以下に、入力タイム・スタンプ・マスクが提供されています。

```
InputTimeStampFormat.1 = yyyy-DD hh:mm:ss.SSSSSS
InputTimeStampFormat.2 = yyyy-MM-dd hh:mm:ss.SSSSSS
InputTimeStampFormat.3 = yyyy-DD-hh.mm.ss.SSSSSS
InputTimeStampFormat.4 = yyyy-MM-dd-HH.mm.ss.SSSSSS
InputTimeStampFormat.5 = yyyy-MM-dd-hh.mm.ss.SSSSSS
InputTimeStampFormat.6 = yyyy-MM-dd HH:mm:ss.SSSSSS
InputTimeStampFormat.7 = yyyy-DD HH:mm:ss.SSSSSS
```

入力日付フォーマットのデフォルト・パターンは以下のとおりです。

```
InputDateFormat.1 = MM-dd-yyyy
InputDateFormat.2 = yyyy-dd-MM
InputDateFormat.3 = yyyy-MM-dd
InputDateFormat.4 = MM/dd/yyyy
InputDateFormat.5 = yyyy/dd/MM
InputDateFormat.6 = yyyy-DD
```

これらのタイム・スタンプおよび日付マスクを変更するか、必要な数だけマスクを追加することができます。これらのマスクは、ローダーのカスタマイザー・プロパティー・ファイル内に、入力タイプ・スタンプと比較する数値順で指定してください。入力タイム・スタンプを追加する場合は、現行シーケンスの次の番号を使用する必要があります。 (たとえば、上記のリストに追加する場合、次の入力タイム・スタンプ・マスクは `InputTimeStampFormat.8` になります。)

タイム・スタンプおよび日付の出力に対する入力データのフォーマット・パターンは、以下のようになります。

```
TimeStampFormat.JDBC = yyyy-MM-dd hh:mm:ss.SSSSSS
TimeStampFormat.Load = yyyy-MM-dd-hh.mm.ss.SSSSSS

DateFormat.JDBC = yyyy-MM-dd
DateFormat.Load = yyyy-MM-dd
```

通常、出力日付およびタイム・スタンプのフォーマットは、カスタマイズ***されません***。

現行タイム・スタンプをロードする

ローダーは、時刻機構の解釈に基づくタイム・スタンプ・データ・タイプと共に、値を列に挿入することができます。たとえば、WebSphere Commerce のオファーの STARTDATE および ENDDATE には、そのオファーがテーブルに挿入された時刻に基づく値が含まれていることがあります。この機能をサポートするために、ローダー・パッケージは、MLTIME テーブルを使用してタイム・スタンプ・インスタンスを保持します。このテーブルのスキーマは、以下のとおりです。

```
table MLTIME
  (
  INSTANCEID BIGINT not null,
  MLTIMESTAMP TIMESTAMP
  )
```

テーブルとその列の名前は、ローダーのカスタマイザー・プロパティー・ファイル内の以下のプロパティーを変更することによってカスタマイズできます。

```
TimestampTableName = MLTIME
TimestampIdColumn = INSTANCEID
TimestampValueColumn = MLTIMESTAMP
```

現行タイム・スタンプ値を指定するための入力データは、タイム・スタンプのストリング・パターンに基づきます。以下のマスクは、タイム・スタンプの存続時間を指定するために使用されます。

```
%D for days
%M for months
%Y for years
%H for hours
%m for minutes
%s for seconds
```

現行タイム・スタンプのフォーマットは、ローダーのカスタマイザー・プロパティー・ファイル内のマスクを変更または追加することによってカスタマイズできます。以下の入力マスクが提供されています。

```
InputCurrentTimestampFormat.1 = CURRENT TIMESTAMP
InputCurrentTimestampFormat.2 = CURRENT TIMESTAMP %D DAYS
InputCurrentTimestampFormat.3 = CURRENT TIMESTAMP %D DAYS %M MONTHS
InputCurrentTimestampFormat.4 = CURRENT TIMESTAMP %D DAYS %M MONTHS %Y YEARS
InputCurrentTimestampFormat.5 = CURRENT TIMESTAMP %Y YEARS %M MONTHS %D DAYS
InputCurrentTimestampFormat.6 = SYSDATE
InputCurrentTimestampFormat.7 = ADDDAYS(SYSDATE,%D)
InputCurrentTimestampFormat.8 = ADDDAYS(ADDMONTHS(SYSDATE,%M),%D)
InputCurrentTimestampFormat.9 = ADDDAYS(ADDMONTHS(ADDYEARS(SYSDATE,%Y),%M),%D)
```

現行タイム・スタンプの入力データは、指定されたパターンと突き合わせられます。データが指定された入力パターンと一致する場合、そのパターンがその入力データの構文解析に使用され、ローダーはそのデータをデータベースに挿入する前に適切な出力フォーマットに変換します。サブスクリプト番号が順番に配列されていれば、上記のリストに新しいパターンを追加することができます。

現行タイム・スタンプを指定するためのターゲット出力フォーマットには、以下の 2 つがあります。

1. CurrentTimestampFormat.Load は、ローダーがロードまたはインポート・モードで操作している場合に使用されます。

2. CurrentTimestampFormat.JDBC は、ローダーが JDBC を使用してデータベース内の値の挿入、更新、または削除を行っている場合に使用されます。

ローダーのデフォルトのターゲット・パターンは、以下のとおりです。

```
CurrentTimestampFormat.Load = CURRENT TIMESTAMP %Y YEARS %M MONTHS %D DAYS
  %h HOURS %m MINUTES %s SECONDS
CurrentTimestampFormat.JDBC = CURRENT TIMESTAMP %Y YEARS %M MONTHS %D DAYS
  %h HOURS %m MINUTES %s SECONDS
```

これらのパターンのプロパティーも、ローダーのカスタマイザー・プロパティー・ファイルでカスタマイズできます。 CurrentTimestampFormat.Load および CurrentTimestampFormat.JDBC をカスタマイズする場合、結果のステートメントの構文が、指定されているデータベース管理システムに対して有効なものになるようにする必要があります。

CurrentTimestampLiteral プロパティーは、タイム・スタンプ列の値が現行タイム・スタンプのフォーマットであるかどうかの早期判別を行うために使用されます。こうして、値がタイム・スタンプのストリング表記でないことを判別するための高価な計算を避けることができます。

```
CurrentTimestampLiteral = CURRENT TIMESTAMP
```

DB2 DB2 の場合のこのプロパティーのデフォルト値は CURRENT TIMESTAMP です。

Oracle Oracle データベースの場合のデフォルト値は SYSDATE です。

現行タイム・スタンプのロード例

ローダーには、 10123 という OFFER_ID を持つオファーを更新するために、以下の情報が与えられています。開始日には "CURRENT TIMESTAMP" という値が入り、終了日には "CURRENT TIMESTAMP + 14 DAYS." という値が入っています。

```
<OFFER
  OFFER_ID="10123"
  STARTDATE="CURRENT TIMESTAMP">
  ENDDATE="CURRENT TIMESTAMP + 14 DAYS"
/>
```

ローダーは、列 STARTDATE および ENDDATE が、データベース内のタイム・スタンプ・データ・タイプであることを認識します。 CurrentTimeStampLiteral プロパティーに基づいて、この値が現行タイム・スタンプ・フォーマットで指定された値を持つことが判別されます。 STARTDATE の値は InputCurrentTimeStampFormat.1 のパターンと一致し、これは CurrentTimeStampFormat.JDBC プロパティーで指定されたパターンに変換されます。 ENDDATE の値は InputCurrentTimeStampFormat.2 プロパティーのフォーマットと一致し、これも CurrentTimeStampFormat.JDBC プロパティーで指定されたパターンに変換されます。

現行タイム・スタンプへの存続時間の追加例

ローダーには、現行タイム・スタンプに存続時間を追加する機能があります。たとえば、ユーザーは、特定の日付を入力せずにオファーをロードすることができま

す。これを行うには、開始日からいくらかの期間が経過した終了日を作成する必要があります。以下の例は、DB2 で効果的に機能します。

```
<Offer
  Startdate="Current Timestamp"
  Enddate="Current Timestamp +14 Days +4 Months +1 Year +0 Hours
    +0 Minutes +0 Seconds"
/>
```

ただし、プラットフォームに依存しない方法で現行タイム・スタンプの存続期間を処理するには、ローダーのカスタマイザー・プロパティー・ファイル内のマスクを変更することによって、現行タイム・スタンプのフォーマットをカスタマイズする必要があります。以下は、カスタマイズされた現行タイム・スタンプのプロパティー仕様の例です。

```
CurrentTimestampLiteral=Current Timestamp

InputCurrentTimestampFormat.0=Current Timestamp
InputCurrentTimestampFormat.1=Current Timestamp %D Days
InputCurrentTimestampFormat.2=Current Timestamp %M Months
InputCurrentTimestampFormat.3=Current Timestamp %Y Years
InputCurrentTimestampFormat.4=Current Timestamp %D Days %M Months
InputCurrentTimestampFormat.5=Current Timestamp %D Days %M Months %Y Years
InputCurrentTimestampFormat.5=Current Timestamp %H Hours %m Minutes %s Seconds

CurrentTimestampFormat.JDBC=Current Timestamp %D Days %M Months %Y Years
  %H Hours %m Minutes %s Seconds
```

オファー例およびこれらのプロパティー仕様を使用する場合、オファーの終了日は InputCurrentTimestampFormat.5 のパターンと一致します。これにより、CurrentTimestampFormat.JDBC を使用して以下のオファー情報が生成されます。

```
<Offer
  Startdate="Current Timestamp"
  Enddate="Current Timestamp +14 Days +4 Months +1 Year +0 Hours +0 Minutes +0 Seconds"
/>
```

上記の例は、ローダーが複数の現行タイム・スタンプ・フォーマットを入力し、それを必要な出力フォーマットに適切にフォーマットする方法を示しています。以下の例は、プラットフォームに依存しないフォーマットを処理し、それらをプラットフォーム固有の出力フォーマットにマップする方法を示しています。

```
<Offer
  Startdate="Now"
  Enddate="Now +14D +4M +1Y"
/>

CurrentTimestampLiteral=Now

InputCurrentTimestampFormat.0=Now
InputCurrentTimestampFormat.1=Now %DD
InputCurrentTimestampFormat.2=Now %MM
InputCurrentTimestampFormat.3=Now %YY
InputCurrentTimestampFormat.4=Now %DD %MM
InputCurrentTimestampFormat.5=Now %DD %MM %YY
InputCurrentTimestampFormat.5=Sysdate %HH %mm %ss

CurrentTimestampFormat.JDBC=AddYears(AddMonths(AddDays(AddHours(AddMinutes(AddSeconds
  (Sysdate,%s),%m),%H),%D),%M),%Y)
```

注: 上記のステートメントは単なる例に過ぎません。これは、ただ仮説上のデータベース管理システムにおけるカスタマイズ・フィーチャーを表すために使用されています。 DB2 や Oracle データベースに有効なものではありません。

オファー例およびこれらのプロパティー仕様を使用する場合、オファーの終了日は InputCurrentTimestampFormat.5 のパターンと一致します。これにより、CurrentTimestampFormat.JDBC を使用して以下のオファー情報が生成されます。

```
<Offer
  Startdate="Current Timestamp"
  Enddate="AddYears(AddMonths(AddDays(AddMinutes(AddSeconds(Sysdate,0),0),0),14),4),1)"
/>
```

イベント・キューを管理する

ローダーのイベント・キューは、ローダーのカスタマイザー・プロパティー・ファイル内の設定を変更することによってカスタマイズできます。たとえば、以下のようにします。

```
QueueLowCount = 35
QueueHighCount = 90
```

キューを埋めるイベントのソースは、キューの中のエレメントの数が上限に達した時点でブロックされ、それ以上のイベントのキューイングを行えなくなります。キューの中のエレメントの数が下限より下に落ち込むと、キューは再びイベントの受け入れを開始します。

別のデータベース・ソフトウェアとオペレーティング・システムを使用して実行する

別のデータベース・ソフトウェアとオペレーティング・システムで実行するようにローダーをカスタマイズするには、ローダーのカスタマイザー・プロパティー・ファイル内の以下のエレメントのパラメーターを変更して、別のデータベース・ソフトウェアとオペレーティング・システムを指定します。

- データベース接続コマンド
- データベース・ロード・テーブル・コマンド
- データベース・インポート・コマンド
- ロードを開始するシステム・コマンド

これらのアイテムのいずれかをカスタマイズするには、ローダーのカスタマイザー・プロパティー・ファイル内のそのコマンドの前に付いているダブルスラッシュ・コマンド文字 (//) を除去して、デフォルトを変更します。

データベース接続コマンド: デフォルト (DB2 を使用していることを想定) を変更する場合は、データベース接続コマンドのパラメーターを変更します。

```
DBConnectCommand = connect to {0} user {1} using {2};
```

ここで、

0 = データベース名
1 = データベース・ユーザー
2 = ユーザー・パスワード

データベース・ロード・テーブル・コマンド: デフォルトを変更する場合は、データベース・ロード・テーブル・コマンドのパラメーターを変更します。

```
DBLoadTableCommand = load from {0} of del modified by coldel{1}
chardel{2} insert into {3} ({4});
```

ここで、

0 = ファイル名
1 = 列区切り文字
2 = 文字区切り文字
3 = テーブル名
4 = コンマ (,) で区切られた列名

データベース・インポート・コマンド: デフォルトを変更する場合は、データベース・インポート・コマンドのパラメーターを変更します。

```
DBImportCommand = import from {0} of del modified by coldel{1} chardel{2}
insert_update into {3} ({4});
```

ここで、

0 = ファイル名
1 = 列区切り文字
2 = 文字区切り文字
3 = テーブル名
4 = コンマ (,) で区切られた列名

ロードを開始するシステム・コマンド: デフォルトを変更するには、ロードを開始するシステム・コマンドのパラメーターを変更します。このコマンドはネイティブ・ロードを実行し、ローダーによって生成されたスクリプトをインポートします。

- DB2

  ```
  DBLoadCommand = db2clpex DB2 -z {0} -astvf {1}
  ```

  ここで、

  0 = ログ・ファイル名
  1 = コマンド・ファイル名

  たとえば、AIX 上で実行する DB2 の場合、 DBLoadCommand プロパティーの値は以下のようになります。

  ```
  db2 -tvf {1} -z {0}
  ```

- AIX Solaris Linux Oracle

  ```
  DBLoadCommand = sqlldr log={0} control={1} USERID={2}
  ```

ここで、

0 = ログ・ファイル名
1 = コマンド・ファイル名
2 = データベース・ユーザー名

各種のデータベースとオペレーティング・システムの組み合わせに応じて以下の設定を使用してください。

- NT 2000 DB2

  Windows NT または Windows 2000 上で実行する DB2 の場合は、 sqlimport、load、import、または delete メソッドを使用して、クラスパス・システム環境変数に db2/dbconnect.zip が含まれるように設定します。

- AIX Solaris Linux DB2

  AIX、Solaris、または Linux 環境で実行する DB2 の場合は、以下のようにします。

  – sqlimport、load、import、または delete メソッドを使用して、クラスパス・システム環境変数に db2/dbconnect.zip が含まれるように設定します。
  – load または import メソッドを使用して、ローダー・カスタマイザー・プロパティー・ファイル内の以下のプロパティーを変更します。

```
/**
  * Connection command. (Default is for DB2)
  * parameter 0 = dbName
  * parameter 1 = dbUser
  * parameter 2 = userPasswd
*/

DBConnectCommand = connect to {0} user {1} using {2};

/**
  * Load Data into Table command. (Default is for DB2)
  * parameter 0 = filename
  * parameter 1 = column delimiter
  * parameter 2 = character delimiter
  * parameter 3 = name of the table
  * parameter 4 = name of the columns, separated by comma(,)s
*/

DBLoadTableCommand = load from {0} of del modified by coldel{1}
insert into {3} ({4});

/**
  * Insert Data into Table command. (Default is for DB2)
  * parameter 0 = filename
  * parameter 1 = column delimiter
  * parameter 2 = character delimiter
  * parameter 3 = name of the table
  * parameter 4 = name of the columns, separated by comma(,)s
*/

DBUpdateTableCommand = import from {0} of del modified by coldel{1}
insert_update into {3} ({4});

/**
  * System command to invoke load (Default is for DB2)
  * parameter 0 = logFileName
```

```
  * parameter 1 = commandFileName
*/

DBLoadCommand = db2 -z {0} -tf {1}
```

- 400 DB2 iSeries 上で実行する DB2 の場合は、 sqlimport、load、import、または delete メソッドを使用してローダー・カスタマイザー・プロパティー・ファイル内の以下のプロパティーを変更します。

```
/**
  * The connect string.
*/

ConnectStringID = jdbc:db2://

/**
  * The JDBC driver information.
*/

JDBCDriverName = com.ibm.db2.jdbc.app.DB2Driver
DbVendorName=DB2/iSeries

/**
  * Custom writer for load/import methods.
*/

WriterName=com.ibm.wca.MassLoader.Writer.ISeriesWriter
```

- NT 2000 AIX Solaris Linux Oracle

  Windows NT、Windows 2000、AIX、Solaris、または Linux 環境で実行する Oracle データベースの場合は、 sqlimport、load、import、または delete メソッドを使用して、クラスパス・システム環境変数に `oracle/dbconnect.zip` が含まれるように設定します。

コンポーネントを置換する

ローダーのコンポーネントを置換するには、ローダーのカスタマイザー・プロパティー・ファイル内の以下のエレメントに、デフォルトのインプリメンテーションと置換したいクラスの値を指定します。

ParserName
使用するパーサーの名前

ValidatorName
使用するバリデーターの名前

FormatterName
使用するフォーマッターの名前

JDBCFormatterName
SQL import メソッドが使用される場合のフォーマッターの名前

WriterName
使用するライターの名前

JDBCWriterName
SQL import メソッドが使用される場合のライターの名前

たとえば、ローダーのデフォルトのライター (DefaultWriter) をライター com.abc.writer.SpecialWriter に置換するには、ローダーのカスタマイザー・プロパティー・ファイルで以下を指定します。

```
WriterName = com.abc.writer.SpecialWriter
```

ローダーは「com.abc.writer.SpecialWriter」を使用して、書き込み機能を実行するようになります。

商品アドバイザー検索スペースの同期を使用する

商品アドバイザー検索スペースの同期を使用するには、以下のようにします。

1. 「PASyncInfo.xml」という名前の、同期用の XML 構成情報ファイルを作成します。
2. PASyncInfo.xml で、使用する XML スキーマとして PASync.xsd を指定します。たとえば、以下のようにします。

```
<PASync
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation='PASync.xsd'
```

PASync.xsd ファイルが提供されます。以下のテキストは、PASync.xsd の内容を示しています。

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd='http://www.w3.org/2001/XMLSchema'>

  <xsd:element name="PASync">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element ref="SearchScheme" />
       <xsd:element ref="Command" minOccurs="1" maxOccurs="unbounded" />
     </xsd:sequence>
     <xsd:attribute name = "member" type="xsd:string" use="required" />
     <xsd:attribute name = "store" type="xsd:string" use="required" />
   </xsd:complexType>
  </xsd:element>

  <xsd:element name="SearchScheme">
   <xsd:complexType>
     <xsd:sequence>
        <xsd:element ref="RelatedTable" />
          <xsd:element ref="Search" minOccurs="1" maxOccurs="unbounded" />
     </xsd:sequence>
     <xsd:attribute name = "tableName" type="xsd:string" use="required" />
     <xsd:attribute name = "primary" type="xsd:string" use="required" />
     <xsd:attribute name = "colName" type="xsd:string" use="required" />
   </xsd:complexType>
  </xsd:element>

  <xsd:element name="RelatedTable">
   <xsd:complexType>
     <xsd:attribute name = "tableName" type="xsd:string" use="required" />
     <xsd:attribute name = "from" type="xsd:string" use="required" />
     <xsd:attribute name = "to" type="xsd:string" use="required" />
   </xsd:complexType>
  </xsd:element>

  <xsd:element name="Search">
   <xsd:complexType>
     <xsd:attribute name = "value" type="xsd:string" use="required" />
   </xsd:complexType>
  </xsd:element>
```

```
  <xsd:element name="Command">
   <xsd:complexType>
     <xsd:attribute name = "tableName" type="xsd:string" use="required" />
     <xsd:attribute name = "idColumnName" type="xsd:string" use="required" />
     <xsd:attribute name = "addCommand" type="xsd:string" />
     <xsd:attribute name = "updateCommand" type="xsd:string" />
     <xsd:attribute name = "deleteCommand" type="xsd:string" />
   </xsd:complexType>
  </xsd:element>

</xsd:schema>
```

3. PASyncInfo.xml で、同期を行う必要のあるメンバー ID とストア・エンティティー ID を指定します。たとえば、以下のようにします。

```
member = "-2000"
store = "10351"
```

4. PASyncInfo.xml 内の検索スキーマ・エレメントの下に、検索スペースを構成する CATGROUP ID を指定します。たとえば、以下のようにします。

```
<SearchScheme
  tableName = "catgroup"
  primary = "CATGROUP_ID"
  colName = "identifier" >

  <RelatedTable
    tableName = "catgpenrel"
    from = "CATGROUP_ID"
    to = "CATENTRY_ID" />

  <Search value="Pants" />
  <Search value="Shirts" />

</SearchScheme>
```

この例では、″Pants″ と ″Shirts″ が指定されています。 CATGROUP ID は必要な数だけ指定できます。

5. PASyncInfo.xml で、コマンドがスケジュールするものを決定する属性を指定します。たとえば、以下のようにします。

```
    <Command tableName = "CATENTRY" idColumnName = "CATENTRY_ID"
      updateCommand = "UpdateSearchSpaces"
      deleteCommand = "RemoveProductsFromAllSearchSpaces"
    />

    <Command tableName = "CATENTDESC" idColumnName = "CATENTRY_ID"
      addCommand = "UpdateSearchSpaces"
      updateCommand = "UpdateSearchSpaces"
      deleteCommand = "UpdateSearchSpaces"
    />

    <Command tableName = "LISTPRICE" idColumnName = "CATENTRY_ID"
      addCommand = "UpdateSearchSpaces"
      updateCommand = "UpdateSearchSpaces"
      deleteCommand = "UpdateSearchSpaces"
    />

    <Command tableName = "ATTRVALUE" idColumnName = "CATENTRY_ID"
      addCommand = "UpdateSearchSpaces"
      updateCommand = "UpdateSearchSpaces"
      deleteCommand = "UpdateSearchSpaces"
    />

    <Command tableName = "CATENTATTR" idColumnName = "CATENTRY_ID"
      addCommand = "UpdateSearchSpaces"
```

```
      updateCommand = "UpdateSearchSpaces"
      deleteCommand = "UpdateSearchSpaces"
    />

    <Command tableName = "CATGPENREL" idColumnName = "CATENTRY_ID"
      addCommand = "AddProductsToSearchSpace"
      deleteCommand = "RemoveProductsFromSearchSpace"
    />

</PASync>
```

6. 新しいローダーのカスタマイザー・プロパティー・ファイルで、XML 構成情報ファイルを指定します。たとえば、以下のようにします。

```
PASyncDocumentURL = PASyncInfo.xml
```

7. 新しいローダーのカスタマイザー・プロパティー・ファイルで、同期を使用可能にします。たとえば、以下のようにします。

```
PASyncEnabled = true
```

8. Load コマンドと共に SQL import または delete のいずれかのメソッドを使用します。以下は、ローダーに対する正しい XML 入力データの例です。

```
<store-asset>

   <catentry
       CATENTRY_ID="10351"
       MEMBER_ID="-2000"
       PARTNUMBER="000051"
       CATENTTYPE_ID="ProductBean"
       MFPARTNUMBER="m000051"
       MARKFORDELETE="0"
       BUYABLE="1"
     />

   <catentry
       CATENTRY_ID="10352"
       MEMBER_ID="-2000"
       PARTNUMBER="000052"
       CATENTTYPE_ID="ProductBean"
       MFPARTNUMBER="m000052"
       MARKFORDELETE="0"
       BUYABLE="1"
     />

   <catentry
       CATENTRY_ID="10353"
       MEMBER_ID="-2000"
       PARTNUMBER="000053"
       CATENTTYPE_ID="ProductBean"
       MFPARTNUMBER="m000053"
       MARKFORDELETE="0"
       BUYABLE="1"
     />

   <catentry
       CATENTRY_ID="10358"
       MEMBER_ID="-2000"
       PARTNUMBER="000058"
       CATENTTYPE_ID="ProductBean"
       MFPARTNUMBER="m000058"
       MARKFORDELETE="0"
       BUYABLE="1"
     />

   <catentry
       CATENTRY_ID="10365"
       MEMBER_ID="-2000"
```

```
        PARTNUMBER="000065"
        CATENTTYPE_ID="ProductBean"
        MFPARTNUMBER="m000065"
        MARKFORDELETE="0"
        BUYABLE="1"
      />

    <catentry
        CATENTRY_ID="10372"
        MEMBER_ID="-2000"
        PARTNUMBER="000072"
        CATENTTYPE_ID="ProductBean"
        MFPARTNUMBER="m000072"
        MARKFORDELETE="0"
        BUYABLE="1"
      />

      <catgpenrel
        CATGROUP_ID="10354"
        CATENTRY_ID="10372"
        CATALOG_ID="10351"
        SEQUENCE="3"
      />

      <catgpenrel
        CATGROUP_ID="10354"
        CATENTRY_ID="10365"
        CATALOG_ID="10351"
        SEQUENCE="4"
      />

      <catgpenrel
        CATGROUP_ID="10354"
        CATENTRY_ID="10358"
        CATALOG_ID="10351"
        SEQUENCE="5"
      />

      <catgpenrel
        CATGROUP_ID="10355"
        CATENTRY_ID="10372"
        CATALOG_ID="10351"
        SEQUENCE="3"
      />

  </store-asset>
```

注: 商品アドバイザー検索スペースの同期を使用不可にすると、ローダーのパフォーマンスが向上します。したがって、この機能は必要な場合に限り使用してください。

商品アドバイザー検索スペースの同期のカスタマイズ

ローダー・パッケージを使用すると、ローダーのカスタマイザー・プロパティー・ファイルを変更して以下を行うことによって、商品アドバイザー検索スペースの同期をカスタマイズできます。

- **同期を使用可能または使用不可にする**

  同期は、ローダーのカスタマイザー・プロパティー・ファイル内の以下のプロパティーの値として `true` または `false` を指定することによって、使用可能または使用不可にすることができます。

  ```
  PASyncEnabled = true
  ```

- **同期のための構成情報ファイルを指定する**

ローダーのカスタマイザー・プロパティー・ファイル内の以下のプロパティーの値を設定することによって、同期に使用される XML 構成情報ファイルを指定することができます。

```
PASyncDocumentURL = PASyncInfo.xml
```

- **スケジュール照会の長さを指定する**

  スケジュール照会の長さは、ローダーのカスタマイザー・プロパティー・ファイル内の以下のプロパティーの値を設定することによって指定できます。

  ```
  PAScheduleQueryLength = 30
  ```

  このプロパティーの値は、20 ～ 900 の範囲内でなければなりません。

- **スケジュールされている開始時刻を指定する**

  スケジュールされている開始時刻を指定するには、ローダーのカスタマイザー・プロパティー・ファイル内の PAScheduledStartTime プロパティーの値として、絶対タイム・スタンプ、現行タイム・スタンプ、または存続時間付きの現行タイム・スタンプのいずれかを指定します。

  注: タイム・スタンプのフォーマットは、ご使用のデータベースに適したものでなければなりません。

  以下は、ロードの 5 分後にスケジュールされているジョブを実行する DB2 の例です。

  ```
  PAScheduledStartTime = CURRENT TIMESTAMP + 5 MINUTES
  ```

  Oracle 以下は、即時にジョブを実行する Oracle データベースの例です。

  ```
  PAScheduledStartTime = SYSDATE
  ```

ローダーを使用する場合のデータ処理方法の決定

ローダーは、Load コマンドを使用したデータ処理に、以下のオプションを提供しています。

- ロード
- インポート
- SQL インポート機能の使用

データをロードする前に、どの処理方法が最善の結果を生成するかを判別する必要があります。

load メソッドの選択

以下の場合は、load メソッドを使用することを考慮してください。

- データがクリーンであることが分かっている場合、およびデータベースにデータが含まれていない場合
- データがクリーンであることが分かっている場合、およびロードされるデータがデータベースに含まれていないことが分かっている場合
- データがクリーンであることが分かっている場合、ターゲットのテーブルに 1 次鍵が含まれていない場合、およびロードされるデータがデータベースに含まれていないことが分かっている場合
- ロード時間が主な関心事である場合

- データベースがローカル DB2 データベースである場合

400 データは load メソッドでデータベースにロードされます。データがすでに存在している場合、このコマンドは重複エラーのために失敗し、重複エラー・メッセージが表示されます。

import メソッドの選択

NT 2000 AIX Solaris Linux DB2 の場合の import メソッドでも、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している場合、それは削除されずに新しい値で更新されます。以下の場合は、このメソッドを使用することを考慮してください。

- データベース管理システムが DB2 である場合
- データがクリーンかどうか分からない場合
- 大規模な同種データ・セットを列レベルで更新しなければならない場合
- ロード時間が主な関心事でない場合
- データをインポートするテーブルに 1 次鍵が含まれている場合

400 import メソッドで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している場合、それは削除されずに新しい値で更新されます。以下の場合は、このメソッドを使用することを考慮してください。

- データがクリーンかどうか分からない場合
- データベース内にすでにデータが存在している場合
- ロード時間が主な関心事でない場合
- データをインポートするテーブルに 1 次鍵が含まれている場合

SQL import メソッドの選択

SQL import メソッドでは、データベース内のデータの更新または挿入に、JDBC または SQL ステートメントが使用されます。データはまだ存在していない場合に挿入され、既存のデータが更新されます。以下の場合は、このメソッドを使用することを考慮してください。

- 既存のデータを更新するときに列レベルの更新が必要な場合

  このメソッドでは、制約違反とデータ・タイプ・エラーについての優れたエラー・レポートが行われます。
- データの一部がクリーンでないことが分かっている場合
- データベースの保全性が主な関心事である場合
- データベースがローカルでない場合
- 商品アドバイザー検索スペースの同期を使用する場合

その他の考慮事項

- **load メソッドの使用上の制約事項**

  load メソッドは、ビット・データ・フィールドのデータの挿入または更新を行うことはできません。

DB2 load メソッドでは、新規レコードのみがデータベースに挿入され、既存のレコードは更新されません。

- **import メソッドの使用上の制約事項**

  import メソッドは、ビット・データ・フィールドのデータの挿入または更新を行うことはできません。

  DB2 import メソッドでは、ローダーは 1 次鍵が定義されているテーブルにのみ挿入または更新を行います。 import メソッドは、 1 次鍵を持たないテーブル内でデータを挿入または更新することはできません。入力レコードが基本の列の値しか持たない場合、そのレコードは拒否されます。

- **SQL import メソッドと load メソッドの比較**

  SQL import メソッドは外部参照を含めたデータ整合性を検査するため、これを使用すると既存のデータの更新が可能になります。 load メソッドはこれを行いません。

- **import メソッドと SQL import メソッドの比較**

  import および SQL import メソッドは、類似した機能を実行します。 import メソッドは基本的に高速ですが、一次ファイル用のディスク・スペースを必要とします。

  DB2 import メソッドは、 1 次鍵が定義されているテーブルにのみ挿入または更新を行うことができます。一方、SQL import メソッドでは、テーブルに 1 次鍵が含まれている必要はありません。

- **その他の考慮事項**

  delete メソッドは、入力 XML 文書内にあるデータをデータベースから削除するために使用されます。エレメントには、テーブルの 1 次鍵または固有索引の値が含まれていなければなりません。削除されるデータに、「cascade on delete」が使用可能になっている他のテーブル内のデータに対する従属関係がある場合は、その従属関係も削除されます。

  商品アドバイザー検索スペースの同期を使用する場合、データのロードには SQL import メソッドを使用する必要があります。

  import および load メソッドは DB2 用に最適化されているネイティブ・ユーティリティーを使用しますが、 SQL import メソッドは (多くのデータベース製品にとって一般的な) JDBC 呼び出しを使用します。

大規模な文書のロード

ローダー・パッケージのユーティリティーを使用して大規模な文書をデータベースにロードする場合は、以下の事項について考慮してください。

- **Java 仮想マシン (JVM) のヒープ・サイズ**

  デフォルトで、JVM ヒープに割り振られるメモリーの最大量は 64 MB です。これを大きくしないと、ロード・プロセス中に、JVM はいつかはメモリー不足になる可能性があります。 Java ヒープに割り振られるメモリーの最大量は、 Java コマンドの JVM -mx オプションを使用することによって変更できます。

- **トレース・ログ**

  トレース・ロガーは、大規模な XML 文書をロードする際に JVM ヒープを使い尽くす可能性があります。トレース情報は、主に実行が失敗した場合にその実行

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必要でなければ、トレースはオフにすべきです。トレースをオフにすると、パフォーマンスは大きく向上します。トレースをオフにするには、ログ構成 XML 文書を変更します。

デフォルトのログ構成ファイルは WCALoggerConfig.xml です。トレース・ログをオフにするには、ローダーの以下のトレース・ロガー構成を変更します。

```
<logger type="trace">
   <handler type="file">
      <filePath>MassLoadTrace.log</filePath>
      <filter type="Any">
           <messageType name="PUBLIC" />
      </filter>
   </handler>
</logger>
```

以下のようにします。

```
<logger type="trace">
   <handler type="file">
      <filePath>MassLoadTrace.log</filePath>
      <filter type="Any">
      </filter>
   </handler>
</logger>
```

WCALoggerConfig.xml ファイルの変更についての詳細は、 74 ページの 『ローダー・パッケージ用のロギングのカスタマイズ』を参照してください。

- **コミット・カウント**

  SQL インポート・モードで操作している場合のローダーのデフォルトのコミット・カウントは 1 です。したがって、デフォルトではデータベースへの更新または挿入のたびにトランザクションがコミットされます。大規模な文書用のローダーのパフォーマンスを改善するには、コミット・カウントを増加させる必要があります。値は「100」にすることをお勧めします。しかし、これはサーバーの物理メモリーの量や DBMS のトランザクション・ログ・サイズなどに応じて、もっと大きくすることができます。

  ローダーのコミット・カウントは、 Load コマンドの -commitcount *count* オプション (*count* は、トランザクションがコミットされるまでに実行されるステートメントの数) を使用して変更します。

トラブルシューティングのヒント

データをロードする際の進行速度が異常に遅い場合は、ローダーのロガーに正しく構成されていないファイル・ハンドラーが含まれている可能性があります。このことは、以下のいずれかの状況から発生すると考えられます。

- ローダーを起動しているユーザーが、ディレクトリーへの書き込みまたはログ構成文書内で指定されているファイルの更新を行うための許可を持っていない。
- ログ構成文書内でファイルの場所として指定されているディレクトリーが存在していない。
- ログ構成文書内でファイルの場所として指定されているドライブに、十分なスペースがない。

これらの問題のいずれかを訂正するときは、ログ構成文書 (デフォルトでは WCALoggerConfig.xml) を変更することによって、指定されているファイルの場所を変更する必要がある場合があります。ファイル・ハンドラーおよび WCALoggerConfig.xml ファイルについての詳細は、74 ページの『ローダー・パッケージ用のロギングのカスタマイズ』を参照してください。

第 7 章 データの抽出

Extractor を使用し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・ファイルを使用してデータベースから抽出するデータを指定する必要があります。使用する抽出フィルターは、抽出するデータのタイプによって異なります。

以下の例では、抽出フィルターとして MemberSubsystemFilter.xml を使用して、データベースからメンバー・サブシステム・データを抽出します。

- NT 2000 AIX Solaris Linux

```
java com.ibm.wca.MassExtract.Extract -filter MemberSubsystemFilter.xml
-outfile MemberSubsystemExtracted.xml -dbname mall -dbuser myname
-dbpwd mypassword -customizer MemberSubsystemCustomizer
```

- 400

```
QWEBCOMM/EXTWCSDTA FILTER(MemberSubsystemFilter.xml)
OUTFILE(MemberSubsystemExtracted.xml) DATABASE(database_name)
SCHEMA(mall) INSTROOT(/QIBM/UserData/WebCommerce/instances/mser)
PASSWD(mypassword)
```

抽出フィルターの作成

以下の抽出フィルターの例では、 CATGROUP、CATGRPDESC、CATGRPREL、CATENTRY、CATENTSHIP、OFFER、CATENTREL、CATGPENREL、CATENTDESC、および ATTRVALUE テーブルからカテゴリーおよび商品情報を抽出します。

```
<sqlx>

    <!-- ************************** -->
    <!-- extract Category information -->
    <!-- ************************** -->

    <functionDef id="Category" description="Extract Categories" schemaentity="catgroup">
      <paramDef name=":lastRecord" type="string" value="10301" description="Last record
       before loading new data" />
      <body>
        select * from catgroup where catgroup_id > :lastRecord
      </body>
    </functionDef>

    <execute id="Category" description="Extract Categories" schemaentity="catgroup">
      <param name=":lastRecord" type="string" value="10300" description="Last record
       before loading new data" />
    </execute>

    <functionDef id="Category Description" description="Extract Category Descriptions
     for a Locale" schemaentity="catgrpdesc">
      <paramDef name=":lastRecord" type="string" value="10300" description="Last record
       before loading new data" />
      <body>
        select * from catgrpdesc where catgroup_id > :lastRecord
      </body>
    </functionDef>

    <execute id="Category Description" description="Extract Category Descriptions
     for a Locale" schemaentity="catgrpdesc">
      <param name=":lastRecord" type="string" value="10300" description="Last record
       before loading new data" />
```

```
</execute>

<functionDef id="Category Relationship" description="Extract Category-Relations
 for a Locale" schemaentity="catgrprel">
  <paramDef name=":lastRecord" type="string" value="10300" description="Last record
   before loading new data" />
  <body>
    select * from catgrprel where catgroup_id_child > :lastRecord
  </body>
</functionDef>

<execute id="Category Relationship" description="Extract Category-Relations for
 a Locale" schemaentity="catgrprel">
  <param name=":lastRecord" type="string" value="10300" description="Last record
   before loading new data" />
</execute>

<!-- ************************** -->
<!-- extract Product information -->
<!-- ************************** -->

<functionDef id="Product" description="Extract Product" schemaentity="catentry">
  <paramDef name=":lastrecord" type="string" value="10300" description="Last record
   before loading new data" />
  <body>
    select * from catentry where catentry_id > :lastrecord
  </body>
</functionDef>

<execute id="Product" description="Extract Product" schemaentity="catentry">
  <param name=":lastrecord" type="string" value="10300" description="Last record
   before loading new data" />
</execute>

<functionDef id="Product Relationship" description="Extract Product Ship
 information" schemaentity="catentrel">
  <paramDef name=":lastrecord" type="string" value="10300" description="Last record
   before loading new data" />
  <body>
    select * from catentrel where catentry_id_child > :lastrecord
  </body>
</functionDef>

<execute id="Product Relationship" description="Extract Product Ship information"
 schemaentity="catentrel">
  <param name=":lastrecord" type="string" value="10300" description="Last record
   before loading new data" />
</execute>

<functionDef id="Product Description" description="Extract Product Description"
 schemaentity="catentdesc">
  <paramDef name=":lastrecord" type="string" value="10300" description="Last record
   before loading new data" />
  <body>
    select * from catentdesc where catentry_id > :lastrecord
  </body>
</functionDef>

<execute id="Product Description" description="Extract Product Description"
 schemaentity="catentdesc">
  <param name=":lastrecord" type="string" value="10300" description="Last record
   before loading new data" />
</execute>

<functionDef id="Product Ship" description="Extract Product Ship information"
 schemaentity="catentship">
  <paramDef name=":lastrecord" type="string" value="10300" description="Last record
   before loading new data" />
  <body>
    select * from catentship where catentry_id > :lastrecord
  </body>
</functionDef>
```

```
    <execute id="Product Ship" description="Extract Product Ship information"
     schemaentity="catentship">
      <param name=":lastrecord" type="string" value="10300" description="Last record
       before loading new data" />
    </execute>

    <functionDef id="Category Product Relationship" description="Extract Category
     Product Relations" schemaentity="catgpenrel">
      <paramDef name=":lastrecord" type="string" value="10300" description="Last record
       before loading new data" />
      <body>
        select * from catgpenrel where catgroup_id > :lastrecord
      </body>
    </functionDef>

    <execute id="Category Product Relationship" description="Extract Category Product
     Relations" schemaentity="catgpenrel">
      <param name=":lastrecord" type="string" value="10300" description="Last record
       before loading new data" />
    </execute>

    <!-- ************************************** -->
    <!-- Extract Product Attribute Information    -->
    <!-- ************************************** -->

    <functionDef id="Product Attribute Values" description="Extract Product Attribute
     values for a Locale" schemaentity="attrvalue">
      <paramDef name=":lastrecord" type="string" value="10300" description="Last record
       before loading new data" />
      <body>
        select * from attrvalue where catentry_id > :lastrecord
      </body>
    </functionDef>

    <execute id="Product Attribute Values" description="Extract Product Attribute values
     for a Locale" schemaentity="attrvalue">
      <param name=":lastrecord" type="string" value="10300" description="Last record
       before loading new data" />
    </execute>

    <!-- ********************************************** -->
    <!-- Extract Product Price Information              -->
    <!-- ********************************************** -->

    <functionDef id="Offer" description="Extract Offer" schemaentity="offer">
      <paramDef name=":lastrecord" type="string" value="10300" description="Last record
       before loading new data" />
      <body>
        select * from offer where catentry_id > :lastrecord
      </body>
    </functionDef>

    <execute id="Offer" description="Extract Offer" schemaentity="offer">
      <param name=":lastrecord" type="string" value="10300" description="Last record
       before loading new data" />
    </execute>

</sqlx>
```

Extractor のセットアップ

以下のことを実行することにより、Extractor が使用するデータベース・ドライバーを変更することができます。

1. 新規の Extractor カスタマイザー・プロパティー・ファイルを作成します。
   - NT 2000 AIX Solaris Linux

     DB2ConnectionCustomizer.properties は、MassExtract.zip アーカイブにあります。このファイルを抽出し、拡張子は .properties のままでファイルの名前を変更してから、そのファイルをクラスパス内にあるディレクトリー内に入れます。**重要:** 既存の DB2ConnectionCustomizer.properties ファイルは除去または変更しないでください。
   - 400

     ISeries_EXTWCSDTA_Customizer.properties が /QIBM/ProdData/WebCommerce/properties ディレクトリーにあります。このファイルを */instroot*/xml ディレクトリーにコピーし、拡張子は .properties のままで新規ファイルの名前を変更してから、新規ファイルに必要な変更を加えます。**重要:** 元の ISeries_EXTWCSDTA_Customizer.properties ファイルは除去または変更しないでください。
2. 新規ファイル内のデータベース・ドライバー値を変更します。
3. Extract コマンドのカスタマイザー・パラメーターの値として新規ファイル名を指定します。

以下は、Extractor カスタマイザー・プロパティー・ファイルからの抜粋です。

```
DBVendorName = DB2
DBDriverName = COM.ibm.db2.jdbc.app.DB2Driver
DBURL = jdbc:db2:
```

ここで、

- DBVendorName は、データベースのタイプを選択するために使用されます。

  オプションは、以下のとおりです。
  - DB2 Universal Database for iSeries (DB2/iSeries)
  - その他のオペレーティング・システム用の DB2 (DB2)
  - Oracle データベース (Oracle)
- DBDriverName は、JDBC ドライバーを選択するために使用されます。

  オプションは、以下のとおりです。
  - DB2 Universal Database for iSeries (com.ibm.db2.jdbc.app.DB2Driver)
  - その他のオペレーティング・システム用の DB2 (COM.ibm.db2.jdbc.app.DB2Driver)
  - Oracle データベース (oracle.jdbc.driver.OracleDriver)
- DBURL は、データベースにアクセスするための URL を指定するために使用されます。

  オプションは、以下のとおりです。
  - DB2 Universal Database for iSeries (jdbc:db2://)
  - その他のオペレーティング・システム用の DB2 (jdbc:db2:)

– Oracle データベース (`jdbc:oracle:oci8:@`)

第 8 章 ローダー・パッケージ・ロガーの使用

ローダー・パッケージ内のそれぞれのユーティリティーは、プログラム・トレース情報を提供するだけでなく、成功、失敗、およびエラーを示すメッセージを作成します。

ローダー・パッケージのユーティリティーは、WCALoggerConfig.xml ファイルを参照します。

- NT 2000 AIX Solaris Linux

  このファイルは、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あります。また、このファイルは、com.ibm.wca.logging.configFile Java システム・プロパティーによっても指定できます。

- 400

  このファイルは、*/instroot*/xml ディレクトリーにあります。

WCALoggerConfig.xml は、それぞれのユーティリティーが提供するロギング情報、およびその情報が表示または保管される場所を判別します。このファイルをカスタマイズし、ログ記録されるメッセージのタイプだけでなく、作成されるログのタイプを指定することができます。

ご使用の環境 (Windows NT、Windows 2000、 AIX、Linux、および Solaris システム) でのロギングの構成

ご使用の環境でロギングをセットアップするには、ファイル WCALoggerConfig.xml を含むようにクラスパス変数を設定するか、または com.ibm.wca.logging.configFile システム・プロパティーを指定する必要があります。

クラスパス変数の設定の例

WCALoggerConfig.xml ファイルがディレクトリー d:¥WebSphere¥CommerceServer¥xml¥loader (Windows NT マシンの場合)、以下のステートメントを使用してクラスパス変数を設定することができます。

```
SET CLASSPATH=%CLASSPATH%;D:¥WebSphere¥CommerceServer¥xml¥loader
```

com.ibm.wca.logging.configFile システム・プロパティーの設定の例

com.ibm.wca.logging.configFile システム・プロパティーを指定するには、 Java インタープリターを呼び出す際に -D オプションを使用します。以下に例を示します。

```
java -Dcom.ibm.wca.logging.configFile=D:¥ice_tea¥src¥classlib¥logger¥xml¥WC.xml
com.ibm.wca.DTDGenerator.GenerateDTD
```

ローダー・パッケージ用のロギングのカスタマイズ

ローダー・パッケージ用のロギングをカスタマイズするには、WCALoggerConfig.xml ファイルを使用します。

- NT 2000 AIX Solaris Linux

  このファイルは、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あります。また、このファイルは、com.ibm.wca.logging.configFile Java システム・プロパティーによっても指定できます。

- 400

  このファイルは、*/instroot*/xml ディレクトリーにあります。

WCALoggerConfig.xml には、1 つまたは複数のコンポーネント・タグ (たとえば、<component name="DTDGenerator">) が含まれています。それぞれのタグ内に、ロガーおよびハンドラーを追加することができます。システムで提供されているユーティリティーおよびロガー・タグは変更できません が、ハンドラー・タグをロガーに追加することはできます。このファイル内に含めることができるタグについては、WCALogger.dtd ファイルを参照してください。

ローダー・パッケージ・ログは、以下のディレクトリーの messages.txt ファイル内にあります。

- NT *drive*:¥WebSphere¥CommerceServer¥instances¥*instance_name*¥logs
- 2000 *drive*:¥Program Files¥WebSphere¥CommerceServer¥instances¥*instance_name*¥logs
- AIX /usr/WebSphere/CommerceServer/instances/*instance_name*/logs
- Solaris Linux /opt/WebSphere/CommerceServer/instances/*instance_name*/logs
- 400 /QIBM/UserData/WebCommerce/instances/*instance_name*/logs

ハンドラー

ハンドラーをロガーに追加するには、WCALoggerConfig.xml ファイル内にハンドラー・タイプを指定します。複数のハンドラーをロガーに追加することができます。それぞれのハンドラーは、独自の属性および従属タグを持っており、それらは必ずしも他のハンドラーに適用する必要がないことに注意してください。ハンドラー・タイプには、以下のものがあります。

ハンドラー・タイプ	説明および属性
console	標準出力 (通常は、コマンド行) にメッセージを送信します。
file	テキスト・ファイル内にメッセージを保管します。 このハンドラーに従属タグとして "<filePath>log path</filePath>" を追加する必要があります。
multifile	ファイルの循環ログを作成します。 "<filePath>log path</filePath>" を指定する必要があります。ログ・ファイル 1 ～ *n* が作成されます。以下の属性を追加することができます。 **MaxFiles** 最初のログ・ファイルを消去する前に使用するログ・ファイル数を示す整数 **MaxKBFileSize** それぞれのログ・ファイル内に保管する最大 K バイト数を示す整数

NT 2000 AIX Solaris Linux database	循環ログ内の DB2 表内にメッセージを保管します。以下の属性を追加することができます。 **brand** データベース・ブランド名。 DB2 は、現在サポートされている唯一のデータベースです。 **maxRows** 最も古いエントリーを消去する前にテーブル内に保管する最大レコード数 従属タグとして "<jdbc/>" を含め、以下の属性を含めることができます。 **url** データベースにアクセスするために JDBC で使用される URL (たとえば、 "Jdbc:db2:"*wcm*""。ここで、*"wcm"* はデータベースの名前です)。データベースは、ユーティリティーを実行する前に存在していなければなりません。 **table** メッセージがログ記録されるデータベース・テーブルの名前。テーブルは、以下の DB2 ステートメントを使用して作成しなければなりません。 "CREATE TABLE"*tablename* (KEY char(13) FOR BIT DATA NOT NULL, COMPONENTNAME VARCHAR(30), ENTRY VARCHAR(2000), PRIMARY KEY(key))"" **userid** データベース・ユーザー名。ユーザーは、テーブルを更新するための許可を割り当てられている必要があります。以下の DB2 ステートメントによって許可が割り当てられます。 "GRANT SELECT, UPDATE, INSERT, DELETE ON TABLE"*tablename* TO USER *userid*"" **password** 指定されているユーザー名用のデータベース・パスワード

以下の例では、″database″ というタイプのハンドラーをロガーに追加します。

```
<handler type="database" brand="DB2" maxRows="50">
 <jdbc url="jdbc:db2:wcm"
    table="wcm.log"
    userid= "wasuser"
    password="123456"/>
 <filter type="Any">
   <messageType name="FATAL"/>
   <messageType name="ERROR"/>
   <messageType name="WARNING"/>
 </filter>
</handler>
```

フィルター

フィルターをハンドラーに追加したり、ハンドラーから除去したりして、メッセージ・タイプを組み込んだり、除外したりすることができます。ロガーにフィルター

がない場合には、メッセージはログ記録されません。それぞれのフィルター・タグにはメッセージ・タイプをリストする従属の messageType タグがあり、通常は以下のいずれかになります。

- INFO
- ERROR
- FATAL
- WARNING

その他のメッセージ・タイプは、WCALogger.dtd ファイル内にリストされていますが、大部分は通常、ローダー・パッケージでは使用されません。

フィルター・タイプには以下のものがあります。

フィルター・タイプ	説明および属性
Any	指定されている messageType タイプの 1 つとしてフラグが付いているメッセージをログ・ファイル内に含みます。 たとえば、messageType リストに ERROR が含まれており、アプリケーションが ERROR タイプ・メッセージを生成する場合、メッセージはログ記録されます。
All	メッセージは、ログに組み込まれる前に、指定されたすべての messageType タイプ属性を持っている必要があります。
Exclude	messageType タグのリスト内に指定されていないすべてのメッセージをログ記録します。

以下の例ではフィルターをハンドラーに追加しますが、その際に ERROR メッセージ・タイプだけでなく FATAL メッセージ・タイプもログ記録され、その他のメッセージは無視されます。

```
<handler type="file">
   <filter type="Any">
  <messageType name="FATAL"/>
  <messageType name="ERROR"/>
   </filter>
</handler>
```

フォーマット

以下のメッセージ・フォーマット用の 2 つのフォーマッター・タイプの 1 つを指定することができます。

フォーマッター・タイプ	説明および属性
safe (デフォルト)	プロパティー・ファイル内にメッセージが見つからない場合に例外が設定されないようにします。 このフォーマッターは、リソースが欠落していることを示すメッセージを作成します。

xml	XML フォーマットでメッセージをフォーマットします。 メッセージが見つからない場合、このフォーマッターは、例外を設定する代わりにメッセージも書き込みます。

例: WCALoggerConfig.xml および WCALogger.dtd

WCALoggerConfig.xml

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE WCALoggerConfig SYSTEM "WCALogger.dtd">
<WCALoggerConfig>
<component name="MassLoader">
 <logger type="message">
  <handler type="file">
   <filePath>c:¥temp¥out.txt</filePath>
    <filter type="Any">
     <messageType name="FATAL"/>
     <messageType name="ERROR"/>
     <messageType name="WARNING"/>
     <messageType name="INFO"/>
    </filter>
  </handler>
 </logger>
 <logger type="trace">
  <handler type="file">
   <filePath>out2.txt</filePath>
    <filter type="Any">
     <messageType name="PUBLIC"/>
    </filter>
 </handler>
 </logger>
 <logger type="typedMessage">
  <handler type="file">
   <filePath>tout.txt</filePath>
    <filter type="Any">
     <messageType name="FATAL"/>
     <messageType name="ERROR"/>
     <messageType name="WARNING"/>
     <messageType name="INFO"/>
    </filter>
 </handler>
 </logger>
 <logger type="progress">
  <handler type="console" format="safe">
   <filter type="Any">
     <messageType name="FATAL"/>
     <messageType name="ERROR"/>
     <messageType name="WARNING"/>
     <messageType name="INFO"/>
   </filter>
  </handler>
 </logger>
</component>
<component name="DTDGenerator">
 <logger type="message">
  <handler type="console">
   <filter type="Any">
     <messageType name="FATAL"/>
     <messageType name="ERROR"/>
     <messageType name="WARNING"/>
     <messageType name="INFO"/>
   </filter>
```

```
  </handler>
 </logger>
 <logger type="trace">
  <handler type="console">
   <filter type="Any">
     <messageType name="FATAL"/>
     <messageType name="ERROR"/>
     <messageType name="WARNING"/>
     <messageType name="INFO"/>
   </filter>
  </handler>
 </logger>
</component>
</WCALoggerConfig>
```

WCALogger.dtd

```
<!-- This DTD describes how a WCALoggerConfig XML can be structured.
A WCALoggerConfig XML document is the input configuration file for
the WCALoggerFactory class.
-->

<!ELEMENT WCALoggerConfig (component)+>

<!ELEMENT component (logger)+>
<!ATTLIST component name CDATA #REQUIRED>
<!ELEMENT logger (handler+,messageFile?)>
<!ATTLIST logger type (message | trace | typedMessage | progress) "typedMessage">

<!-- messageFile is an optional default properties files that can be used to
make messages locale specific
-->
<!ELEMENT messageFile (#PCDATA)>
<!ELEMENT handler (filePath?, filter, jdbc?)>
<!ATTLIST handler
type ( file|multiFile|console|error|textArea|database|ejbQueue|queue ) "console">

<!-- maxFiles & maxKBFileSize only applies to the multiFile type of handler
-->
<!-- filePath & encoding applies only when the handler is of type file or
multiFile
-->
<!ATTLIST handler maxFiles CDATA #IMPLIED>
<!ATTLIST handler maxKBFileSize CDATA #IMPLIED>
<!ATTLIST handler encoding CDATA #IMPLIED>
<!ATTLIST handler format (safe | xml) "safe">
<!-- maxRecords & brand are only applicable to database handler type
-->
<!ATTLIST handler maxRecords CDATA #IMPLIED>
<!ATTLIST handler brand (DB2) #IMPLIED>
<!-- the jdbc tag must be present within a database handler type tag
-->
<!ELEMENT jdbc EMPTY>
<!ATTLIST jdbc url CDATA #IMPLIED>
<!ATTLIST jdbc table CDATA #IMPLIED>
<!ATTLIST jdbc userid CDATA #IMPLIED>
<!ATTLIST jdbc password CDATA #IMPLIED>

<!ELEMENT filter (messageType+)>
<!ATTLIST filter type (Any | All | Exclude ) "Any">

<!-- the messageType attribute name is one of these JLog IRecordType
constants
-->
<!ELEMENT messageType EMPTY>
<!ATTLIST messageType name ( NONE | ALL | INFO |
```

```
INFORMATION | WARN | WARNING | ERR | ERROR |
FATAL | DEFAULT_MESSAGE | API | CALLBACK |
ENTRY_EXIT | ENTRY | EXIT | ERROR_EXC |
MISC_DATA | OBJ_CREATE | OBJ_DELETE |
PRIVATE | PUBLIC | STATIC | SVC | PERF |
LEVEL1 | LEVEL2 | LEVEL3 ) "ALL">
<!ELEMENT filePath (#PCDATA)>
```

第 9 章 ローダー・パッケージのエラー・レポーターの使用

ローダーおよび ID リゾルバーには、エラーがある場合に例外文書を生成するエラー・レポーターが含まれています。

デフォルトでは、例外文書は以下のディレクトリーに書き込まれます。

- NT 2000 AIX Solaris Linux 入力文書が存在するディレクトリー
- 400 `/instroot/logs`

例外文書の書き込み先のディレクトリーを指定するには、 Java プロパティー com.ibm.wcm.ErrorReporterDir を使用します。 Windows NT 環境のローダーを例にした場合、始まりは次のようになります。

```
java -Dcom.ibm.wcm.ErrorReporterDir=d:¥massloaderrors
com.ibm.wca.MassLoader.MassLoad -dbname . . .
```

注: ユーザーには、指定されたディレクトリーへの書き込み許可が必要です。

以下は、エラー・レポーター用のサンプル DTD (`store-all-error.dtd`) です。

```
<!ENTITY % TABLE "calrule | catentry">
<!ELEMENT store-asset (error, (%TABLE;)*)>
<!ELEMENT message (#PCDATA) >
<!ELEMENT error ( message ) >
<!ATTLIST error
  locus             CDATA     #REQUIRED
  id                CDATA     #REQUIRED
>
<!ELEMENT calrule (error)>
<!ATTLIST calrule
  identifier        CDATA     #REQUIRED
  calrule_id        CDATA     #REQUIRED
  calcode_id        CDATA     #REQUIRED
  startdate         CDATA     #IMPLIED
  taxcgry_id        CDATA     #IMPLIED
  enddate           CDATA     #IMPLIED
  sequence          CDATA     #REQUIRED
  combination       CDATA     #REQUIRED
  calmethod_id      CDATA     #REQUIRED
  calmethod_id_qfy  CDATA     #REQUIRED
  flags             CDATA     #REQUIRED
  field1            CDATA     #IMPLIED
  field2            CDATA     #IMPLIED
>
<!ELEMENT catentry (error)>
<!ATTLIST catentry
  catentry_id       CDATA     #REQUIRED
  member_id         CDATA     #REQUIRED
  catenttype_id     CDATA     #REQUIRED
  partnumber        CDATA     #IMPLIED
  mfpartnumber      CDATA     #IMPLIED
  mfname            CDATA     #IMPLIED
  markfordelete     CDATA     #REQUIRED
  url               CDATA     #IMPLIED
  field1            CDATA     #IMPLIED
  field2            CDATA     #IMPLIED
```

```
  lastupdate        CDATA     #IMPLIED
  field3            CDATA     #IMPLIED
  onspecial         CDATA     #IMPLIED
  onauction         CDATA     #IMPLIED
  field4            CDATA     #IMPLIED
  field5            CDATA     #IMPLIED
  buyable           CDATA     #IMPLIED
>
```

以下は、ローダーからのサンプル・エラー・レポート文書です。

```
<?xml version="1.0"?>
<!DOCTYPE store-asset SYSTEM "store-all-error.dtd">
<store-asset>
   <error
      locus="Parser"
      id="SAXParseFatalError" >
      <message>
         Error The string "--" is not permitted within comments. : 155 : 18
      </message>
   </error>
   <calrule
      calcode_id="30"
      enddate="2100-01 10:20:30.000000"
      calmethod_id="-47"
      identifier="7"
      taxcgry_id="9"
      calmethod_id_qfy="-46"
      startdate="1900-01-01-00.00.00.000000"
      flags="1"
      combination="2"
      calrule_id="44"
      sequence="9.0E+1">
      <error
         locus="Writer"
         id="SQLException" >
         <message>
            A SQL Exception was received [IBM][CLI Driver][DB2/NT] SQL0530N
            The insert or update value of the FOREIGN KEY
            "JANTONY.CALRULE.F_CALRULE4" is not equal to any value of the
            parent key of the parent table.  SQLSTATE=23503
         </message>
      </error>
   </calrule>
   <catentry
      catentry_id="10118"
      member_id="-2001"
      partnumber="1254"
      mfpartnumber="sku-163"
      mfname="InFashion"
      markfordelete="0"
      buyable="1"
      field1="abc" >
      <error
         locus="Formatter"
         id="FormattingError" >
         <message>
            Error when formatting value for CATENTRY.FIELD1 : abc with error
            [class java.lang.NumberFormatException(abc)].
         </message>
      </error>
   </catentry>
</store-asset>
```

第 10 章 ローダー・パッケージのコマンドおよびスクリプトの構成

ローダー・パッケージを立ち上げて、そのコマンドを実行するには、以下の WebSphere Commerce ディレクトリーに提供されてい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 ネイティブ・ディレクトリー

スクリプトおよびコマンドは、以下のとおりです。

400

GENWCSDTD
DTD Generate コマンド

RESWCSID
ID Resolve コマンド

EXTWCSDTA
Extract コマンド

LODWCSDTA
Load コマンド

TRNWCSTXT
Text Transform コマンド

TRNWCSXML
XML Transform コマンド

NT 2000

dtdgen.cmd
DTD Generate コマンド

idresgen.cmd
ID Resolve コマンド

massextract.cmd
Extract 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

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

massextract.sh
Extract 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

第 3 部 Web エディターの使用

このセクションでは、Catalog Manager Web エディターを管理し、使用する方法について説明します。

Web エディターを使用すると、Web ブラウザーを使用してカタログ・データを作成、削除、および変更することができます。情報を表示および更新するためのデータ入力フォームは、Web エディターの主要なものです。最も単純な例では、データ入力フォームは 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のフォームを使用するか、または使用可能なフォームをカスタマイズするかを選択することができます。

注: Web エディターは、Internet Explorer 5 以降を使用します。

第 11 章 Web エディターのセットアップ

管理者は、WebSphere Commerce スキーマへの拡張およびカスタマイズを含め、WebSphere Commerce にデータを提供するように Web エディターを構成することができます。 Web エディターは、 WebSphere Commerce の特定のインスタンスに固有の Web 入力フォームのセットとしては設計されていません。 Web エディターは、異なる組織の個々の必要および役割をサポートするために柔軟にカスタマイズできるように設計されました。

データベース・ビューは、データベース内の 1 つまたは複数のテーブル上にある保管照会文です。 Catalog Manager のインストール中に、 WebSphere Commerce 製品の論理ビューを作成するためのサンプル・ファイル (db2、oracle、および os400) が、以下のディレクトリーに入れられます。

- NT *drive*:¥WebSphere¥CommerceServer¥schema
- 2000 *drive*:¥Program Files¥WebSphere¥CommerceServer¥schema
- AIX /usr/WebSphere/CommerceServer/schema
- Solaris Linux /opt/WebSphere/CommerceServer/schema
- 400 /QIBM/ProdData/WebCommerce/schema

このサンプル・ファイル wcs.view.sql には、複数のテーブルからの製品に関する情報を結合するために作成された製品ビューが含まれています。このサンプル・ファイルには、製品ビューの SQL データ定義が含まれます。管理者は、データベースの独自のビューの開発を計画するためにこのサンプル・ファイルを研究することができます。

デフォルトの XML フォーム記述ファイル (forms51_be.xml) には、 WebSphere Commerce データベース内のデータの追加、編集、および削除を容易に行うことができるように設計されたフォームが含まれています。このファイルのコピーは、以下のディレクトリーにあります。

- NT *drive*:¥WebSphere¥CommerceServer¥xml¥wcwebeditor¥xml
- 2000 *drive*:¥Program Files¥WebSphere¥CommerceServer¥xml¥wcwebeditor¥xml
- AIX /usr/WebSphere/CommerceServer/xml/wcwebeditor/xml
- Solaris Linux /opt/WebSphere/CommerceServer/xml/wcwebeditor/xml
- 400 /*instroot*/xml/wcwebeditor/xml

この構成ファイルは、現状のままで使用しても、または Web エディターをセットアップする管理者が変更および拡張してもかまいません。 XML フォーム記述ファイルをカスタマイズするには、このセクションの後の部分に記載されている指示を参照してください。

注: DTD ジェネレーターは、 Web エディターが使用するフォームを自動的に作成することができます。

Web エディターの構成

このセクションでは、Web エディターの構成方法について説明します。インストール・プロセスはこの構成を最初に処理しますが、管理者はこの情報を使用して、別のデータベースを使用するための Web エディターの再構成などを行うことができます。

webeditor.properties ファイルの編集

Web エディターは、webeditor.properties ファイルに設定されている、特定のアプリケーション・パラメーターを持っています。これは、以下のディレクトリーにあります。

- NT *drive*:¥WebSphere¥AppServer¥installedApps¥WC_Enterprise_App_demo.ear¥wcwebeditor.war¥WEB-INF¥classes¥webeditor.properties
- 2000 *drive*:¥Program Files¥WebSphere¥AppServer¥installedApps¥WC_Enterprise_App_demo.ear¥wcwebeditor.war¥WEB-INF¥classes¥webeditor.properties
- AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_demo.ear/wcwebeditor.war/WEB-INF/classes/webeditor.properties
- Solaris Linux /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_demo.ear/wcwebeditor.war/WEB-INF/classes/webeditor.properties
- 400 /QIBM/UserData/WEBASADV4/*was_instance_name*/installedApps/WC_Enterprise_App_*wcs_instance_name*.ear/wcwebeditor.war/WEN-INF/classes

webeditor.properties ファイルを編集することにより、管理者は、Web エディターに表示されているフォームを説明するために使用されるファイルの変更などを行うことができます。以下に、webeditor.properties ファイルの内容の例を示します。

```
# Properties file for WebEditor

(The following specifies where the customized process list and the Catalog Manager
utility configuration envelopes are located.)
# URI Location of Process and WCM Subsystem configuration envelopes
ProcessConfigFile=file:///D:/WebSphere/CommerceServer/xml/wcwebeditor/xml/weProcessList.xml

(The following specifies where the forms are defined.)
# Location of Forms file
FormsURL=file:///D:/WebSphere/CommerceServer/xml/wcwebeditor/xml/forms51_be.xml

(The following setting specifies where the XML-to-HTML style sheet is located)
# Location of XML to HTML StyleSheet
StyleSheetURI=file:///D:/WebSphere/CommerceServer/xml/wcwebeditor/xsl/webeditor.xsl

(The following setting specifies the location for temporary files)
# Location of Temporary Directory
#temp.dir=
```

```
# WebSphere datasource. This is used to build form drop-down lists and field
# default values. However, when publishing and searching, the Web Editor utilizes
# the WCM subsystems ID Resolver, Mass Loader, and Mass Extractor. Database access
# for these WCM subsystems must be configured separately.
# Name of WAS database source
dbsource=jdbc/WebSphere Commerce DB2 DataSource demo

# Name of WAS database. If specified, this takes preference over the database name
# value in the forms XML file.  This value will be utilized as a parameter when
# invoking the WCM subsystems such as the Mass Loader.
dbname=mall

(The following setting specifies the character set to use.  By default, it is
set to the single-byte character set.  For other countries, choose from the values
that are commented out.)
# Encoding Character Set
Encoding=ISO-8859-1

# CN
#Encoding=gb2312
# TW
#Encoding=Big5
# KR
#Encoding=EUC-KR
# JP
#Encoding=Shift_JIS

(The following setting is used to allow images to be previewed.  The hostname of
the server plus any leading directory information before the image information that
is stored in WebSphere Commerce should be specified here.  The WebSphere Commerce
catalog information is appended to this value to construct an image URL.)
# Specifies the base href location for images
# This should be set so that this info plus the info stored in WCS
# (e.g. /image/char.gif) combines to create a URL to an image

#imageRootURL=http://%HOSTDOMAIN
imageRootURL=http://localHost/webeditor

(The following setting is used to set a date format for the application.  If these
properties are not available (i.e., they are commented out), then the Java-locale
specific format will be used.  Below is a reference table for setting these values.)
# Use these properties to specify a date format if the Java locale-specific
# format is not desired

#dateFormat=yyyy-MM-dd
#dateTimeFormat=yyyy-MM-dd HH:mm:ss

# Symbol Meaning               Presentation             Example
# ------ -------               ------------             -------
# G      era designator        (Text)                   AD
# y      year                  (Number)                 1996
# M      month in year         (Text & Number)          July & 07
# d      day in month          (Number)                 10
# h      hour in am/pm         (1~12) (Number)          12
# H      hour in day           (0~23) (Number)          0
# m      minute in hour        (Number)                 30
# s      second in minute      (Number)                 55
# S      millisecond           (Number)                 978
# E      day in week           (Text)                   Tuesday
# D      day in year           (Number)                 189
# F      day of week in month  (Number)                 2 (2nd Wed in July)
# w      week in year          (Number)                 27
# W      week in month         (Number)                 2
# a      am/pm marker          (Text)                   PM
# k      hour in day           (1~24) (Number)          24
# K      hour in am/pm         (0~11) (Number)          0
# z      time zone             (Text)                   Pacific Standard Time
# '      escape for text       (Delimiter)
# ''     single quote          (Literal)                '
```

一時ファイルの場所の変更

一時ファイルの場所を変更するには、コメント・マークを除去し、 webeditor.properties ファイル内の temp.dir プロパティーに値を追加します。

また、java.io.tmpdir Java プロパティーは、一時ファイルが作成される場所を判別するために使用されます。

DTD ジェネレーターを使用した XML フォーム記述ファイルの作成

forms51_be.xml ファイルは、XML フォーム記述ファイルの一例です。これは、Web エディターによって使用されるフォームのセットを提供します。このファイルのコピーは、以下のディレクトリーにあります。

- NT *drive*:¥WebSphere¥CommerceServer¥xml¥wcwebeditor¥xml
- 2000 *drive*:¥Program Files¥WebSphere¥CommerceServer¥xml¥wcwebeditor¥xml
- AIX /usr/WebSphere/CommerceServer/xml/wcwebeditor/xml
- Solaris Linux /opt/WebSphere/CommerceServer/xml/wcwebeditor/xml
- 400 /*instroot*/xml/wcwebeditor/xml

以下のステップでは、システム管理者が DTD ジェネレーターを使用して新しい XML フォームを追加する方法について説明します。

注: 以下に説明されている手順を実行する前に、 Web エディター・ディレクトリーにある既存の forms51_be.xml の名前を変更してください。あるいは、以下の手順を行う際に新しい名前を持つ出力ファイルを作成し、その新しく作成されたファイルを使用するように Web エディターを再構成してください。これを行う方法については、前のセクションを参照してください。

XML フォームを作成するには、DTD Generate コマンドを実行してください。

以下のステップでは、XML フォームを作成する方法について説明します。

1. XML フォームで使用するテーブルの名前を含む "tables.txt" という名前の一時ファイルを作成します。

   以下の例のように、それぞれのテーブル名は 1 つの行に入力します。

   ```
   catentry
   catentdesc
   catentship
   inventory
   ```

2. DTD Generate コマンドがあるディレクトリーに tables.txt を保管します。(このコマンドのインストール場所については、83 ページの 『第 10 章 ローダー・パッケージのコマンドおよびスクリプトの構成』を参照してください。)
3. オペレーティング・システムのコマンド・プロンプトで、 DTD Generate コマンドがあるディレクトリーに変更します。
4. 以下のものを入力して、DTD Generate コマンドを実行します。

   NT 2000

```
dtdgen -infile tables.txt -outfile tables51.dtd
-dbname dbname -dbuser userid -dbpwd password
-xmlTableDesc tableFORMS.xml -schemaname schema -propfile filename
```

AIX Solaris Linux

```
./dtdgen.sh -infile tables.txt -outfile tables51.dtd
-dbname dbname -dbuser userid -dbpwd password
-xmlTableDesc tableFORMS.xml -schemaname schema -propfile filename
```

400

```
QWEBCOMM/GENWCSDTD DATABASE(database) SCHEMA(schema)
INSTROOT(instroot) PASSWD(password) OUTFILE(tables51.dtd)
INFILE(tables.txt) XMLTABDESC(tableFORMS.xml)
```

テーブル記述スイッチ (-xmlTableDesc または XMLTABDESC) を使用すると、DTD ジェネレーターは、DTD に加えて、テーブルの新規フォーム記述を作成します。

NT 2000 AIX Solaris Linux -propFile オプションは、ヘルプ・テキスト、デフォルト値、およびフィールド記述情報を保管できる外部プロパティー・ファイルの名前を指定します。

5. 前のセクションで説明されているように、新しく作成されたファイルを使用するように Web エディターを再構成します。
6. WebSphere 拡張管理コンソールで Web エディターを再始動します。これを行うには、以下のステップを実行します。
   a. **「WebSphere Administrative Domain (WebSphere 管理ドメイン)」**を展開します。
   b. **「Enterprise Applications」**を拡張します。
   c. **「WebSphere Commerce Enterprise Application - demo (WebSphere Commerce エンタープライズ・アプリケーション - デモ)」**を右マウス・ボタン・クリックし、**「停止」**を選択します。
   d. アプリケーションが停止したことを示すメッセージが表示されるまで待ちます。
   e. **「WebSphere Commerce Enterprise Application - demo (WebSphere Commerce エンタープライズ・アプリケーション - デモ)」**を右マウス・ボタン・クリックし、**「開始」**を選択します。
   f. エンタープライズ・アプリケーションが開始したことを示すメッセージが表示されるまで待ちます。
7. Web ブラウザーで新規フォームを表示するには、以下の URL をオープンします。

   ```
   https://host_name:8000/wcm/webeditor
   ```

   ここでの *host_name* は、ユーザーの WebSphere Application Server の完全修飾された HTTP ホスト名です。

   Web エディターは、すべてのテーブル名のリストを含むブラウザー・ウィンドウで表示します。

XML フォーム記述のカスタマイズ

このセクションでは、管理者が Web エディターによって表示されたフォームを拡張する方法について説明します。

XML フォーム記述は、XML フォーム記述ファイル自体か、または別個のプロパティー・ファイルのいずれかに属性および値を設定することによって、カスタマイズおよび拡張できます。

注: このプロパティー・ファイルの名前は、 formList タグの resourcePackage 属性の値として指定される必要があります。ファイル名がクラス・パス内のディレクトリーのサブディレクトリーに表示される場合、パッケージ (ドット) 仕様を使用する必要があります。

以下の表では、管理者が変更できる Web エディター形式のフィールド属性をリストします。

フィールド属性	説明
Currency	地域固有の数字分離記号 (米国の場合、3 桁ごとの区切りでコンマを入れる) で値が表示されます。
DbColumn	フィールド名をプロパティー・ファイル鍵にマップするために使用されます。 地域固有のプロパティー・ファイルが使用される場合、このエントリー内の値は、プロパティー・ファイル内に入力されている値と一致していなければなりません。 DTD ジェネレーターは、このエントリーにスキーマを追加します。

DefaultValue	ユーザーが新規フォームに書き込みを行う際に、データ・エントリーに表示される値を指定します。
	
	この属性は、DTD ジェネレーターによってデータベースのデフォルト値に設定されます。この属性は静的ストリングにすることができますが、単一行テーブルに対する SQL スカラー照会も含めることも可能です。外部データを検索するには、照会内でユーザー定義関数を使用することができます。たとえば、以下のようにします。
	
	`DefaultValue="SELECT CURRENT TIMETAMP FROM EXEC"`
	
	ここで、EXEC は、以下のように定義および入力されたユーザー定義関数です。
	
	`CREATE TABLE EXEC (A CHAR(1));`
	`INSERT INTO A VALUES('A');`

dynamicSqlSelectionList	ドロップダウン・メニューが各フォームごとに再作成されます。

フィールド属性	説明
FieldDescription	フォーム上の入力フィールドの隣に表示されている記述を提供します。 DTD ジェネレーターは、この属性の作成時にコメント (それがある場合) を使用します。列上にコメントがない場合には、デフォルトは列名になります。 この属性は、地域固有のプロパティー・ファイル内または XML フォーム記述ファイル内に設定できます。プロパティー・ファイル内に値が指定されている場合には、その値が優先されます。

フィールド属性	説明
FieldHelp	フィールドがフォームにフォーカスしているとき、フィールドの簡単なヘルプ説明がブラウザーの下部のメッセージ・バー上に表示されるようにします。 デフォルトでは、データの列タイプとともに、指定されたフィールドに関するデータを入力するための単純なメッセージが含まれています。 この属性は、地域固有のプロパティー・ファイル内または XML フォーム記述ファイル内に設定できます。プロパティー・ファイル内に値が指定されている場合には、その値が優先されます。
formatNumber	どんな場合でも数字を処理しないように Web エディターに通知するために使用されます。 formatNumber 属性を「false」に設定して、照会の抽出中を除き、入力された値をストリングとして扱います (値がストリングとして扱われる場合、引用符で囲まれません)。この属性のデフォルトは「true」です。
Hidden	フォーム上に値が表示されていませんが、HTML 隠しフィールドとして現在も使用可能であることを示します。
HideOnCreate	新規フォームが構成されるときに、フィールドが使用可能であることを示します。 showInCreateMode=″false″ と類似していますが、フィールド名を隠しタイプとして追加します。
Maxlength	データベース列の長さを指定します。 ユーザーが、データベース内に保管できる長さを超える長さの値を入力しないようにするために使用されます。

フィールド属性	説明
MinOccurs	フィールドが必須かどうかを示します。 「1」という値は必須を意味し、「0」という値はオプションを意味します。
Name	データベース列の名前を指定します。
readOnly readOnlyForCreate readOnlyForEdit	フィールドが編集可能である時、およびフィールドがユーザー読み取り専用である時を制御します。 readOnly=″true″ は、フィールドが常に読み取り専用モードであることを意味します。 readOnlyForCreate=″true″ は、新規フォームが構成されるときにフィールドが読み取り専用モードであることを意味します。 readOnlyForEdit=″true″ は、既存のフォームが編集されるときにフィールドが読み取り専用モードであることを意味します。
ShowColumnInList	「true」に設定するとき、フィールドがデータの複数レコード表示を構成する列の 1 つであることを指定します。 DTD ジェネレーターは、最初の 6 つの列を「true」に設定します。

フィールド属性	説明
showInCreateMode	作成されたフォームからフィールドを非表示にするために使用されます。 showInCreateMode 属性を「false」に設定し、表示されている作成されたフォーム上のフィールドを非表示にします。この属性のデフォルトは「true」です。
ShowInSearchMode	検索基準ページから特定の列を非表示にするために使用されます。 この属性が「false」に設定される場合、指定されたフィールドは検索フォームに表示されません。
SqlSelectionList	新規フォームを構成するときに、ユーザー用のドロップダウン・メニューを作成します。 照会は、1 列または 2 列の結果セットを戻さなければなりません。最初の列はユーザーが選択できるラベルのリストで、 2 番目の列はデータベース・テーブル内に保管するための実際の値のリストです。 1 列だけが使用される場合、表示されている値はテーブルに保管されます。この機能は、外部鍵関係を強制するために非常に役立ちます。

フィールド属性	説明
Type	データベース列のタイプ、およびデータ上で実行する必要のある妥当性検査の種類を示します。 たとえば、タイプが「integer」である場合、アプリケーションは、有効な整数だけがこのフィールドに入力されるようにします。
UniqueKey	フィールドがテーブルの 1 次鍵であることを示します。 この属性が「true」に設定される場合、アプリケーションは、この列に入力されたデータに対する固有制約を強制します。この検査は、アプリケーション内にロードされているレコードに対してのみ実行されます。データベース内のレコードには行われません。
ValidateInput	妥当性検査をオフにするために使用されます。 この属性を使用すると、ユーザーは、ID リゾルバーまたは XML トランスフォーマーによって処理される数値フィールドにテキストを入力することができます。

フォーム名の編集

XML フォーム記述内のフォーム・タグには、名前と displayName 属性の両方があります。

- 名前属性は、表示されるデータベース・テーブルまたはビューの名前に設定する必要があります。
- displayName 属性は、Web エディター・アプリケーション内で使いやすい名前を提供します。また、フォーム用の displayName 属性は、formList タグの resourcePackage 属性で指定されているプロパティー・ファイル内に設定することもできます。 (resourcePackage 属性は、複数言語サポート用に使用されます。)

  注: エントリーがプロパティー・ファイル内に表示されている場合には、 XML フォーム記述ファイルに入力されているエントリーよりも優先されます。

フィールド記述の変更

以下に、XML フォーム記述ファイル内のフィールド記述の例を示します。

```
<form name = "CATALOG.CATENTRY"
  .
  .
  <field name="MEMBER_ID"
  showInCreateMode="false"
    fieldDescription="MEMBER_ID"
    type="integer"
    maxlength="19"
    defaultValue=""
    .
    .
<field name="CATENTTYPE_ID"
    fieldDescription="CATENTTYPE_ID"
```

```
    type="string"
    maxlength="16"
    defaultValue=""
    .
    .
```

以下のようにフィールド記述を変更することもできます。

```
<form name = "CATALOG.CATENTRY"
  displayName="Product"
  .
  .
  <field name="MEMBER_ID"
  showInCreateMode="false"
  fieldDescription="Member Identifier"
    type="integer"
    maxlength="19"
    defaultValue=""
    .
    .
<field name="CATENTTYPE_ID"
    fieldDescription="Product Type"
    type="string"
    maxlength="16"
    defaultValue=""
    .
    .
```

ドロップダウン・メニューの追加

フォームの完成を容易にしたり、ユーザーが有効な選択項目のセットから選択することを制限するために、ドロップダウン・メニューのフォームの選択リストを Web エディターのフォームに追加することができます。

選択リストを最新表示するために、管理者は Web エディターを再始動することができます。フォームの記述フィールド属性には dynamicSqlSelectionList もあります。これは、管理者が「true」に設定すると、ドロップダウン・メニューは毎回再ロードします。

選択リストの照会には、1 列または 2 列のいずれかを戻す結果セットを含めることができます。結果セットが 2 列戻す場合、2 番目の列には格納された実際の値が含まれており、最初の列にはユーザー・ラベルが含まれています。

選択リストを作成するには、フィールド・タグの sqlSelectionList 属性内に SQL 照会を入力するか、または XML フォーム記述ファイル内に一覧表を作成します。以下の例に、両方の方法を示します。

```
<field name="MEMBER_ID"
  showInCreateMode="false"
  fieldDescription="Member Identifier"
  .
  .
  readOnly="false"
  sqlSelectionList="select orgentityname,orgentity.orgentity_id
   from member,orgentity where member.type='O'
   and member.member_id=orgentity.orgentity_id"
  .
  .
  fieldHelp=""
  .
```

```
  .
    </field>
      <field name="MARKFORDELETE"
        fieldDescription="Mark for Delete"
        type="NMTOKEN"
        sqlSelectionList=""
      >
      <datatype source="integer">
        <enumeration label="No" value="0"/>
        <enumeration label="Yes" value="1"/>
      </datatype>
    </field>
>
```

WebSphere Commerce データベース・テーブルには、true について 1、false について 0 を使用するフィールドがあります。ユーザーがもっと直観的に分かるようなフィールドを作成するには、これらのフィールドの一覧表を作成することができます。 Catalog Manager では、″NUMDESC″ という名前の一覧表テーブルを作成するための SQL スクリプトを提供しています。このスクリプトの名前は、″createEnum.sql″ です。以下の例に示されているように、管理者は ENUMDESC テーブルを使用して選択リストを作成するために sqlSelectionList 属性を編集することができます。

```
fieldDescription="On Special"
.
.
readOnly="false"
sqlSelectionList="select description,value from
  enumdesc where columnname='ALL' and type='YESNO'"
fieldHelp=""
.
.
```

フィールド・ヘルプの追加

フォーカスしている現行フィールド用のフィールド・ヘルプは、 Web ブラウザー・ウィンドウの下部に表示されます。 XML フォーム記述ファイル内のフィールド・タグの属性 fieldHelp またはプロパティー・ファイル内の fieldHelp キーは、この値を設定するために使用できます。

たとえば、プロパティー・ファイルには、以下のフィールド・ヘルプ仕様を含めることができます。

```
CATEGORY.MARKFORDELETE.defaultValue=No
CATEGORY.MARKFORDELETE.fieldDescription=Delete Entry
```

値が指定されていない場合、「Enter a value for *field_name* here *field_type*」というデフォルトのメッセージが作成されます。

注: フォーム情報を含むプロパティー・ファイルがある場合には、 XML フォーム記述ファイル内のエントリーよりも優先されます。

検索結果およびワーク・セッション・リストのカスタマイズ

検索結果ページ上に表示される列のセットは、カスタマイズできます。 XML フォーム記述ファイル内に、フィールド・タグの showColumnInList 属性があります。検

索結果間にフィールドを含めるには、この属性を「true」に設定します。そうしないと、検索結果ビュー内に列の 1 つとして表示されません。フィールドはフォームが表示されるときに表示されます。

weProcessList ファイルの編集

weProcessList ファイルを使用すると、管理者は、Web エディターのワーク・セッションが処理されるときに実行される Catalog Manager ユーティリティーをカスタマイズできます。

- DB2 NT 2000 AIX Solaris Linux

  weProcessList.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¥xml¥wcwebeditor¥xml
  - 2000 *drive*:¥Program Files¥WebSphere¥CommerceServer¥xml¥wcwebeditor¥xml
  - AIX /usr/WebSphere/CommerceServer/xml/wcwebeditor/xml
  - Solaris Linux /opt/WebSphere/CommerceServer/xml/wcwebeditor/xml

- Oracle NT 2000 AIX Solaris Linux

  weProcessListOracle.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¥xml¥wcwebeditor¥xml
  - 2000 *drive*:¥Program Files¥WebSphere¥CommerceServer¥xml¥wcwebeditor¥xml
  - AIX /usr/WebSphere/CommerceServer/xml/wcwebeditor/xml
  - Solaris Linux /opt/WebSphere/CommerceServer/xml/wcwebeditor/xml

- 400 weProcessListAS400.xml ファイルは、以下のディレクトリーにあります。

  - */instroot*/xml/wcwebeditor/xml

このファイルには、各種のユーティリティー用のエンベロープ・テンプレートが含まれています。また、このファイルには、管理者が実行するカスタム・アプリケーションへの参照を含めることもできます。

このファイル内で使用できるシステム変数のセットがあります。たとえば、システム変数 %-dbname% を使用すると、ローダーなどの特定のユーティリティーの呼び出しのために生成されるエンベロープ内にデータベース名が挿入されます。 XML フォーム記述ファイルには、追加、編集、または削除のためにどのシステム変数を呼び出す必要があるかを示す、これらのプロセスへの参照が含まれています。

以下に、 weProcessList.xml ファイルの例を示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<processSet>
  <!-- Do not change name of extract -->
  <process name="extract"
      subsystem="com.ibm.wca.MassExtract.extract.ExtractSubSystem">
      <envelope-input xmlns='saf_params.xsd'>
         <param name="-dbname" type="scalar" value="%-dbname%"/>
         <param name="-dbuser" type="scalar" value="%-dbuser%"/>
         <param name="-dbpwd" type="scalar" value="%-dbpwd%"/>
        <param name="-outfile" type="file" reside="local" value="%tempFilePath%"/>
```

```
      <param name="-filter" type="file" reside="local" value="%tempFileURI1%"/>
     </envelope-input>
</process>
<process name="transformer"
    subsystem="com.ibm.wca.XMLTransformer.XMLTransformerSubSystem">
    <envelope-input xmlns='saf_params.xsd'>
       <param name="-infile" type="file" reside="local" value="%tempFileURI%"/>
       <param name="-transform" type="file" reside="local"
         value="%webEditorDir%/xsl/ViewsToWCS51.XSL"/>
       <param name="-outfile" type="file" reside="local" value="%tempFilePath1%"/>
       <param name="-param" value="root=%-dbname%"/>
       <param name="-param" value="dtdname=%-dtdname%"/>
     </envelope-input>
</process>
<process name="transformerForDelete"
    subsystem="com.ibm.wca.XMLTransformer.XMLTransformerSubSystem">
    <envelope-input xmlns='saf_params.xsd'>
       <param name="-infile" type="file" reside="local" value="%tempFileURI%"/>
       <param name="-transform" type="file" reside="local"
         value="%webEditorDir%/xsl/ViewsToWCS51.XSL"/>
       <param name="-outfile" type="file" reside="local" value="%tempFilePath1%"/>
       <param name="-param" value="root=%-dbname%"/>
       <param name="-param" value="dtdname=%-dtdname%"/>
       <param name="-param" value="forDelete=true"/>
     </envelope-input>
</process>
<process name="resolver"
    subsystem="com.ibm.wca.IdResGen.IdResGenSubSystem">
    <envelope-input xmlns='saf_params.xsd'>
       <param name="-dbname" type="scalar" value="%-dbname%"/>
       <param name="-dbuser" type="scalar" value="%-dbuser%"/>
       <param name="-dbpwd" type="scalar" value="%-dbpwd%"/>
       <param name="-infile" type="file" reside="local"
         value="%previousOutFileAsURI%"/>
       <param name="-outfile" type="file" reside="local"
         value="%tempFilePath2%"/>
       <param name="-propfile" type="file" reside="local"
         value="propertyFiles.IdKeys"/>
       <param name="-method" type="scalar" value="mixed"/>
     </envelope-input>
</process>
<!-- Resolver as first process -->
<process name="resolverFirstProcess"
    subsystem="com.ibm.wca.IdResGen.IdResGenSubSystem">
    <envelope-input xmlns='saf_params.xsd'>
       <param name="-dbname" type="scalar" value="%-dbname%"/>
       <param name="-dbuser" type="scalar" value="%-dbuser%"/>
       <param name="-dbpwd" type="scalar" value="%-dbpwd%"/>
       <param name="-infile" type="file" reside="local" value="%tempFileURI%"/>
       <param name="-outfile" type="file" reside="local" value="%tempFilePath2%"/>
       <param name="-propfile" type="file" reside="local" value="propertyFiles.IdKeys"/>
       <param name="-method" type="scalar" value="mixed"/>
     </envelope-input>
</process>
<process name="loader"
    subsystem="com.ibm.wca.MassLoader.MassLoadSubSystem">
    <envelope-input xmlns='saf_params.xsd'>
       <param name="-dbname" type="scalar" value="%-dbname%"/>
       <param name="-dbuser" type="scalar" value="%-dbuser%"/>
       <param name="-dbpwd" type="scalar" value="%-dbpwd%"/>
       <param name="-commitcount" type="scalar" value="1000"/>
       <param name="-infile" type="file" reside="local" value="%previousOutFileAsURI%"/>
       <param name="-method" type="scalar" value="sqlimport"/>
       <param name="-noprimary" type="scalar" value="insert"/>
     </envelope-input>
</process>
<process name="loaderFirstProcess"
    subsystem="com.ibm.wca.MassLoader.MassLoadSubSystem">
    <envelope-input xmlns='saf_params.xsd'>
       <param name="-dbname" type="scalar" value="%-dbname%"/>
       <param name="-dbuser" type="scalar" value="%-dbuser%"/>
       <param name="-dbpwd" type="scalar" value="%-dbpwd%"/>
       <param name="-commitcount" type="scalar" value="1000"/>
       <param name="-infile" type="file" reside="local" value="%tempFileURI%"/>
```

```
        <param name="-method" type="scalar" value="sqlimport"/>
        <param name="-noprimary" type="scalar" value="insert"/>
      </envelope-input>
  </process>
  <process name="loaderForDelete"
      subsystem="com.ibm.wca.MassLoader.MassLoadSubSystem">
      <envelope-input xmlns='saf_params.xsd'>
        <param name="-dbname" type="scalar" value="%-dbname%"/>
        <param name="-dbuser" type="scalar" value="%-dbuser%"/>
        <param name="-dbpwd" type="scalar" value="%-dbpwd%"/>
        <param name="-commitcount" type="scalar" value="1000"/>
        <param name="-infile" type="file" reside="local" value="%previousOutFileAsURI%"/>
        <param name="-delete" type="scalar" value=""/>
      </envelope-input>
  </process>
  <process name="loaderForDeleteFirstProcess"
      subsystem="com.ibm.wca.MassLoader.MassLoadSubSystem">
      <envelope-input xmlns='saf_params.xsd'>
        <param name="-dbname" type="scalar" value="%-dbname%"/>
        <param name="-dbuser" type="scalar" value="%-dbuser%"/>
        <param name="-dbpwd" type="scalar" value="%-dbpwd%"/>
        <param name="-commitcount" type="scalar" value="1000"/>
        <param name="-infile" type="file" reside="local" value="%tempFileURI%"/>
        <param name="-delete" type="scalar" value=""/>
      </envelope-input>
  </process>
 <process name="saveToFile"
           cmd="cmd.exe /c c:¥temp¥theBatchFile.bat"
           args="-infile %tempFilePath% -dbname %-dbname%"
  />
</processSet>
```

注: ファイル参照には大文字小文字の区別があります。

以下の表には、アプリケーションが認識する有効な置換変数のリストが含まれています。

% 置換変数	戻り
%-dbname%	現行データベースの名前
%-dbuser%	データベース・ユーザー名
%-dtdname%	XML ファイル用の DTD の URI の場所
%-dbpwd%	データベース・ユーザー名のパスワード
%tempFilePath% %tempFilePath1% %tempFilePath2%	一時ファイルへの絶対パス これらは固有の一時ファイル名です。これらは、エンベロープ定義またはコマンド行の一時構文に入れられます。たとえば、%tempFilePath% がエンベロープ・テンプレート内の -infile パラメーターの値属性に入れられる場合、Web エディターは一時ファイルの場所にワーク・セッションのデータを書き込みます。
サブシステム上で %tempFileURI% %tempFileURI1% %tempFileURI2%	一時 URI 一時 URI は、%tempFilePath%ä%tempFilePath2% で表示されるものと同じファイルに対する URI です。これは追加のファイル・セットではありませんが、異なる構文で戻される、生成された同じ一時ファイルを検索するための方法です。
%previousOutFileAsURI%	前のタスク -outfile パラメーターの URI 表記を URI として提供します。
%webEditorDir%	Web エディターのインストールの場所

webeditor.xsl ファイルの編集

Web エディターは、デフォルトの XSL スタイルシートとして webeditor.xsl ファイルを使用します。これは、以下のディレクトリーにあります。

- NT *drive*:¥WebSphere¥CommerceServer¥xml¥wcwebeditor¥xsl
- 2000 *drive*:¥Program Files¥WebSphere¥CommerceServer¥xml¥wcwebeditor¥xsl
- AIX /usr/WebSphere/CommerceServer/xml/wcwebeditor/xsl
- Solaris Linux /opt/WebSphere/CommerceServer/xml/wcwebeditor/xsl
- 400 /*instroot*/xml/wcwebeditor/xsl

webeditor.xsl ファイルを編集することにより、管理者は Web エディターの出力形式を変更することができます。以下に、webeditor.xsl ファイルの内容のサンプルを示します。

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="html"/>
  <!-- Largest single line entry field value for fields larger than this a
    TEXTAREA is created -->
  <xsl:variable name="maxEntryFieldSize" select="80"/>
  <xsl:template match ="/">
    <xsl:apply-templates/>
 </xsl:template>
 <!-- Read Only field -->
 <xsl:template match="readOnly" name="readOnly">
    <xsl:element name="input">
      <xsl:attribute name="name"><xsl:value-of select="@name"/></xsl:attribute>
      <xsl:attribute name="type">hidden</xsl:attribute>
      <xsl:attribute name="value"><xsl:value-of select="@defaultValue"/></xsl:attribute>
        <!--
        <xsl:attribute name="onFocus">this.blur()</xsl:attribute>
        <xsl:attribute name="style">border-style:groove</xsl:attribute>
         -->
      </xsl:element>
      <table border="1" cellpadding="0" cellspacing="0" width="155" bgcolor="#C0C0C0">
         <tr>
           <td>
              <img border="0" src="/webeditor/image/space.gif" width="1"
                height="17"/><xsl:value-of select="@defaultValue"/>
           </td>
       </tr>
     </table>
 </xsl:template>
</xsl:stylesheet>
```

第 12 章 カタログの処理

以下の手順では、カタログ・データを含むデータベース・テーブル内でレコードを追加、変更、および削除するために、Web エディターを使用してカタログを処理する方法を説明します。

Web エディターを使用したテーブルへのレコードの追加

Web エディターを使用してテーブルにレコードを追加するには、以下のようにします。

1. Web ブラウザーから、以下の URL をオープンします。

   ```
   https://host_name:8000/wcm/webeditor
   ```

   ここでの *host_name* は、ユーザーの WebSphere Application Server の完全修飾された HTTP ホスト名です。

   Web エディターのデータベース・ログオン・ウィンドウが表示されます。
2. データベース・ユーザー名とパスワードを入力してから、**「ログオン」**をクリックします。

   Web エディターはブラウザー・ウィンドウで表示し、その左のメニュー・バーにはテーブル名のリストが含まれています。
3. 左のメニュー・バーの「追加」サブメニューにある適切なハイパーリンクをクリックします。

   該当するフォームが表示されます。
4. フォーム内に必要なすべてのデータを入力します。
5. **「Move to work session (ワーク・セッションへの移動)」**をクリックします。

   フォーム用のワーク・セッション結果が表示されます。これらの結果には、行われたすべての編集、追加、および削除のうち、この Web エディターのワーク・セッション中に消去または処理されなかったものが含まれます。
6. ワーク・セッションからレコード変更を除去する場合には、除去したいそれぞれのレコード変更の前にあるボックスにチェックマークを付け、**「Clear selected (選択したレコード変更の消去)」**をクリックします。
7. **「Process work session (ワーク・セッションの処理)」**をクリックして、選択したデータベースへの変更を実行依頼します。

   状況ページには、処理が正常に完了したことを示すメッセージが表示されます。
8. Web サイトにナビゲートし、適切なハイパーリンクをクリックし、変更が行われたことを確認します。

Web エディターを使用したテーブル内のレコードの変更

Web エディターを使用してテーブル内のレコードを変更するには、以下のようにします。

1. Web ブラウザーから、以下の URL をオープンします。

   ```
   https://host_name:8000/wcm/webeditor
   ```

   ここでの *host_name* は、ユーザーの WebSphere Application Server の完全修飾された HTTP ホスト名です。

   Web エディターのデータベース・ログオン・ウィンドウが表示されます。
2. データベース・ユーザー名とパスワードを入力してから、**「ログオン」**をクリックします。

   Web エディターはブラウザー・ウィンドウで表示し、その左のメニュー・バーにはテーブル名のリストが含まれています。
3. 左のメニュー・バーの「検索」サブメニューにある適切なハイパーリンクをクリックします。

   該当する検索ページが表示されます。
4. 以下のことを行って検索基準を指定します。
   a. 検索で指定するそれぞれの属性のそばにあるチェック・ボックスを選択します。
   b. 選択したそれぞれの属性用の適切なドロップダウン・メニューを使用して、検索で使用するロジックを選択します。
   c. 選択したそれぞれの属性の隣のフィールドに、検索で使用する値を入力するか、または選択します。
5. **「検索」**をクリックします。

   これにより、Web エディターへの検索基準が送信されます。

   状況ページには、検索基準を満たすレコード数が表示されます。
6. 以下のいずれかを行います。
   - **「Load data (データのロード)」**をクリックし、見つかったレコードのリストを表示します。

     Web エディターは、ユーザー照会から検索したレコードのリストを表示します。

     ステップ 6 に進みます。
   - **「New search (新規検索)」**をクリックし、検索ページに戻ります。

     ステップ 3 に戻ります。
7. 編集するレコードを選択します。

   該当するフォームが表示されます。
8. 編集するフィールドにスクロールダウンし、その内容を変更します。
9. **「Move to work session (ワーク・セッションへの移動)」**をクリックします。

フォーム用のワーク・セッション結果が表示されます。これらの結果には、行われたすべての編集、追加、および削除のうち、この Web エディターのワーク・セッション中に消去または処理されなかったものが含まれます。

10. ワーク・セッションからレコード変更を除去する場合には、除去したいそれぞれのレコード変更の前にあるボックスにチェックマークを付け、**「Clear selected (選択したレコード変更の消去)」**をクリックします。
11. **「Process work session (ワーク・セッションの処理)」**をクリックして、選択したデータベースへの変更を実行依頼します。

    状況ページには、処理が正常に完了したことを示すメッセージが表示されます。
12. Web サイトにナビゲートし、適切なハイパーリンクをクリックし、変更が行われたことを確認します。

Web エディターを使用したテーブルからのレコードの削除

Web エディターを使用してテーブルからレコードを削除するには、以下のようにします。

1. Web ブラウザーから、以下の URL をオープンします。

   ```
   https://host_name:8000/wcm/webeditor
   ```

   ここでの *host_name* は、ユーザーの WebSphere Application Server の完全修飾された HTTP ホスト名です。

   Web エディターのデータベース・ログオン・ウィンドウが表示されます。
2. データベース・ユーザー名とパスワードを入力してから、**「ログオン」**をクリックします。

   Web エディターはブラウザー・ウィンドウで表示し、その左のメニュー・バーにはテーブル名のリストが含まれています。
3. 左のメニュー・バーの「検索」サブメニューにある適切なハイパーリンクをクリックします。

   適切な検索ページが表示されます。
4. 以下のことを行って検索基準を指定します。
   a. 検索で指定するそれぞれの属性のそばにあるチェック・ボックスを選択します。
   b. 選択したそれぞれの属性用の適切なドロップダウン・メニューを使用して、検索で使用するロジックを選択します。
   c. 選択したそれぞれの属性の隣のフィールドに、検索で使用する値を入力するか、または選択します。
5. **「検索」**をクリックします。

   これにより、Web エディターへの検索基準が送信されます。

   状況ページには、検索基準を満たすレコード数が表示されます。
6. 以下のいずれかを行います。
   - **「Load data (データのロード)」**をクリックし、見つかったレコードのリストを表示します。

Web エディターは、ユーザー照会から検索したレコードのリストを表示します。

ステップ 6 に進みます。

- **「New search (新規検索)」**をクリックし、検索ページに戻ります。

  ステップ 3 に戻ります。

7. 削除するそれぞれのレコードのそばにあるチェック・ボックスを選択します。
8. **「Move to delete list (削除リストへの移動)」**をクリックします。
9. 左のメニュー・バーの「Work Session (ワーク・セッション)」サブメニューにある適切なハイパーリンクをクリックします。

   フォーム用のワーク・セッション結果が表示されます。これらの結果には、行われたすべての編集、追加、および削除のうち、この Web エディターのワーク・セッション中に消去または処理されなかったものが含まれます。
10. ワーク・セッションからレコード変更を除去する場合には、除去したいそれぞれのレコード変更の前にあるボックスにチェックマークを付け、**「Clear selected (選択したレコード変更の消去)」**をクリックします。
11. **「Process work session (ワーク・セッションの処理)」**をクリックして、データベースへの変更を実行依頼します。

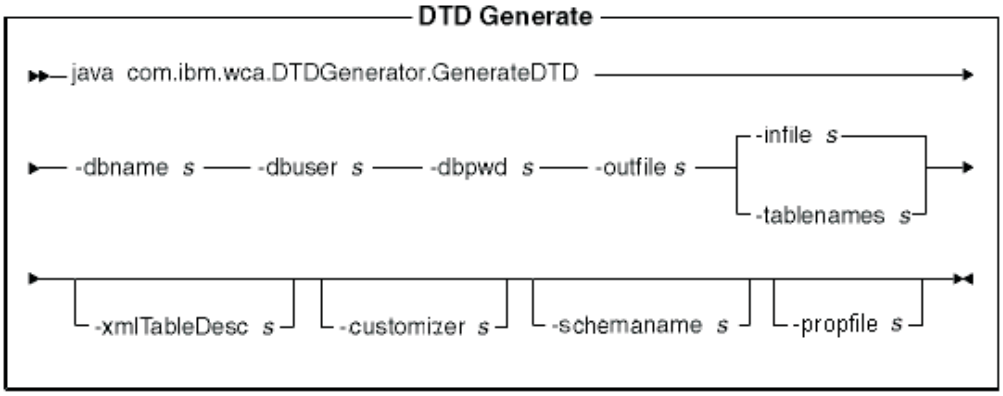
    状況ページには、処理が正常に完了したことを示すメッセージが表示されます。
12. Web サイトにナビゲートし、適切なハイパーリンクをクリックし、変更が行われたことを確認します。

第 4 部 コマンド解説

第 13 章 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ための DTD およびスキーマ・ファイルを作成します。

Windows、AIX、Linux、および Solaris システム用の DTD Generate コマンド



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

-infile それぞれの行にデータベース・テーブル名がある入力ファイルの名前

-tablenames
コンマで区切られたテーブルの名前

-xmlTableDesc
作成されるスキーマ・ファイルのファイル・パス

-customizer
使用されるカスタマイザー・プロパティー・ファイルの名前。
`DB2ConnectionCustomizer.properties` は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・ファイルは、次の例のように指定できます。

```
-customizer d:¥wc¥prop¥dtdgen.properties
```

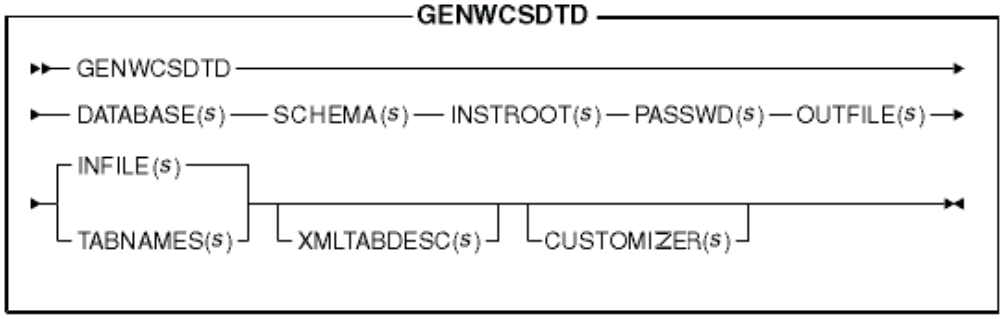
このファイルがクラスパス・システム環境変数で指定されたディレクトリーにある場合は、次の例のようにして同じファイルを指定できます。

```
-customizer dtdgen
```

-schemaname
ターゲット・データベース・スキーマの名前

-propFile
外部プロパティー・ファイルなどのプロパティーを含むファイル。ここには、Web エディターのフォームの記述用のヘルプ・テキスト、デフォルト値、およびフィールド記述情報を保管できます。

iSeries システム用の DTD Generate コマンド



パラメーター値:

DATABASE
リレーショナル・データベース・ディレクトリー内に表示されているターゲット・データベースの名前

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・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

INFILE
それぞれの行にデータベース・テーブル名がある入力ファイルの名前

TABNAMES
コンマで区切られたテーブルの名前

XMLTABDESC
作成されるスキーマ・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
使用されるカスタマイザー・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_GENWCSDTD_Customizer.properties です。カスタマイザー・プロパティー・ファイルは、次の例のように指定できます。

```
CUSTOMIZER(/wc/prop/dtdgen.properties)
```

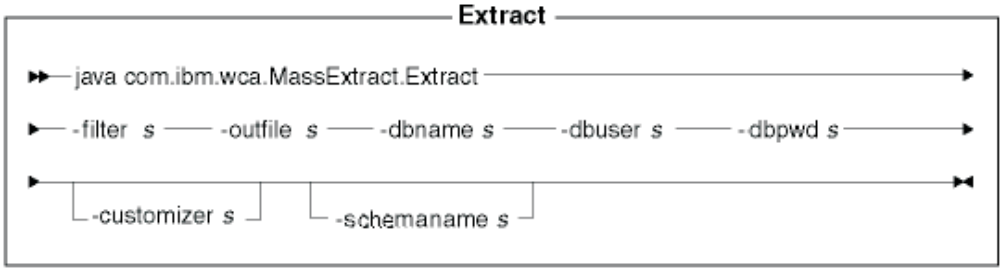
このファイルがクラスパス・システム環境変数で指定されたディレクトリーにある場合は、次の例のようにして同じファイルを指定できます。

```
CUSTOMIZER(dtdgen)
```

第 14 章 Extract コマンド

このコマンドは、XML ファイル形式でデータベースから選択したデータのサブセットを抽出します。

Windows、AIX、Linux、および Solaris システム用の Extract コマンド



パラメーター値:

-filter 抽出フィルター・ファイルの名前

-outfile
抽出されたデータが格納される出力 XML ファイルの名前

-dbname
データが抽出されるデータベースの名前

-dbuser
データが抽出されるデータベースのデータベース・ユーザー名

-dbpwd
データが抽出されるデータベースのユーザー名と関連したパスワード

-customizer
使用されるカスタマイザー・プロパティー・ファイルの名前。
`DB2ConnectionCustomizer.properties` は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・ファイルは、次の例のように指定できます。

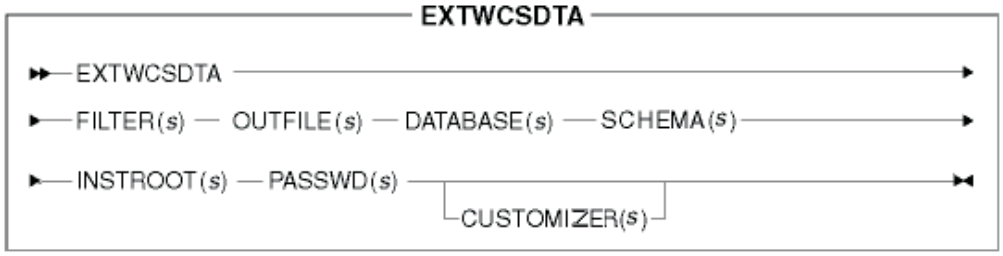
```
-customizer d:¥wc¥prop¥extract.properties
```

このファイルがクラスパス・システム環境変数で指定されたディレクトリーにある場合は、次の例のようにして同じファイルを指定できます。

```
-customizer extract
```

-schemaname
ターゲット・データベース・スキーマの名前

iSeries システム用の Extract コマンド



パラメーター値:

FILTER
抽出フィルター・ファイルの名前

OUTFILE
抽出されたデータが格納される出力 XML ファイルの名前

DATABASE
リレーショナル・データベース・ディレクトリー内に表示されている、データが抽出されるデータベースの名前

SCHEMA
データが抽出されるデータベース・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・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
使用されるカスタマイザー・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・ファイルは、次の例のように指定できます。

```
CUSTOMIZER(/wc/prop/extract.properties)
```

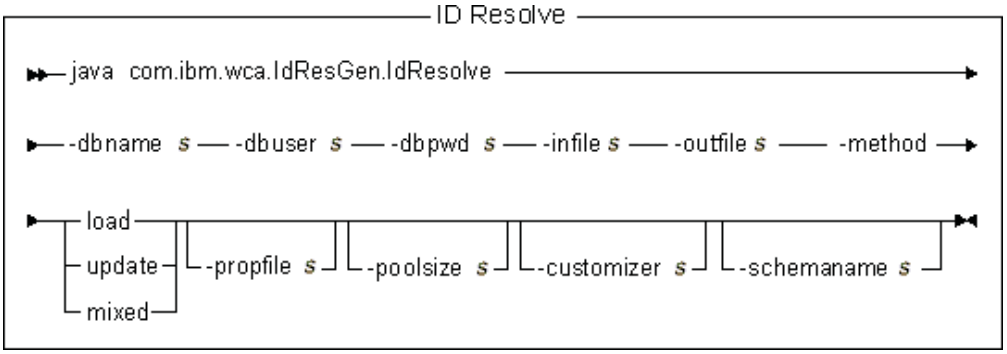
このファイルがクラスパス・システム環境変数で指定されたディレクトリーにある場合は、次の例のようにして同じファイルを指定できます。

```
CUSTOMIZER(extract)
```

第 15 章 ID Resolve コマンド

このコマンドは、データベースにロードする前に必要な XML エレメントの ID を生成します。

Windows、AIX、Linux、および Solaris システム用の ID Resolve コマンド



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・レコード記述子を含む入力 XML 文書の名前

-outfile
生成される出力 XML ファイルの名前。このファイルは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されるメソッド。このコマンドは、レコードがデータベース内に存在しないかのように (load)、またはすでに入力オブジェクトの ID があるかのように (update)、入力ファイルを処理することができます。一部のレコードがデータベース内に存在せず、一部のレコードがデータベース内に存在するときには、混合メソッドを使用します。デフォルトのメソッドがロードされます。

-propfile
name=value の対の形式の Java プロパティーを含むテキスト・ファイル。このファイルは、外部鍵 ID を検索するための検索列名、およびメイン・テーブル (CATEGORY および PRODUCT など) 照会用の選択述部を定義するために使用されます。 ID を含まない定義済みの固有索引を持つテーブルについては、このファイル内のエントリーを省略することができます。デフォ

ルトのファイルは、IdResolveKeys.properties です。このファイルには .properties という拡張子が付いていますが、値を指定するときには拡張子を使用しないでください。

-poolsize
予約される ID の数。デフォルトの数は 50 です。

-customizer
使用されるカスタマイザー・プロパティー・ファイルの名前。DB2ConnectionCustomizer.properties は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・ファイルは、以下のいずれかの例に示されている方法で指定できます。

```
-customizer d:¥wc¥prop¥idres.properties
```

```
-customizer d:¥wc¥prop¥idres
```

このファイルが現行ディレクトリーにある場合は、次の例のようにして同じファイルを指定できます。

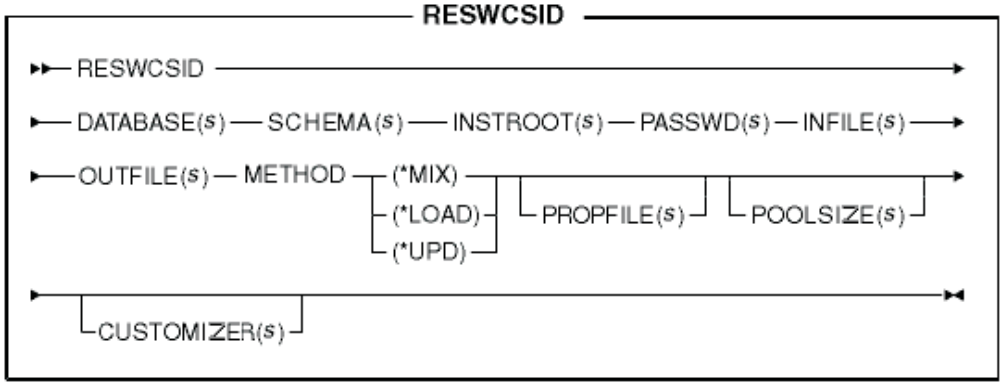
```
-customizer idres.properties
```

このファイルがクラスパス・システム環境変数で指定されたディレクトリーにある場合は、次の例の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

iSeries システム用の ID Resolve コマンド



パラメーター値:

DATABASE
リレーショナル・データベース・ディレクトリー内に表示されているターゲット・データベースの名前

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・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・レコード記述子を含む入力 XML 文書の名前

OUTFILE

生成される出力 XML ファイルの名前。このファイルは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されるメソッド。このコマンドは、レコードがデータベースに存在しないかのように (*LOAD)、またはすでに入力オブジェクト用の ID があるかのように (*UPD)、入力ファイルを処理することができます。一部のレコードがデータベース内に存在せず、一部のレコードがデータベース内に存在するときには、混合メソッド (*MIX) を使用します。

PROPFILE

name=value の対の形式の Java プロパティーを含むテキスト・ファイル。このファイルは、外部鍵 ID を検索するための検索列名、およびメイン・テーブル (CATEGORY および PRODUCT など) 照会用の選択述部を定義するために使用されます。 ID を含まない定義済みの固有索引を持つテーブルについては、このファイル内のエントリーを省略することができます。デフォルトのファイルは、IdResolveKeys.properties です。このファイルには .properties という拡張子が付いていますが、値を指定するときには拡張子を使用しないでください。

POOLSIZE

予約される ID の数。デフォルトの数は 50 です。

CUSTOMIZER

使用されるカスタマイザー・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_RESWCSID_Customizer.properties です。カスタマイザー・プロパティー・ファイルは、以下のいずれかの例に示されている方法で指定できます。

```
CUSTOMIZER(/wc/prop/idres.properties)
CUSTOMIZER(/wc/prop/idres)
```

このファイルが現行ディレクトリーにある場合は、次の例のようにして同じファイルを指定できます。

```
CUSTOMIZER(idres.properties)
```

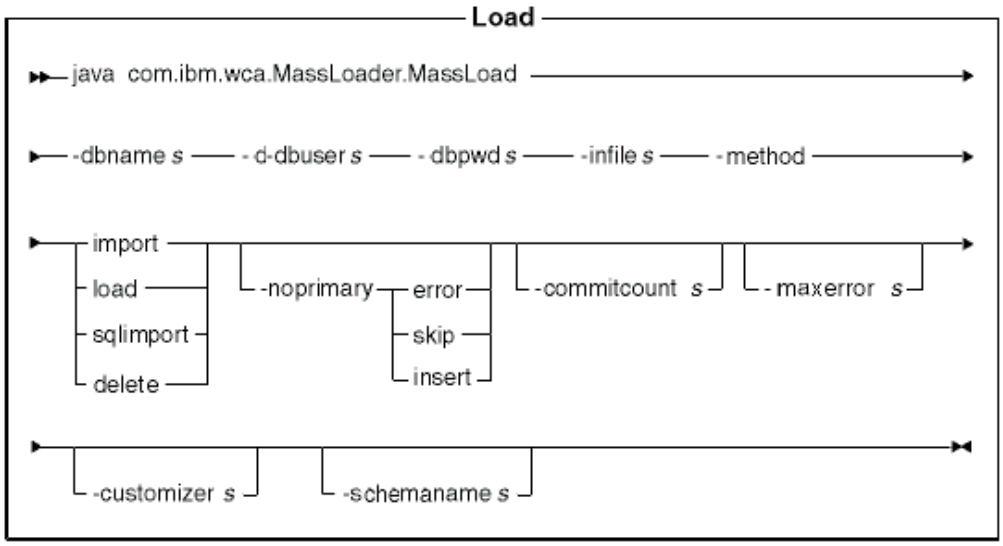
このファイルがクラスパス・システム環境変数で指定されたディレクトリーにある場合は、次の例のようにして同じファイルを指定できます。

```
CUSTOMIZER(idres)
```

第 16 章 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、Linux、および Solaris システム用の Load コマンド



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ときに使用するローダーの操作モード。 load メソッドは、データベース・ベンダーからの固有のローダーを使用します。 load メソッドは、Oracle データベースの場合はローカルとリモートの両方に使用できますが、 DB2 データベースの場合はローカルにのみ使用できます。データをリモートの DB2 データベースにロードするには、import オプションを使用してください。 import メソッドは、データベース・ベンダーから入手可能であれば、 import または update オプションを使用します。 import または update オプションを入手できない場合には、JDBC を使用する SQL ステートメントがデータベースの更新に使用されます。 SQL import (sqlimport) メソッドは、ローカルとリモートの両方のデータベースで使用できます。 delete メソッドは、データベースからデータを

削除します。商品アドバイザーの検索スペースの同期を使用している場合には、 sqlimport または delete メソッドのいずれかを使用する必要があります。

-noprimary

入力ファイル内でレコードの 1 次鍵が落しているときにローダーがとらなければならないアクション。 error オプションは、欠落している 1 次鍵をエラーとして報告し、終了する必要があることを示します。 skip オプションは、入力ファイル内の 1 次鍵がないレコードをスキップします。 insert オプションは、データの処理 (挿入または削除) を試行します。デフォルトのアクションは、エラーです。

-commitcount

操作の SQL update メソッドを使用するときに、データベース・コミットが発生する前に処理されるレコードの数。デフォルトの数は 1 です。

-maxerror

ローダーが操作の SQL update メソッドで終了するエラーの数

-customizer

使用されるカスタマイザー・プロパティー・ファイルの名前。
`MassLoadCustomizer.properties` はデフォルト・ファイルです。カスタマイザー・プロパティー・ファイルは、次の例のように指定できます。

```
-customizer d:¥wc¥prop¥ml.properties
```

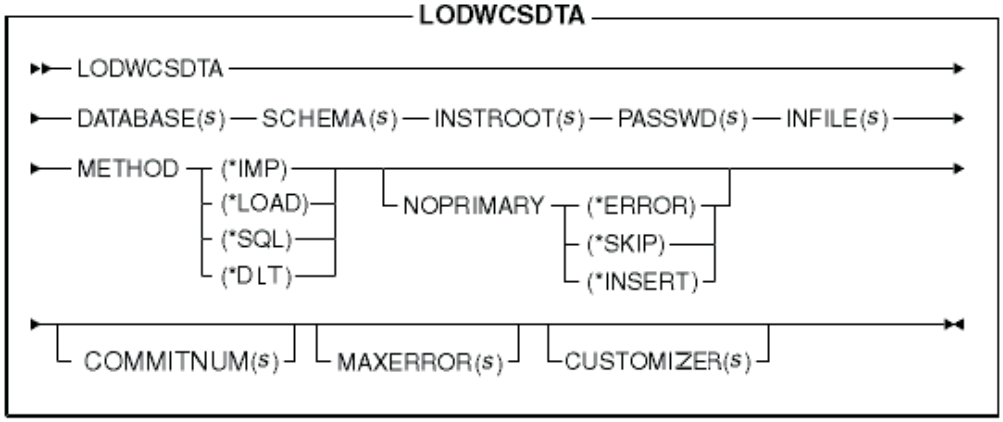
このファイルがクラスパス・システム環境変数で指定されたディレクトリーにある場合は、次の例のようにして同じファイルを指定できます。

```
-customizer ml
```

-schemaname

ターゲット・データベース・スキーマの名前

iSeries システム用の Load コマンド



パラメーター値:

DATABASE

リレーショナル・データベース・ディレクトリー内に表示されているターゲット・データベースの名前

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・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ときに使用するローダーの操作モード。load メソッド (*LOAD) は、データベース・ベンダーからの固有のローダーを使用します。 load メソッドは、Oracle データベースの場合はローカルとリモートの両方に使用できますが、 DB2 データベースの場合はローカルにのみ使用できます。データをリモートの DB2 データベースにロードするには、 import (*IMP) オプションを使用してください。 import (*IMP) メソッドは、データベース・ベンダーから入手可能であれば、 import または update オプションを使用します。 import または update オプションを入手できない場合には、 JDBC を使用する SQL ステートメントがデータベースの更新に使用されます。 SQL import (*SQL) メソッドは、ローカルおよびリモートの両方のデータベースで使用できます。 delete (*DLT) メソッドは、データベースからデータを削除します。商品アドバイザーの検索スペースの同期を使用している場合には、 SQL import または delete メソッドのいずれかを使用しなければなりません。

NOPRIMARY

入力ファイル内でレコードの 1 次鍵が欠落しているときにローダーがとらなければならないアクション。 error オプション (*ERROR) は、欠落している 1 次鍵をエラーとして報告し、終了する必要があることを示します。skip オプション (*SKIP) は、入力ファイル内の 1 次鍵がないレコードをスキップします。 insert オプション (*INSERT) はデータの処理 (挿入または削除) を試行します。デフォルトのアクションは、エラーです。

COMMITNUM

操作の SQL update メソッドを使用するときに、データベース・コミットが発生する前に処理されるレコードの数。デフォルトの数は 1 です。

MAXERROR

ローダーが操作の SQL update メソッドで終了するエラーの数

CUSTOMIZER

使用されるカスタマイザー・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_LODWCSDTA_Customizer.properties です。カスタマイザー・プロパティー・ファイルは、次の例のように指定できます。

```
CUSTOMIZER(/wc/prop/ml.properties)
```

このファイルがクラスパス・システム環境変数で指定されたディレクトリーにある場合は、次の例のようにして同じファイルを指定できます。

```
CUSTOMIZER(ml)
```

第 17 章 Text Transform コマンド

このコマンドは、データを、文字区切り可変長フォーマットと XML フォーマットの間でトランスフォームします。

Windows、AIX、Linux、および Solaris システム用の Text Transform コマンド

パラメーター値

以下の値が、コンマで区切られてパラメーター・ファイル (*parameter.txt*) に指定されます。

- 入力ファイル

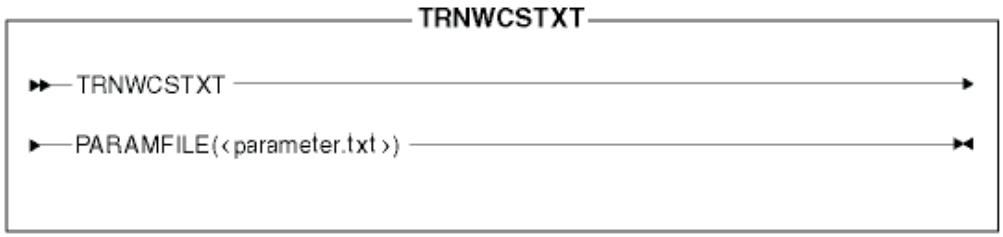
  トランスフォームされるファイルの名前
- スキーマ・ファイル

  トランスフォーメーションで使用される XML スキーマ・ファイルの名前
- 出力ファイル

  トランスフォームされたデータが格納される出力ファイルの名前
- transformation メソッド

  出力ファイルにデータを追加するために使用されるメソッド。新規ファイルを作成する場合は**「作成」**を、出力データを既存のデータ・ファイルに付加する場合は**「Append (付加)」**を指定します。

iSeries システム用の Text Transform コマンド



パラメーター値

以下の値が、コンマで区切られてパラメーター・ファイル (*parameter.txt*) に指定されます。

- 入力ファイル

  トランスフォームされるファイルの名前
- スキーマ・ファイル

  トランスフォーメーションで使用される XML スキーマ・ファイルの名前
- 出力ファイル

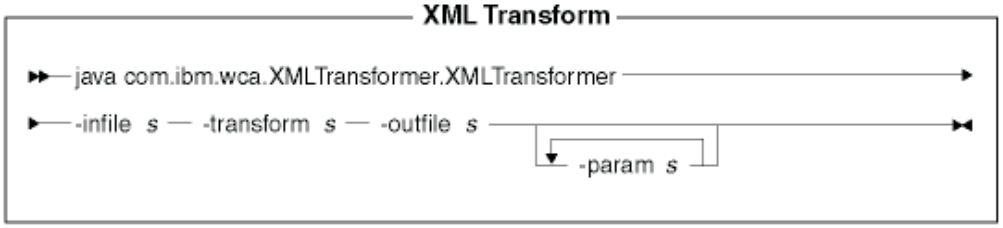
  トランスフォームされたデータが格納される出力ファイルの名前
- transformation メソッド

  出力ファイルにデータを追加するために使用されるメソッド。新規ファイルを作成する場合は**「作成」**を、出力データを既存のデータ・ファイルに付加する場合は**「Append (付加)」**を指定します。

第 18 章 XML Transform コマンド

このコマンドは、XML ファイルを代替 XML フォーマットに変換します。

Windows、AIX、Linux、および Solaris システム用の XML Transform コマンド



パラメーター値:

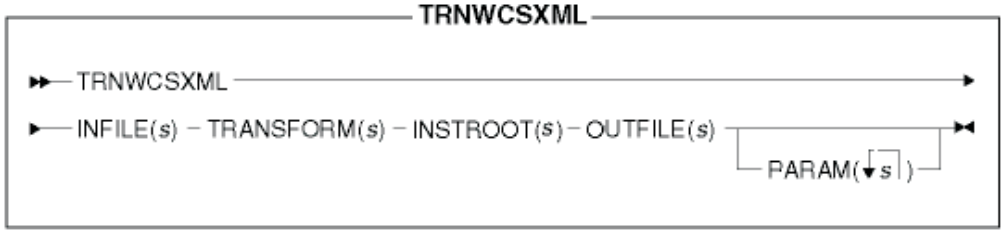
-infile トランスフォームされるファイルの名前

-transform
トランスフォーム XSL ルール・ファイルの名前

-outfile
トランスフォームされたデータが格納される出力 XML ファイルの名前

-param
XSL 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。*** このパラメーターは、複数の ″name=value″ の対を渡すために複数回指定できます。

iSeries システム用の XML Transform コマンド



パラメーター値:

INFILE
トランスフォームされるファイルの名前

TRANSFORM
トランスフォーム XSL ルール・ファイルの名前

INSTROOT
WebSphere Commerce インスタンス・ルート・パスの絶対パス名 (/QIBM/UserData/WebCommerce/instances/*instance_name* など)

OUTFILE
トランスフォームされたデータが格納される出力 XML ファイルの名前

PARAM
XSL 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。*** このストリングには、複数の ″name=value″ の対を渡すために複数の値を含めることができます。

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・プログラムのすべての部分、またはすべての派生した創作物には、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。

商標

以下は、IBM Corporation の商標です。

AIX
DB2
DB2 Universal Database
IBM
iSeries
OS/400
WebSphere

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

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

他の会社名、製品名およびサービス名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